Study on Adaptive Scheduling Method based on Anytime Algorithm for Real-Time Image Processing

Wyne Wyne Kywe

Study on Adaptive Scheduling Method based on Anytime Algorithm for Real-Time Image Processing

by

Wyne Wyne Kywe

M.I.Sc (University of Computer Studies, Yangon, Myanmar) 1998

DISSERTATION

Submitted in partial satisfaction of the requirements for the degree of

DOCTOR OF PHILOSOPHY

in

INFORMATION SCIENCE AND TECHNOLOGY

in the

GRADUATE SCHOOL OF INFORMATION SCIENCE AND TECHNOLOGY

DEPARTMENT

of the

AICHI PREFECTURAL UNIVERSITY

at AICHI, JAPAN

Committee in charge:

Professor Kazuhito Murakami, Chair

Professor Takashi Okuda

Professor He Lifeng

2013

Study on Adaptive Scheduling Method based on Anytime Algorithm for Real-Time Image Processing

Contents

List of Figures

List of Tables

Preface

In the real time environment, the digital contents such as image, video, and sound are being involved due to the technical improvement of information technology. In the low performance machine environment, it would be difficult to extract the useful information of the combination of tasks when the time is restricted.

In order to perform the tasks partially and optimize the overall processing result of combination of tasks in the restricted time, the objective of this dissertation concentrated on reconstruction of the digital image processing method, and construction of image processing library based on anytime algorithm from the software and algorithmic point of view.

Focusing on the above problems, this work is developed from the viewpoints of algorithmic and software engineering for the predictable performance of processing result from the available processing time rather than just fast processing by anytime algorithmic approach, according to its definition and features. Because anytime algorithm offers a trade-off between computation time and the quality of the processing result returned.

From the viewpoint of the quality of image processing and the processing time, this work proposes the adaptive static scheduling scheme based on anytime algorithm and imprecise computation for the overall image processing result under a condition that the processing time is restricted.

In order to optimize the overall image processing result by the proposed adaptive scheduling scheme, the detailed explanations of construction of adaptive scheduling scheme is described. Then, the proposed scheduling method is applied to the image processing task as an example. Thus, how to modify CIP (Conventional Image Processing) method such as filter type, gradient type, morphological type, and condition type to AAIP (Anytime Algorithmic Image Processing) method is explained. After that, how to realize the quality of each image processing task and how to optimize the overall processing result of image processing tasks are explained by using typical image processing tasks like noise reduction, edge detection, sharpening, thinning, and boundary detection.

The quality function of image processing result is expressed by using CDF (cumulative probability distribution function) in probability theory for each AAIP methods. The experimental results of the AAIP method expressed that the intermediate result are obtained at any time. Overall processing results are presented by using performance curves which shows that the quality of result becomes better when the processing time increases according to the definition and the properties of anytime algorithm.

The proposed AAIP method and the adaptive scheduling scheme are coded in C/C++ programming languages because of its wide acceptance in DIP (Digital Image Processing) community. The programming environment is Microsoft visual studio 2005 and the system environments are Windows XP, Intel (R) Pentium (R) M processor 1.30 GHz, 512 MB memory, and Sony DSC T7 camera.

Wyne Wyne Kywe
Graduate School of Information Science and Technology
Aichi Prefectural University
JAPAN

Acknowledgements

With great pleasure, I would like to express my thanks to all the people who helped me through this research period. At first, special thanks are due to Professor Kazuhito Murakami, my advisor, who gave me invaluable advice and instructions and helps me to implement this research work by questioning and providing research materials.

I am grateful to Professor Yasuyoshi Inagaki, former Dean of Graduate School of Information Science and Technology, for his kindness and suggestion to attend the Ph.D course.

I would like to express my gratitude to Professor Dr. Si Si, and Professor Dr. Win Win Htay, Head of Department of Engineering Mathematic, Yangon Technological University, Yangon, Myanmar who have helped me to extend the study of my life long fascinated subject.

I would like to express my thanks to Professor Tetsuo Ideguchi, Dean of Graduate School of Information Science and Technology, Aichi Prefectural University and Professors, Associate Professors and teachers who kindly treat and help me.

Also special thanks to Professor Takashi Okuda and Professor He Lifeng, the members of my dissertation committee for their kindness reading my dissertation and giving suggestions.

I always thanks to the past and present members of Murkami's laboratory who eager to help me by giving invaluable advice, and made discussions for research theme, by providing useful comments and giving suggestion, and listened patiently to my ideas.

Finally, I owe a lot to my beloved parents, who encouraged and helped me at every stage of my personal and academic life. I would like to express my gratitude to my respectful father, U Chit Kywe, who wants me to be an educated person, deeply gratitude to my mother, Daw Phwar Zin, who always care and encourage me to continue my study. I would like to express my greatest gratitude for all of my brothers and sisters who support and help me to continue my study. This work was supported in part by the Aichi Prefectural Government, and the NEC C&C foundation. This work would have been impossible without their collective support.

Chapter 1
Introduction

1.1 Background, Problems, and Related Works

In our daily life, we all have to do many tasks such as shopping, having lunch, doing household works and many more, and we have a time limit i.e., 24 hours in a day. Mostly, we are not considered on the time and the overall performance or quality of results of our tasks in an enough time. In case of limited/restricted time, for e.g., a professor has to do 3 tasks, i.e., task 1 (having breakfast) takes 20 min, task 2 (preparation documents for conference) takes 30 min, and task 3 (checking documents for trip) takes 15 min in the limited time 60 min. In such case, how do you finish these tasks and evaluate the overall performance (i.e., to finish all tasks in the limited time) of the quality of result? One can finish task 1 in 15 min, and then do other tasks in the required time or finish task 2 in 25 min and do other tasks or finish task 1 in 18 min, task 2 in 28 min and task 3 in 14 min and so on. So, the overall performance would not be 100%, but all tasks can be partially finished and the overall performance would be less than 100%.

In such example, there are several different ways to perform the tasks partially in the restricted time by dividing the task or by reducing the required time. Thus, it is necessary to consider the questions like "How to perform the task to be partially finished?" or "How to reduce the required time?"The basic idea is by dividing the task into sub-tasks, so that the required time of each sub-task is less than the total required time of task and this task can be partially performed.

Thus, this idea can be applied to the real-time system that is encountered in the time and quality trade-off problem. Some of the trade-off problem between the processing time and the quality of result can be solved by reducing the required time of a task or by partially performing a task. By performing the sub-tasks of each task with required time, the overall performance of the quality of result can be realized. How to schedule these tasks in the limited time in order to realize the overall performance of the quality of result is also important.

In the research environment, when the system is required to work in real-time, the problem of time constraint has been involved. The computation would become meaningless if the timely result is not acquired under resource constraint such as execution time, CPU power, cost, and memory even though the result is highly accurate. To give a priority or a weight to each task is a method to control CPU, but there is no guarantee of finishing the task within the deadline or tact time. Furthermore, the processing time involvement is the important aspect in the hard real-time system. The solutions to solve this kind of problem are by using hardware such as multi-processor, parallel processing and/or by using software such as algorithm, pipeline, cache or reduce the quality by size. From the algorithmic and/or software point of view, the guarantee of hard real-time deadline has been involved as a challenging problem.

Generally, there are many kinds of problems encountered in real-time system, for e.g., in the hard real-time system, most of the tasks are scheduled by the required time that is based on the given deadlines. If the tasks do not meet the hard real-time deadline are discarded. Conventional scheduling method by imprecise computation solves this problem by discarding the optional sub-tasks that could not meet their deadlines. J.W.S. Liu et al. reviewed and proposed the scheduling by their imprecise computation method that provides the scheduling flexibility by trading off the quality of result to meet the computation deadlines [1]. W.K Shih et al. proposed the fast algorithm to minimize the weighted errors for the scheduling under timing constraints by the approach of imprecise computation [2].

In addition, if a system has to perform many tasks or the combination of many tasks in a limited time, it would be difficult to achieve the overall performance of the quality of result in that limited time. If each task could obtain the result at every step in the way of computation time the overall performance or result of all tasks could be partially realized during the computation time. The allocation of the time requirement of each task could be distributed even though the quality is not perfect. Conventional scheduling methods solve this problem by using high performance CPUs or by using parallel processing and so on. There still remains a possibility to obtain more accurate result if CPU's computational power was left over.

The purpose of this work intended such a field that the cost or the generation of heat is limited as automobiles or embedded systems and RTIP (Real-Time Image Processing) system such as object tracking and image transmission systems. It is especially emphasized on the task assignment and scheduling problem from the viewpoint of algorithmic approach by the fixed priority pre-emptive scheduling for the short-term scheduler on uni-processor system by using the modified imprecise computation method and the concept of anytime algorithm.

1.2 Real-Time Image Processing (RTIP)

In real-time image/video processing, if we seriously consider the less processing time in a particular task, the quality of image processing would becomes sacrifice and conversely, if we adhere to the quality of image processing too much, the processing time might needed much time. The purpose of real-time image processing system involves with the improvement of the quality of video (image sequence) by enhancing the image in pre-processing like noise reduction.

Furthermore, many imaging applications are time critical and are computationally intensive. For example, in image transmission system, digital images require huge amounts of space for storage and large bandwidths before the transmission of image, so it is necessary to process these images in pre-processing such as filtering and enhancement. The outputs are required not only the perfect but also the timely imperfect results with deadline satisfaction. Moreover, the quality of image processing is usually evaluated by high extraction rate or low error rate. It is easy and clear to evaluate single image processing task, but if the system is composed of many tasks, it would becomes difficult to evaluate the combined processing result because of the results of image processing vary according to the combination of the methods and tasks. For instance, in image/video tracking or image transmission which is necessary to realize the intermediate result i.e., to achieve the better resolution with data transmissions and computations as low as possible, during the execution time.

There are many methods for image/video processing results reported in the real-time image processing from the different viewpoints like hardware platform such as FPGAs, DSP (Digital Signal Processors), GPU, Hybrid and PC based systems, and software platform such as pipeline, parallel processing and algorithm. It is dealing with the vast amount of data and the computation time and there is no concept of given computation time.

So, the problems of time constraint have been involved with the processing of image processing tasks for e.g., the processing of image sequence is a typical real-time system whose processing time is restricted in the duration of 1 frame, the required computation time of image sequence is 30 fps i.e., the processing of 30 image frames required 1 sec computation time.

Hasegawa, et al., solved the time-quality trade-off problem by selecting the best combination of system parameters in order to fit the result by sample figure [3]. Garvey et al., solved like this kind of problem with their design-to-time real-time scheduling approach to real-time problem solving demonstrated its feasibility in a complex real-time application by describing simulation experiments for Distributed Vehicle Monitoring Testbed (DVMT) application [4][5].

Although many approaches from the viewpoint of processing time are reported in the region of system planning in AI [6-10], on the other hand, there are a few reports in the scheduling of image processing [3]. There are many methods for real-time application are discussed as above, the approach to the digital image processing methods by anytime algorithm and imprecise computation to solve time-quality trade-off problem in real-time image processing is not reported yet.

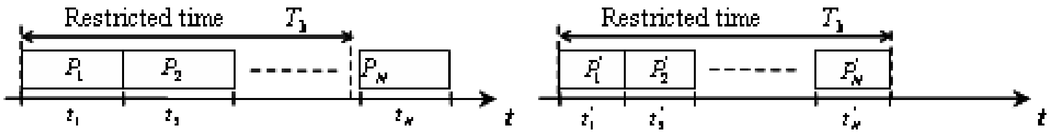
So, in this thesis, I introduce an approach to how to schedule the digital image processing tasks by the adaptive scheduling method by the concept of anytime algorithm in order to perform the task by its sub-task in pre-processing in RTIP under time restriction. By using the basic idea, in RTIP system, a task can be divided into small sub-tasks in order to perform the combination of tasks with some restrictions like processing time and memory usage by giving the partial overall processing result. Thus, it is necessary to analyze what kind of image processing tasks can be divided into sub-tasks. These explanations will be expressed in the later chapters.

1.3 Scheduling with Time Constraint

The most important fact to realize an adaptive scheduling is to modify a task which returns the result at anytime in the way of computation time. If each task could return the result at anytime, the overall processing result would be realized even though the quality of result is not perfect under the processing time constraint. Thus, the adaptive scheduling can be constructed by partially performed the task by its sub-task with allocated time requirement in order to provide the overall processing result under time constraint.

The basic idea of a task to be performed partially is at first a task is logically divided into mandatory sub-tasks in mandatory part and the optional sub-tasks in the optional part respectively. Then, these sub-tasks are performed by mandatory steps and the optional steps by giving the parameter such as tact time and the required sub-task numbers of the mandatory sub-tasks. Here, it is necessary to check the condition that the sum of the computation time of each sub-task is less than or equal to the current computation time. The overall processing result can be realized by checking the condition that the current overall processing result is greater than or equal to the pre-defined overall processing result which is evaluated by the minimum required number of mandatory sub-task. So, the number of discarded optional tasks can be minimized while realizing the sub-optimal overall processing result.

The second important idea is how to optimize the performance of the overall processing result. There are too many combinatorial procedures to satisfy the condition that the total time is less than the tact time. For example, if the system is composed of N tasks, say P_1, P_2, …, P_N and the required processing time t_1, t_2,…, t_N respectively. So, the total processing time $t_n = t_1 + t_2 + \ldots + t_N$ will be needed to perform all of N tasks. If the processing time is restricted into T_k, here, $T_k < t_n$, it is difficult to produce the intermediate result at current execution time t. Figure 1.1(a) shows pictorially the relation between the restricted time T_k and the required processing time t_1, t_2,…, t_N for the tasks P_1, P_2, …, P_N. As shown in this figure, when the time is restricted, at least one of the task e.g., P_N could not be performed. In order to perform all of the tasks in a restricted time, by modifying the tasks P_1, P_2, …, P_N to P_1', P_2', …, P_N' respectively, so that the total processing time $t_1'+t_2'+\ldots+t_N'$ becomes less than or equal to T_k as shown in Fig. 1.1(b).



(a) Conventional (b) Proposed

Fig. 1.1: Combination of tasks in the restricted time

Thus, in this thesis, how to modify the tasks to be partially performed and how to schedule these tasks in the restricted time will be mainly discussed using image processing tasks as an example. By using the concept of anytime algorithm and imprecise computation, some of the image processing algorithms are modified and the quality functions for the overall performance are evaluated.

Figure 1.2 shows the basic frame work of an image processing system which is composed of many tasks in low, mid and high level processing. There are some restriction like processing time and memory storage, and the tasks are performed by many steps. For this case, it is difficult to extract the useful information for e.g., extraction of the image attribute for the classification. In such kind of problem, it can be solved by using anytime algorithm which is suitable for solving the time-quality trade-off problem.

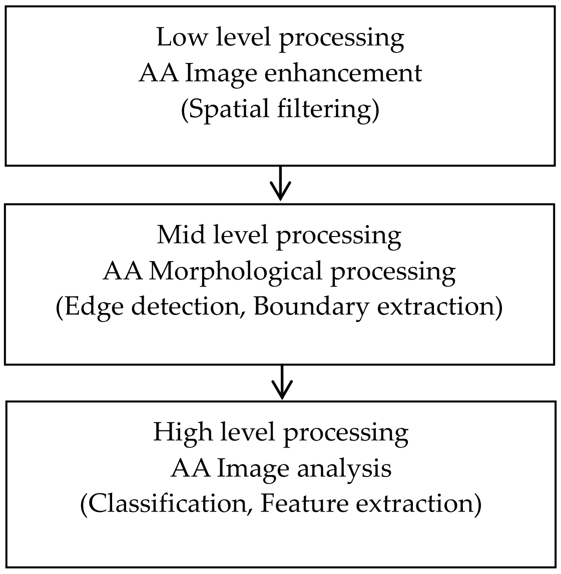


Fig. 1.2: The basic framework

1.4 Organization of Thesis

Anytime algorithm tool is used in order to implement the proposed method, so, the general concepts of anytime algorithm including its definition, properties with specific features of anytime algorithm, type of anytime algorithm (contract and interruptible), performance profile/curve (i.e., to measure the quality of a task), type of performance profile (i.e., which performance curve should be used to present the result), reduction theorem, compilation process, and monitoring are described in chapter 2. Moreover, the general explanations of imprecise computation such as its definition, the basic workload model, and the overhead time that is related to the proposed scheduling method are also described in chapter 2. Finally, the chapter summary is included.

In chapter 3, the general explanation about scheduling mechanism including time constraint and the type of task (i.e., dependent and independent), and the characteristic of scheduling are described. Then, the proposed scheduling model is expressed by its basic workload model, definition and terminology, scheduling algorithm, evaluation of the overall processing result, determination and analysis of the quality of overall processing result, and discussion. Finally, the chapter summary is explained.

In chapter 4, the briefly description of existing digital image processing methods and operations are presented. The conventional image processing (CIP) methods such as filter type, gradient type, morphological type, and condition type are converted to anytime algorithmic image processing (AAIP) methods by the concept of anytime algorithm. In particular, AAIP methods using filter type for low pass filtering by *Gaussian* and *Mean* filters, high pass filtering by basic hi-pass spatial filter, using gradient type for edge detection by *Prewitt* and *Sobel* filters, using morphological type for boundary extraction by structuring element, and using condition type for thinning by *Hilditch's* method respectively are expressed. The experimental results of AA noise reduction, AA edge detection, AA sharpening, AA thinning, and AA boundary extraction show that coarse to fine results can be obtained at intermediate processing time. After that, the quality of result of each task is presented by using performance curves according to the properties of anytime algorithm. Finally, the chapter summary is described.

In chapter 5, how to schedule the tasks by imprecise computation is explained by theoretical explanation of imprecise computation model, and definition and terminology. Then, the algorithms for task assignment and scheduling under time constraint are expressed by algorithms 1, 2, 3, and 4. The experiment is done by simulation and its experimental results for the overall processing result are expressed by figures and tables. After that, the adaptive scheduling of the combination of image processing tasks in the restricted time is experimented by using the sub-tasks number as input parameter for the dependent case. Then, its experimental result is presented by graphical and tabular forms. After that, the effectiveness of the proposed method is described. Finally, the chapter summary is expressed.

In chapter 6, how to evaluate the formulation of the quality function of each AAIP method is described. Then, the overall performance of image processing tasks including realization, evaluation i.e., modeling of the time-precision functions, and performance curves are explained for the dependent case. Then, realization of the overall quality in image processing and the scheduling is expressed by the experimental results and performance curves. Finally, the chapter summary is described.

In chapter 7, the relation of anytime algorithmic image processing and the biometric theme in order to apply to security system is explained. In particular, the biometric theme for the extraction of contact lens for the security purpose is realized. So, how to extract the contact lens by using thermo vision images as the pre-processing steps in the application of biometric is described. Then, the necessary condition of how to extract/detect the contact lenses for the hard and the soft lens including thermal property, transition of temperature, and procedures of contact lenses extraction are expressed. After that, the experimental results and discussion are presented. Finally, the chapter summary is explained. Finally, the contribution of this work, discussion, conclusion and the future works are explained in chapter 8.

References

[1] J.W.S. Liu, K.-J. Lin, W.K. Shin, A.C.S Yu "Algorithms for Scheduling Imprecise Computation" IEEE Trans. Computers ,Vol. 19,No.9, Sept. 1991,pp. 156-1,173.

[2] W.-K. Shih and J.W.-S. Liu, "Algorithms for Scheduling Imprecise Computations with Timing Constraints to Minimize Maximum Error," IEEE Trans. Computers., vol. 44, no.3, pp. 466-471, Mar 1995.

[3] Hasegawa, H. Kubota, and J. Toriwaki, "Automated construction of image processing procedures by sample figure presentation", Proc. of 8th Int'l Conf. on Pattern Recognition (ICPR1986), Vol. 1, pp.586-588, Oct. 1986.

[4] A. Garvey, V. Lesser. "Design-to-Time Real-Time Scheduling", IEEE Transactions on Systems, Man and Cybernetics, 1993 ftp://ftp.cs.umass.edu/pub/lesser/garvey-dtt-smc.ps

[5] Garvey, Alan and Victor Lesser. "Design-to-time Scheduling and Anytime Algorithms", SIGART Bulletin, 7 (2):16--19 (1996).

[6] T. Dean and M. Boddy, "An analysis of time-dependent planning", Proc. AAAI-88, pp.49-54, AAAI, 1988.

[7] S. Zilberstein and S. J. Russell. In S. Natarajan ed., Approximate reasoning using anytime algorithms, imprecise and approximate computation, Kluwer Academic Publishers, 1995.

[8] J. Grass and S. Zilberstein. In M. Pittarelli ed., Anytime algorithm development tools, SIGART Bulletin Special Issue on Anytime Algorithms and Deliberation Scheduling, 7(2):20-27, 1996.

[9] S. Zilberstein, Ph.D dissertation, "Operational rationality through compilation of anytime algorithm", Computer Science Division, University of California at Berkeley, 1993.

[10] S. Zilberstein, "Using anytime algorithms in intelligent systems", AI Magazine, vol. 17, no. 3, pp. 73-83, 1996.

Chapter 2
Anytime Algorithm and Imprecise Computation

2.1 Introduction

The term "*anytime algorithm*" was coined by *Thomas Dean* and *Mark Boddy* in the late 1980s in the context of their work on the time dependent planning [1]. There has been a considerable amount of work on designing and using algorithms that offer gradual improvement of quality of results, both before and after *Dean's* coining of the term "*anytime algorithm*". Early application of such algorithms can be found in medical diagnosis and mobile robot navigation.

The approach is known under a variety of names, including flexible computation, resource bounded computation, just-in time computing, imprecise computation, design-to-time scheduling, or decision-theoretic meta reasoning. All these methods attempt to find the best possible answer in a given operational constraints such as time, cost, and resource. It has been applied for the application in such are sensor interpretation and path planning (*Zilberstein* 1996; *Zilberstein* and *Russell* 1995) [2][3] and in feature selection which is used to improve the performance of learning algorithms by finding a minimal subset of relevant features by *Mark Last* et al. [4] and so on.

For the conversion of CIP method to AAIP method, anytime algorithm is used as a tool. Thus, this chapter provides the general concept of anytime algorithm including its definition, the properties that has specific features to apply to the image processing methods, type of anytime algorithm i.e., which type of anytime algorithm can be used for the implementation of AAIP tasks, type of performance curve i.e., what kind of performance curve can be used to represent the quality of image processing result for each AAIP method, monitoring and so on. It is to be noted that the following explanations are based on S. Zilberstein's papers [2][3][5][6][7].

2.2 Anytime Algorithm

2.2.1 Definition

Anytime algorithm is an algorithm whose quality of results improves gradually as computation time increases and it offers a trade-off between the resource consumption and output quality.

2.2.2 General Concepts

Anytime algorithm is suited for the problem which has the trade-off between the processing (computation) time and the accuracy since the accuracy will be improved according to the increase of computation time. The computation of anytime algorithm extends the traditional idea of computational procedures by allowing it to return many possible approximate answers to any given inputs. The speciality of anytime algorithm is the use of well-defined quality measures to monitor the progress in problem solving and allocate the computational resources effectively. In this approach, these concepts are applied to the CIP methods that have iterative tasks by using some of the following metrics which are applied in the construction of AAIP methods.

According to *S. Zilberstein's* paper [2] various metrics can be used to measure the quality of a result which is produced by an anytime algorithm. The following metrics have been proved useful in the construction of AAIP methods:

(1) **Certainty** – this metric reflects the degree of certainty that the result is correct. The degree of certainty can be expressed using probabilities, certainty factors, or any other approaches.

In this approach, the CDF (Cumulative probability Distribution Function) is used for the measuring of certainty for each task and it is described by performance curve.

(2) **Accuracy** – this metric reflects a measure of the difference between the approximate result and the exact answer. Many anytime algorithms can provide a *guarantee* of a bound on the error, where the bound is reduced over time.

It can be expressed by numerical methods such as forward differences, backward differences, and relative errors.

(3) **Specificity** – this metric reflects the level of detail of the result. In this case, anytime algorithm always produces the *correct* results, but the level of detail increases over time.

In order to perform the evaluation of approximated result at current step, it is necessary to use the previous result according to the concept of anytime algorithm. Also, the result of the current step is used for the calculation of next step and so on. Thus, the level of detail increases as processing time increases.

An anytime algorithm that does not use another anytime algorithm as a component is called an **elementary** anytime algorithm. A non-elementary anytime algorithm is also called a **compound** algorithm. In this part, the differences of the traditional programming and anytime algorithmic programming are explained. Existing programming techniques produce useful anytime algorithm. Although many traditional programming techniques can produce useful anytime algorithm, programming with anytime algorithm is different from the traditional programming. It can be efficiently constructed using standard programming techniques and it has been partly validated by a number of applications.

The principles of modularity can be applied to anytime algorithmic computation. In addition, large real time systems can be composed of anytime algorithm components. A real-time environment can be characterized by a time-dependent utility function. The problem of time allocation within such systems can be handled by a special compilation technique. Hence, depending on these facts, anytime algorithmic programming is based on the existing programming techniques.

2.3 Properties of Anytime Algorithm

This section explains the properties of anytime algorithm according to *S. Zilberstein's* paper [1]. In general, anytime algorithm has the properties that satisfy the following features.

(1) Measurable quality:

The quality of result can be defined exactly.

(2) Recognizable quality:

The quality of an approximated result can easy to determine at intermediate processing time.

(3) Monotonicity:

The quality of result is an increasing function of time and input quality.

(4) Consistency:

The quality of result is connected with computation time and input quality.

(5) Diminishing returns:

The solution's quality improves much larger than previous stages of computation and diminishes over time.

(6) Interruptibility:

The algorithm can be stopped at any time and given some answer.

(7) Preemptability:

The algorithm can be stopped and started again at any time with minimal overhead.

Modification of CIP methods to AAIP methods is based on these properties. So, the detailed explanations of how to modify these algorithms are described in the later chapters.

Figure 2.1 expresses pictorially the anytime algorithmic form of a task based on the definition of anytime algorithm and its properties.

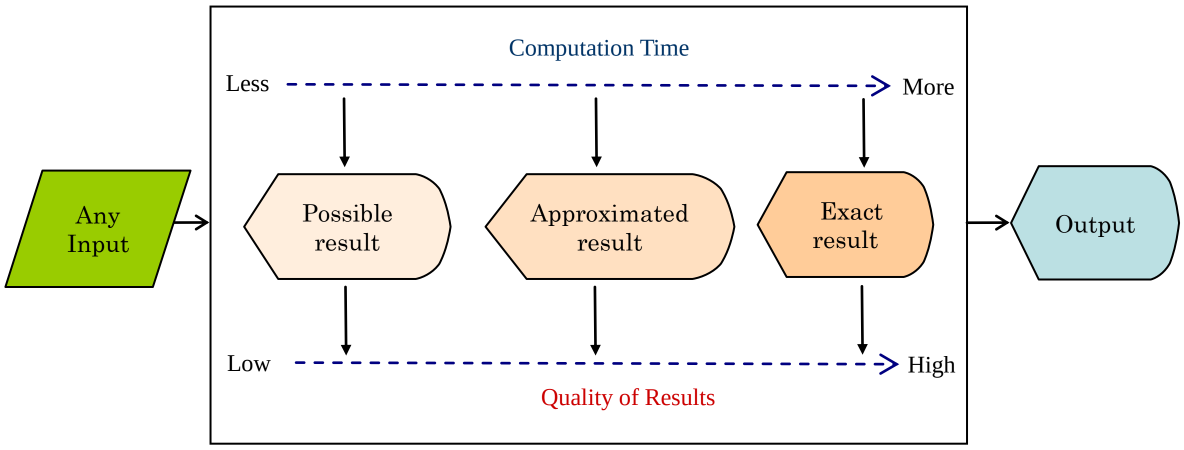


Fig. 2.1: Anytime algorithmic form of a task

2.4 Type of Anytime Algorithm

There are two types of anytime algorithm, i.e., contract and interruptible.

2.4.1 Contract

Anytime algorithm whose quality varies with time allocation although capable of producing results, it must be given a particular time allocation (total computation time) in advance. It requires the determination of the total run-time when activated. Although this algorithm can produce the results for any given time allocation, if it is interrupted before the expiration of the allocation, it may not obtained the require results.

2.4.2 Interruptible

Anytime algorithm produces the results of the "advertised quality" even when interrupted unexpectedly, whose total run-time is unknown in advance. It can be queried at anytime algorithm for a solution and output the result at any step.

2.4.3 Differences Between Interruptible and Contract Anytime Algorithms

Interruptible

- the total execution time is unknown in advance
- can be interrupted at any time to produce the results
- it is always contract algorithms
- more complicated to construct than contract algorithm and can be constructed based upon a contract algorithm
- flexible and widely apply

Contract

- total execution time must be known in advance
- cannot be interrupted at any time, if it interrupted between the execution time, it cannot obtain the useful results
- it is not interruptible algorithms
- it is easy to construct

2.5 Performance Profile

Performance profile, that is necessary to monitor the quality of result.

2.5.1 Definition

A performance profile of an anytime algorithm, *Q(t)*, denotes the expected output quality with execution time *t*, i.e., a mapping from time allocation to the expected output quality. It specifies the quality distribution for any given time allocation.

2.5.2 *Representation of Performance Profile*

Performance profiles can be represented either by a closed formula or as a table of discrete entries. The following descriptions are especially referred to *S. Zilberstein*'s paper [6].

(1) Closed Formula Representation

Since performance profiles are normally monotone functions of time, they can be approximated using a certain family of functions. Once the quality map is known, the performance information can be derived by various curve fitting techniques. For example, *Boddy* and *Dean* [1989] used the function: $Q(t) = 1 - e^{-\lambda t}$ to model the expected performance of their anytime algorithm planner and also *S. Zilberstein* and *S. Russell* used this function to define the performance profile for the calculation of maximal quality of composition of two tasks as an example [3]. *M. Last* et al. defined a criterion for measuring the quality of the algorithm results and study the algorithm performance profile on several benchmark datasets [4].

Performance distribution profiles can be approximated using a similar method by using a certain family of distributions. For instance, if the normal distribution is used, one can apply the same curve fitting techniques to approximate the mean and variance of the distribution as a function of time. In this approach, cumulative probability distribution function (CDF) is used for the representation of the performance profile of each modified AAIP procedures. The advantage of using a closed formula representation of performance profiles is that symbolic compilation can be performed once a parametric representation of each profile is given. The result of such compilation can be used each time members of that family are compiled. Figure 2.2 represents the graphical representation of performance profiles.

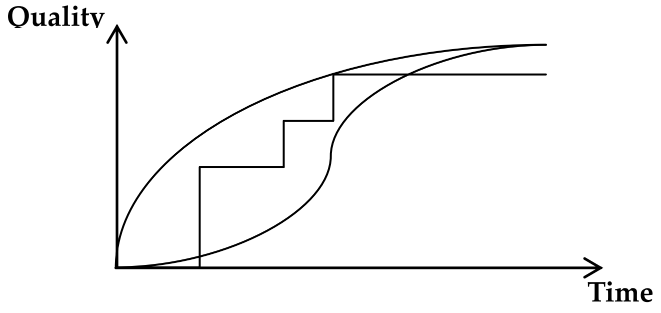


Fig. 2.2: Graphical representation of performance profiles

(2) Discrete Representation

Table 2.1: Tabular representation of performance profile for 4 tasks

Processing time (s)	Assignment of tasks by processing time and quality								Total Quality Q
	t_1	q_1	t_2	q_2	t_3	q_3	t_4	q_4	
0	0	0	0	0	0	0	0	0	0
0.01	0	0	0	0	0	0	0	0	0
0.02	0	0	0	0	0	0	0	0	0
|	|	|	|	|	|	|	|	|	|
|	|	|	|	|	|	|	|	|	|
0.19	0	0	0.078	0.498	0	0	0.11	1	0.405138
|	|	|	|	|	|	|	|	|	|
|	|	|	|	|	|	|	|	|	|
|	|	|	|	|	|	|	|	|	|
|	|	|	|	|	|	|	|	|	|
1.39	0.406	1	0.14	1	0.734	1	0.11	1	1

Table 2.1 shows the performance distribution profile of the proposed scheduling algorithm for 4 tasks as an example. The discrete representation of performance profiles is based on a table that specifies the discrete probability distribution of quality for certain possible time allocations. For this purpose, the complete range of qualities has to be divided into discrete qualities $q_1, \ldots, q_4$ depends on each task, the range is [0, 1]. The entry $q_1, \ldots, q_4$ in the table represents the discrete probability that has corresponding time allocation $t_1, \ldots, t_4$ for the tasks 1, 2, 3, and, 4 with the actual total output quality Q that would be in the range [0, 1]. The size of the table is a system parameter that controls the accuracy of performance information.

2.5.3 Type of Performance Profile

An algorithm may have several performance profiles, each representing its performance when operating in a different machine environment i.e., depending on CPU performance. There are many types of performance profile and the following are the few of ones. Performance Distribution Profile (PDP) is used for the measuring of the quality of result for the AAIP procedure and conditional performance profile (CPP) is used to the evaluation of quality function for the optimization of total performance of the scheduling method.

(1) *performance distribution profile (PDP)*
(2) *expected performance profile (EPP)*
(3) *performance interval profile (PIP)*
(4) *conditional performance profile (CPP))*
and so on.

2.5.4 Finding the Performance Profile of an Algorithm

When an anytime algorithm is implemented on a certain machine environment, "How to determine its performance profile?" must be considered. In some cases, the performance profile can be calculated by performing a *structural analysis* of the algorithm. For example, an iterative algorithm, e.g., *Newton's* method, the error in the result is bounded by a function that depends on the number of iterations. In such cases, the performance profile can be calculated once the execution time of a single iteration is determined. In general, however, such structural analysis of the code is hard because of the improvement rate of quality in each iteration and its execution time may be unpredictable. Moreover, it is depended on the machine environment. To overcome this difficulty, a general *simulation* method can be used. In this approach, *Monte Carlo* simulation method is used.

Given an anytime algorithm $\boldsymbol{A}$, let $q_A\ (x,\ t)$ be the quality of results produced by $\boldsymbol{A}$ with input x and computation time t; let $q_A(t)$ be the expected quality of results with computation time t; and let $\boldsymbol{p_{A,t}(q)}$ be the probability (density function in the continuous case) that $\boldsymbol{A}$ with computation time t produces the results of quality $\boldsymbol{q}$.

Conditional Performance Profile

Conditional performance profile (CPP) that includes a mapping from input quality and execution time to a probability distribution of output quality. A conditional performance profile captures the dependency of output quality on time allocation as well as on input quality. It characterized the performance of each elementary anytime algorithm as function of execution time and input quality.

A CPP of an anytime algorithm, $Pr(q_{out}\ (q_{in},\ t))$, denotes the probability of getting a solution of quality when the algorithm is activated with input of quality q_{out} and execution time t.

