Comparison Analysis of CPU Scheduling : FCFS, SJF and Round Robin

Andysah Putera Utama Siahaan
Universitas Pembangunan Panca Budi
Jl. Jend. Gatot Subroto Km. 4,5 Sei Sikambing, 20122, Medan, Sumatera Utara, Indonesia

***Abstract* - Task scheduling is needed to maintain every process that comes with a processor in parallel processing. In several conditions, not every algorithm works better on the significant problem. Sometimes FCFS algorithm is better than the other in short burst time while Round Robin is better for multiple processes in every single time. However, it cannot be predicted what process will come after. Average Waiting Time is a standard measure for giving credit to the scheduling algorithm. Several techniques have been applied to maintain the process to make the CPU performance in normal. The objective of this paper is to compare three algorithms, FCFS, SJF, and Round Robin. The target is to know which algorithm is more suitable for the certain process.**

***Index Term* - FCFS, SJF, Round Robin, Schedule, Operating System.**

I. INTRODUCTION

Scheduling is already part of a parallel process [1]. In scheduling, there are several methods used to perform queue process that comes to the processor. Some algorithms are popular among other First Come First Serve, Shortest Job First, and Round Robin. In this study, The discussion involves the comparison of the average waiting time of each of these algorithms. The purpose of this comparison to determine what algorithm is more suitable for some processes that are in the ready queue. The priority of the processes that occur on the processor is something that determines when the process will be done. However, it does not discuss the priority in this study. The scheduling assumes all of which occurred in the queue have the same priority value.

II. THEORIES

The process is the state when a program is executed [12]. When the computer is running, there are many processes running simultaneously. A process may create a derivative process is carried out by a parent process. The derivation process is also able to create a new process so that all of these processes ultimately forming process tree. When a process is made then the process can obtain these resources such as CPU time, memory, files, or I/O devices [2][4]. These resources can be obtained directly from the operating system, from the parent process that dispenses resources to each process its derivative, or the derivative and the parent process share the resources of a given operating system. CPU scheduling is part of a multi-programming operating system. Job scheduling is to move the CPU work among the processes[3]. This serves to make the computer work more productive. A process generally consists of two cycles of Burst I/O and CPU Burst performed alternately until the process is complete.

FCFS or FIFO can be defined as a process that arrives first will be served first. If there is a process to arrive at the same time, the services they carried through their order in the queue[5][10]. The process in the queue behind had to wait until all the process in front of him is complete. Any process that is on ready status put in FCFS queue according to the time of arrival. Figure 1 shows how the FCFS works.

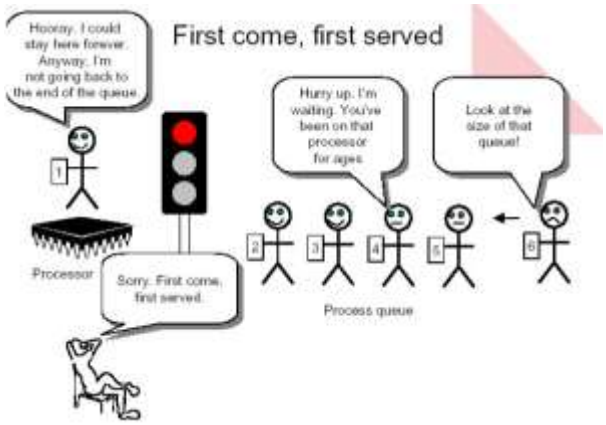


Fig. 1 Illustration of FCFS

The process which has been done does enter the queue anymore. However, here the process which burst time is short must wait for the other process to be finished.

SJF algorithm is a special case of priority scheduling. Each process is equipped with a priority number that is burst time. The CPU is allocated to the process that has the highest priority (smallest integer value is usually the biggest priority) [11]. If several processes have the same priority, then it will use FCFS algorithm. Scheduling priority consists of two schemes, non-preemptive and preemptive. If there is a process P1 comes on when P0 is running, it will be a priority P1. Suppose P1 greater priority than the priority P0; then the non-preemptive, fixed algorithm will complete P0 to the end of its CPU burst, and put P1 in the head position in the queue. While in preemptive, P0 will be stopped first, and replace the CPU allocated to P1. Figure 2 shows the waiting time calculation of SJF.

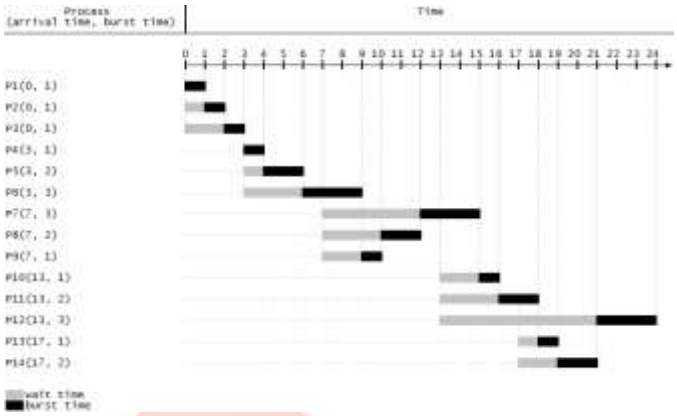


Fig. 2 Illustration of Shortest Job First

In Figure 2, the smallest burst time is 1. However, there are many processes which burst times are 1. Those processes are not in the same ready queue. The ready queue will be the same arrival time such as P1-P3, P4-P6 and so on. Let's see the P7-P9; the smallest burst time is in P9 so that P9 will be performed first. However, in P4-P6, the FCFS is used.

In Round Robin concept, the algorithm uses time-sharing. The algorithm is same as FCFS, but it is preemted [7]. Each process gets CPU time called a quantum time to limit the processing time, typically 1-100 milliseconds. After time runs out, the process is delayed and added to the ready queue. If a process has a CPU burst is smaller than the time quantum, then the process will release the CPU if it has finished its work so that the CPU can be immediately used by the next process[8][9]. Conversely, if a process has a CPU burst greater than the time quantum, the process will be suspended if it reaches a quantum of time, and then queuing back to the position of the tail of the ready queue and then execute the next process. If there are n processes in the ready queue and the time quantum q, then each process of getting 1/n of the CPU time at most q time units at once CPU scheduling. No process waits for more than (n-1) q time units. Round Robin algorithm performance can be explained as the following figure when the queue is large, then the algorithm used is FCFS, but if the queue is small then frequent context switches is happening.

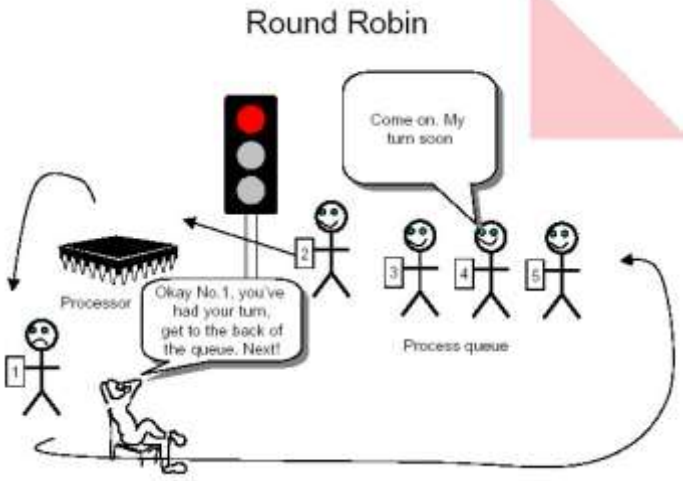


Fig. 3 Illustration of Round Robin

III. TESTING AND ANALYSIS

The creation of data sampling process must be established. Some of them have the same arrival time and different burst time. Table 1 shows the data in the process.