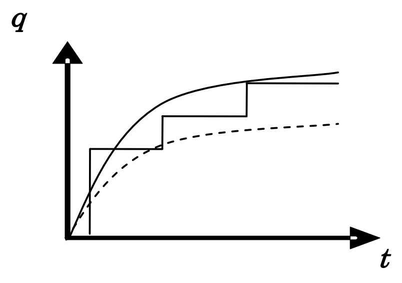


Fig. 2.3: Graphical representation of a CPP

Figure 2.3 shows a graphical representation of typical CPP. Each curve represents the expected output quality as a function of time for a *given* input quality.

How to construct (create) a conditional performance profile?
Example procedure for constructing a CPP

Step-1: Estimate the average completion time of the algorithms.
Step-2: The system automatically generates a large number of problem instances and records the quality improvement of the algorithm as a function of time for each problem.
Step-3: The statistics data, called the quality map of the algorithm, is used to construct the probability distribution of output quality for a given time.
Step-4: If input quality is a variable, the system repeats the step 1 – 3 for a set of different initial input qualities.

Sample* anytime algorithm *(A description of the model of execution)

The following example algorithm represents the typical implementation of an interruptible anytime algorithm for the construction of pyramidal images.

```
Input(x, y)
Result ← INITIALIZATION-STEP (Input(x, y))
REGISTER-RESULT (Result)
x ← 0; y ← 0;
while (x < h)
{
        while (y < w)
        {
                Output(x, y) ← Input(x, y);
                y ← y + 2;
        }
        SIGNAL (TERMINATION)
        HALT
}
w ← w/2;
h ← h/2;
```

2.6 Reduction Theorem

Reduction theorem is a theorem which allows for the construction of contract anytime algorithms as an *intermediate* step, before the system is made interruptible. (S. Zilberstein, 1993)

*For any contract algorithm an interruptible algorithm **B** can be constructed such that for any particular input $q_B(4t) \geq q_A(t)$.*

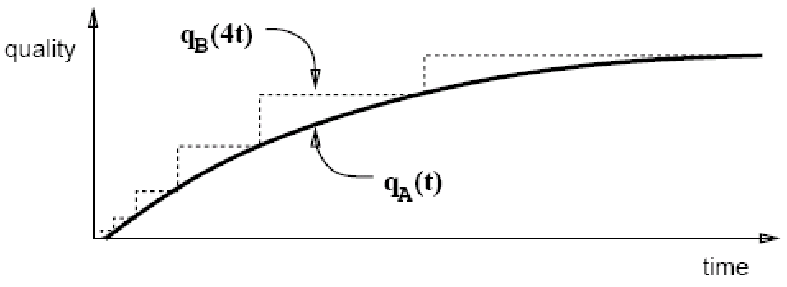


Fig. 2.4: Performance profiles of interruptible and contract algorithms

2.7 Compilation Process

Given a system composed of anytime algorithm i.e., compound algorithms, the compilation process is designed to: (a) determine the optimal performance profile of the complete system and (b) insert into the composite module the necessary code to achieve that performance. The precise definition and solution of the problem depend on the following factors:

1. **Composite program structure** – what type of programming operators are used to compose anytime algorithms?
2. **Type of performance profiles** – what kind of performance profiles are used to characterize elementary anytime algorithms?
3. **Type of anytime algorithm** – what types of elementary anytime algorithms are used as input? What type of anytime algorithm should the resulting

system be?

4. **Type of monitoring** – what type of run-time monitoring is used to activate and interrupt the execution of the elementary components?
5. **Quality of intermediate results** – what access does the monitoring component have to intermediate results? Is the actual quality of an intermediate result known to the monitor?

Depending on these factors, the different types of compilation and monitoring strategies are necessary to be considered.

2.8 Runtime Monitoring

Monitoring of execution is an important component of an anytime algorithmic system. It plays a vital role in anytime algorithm computation. The purpose of run-time monitoring is to reduce the effect of uncertainty on the performance of the system. Uncertainty regarding the performance of the system is characterized by its CPP.

Two monitoring strategies have been developed for the system. The first is a fixed-contract monitoring scheme and the second is an adaptive monitoring scheme. The appropriate type of monitoring will typically depend on the source of uncertainty and the degree of uncertainty.

2.9 Imprecise Computation

2.9.1 What Is Imprecise Computation

Imprecise computation is the particularization of anytime algorithm, and its concept is logically dividing the task into mandatory and optional part. Then, mandatory task is performed by mandatory step and the optional part is performed by the left steps [8-10].

2.9.2 Imprecise Computation Model

In the conventional imprecise computation, there are many methods like milestone, sieve, and multiple versions. In the conventional method, the task is logically divided into mandatory and optional sub-tasks as shown in Fig. 2.5. Optional sub-task is used to refine the result of mandatory sub-task. Thus, it has only 2 results.

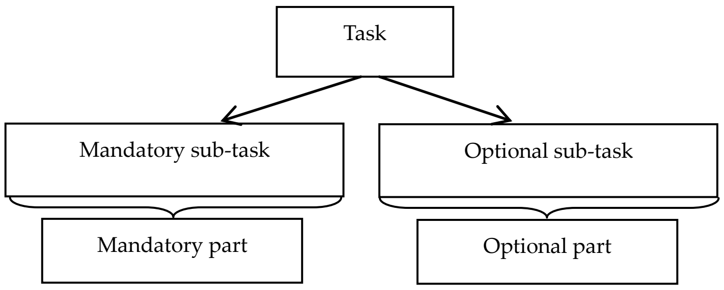


Fig. 2.5: Ways in the conventional method

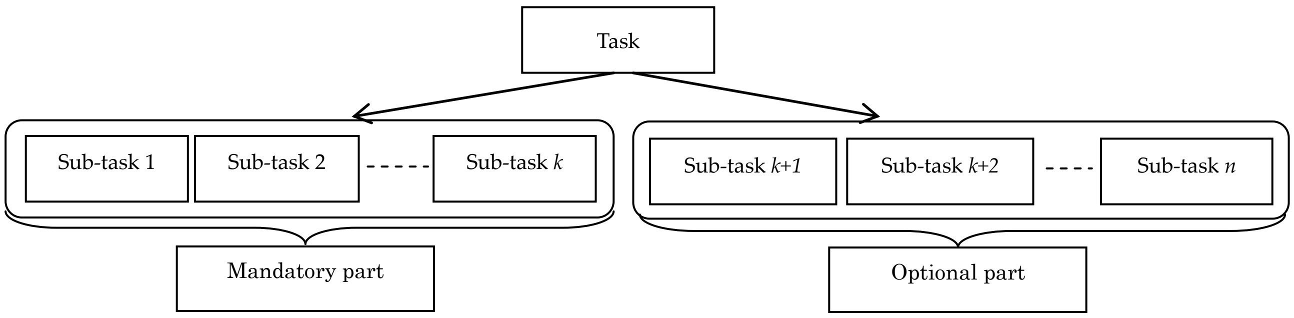


Fig. 2.6: Ways in the proposed method

In this proposed method, the task is logically divided into n sub-tasks and these sub-tasks are performed by many steps as shown in Fig. 2.6. Thus, it has more than 2 different results.

2.9.3 Overhead Time

Suppose that a task is divided into $\boldsymbol{n}$ sub-tasks performed by $\boldsymbol{n}$ steps, and sub-task 1 is performed by step 1, sub-task 2 is performed by step 1 and 2, and so on. For each sub-task the previous result is used for the calculation of next step. So, there is a few amount of processing time for the next step. The approximated processing time $t_1, t_2, \ldots, t_k$ of sub-task 1, 2, …, k are

$$t_1 = t_0 + \triangle t_1$$
$$t_2 = t_1 + \triangle t_2$$
$$\vdots$$
$$t_k = t_{k-1} + \triangle t_k$$

Where

t_0 = initial processing time

t_i = the required processing time at i^{th} step or ready time for execution,

$i = 1, 2, \ldots, k$

$\triangle t_i$ = a few amount of additional processing time ≥ 0, $i = 1, 2, \ldots, k$

In general,

$t_n = t_{n-1} + \triangle t_n$

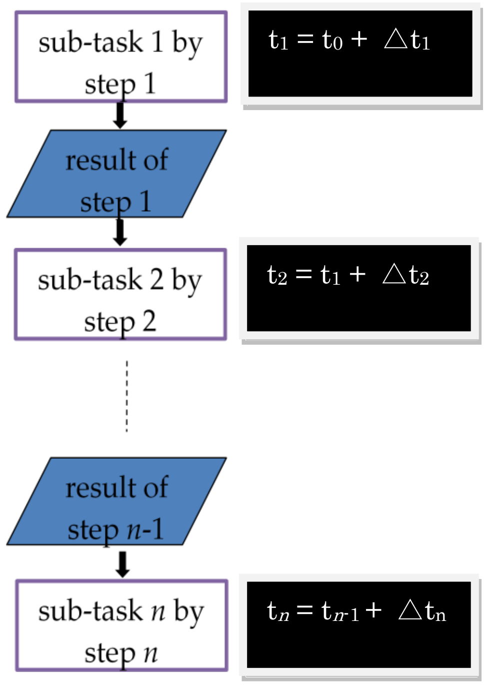


Fig. 2.7: Process flow of n sub-tasks of a task

Thus, it might be said that there is a few amount of overhead time between the processing steps. This proposed method is modified from the conventional imprecise computation technique and it can be applied to the task assignment and scheduling. For each sub-task, it has corresponding steps to be performed the operation, the processing stage for the sub-task k is expressed as shown in Fig. 2.7.

2.10 Summary

This chapter explained the general concept of anytime algorithm which is applied to the construction of anytime algorithmic image processing (AAIP) procedures including definition, properties, kinds of anytime algorithm namely interruptible and contract and their differences. Then, performance profiles and its representation by graphical and tabular and sample anytime algorithm for the construction of pyramidal images as an example. After that, compilation process and rum-time monitoring are explained. The type of performance profiles i.e., PDP, EPP, PIP, and CPP, and how to construct them, in particular, CPP construction is described. Finally, the general concept of imprecise computation including its definition, the basic workload model and overhead time are explained.

References

[1] T. Dean and M. Boddy, "An analysis of time-dependent planning", Proc. AAAI-88, pp.49-54, AAAI, 1988.

[2] S. Zilberstein, "Using Anytime Algorithms in Intelligent Systems", AI Magazine, vol. 17, no. 3, pp. 73-83, (1996).

[3] S. Zilberstein and S. J. Russell. In S. Natarajan (Ed.), "Approximate Reasoning Using Anytime Algorithms, Imprecise and Approximate Computation", Kluwer Academic Publishers, (1995).

[4] M. Last, A. Kandel, O. Maimon, E. Eberbach, "Anytime Algorithm for Feature Selection", Department of Computer Science and Engineering, University of South Florida, 4202 E. Fowler Avenue, ENB 118, Tampa, FL 33620, USA.

[5] J. Grass and S. Zilberstein. In M. Pittarelli (Ed.), "Anytime Algorithm Development Tools", SIGART Bulletin Special Issue on Anytime Algorithms and Deliberation Scheduling, 7(2):20-27, (1996).

[6] S. Zilberstein. Ph.D. dissertation, "Operational Rationality through Compilation of Anytime Algorithm", Computer Science Division, University of California at Berkeley, (1993).

[7] J. Grass and S. Zilberstein. In M. Pittarelli (Ed.), "Anytime algorithm development tools", SIGART Bulletin Special Issue on Anytime Algorithms and Deliberation Scheduling, 7(2):20-27, (1996).

[8] J.W.S. Liu, K.-J. Lin, W.K. Shin, A.C.S Yu "Algorithms for Scheduling Imprecise Computation" IEEE Trans. Computers ,Vol. 19,No.9, Sept. 1991, pp. 156-1,173.

[9] J.W.-S. Liu, K.-J. Lin, W.-K. Shih, A.C.-S. Yu, J.-Y. Chung, and W. Zhao, "Algorithms for Scheduling Imprecise Computations," Computer, vol. 24, no. 5, pp. 58–68, May 1991.

[10] J.-Y. Chung, J.W.-S.Liu, and K.-J. Lin, "Scheduling Periodic Jobs that Allow Imprecise Results," IEEE Trans. Computers, vol. 19, no. 9, pp. 1,156–1,173, Sept. 1990

[11] W.-K. Shih, J.W.-S. Liu, and J.-Y. Chung, "Algorithms for Scheduling Imprecise Computations to Minimize Total Error,“ SIAM J. Computing, vol. 20, no. 3, July 1991.

[12] W.-K. Shih and J.W.-S. Liu, "On-Line Scheduling of Imprecise Computations to Minimize Error," Proc. 13th Real-Time Systems Symp., IEEE, Dec. 1992.

[13] W.-K. Shih and J.W.-S. Liu, "Algorithms for Scheduling Imprecise Computations with Timing Constraints to Minimize Maximum Error," IEEE Trans. Computers., vol. 44, no.3, pp. 466-471, Mar 1995.

[14] K.-J. Lin and S. Natarajan, "Concord: A System of Imprecise Computations," Proc. Compsac, IEEE , pp. 75–81, Oct. 1987.

Chapter 3
Scheduling with Resource Constraint

3.1 Introduction

In everyday life, we all have to do many things such as household work, school work and company work. These works have many tasks like shopping, going to bank for payments, and going to university for study etc. It is necessary to make a schedule to do these tasks in time, and not in time. For the case of in time, sometimes it has constraints like time, and cost. In order to make a schedule of a set of computer tasks is to determine when to execute which task, and what is their order etc. for uni-processor, multi-processor or distributed system. Then, assign the task to the specific processors.

In a real time system such as embedded systems, most of the tasks have been scheduled by the required time that is based on the given deadlines. Processes or tasks that are not able to meet the given deadline are aborted. So, it could not perform the tasks that cannot meet the deadline. And also, if the system has to perform many tasks in a limited time, it would be difficult to obtain the overall performance in that limited time. If each task can obtain the result at every step in the way of computation time, the overall performance for all of the tasks could be realized and total processing time could be reduced although the quality is not perfect.

Moreover, in a real time scheduling system, a task has certain amount of processing time to complete its execution i.e., deadline. After finished its execution the maximum response of this task is obtained. It can be said that every tasks have its start time and deadline time to execute its task. Thus, the result of a task can be obtained if it meets its deadline. If it doesn't meet its deadline, it has been aborted.

Garvey and Lesser, 1993; *Garvey et al.*, 1993; *Garvey et al.*, 1994, solved like this kind of problem with their Design-to-time real-time scheduling approach which is to solve time-sensitive problems where multiple methods are available for many sub problems by describing how it can be used to schedule anytime algorithms [4]. *Garvey et al.*, 1994 also solved the Design to time approach which is to problem solving that involves designing a solution plan dynamically at runtime that uses all of the time available to find as good a solution as it can by modeling a function of the quality of individual sub-tasks as a set of interrelated computational tasks, with alternative ways of accomplishing the overall task to provide a range answers of different qualities for the overall quality of these tasks [5].

3.2 Scheduling Mechanism

This section describes the pre-emptive static scheduling scheme based on anytime algorithm and imprecise computation in order to optimize the overall performance while reducing the idle/rest time under time constraint on uni-processor system. In order to achieve the optimal overall performance in the restricted time, by assigning the tasks that have already known their computation time at each step and by distributing the allocation of required processing time of each sub-task.

3.2.1 Scheduling with Time Constraint

When the system has many tasks to do under time restriction, it is difficult to obtain the overall performance in that restricted time i.e., hard or soft real-time deadlines. If all of the tasks have the relations i.e., dependent task, it is difficult to obtain the combined overall processing result of these tasks. Depending on the specialized purpose, a system which composed of many tasks should be considered the following cases.

3.2.1.1 Dependent task

If a system has *N* tasks, and these tasks have a relation i.e., the result of task 1 has to use in the calculation of task 2, and the result of task 2 is has to use in the calculation of task 3, and so on. Thus, these tasks are called dependent tasks.

3.2.1.2 Independent task

If the tasks are not related each other, i.e., the result of a task is not necessary to use in another task. Thus, it is necessary to consider the priority of tasks i.e., which task will do first and other will do next and so on and these tasks are called independent tasks.

3.2.1.3 Dependent and independent tasks

If some of the tasks are related each other and others are not. So, these tasks can be considered as just combination of tasks.

3.2.2 *Characteristic of Scheduling*

Table 3.1 shows the characteristic of scheduling problems and the characteristic such as the type of task, release time, and deadlines of proposed algorithm.

Table 3.1: Characteristic of scheduling problems

Characteristic of scheduling problems		**Proposed algorithm**
Process/task type	*(a) Static*	Yes
	(b) Periodic	
	(c) Asynchronous	
	(d) Both	
Release time	(a) Unknown in advance	
	(b) Same	
	(c) Start of period	
	(d) Arbitrary	Yes
Deadlines	(a) Unknown in advance	
	(b) Same	
	(c) End of period	
	(d) Arbitrary	Yes
Process synchronous	(a) Preemptive (interruptible)	
	(b) Resource constraint	Yes
	(c) Run-time mechanism	
	(d) Non-preemptive	
	(e) General exclusion	
Computation times	(a) Uniform	
	(b) Arbitrary integer	
	(c) Arbitrary real	Yes
Number of processors	*(a) 1*	Yes
	(b) 2	
	(c) n (pre-assigned)	
	(d) n	
Processor speeds	(a) Identical	
	(b) Arbitrary	Yes
Measures of performance	*(a) Schedule length*	Yes
	(b) # of processors	
	(c) Lateness	
Optimality	*(a) Heuristic*	
	(b) Optimal	Yes

3.3 Proposed Scheduling Mechanism

This proposed scheduling model is constructed based on the imprecise computation model and the concept of anytime algorithm focuses on the time-quality trade-off problems encountered in real-time system. If a system has combination of tasks and these tasks return a partial result at every step, then they can be scheduled by distribution of allocating time requirement of each task to provide the overall processing result within restricted time for all tasks. It is similar to deliberation scheduling which is the process of allocating computation time by explicit manipulation of expectations on the behavior of the environment and taking account for the costs and benefits of the computational resources (time, cost, etc.) [1],[2].

3.3.1 Basic Workload Model

In this sub-section, how to construct the proposed scheduling model is described. In the hard real-time system, if the execution time is restricted, i.e., the given execution time is less than the required execution time then some of the combined tasks cannot be finished, so that the overall processing result cannot be realized. The less important tasks which are not able to meet the hard real-time deadline are left and unfinished.

A system based on the imprecise computation method is called an imprecise task system. The imprecise computation method i.e., each task is broken up into multiple problem solving steps, which is one of the tools to solve the trade-off problems between the processing time and the quality of result. Basically, the imprecise task system has two parts say mandatory and optional. In the conventional method, a task is logically divided into 2 sub-tasks in two parts say mandatory M and optional O as shown in Fig. 3.1. The mandatory task must be completed before the deadline of the task to produce the imprecise result with the acceptable quality. Thus, the imprecise result can be obtained from the mandatory part. The optional sub-tasks are used to refine the result of mandatory sub-tasks, so that the more imprecise result or the precise result can be obtained from the optional part.

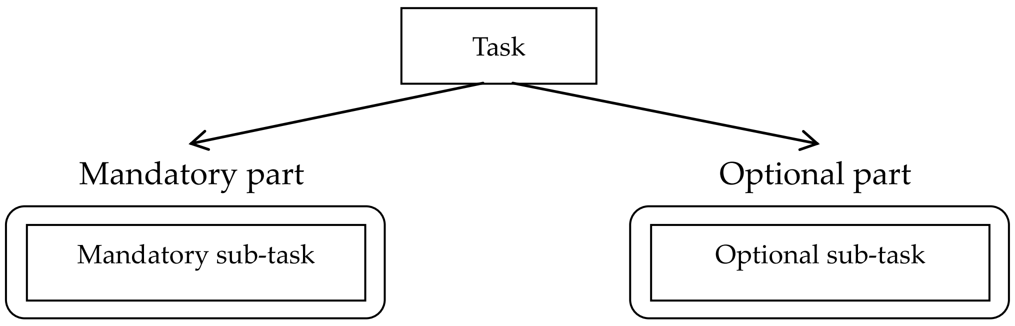


Fig. 3.1: Imprecise computation model in the conventional method

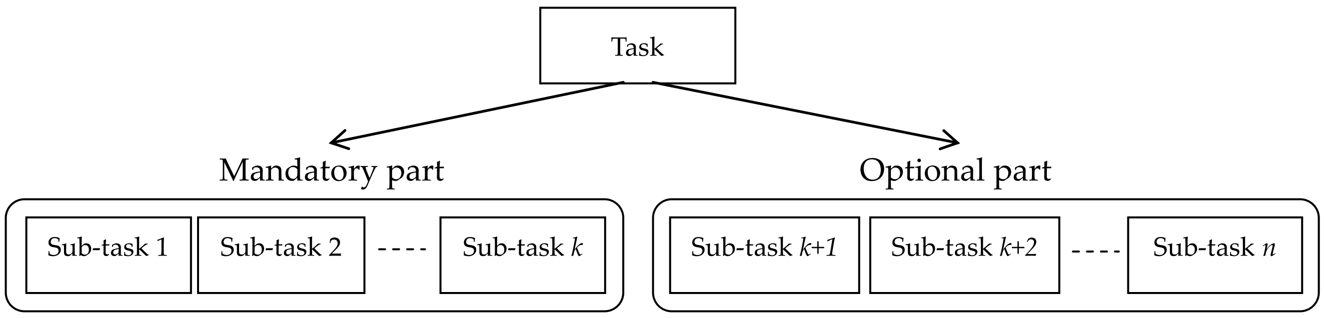


Fig. 3.2: Imprecise computation model in the proposed method

In this proposed method, each task is logically divided into n sub-tasks, say 1, 2, …, n and $n > 2$. For the division of mandatory and optional, the random number k is used and which is $1 < k < n$. The mandatory part has k sub-tasks and, the optional part has (n-k) sub-tasks to refine the result of the mandatory sub-tasks as shown in Fig. 3.2.

Definition and Terminology

The followings are the definitions and terminology used in this proposed scheduling mechanism.

Let us consider the imprecise task system T that composed of a set of N tasks with the hard/soft real-time deadlines in order to obtain the sub-optimal schedule for the imprecise overall result under the processing time constraint on uni-processor system.

Suppose that

$$T = \{T_i^{n_i}\}\ ,\ i = 1, 2, \ldots, N$$

In each task, $T_i^{n_i}$, the subscript i denotes the task number and the superscript n_i denotes the total number of sub-task of each task i. Each task $T_i^{n_i}$ is characterized by the parameters which are the rational numbers:

r^i = ready time of task i at which $T_i^{n_i}$ becomes ready for execution
d^i = deadline of task i at which $T_i^{n_i}$ must be completed
$m_i^{n_i}$ = mandatory processing time of mandatory sub-task of task $T_i^{n_i}$ for the execution of feasible result

Here, the mandatory part is performed for the feasible (acceptable) result and the required processing time of all sub-tasks in the mandatory part is combined into m_i which is the total mandatory processing time of sub-task m_i^j, $i = 1, 2, \ldots, N$.

i.e., $$m_i = \sum_{j=1}^{k} m_i^j$$

$o_i^{n_i}$ = optional processing time of optional sub-task of task $T_i^{n_i}$ for the execution of sub-optimal result
t_i = the amount of processing time assigned to the task $T_i^{n_i}$
$p_i = m_i + o_i^{n_i}$ = the total processing time of each task $T_i^{n_i}$ to completion

Feasible schedule

It is a valid schedule which contains a list of the mandatory sub-tasks of tasks completed by its deadline for the acceptable overall processing result.

Sub-optimal schedule

It is a valid schedule which contains a list of the mandatory sub-tasks and the optional sub-tasks of tasks completed by its deadline for the imprecise overall processing result.

Optimal schedule

It is a valid schedule which contains a list of mandatory sub-tasks and the final optional sub-tasks of tasks for the precise overall processing result.

Rescheduling

If the two schedules have different sub-task numbers with the same percentage of overall processing result due to the unexpected event or inaccurate predictions, then it is necessary to do rescheduling.

Idle processing time

The processing time that is not enough to perform any sub-tasks before the deadline of related task and before the ready time of next task. If $t_i < p_i$, then $p_i - t_i$ is the amount of idle processing time of discarded optional sub-tasks and let $s_i = t_i - m_i < o_i$ is the amount of processing time of absolutely discarded optional sub-task. If $s_i = o_i$ or $p_i = t_i$, then the task $T_i^{n_i}$ can be said that it is precisely scheduled.

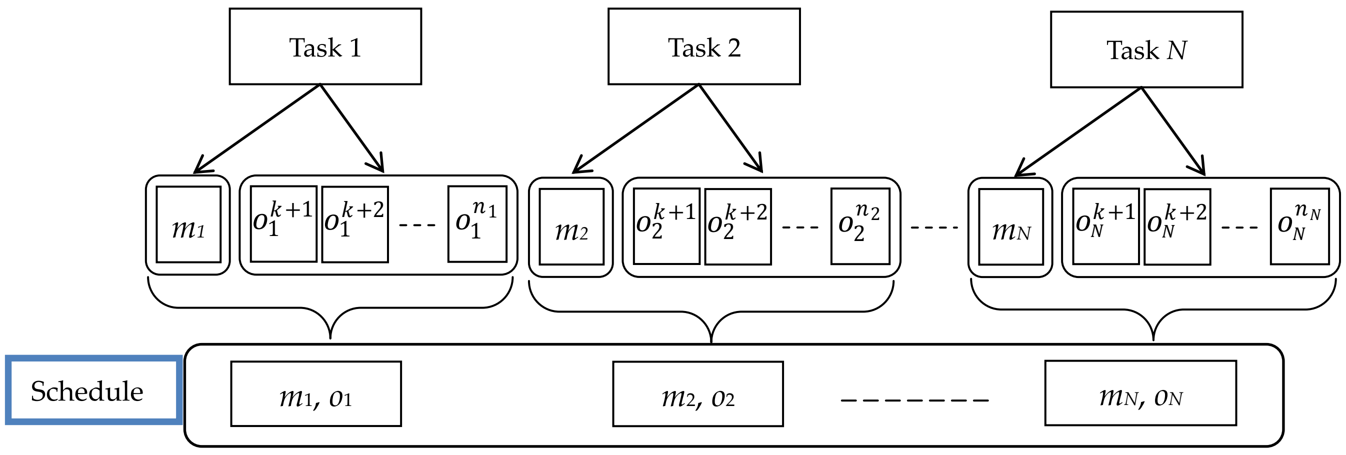


Fig. 3.3: Basic workload model of the proposed method

The sub-tasks are performed by the corresponding steps based on the concept of anytime algorithm as shown in Fig. 3.4 as an example.

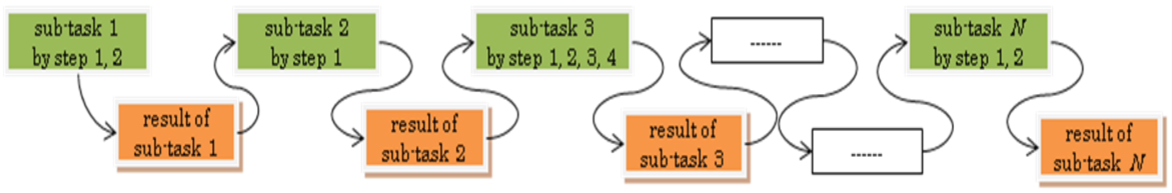


Fig. 3.4: Process flow of *N* tasks

The processing time of each sub-task by the related steps is less than or equal to the processing time of the whole task at final step. It is assumed that these tasks are independent and performed by priority i.e., task 1 by its corresponding sub-task is performed first as the priority work, then task 2 by its corresponding sub-task and so on. The current computation at each step is related to the previous step, i.e., each sub-task can be stopped at anytime with the approximated result and then it can be resumed with the minimal overhead time.

The returned result of the current sub-task is inputted to the next sub-task for e.g., if the sub-task of task 1 is stopped at step 3, the approximated result at step 3 of this task can be outputted or the output of task 1 at step 3 is applied to perform the computation of the sub-task of task 2, and so on. Thus, the different intermediate (imprecise) result could be obtained at the intermediate processing time. So, the amount of processing time of discarded optional sub-tasks would be reduced and the imprecise overall result of all tasks could be realized even though the quality of result is not perfect.

3.3.2 *Scheduling Algorithm*

***Earliest Deadline First algorithm (EDF)* [*without priority task*]**

1. Input tact time (pre-run time) T, number of tasks and the deadline of each task performed by steps.
2. Search the minimum deadline or (the earliest deadline first) of each sub-task of task i which is less than or equal to T, and let it be T_1, then search the next minimum deadline which is $\leq (T - T_1)$, and let it be T_2 until no more execution time left or insufficient execution time for any deadline left i.e., $0 \leq T_N \leq (T - (T_1 + T_2 + \ldots + T_{N-1})$
3. If all or some of the tasks assigned by the related processing time, then first schedule i.e., S_1 is obtained.
4. If the system received the stop signal from the checking point, then output the intermediate result.
5. Repeat step 2 to 4 until no more processing time left or all of the tasks are assigned by related processing time.

3.3.3 *Evaluation of the Overall Processing Result of Dependent Tasks*

In this sub-section, how to evaluate the overall processing result of the combination of dependent tasks is explained. A task is divided into many sub-tasks that are performed by many steps. For example, a task is divided into n sub-tasks performed by n steps, the precise or 100% result is obtained by performing step n e.g., if the task is performed only step k, $k < n$, the intermediate result can be obtained at step k of this task.

Let us consider a set of dependent tasks

$$T = \{T_i^{n_i}\}\ ,\ i = 1, 2, \ldots, N$$

Where

n_i = the steps of each tasks $T_1, T_2, \ldots, T_N$ respectively

$f_1, f_2, \ldots, f_N$ = the executable functions 1, 2, …, N respectively and,

$x_1^{n_1}$ = $f_1(x)$ = the output $x_i^{n_1}$ of task T_1 at step n_1 is evaluated over the function f_1 using input x

$x_2^{n_2}$ = $f_2(x_1^{n_1})$ = the output $x_2^{n_2}$ of task T_2 at step n_2 which is evaluated over the function f_1 using input $x_1^{n_1}$

⋮

$x_N^{n_N}$ = $f_N(x_{N-1}^{n_{N-1}})$ = the output $x_N^{n_N}$ of task T_N at step n_N is evaluated over the function f_1 using input $x_{N-1}^{n_{N-1}}$

In case of dependent tasks, the current result is evaluated by using the previous result which is one of the concept of anytime algorithm, thus, the overall processing result y can be realized by the composition of the functions $f_1, f_2, \ldots, f_N$

i.e.,

$$y = f_N \circ f_{N-1} \circ f_{N-2} \circ \ldots f_k \circ f_{k-1} \circ \ldots f_2 \circ f_1 \circ x \qquad (3.1)$$

Where

x = input

y = output

If the system received the stop signal, the current intermediate result can be obtained. The percentage of relative error is calculated by

$$\text{Percentage of relative error} = \frac{[\text{the precise result} - \text{the approximated result}]}{\text{the precise result}} \times 100 \qquad (3.2)$$

(1) Is the intermediate overall processing result acceptable or not?

It is necessary to examine that whether the intermediate result is acceptable or not, so the necessary *conditions* that the minimum required steps of each task for the acceptable result are determined as follows:

Suppose that

k_i = minimum required steps that must be performed for the acceptable results of task T_i , $1 < k_i < n_i$

and,

$x_1^{k_1}$ = $f_1(x)$ = the output $x_i^{k_1}$ of task T_1 at step k_1 is evaluated over the function f_1 using input x

$x_2^{k_2}$ = $f_2(x_1^{k_1})$ = the output $x_2^{k_2}$ of task T_2 at step k_2 which is evaluated over the function f_1 using input $x_1^{k_1}$

$\vdots$

$x_N^{k_N} = f_N(x_{N-1}^{k_{N-1}})$ = the output $x_N^{k_N}$ of task T_N at step k_N is evaluated over the function f_1 using input $x_{N-1}^{k_{N-1}}$

Thus, the acceptable overall result at intermediate step k is determined by

$y = f_k \circ f_{k-1} \circ \ldots f_2 \circ f_1 \circ x$ the acceptable result of task T_i at each step k_i, $1 < k_i < n_i$, and $1 \leq i < N$

(2) How does the quality of the overall processing result become?

In this approach, the result of the previous function is applied to evaluate the current step, so the quality of the current result becomes gradually improve than the previous result that is satisfied one of the properties of anytime algorithm.

From the viewpoint of probability theory, the evaluation of the overall processing result is described as follows:

Let

Sample space S: the set of all possible attention points at final step, thus, no. of all resultant points in $S = |S|$

and, its discrete random variable X can be defined as

X: no. of resultant attention points at each step

Therefore,

the elements of X are $x_1, x_2, \ldots, x_n$ for the steps $1, 2, \ldots, n_1$ and, its values are $|x_1|, |x_2|, \ldots, |x_{n_1}|$

i.e., $|x_1|$ = no. of resultant attention points at step 1

$|x_2|$ = no. of resultant attention points at step 2, and so on.

The execution of the current step is used the result of the previous step, i.e., the result of step 1 is used in step 2, and the result of step 2 is used in step 3 and so on. So, the result will be gradually improved at each step and the accuracy of result of a task at each step is generally considered by Cumulative Distribution Function (CDF), $F(x_i)$.

$$\begin{array}{c}\text{Intermediate result}\\ \text{at current step } i\end{array} = F(x_i) = \frac{\text{no. of approximated results at current step}}{\text{no. of precise results at final step}} \tag{3.3}$$

i.e.,

$$\text{the result at step } 1 = \frac{|x_1|}{|S|} = p(x_1) = P(X \le x_1) = F(x_1)$$

$$\text{the result at step } 2 = \frac{|x_1|}{|S|} + \frac{|x_2|}{|S|} = p(x_1) + p(x_2) = P(X \le x_2) = F(x_2)$$

$$\vdots$$

$$\begin{aligned}\text{the result at step } n_1 &= \frac{|x_1|}{|S|} + \frac{|x_2|}{|S|} + \ldots.. + \frac{|x_{n_1}|}{|S|}\\ &= p(x_1) + p(x_2) + \ldots + p(x_{n_1}) = P(X \le x_{n_1}) = F(x_{n_1})\end{aligned}$$

In general,

$$\text{the result at step } i = \sum_{i=1}^{n_1} \frac{|x_i|}{|S|} = P(X \le x_i) = F(x_i), \quad i = 1,2,\ldots,n_1 \tag{3.4}$$

Where

$$F(x_1) \le F(x_2) \le \ldots \le F(x_{n_1})$$

As mentioned above, the quality of result becomes increasingly at each step and it is a monotonically increasing function that can be defined at each step of the task. Therefore, its performance curve can be expressed as shown in Fig. 3.5 as an example. It expresses the quality of result of a task at each step *i* by the performance curve which improves as the processing time increases.

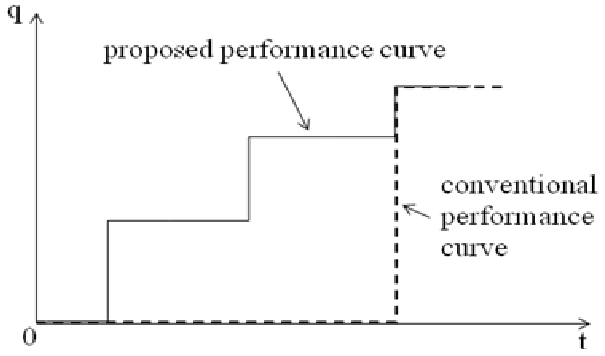


Fig. 3.5: Performance curves of the quality of result of the task

3.3.4 Determination of Quality of Overall Processing Result

In order to determine the quality of overall processing result of the combination of many tasks, unequal chance i.e., non-uniformly distribution is considered.

For the combination of many tasks, the quality of overall processing result of tasks $T_1, T_2, \dots, T_{\text{N}}$ with respect to the steps n_1, n_2, …, n_N respectively would be unequally or non-uniformly distributed by

$$Q_1 = \begin{cases} F_1(x_1) \\ F_1(x_2) \\ \vdots \\ F_1(x_{n_1}) \end{cases}$$
$$Q_2 = \begin{cases} F_2(x_1) \\ F_2(x_2) \\ \vdots \\ F_2(x_{n_2}) \end{cases}$$
$$\vdots$$
$$Q_N = \begin{cases} F_N(x_1) \\ F_N(x_2) \\ \vdots \\ F_N(x_N) \end{cases}$$

In case of the dependent tasks, i.e., the result of one task affects the result of a next task. Therefore, the overall quality (performance) of the combination of many tasks can be defined by

$$Q = Q_1 \circ Q_2 \circ \dots \circ Q_N \qquad (3.5)$$

3.3.5 Analysis of Quality of the Overall Processing Result

Algorithm for the estimation of the overall processing result

(1) Determine the condition that minimum required step number of task 1, 2, …, N and let them be $k_1, k_2, \ldots, k_N$.
(2) Select the executable function f_1 with input value x and the step no. of task 1, i.e., c_1 which must be satisfied the condition that $c_1 \geq k_1$.
(3) Execute the function f_1 with input value x and c_1, i.e., $f_1(x, c_1)$ and let the result be y_1.
(4) Select the next executable function f_2 with input value y_1 and the step no. of the current task, i.e., c_2 which must be satisfied the condition that $c_2 \geq k_2$.
(5) Execute the function f_2 with input value or the previous result y_1 and c_2, i.e., $f_2(y_1, c_2)$ and let the result be y_2.
(6) Repeat the step 4 until the result y_N would be obtained.
(7) Evaluate the overall processing result: $y = y_N \circ y_{N-1} \circ \ldots \circ y_2 \circ y_1 \circ x$
(8) Calculate the quality of the overall processing result by equation (5).

Dependent case

In this case, if the system is composed of the combination of many tasks that are inter-related each other in a particular time constraint. The calculation of current result for a task is depend on the result of previous task i.e., calculation of task 2 is based on the result of task 1, and calculation of task 3 is based on the result of task 2, and so on. Then, the total performance (combined quality) can be evaluated by using the **conditional performance profile (CPP)** that is described in chapter 2. Formulation of quality functions for dependent case is described as follows:

Let

$T_1, T_2, \ldots, T_N$ = task 1, 2, 3, …, N respectively
$Q_1, Q_2, Q_3, \ldots, Q_N$= quality of task 1, 2, 3, …, N respectively
$t_1, t_2, t_3, \ldots, t_N$ = total required time for each tasks $T_1, T_2, \ldots, T_N$ respectively
The total performance Q of combination of N tasks is

$$Q = Q_N\{Q_{N-1}, \ldots [Q_2 (Q_1 (Q_0, t_0), t_1), t_2], \ldots, t_{N-1}\}$$

Here,

Q_0 and t_0 means that the initial quality 0 and initial starting time 0 at step 0

$Q_1(Q_0, t_0)$ means that the quality of task 1 has input quality Q_0 and processing time t_0

$Q_2(Q_1(Q_0, t_0), t_1)$ means that the quality of task 2 has input quality Q_1 and processing time t_1 and so on.

In this case, the total quality Q might be less than or equal to the maximum quality i.e., 1.

Independence case

In order to obtain the maximum performance at any time, by choosing the priority task that has predefined executing time with high quality result if it meets the current processing time. If the time is not enough i.e., its execution time is greater than current execution time, then choose the suitable processing time with appropriate quality of result to avoid the huge amount of rest time and so on. Formulation of quality functions for independent case is described as follows:

Suppose that when the system is composed of 4 tasks and the total processing time is restricted, then the maximum performance in a restricted time can be obtained as described in the following.

Let

T = Restricted time

$T_1, T_2, \ldots, T_4$ = task 1, 2, …, 4 respectively

q_{1i} = quality of task T_1 at step i, where i = 1, 2, …, a

q_{2j} = quality of task T_2 at step j, where j = 1, 2, …, b

q_{3k} = quality of task T_3 at step k, where k = 1, 2, …, c

q_{4l} = quality of task T_4 at step l, where l = 1, 2, …, d

t_1, t_2, t_3, t_4 = total required time for task T_1, T_2, T_3 and T_4respectively

Q = maximum performance of above 4 tasks

In order to optimize the overall performance in a restricted time, the constructed optimization model by the linear programming model is

$$Max\ \ Q = \sum_{t=0.0}^{T} (q_1(t_1) + q_2(t_2) + q_3(t_3) + q_4(t_4)) / 4.0 \qquad (3.6)$$

Subject to

$$\begin{aligned} t_{11} + t_{12} + t_{13} + \ldots + t_{1a} &\leq t_1 \\ t_{21} + t_{22} + t_{23} + \ldots + t_{2b} &\leq t_2 \\ t_{31} + t_{32} + t_{33} + \ldots + t_{3c} &\leq t_3 \\ t_{41} + t_{42} + t_{43} + \ldots + t_{4d} &\leq t_4 \end{aligned}$$

and

$$t_1 + t_2 + t_3 + t_4 \leq T$$
$$t_1, t_2, t_3, t_4 \geq 0$$

Figure 3.6 shows one of the experimental results of the proposed scheduling mechanism. Here, we can easily see that the assignment of processing time is effectively distributed with less idle time. The performance curve shown in Fig. 3.7 represents the overall performance of 4 tasks and it is satisfies the definition of performance profile. Hence, the scheduling result realizes the optimal performance. Furthermore, the proposed method can reduce the idle/rest time.

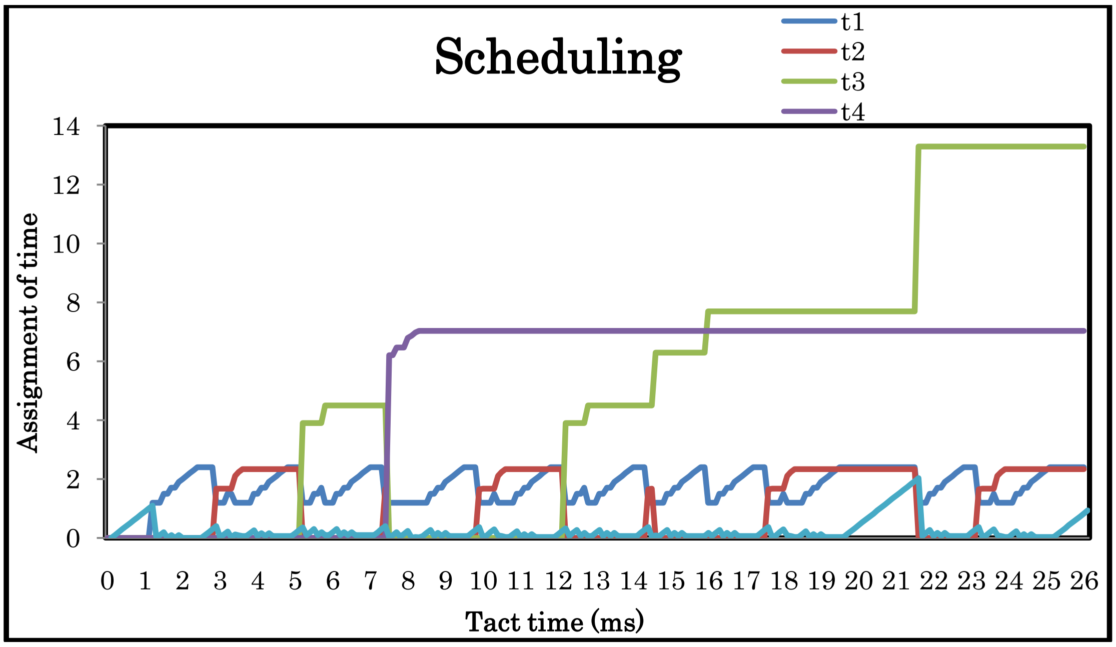


Fig. 3.6: Scheduling result for 4 tasks

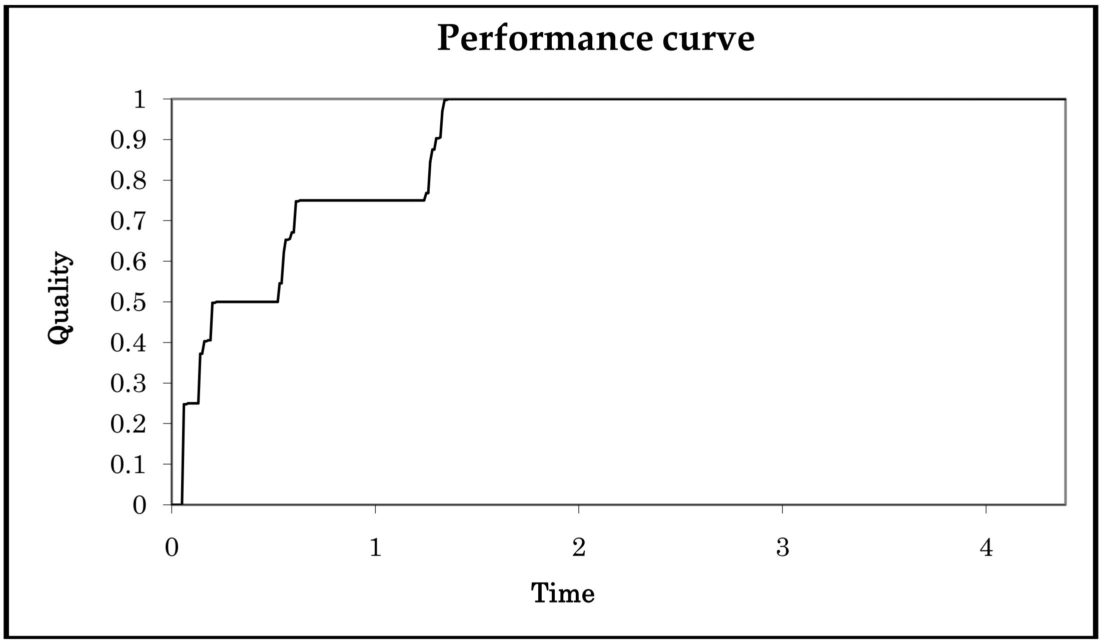


Fig. 3.7: Overall performance of 4 tasks

Table 3.2 shows the experimental data of proposed scheduling mechanism.

Table 3.2: Scheduling result for independent case

Processing time (s)	**Assignment of tasks by processing time**								**Total Quality Q**	**Rest/Idle time (s)**
	t_1	q_1	t_2	q_2	t_3	q_3	t_4	q_4		
0	0	0	0	0	0	0	0	0	0	0
0.1	0	0	0	0	0	0	0	0	0	0.1
0.2	0	0	0	0	0	0	0	0	0	0.2
0.3	0	0	0	0	0	0	0	0	0	0.3
|	|	|	|	|	|	|	|	|	|	|
|	|	|	|	|	|	|	|	|	|	|
8.1	1.2	0.23	0	0	0	0	6.87	0.86	0.2735	0.03
8.2	1.2	0.23	0	0	0	0	6.98	0.93	0.291	0.02
|	|	|	|	|	|	|	|	|	|	|
|	|	|	|	|	|	|	|	|	|	|
17	2.2	1	0	0	7.7	0.956	7.03	1	0.739	0.07
17.1	2.3	1	0	0	7.7	0.956	7.03	1	0.739	0.07
17.2	2.4	1	0	0	7.7	0.956	7.03	1	0.739	0.07
|	|	|	|	|	|	|	|	|	|	|
|	|	|	|	|	|	|	|	|	|	|

20.3	2.4	1	2.34	1	7.7	0.956	7.03	1	0.989	0.83
20.4	2.4	1	2.34	1	7.7	0.956	7.03	1	0.989	0.93
20.5	2.4	1	2.34	1	7.7	0.956	7.03	1	0.989	1.03
20.6	2.4	1	2.34	1	7.7	0.956	7.03	1	0.989	1.13
20.7	2.4	1	2.34	1	7.7	0.956	7.03	1	0.989	1.23
|	|	|	|	|	|	|	|	|	|	|
|	|	|	|	|	|	|	|	|	|	|
23.6	1.5	0.26	1.68	0.213	13.3	1	7.03	1	0.61725	0.09
23.7	1.2	0.23	2.11	0.547	13.3	1	7.03	1	0.69525	0.06
25.5	2.4	1	2.34	1	13.3	1	7.03	1	1	0.43
25.6	2.4	1	2.34	1	13.3	1	7.03	1	1	0.53
25.7	2.4	1	2.34	1	13.3	1	7.03	1	1	0.63
25.8	2.4	1	2.34	1	13.3	1	7.03	1	1	0.73
25.9	2.4	1	2.34	1	13.3	1	7.03	1	1	0.83
26	2.4	1	2.34	1	13.3	1	7.03	1	1	0.93

3.3.6 *Discussion*

This section describes the comparison of the proposed scheduling method and the conventional scheduling method by imprecise computation.

Table 3.3: Comparison of scheduling result for independent case

	Milestone	**Sieve**	**Multiple version**	**Proposed method**
Processing time		Less		Reduce the overhead time
Cost	Overhead in recording intermediate result	Less	Overhead to store multiple version	Overhead in recording intermediate result
Higher scheduling		Higher scheduling overhead	Overhead	Overhead
Intermediate result	1	1	Many	Range answer
Precise result	1	1	1	1

3.4 Summary

This chapter explained the problems in real time system and their approaches, and the basic idea or solving method for scheduling based on imprecise computation and anytime algorithm. First, the proposed scheduling mechanism under time constraint is described for dependent and independent case, its basic workload model and algorithm. Then, characteristic of scheduling is shown in table. After that, evaluation of overall processing result for the combination of many tasks is described.

The quality function to optimize the overall performance under the processing time constraint for the dependent, independent and both cases are constructed based on the concept of linear programming and imprecise computation. The experimental result is shown by graphical and tabular representations. The performance curve of time and quality graph shows that the overall processing result is gradually improved as the processing time increases that is satisfied the definition of performance profile. Finally, the comparison of proposed scheduling and the conventional imprecise scheduling method is described by table.

References

[1] M. Boddy and T. Dean. "*Decision-theoretic deliberation scheduling for problem solving in time-constrained environments*", Artificial Intelligence, 67(2):245--286, 1994.

[2] Thomas Dean, "*Deliberation Scheduling for Time-Critical Scheduling in Stochastic Domains*"

[3] S. Zilberstein, "Monitoring Anytime Algorithms"

[4] A. Garvey, V. Lesser. "*Design-to-Time Real-Time Scheduling*", IEEE Transactions on Systems, Man and Cybernetics, 1993 ftp://ftp.cs.umass.edu/pub/lesser/garvey-dtt-smc.ps

[5] Garvey, Alan and Victor Lesser, "*Design-to-time Scheduling and Anytime Algorithms*", SIGART Bulletin, 7 (2):16--19 (1996). http://citeseer.comp.nus.edu.sg/85186.html

[6] Dean, Thomas and Boddy, Mark, "*An Analysis of Time_Dependent Planning*", Proceedings Anytime Algorithm AI-88, St. Paul, Minnesota I, 49–54, 1988.

[7] Hasegawa, H. Kubota and J. Toriwaki, "*Automated construction of image processing procedures by sample figure presentation*", Proc. of 8th Int'l Conference on Pattern Recognition (ICPR1986), Vol. 1, pp.586-588 (Oct.1986).

[8] J.-Y. Chung, J.W.-S.Liu, and K.-J. Lin, "S*cheduling Periodic Jobs that Allow Imprecise Results*", IEEE Trans. Computers, vol. 19, no. 9, pp. 1,156–1,173, Sept. 1990

[9] W.-K. Shih, J.W.-S. Liu, and J.-Y. Chung, "*Algorithms for Scheduling Imprecise*

Computations to Minimize Total Error", SIAM J. Computing, vol. 20, no. 3, July 1991.

[10] W.-K. Shih and J.W.-S. Liu, "*On-Line Scheduling of Imprecise Computations to Minimize Error*", Proc. 13th Real-Time Systems Symp., IEEE, Dec. 1992.

[11] W.-K. Shih and J.W.-S. Liu, "*Algorithms for Scheduling Imprecise Computations with Timing Constraints to Minimize Maximum Error*", IEEE Trans. Computers., vol. 44, no.3, pp. 466-471, Mar 1995.

Chapter 4
Anytime Algorithmic Image Processing

4.1 Introduction

In real-time image/video processing system, if we seriously consider the less processing time in a particular task, the processing quality would becomes sacrifice and conversely, if we adhere to the processing quality too much, the processing time might needed much time.

The purpose of real-time image processing system involves with the improvement of the quality of video (image sequence) by enhancing the image in pre-processing like noise reduction. Furthermore, many imaging applications are time critical and are computationally intensive. For example, in image transmission system, the digital images require huge amounts of space for storage and large bandwidths before the transmission of image, so it is necessary to process the images in pre-processing such as filtering and enhancement.

The outputs are required not only the perfect but also the timely imperfect results with deadline satisfaction. Moreover, the quality of image processing is usually evaluated by high extraction rate or low error rate. It is easy and clear to evaluate the single image processing task, but if the system is composed of many tasks, it would becomes difficult to evaluate the combined processing result due to the results of image processing vary according to the combination of the methods and tasks. For instance, in image/video tracking or image transmission which is necessary to realize the intermediate result i.e., to achieve the better resolution with data transmissions and computations as low as possible, at intermediate processing time.

There are many methods for image/video processing results reported in the real-time image processing from the different viewpoints such as hardware platform i.e., FPGAs, DSP (Digital Signal Processors), GPU, Hybrid and PC based systems, and software platform i.e., pipeline, parallel processing and algorithm.

In real time image processing system, a task can be divided into small sub-tasks in order to perform the combination of tasks with some restrictions like processing time and memory usage by giving the partial overall result. In order to obtain a result at midway processing time and the overall result of the combination of many tasks at optimal time, this chapter describes how to modify some of the conventional image processing (CIP) methods to anytime algorithmic image processing, hereafter AAIP methods, step by step and it is satisfied by all of the properties of anytime algorithm. Thus, the rest of this chapter describes as follows:

First of all, the conventional image processing operations in low, intermediate, and high level, and its existing methods/techniques are analyzed. Then, categorize the conventional image processing methods from the viewpoint of anytime algorithm. After that, how to modify the conventional image processing methods to AAIP is explained for each method. Finally, the experimental results of AAIP methods and its related performance curves are expressed.

The general concept of digital image processing is briefly explained as follows:

Digital image processing is one of the interested subjects in computer science. It is to analyse and manipulate digital images with a computer. Most of the digital image processing methods have been proposed and there have been almost all of them are going to emphasize on image quality for visualization, compression of image size for web application, and processing time for the image retrieval system etc. It is especially for human visualization like enhancement, restoration etc. by analyzing with computer for the specific purposes such as documents analysis, textures analysis, biometrics, object recognition and so on. It has generally three steps:

1. Import an image with a scanner or a digital camera.
2. Manipulate and analyse the image in some ways.
3. Output the image.

The result may be an image or a report of information based on analysis of the image. In order to be able to produce the useful result, there are many different methods and techniques which are dealing with three types of image processing level such as low level processing, intermediate level processing, and high level processing.

(1) Low level processing

Low level processing involves primitive operations such as noise reduction, image sharpening, brightness, contrast enhancement, and thresholding etc. In this level, it is characterised by the fact that both its inputs and outputs are images. It is concerned with work at the binary image, typically creating a second "better" image which depends on the first processed image by changing the representation of the image by removing unwanted data e.g., noise, and enhancing wanted data e.g., brightness using filters, operators etc.

(2) Intermediate level processing

To process an image in this level involves the tasks like segmentation that partitioning an image into regions or objects and classification or recognition of individual objects for object recognition. It is characterized by the fact that its inputs generally are images, but its outputs are image attributes or information that are extracted from those images such as edges, contours, and the identity of individual objects. The operations in this level are edge detection, labelling, morphological processing, segmentation, template matching, boundary detection, and so on.

(3) High level processing

This level includes "making sense" of an ensemble of recognized objects, as in image analysis, and performing the cognitive functions normally associated with computer vision. The operations in this level are feature extraction, coding, image reconstruction, image understanding etc. High level preprocessing interfaces the image to some knowledge base. This associates shapes discovered during previous level of processing with known shapes of real objects. The results from the algorithms at this level are passed on to non image procedures, which make decisions about actions following from the analysis of the image.

4.2 Anytime Algorithmic Image Processing

This section describes operations and methods that are fundamental to digital image processing. There are various image processing operations and methods depending on the specialized purposes for the specific applications. So, it is necessary to apply the appropriate operators and suitable methods in which operations in order to achieve the goal. These operations are based on the image histogram, simple mathematics, convolution, and mathematical morphology and so on. Generally, image processing operations can be roughly divided into the following categories:

- Image enhancement and restoration
- Image compression/data compression
- Measurement extraction
- Coding
- Feature detection
- Morphological operation
- Object description and classification

  etc.

And, the typical image processing methods are

- Filter type
- Histogram based type
- Condition type
- Morphological type
- Gradient type

  etc.

The conversion of CIP method to AAIP method, anytime algorithm is used as a basic tool. It is an algorithm which is different from the traditional algorithm and it is satisfied the features that the quality of result can be improved when the processing time increases and the result can be produced at anytime [3], [4], [5], [6]. Hence, anytime algorithmic form of an image processing task in a system can be represented as shown in Fig. 4.1.

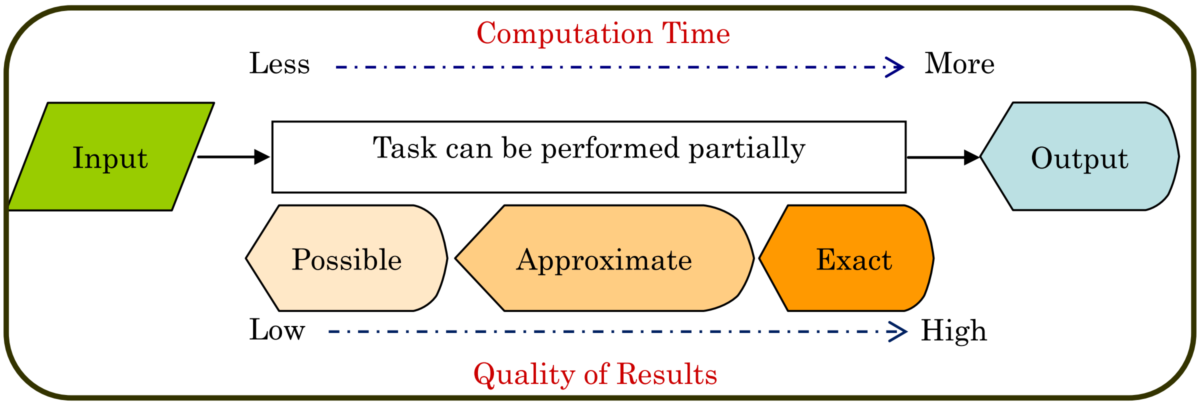


Fig. 4.1: Anytime algorithmic form of an image processing task

Therefore, it is necessary to analyze what kind of image processing tasks can be divided into sub-tasks. According to the definition of anytime algorithm and its properties, presently, the following image processing tasks with discrete type could be divided into sub-tasks in order to obtain the intermediate result at any time. These are

(1) Filter type
(2) Gradient type
(3) Morphological type, and
(4) Condition type

Hence, the following sub-sections explain how to convert conventional image processing methods to anytime algorithmic form by anytime algorithm in order to provide the better solution of time-quality trade-off problem from the viewpoint of image quality and/or processing time. Modification of some of spatial filtering method like *averaging* by *Mean* filter for noise reduction and *Gradient* by *Prewitt* filter for edge detection by the concept of anytime algorithm are found in [8], [9].

4.2.1 AA Spatial Filtering

The fundamental process in digital image processing is the image enhancement which is applied in every field for e.g., medical image analysis, and analysis of images from satellites like weather map where the images are required to be understood and analyzed. Moreover, there are large amount of data usage to process an image in pre-processing steps rather than that of intermediate and high level processing in digital image processing. Spatial filtering method is widely used in image processing, either as a preprocessing step or as a mean for gathering some interesting features such as edge, boundary, noise in an image for other image processing processes like object detection, and video tracking in image analysis.

Spatial filtering operation by discrete convolution filter is widely used in image processing, either as a preprocessing step or as a mean for gathering some interesting features such as edge, boundary, and noise from the input raw image. There are many filtering operations according to the type of the operators, their usages and the purposes. Some filtering methods are spatial filtering, linear filtering, non-linear filtering and so on. In this approach, the *Mean* filter, *Gaussian* filter, and *Gradient* filter are considered in particular. For instance, *Prewitt* operator can be used for the edge detection purpose and simple averaging method can be applied for noise reduction etc. These discrete convolution filters can be divided into different types of filter according to the concept of anytime algorithm. This concept is useful in implementing anytime function in order to perform the sub-tasks and output the partial result in intermediate processing time for the linear spatial filtering such as low pass filtering and high pass filtering.

The basic idea of how to apply anytime algorithm to the spatial filtering method by convolution filter is by dividing the filter into many different types of filter, then the task is performed by many steps using different types of filter. Here, the step would depend on the size of filter mask and the operator used, for e.g., if the mask size is 3x3, the step would be at most 9, if it is 5x5 mask the step would be at most 25 with the specific operators and operations.

Conventional linear spatial filtering

In the conventional spatial filtering, there are two types: linear and non-linear. Linear filters such as *mean* and *Gaussian* for smoothing and Gradient operators such as *Sobel*, and *Prewitt* filters for edge detection and *basic hi-pass spatial* filter for sharpening are considered as examples in this proposed method.

Generally, the response R of an m x n mask at any point (x, y) in an image, the linear spatial filtering is considered by the following expression:

$$R = w_1f_1 + w_2f_2 + \ldots + w_{mn}f_{mn} = \sum_{i=1}^{mn} w_i f_i \qquad (4.1)$$

where the w's are the coefficients of mask, the f's are the values of the image gray levels corresponding to the mask coefficients, and mn is the total number of coefficients in the mask. In general, linear filtering of an image f of size MxN with a filter mask of size mxn is given by the expression:

$$g(x,y) = w(x,y) * f(x,y) = \sum_{i=-a}^{a}\sum_{j=-b}^{b} w(i,j)\, f(x-i, y-j) \qquad (4.2)$$

Where

f = input image

g = output image

$m = 2a+1$

$n = 2b+1$

w = the elements of mxn mask

and

a and b are non-negative integers.

Proposed linear spatial filtering

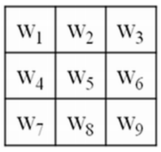


Fig. 4.2: 3x3 filter mask

For 3 x 3 filter mask as shown in Fig. 4.2, the response R at any point (x, y) in the image is given by

$$R = w_1 f_1 + w_2 f_2 + \ldots + w_9 f_9 = \sum_{i=1}^{9} w_i f_i \tag{4.3}$$

In this spatial filtering method, 3x3 filter mask is divided into 8 sub-masks as shown in Fig. 4.3 by increasing the corresponding elements in each sub-mask. Here, many different patterns (i.e., 40320=8!) of divided sub-masks like HLAC features are constructed. Among 40320 patterns, 20738 different results with different patterns are obtained after removing the patterns with same results when the image or filter mask is rotated to 180 degree.

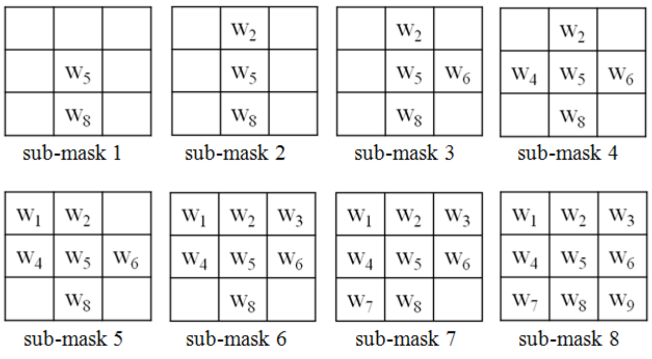


Fig. 4.3: Divided sub-masks of 3x3 filter mask

Figure 4.3 is one of the patterns with the better result among many other different patterns according to the experiments. As shown in this figure, sub-mask 1 has two elements and the first element is started from center as reference pixel in HLAC features and the other element is the lower element. In sub-mask 2, it has three elements defined by the elements of sub-mask 1 and added by the upper element. Sub-mask 3 is defined by the elements of sub-mask 2, and added by the right element, and then the upper left, the upper right and so on in the next sub-mask up to sub-mask 8.

Then, the operation is performed by divided sub-masks using 8 steps. So, the response R at any point (x, y) in the image can be performed by the following 8 steps:

Step-1 : $R_1 = w_5f_5 + w_8f_8$
Step-2 : $R_2 = R_1 + w_2f_2$
Step-3 : $R_3 = R_2 + w_6f_6$
Step-4 : $R_4 = R_3 + w_4f_4$
Step-5 : $R_5 = R_4 + w_1f_1$
Step-6 : $R_6 = R_5 + w_3f_3$
Step-7 : $R_7 = R_6 + w_7f_7$
Step-8 : $R_8 = R_7 + w_9f_9$

In each step, the previous result is used to perform the calculation of current steps in order to perform the task efficiently according to the properties of anytime algorithm. Thus, 8 different intermediate responses R's can be obtained at each step with the related processing time.

This proposed method is applied to averaging (smoothing) task by *Gaussian* filter and *Mean* filter, and sharpening task by *basic hi-pass spatial* filter using the above patterns and edge detection task by *Sobel* filter using other pattern which will be explained in the later part.

4.2.1.1 AA Low pass filtering

(1) Smoothing by Gaussian filter

In anytime algorithmic linear spatial filtering, smoothing task by using *Gaussian* 3x3 mask is done by applying divided sub-masks as shown in Fig. 4.4 Figure 4.4 shows that how to perform anytime algorithmic smoothing method by using *Gaussian* 3x3 mask. In particular, *Gaussian* 3x3 filter mask is applied as an example.

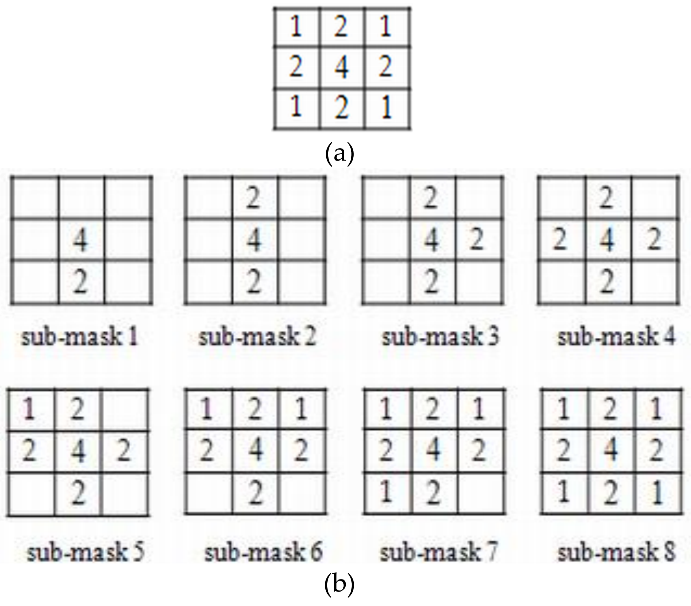


Fig. 4.4: (a) *Gaussian* 3x3 mask (b) Divided sub-masks of *Gaussian*

In general, the output image is evaluated by anytime low pass spatial function which is modified from (4.2) for anytime algorithmic low pass filtering and is given by

$$g_k(i,j) = g_{k-1}(i,j) + \frac{w_{k+1}(x,y)}{\sum_{l=1}^{k+1} w_l(x,y)} [\, f_{k+1}(x,y) - g_{k-1}(i,j)] \qquad (4.4)$$

Where

$k = 1, 2, \ldots, 8$
$i = 1, 2, \ldots, W - 1$
$j = 1, 2, \ldots, H - 1$
$f_k\,(x, y)$ = input image
$g_k\,(i, j)$ = current output
$g_{k\text{-}1}\,(i, j)$ = previous output
W and H = image's width and height
$w_k(x, y)$ = the elements of 3x3 mask

here,

w_k are positive for low pass filtering

(2) Smoothing by mean filter

AA noise reduction by simple averaging operation using 3 x 3 *Mean* filter mask could be modified by weighting each pixel value step by step. In the smoothing operator, there are 8 different results whose quality of result gradually improves as the computation time increases in the way of computing because there is 8 times summation at most. So, it has 8 steps to perform its task by using the concept of anytime algorithm. For instance, an anytime algorithmic smoothing operator could be realized by weighting each pixel in another pattern as shown in Fig. 4.5.

In AAIP, by replacing each pixel value with weighted sum (average value) of its neighboring pixel including itself step by step, the weighted values are 1/2, 1/3, …, and 1/9. In each step, the operation is performed by each divided filter, thus, the qualities of results gradually become better as processing time increases.