Table 1 Data sampling.

Process	Arrival Time	Burst Time
P1	0	1
P2	0	1
P3	0	1
P4	3	1
P5	3	2
P6	3	3
P7	7	3
P8	7	2
P9	7	1
P10	13	1
P11	13	2
P12	13	3
P13	17	5
P14	17	2
P15	19	4
P16	21	7
P17	23	3
P18	24	5
P19	24	1
P20	24	3
P21	24	5
P22	24	4
P23	25	8
P24	25	4
P25	26	3

Let's consider that Process is P, Arrival Time is AT, Burst Time is BT, Waiting Time is WT. The formula below is to obtain the waiting time.

$$\boldsymbol{WTn} = \left(\sum_{\boldsymbol{i=1}}^{\boldsymbol{n-1}} \boldsymbol{BT}\right) - \boldsymbol{ATn} \qquad (1)$$

$$\boldsymbol{TWT} = \sum \boldsymbol{WT} \qquad (2)$$

First Come First Serve

Every single waiting time must be calculated. Then calculate the Total Waiting Time and the Average Waiting Time of the FCFS algorithm.

WT1 = $P1_{AT}$
= 0
WT2 = $P1_{BT}$ - $P2_{AT}$
= 1 - 0
= 1
WT3 = $P1_{BT}$ + $P2_{BT}$ - $P3_{AT}$
= 1 + 1 - 0
= 2

WT4 = $P1_{BT}$ + $P2_{BT}$ + $P3_{BT}$ – $P4_{AT}$
= 1 + 1 + 1 - 3
= 0
WT5 = $P1_{BT}$ + $P2_{BT}$ + $P3_{BT}$ + $P4_{BT}$ – $P5_{AT}$
= 1 + 1 + 1 + 1 - 3
= 1

WT25 = $P1_{BT}$ + $P2_{BT}$ + $P3_{BT}$ + .. + $P24_{BT}$ – $P25_{AT}$
= 1 + 1 + 1 +1 + .. + 4 – 26
= 46

The calculation continues until the last process. It produces the total waiting time and finally the average waiting time is obtained.

TWT = WT1 + WT2 + WT3 + WT4 + WT5 + ... + WT25
= 0 + 1 + 2 + 0 + 1 + ... + 46
= 328

AWT = TWT / TOTAL PROCESS
= 328 / 25
= **13.12**

Figure 4 shows the time split which represents each process of the FCFS algorithm.

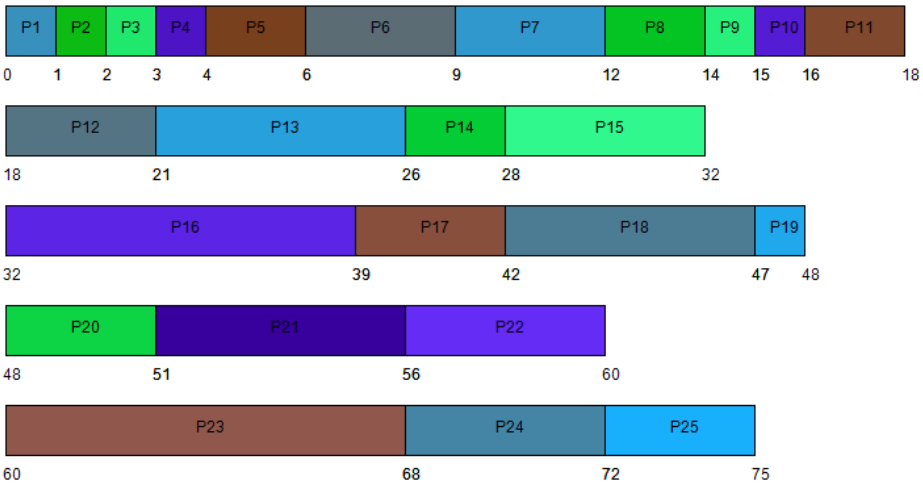


Fig. 4 : Process of FCFS

Shotest Job First

The second calculation is using SJF algorithm. It is FCFS concept, but the process which comes at the same time or the same ready queue must be sorted in ascending order. The smallest burst time will be the first order while the bigger will be the last one. The table must be reconstructed in SJF model. The same arrival time must be sorted from low to high. After being changed, the form of the process can be seen in Table 2. The highlighted rows are the processes which have been sorted.