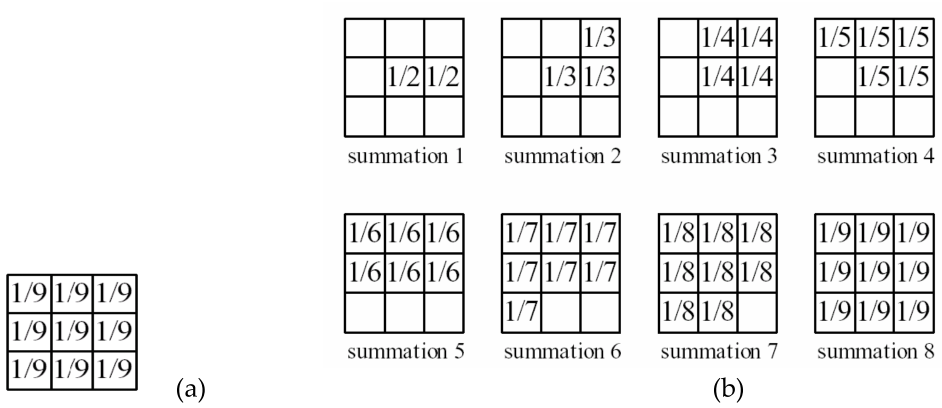
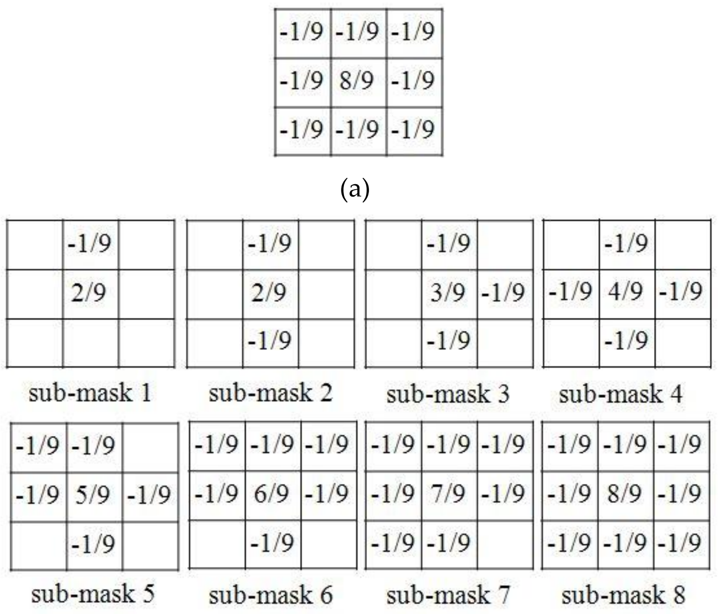


Fig. 4.5: (a) *Mean* 3x3 filter (b) Divided sub-masks of *Mean* filter

4.2.1.2 AA High pass filtering

Sharpening by basic hi-pass spatial filter

In anytime algorithmic linear spatial filtering, sharpening task by using *basic hi-pass* spatial 3x3 mask is done by the divided sub-masks as shown in Fig. 4.6. Figure 4.6 (a) is the *Basic hi-pass spatial* 3x3 mask and (b) is the divided sub-masks of *Basic hi-pass spatial* filter.



-1/9	-1/9	-1/9
-1/9	8/9	-1/9
-1/9	-1/9	-1/9

(a)

	-1/9	
	2/9	

sub-mask 1

	-1/9	
	2/9	
	-1/9	

sub-mask 2

	-1/9	
	3/9	-1/9
	-1/9	

sub-mask 3

	-1/9	
-1/9	4/9	-1/9
	-1/9	

sub-mask 4

-1/9	-1/9	
-1/9	5/9	-1/9
	-1/9	

sub-mask 5

-1/9	-1/9	-1/9
-1/9	6/9	-1/9
	-1/9	

sub-mask 6

-1/9	-1/9	-1/9
-1/9	7/9	-1/9
-1/9	-1/9	

sub-mask 7

-1/9	-1/9	-1/9
-1/9	8/9	-1/9
-1/9	-1/9	-1/9

sub-mask 8

(b)

Fig. 4.6: (a) *Basic hi-pass* spatial 3x3 mask (b) Divided sub-masks of *basic hi-pass* spatial filter

The output image is evaluated by anytime high pass spatial function for anytime algorithmic high pass filtering and is given by

$$g_k(i,j) = g_{k-1}(i,j) + g_0 + cw_k(x,y)\, f_k(x,y) \tag{4.5}$$

Where

k = 1, 2, …, 8
i = 1, 2, ..., W – 1
j = 1, 2, ..., H – 1
f_k (x, y) = input image
g_k (i, j) = current output
g_{k-1} (i, j) = previous output
$g_0 = c\, w_{centre}\,(x, y)\, f_{centre}\,(x, y)$, c = constant
W and H = image's width and height
$w_k(x, y)$ = the elements of 3x3 mask,

Here,
w_k are negative except from centre element w_{centre} (x, y) for high pass filtering

In general, the output sharpened image is obtained by

$$G(x,y) = f(x,y) + g_k(i,j) \tag{4.6}$$

Where

$G(x,y)$ = output image
$f(x,y)$ = input image
$g_k(i,j)$ = filtered image

The AA filtering method as mentioned above, the following facts that are satisfied all of the properties of anytime algorithm. That is, the result can be defined exactly at every step, thus, it is satisfied the properties (1), (2), and (3). In addition, step 1 result is applied in the evaluation of step 2, then step 2 result is applied in step 3 and so on, in order to obtain the result at every step that is satisfied by the property (4) and (5). Moreover, if the algorithm is stopped for e.g., at step–3, then the partial result of this task can obtain for this step and started again at current step, so, it is satisfied the properties (6) and (7). Hence, this method can apply to the typical image processing operations such as smoothing, noise reduction, and sharpening using the appropriate filters under the condition that the processing time is restricted.

4.2.2 *AA Gradient method*

In anytime algorithmic gradient method, edge detection task by using 3x3 mask is done by divided sub-masks as shown in Fig. 4.7. Figure 4.7 shows how to perform anytime algorithmic edge detection method by using 3x3 mask. For example, in the edge detection operator using 3 x 3 mask, there are too many combinations of weighting. This figure shows an example of the mask and their weights for each pixel. This scheme outputs 6 steps of result whose qualities gradually become better. In general, the output image is evaluated by anytime spatial function for *Gradient* method which is given by

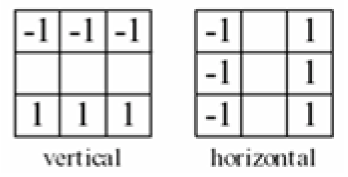
$$G_Y = G_{Y-1} + T[f_Y(x,y)] \tag{4.7}$$
$$G_X = G_{X-1} + T[f_X(x,y)] \tag{4.8}$$
$$G = |G_X| + |G_Y| \tag{4.9}$$

Where

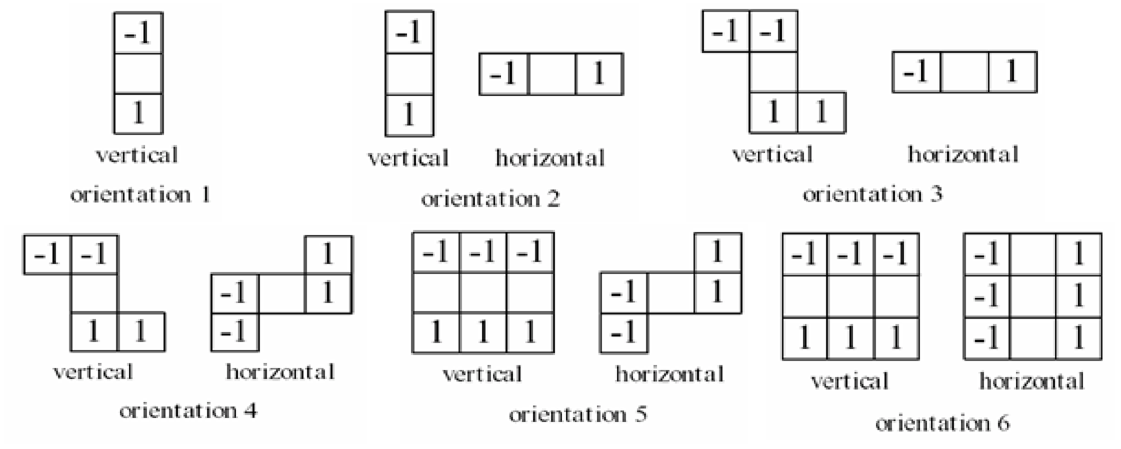
$f(x,y)$ = input image
G = output image
G_X = gradient by horizontal
G_Y = gradient by vertical
T = transformation on $f(x,y)$

(1) Edge detection by Prewitt filter

In anytime algorithmic gradient method, edge detection task by using *Prewitt* 3x3 mask is done by divided sub-masks as shown in Fig. 4.7. Figure 4.7 (a) is the *Prewitt* 3x3 mask and (b) is the divided sub-masks of *Prewitt* 3x3 mask.



(a)



(b)

Fig. 4.7: (a) *Prewitt* 3x3 mask (b) Divided sub-masks of *Prewitt*

(2) Edge detection by Sobel filter

In anytime algorithmic gradient method, edge detection task by using *Sobel* 3x3 mask is done by divided sub-masks as shown in Fig. 4.8. Figure 4.8 (a) is the *Sobel* 3x3 mask and (b) is the divided sub-masks of *Sobel* 3x3 mask.

-1	-2	-1	-1	0	1
0	0	0	-2	0	2
1	2	1	-1	0	1

(a)

0	-2	0	0	0	0
0	0	0	0	0	0
0	2	0	0	0	0
vertical			horizontal		

sub-mask 1

0	-2	0	0	0	0
0	0	0	-2	0	2
0	2	0	0	0	0
vertical			horizontal		

sub-mask 2

-1	-2	0	0	0	0
0	0	0	-2	0	2
0	2	1	0	0	0
vertical			horizontal		

sub-mask 3

-1	-2	0	0	0	1
0	0	0	-2	0	2
0	2	1	-1	0	0
vertical			horizontal		

sub-mask 4

-1	-2	-1	0	0	1
0	0	0	-2	0	2
1	2	1	-1	0	0
vertical			horizontal		

sub-mask 5

-1	-2	-1	-1	0	1
0	0	0	-2	0	2
1	2	1	-1	0	1
vertical			horizontal		

sub-mask 6

(b)

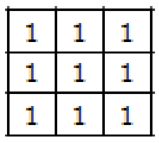
Fig. 4.8: (a) *Sobel* 3x3 mask (b) Divided sub-masks of *Sobel*

The AA Gradient type method as mentioned above, the following facts that are satisfied all of the properties of anytime algorithm. That is, the result can be defined exactly at every step, thus, it is satisfied the properties (1), (2), and (3). Step 1 result is applied in the evaluation of step 2, then, step 2 result is applied in step 3 and so on, in order to obtain the result at every step that is satisfied by the property (4) and (5). Moreover, if the algorithm is stopped for e.g., at step–3, then the partial result of this task can obtain for this step and started again at current step, so, it is satisfied the properties (6) and (7). Hence, this method can apply to the typical image processing operations such as point detection, line detection, and edge detection using the appropriate filters under the condition that the processing time is restricted.

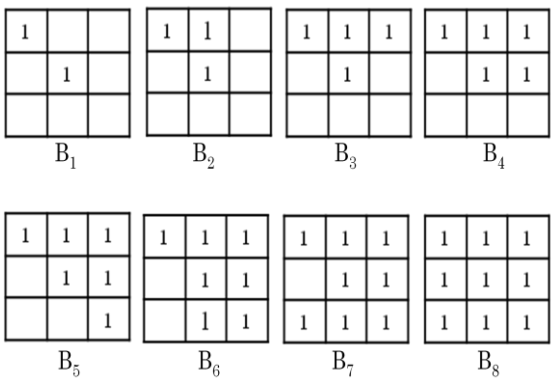
4.2.3 *AA Morphological Processing*

Morphological method is a technique for the analysis and processing of geometrical structures and features based on set theory. It is a useful theory to extract the regional shape such as boundaries, skeletons, and convex hull, especially applied in image preprocessing [1]. It is used the structuring elements to process the image for the morphological operation including erosion, dilation, opening and closing etc. It is one of the principal operations to the applications of extracting the image components or features that are useful in the representation and description of object shape e.g., boundary extraction. I consider that it can be applied by anytime algorithm tool by dividing the structuring element into sub-structuring element like anytime algorithmic spatial filtering methods that already proposed in [9].

In this section, how to modify the anytime algorithmic morphological method by using anytime algorithm is explained. Structuring element is the probe in morphological operations like dilation, erosion, opening, and closing. In this proposed method, the structuring element is divided into sub-structuring element as shown in the Fig.4.9. Figure 4.9(a) is the 3x3 structuring element and (b) is the divided sub-structuring elements *B*$_1$ to *B*$_8$ of 3x3 structuring element.



(a)



(b)

Fig. 4.9: (a) 3x3 structuring element (b) its sub-structuring elements

These sub-structuring elements are used into morphological erosion operation in anytime algorithmic form and its algorithm is described as follows:

Anytime algorithmic morphological erosion

Algorithm 1: *Erosion by different patterns of sub-structuring elements*

Input – binary image A *and sub-structuring elements* B$_1$ *to* B$_8$
Output – eroded image

Step-1: Perform the erosion operation using the input image A with structuring element B_1, i.e., $A \ominus B_1$ by using the first one of different patterns of structuring element shown in Fig.4.9, then output the eroded image i.e., A_1

Step-2: Perform the erosion operation using the previous output image A_1 obtained from step-1 as input image with structuring element B_2, here the element used in B_2 is the added element only. Then output the eroded image.

Step-3: Repeat step-2 with corresponding structuring elements until the output eroded image A_7 as input image with structuring element B_8

Step-4: Output the final eroded image A_8.

Here are the correspondence operations and its output image A_1 to A_8 at each step.

$$A_1 = A \ominus B_1$$
$$A_2 = A_1 \ominus B_2$$
$$A_3 = A_2 \ominus B_3$$
$$A_4 = A_3 \ominus B_4$$
$$A_5 = A_4 \ominus B_5$$
$$A_6 = A_5 \ominus B_6$$
$$A_7 = A_6 \ominus B_7$$
$$A_8 = A_7 \ominus B_8$$

So, the final output of the combination of these operations can be evaluated in the mathematical form as follows:

$$A_8 = (A_7\,(A_6 \ldots (A_1(A \ominus B_1)) \ldots \ominus B_7) \ominus B_8) \tag{4.10}$$

Note: $\ominus$ = erosion symbol

The AA morphological type method as mentioned above, the following facts that are satisfied all of the properties of anytime algorithm. That is, the result can be defined exactly at every step, thus, it is satisfied the properties (1), (2), and (3). In addition, step 1 result is applied in the evaluation of step 2, then the step 2 result is applied in step 3 and so on, in order to obtain the result at every step that is satisfied by the property (4) and (5). Moreover, if the algorithm is stopped for e.g., at step–3, then the partial result of this task can obtain for this step and started again at current step, so, it is satisfied the properties (6) and (7). This method can apply to the other morphological image processing operations such as dilation, opening, and closing using the appropriate structuring elements under the condition that the processing time is restricted.

4.2.4 *AA Conditional Processing*

Some of the properties of anytime algorithm resemble the properties of iterative method. That is why, anytime algorithm could be applied to this condition type method. If the operation is performed by using many conditions such as morphological operations that have iterative process like thinning. By dividing the conditions into steps according to the properties of anytime algorithm, for e.g., if the method has n conditions i.e., $condition(1 \wedge 2 \wedge \ldots \wedge n)$, then n conditions are divided into many conditions, and the step 1 is performed by $condition(1)$, step 2 by $condition(2)$, and so on as shown in Fig. 4.10. Thus, n different results could be obtained in the intermediate processing time.

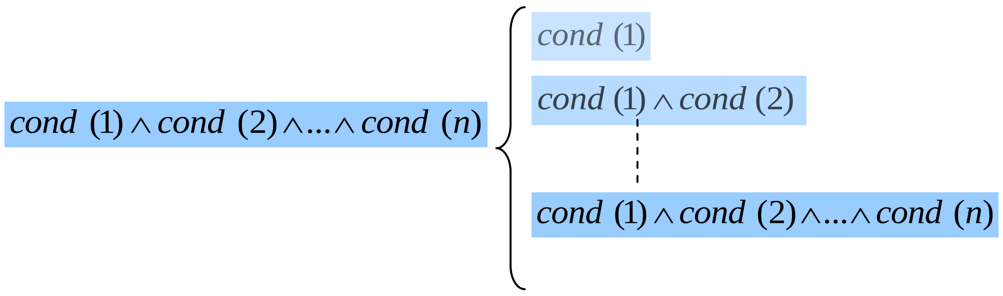

$cond\ (1) \wedge cond\ (2) \wedge \ldots \wedge cond\ (n)$
$cond\ (1)$
$cond\ (1) \wedge cond\ (2)$
$cond\ (1) \wedge cond\ (2) \wedge \ldots \wedge cond\ (n)$

Fig. 4.10: Divided sub-conditions

Even though there is no standard for the quality of image processing result, how to modify some of the image processing methods like filter type, gradient type, morphological type, and condition type into anytime algorithmic forms are described as mentioned above from the viewpoint of image processing's quality and/or processing time according to the concept and properties of anytime algorithm.

The AA condition type method as mentioned above, the following facts that are satisfied all of the properties of anytime algorithm. That is, the result can be defined exactly at every step from 1 to N, so, it is satisfied the properties (1), (2), and (3). The result at step 1 is applied in the evaluation of step 2, then the result at step 2 is applied in step 3 and so on until step N has been reached, in order to obtain the result at every step that is satisfied by the property (4) and (5). In addition, if the algorithm is stopped for e.g., at step–3, then the partial result can be obtained for this step and started again at current step, so, it is satisfied the properties (6) and (7). Therefore, anytime algorithm can be applied to the morphological operations by using AA condition type methods to other morphological operations such as thinning, dilation, erosion, opening, closing and pruning.

4.3 Experimental Results

4.3.1 AA Noise Reduction

For anytime noise reduction operation, the simple averaging operator is converted to anytime algorithmic form and the noise reduction operation is performed by many steps using these sub-masks satisfying the condition that the sum of all entry elements in the filter must be 1. An anytime algorithmic simple averaging process using 3 x 3 mask could be realized by weighting each pixel as shown in Fig. 4.11. In the averaging filter, there are 8 different results in the way of computing because it has at most 8 times summation. In each step, the quality of result gradually becomes better as processing time increases.

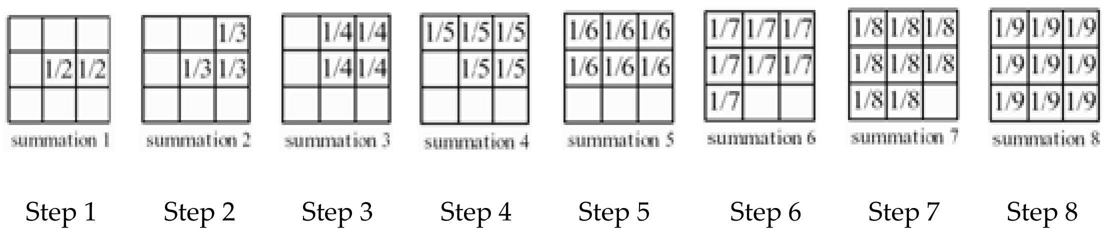


Fig. 4.11: Divided anytime simple averaging step

<u>Anytime simple averaging step</u>

Step 1: *sum* = f(x, y) + f(x+1, y)
Step 2: *sum* = f(x, y) + f(x+1, y) + f(x+1, y–1)
Step 3: *sum* = f(x, y) + f(x+1, y) + f(x+1, y–1) + f(x, y–1)
Step 4: *sum* = f(x, y) + f(x+1, y) + f(x+1, y–1) + f(x, y–1) + f(x–1, y–1)
Step 5: *sum* = f(x, y) + f(x+1, y) + f(x+1, y–1) + f(x, y–1) + f(x–1, y–1) + f(x–1, y)
Step 6: *sum* = f(x, y) + f(x+1, y) + f(x+1, y–1) + f(x, y–1) + f(x–1, y–1) + f(x–1, y) + f(x–1, y+1)
Step 7: *sum* = f(x, y) + f(x+1, y) + f(x+1, y–1) + f(x, y–1) + f(x–1, y–1) + f(x–1, y) + f(x–1, y+1)+f(x, y+1)
Step 8: *sum* = f(x, y) + f(x+1, y) + f(x+1, y–1) + f(x, y–1) + f(x–1, y–1) + f(x–1, y) + f(x–1, y+1)+f(x, y+1) + f(x+1, y+1)

By using the formula

$$O(x,y)=\sum_{n=1}^{8}\{\sum_{x=1}^{w-1}\sum_{y=1}^{h-1}\frac{sum}{n+1}\} \tag{4.11}$$

Where

O(x, y) = the output pixel value

n = step number

w = image width

h = image height

Experimental results

(a) Image 1 (b) Image 2 (c) Image 3

Fig. 4.12: Test images for anytime noise reduction

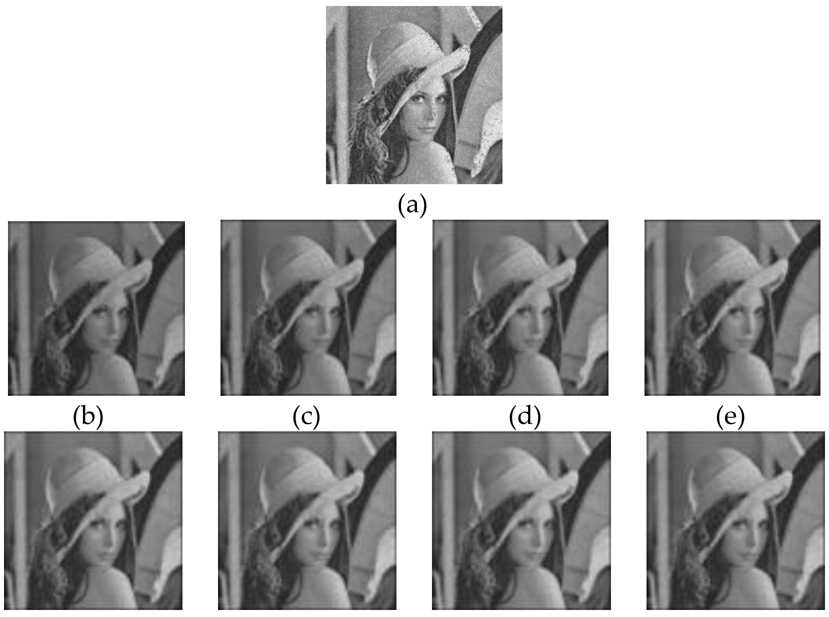


Fig. 4.13: (a) Standard Lena image with size 1024x1024 (b) ~ (i) smoothing result images by divided 8 sub-masks using 3x3 *Gaussian* filter

Figure 4.12 shows some of the test images for anytime noise reduction. First, the random noise is inputted to the standard input images, then, the noise reduction task is performed by the related sub-masks using 3x3 *Gaussian* filter step by step. Figure-4.13 (a) is the input standard *Lena* image and Fig. 4.13 (b) to (i) are the experimental results of the smoothed image performed by divided sub-masks. Here, the different intermediate results can be obtained by applying the sub-masks of 3x3 *mean* filter even though the noise reduction quality is evaluated by SNR.

Figure 4.14 displays the performance curve of step vs quality of result i.e., the probability of number of reduced noise. It expresses that the average performance curve of the quality of noise reduction result gradually improves as the processing time increases with related steps.

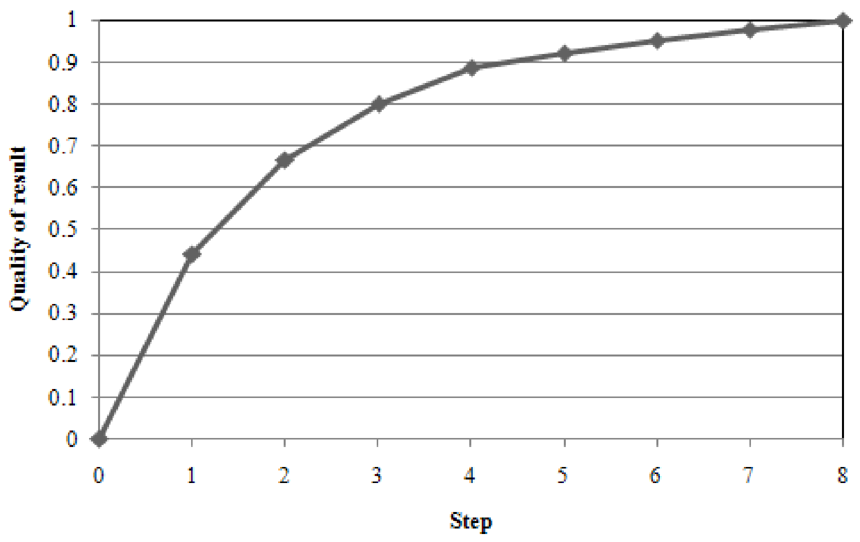


Fig.4.14: Performance curve of noise reduction by *Gaussian* filter

Another experiment is done by using the artificial noisy images as shown in Fig. 4.15 by putting quantization noise (uniform noise) as an example. Then, perform the anytime simple averaging algorithm to the input noisy image 2 shown in Fig. 4.12. Figure 4.16 represents the noise reduced images for image 2, here, the results of other images are not represented because of the lack of space. Figure 4.17 represents the noise reduced rate (improvement rate) by performance curve for AA noise reduction for the tested images. We can see that noise reduced rate gradually improve as processing time increases as shown in Fig. 4.17 for image 1, 2, and 3 respectively.

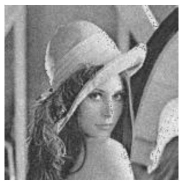
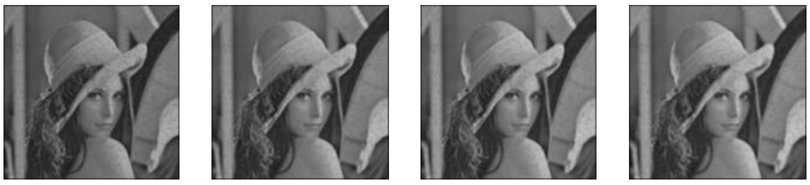
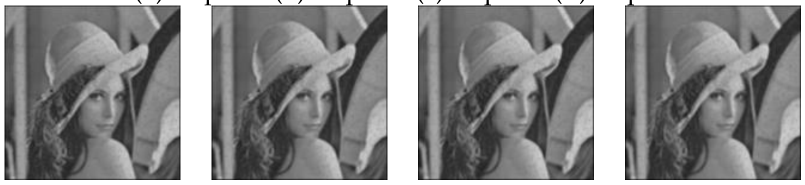


Fig. 4.15: Input noisy image



(a) step–1 (b) step–2 (c) step–3 (d) step–4



(e) step–5 (f) step–6 (g) step–7 (h) step–8

Fig. 4.16: Noise reduced images applying step 1 to 8 by using *Mean* filter

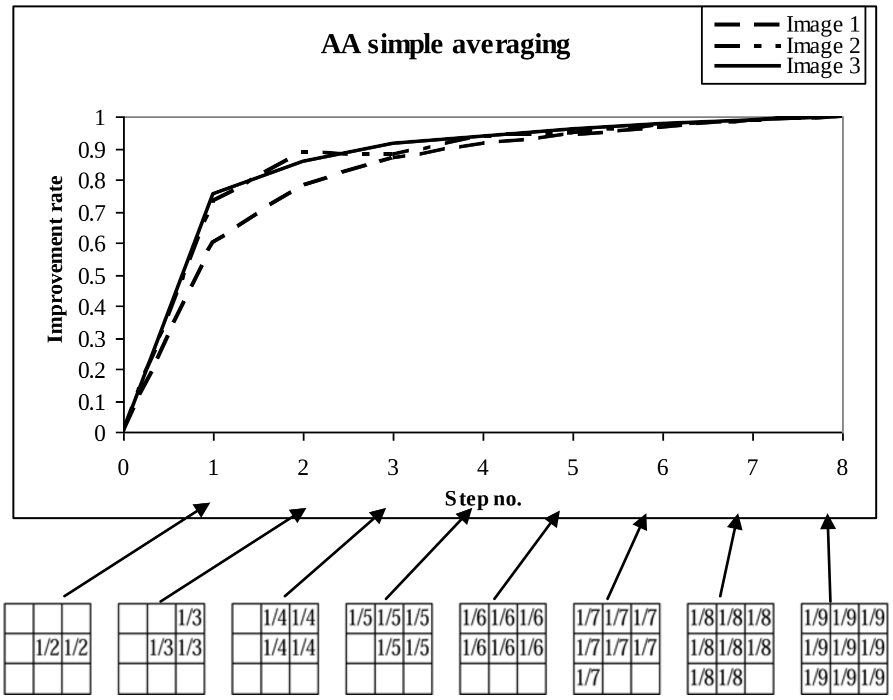


Fig. 4.17: Performance curves for AA noise reduction

4.3.2 AA Edge Detection

In the conventional edge detection operation, the *gradient* method is applied to measure the gradient of the image along two orthogonal axes, and it is the best for abrupt discontinuities. The useful operators are *Sobel, Canny, Prewitt*, and so on. In this AA edge detection, *Prewitt* operator is applied using two 3 x 3 masks (kernels), which are convolved with the original image to calculate the approximations of the derivatives - one for the horizontal changes, and one for the vertical.

Let

S = input source image's pixel

G_x and G_y = two sub images which at each point contain the horizontal and vertical derivative approximations

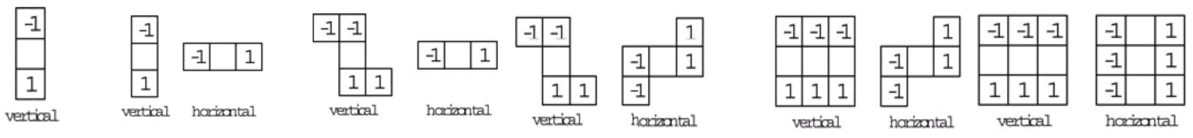
<u>Prewitt operator</u>

$$G_x = S * \begin{array}{|c|c|c|}\hline -1 & & 1 \\ \hline -1 & & 1 \\ \hline -1 & & 1 \\ \hline \end{array} \quad \text{and} \quad G_y = S * \begin{array}{|c|c|c|}\hline -1 & -1 & -1 \\ \hline & & \\ \hline 1 & 1 & 1 \\ \hline \end{array}$$

In anytime *gradient* method, the 6 different results are obtained by changing orientation or filter type horizontally and vertically alternately as shown in Fig. 4.18. At each point $G(x, y)$ in the image, the resulting gradient approximations can be computed step by step using the gradient formula

$$G = \sqrt{G_x^2 + G_y^2} \tag{4.12}$$

For the edge detection task, *Prewitt* 3x3 filter mask is divided into 6 different filters as shown in Fig. 4.18. Thus, it is performed by 6 steps by changing the filter type 1 to 6 and the 6 different results can be obtained at each step and the quality of result gradually becomes better as the processing time increases. So, the result can be defined exactly at each step, that is satisfying the AA properties (1) and (2), and also in these steps, the result of step 1 is used in step 2, the result of step 2 is used in step 3 and so on. Hence, the quality of result is improved increasingly at each step, so it is monotonicity that satisfies the AA property (3). Moreover, the result of step 2 is used for performing step 3 to obtain the result at every step, so the quality of previous result is connected at every step which is satisfying the properties (4) and (5). In case that, if the algorithm is stopped for e.g., at step 4 in this procedure, the partial or intermediate result can be obtained for this step and re-started again at current step, thus it is satisfying the AA properties (6) and (7).



Sub-filter 1 Sub-filter 2 Sub-filter 3 Sub-filter 4 Sub-filter 5 Sub-filter 6
Step 1 Step 2 Step 3 Step 4 Step 5 Step 6

Fig. 4.18: Anytime gradient method by *Prewitt* filter

Experimental results

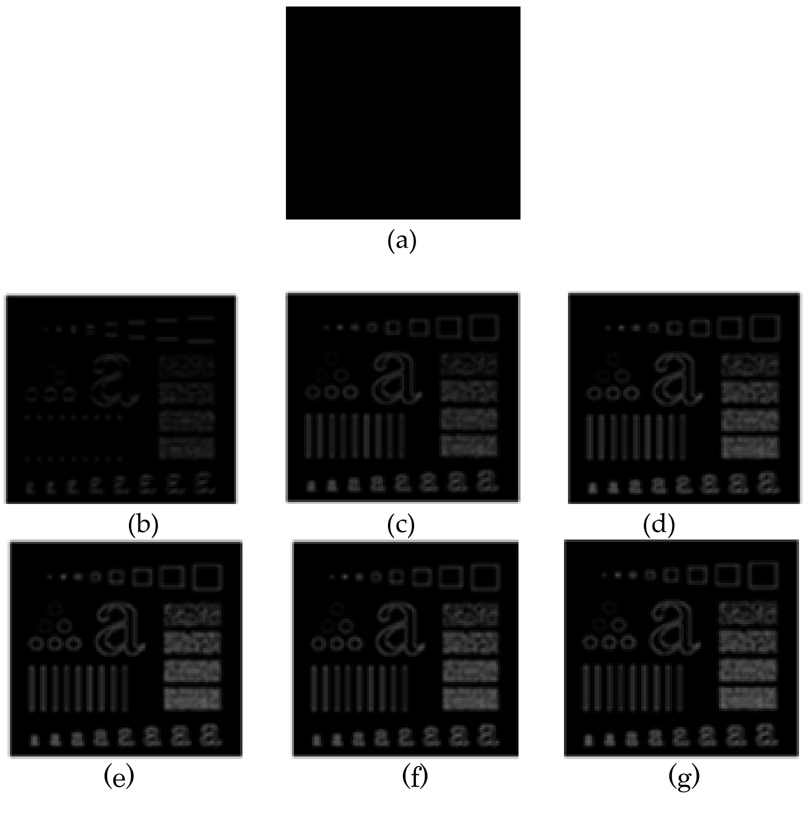


Fig. 4.19: (a) Original image with size 2048x2048 (b) ~ (g) edge detection results by divided sub-masks of *Sobel* filter

Figure 4.19 shows the experimental result of anytime algorithmic edge detection by divided sub-masks of *Sobel* 3x3 filter. The 6 different results are obtained with required processing time and related steps, and the quality of edge detection result at each step is increasingly improved as the processing time increased as shown in Fig. 4.20.

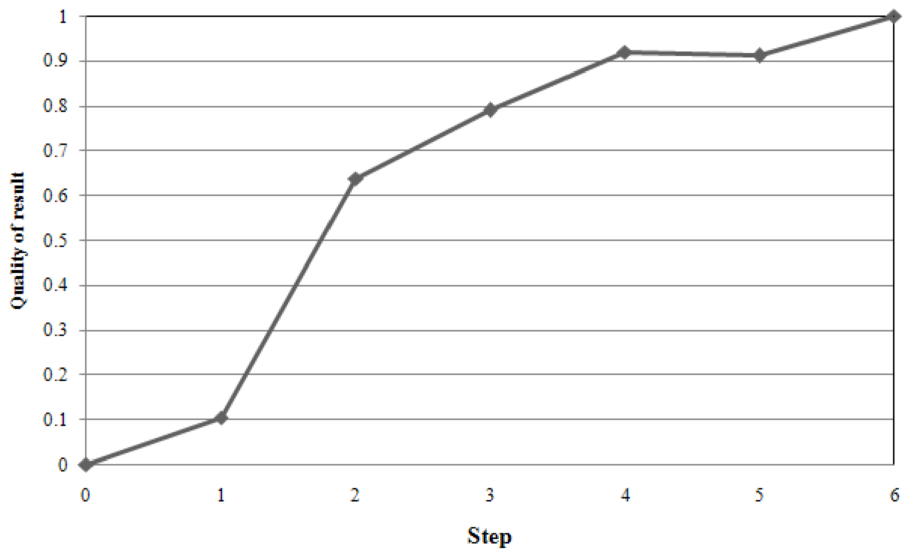
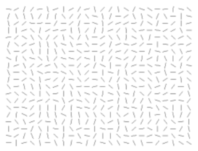


Fig. 4.20: Performance curve of anytime algorithmic edge detection by *Sobel* filter

In Fig. 4.21(a), it shows the input gray image that has different orientation of lines as tested image for edge detection and Fig. 4.21 (b) to (g) are the experimental result of anytime algorithmic edge detection by divided sub-masks of *Prewitt* 3x3 filter.



(a) Input gray image

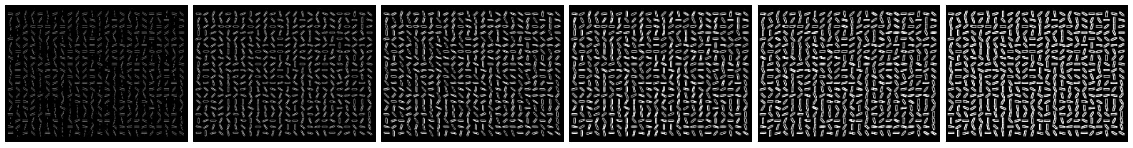
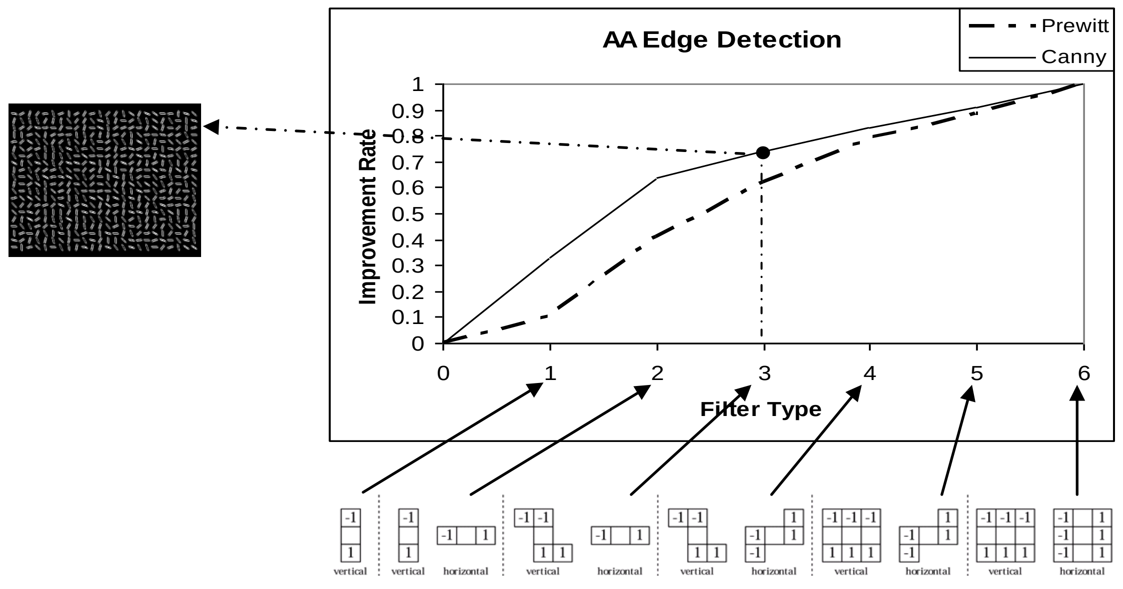


Fig. 4.21: (a) input gray image (b) ~ (g) Edge detected images by filter type 1 to 6 using Prewitt operator

As we can see in Fig. 4.21, the different 6 steps of result can be obtained at each step and its qualities gradually become better as processing time increases represented by using performance curve as shown in Fig. 4.22(b). Fig. 4.22(a) represents the midway result at step 3, and (b) is the performance cure of edge detection and it is satisfied the properties of anytime algorithm (6) and (7).



(a) (b)

Fig. 4.22: (a) Midway result at step 3 (b) Performance curve for edge detection

4.3.3 *AA Sharpening*

Experimental results

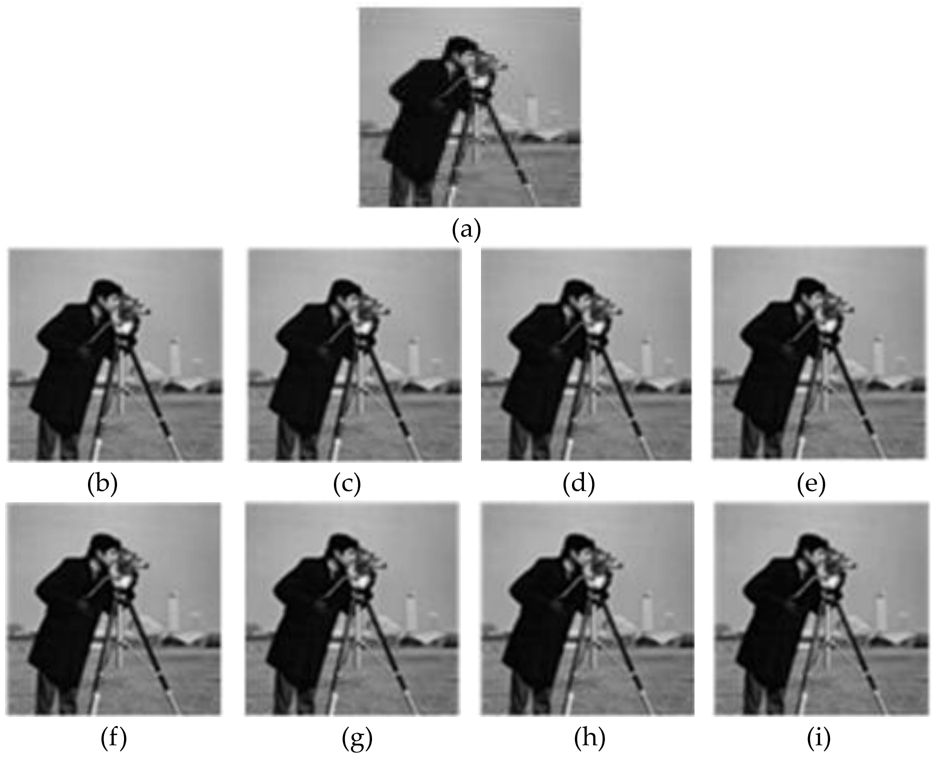


Fig. 4.23: (a) Original image with size 1024x1024 (b) ~ (i) sharpening result images by divided sub-masks of *basic hi-pass* filter

Figure 4.23 (a) is the input camera man image for AA sharpening and (b) to (i) show the experimental result of sharpening by divided sub-masks of *basic hi-pass* 3x3 filter. We can see that the 8 different results are obtained with required processing time and related steps. Thus, the quality of sharpened result at each step is increasingly improved as the processing time increased. Figure 4.24 shows the performance curve of sharpening result by *basic hi-pass* filter.

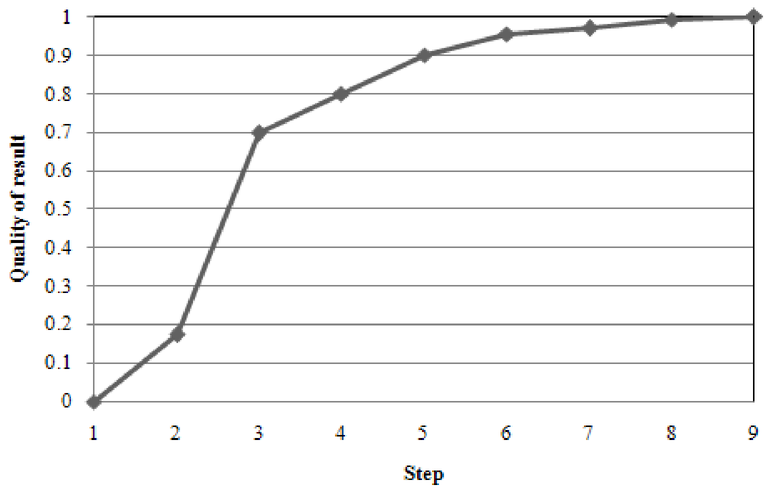


Fig.4.24: Performance curve of sharpening by *basic hi-pass* filter

4.3.4 AA thinning

Thinning is one of the morphological operations that have iterative task and performs on binary image. The objective is to reduce the connected pixel into one line character pixel. In a conventional thinning algorithm, there are many conditions to remove successive pixels by maintaining the object connectivity. An anytime algorithmic thinning could be realized by changing conditions with 5 steps and delete connected border pixels by maintaining the object connectivity at each step.

In AAIP, an anytime algorithmic thinning could be realized by changing the conditions as follows:

Condition number 1 : cond(A)
Condition number 2 : cond(A) and (B)
Condition number 3 : cond(A) and (B) and (C)
Condition number 4 : cond(A) and (B) and (C) and (D)
Condition number 5 : cond(A) and (B) and (C) and (D) and (E)

Here,

cond(A) : connectedness of the point is not 1
cond(B) : connectedness of the point is not 0
cond(C) : connectedness of the point is not 2
cond(D) : keeps connectivity
cond(E) : lines 2 pixels wide

This scheme outputs 5 steps of result whose qualities gradually become better when the processing time increases. Fig. 4.25 is the input binary image that has thin and elongated object and Fig. 4.26 (a) to (e) are the results thinned images at each step by changing the condition 1 to 5 step by step using *Hilditch's* method.

Experimental results

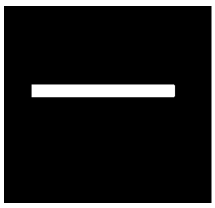


Fig. 4.25: Input binary image

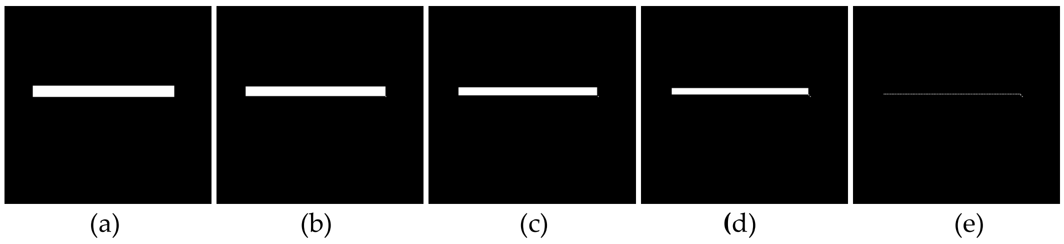


Fig. 4.26: Thinned image by *Hilditch's* method

Figure 4.27 displays the performance curve of AA thinning method for above input image and it represents the quality of result become better as increasingly at each step.

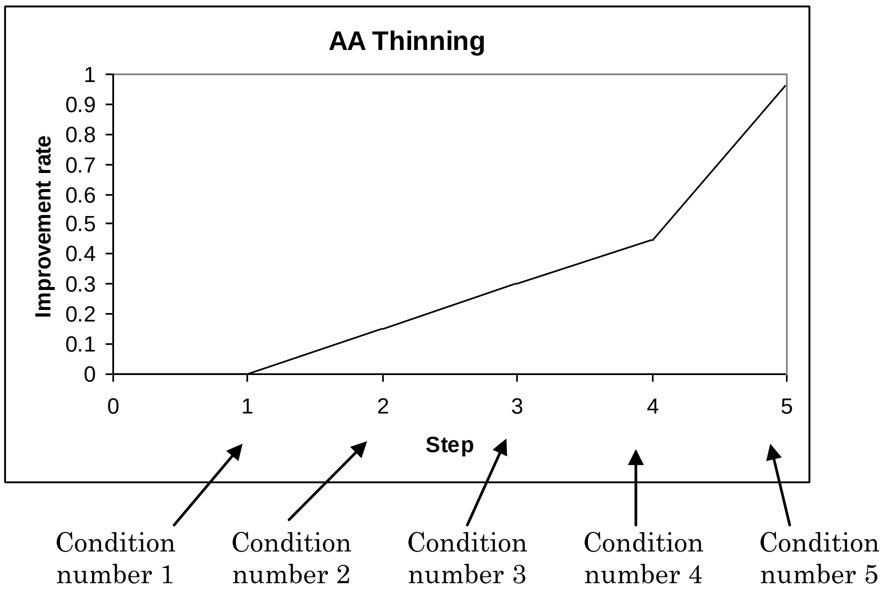


Fig. 4.27: Performance curve for thinning by *Hilditch's* method

4.3.5 AA Boundary Detection

In this sub-section, how to apply anytime algorithm to boundary extraction using anytime algorithmic erosion method is described.

Boundary extraction by anytime algorithmic erosion

Algorithm 2: *Anytime algorithmic boundary extraction*

Step-1: Input gray scale image and sub-structuring element B_1. Convert gray scale to binary image A.

Step-2: Perform the anytime algorithmic erosion operation (Algorithm 1) of input image by B_1 and store the result to temporary storage.

Step-3: Output the current eroded image A_1.

Step-4: Perform the operation $(A_1 \ominus B_1)^c$ i.e., the complement of $A_1 \ominus B$.

Step-5: Perform the boundary extraction operation $\beta_1(A)=A_1 \cap (A_1 \ominus B)^c$.

Step-6: Output the extracted the boundary points $\beta_1(A)$.

Step-7: Repeat step 4 to 6 by using sub-structuring elements B_2 to B_7, the eroded images A_2 to A_8 and the boundary extracted image $\beta_2(A_2)$ to $\beta_8(A_8)$.

Step-8: Output the final extracted boundary points.

So, the processing steps are considered as follows:

$$A_1 = A \ominus B_1$$
$$\beta(A_1) = A \cap (A \ominus B_1)^c = A \cap A_1^c$$
$$A_2 = A_1 \ominus B_2$$
$$\beta(A_2) = A \cap (A_1 \ominus B_2)^c = A \cap A_2^c$$
$$A_3 = A_2 \ominus B_3$$
$$\beta(A_3) = A \cap (A_2 \ominus B_3)^c = A \cap A_3^c$$
$$A_4 = A_3 \ominus B_4$$
$$\beta(A_4) = A \cap (A_3 \ominus B_4)^c = A \cap A_4^c$$
$$A_5 = A_4 \ominus B_5$$
$$\beta(A_5) = A \cap (A_4 \ominus B_5)^c = A \cap A_5^c$$
$$A_6 = A_5 \ominus B_6$$
$$\beta(A_6) = A \cap (A_5 \ominus B_6)^c = A \cap A_6^c$$
$$A_7 = A_6 \ominus B_7$$
$$\beta(A_7) = A \cap (A_6 \ominus B_7)^c = A \cap A_7^c$$
$$A_8 = A_7 \ominus B_8$$
$$\beta(A_8) = A \cap (A_7 \ominus B_8)^c = A \cap A_8^c$$

In general, the result or output of the boundary extraction can be performed by

$$\beta(A_i) = A \cap (A_{i-1} \ominus B_i)^c \qquad (4.13)$$

Experimental results

Erosion and boundary extraction with single iteration

First, smoothing operation is applied for the gray scale image as described in [9], then convert it to binary image, after that anytime algorithmic erosion is performed. Then boundary extraction by algorithm 2 is performed. Figure 4.28 expressed that the experimental results by the proposed method. Figure 4.28 (a) is the input image and (b) to (i), the first and third columns show that eroded images applied by sub-structuring elements B_1 to B_8 and Fig.3 (b)′ to (i)′, the second and fourth columns show that the boundary extracted results of corresponding images in (b) to (i). These experimental results are obtained by using the algorithm 2 which is repeatedly performed until the final approximation of the correct output is reached.

(a)

Fig. 4.28: (a) Input image (b) ~ (i) are eroded images applied by sub-structuring elements B_1 to B_8 and (b)′ ~ (i)′ are the boundary extracted results of corresponding images in (b) ~ (i)

Fig. 4.29: Some of tested images

The tested standard images as shown in Fig. 4.29 and its information are expressed as follows:

Gray scale images with pgm format and size 256x256

Standard images for the morphological operations are acquired from

the following link
http://www.imageprocessingplace.com/root_files_V3/image_databases.htm

4.4 Summary

This chapter provided the realization of digital image processing operations and methods in low, mid and high level from the viewpoint of anytime algorithm is described. The basic framework of the proposed method is expressed by figures. Then, the conventional image processing methods are categorized as filter type, gradient type, condition type, and morphological processing type. After that, how to modify these methods to AAIP methods are explained and presented anytime algorithmic erosion and boundary extraction algorithms according to the concept of anytime algorithm. The experimental results for noise reduction, edge detection, sharpening, thinning, and morphological operations show that the better intermediate result can be obtained with less processing time. The performance curves from the viewpoint of probability theory for the correctness and certainty show that the quality of result improves as the processing time increases. Some of test standard images are applied to the boundary extraction application by morphological erosion method. So, it is useful for the real-time image processing system under time constraint and the implementation of embedded systems application in real-time system.

References

[1] R.C. Gonzales, R.E. Woods: "Digital Image Processing", 2nd Edition, Prentice Hall, 2002.

[2] T. Dean, and M. Boddy : "An analysis of time dependent planning", Proceedings AAAI-88, St. Paul, Minnesota, AAAI, 49–54, (1988).

[3] S. Zilberstein and S. J. Russell In S. Natarajan (Ed.) : "Approximate Reasoning Using Anytime Algorithms", Imprecise and Approximate Computation, Kluwer Academic Publishers, (1995).

[4] J. Grass and S. Zilberstein In M. Pittarelli (Ed.) : "Anytime algorithm development tools", SIGART Bulletin Special Issue on Anytime Algorithms and Deliberation Scheduling, 7(2):20-27, (1996).

[5] S. Zilberstein Ph.D dissertation : "Operational Rationality through Compilation of Anytime Algorithm," Computer Science Division, University of California at Berkeley, (1993).

[6] E. A. Hansen and S. Zilberstein : "Monitoring anytime algorithms" SIGART Bulletin, 7(2), 1997.

[7] W.W.Kywe, D.Fujiwara, and K.Murakami : "Scheduling of Image Processing Using Anytime Algorithm for Real-time System", Proc. of the 18th International Conference on Pattern Recognition (ICPR2006), Vol.3, pp.1095-1098, Hong Kong/China, (Aug. 2006) IEEE Computer Society, 2006

[8] W.W.Kywe and K.Murakami : "Anytime Noise Reduction and Edge Detection Algorithms for Time-Restricted Image Processing System", Proc. of the 15th Japan-Korea Joint Workshop on Frontiers of Computer Vision (FCV 2009), pp.65-70, Andong/Korea, (Feb. 2009)

[9] W.W.Kywe and K.Murakami, "An Approach to Linear Spatial Filtering Method based on Anytime Algorithm for Real-time Image Processing", Journal of Computing Press, NY, USA, ISSN 2151-9617, Volume 4, Issue 12, pp.26-32, (Dec. 2012)

Chapter 5
Algorithms for Task Assignment and Scheduling

5.1 Introduction

In the hard real-time system, each task must be performed in order to extract the required information by its hard deadline. In the real-time scheduling system, most of the tasks are scheduled by the required time that is based on the given deadlines. If the combination of many tasks is required to work in the hard real-time, the problems of resource constraints such as processing time have been involved for the completion of tasks by their deadline.

The solutions to solve this kind of problems are by using hardware such as multi-processor, parallel processing and/or by using software such as algorithm, pipeline, and cache. From the algorithmic and/or software point of view, the guarantee of hard real-time deadline has been involved as a challenging problem.

The imprecise computation method, i.e., generalization of anytime algorithm, is one of the approximation methods to solve the trade-off problems between the computation time and the quality of result. Conventional imprecise scheduling methods solve this problem by discarding the optional sub-tasks i.e., the less importance tasks that could not meet their deadlines.

J.W.S. Liu et al. reviewed and proposed the scheduling algorithm by their imprecise computation method that provides the scheduling flexibility by trading off the quality of result to meet the computation deadlines [1]. W.K Shih et al. proposed the fast algorithm to minimize the maximum errors for the scheduling under timing constraints by the imprecise computation approach [2].

In this chapter, how to solve the above problems from the algorithmic point of view by the proposed imprecise scheduling in order to realize the sub-optimal overall processing result while minimizing the discarded optional sub-tasks is discussed.

The idea is by dividing the task into many sub-tasks performed by many steps. Then, the sub-tasks with the necessary steps for the acceptable result are divided into mandatory part and the left sub-tasks are in optional part according to the pre-defined conditions. Mandatory sub-tasks are performed by the mandatory steps and the optional sub-tasks are performed by the left steps. Thus, the schedule of the combination of sub-tasks with related processing time can be performed by the given parameter such as tact time and the required sub-task numbers of the mandatory and optional sub-tasks. So, the number of discarded optional sub-task can be reduced and overall result can be realized. In particular, how to schedule the image processing tasks in order to solve the time quality trade-off problem by the concept of anytime algorithm is expressed [3].

5.2 How to Schedule the Tasks by Imprecise Computation

Imprecise Computation Model

A system based on the imprecise computation method is called an imprecise task system. In the conventional imprecise computation method, there are two main parts called mandatory part *M* and optional part *O* which is used to refine the result of mandatory part. In this approach, the task is logically divided into *n* sub-tasks performed by *n* steps, here $n > 2$. The mandatory part has *k* sub-tasks called m_1, m_2, …, m_k performed by *k* mandatory steps and $1 \leq k \leq n$. In this part, m_1 is performed by step 1, m_2 is performed by step 1 and 2, …, and m_k is performed by step 1, 2, …, *k* respectively. Thus, the optional part will have (*n*-*k*) sub-tasks called o_{k+1}, o_{k+2}, …, o_n respectively and these are performed by the left (*n*–*k*) optional steps. In this part, o_{k+1} is performed by *k*+1 step, then o_{k+2} is performed by *k*+1 and *k*+2 steps etc. to refine the result of the mandatory sub-tasks.

Definition and Terminology

Let us consider the imprecise task system *T* that composed of a set of *N* tasks

$$T = \{T_i^{n_i}\}\ ,\ i = 1, 2, \ldots, N$$

In each task $T_i^{n_i}$, the subscript *i* denotes the task number and the superscript n_i denotes the total number of sub-task of each task *i*. Each task $T_i^{n_i}$ is characterized by the parameters which are the rational numbers:
r^i = ready time of task *i* at which $T_i^{n_i}$ becomes ready for execution
d^i = deadline of task *i* at which $T_i^{n_i}$ must be completed
$m_i^{n_i}$ = mandatory processing time of mandatory sub-task of task $T_i^{n_i}$ for the

execution of feasible result

Here, the mandatory part is performed for the feasible (acceptable) result and the required processing time of all sub-tasks in the mandatory part is combined into m_i which is the total mandatory processing time of sub-task m_i^j, $i = 1, 2, \ldots, N$.

i.e., $$m_i = \sum_{j=1}^{k} m_i^j \qquad (5.1)$$

$o_i^{n_i}$ = optional processing time of optional sub-task of task $T_i^{n_i}$ for the execution of sub-optimal result
o_i = the total optional processing time of sub-task $o_i^{n_i}$, $i = 1, 2, \ldots, N$.
t_i = the amount of processing time assigned to the task $T_i^{n_i}$
$p_i = m_i + o_i^{n_i}$ = the total processing time of each task $T_i^{n_i}$ to completion

Each sub-task has the related processing time which is less than the processing time of the whole task. If the time is restricted, the schedule can be obtained with the combination of the suitable sub-tasks with the minimum processing time. Thus, the number of discarded optional sub-tasks would be reduced and the intermediate sub-optimal overall result of all tasks could be realized even though the quality of result is not perfect. The current step computation is used the result of the previous step.

Each sub-task can be stopped at anytime with the approximated result and then it can be resumed with the minimal overhead time. For the processing of all tasks, the returned result of the first task is inputted to the next task and so on. Thus, the feasible schedule can be realized by checking the pre-defined condition according to the current execution time. Furthermore, the sub-optimal overall result can be realized by checking the condition that the current overall result is greater than or equal to the pre-defined overall result which is evaluated by the minimum required number of mandatory sub-task. So, the number of discarded optional tasks can be minimized while approximating the sub-optimal overall result.

Each task can be partially performed by the required sub-tasks number with the corresponding divided steps or the percentage of the required sub-tasks. Thus, the different schedules are obtained with their related processing time as shown in Fig. 5.1 which is the conceptual scheduling model of the proposed scheduling scheme as an example.

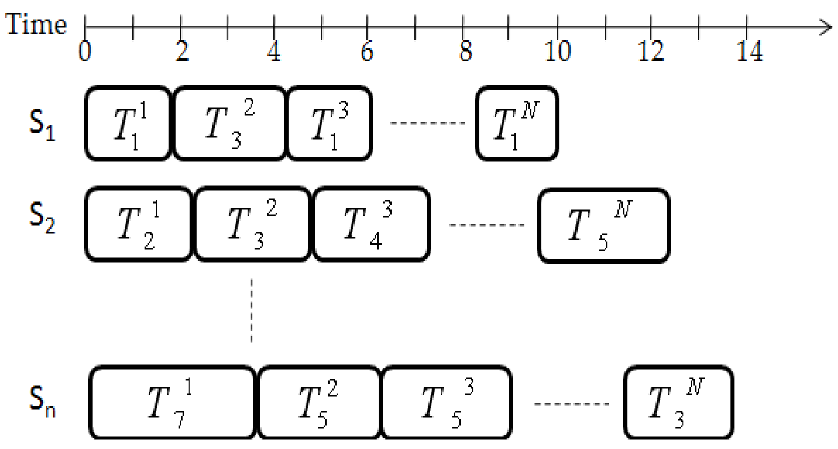


Fig. 5.1: Conceptual scheduling model

Table 5.1 and table 5.2 are the example data use for the scheduling in the conventional and proposed methods used for the imprecise scheduling.

Table 5.1: Example data by the conventional scheduling method

	r^i	d^i	p_i	m_i	o_i
Task1	3	10	12	4	8
Task2	0	13	9	2	7
Task3	2	18	10	3	7
Task4	6	24	13	5	8

Table 5.2: Example data by the proposed scheduling method

	r^i	d^i	p_i	m_i	o^i			
					o^i_1	o^i_2	o^i_3	o^i_4
Task1	3	10	12	4	3	4	6	8
Task2	0	13	9	2	2	3	4	7
Task3	2	18	10	3	2	4	7	
Task4	6	24	13	5	3	5	8	

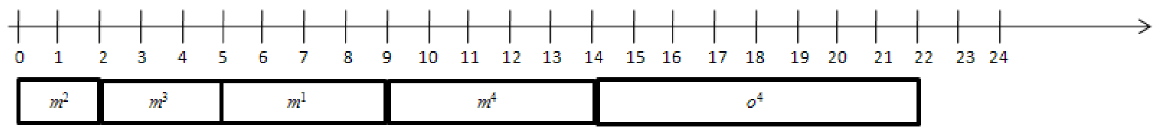


Fig. 5.2: Schedule by the conventional method

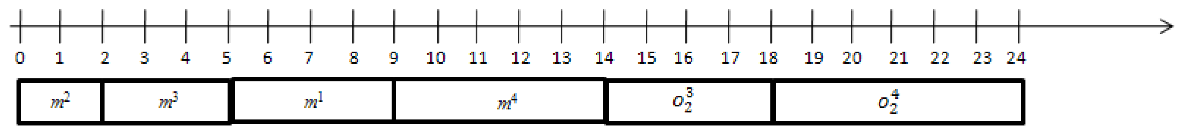


Fig. 5.3: Schedule by the proposed method

Figures 5.2 and 5.3 display the illustrative example schedules in the conventional and the proposed methods. By comparing these schedules, the proposed method is efficient than the conventional and reduce the idle processing time.

5.2.1 Algorithms for Scheduling under Time Constraint

Algorithm 1

How to realize the overall processing result while reducing the discarded optional sub-tasks?

Step – 1: Search the mandatory task with minimum ready time r^i and check its processing time m_i which is in its ready time and deadline, i.e., $r^i \leq m_i \leq d^i$. If it is, assign the processing time, $t_i = t_i + m_i$, and current processing time: $t_c = t_c + t_i$

Step – 2: If $t_c \leq d^i$, calculate the difference of the time interval $t_d = d^i - t_c$ then search the optional sub-task with minimum $o_i^{n_i}$ of current task which is satisfied the condition that $t_c \leq o_i^{n_i} \leq d^i$. If it is, $t_i = t_i + o_i^{n_i}$, and $t_c = t_c + t_i$

Step – 3: Repeat step 1 to 2 until feasible schedule is obtained i.e., all mandatory sub-tasks and some of optional sub-tasks are assigned.

Step – 4: Check the assigned sub-task numbers $c_i \in s_i$ of each task $T_i^{n_i}$ which is satisfied the pre-defined condition $k_i \leq c_i \leq s_i$. If it is satisfied, go to step 5, else go to step 1 for rescheduling in order to obtain the sub-optimal overall result.

Step – 5: Perform the operation of tasks assigned by the above schedule.

Step – 6: Calculate the weighted value: $w_i = \frac{c_i}{k_i}$ for determining the probability of result R^i of each task $T_i^{n_i}$.

Step – 7: Calculate the probability of sub-optimal overall processing result O by

$$O = \frac{\sum_{i=1}^{N} w_i R^i}{\sum_{i=1}^{N} R^i}, w_i \leq 1, i = 1, 2, \ldots, N. \tag{5.2}$$

Algorithm 2

How to distribute the required processing time of each task?
Earliest Deadline First algorithm (EDF) [without priority task]

Step – 1: Input tact time (pre-run time) *TT*, number of tasks T_i, required processing time t_i and the deadline D_i of each task performed by steps.

Step – 2: Search the minimum deadline D_i or (the earliest deadline first) of each sub-task of task T_i and check the condition that t_i is less than or equal to TT, then search the next minimum deadline which is ≤ (TT – T_1), and let it be T_2 until no more execution time left or insufficient execution time for any deadline left i.e., $0 \leq T_N \leq (T - (T_1 + T_2 + \ldots + T_{N-1})$

Step – 3: If all or some of the tasks assigned by the related processing time, then first schedule i.e., S_1 is obtained.

Step – 4: If the system received the stop signal from the checking point, then output the intermediate result.

Step – 5: Repeat step 2 to 4 until no more processing time left or all of the tasks are assigned by related processing time.

Algorithm 3

For the task assignment and scheduling by the imprecise computation method

Step 1: The required steps of each task is manually defined for the data consistency.

Generate the random numbers for

- the execution time of each sub-task of N independent tasks
- the outcomes or results (i.e., the values of discrete random variables X's) at each step of N tasks within the particular ranges

Step 2: Calculate the PDF (Probability Distribution Function) $P_i^{x_i}$ of the outcomes of each sub-task

i.e., $P_i^{x_i} = \dfrac{\text{no.of outcomes of each sub-task } x_i \text{ of task } i}{\text{total no.of outcomes of task } i}$, $i = 1, 2, \ldots, N$

Step 3: Calculate the probability of the overall outcomes R of the required sub-task number of mandatory part which is manually input as parameters r_i, the left sub-tasks are assumed to the optional sub-tasks.

i.e., $R = \dfrac{1}{N}\sum_{i=1}^{N} P_i^{r_i}$, $1 \leq r_i \leq x_i$

Step 4: Check the conditions that the current step number of each sub-task ≥ the required step numbers of each mandatory sub-task, for the schedulability, and the current probability of overall outcomes ≥ the previous maximum probability, for the sub-optimal overall outcomes.

Step 5: Check the condition for the probability of success by Monte-Carlo simulation method i.e., the current probability of the overall outcomes ≥ R, then count the no. of success and do the statements.

Step 6: Calculate the probability of success by using the equation:

$$\text{The probability of success at each trial run} = \frac{\text{no. of success at each trial run}}{\text{total no. of schedule at each trial run}}$$

Step 7: Repeat step 1 to 6 for *n* times.

Step 8: The possibility of the proposed method is calculated by the equation:

$$\text{The average probability of success at each trial run k} = \sum_{k=1}^{n} \frac{\text{the probability of success at trial run k}}{\text{n}}$$

Algorithm 4

Estimation of the overall processing result for the boundary detection application

(1) Determine the condition that minimum required step number of task 1 (noise reduction), task 2 (edge detection), task 3 (thinning), and task 4 (boundary detection) and let them be k_1, k_2, k_3, and k_4 for e.g., $k_1 = 1$, $k_2 = 3$, and $k_3 = 3$, $k_4 = 2$.

(2) Select the executable function f_1 i.e., convolution function for noise reduction by arithmetic mean operator with input value x i.e., input image and the step no. $c_1 = 2$ of task 1, which is satisfied the condition that $(c_1=2) \geq (k_1=1)$.

(3) Execute the function $f_1(x, c_1)$ and let the result be y_1, i.e., the amount of reduced noise expressed by the noise cleaned image.

(4) Select the next executable function f_2 i.e., derivative function for edge detection by gradient method (Prewitt operator) with input value y_1 (the result of the previous function) and the step no. $c_2 = 3$ of task 2 which is satisfied the condition that $(c_2 = 3) \geq (k_2 = 3)$.

(5) Execute the function $f_2(y_1, c_2)$ and let the result be y_2 i.e., the detected edge points.

(6) Select the next executable function f_3 i.e., thinning function by Hiltdich method with input value y_2 (the result of the previous function) and the step no. $c_3 = 4$ of task 3 which is satisfied the condition that $(c_3 = 4) \geq (k_3 = 3)$.

(7) Execute the function $f_3(y_2, c_3)$ and let the result be y_3 i.e., the thinned image.

(8) Select the next executable function f_4 i.e., boundary detection function by chain coding method with input value y_3 (the result of the previous function) and the step no. $c_4 = 3$ of task 4 which is satisfied the condition that $(c_4 = 3) \geq (k_4 = 2)$.

(9) Execute the function $f_4(y_3, c_4)$ and let the result be y_4 i.e., the detected boundary points.

(10) Display the overall processing result: $y = f_4 \circ f_3 \circ f_2 \circ f_1 \circ x$

(11) Calculate the overall processing result.