Table 2 Data sampling of SJF.

Process	Arrival Time	Burst Time
P1	0	1
P2	0	1
P3	0	1
P4	3	1
P5	3	2
P6	3	3
P9	7	1
P8	7	2
P7	7	3
P10	13	1
P11	13	2
P12	13	3
P14	17	2
P13	17	5
P15	19	4
P16	21	7
P17	23	3
P19	24	1

P20	24	3
P22	24	4
P21	24	5
P18	24	5
P24	25	4
P23	25	8
P25	26	3

Afterward, it proceed to calculate the Total Waiting Time and the Average Waiting Time of the SJF algorithm.

$WT1 = P1_{AT}$
$= 0$
$WT2 = P1_{BT} - P2_{AT}$
$= 1 - 0$
$= 1$
$WT3 = P1_{BT} + P2_{BT} - P3_{AT}$
$= 1 + 1 - 0$
$= 2$
$WT4 = P1_{BT} + P2_{BT} + P3_{BT} – P4_{AT}$
$= 1 + 1 + 1 - 3$
$= 0$
$WT5 = P1_{BT} + P2_{BT} + P3_{BT} + P4_{BT} – P5_{AT}$
$= 1 + 1 + 1 + 1 - 3$
$= 1$
$WT25 = P1_{BT} + P2_{BT} + P3_{BT} + .. + P24_{BT} – P25_{AT}$
$= 1 + 1 + 1 +1 + .. + 4 – 26$
$= 46$

The calculation continues until the last process. Then it produces the total waiting time and finally the average waiting time is obtained.

$TWT = WT1 + WT2 + WT3 + WT4 + WT5 + ... + WT25$
$= 0 + 1 + 2 + 0 + 1 + ... + 46$
$= 309$

$AWT = TWT / TOTAL PROCESS$
$= 328 / 25$
$= \mathbf{12.36}$

Figure 5 shows the time split which represents each process of the SJF algorithm.

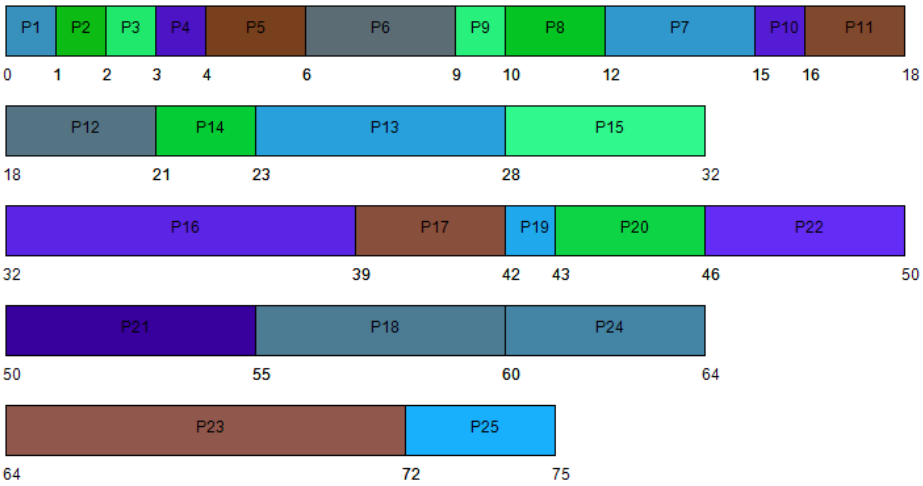


Fig. 5 Process of SJF

Round Robin

The calculation in Round Robin is more difficult than the earlier calculations. In this algorithm, it needs to split the best time into several subprocesses in a period. Quantum Time is a time slicing technique to split burst time into several subprocesses. Testing the same data with different quantum times is giving the different AWT.