5.2.2 *Experiments and Results*

(1) Experiment for the task assignment and scheduling by the imprecise computation method by algorithm 1

The experiment is done by 100 trials run by simulation using *Monte-Carlo* method for the realization of the sub-optimal overall result. Here, I assume that the schedule is already obtained by the proposed algorithm step 1 to 4. Each task $T_i^{n_i}$ has the assigned sub-task c_i, the corresponding output of sub-optimal result R^i. The known data such as the processing time (ready time, deadline, mandatory processing time, optional processing time), and the required steps of each task and their related result are generated by the random number generator. Here, 4 tasks, i.e., task 1, 2, 3, and 4 are considered, and these tasks are performed by the steps 8, 8, 7 and, 7 respectively and the required steps of the mandatory sub-task of these tasks are assigned to 4, 4, 4, 4 respectively.

i.e., $c_1 \geq 4, c_2 \geq 4, c_3 \geq 4,$ and $c_4 \geq 4$

Step number

Task 1	1, 2, 3, 4, 5, 6, 7, 8
Task 2	1, 2, 3, 4, 5, 6, 7, 8
Task 3	1, 2, 3, 4, 5, 6, 7
Task 4	1, 2, 3, 4, 5, 6, 7

Thus, we have

$$\binom{5}{1}\binom{5}{1}\binom{4}{1}\binom{4}{1} = \frac{5!}{1!(5-1)!} \times \frac{5!}{1!(5-1)!} \times \frac{4!}{1!(4-1)!} \times \frac{4!}{1!(4-1)!} = 5 \times 5 \times 4 \times 4 = 400$$

schedules of combinations of sub-tasks for the acceptable result. Figure 5.4 represents one of the performance curves of the probability of overall result by processing time. Among these 400 schedules, the different 15 sub-optimal schedules by tact time is obtained as shown in Table 5.3 and Fig. 5.5 due to related processing time of each sub-task and time distribution according to the proposed algorithm as one of examples.

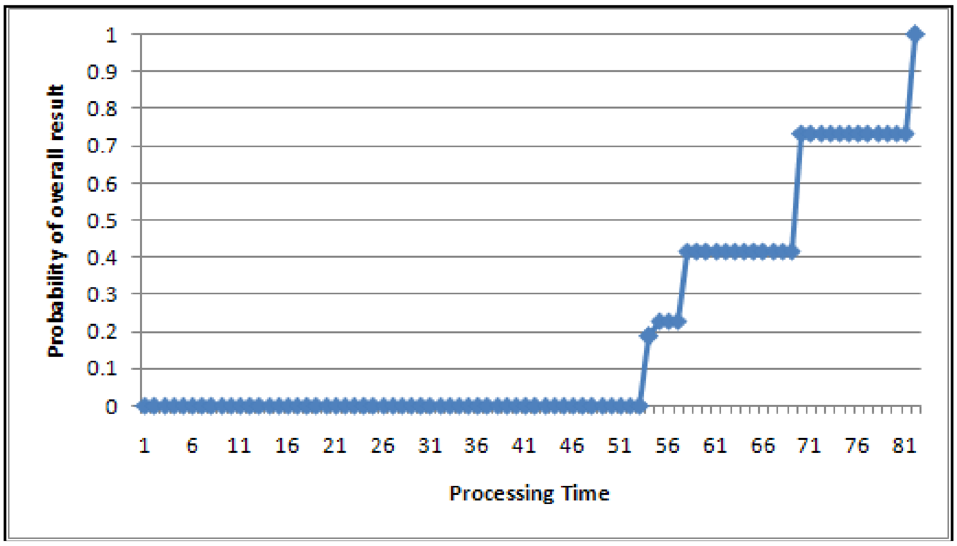
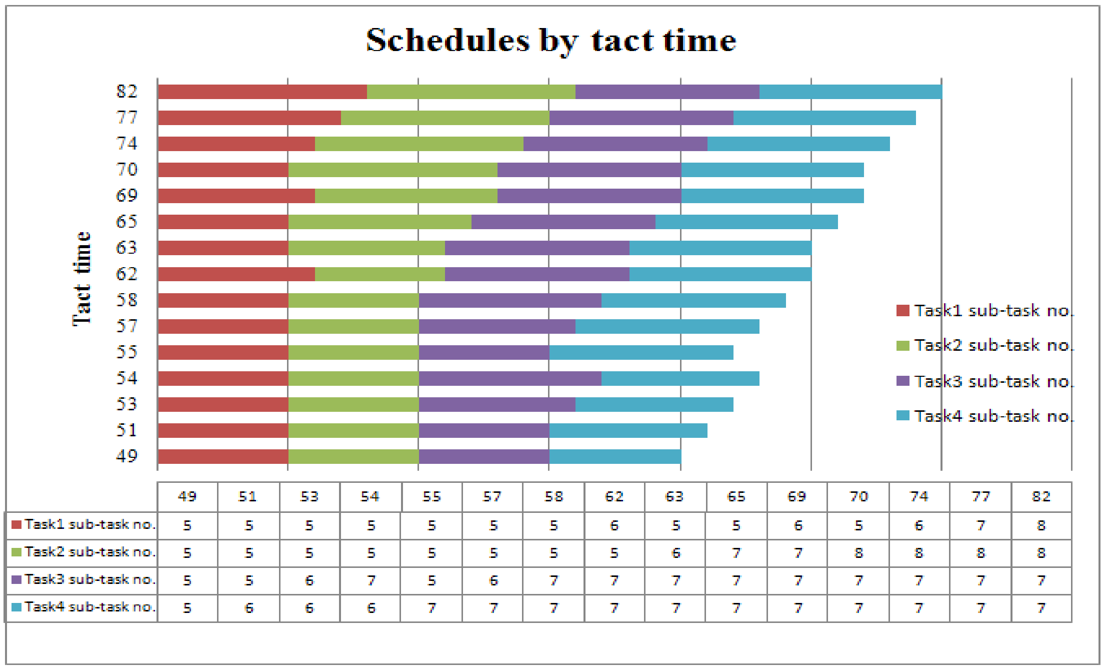


Fig. 5.4: Probability of overall result by processing time

Table 5.3: Scheduling result of sub-task number by related processing time and sub-optimal overall result

Processing time	Task1 sub-task no.	Task2 sub-task no.	Task3 sub-task no.	Task4 sub-task no.	Sub-optimal overall result
49	5	5	5	5	0.415803
51	5	5	5	6	0.415803
53	5	5	6	6	0.415803
54	5	5	7	6	0.415803
55	5	5	5	7	0.415803
57	5	5	6	7	0.415803
58	5	5	7	7	0.415803
62	6	5	7	7	0.735854
63	5	6	7	7	0.735854
69	6	7	7	7	0.735854
70	5	8	7	7	0.735854
77	7	8	7	7	0.735854
82	8	8	7	7	1



	49	51	53	54	55	57	58	62	63	65	69	70	74	77	82
🟥 Task1 sub-task no.	5	5	5	5	5	5	5	6	5	5	6	5	6	7	8
🟩 Task2 sub-task no.	5	5	5	5	5	5	5	5	6	7	7	8	8	8	8
🟪 Task3 sub-task no.	5	5	6	7	5	6	7	7	7	7	7	7	7	7	7
🟦 Task4 sub-task no.	5	6	6	6	7	7	7	7	7	7	7	7	7	7	7

Fig. 5.5: Different sub-optimal schedules by tact time

Figure 5.6 expresses the quality of sub-optimal overall processing result which gradually improves the processing time increases even though the result is not perfect. The average percentage of sub-optimal overall processing result is 72% and the average processing time is 89.1089 ms upon the 100 times trial run. The time complexity of the proposed method is $O(n)$.

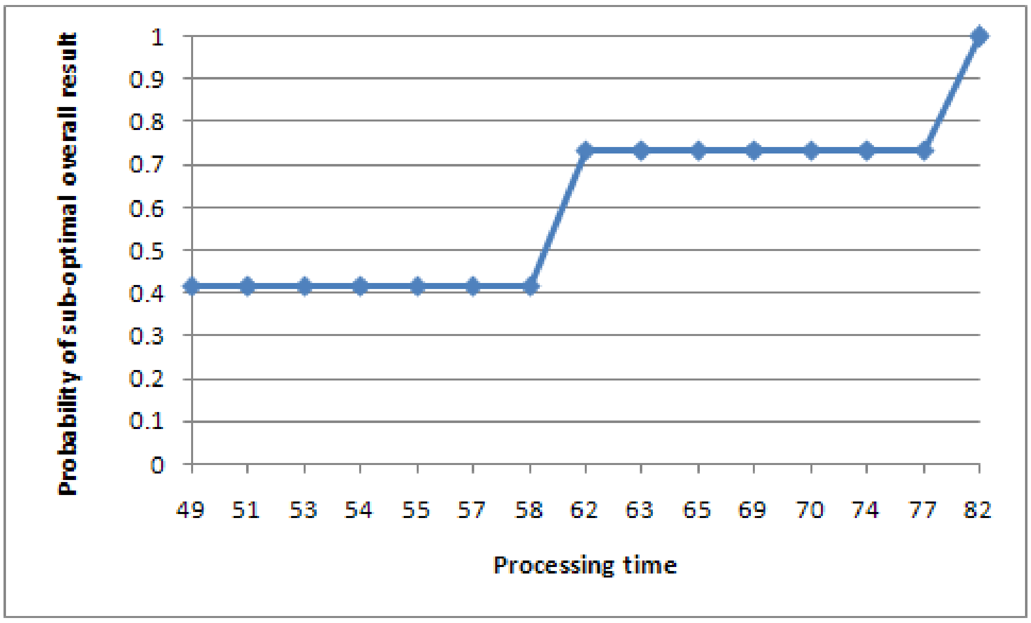


Fig. 5.6: Probability of sub-optimal overall result by processing time

(3) Experiment for the distribution of the required processing time of each task by algorithm 2

Experiment by simulation

The number of tasks is 4 and the number of corresponding sub-tasks of task 1, 2, 3, and 4 are 8, 6, 6, and 5 respectively. Pre run-time for all tasks is 10 and the deadline of sub-tasks for each task is shown in the Table 5.4.

Table 5.4: Simulation experiment of the proposed imprecise task system

Task	**Ready time** r_i	**Deadline** d_i	$t_i = m_i + o_i$	**Mandatory part** m_i				**Optional part** o_i			
				sub-task no.				sub-task no.			
				1	2	3	4	5	6	7	8
Task 1	0	5	5	0.5	1	1.5	2	3	4	4.5	5
Task 2	2	3	3	0.4	0.6	1	2	2.7	3		
Task 3	4	5	5	0.9	1.7	2.1	3	3.8	5		
Task 4	10	4	4	1	1.5	2	3	4			

The utilization of the processing time of each sub-task at each schedule will be

$$\text{utilization of processing time} = \sum_{i=1}^{N} \frac{\text{assigned time of each sub} - \text{task of task } i}{\text{deadline time of each sub} - \text{task of task } i} \qquad (5.6)$$

Table 5.5: An imprecise task system

Task	**Ready time** r_i	**Deadline** d_i	$t_i = m_i + o_i$	**Mandatory** m_i	**Optional** o_i
Task 1	0	15	3	2	1
Task 2	2	8	5	3	2
Task 3	4	10	5	2	3
Task 4	5	13	3	1	2

Table 5.6: A weighted imprecise task system

Task	**Ready time** r_i	**Deadline** d_i	$t_i = m_i + o_i$	**Mandatory** m_i	**Optional** o_i
Task 1	0	14	5	2	3
Task 2	2	7	3	1	2
Task 3	4	9	5	3	2
Task 4	10	15	4	2	2

Table 5.7: Expected experimental result

Schedule	Time (ms)	Assignment of tasks by its sub-task and distributed processing time								Idle time (ms)
		Task 1		Task 2		---		Task *N*		
		Sub-task #	Time	Sub-task #	Time	---	---	Sub-task #	Time	
S1	T1	0	0	1	t_1^2			1	t_1^N	I1
S2	T2	1	t_1^1	1, 2	t_2^2			1, 2, 3	t_3^N	I2
S3	T3	1, 2	t_2^1	1	t_1^2			1, 2	t_2^N	I3
	| | |									| | |
Sk	*Tk*	1, 2, …, k	t_k^1	1, 2, 3	t_3^2			1, 2, …, *kN*	t_{kN}^N	IK
	| | |									| | |
SN	TN Pre run-time	1, 2, …, n1	t_{n1}^1	1, 2, …, n2	t_{n2}^1			1, 2, …, *nN*	t_{nN}^N	In

Let

q(t_i) → the quality of result at processing time t_i

$\epsilon_i = E_i(t_k - t_i)$ is the error in the result produced by T_i

$E_i\,(t_i)$ gives the error of the task T_i as a function of t_i.

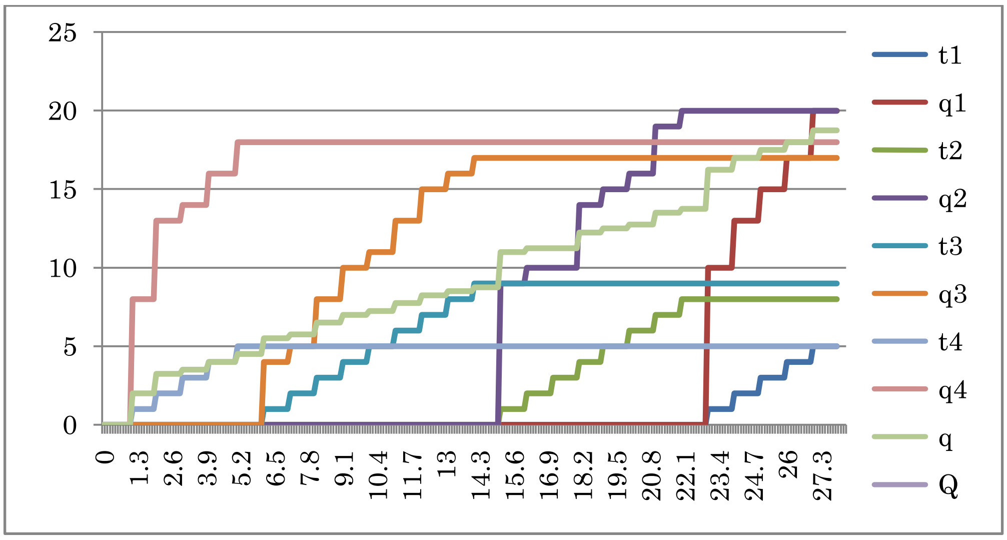


Fig. 5.7: Schedules by distributed time

(3) Experiment for the task assignment and scheduling by the imprecise computation method by algorithm 3

Objective

- Minimization of the discarded optional sub-tasks
- Realization of the overall outcomes of combination of many tasks

Experiment by simulation

The experiment is done by Monte-Carlo simulation method for 4 independent tasks as a sample.

- ◆ The required steps of each task is manually defined by 10, 15, 18, and 16 for the data consistency.
- ◆ The values of discrete random variables X's as the value of overall outcomes of each sub-task are generated within the range [100, 1000], and the execution times of each sub-task of 4 independent tasks are generated within the range [1, 5] in ms.
- ◆ The required steps number of each sub-task as parameters for the sub-optimal overall outcomes are 5, 6, 7, and 4 respectively. So, the left optional sub-tasks of 4 tasks are 5, 9, 11, and 12 respectively. The probabilities of required mandatory parts are $P_1^5, P_2^6, P_3^7, and\, P_4^4$ respectively.
- ◆ $R = \frac{1}{4}(P_1^5 + P_2^6 + P_3^7 + P_4^4)$
- ◆ 100 times trial runs

$$\begin{array}{l}\text{The average probability of} \\ \text{success at each trial run } k\end{array} = \sum_{k=1}^{100} \frac{\text{the probability of success at trial run } k}{100}$$

and, it is 0.847509, so the possibility of success rate of the proposed method is 84.75%.

Experimental results

Table 5.8 and Fig. 5.8 show the experimental results of the probability of success for 10 trial runs.

Table 5.8: The probability of success at each trial run and their execution time

Trial run no.	Prob. of success	No. of success	Total no. of schedules	Total execution time	Mean
1	0.886364	78	88	183	0.886364
2	0.852273	75	88	378	0.869318
3	0.860465	74	86	548	0.866367
4	0.911111	82	90	724	0.877553
5	0.463158	44	95	884	0.794674
6	0.911504	103	113	1054	0.814146
7	0.895833	86	96	1229	0.825815
8	0.873874	97	111	1413	0.831823
9	0.882353	90	102	1589	0.837437
10	0.851351	63	74	1758	0.838829

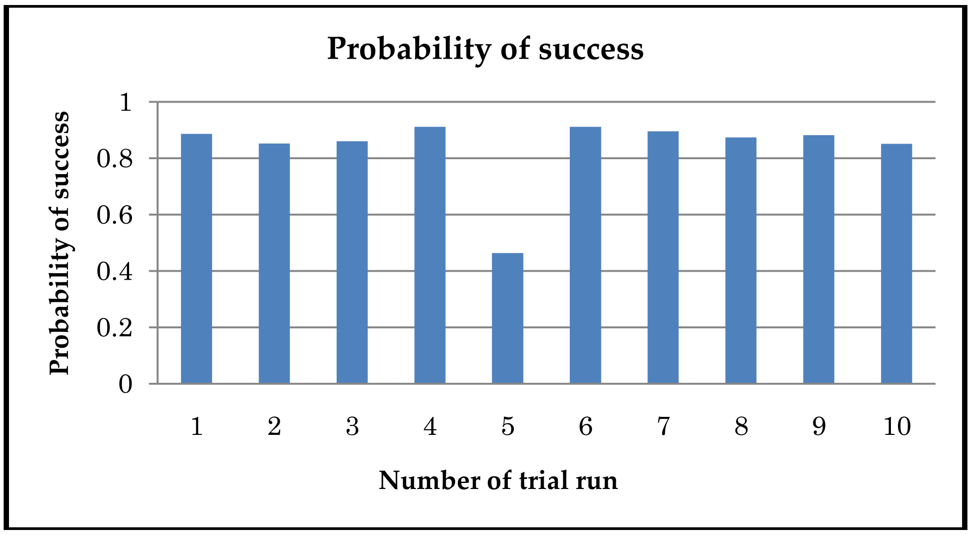


Fig. 5.8: The probability of success by graph

The schedule of each sub-task by the required step numbers for the sub-optimal overall outcomes for the first trial run is shown in Table 5.9 and Fig. 5.9.

Table 5.9: The different sub-optimal schedules by the required sub-task number of each task with their related execution time

Trial run no.	Exe. Time	Optimum time1	Sub-task no. of task1	Optimum time2	Sub-task no. of task2	Optimum time3	Sub-task no. of task3	Optimum time4	Sub-task no. of task4
1	87.1	27	9	26	9	20	7	14	4
1	88.1	29	10	17	6	28	10	14	4
1	89.1	29	10	26	9	20	7	14	4
1	91.1	27	9	30	10	20	7	14	4
1	92.1	29	10	26	9	23	8	14	4
1	93.1	29	10	30	10	20	7	14	4
1	95.1	27	9	26	9	28	10	14	4
1	97.1	29	10	26	9	28	10	14	4
1	99.1	27	9	30	10	28	10	14	4
1	101.1	29	10	30	10	28	10	14	4
1	104.1	29	10	33	11	28	10	14	4
1	106.1	29	10	43	14	20	7	14	4
1	108.1	27	9	39	13	28	10	14	4
1	109.1	29	10	46	15	20	7	14	4
1	110.1	29	10	39	13	28	10	14	4
1	112.1	29	10	17	6	52	18	14	4
1	114.1	29	10	43	14	28	10	14	4
1	117.1	29	10	46	15	28	10	14	4
1	119.1	27	9	26	9	52	18	14	4
1	121.1	29	10	26	9	52	18	14	4
1	123.1	27	9	30	10	52	18	14	4
1	125.1	29	10	30	10	52	18	14	4
1	128.1	29	10	33	11	52	18	14	4
1	130.1	29	10	30	10	52	18	19	5
1	132.1	27	9	39	13	52	18	14	4
1	134.1	29	10	39	13	52	18	14	4
1	136.1	27	9	43	14	52	18	14	4
1	138.1	29	10	43	14	52	18	14	4
1	141.1	29	10	46	15	52	18	14	4
1	143.1	29	10	43	14	52	18	19	5
1	146.1	29	10	46	15	52	18	19	5
1	150.1	29	10	46	15	52	18	23	6
1	151.1	29	10	43	14	28	10	51	14
1	152.1	29	10	39	13	28	10	56	16
1	153.1	29	10	30	10	52	18	42	11
1	154.1	29	10	43	14	28	10	54	15
1	155.1	29	10	26	9	52	18	48	13
1	156.1	29	10	43	14	28	10	56	16
1	157.1	29	10	46	15	28	10	54	15
1	158.1	29	10	26	9	52	18	51	14
1	159.1	29	10	46	15	28	10	56	16

1	160.1	29	10	46	15	52	18	33	8
1	161.1	29	10	26	9	52	18	54	15
1	162.1	29	10	30	10	52	18	51	14
1	163.1	29	10	26	9	52	18	56	16
1	165.1	29	10	30	10	52	18	54	15
1	166.1	29	10	43	14	52	18	42	11
1	167.1	29	10	30	10	52	18	56	16
1	169.1	29	10	46	15	52	18	42	11
1	170.1	29	10	33	11	52	18	56	16
1	171.1	29	10	39	13	52	18	51	14
1	172.1	29	10	43	14	52	18	48	13
1	173.1	27	9	43	14	52	18	51	14
1	174.1	29	10	39	13	52	18	54	15
1	175.1	29	10	43	14	52	18	51	14
1	176.1	29	10	39	13	52	18	56	16
1	178.1	29	10	43	14	52	18	54	15
1	180.1	29	10	43	14	52	18	56	16
1	181.1	29	10	46	15	52	18	54	15

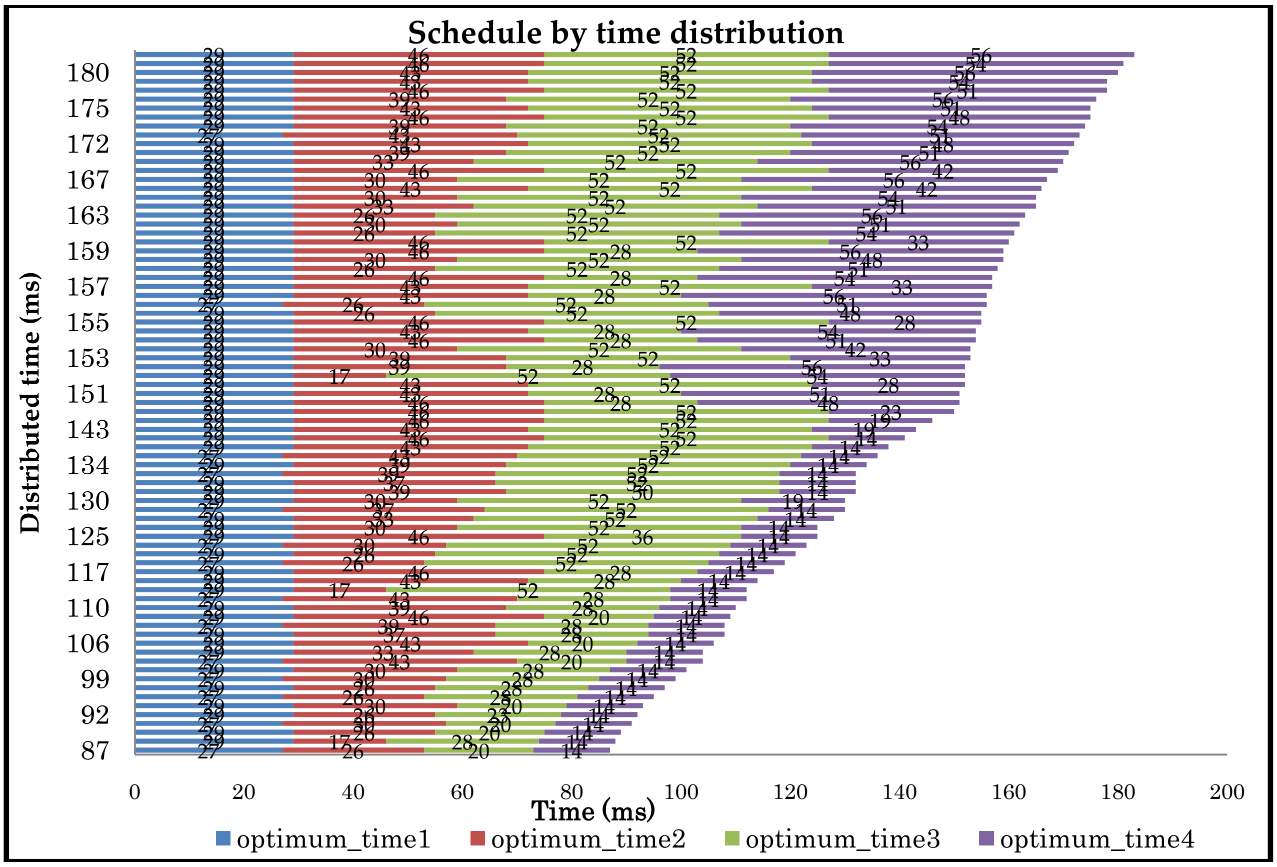


Fig. 5.9: Different sub-optimal schedules by distributed time

The distributed execution time and the outcomes of each sub-task and their overall outcomes for the first trial run are shown in Table 5.10.

Table 5.10: Distributed time of each sub-task and their related probability and the probability of the overall outcomes

Trial no.	Execution time (ms)	Optimum time1	P_1	Optimum time2	P_2	Optimum time3	P_3	Optimum time4	P_4	Probability of overall outcomes
1	87	27	0.9	26	0.580753	20	0.3	14	0.199351	0.500819
1	88	29	1	17	0.363027	28	0.5	14	0.199351	0.512851
1	89	29	1	26	0.580753	20	0.3	14	0.199351	0.5181
1	91	27	0.9	30	0.663227	20	0.3	14	0.199351	0.521437
1	92	29	1	26	0.580753	23	0.3	14	0.199351	0.527522
1	93	29	1	30	0.663227	20	0.3	14	0.199351	0.538719
1	95	27	0.9	26	0.580753	28	0.5	14	0.199351	0.550001
1	97	29	1	26	0.580753	28	0.5	14	0.199351	0.567282
1	99	27	0.9	30	0.663227	28	0.5	14	0.199351	0.570619
1	101	29	1	30	0.663227	28	0.5	14	0.199351	0.587901
1	104	27	0.9	43	0.932645	20	0.3	14	0.199351	0.588792
1	104	29	1	33	0.702426	28	0.5	14	0.199351	0.597701
1	106	29	1	43	0.932645	20	0.3	14	0.199351	0.606073
1	108	29	1	37	0.771592	28	0.5	14	0.199351	0.614992
1	108	27	0.9	39	0.845012	28	0.5	14	0.199351	0.616065
1	109	29	1	46	1	20	0.3	14	0.199351	0.622912
1	110	29	1	39	0.845012	28	0.5	14	0.199351	0.633347
1	112	27	0.9	43	0.932645	28	0.5	14	0.199351	0.637974
1	112	29	1	17	0.363027	52	1	14	0.199351	0.640595
1	114	29	1	43	0.932645	28	0.5	14	0.199351	0.655255
1	117	29	1	46	1	28	0.5	14	0.199351	0.672094
1	119	27	0.9	26	0.580753	52	1	14	0.199351	0.677745
1	121	29	1	26	0.580753	52	1	14	0.199351	0.695026
1	123	27	0.9	30	0.663227	52	1	14	0.199351	0.698363
1	125	29	1	46	1	36	0.6	14	0.199351	0.699765
1	125	29	1	30	0.663227	52	1	14	0.199351	0.715644
1	128	29	1	33	0.702426	52	1	14	0.199351	0.725444
1	130	27	0.9	37	0.771592	52	1	14	0.199351	0.725454
1	130	29	1	30	0.663227	52	1	19	0.271059	0.733571
1	132	29	1	39	0.845012	50	0.9	14	0.199351	0.734403
1	132	29	1	37	0.771592	52	1	14	0.199351	0.742736
1	132	27	0.9	39	0.845012	52	1	14	0.199351	0.743809
1	134	29	1	39	0.845012	52	1	14	0.199351	0.761091
1	136	27	0.9	43	0.932645	52	1	14	0.199351	0.765718
1	138	29	1	43	0.932645	52	1	14	0.199351	0.782999

1	141	29	1	46	1	52	1	14	0.199351	0.799838
1	143	29	1	43	0.932645	52	1	19	0.271059	0.800926
1	146	29	1	46	1	52	1	19	0.271059	0.817765
1	150	29	1	46	1	52	1	23	0.295111	0.823778
1	151	29	1	46	1	28	0.5	48	0.812619	0.825411
1	151	29	1	43	0.932645	28	0.5	51	0.887907	0.827394
1	152	29	1	43	0.932645	52	1	28	0.379908	0.828138
1	152	29	1	17	0.363027	52	1	54	0.956371	0.82985
1	152	29	1	39	0.845012	28	0.5	56	1	0.833509
1	153	29	1	39	0.845012	52	1	33	0.495134	0.835036
1	153	29	1	30	0.663227	52	1	42	0.689451	0.838169
1	154	29	1	46	1	28	0.5	51	0.887907	0.844233
1	154	29	1	43	0.932645	28	0.5	54	0.956371	0.84451
1	155	29	1	46	1	52	1	28	0.379908	0.844977
1	155	29	1	26	0.580753	52	1	48	0.812619	0.848343
1	156	27	0.9	26	0.580753	52	1	51	0.887907	0.849884
1	156	29	1	43	0.932645	28	0.5	56	1	0.855417
1	157	29	1	43	0.932645	52	1	33	0.495134	0.856945
1	157	29	1	46	1	28	0.5	54	0.956371	0.861349
1	158	29	1	26	0.580753	52	1	51	0.887907	0.867165
1	159	29	1	30	0.663227	52	1	48	0.812619	0.868961
1	159	29	1	46	1	28	0.5	56	1	0.872256
1	160	29	1	46	1	52	1	33	0.495134	0.873783
1	161	29	1	26	0.580753	52	1	54	0.956371	0.884281
1	162	29	1	30	0.663227	52	1	51	0.887907	0.887783
1	163	29	1	26	0.580753	52	1	56	1	0.895188
1	165	29	1	33	0.702426	52	1	51	0.887907	0.897583
1	165	29	1	30	0.663227	52	1	54	0.956371	0.904899
1	166	29	1	43	0.932645	52	1	42	0.689451	0.905524
1	167	29	1	30	0.663227	52	1	56	1	0.915807
1	169	29	1	46	1	52	1	42	0.689451	0.922363
1	170	29	1	33	0.702426	52	1	56	1	0.925607
1	171	29	1	39	0.845012	52	1	51	0.887907	0.93323
1	172	29	1	43	0.932645	52	1	48	0.812619	0.936316
1	173	27	0.9	43	0.932645	52	1	51	0.887907	0.937857
1	174	29	1	39	0.845012	52	1	54	0.956371	0.950346
1	175	29	1	46	1	52	1	48	0.812619	0.953155
1	175	29	1	43	0.932645	52	1	51	0.887907	0.955138
1	176	29	1	39	0.845012	52	1	56	1	0.961253
1	178	29	1	46	1	52	1	51	0.887907	0.971977
1	178	29	1	43	0.932645	52	1	54	0.956371	0.972254
1	180	29	1	43	0.932645	52	1	56	1	0.983161

1	181	29	1	46	1	52	1	54	0.956371	0.989093
1	183	29	1	46	1	52	1	56	1	1

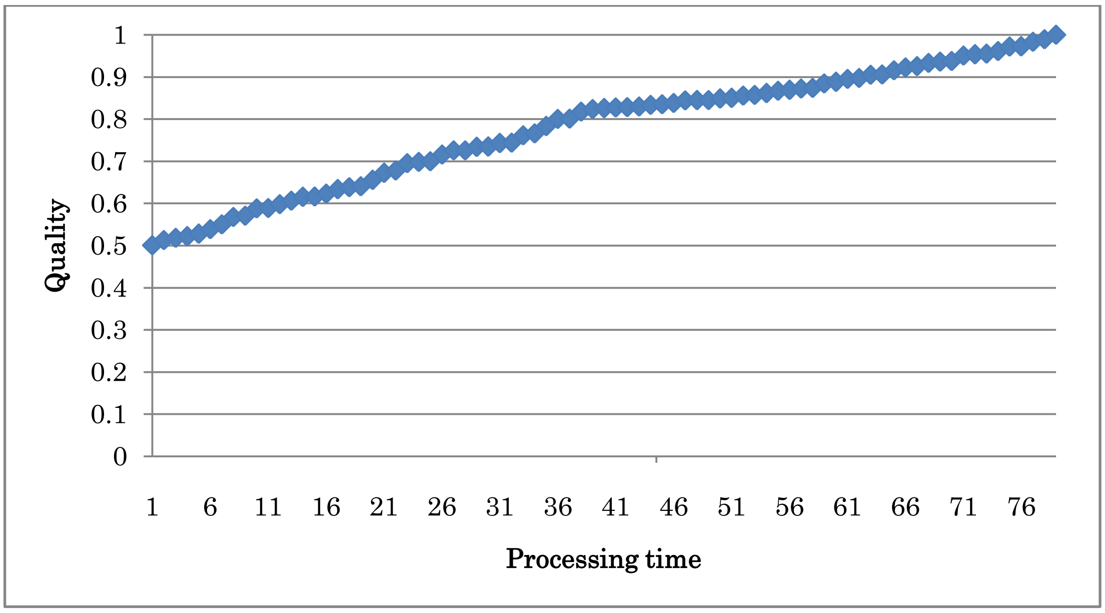


Fig. 5.10: The performance curve of sub-optimal overall outcomes

Figure 5.10 shows the performance curve of the sub-optimal overall outcomes of 4 independent tasks.

Experiment for image processing tasks by an adaptive scheduling using algorithm 4

In particular, the proposed method is applied to the boundary detection application in digital image processing as an example in order to optimize the overall processing result. So, the example tasks are noise reduction (NR), edge detection (ED), thinning (TH), and boundary detection (BD) performed by the steps 8, 6, 5, and 5 respectively. Thus, noise reduction NR has 8 sub-tasks and sub-task 1 is performed by step 1, sub-task 2 is performed by step 1 and 2, and so on. Similarly, ED, TH, and BD have 6 sub-tasks, 5 sub-tasks, and 5 sub-tasks performed by their corresponding steps respectively.