Quantum Time = 2
Total Waiting Time = 523
Average Waiting Time = **20.92**

Quantum Time = 3
Total Waiting Time = 422
Average Waiting Time = **16.88**

Quantum Time = 5
Total Waiting Time = 346
Average Waiting Time = **13.84**

If the quantum time value bigger than the burst times available, the Round Robin turns to FCFS algorithm.

Quantum Time = 10
Total Waiting Time = 328
Average Waiting Time = **13.12**

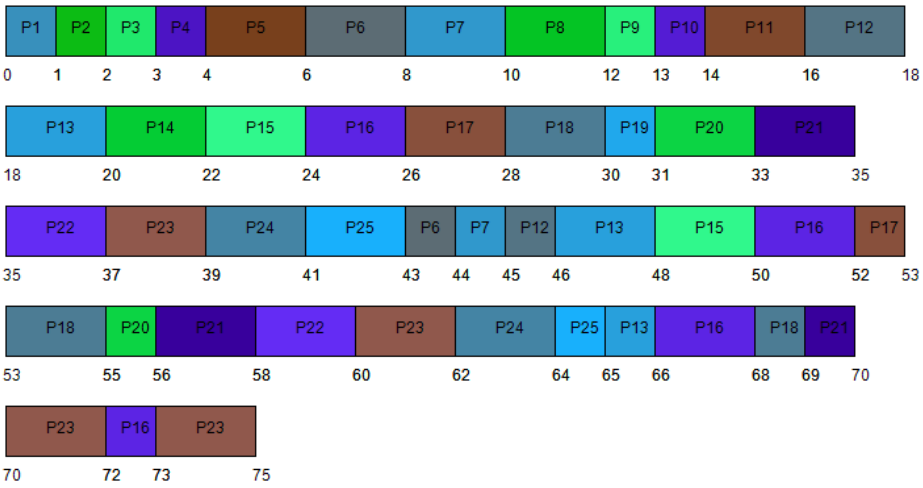


Fig. 6 Process of Round Robin Quantum Time = 2

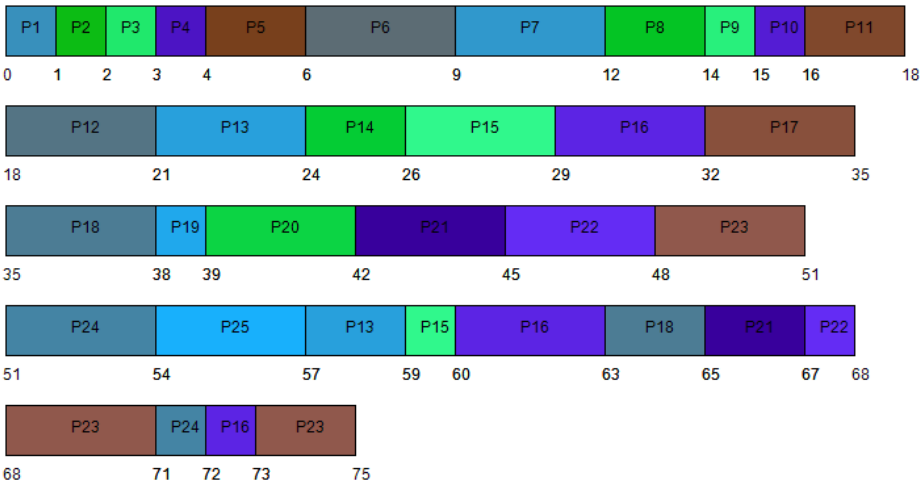


Fig. 7 Process of Round Robin Quantum Time = 3

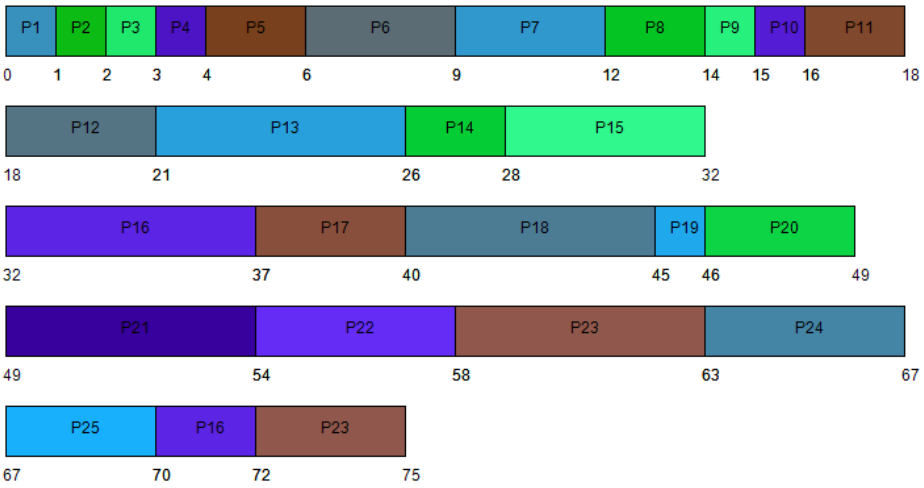


Fig. 8 Process of Round Robin Quantum Time = 5

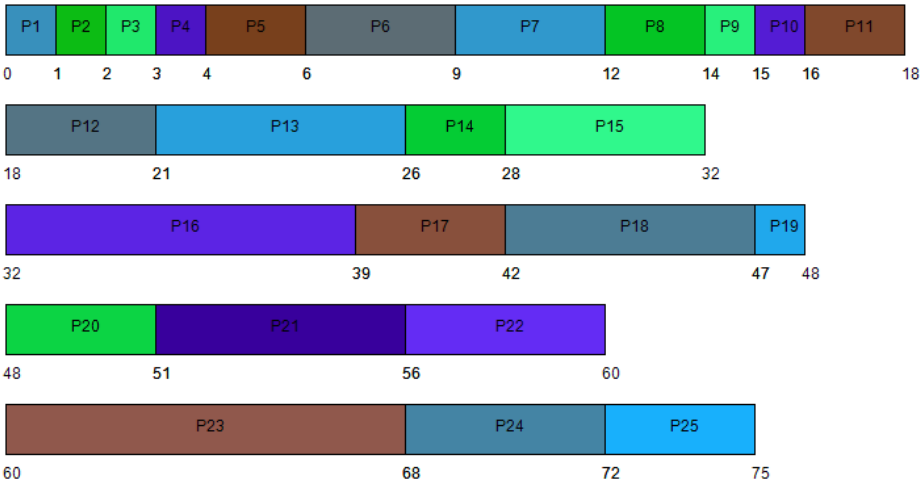


Fig. 9 Process of Round Robin Quantum Time = 10

Figure 6 to 9 show the time split which represents each process by doing Round Robin algorithm in different quantum time values. Quantum time influences the result of the average waiting time. If the quantum time bigger, the average waiting time will result in the small value, but the processes those have been in the ready queue will take more time to get a turn.

IV. CONCLUSION

The calculation of three algorithms shows the different average waiting time. The FCFS is better for a small burst time. The SJF is better if the process comes to processor simultaneously. The last algorithm, Round Robin, is better to adjust the average waiting time desired. Round Robin quantum time will set it towards more SJF or FCFS value. All algorithm is good, but the speed of the process depends on the processor load.

REFERENCES

[1] H. Heidari dan A. Chalechale, "Scheduling in Multiprocessor System Using Genetic Algorithm," *International Journal of Advanced Science and Technology*, vol. 43, pp. 81-94, 2012.

[2] C. Bajaj, A. Dogra dan G. Singh, "Review And Analysis Of Task Scheduling Algorithms," *International Research Journal of Engineering and Technology*, vol. 2, no. 3, pp. 1449-1452, 2015.

[3] N. Hamid, "Scheduling Policies," [Online]. Available: http://homepages.uel.ac.uk/u8902383/scheduling_policies.htm.

[4] Arpaci-Dusseau, "Shortest Job Next," 12 April 2016. [Online]. Available: https://en.wikipedia.org/wiki/Shortest_job_next. [Diakses 1 May 2016].

[5] A. Dusseau, R. H. dan A. C., Operating Systems: Three Easy Pieces, Arpaci-Dusseau Books, 2014.

[6] Tanenbaum, Modern Operating Systems, Pearson Education, Inc., 2008.

[7] A. Noon, A. Kalakech dan S. Kadry, "A New Round Robin Based Scheduling Algorithm for Operating Systems: Dynamic Quantum Using the Mean Average," *IJCSI International Journal of Computer Science Issues,* vol. 8, no. 3, pp. 224-229, 2011.

[8] I. S. Rajput dan D. Gupta, "A Priority based Round Robin CPU Scheduling Algorithm for Real Time Systems," *International Journal of Innovations in Engineering and Technology,* vol. 1, no. 3, pp. 1-11, 2012.

[9] R. Shyam dan S. K. Nandal, "Improved Mean Round Robin with Shortest Job First Scheduling," *International Journal of Advanced Research in Computer Science and Software Engineering,* vol. 4, no. 7, pp. 170-179, 2014.