As mentioned above, the tasks are noise reduction, edge detection, thinning, and boundary detection performed by the steps 8, 6, 5 and, 5 respectively and the minimum required steps of these tasks for the imprecise but acceptable result are determined to **1, 3, 3, 2** respectively. Then, we can choose the step number of each task depends on the required processing time, i.e.,

	Step number
Noise reduction	1, 2, 3, 4, 5, 6, 7, 8
Edge detection	3, 4, 5, 6
Thinning	3, 4, 5
Boundary detection	2, 3, 4, 5

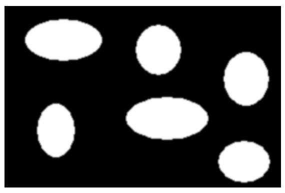
Thus, we have

$$\binom{8}{1}\binom{4}{1}\binom{3}{1}\binom{4}{1} = \frac{8!}{1!(8-1)!} \times \frac{4!}{1!(4-1)!} \times \frac{3!}{1!(3-1)!} \times \frac{4!}{1!(4-1)!} = 8 \times 4 \times 3 \times 4 = 384$$

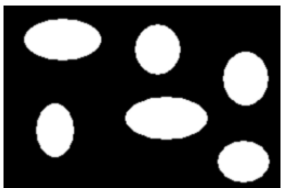
combinations of sub-tasks which is the same number as the possible acceptable overall processing results. Among these results, what solution is the optimal solution that depends on the required processing time of the combination of sub-task. So, it is necessary to be considered the execution time of each sub-task. The required step numbers of each task are determined by k_i, so that it has related processing time t_i i.e., the required processing time at step k_i, $0 < t_i$, $i = 1, 2, \ldots, N$.

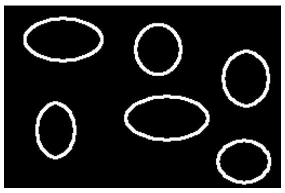
Experiment

In this case, the test images are multiple objects in an image for the boundary detection and single object image for boundary detection and to measure its length, area etc. Figure 5.11 expresses the original image and the result of each task for the combination of 4 tasks i.e., noise reduction, edge detection, thinning, and boundary detection for the purpose of detection of boundary points for the dependent case. Figure 5.11 (b) to (e) represents the result image of final step of each task as described in the above tasks. Figure 5.12 and Table 5.11 express the experimental result of scheduling of combination of these tasks by graphical and tabular representations. Figure 5.13 is the performance curve of the maximum performance of combination of 4 tasks for scheduling in dependent case. This curve can confirm the maximum performance of the combination of 4 tasks in a restricted time. As shown in this figure, the total quality of these tasks gradually improves as the processing time increases.

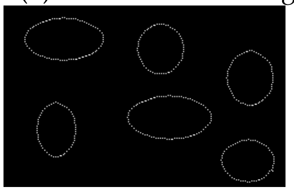


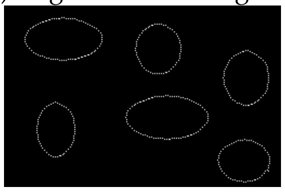
(a) Original input image





(b) Noise reduced image (c) Edge detected image





(d) Thinned image (e) Boundary detected image

Fig. 5.11: (a) Original input image (b) ~ (e) Output images by dependent case

Scheduling Results

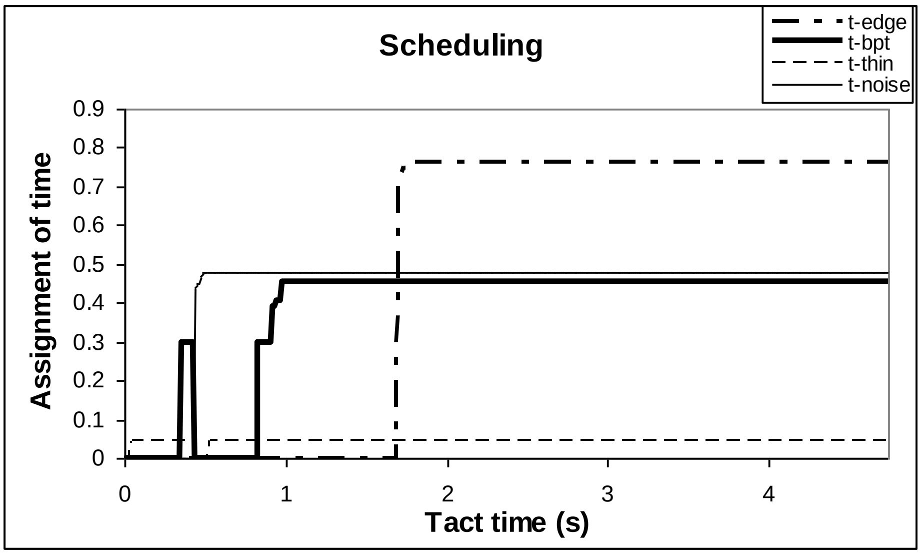


Fig. 5.12: Graphical representation of scheduling of 4 tasks

Table 5.11: Tabular representation of scheduling of 4 tasks for dependent case

Processing time (s)	**Assignment of tasks by processing time**								**Total Quality**
t	**t1**	**q1**	**t2**	**q2**	**t3**	**q3**	**t4**	**q4**	**Q**
0	0	0	0	0	0	0	0	0	0
0.05	0	0	0	0	0.046	1	0	0	0.25
0.1	0	0	0	0	0.046	1	0	0	0.25
|	|	|	|	|	|	|	|	|	|
|	|	|	|	|	|	|	|	|	|
0.3	0	0	0	0	0.046	1	0	0	0.25
0.35	0	0	0.296	0.016	0.046	1	0	0	0.2539
0.4	0	0	0.296	0.016	0.046	1	0	0	0.2539
0.45	0	0	0	0	0	0	0.45	1	0.3745
0.5	0	0	0	0	0	0	0.48	1	0.5
0.55	0	0	0	0	0.046	1	0.48	1	0.5
|	|	|	|	|	|	|	|	|	|
|	|	|	|	|	|	|	|	|	|
1	0	0	0.453	1	0.046	1	0.48	1	0.75
|	|	|	|	|	|	|	|	|	|
|	|	|	|	|	|	|	|	|	|
1.4	0	0	0.453	1	0.046	1	0.48	1	0.75
1.5	0	0	0.453	1	0.046	1	0.48	1	0.75
1.6	0	0	0.453	1	0.046	1	0.48	1	0.7539
1.7	0.718	0	0.453	1	0.046	1	0.48	1	0.9234
1.71	0.718	0	0.453	1	0.046	1	0.48	1	0.9975
1.72	0.734	0.279	0.453	1	0.046	1	0.48	1	1
1.73	0.75	0.689	0.453	1	0.046	1	0.48	1	1
1.74	0.75	0.689	0.453	1	0.046	1	0.48	1	1
1.75	0.765	1	0.453	1	0.046	1	0.48	1	1
1.76	0.765	1	0.453	1	0.046	1	0.48	1	1

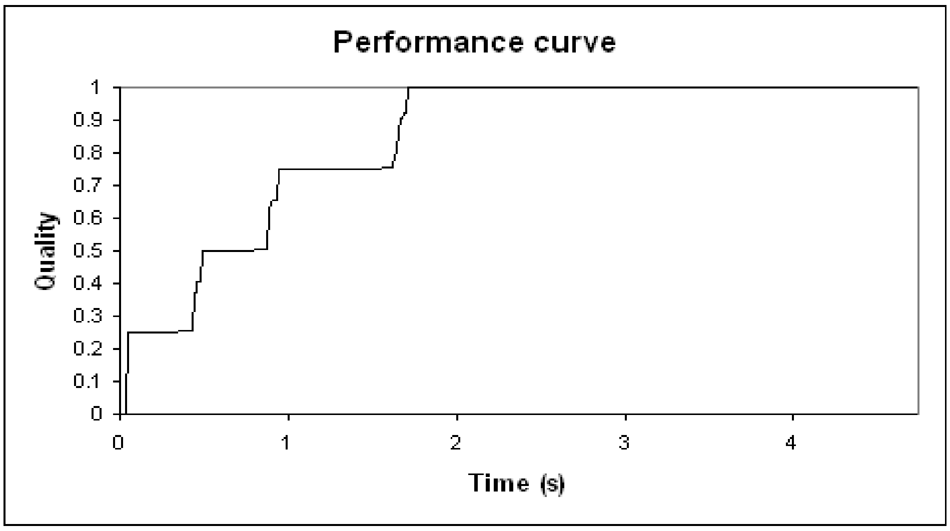


Fig. 5.13: Total performance of scheduling of 4 tasks

5.3 Effectiveness of the Proposed Method

Here, the effectiveness of the proposed method is presented as follows:

- the different intermediate results can choose with the related processing time
- the different performance curves can be achieved
- the adaptive static scheduling can be performed
- it is suitable for the real-time image processing system that uses mega data and restricted processing time such as image and video tracking, and image transmission systems and for the scheduling of tasks under time constraint etc.
- it is one of the idea to solve the time vs quality trade-off problems encountered in real-time system by dividing the task into small sub-tasks

5.4 Summary

In this chapter, how to schedule the tasks by imprecise computation is described by its algorithms case by case and the related experimental results. After that, the scheduling of the combination of tasks is experimented by using different processors and restricted time, undefined tact time without and with priority for the dependent, independent and both cases. The experimental results for the scheduling are shown by the graphical and tabular representations. The performance curve is expressed for the combination of 4 tasks as an example. Then, the effectiveness of the proposed method is expressed.

References

[1] J.W.S. Liu, K.-J. Lin, W.K. Shin, A.C.S Yu "Algorithms for Scheduling Imprecise Computation" *IEEE Trans. Computers ,Vol.* 19,No.9, Sept. 1991,pp. ,156-1,173.

[2] W.-K. Shih and J.W.-S. Liu, "Algorithms for Scheduling Imprecise Computations with Timing Constraints to Minimize Maximum Error," IEEE Trans. Computers., vol. 44, no.3, pp. 466-471, Mar 1995.

[3] W.W. Kywe, D. Fujiwara and K. Murakami, "Scheduling of Image Processing Using Anytime Algorithm for Real-time System," *Proc. of the 18th International Conference on Pattern Recognition* (ICPR2006), Vol.3, pp.1095-1098, Hong Kong/ China, 2006/08.

[4] T. Dean and M. Boddy, "An Analysis of Time-Dependent Planning," *Proc. AAAI-88*, pp.49-54, AAAI, (1988).

[5] S. Zilberstein and S. J. Russell. In S. Natarajan (Ed.), "Approximate Reasoning Using Anytime Algorithms," *Imprecise and Approximate Computation*, Kluwer Academic Publishers, (1995).

[6] J.Grass and S. Zilberstein. In M. Pittarelli (Ed.), "Anytime Algorithm Development Tools," *SIGART Bulletin Special Issue on Anytime Algorithms and Deliberation Scheduling*, 7(2):20-27, (1996).

[7] S. Zilberstein Ph.D dissertation, "Operational Rationality through Compilation of Anytime Algorithm," *Computer Science Division, University of California at Berkeley*, (1993).

[8] J.R. Parker, "Algorithms for image processing and computer vision," *John Wiley & Sons, Inc. U.S.A*, 1997.

Chapter 6
Overall Processing Result in AAIP

6.1 Introduction

This chapter explains the formulation of the quality function in each AAIP method, and also the realization and evaluation of overall performance curves. Then, the quality of image processing for noise reduction, edge detection, thinning, and boundary detection including optimization of the overall processing quality, its evaluation by time precision function, and the performance curves for dependent and independent cases are described. After that, the overall quality of scheduling is also explained. Furthermore, how to evaluate the anytime algorithmic erosion and boundary detection is described from the viewpoint of probability theory.

6.2 Formulation of Quality Function in AAIP

In this section, how to formulate the quality functions in AAIP is described. Suppose that if the system is composed of 4 tasks and the total processing time is restricted.

Let

T = Restricted time
P_1, P_2, P_3, P_4 = task 1, 2, 3, and 4 respectively
q_{1i} = quality of task P_1 at step i, where i = 1, 2, …, a
q_{2j} = quality of task P_2 at step j, where j = 1, 2, …, b
q_{3k} = quality of task P_3 at step k, where k = 1, 2, …, c
q_{4l} = quality of task P_4 at step l, where l = 1, 2, …, d
t_1, t_2, t_3, t_4= total required time for the task P_1, P_2, P_3, and P_4 respectively
Q = maximum performance of above 4 tasks

In general, the intermediate results of each task at current step are described below:

$$q_j^i = F_i(x_j) = \frac{\text{no.of approximated result at step j}}{\text{no.of total result at final step}} \qquad (6.1)$$

In particular,

(1) For noise reduction, the probability of the reduced noises at step 2,

$$q_2^1 = F_1(x_2) = \frac{\text{no. of approximated reduced noises at step 2}}{\text{no. of total reduced noises at final step}}$$
$$= \frac{30908}{32935} = 0.9385$$

$$\begin{matrix}\text{Percentage of intermediate} \\ \text{result at step 2 for noise reduction}\end{matrix} = q_2^1 * 100 = 0.9385 * 100 = 93.85\%$$

(2) For edge detection, the probability of the detected edge points at step 3,

$$q_3^2 = F_2(x_3) = \frac{\text{no. of approximated detected edge points at step 3}}{\text{no. of total detected edge points at final step}}$$
$$= \frac{3609}{5836} = 0.6184$$

$$\begin{matrix}\text{Percentage of intermediate} \\ \text{result at step 3 for edge detection}\end{matrix} = q_3^2 * 100 = 0.6184 * 100 = 61.84\%$$

(3) For thinning, the probability of the thinned pixels or deleted points at step 4,

$$q_4^3 = F_3(x_4) = \frac{\text{no. of approximated deleted points at step 4}}{\text{no. of total deleted points at final step}}$$
$$= \frac{1978}{2412} = 0.8201$$

$$\begin{matrix}\text{Percentage of intermediate} \\ \text{result at step 4 for thinning}\end{matrix} = q_4^3 * 100 = 0.8201 * 100 = 82.01\%$$

(4) For boundary detection, the probability of the detected boundary points at step 3,

$$q_3^4 = F_4(x_3) = \frac{\text{no. of approximated detected boundary points at step 3}}{\text{no. of total detected boundary points at final step}}$$
$$= \frac{925}{2388} = 0.3874$$

$$\begin{matrix}\text{Percentage of intermediate} \\ \text{result at step 3 for boundary detection}\end{matrix} = q_3^4 * 100 = 0.3874 * 100 = 38.74\%$$

Therefore, the percentage of the quality of the overall processing result of the combination of 4 tasks can be defined by

$$Q = (q_2^1 \circ q_3^2 \circ q_4^3 \circ q_3^4) * 100 = (0.9385 * 0.6184 * 0.8201 * 0.3874) * 100$$
$$= 0.1843 * 100 = 18.43\%$$

which expresses one of the quality of the possible overall processing result out of 384 results.

6.3 Overall Performance

6.3.1 Realization of Overall Performance

Dependent case

In this case, if the system is composed of the combination of many tasks that are inter-related each other in a particular time constraint. The calculation of current result for a task is depend on the result of previous task i.e., to perform task 2 is based on the result of task 1, and to perform task 3 is based on the result of task 2, and so on. Then, the total performance (combined quality) can be evaluated by using CPP that is described in chapter 2.

Let

T = Restricted time
$P_1, P_2, P_3, \ldots, P_n$ = task 1, 2, 3, …, n respectively
$Q_1, Q_2, Q_3, \ldots, Q_n$ = quality of task 1, 2, 3, …, n respectively
$t_1, t_2, t_3, \ldots, t_n$ = total required time for each process $P_1, P_2, P_3, \ldots, P_n$ respectively

Thus,

the total performance of combination of n tasks

$$Q = Q_n\{Q_{n-1}, \ldots Q_4\{Q_3[Q_2(Q_1(Q_0, t_0), t_1), t_2], t_3\}, \ldots, t_{n-1}\} \qquad (6.2)$$

Here,

Q_0 and t_0 means that the initial quality 0 and initial starting time 0 at step 0

$Q_1(Q_0, t_0)$ means that the quality at step 1 has input quality Q_0 and processing time t_0

$Q_2(Q_1(Q_0, t_0), t_1)$ means that the quality at step 2 has input quality Q_1 and processing time t_1 and so on.

In this case, the total quality Q might be less than or equal to the maximum quality i.e., 1.

Independent case

In order to obtain the overall performance at each processing time or step, by choosing the prior process that has predefined executing time with high quality result if it meets the current processing time. If the time is not enough i.e., its execution time is greater than current execution time, then we choose the suitable processing time with appropriate quality result to avoid the huge amount of rest time and so on. Formulation of quality functions for independent case is described as follows:

In order to optimize the total performance in a restricted time, the constructed optimization model is

$$\text{Max } Q = \sum_{n=1}^{N} (q_n(t_n)) / n \tag{6.3}$$

Subject to

$$\begin{aligned} t_{11} + t_{12} + t_{13} + \dots + t_{1a} &\le t_1 \\ t_{21} + t_{22} + t_{23} + \dots + t_{2b} &\le t_2 \\ t_{31} + t_{32} + t_{33} + \dots + t_{3c} &\le t_3 \\ t_{41} + t_{42} + t_{43} + \dots + t_{4d} &\le t_4 \end{aligned}$$

and

$$t_1 + t_2 + t_3 + t_4 \le T$$

Here, $t_1, t_2, t_3, t_4 \ge 0$

6.3.2 *Evaluation of Overall Quality*

The overall quality of image processing is defined by the quality of overall result of the combined 4 independent tasks and it is evaluated by

$$\text{The quality of overall processing result} = \frac{1}{4}\sum_{i=1}^{4} w_i q(i, j^i) \tag{6.4}$$

Where

$q(i, j^i)$ = the improvement rate of the quality of result of the task i and its step j^i with related processing time

i = 1, 2, 3, and 4, respectively

w_i = weighted average value

6.4 Quality of Image Processing

In this section, how to define the quality of image processing in AAIP is described. Figure 6.1 to 6.4 represent the performance curves of the average noise reduced rate, the average edge detected rate, the average thinned rate, and the average boundary detected rate by the AAIP method.

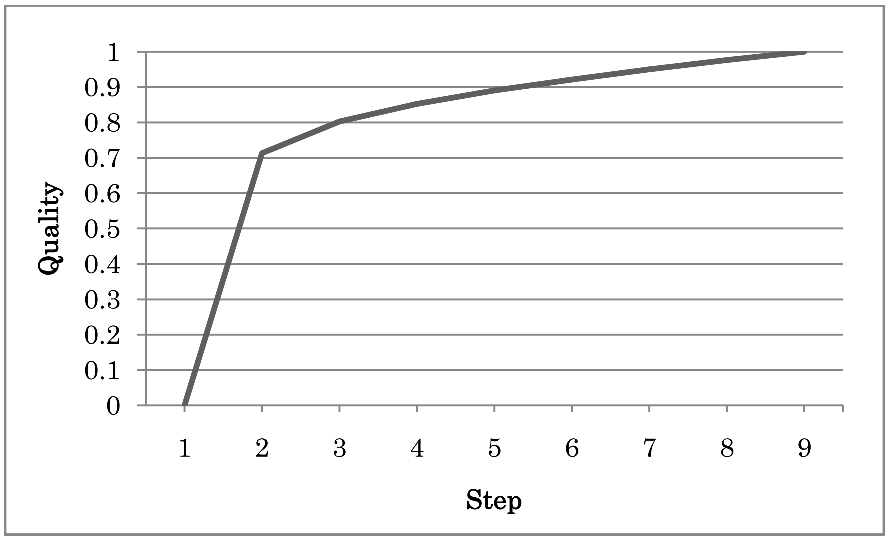


Fig. 6.1: The average performance curve of noise reduction by 8 steps

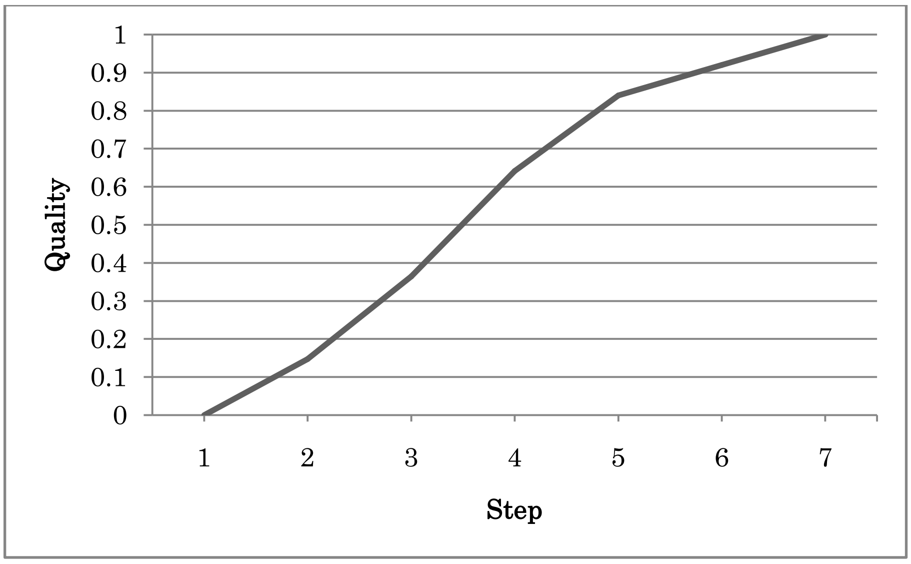


Fig.6.2: The average performance curve of edge detection by 6 steps

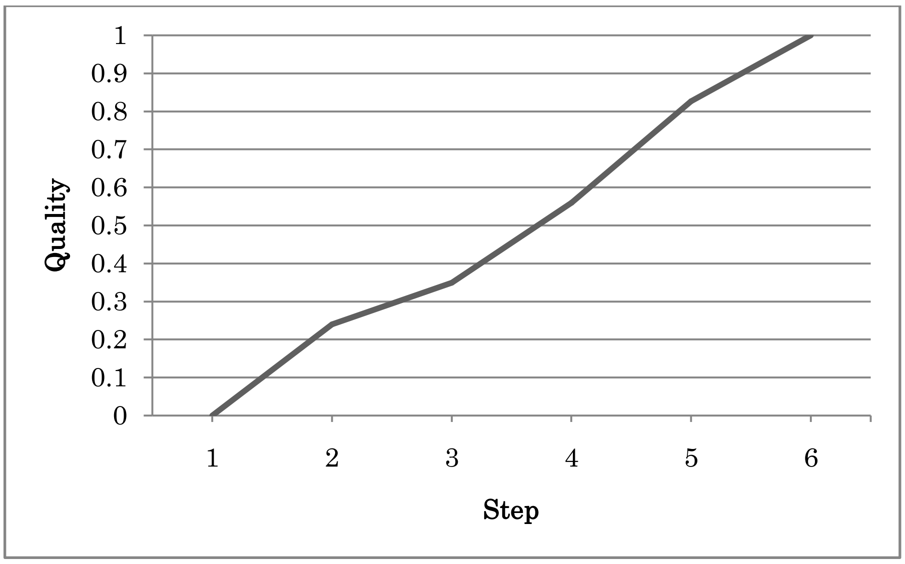


Fig. 6.3: The average performance curve of thinning by 5 steps

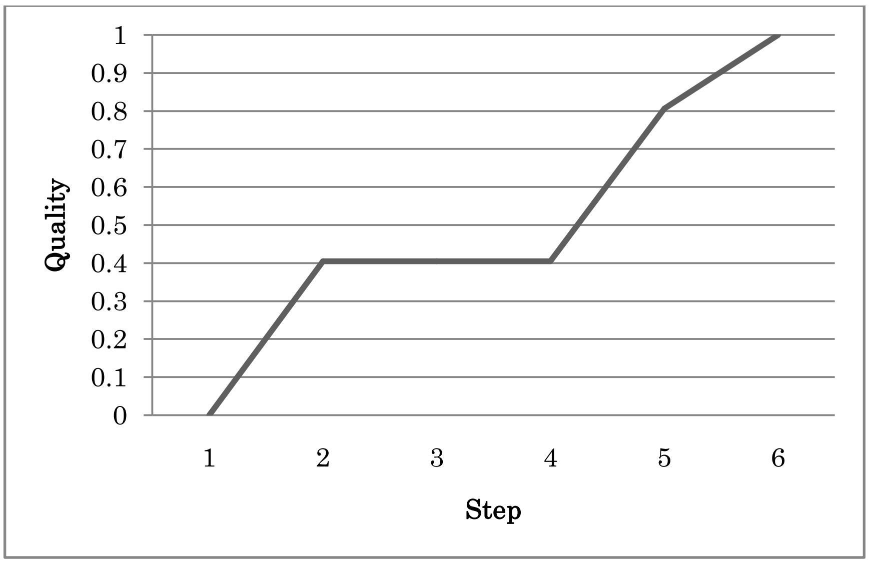


Fig. 6.4: The average performance curve of boundary detection by 5 steps

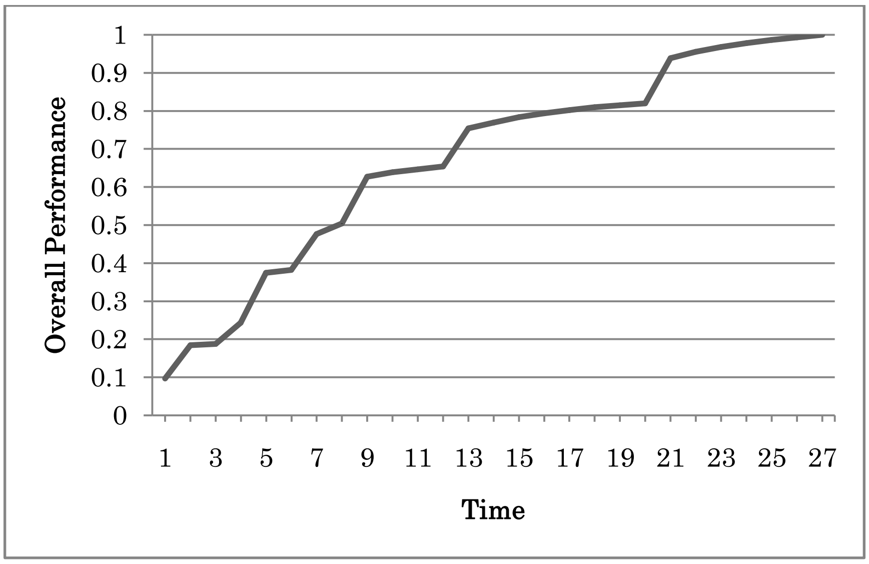


Fig. 6.5: The performance curve for the overall processing result of 4 tasks

Fig. 6.5 expresses the quality of overall processing result evaluated by (6.4) and it shows that the quality of overall processing result gradually improves the processing time increases.

6.4.1 Realization of Overall Quality in Scheduling

Figure 6.6 shows the experimental results of the proposed scheduling mechanism for 4 tasks by assigning the required processing time for each task while reducing the idle/rest time. Figure 6.7 is the overall performance of scheduling of 4 tasks and it confirms the effectiveness of this method satisfying the definition of performance profile.

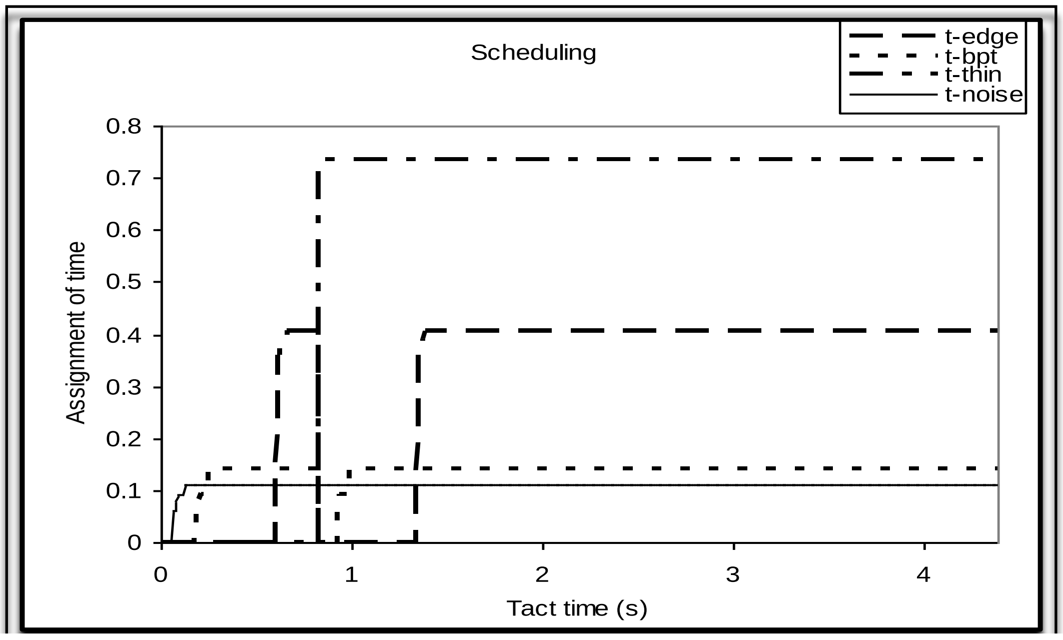


Fig. 6.6: Scheduling result for 4 tasks

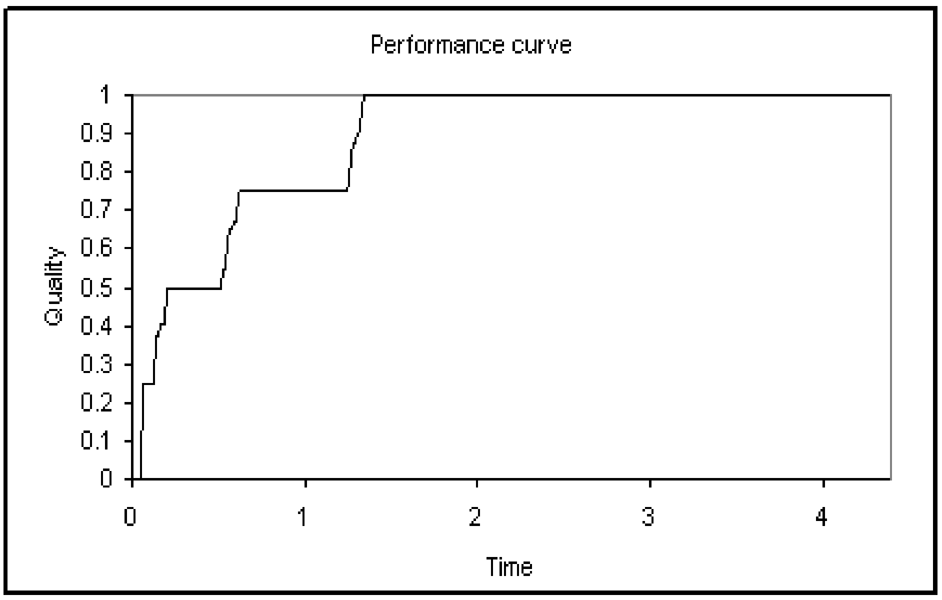


Fig. 6.7: Overall performance of 4 tasks

Table 6.1 shows the experimental data of proposed scheduling mechanism.

Table 6.1: Scheduling result for independent case

Processing time (s)	**Assignment of tasks by processing time**								**Total Quality Q**	**Rest/Idle time (s)**
	t_1	q_1	t_2	q_2	t_3	q_3	t_4	q_4		
0	0	0	0	0	0	0	0	0	0	0
0.01	0	0	0	0	0	0	0	0	0	0.01
0.02	0	0	0	0	0	0	0	0	0	0.02
|	|	|	|	|	|	|	|	|	|	|
|	|	|	|	|	|	|	|	|	|	|
0.19	0	0	0.078	0.498	0	0	0.11	1	0.405138	0.01
|	|	|	|	|	|	|	|	|	|	|
|	|	|	|	|	|	|	|	|	|	|
|	|	|	|	|	|	|	|	|	|	|
|	|	|	|	|	|	|	|	|	|	|
1.39	0.406	1	0.14	1	0.734	1	0.11	1	1	0

6.4.2 *Performance Curves*

In this section, how to evaluate the performance curves for anytime algorithmic erosion and boundary extraction. In this proposed method, let the result be extracted boundary points, and the boundary extraction operation is performed by the anytime algorithmic erosion method using many steps with the different pattern of structuring elements. If the number of processing steps increases then the required processing time also increases and quality of result will be improved. Thus, the quality of result can be expressed by the number of steps with related processing time by the probability of correctness i.e., certainty. The performance curve can be evaluated from the viewpoint of probability theory.

Suppose that the discrete random variable

X = number of extracted boundary points = $\{x_i\}$
x_i = number of extracted boundary points at step i, $x_i \in R^2$, $i = 1, 2, \ldots, 8$.

Thus, the Cumulative Distribution Function (CDF) of X is defined by

$$F(x_i) = \begin{cases} 0 & : \ i <= 0 \\ x_i/x_8 & : \ 0 < i < 8 \\ 1 & : \ 8 <= i \end{cases} \qquad (6.5)$$

Figure 6.8 shows the average performance curve of eroded points of many tested images applied by algorithm 1.

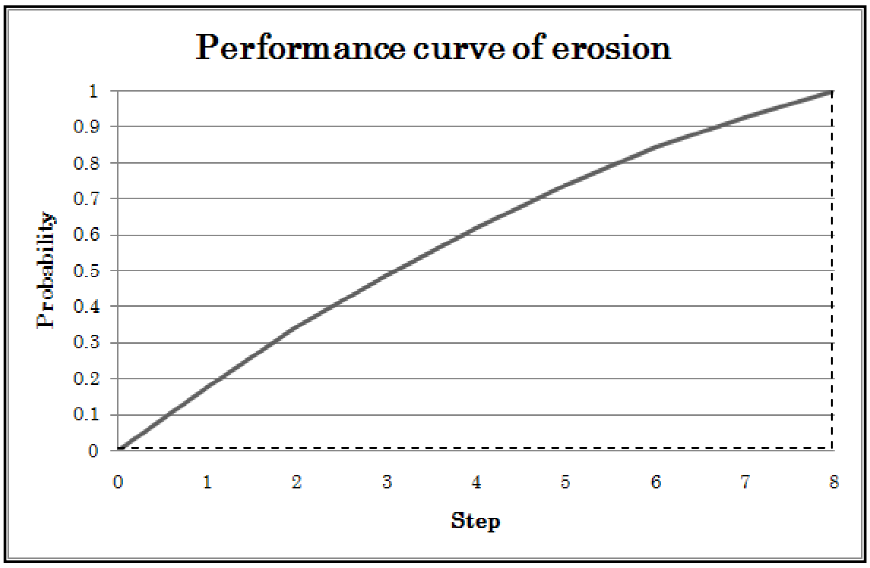


Fig. 6.8: Performance curve of eroded points

—— Proposed ----- Conventional

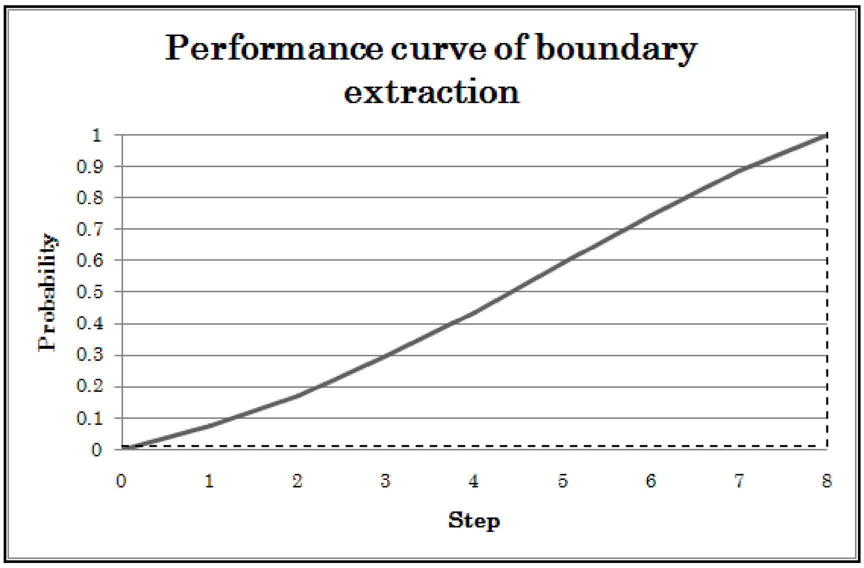


Fig. 6.9: Performance curve of boundary extracted points

—— Proposed ----- Conventional

Figure 6.9 shows the average performance curve of boundary extracted points of many tested images applied by algorithm 2 by using equation (2). As shown in Fig. 6.8 and 6.9, the conventional method can obtain only one result with required processing time and the proposed method can obtain multiple course-to-fine results with required processing time, and the performance curve can be realized.