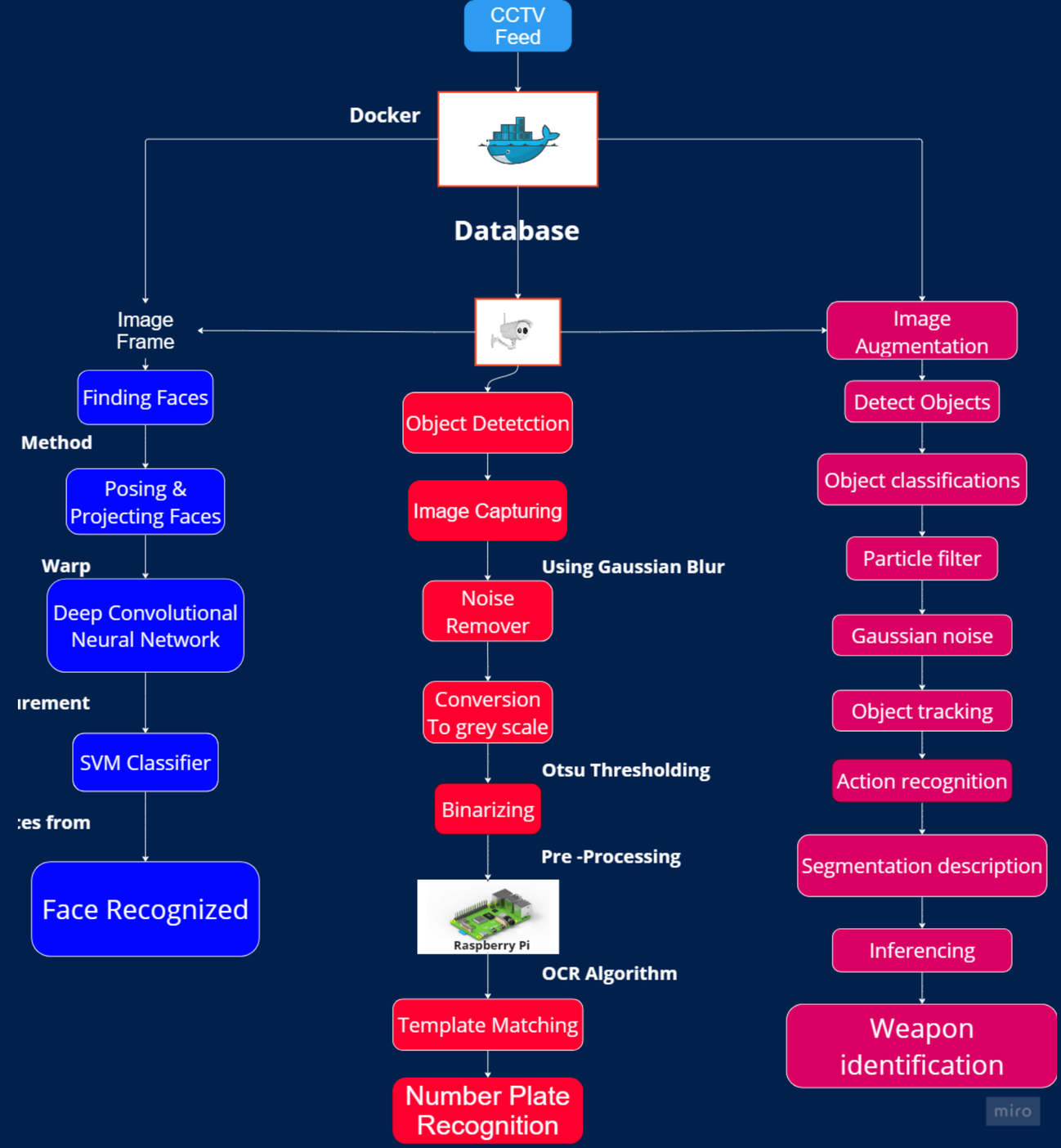
[10] Y. Adekunle, Z. Ogunwobi dan A. S. Jerry, "A Comparative Study of Scheduling Algorithms for Multiprogramming in Real-Time Systems," *International Journal of Innovation and Scientific Research,* vol. 12, no. 1, pp. 180-185, 2014.

[11] P. Singh, V. Singh dan A. Pandey, "Analysis and Comparison of CPU Scheduling Algorithms," *International Journal of Emerging Technology and Advanced Engineering,* vol. 4, no. 1, pp. 91-95, 2014.

[12] A. P. U. Siahaan, "Penyelarasan Pada Masalah Dining Philosophers Menggunakan Algoritma Lock & Release," *Techsi,* vol. 6, no. 1, pp. 14-18, 2015.

BIOGRAPHY

Andysah Putera Utama Siahaan was born in Medan, Indonesia, in 1980. He received the S.Kom. degree in computer science from Universitas Pembangunan Panca Budi, Medan, Indonesia, in 2010, and the M.Kom. in computer science as well from the University of Sumatera Utara, Medan, Indonesia, in 2012. In 2010, he joined the Department of Engineering, Universitas Pembangunan Panca Budi, as a Lecturer, and in 2012 became a junior researcher. He is applying for his Ph. D. degree in 2016. He has written in several international journal and conference. He is now active in writing papers and joining conferences.


CCTV Feed
Docker
Database
Image Frame
Finding Faces
Method
Posing & Projecting Faces
Warp
Deep Convolutional Neural Network
ırement
SVM Classifier
:es from
Face Recognized
Object Detetction
Image Capturing
Using Gaussian Blur
Noise Remover
Conversion To grey scale
Otsu Thresholding
Binarizing
Pre -Processing
Raspberry Pi
OCR Algorithm
Template Matching
Number Plate Recognition
Image Augmentation
Detect Objects
Object classifications
Particle filter
Gaussian noise
Object tracking
Action recognition
Segmentation description
Inferencing
Weapon identification
miro