6.5 Summary

This chapter explained the formulation of the quality function in AAIP for noise reduction, edge detection, thinning, and boundary detection as examples. Then, formulation of the quality function to optimize the overall performance under the processing time constraint for the dependent, independent and both cases are described based on the concept of linear programming and probability theory. After that, the performance curves of each AAIP method and the overall performance of 4 tasks is expressed by figures. How to realize the overall quality in scheduling is described by graphical i.e., performance curve and tabular for the independent case. Moreover, how to evaluate the performance curve for the anytime algorithmic erosion and boundary extraction is also described from the viewpoint of probability theory.

References

[1] M. Boddy and T. Dean. *"Decision-theoretic deliberation scheduling for problem solving in time-constrained environments"*, Artificial Intelligence, 67(2):245--286, 1994.

[2] Thomas Dean, *"Deliberation Scheduling for Time-Critical Scheduling in Stochastic Domains"*

[3] S. Zilberstein, "Monitoring Anytime Algorithms"

[4] A. Garvey, V. Lesser. *"Design-to-Time Real-Time Scheduling"*, IEEE Transactions on Systems, Man and Cybernetics, 1993
ftp://ftp.cs.umass.edu/pub/lesser/garvey-dtt-smc.ps

[5] Garvey, Alan and Victor Lesser. *"Design-to-time Scheduling and Anytime Algorithms"*, SIGART Bulletin, 7 (2):16--19 (1996).
http://citeseer.comp.nus.edu.sg/85186.html

[6] Dean, Thomas and Boddy, Mark, "An Analysis of Time_Dependent Planning", Proceedings of Anytime Algorithm AI-88, St. Paul, Minnesota, pp. 49–54, 1988.

[7] Hasegawa, H. Kubota and J. Toriwaki, "Automated construction of image processing procedures by sample figure presentation", Proc. of 8th Int'l Conference on Pattern Recognition (ICPR1986), Vol. 1, pp.586-588 (Oct.1986).

Chapter 7
Contact Lens Extraction by Thermo-Vision Image

7.1 Introduction

For the security improvement, introduction to biometrics technology is performed positively. It is strongly promoted in order to defend a computer system against injustice accesses and also to construct a secure system. Of course, it is necessary to modify some of image processing method in pre-processing to anytime algorithmic form to apply in real time system for the security purpose e.g., image taken by web camera or thermo image taken by thermo vision camera or finger print image in airport entry system. So, in this research work, I studied about thermo vision camera and its thermo images to support the pre-processing steps in biometric for the security purpose. I particular, I studied how to extraction contact lens from thermo-vision image using thermal properties.

So, in this chapter, I would like to explain about the procedure of detection of contact lens wearing by using thermo-vision image for the pre-processing step in the application of biometric. If the system uses face/iris images, it is usually required the information whether a person wears glasses/contact lenses or not in advance. Some researcher reported a method to recognize glasses from an infrared image based on the property that the infrared rays are hardly to pass a glass. But, there remains a problem that it is difficult to recognize the contact lens from one shot infrared image because of the differences of temperature between around the eye's region and lens is very small, and moreover, the area of eye region is not so wide.

The idea for detection of contact lens wearing or not is based on the properties that the surface of the eye is always covered with liquid (tear), when we widely open eyes, the temperature gradually falls because of the evaporation of liquid (tear). And, there are differences of the transition of the temperature whether we wear contact lenses or not. Moreover, there exist some differences of temperature between soft and hard lenses. Based on these properties, a method to extract and distinguish contact lenses from thermo vision image i.e., infrared image sequences is proposed by using a thermo-vision camera. Some experimental results show about 72% success and this means that the possibility to apply this basic method to the pre-processing of biometrics [1].

The rest of this chapter is organized with five sections, the basic properties of thermal with and without wearing contact lenses is explained in section 2. Then, the differences of the transition of the temperature in wearing hard and soft lenses are expressed in section 3. The algorithms to extract and distinguish hard and soft contact lenses are described in section 4. And then, the experimental environment, experimental results and discussion are expressed in section 5. Finally, the chapter summary including the extension of this research part is described in section 6.

7.2 Basic Properties

7.2.1 Thermal Property with Contact Lenses

When we wear contact lens, there has the transition of temperature and its shape becomes two peaks. Some experiments show that the temperature of the eye when wearing contact lens is about 0.5 ~ 1.5 degree C lower than non lens or without contact lens.

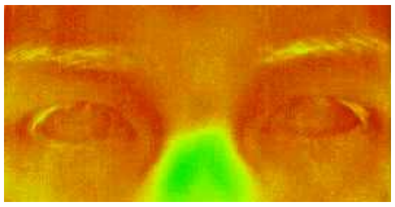
7.2.2 Thermal Property without Contact Lenses

The surface of a person's eye is always covered with liquid (tear). After opens his/her eyes, the temperature gradually falls because of the evaporation of liquid. Furthermore, a wink returns the temperature and liquid as before. This kind of periodical transition appears only in winking and breathing. This research part focuses on the characteristic of the transition of the temperature.

The differences in a usual wink is only 0.1 ~ 0.2 [deg. C] and this would be absorbed in the noises. By some experiments, it was clarified that closing long time amplifies the differences. So, the testers are needed to close the eyes at about 10 seconds and after rise up the temperature as high as possible, then to reopen their eyes.

Figure 7.1 is the examples of thermo-vision images for non lens and contact lens. Although some lower regions caused by sweat exist around the nose as shown in Fig. 7.1 (a), an image of non lens, the difference around the eye region is very small. Conversely, when wearing contact lens, the difference is magnified as shown in Fig. 7.1 (b). This research focuses on the feature of this difference and judges by using the histogram of temperature around the eyes.

Figure 7.2 shows the histogram of the temperature around the eyes related to the thermo vision images as shown in Fig. 7.1. The shape of histogram of non lens is uni-modal distribution, and the shape of histogram of contact lens is bi-modal distribution, respectively.


(a) Non lens

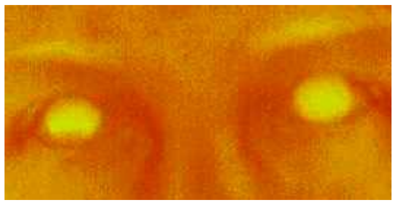
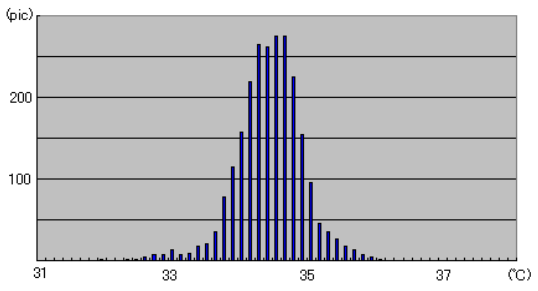
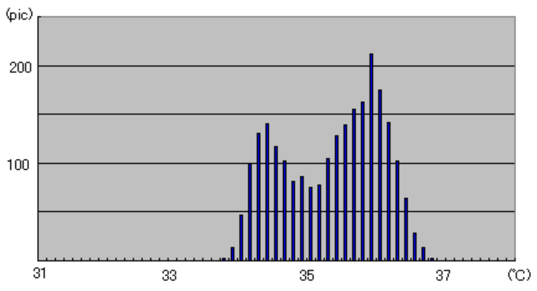

(b) Contact lens

Fig. 7.1: Examples thermo-vision images for non lens and contact lens



(a) Non lens (uni-modal distribution)



(b) Contact lens (bi-modal distribution)

Fig. 7.2: Histograms of temperature around eye region

7.3 Differences of Transition of Temperature in Wearing Soft and Hard Lenses

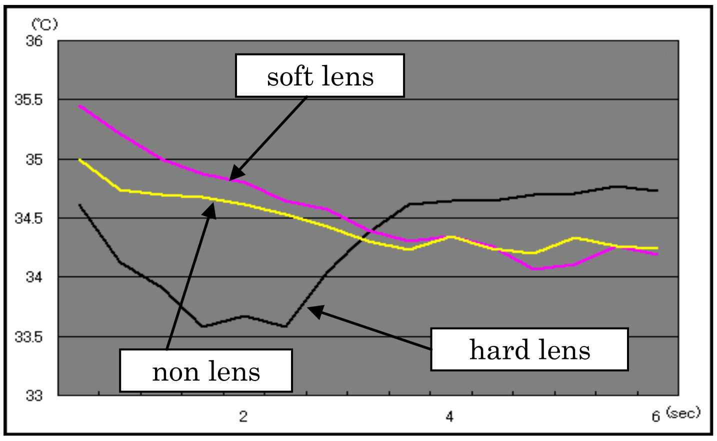


Fig. 7.3: Typical example of the transitions of soft, hard and non lens

Some experiments show that the transition of the temperature differs from the type of lens. Concretely, in the case of soft lens, the temperature monotonously falls after the eye opens. Conversely, in the case of hard lens, once it falls rapidly, then it rises again as shown in Fig. 7.3, (v-type transition). These properties are newly found by this research part and applied these features to extract and distinguish contact lenses [1].

Here, the transition of hard lens for some testers is similar to the soft or non lens, i.e., monotonous fall. Thus, the first step of the recognition algorithm which will be explained in next section judges whether it is hard lens or not, then extracts the eye area and checks the conditions for the soft lens in the area.

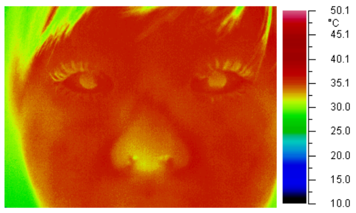
7.4 Algorithms for Contact Lenses Extraction

As described in section 7.3, a hard lens extraction algorithm that expressed below is necessary to apply first. If the hard lens's conditions are not satisfied, then apply the soft lens extraction algorithm.

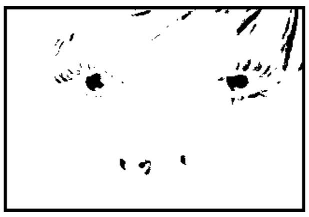
7.4.1 Hard Lens Extraction Algorithm

Step–1: Input a thermo-vision image sequences that are taken for 15 seconds. An example of thermo-vision image for hard lens is shown in Fig. 7.4(a).
Step–2: From these image sequences, cut 7 seconds data with opening eyes (every 0.4 seconds).
Step–3: Apply anytime algorithmic smoothing operator of 3 x 3 in order to remove thermal noises.
Step–4: Extract the candidate pixels which satisfy the hard lens's condition, i.e., v-type transition (Fig. 7.4(b)).
Step–5: Apply a morphology opening process in order to remove small noises and thin lines (Fig. 7.4(c)).
Step–6: After applying the labeling process, extract the 1st and the 2nd widest areas among the pixel region to be the candidates of hard lenses.
Step–7: Check these 2 candidate areas to be satisfied by the locus condition for hard lens. If it is satisfied, then decide this contact lens is 'hard lens' and terminate the algorithm, otherwise go to soft lens extraction algorithm.

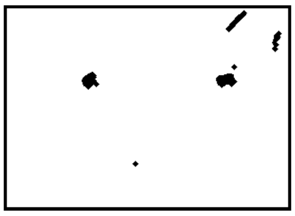
In Step 4, the pixels less than 33.5 or more than 38.0 degree C at the first frame are omitted. If the two conditions which are the difference of the temperature between the minimum frame's and the first frame's is larger than 0.8 degree C, and the difference between the minimum frame's and the last frame's is larger than 0.3 degree C are satisfied, then let it be the candidate pixels of hard lens. These values are obtained by many times of pre-experiments.



(a) Original thermo-vision image (hard lens) (Step 1)



(b) Candidate pixels of hard lens's region (Step 4)



(c) Candidate pixels of hard lens's region after applying morphological opening (Step 5)

Fig. 7.4: Sample output images in processing steps for hard lens

7.4.2 *Soft Lens Extraction Algorithm*

Step–8: Extract the candidate pixels which satisfy the soft lens condition (Fig. 7.5(b)).

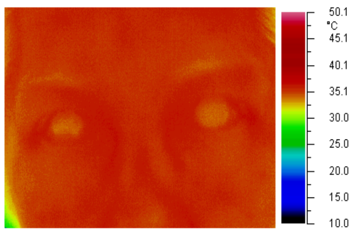
Step–9: Apply a morphology opening process in order to remove small noises and thin lines.

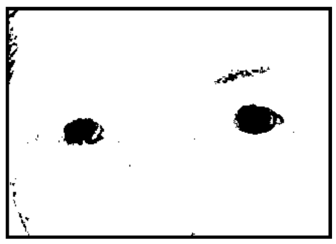
Step–10: After applying a labeling process, then extract the 1st and the 2nd widest areas among the pixel region to be in candidates of soft lens.

Step–11: Check the locus condition for these 2 candidate areas. If it is not satisfied, then decide 'non lens' and terminate the algorithm, otherwise, go to the next step.

Step–12: Calculate the histogram around the eye areas of the 'soft lens' obtained above, and check another histogram condition. If the condition is satisfied, then decide this contact lens is 'soft lens' and terminate the algorithm, otherwise, decide as 'non lens'.

In Step – 8, the pixels less than 33.5 or more than 38.0 degree C at the first frame are omitted. Furthermore, the pixels that the temperature rise up more than 0.7 degree C or fall down less than 1.0 degree C are omitted between the neighboring frames (every 0.4 seconds). Next, let the pixels which satisfy the following two conditions, the difference of the temperature is larger than 0.4 degree C between the first and the last frames', and the difference between the maximum and the minimum is from 0.5 to 2.0 degree C, be the candidate pixels of the soft lens. These values are obtained by many times of pre-experiments. Figure 7.5(a) shows an example of thermo-vision image of 'soft lens' and Fig. 7.5(b) is the result of Step8.





(a) Original thermo-vision image (soft lens) (b) Candidate pixels of soft lens's region (Step 8)

Fig. 7.5: An example extracted result for soft lens

7.5 Experimental Results and Discussions

7.5.1 Machine Environment

The experimental environment is used by the following items:

Thermo-vision camera

Model: TVS-710
- Temperature range : -20 ~ 300 [deg. C]
- Frame rate : 1/30 [seconds]
- Image size : 320(H) X 240(V) [pixels]
- Wave length : 8 ~ 14 [micrometer]

PC
- Processor/memory : Xeon 3.07 GHz / 512 MB RAM
- OS : Windows XP
- Programming language : Visual C++ 7.0

7.5.2 *Experiment*

The experiment has done by the different seasonal conditions such as in winter, in summer and in high humidity and the temperature's conditions like room temperature and humidity are expressed in Table 7.1.

Table 7.1: Experimental data

	Season's condition		
Cases	**A** Winter	**B** Summer	**C** High humidity
Room temperature [degree C]	20 ~ 23	28 ~ 32	24 ~ 25
Humidity	30 ~ 40 %	40 ~ 50 %	57 ~ 70 %

The instructions to the testers are as follows:

- Head position is face to a camera with no inclination
- The distance between the face and camera is at about 30 cm
- Open eyes after close eyes is more than 10 seconds (i.e., winking time)
- Be in quiet position during the measurement

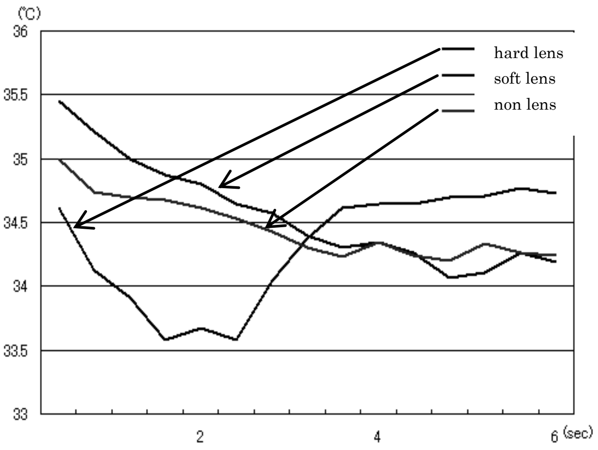
7.5.3 *Results and Discussions*

The following table shows the experimental data and its related result. The numbers of testers are 11 for hard lens, 14 for soft lens and 14 for non lens, respectively. Experimental result shows that the recognition rates are about 80% in winter (case A; 17 persons/21 persons), 50% in summer (case B; 4 persons/8 persons), respectively[2] and 50% in high humidity (case C; 5 persons/10 persons), and the overall recognition rate is 66% (i.e., 26 persons/ 39 persons).

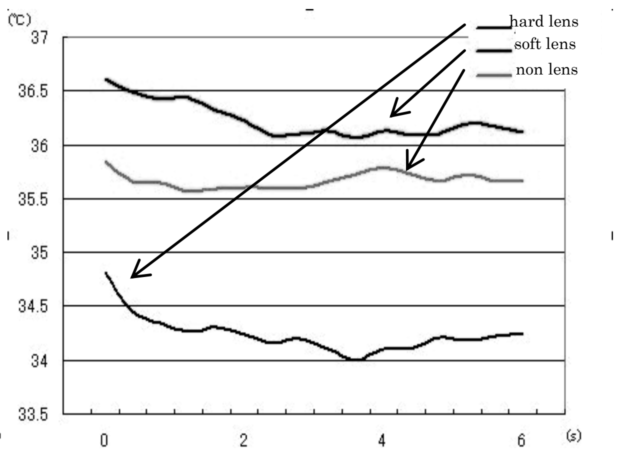
Table 7.2: Experimental result

	Detected performance rate	**Success Persons**	**Total person (Testers)**
Case A	80 %	17	21
Case B	50 %	4	8
Case C	50 %	5	10
Average	66 %	**26**	**39**

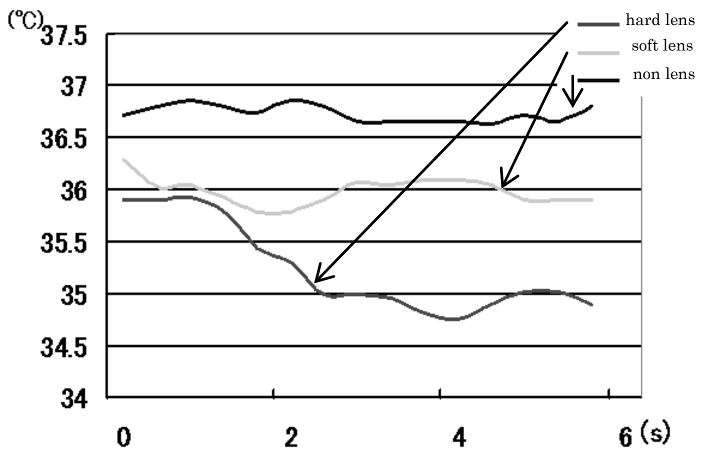
Although the number of tester is not so large and cannot be said definitely, the facts that the reason of the low recognition rate in summer and in high humidity are caused by the difference of the transitions between in winter, in summer and in high humidity as typically shown in Fig.7.6. Many characteristic points appear in the transition in winter as shown in Fig. 7.6 (a). Inversely, they are buried in summer as shown in Fig. 7.6 (b). And, they are buried in high humidity as shown in Fig. 7.6 (c). In these seasons such as in summer and in high humidity, the amplitude is comparatively small because the evaporation from the surface of the eye decreases according to the increase of the environmental humidity and the temperature.



(a) Case A (in winter)



(b) Case B (in summer)



(c) Case C (in high humidity)

Fig. 7.6: Typical examples of the transition of the temperature in different seasonal cases

7.6 Summary

This chapter clarified the new facts that (1) if a person opens his/her eyes, the temperature gradually falls because of the evaporation of liquid (tear), (2) the transition of the temperature differs whether he/she wears contact lenses or not, and, (3) there exists some differences between soft and hard lens. Based on these properties, it expressed a method to extract and distinguish contact lens by using a thermo-vision image sequences for the application of Biometrics. Some experimental results showed about 72% success rate. But, there still remain some subjects to be solved, for example, it is required to absorb the difference in the transition among individuals, and it is also important to realize an autonomous method adaptive to the temperature and humidity in order to tune the parameters used as the magic numbers in the algorithm and examine the influence of movement of a subject when photographing it. Furthermore, to examine the influence of the outside temperature/humidity might also be needed. And, to measure the eye winking rate in a restricted time 33 ms i.e., 30 frames/second are the future works.

References

[1] K. Momiyama, K. Kato, and K. Murakami, "Contact Lens Extraction by Using Thermo-Vision", *Proc. of the 10th Symposium on Sensing via Image Information (SSII 2004),*Yokohama,June2004,pp.471-477(in Japanese).

[2] M. Yoshida and K. Murakami, "A method to extract contact lens by using image sequences of thermo-vision", *ITE Technical Report*, Vol.29, No.47 (ME2005-117), Hokkaido, Aug.2005, pp.41-44 (in Japanese).

[3] Wyne Wyne Kywe, Masashi Yoshida and Kazuhito Murakami, *"Contact Lens Extraction by Using Thermo-Vision"*, Proc. of the 18th International Conference on Pattern Recognition (ICPR2006), Vol.4, pp.570-573, Hong Kong/China, 2006/08

Chapter 8
Conclusion

8.1 Introduction

In this chapter, I would like to summarize the whole contents of this research work, its contribution, the advantages of the proposed methods and the further extension. At first, I analyzed the problems encountered in real time system and real-time image processing system (RTIP). Then, I constructed the adaptive scheduling mechanism which is modified from the conventional imprecise scheduling model using the concept of anytime algorithm for the optimization of the overall processing result under time constraint. After that, I categorized some of the conventional image processing operations and its related procedures from the viewpoint of anytime algorithm.

Some of the conventional image processing (CIP) method is converted to Anytime Algorithmic Image Processing (AAIP) method by the concept of anytime algorithm. Then, AAIP tasks such as AA noise reduction, AA edge detection, AA thinning, AA erosion and AA boundary detection are implemented as an example for the construction of anytime algorithmic image processing library. Evaluation of the quality function for each procedure i.e., performance profile calculation in AAIP are described case by case to present the results of each modified procedure by using the cumulative probability distribution function (CDF) from the viewpoint of probability theory.

An adaptive static scheduling method is proposed for the optimization of the overall performance in a restricted time for a system that is composed of many tasks by the concept of imprecise computation and anytime algorithm. Experiments are done for each AAIP procedures, the adaptive scheduling for the task assignment and scheduling, and the scheduling of image processing tasks. The experimental results show that the intermediate or the imprecise results of each task can be obtained at intermediate processing time and the overall processing result of the combination of many tasks can be realized under the processing time constraint. The quality function to optimize the overall performance in a restricted time is defined by using linear programming model and the concept of imprecise computation. The contact lens extraction algorithm is also actualized for the hard lens and soft lens by using thermo-vision image as a part of this research work.

8.2 Contribution

The model of the quality functions offers a methodological contribution to the field of system planning and scheduling in operation research in general. Modification of CIP to AAIP methods offers a methodological and practical contribution to the construction of image processing library by the concept of anytime algorithm. The main aspects of this contribution are summarized as follows:

(1) AAIP method

This method can provide the possible imprecise results by giving a range answer with different qualities for each task under the condition that the processing time is restricted. It can perform the task partially if the processing time is restricted and it can perform the task completely if the processing time is enough or more. Moreover, the combination of many tasks can also be performed under the processing time constraint by giving the range answer of overall processing result with different qualities. The quality of overall processing result improves when the processing time increases satisfied by the properties of anytime algorithm. Thus, the proposed AAIP method is useful for the RTIP system encountered in time and quality trade-off problem. In addition, it can optimize the overall performance of image processing result under time restriction as well as reducing the idle processing time.

(2) Adaptive scheduling method

The adaptive scheduling method can schedule the combination of tasks for a system under time constraint by distributing the time allocation for each task. It can reduce the idle/rest processing time and it can optimize the total (overall) performance in a restricted time. Furthermore, it can schedule the combination of many tasks under time restriction for the optimization of overall processing result using low performance machine. The utilization of this method is the problem solving of optimal decision making by scheduling with limited resources (processing time) in operation research, in artificial intelligence, and in engineering.

8.3 Discussion

From my research work, I have known and understood the facts that anytime algorithm can be applied to solve the fundamental problems of digital image processing tasks which is in discrete type in order to obtain the intermediate or imprecise result in the restricted time from the view point of the image quality and/or processing time. Some of Conventional Image Processing (CIP) methods in low, mid, and high level could be converted to Anytime Algorithmic Image Processing (AAIP) according to the definition and properties of anytime algorithm. Moreover, the constructed AAIP methods have the advantages that the intermediate result can be produced and the overall processing result of the combination of many tasks can be realized during the processing time. In addition, the adaptive scheduling can be performed for the system which is composed of many tasks that are applied by AAIP methods under a condition that the processing time is restricted in order to obtain the overall processing result. Thus, I conclude that this proposed idea can extend and support for the system that has many tasks for e.g., embedded system, object tracking system, and image transmission system and others in a typical real-time system.

8.4 Future Work

This research work can be extended for other image processing methods like histogram equalization, morphological image processing by anytime algorithm for the construction of anytime algorithmic image processing library from the viewpoint of image processing quality and processing time. Realize and analyze the suitable system for the specific applications which can be applied by the modified AAIP procedures and adaptive scheduling method. In addition, the combination of linear programming model and anytime algorithm to solve the optimization problems under time constraint are the extensions of this research work.

Bibliography

[1] T. Dean and M. Boddy, "An analysis of time-dependent planning", Proc. AAAI-88, pp.49-54, AAAI, 1988.

[2] S. Zilberstein, "Using anytime algorithms in intelligent systems", AI Magazine, vol. 17, no. 3, pp. 73-83, 1996.

[3] S. Zilberstein and S. J. Russell. In S. Natarajan ed., Approximate reasoning using anytime algorithms, imprecise and approximate computation, Kluwer Academic Publishers, 1995.

[4] J. Grass and S. Zilberstein. In M. Pittarelli ed., Anytime algorithm development tools, SIGART Bulletin Special Issue on Anytime Algorithms and Deliberation Scheduling, 7(2):20-27, 1996.

[5] S. Zilberstein Ph.D dissertation, Operational rationality through compilation of anytime algorithm, Computer Science Division, University of California at Berkeley, 1993.

[6] M. Seul, L. O'Gorman, and M.J. Sammon, Practical algorithms for image analysis, Cambridge University Press, NY, 2005.

[7] Rafael C. Gonzalez and Richard E. Woods, Digital image processing, Prentice Hall, 2nd Edition, 2002.

[8] Harley R. Myler and Arthur R. Weeks, The handbook of image processing algorithms in C, Prentice-Hall PTR, 1993.

[9] Hasegawa, H. Kubota, and J. Toriwaki, "Automated construction of image processing procedures by sample figure presentation", Proc. of 8th Int'l Conf. on Pattern Recognition (ICPR1986), Vol. 1, pp.586-588, Oct. 1986.

[10] D. Fujiwara and K. Murakami, "On the scheduling of the image processing using anytime algorithm", Proc. of MIRU2004, Vol.1, pp.87-92, July 2004. (in Japanese).

[11] W. W. Kywe, D. Fujiwara, and K. Murakami, "Scheduling of image processing using anytime algorithm for real-time system", Proc. of the 18th Int'l Conf. on Pattern Recognition (ICPR2006), Vol.3, pp.1095-1098, Hong Kong / China, Aug. 2006.

[12] M. Last, A. Kandel, O. Maimon, E. Eberbach, "Anytime Algorithm for Feature Selection", Department of Computer Science and Engineering, University of South Florida, 4202 E. Fowler Avenue, ENB 118, Tampa, FL 33620, USA.

[13] M. Boddy and T. Dean. *"Decision-theoretic deliberation scheduling for problem solving in time-constrained environments",* Artificial Intelligence, 67(2):245--286, 1994.

[14] A. Garvey, V. Lesser. *"Design-to-Time Real-Time Scheduling",* IEEE Transactions on Systems, Man and Cybernetics, 1993
ftp://ftp.cs.umass.edu/pub/lesser/garvey-dtt-smc.ps

[15] Garvey, Alan and Victor Lesser. *"Design-to-time Scheduling and Anytime Algorithms"*, SIGART Bulletin, 7 (2):16--19 (1996).
http://citeseer.comp.nus.edu.sg/85186.html

[16] W. Zhao, K. Ramamritham and J.A. Stankovic, ''Preemptive Scheduling under Time and Resource Constraints'', *IEEE Transactions on Computers* 38(8), pp. 949-960 (August 1987).

[17] K. Momiyama, K. Kato, and K. Murakami, "Contact Lens Extraction by Using Thermo-Vision", *Proc. of the 10th Symposium on Sensing via Image Information (SSII 2004),*Yokohama,June2004,pp.471-476(in Japanese).

[18] M. Yoshida and K. Murakami, "A method to extract contact lens by using image sequences of thermo-vision", *ITE Technical Report,* Vol.29, No.46 (ME2005-116), Hokkaido, Aug.2005, pp.41-44 (in Japanese).

[19] Harley R. Myler and Arthur R. Weeks, "The handbook of image processing algorithms in C", Prentice-Hall PTR, 1993.

[20] J.R. Parker, "Algorithms for image processing and computer vision", John Wiley & Sons, Inc. U.S.A, 1997.

[21] I. Pitas, "Digital Image Processing Algorithms and Applications", John Wiley & Sons, Inc. U.S.A, 2000.

[22] Anil K. Jain, "Fundamentals of Digital Image Processing", Prentice-Hall, Inc. U.S.A, 1989.

[23] http://www.profc.udec.cl

[24] http://www.eng.iastate.edu/ee528/sonkamaterial/chapter_1.htm

[25] http://www.eng.iastate.edu/ee528/sonkamaterial/chapter_2.htm#Image%20functions

[26] http://en.wikipedia.org/wiki/

[27] http://web.uct.ac.za/depts/physics/laser/hanbury/intro_ip.html

[28] http://moab.eecs.wsu.edu/~cs445/Lecture_1.pdf#search='picture%20of%20digital %20image%20processing

[29] http://www.aprs.org.au/dicta2002/dicta2002_proceedings/Yin258.pdf

[30] http://cs-alb-pc3.massey.ac.nz/notes/59318/l9.html

[31] http://archive.org/details/Lectures_on_Image_Processing

[32] http://ia700307.us.archive.org/7/items/Lectures_on_Image_Processing/

[33] http://www.engineering.uiowa.edu/~dip/LECTURE/lecture.html

[34] http://www.engineering.uiowa.edu/~dip/LECTURE/PreProcessing3.html

[35] http://www.math.tau.ac.il/~turkel/notes.html

[36] http://homepages.inf.ed.ac.uk/rbf/HIPR2/hipr_top.htm

[37] http://www.algonet.se/~staffann/developer/realtimeintro.htm

[38] http://www.roborealm.com/help/Convolution.php

[39] http://www-rohan.sdsu.edu/doc/matlab/toolbox/images/morph3.html

[40] http://www.cs.cf.ac.uk/Dave/Vision_lecture/node50.html

[41] http://ronbigelow.com/articles/sharpen1/sharpen1.htm

[42] http://terpconnect.umd.edu/~toh/spectrum/Smoothing.html

LIST OF PUBLICATIONS

Journal Publication

1. Wyne Wyne Kywe and Kazuhito Murakami: "*An Approach to Linear Spatial Filtering Method based on Anytime Algorithm for Real-time Image Processing*", Journal of Computing Press, NY, USA, ISSN 2151-9617, Volume 4, Issue 12, pp.26-32, December 2012, impact factor:0.21.
   https://WWW.JOURNALOFCOMPUTING.ORG

International Conference Proceedings

1. Wyne Wyne Kywe, Daisuke Fujiwara, and Kazuhito Murakami: *"Scheduling of Image Processing Using Anytime Algorithm for Real-time System"*, Proc. of the 18th International Conference on Pattern Recognition (ICPR2006), Vol.3, pp.1095-1098, HongKong/China, 2006/08. IEEE Computer Society, 2006
   ISSN : 1051-4651, Print ISBN: 0-7695-2521-0, doi : 10.1109/ICPR.2006.1029
2. Wyne Wyne Kywe, Masashi Yoshida and Kazuhito Murakami: *"Contact Lens Extraction by Using Thermo-Vision"*, Proc. of the 18th International Conference on Pattern Recognition (ICPR2006), Vol.4, pp.570-573, HongKong/China, 2006/08
3. Wyne Wyne Kywe and Kazuhito Murakami: *"Anytime Noise Reduction and Edge Detection Algorithms for Time-Restricted Image Processing System"*, Proc. of the 15th Japan-Korea Joint Workshop on Frontiers of Computer Vision (FCV 2009), pp.65-70, Andong/Korea, 2009/02
4. Wyne Wyne Kywe and Kazuhito Murakami: *"New Approach to Image Processing Methods by Anytime Algorithm for the Overall Result under Time Constraint"*, Proc. of the International Workshop on Advanced Image Technology (IWAIT 2010), Kuala Lumpur/Malaysia, 2010/01

Others

1. Wyne Wyne Kywe and Kazuhito Murakami : *"Minimization of the Discarded Optional Sub-tasks and the Realization of the Overall Processing Result for the Task Assignment and Scheduling by Imprecise Computation"*, The Operations Research Society of Japan Chubu Branch, 38th Chubu Branch Workshop Abstracts, pp. 37-40, Central Quality Management Association / Nagoya, Japan, 2011/3
2. Wyne Wyne Kywe, Daisuke Ukai and Kazuhito Murakami
   "Feature Extraction from Thermo-image for Identification", 2013 International Workshop on Smart Info-Media Systems in Asia (SISA2013)
   Aichi Industry & Labor Center, Nagoya, Japan, Sep. 30 - Oct. 2, 2013