

*Seminar II*

**IMPLEMENTASI FILTER SPASIAL LINEAR PADA VIDEO STREAM  
MENGGUNAKAN *FPGA HARDWARE ACCELERATOR***



Oleh  
**SULAEMAN**  
H131 16 002

Pembimbing Utama : Dr. Eng. Armin Lawi, S.Si., M.Eng.  
Pembimbing Pertama : Supri Bin Hj Amir, S.Si., M.Eng.  
Pengaji : 1. Dr. Hendra, S.Si., M.Kom.  
                  2. Nur Hilal A Syahrir, S.Si., M.Si.

**PROGRAM STUDI ILMU KOMPUTER  
DEPARTEMEN MATEMATIKA  
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM  
UNIVERSITAS HASANUDDIN  
MAKASSAR**

**2021**

# **DAFTAR ISI**

<b>DAFTAR ISI . . . . .</b>	<b>ii</b>
<b>DAFTAR TABEL . . . . .</b>	<b>iii</b>
<b>DAFTAR GAMBAR . . . . .</b>	<b>v</b>
<b>BAB I PENDAHULUAN . . . . .</b>	<b>1</b>
1.1 Latar Belakang . . . . .	1
1.2 Rumusan Masalah . . . . .	3
1.3 Tujuan Penelitian . . . . .	3
1.4 Batasan Masalah . . . . .	3
1.5 Manfaat Penelitian . . . . .	3
<b>BAB II TINJAUAN PUSTAKA . . . . .</b>	<b>4</b>
2.1 Landasan Teori . . . . .	4
2.1.1 Citra Digital . . . . .	4
2.1.2 Pengolahan Citra Digital . . . . .	5
2.1.3 Filter Spasial . . . . .	7
2.1.4 Kernel . . . . .	8
2.1.5 Konvolusi . . . . .	9
2.1.6 Video Stream . . . . .	11
2.1.7 FPGA . . . . .	11
2.1.8 Evaluasi Performa . . . . .	12
2.2 Penelitian Terkait . . . . .	17
2.2.1 Spatial Filtering Based Boundary Extraction in Underwater Images for Pipeline Detection: FPGA Implementation . . . . .	17
2.2.2 FPGA Implementation of Spatial Filtering techniques for 2D Images . . . . .	17
2.2.3 Features of Image Spatial Filters Implementation on FPGA . . . . .	18

2.2.4	An FPGA-Oriented Algorithm for Real-Time Filtering of Poisson Noise in Video Streams, with Application to X-Ray Fluoroscopy . . . . .	18
2.2.5	A real-time video denoising algorithm with FPGA implementation for Poisson-Gaussian noise . . . . .	18
<b>BAB III METODE PENENILITIAN</b>	.....	<b>20</b>
3.1	Tahapan Penelitian .....	20
3.2	Waktu dan Lokasi Penelitian .....	21
3.3	Rancangan Sistem .....	21
3.4	Instrumen Penelitian .....	22
<b>BAB IV HASIL DAN PEMBAHASAN</b>	.....	<b>23</b>
4.1	Implementasi pada FPGA Development Board .....	23
4.1.1	Penerapan Filter Spasial .....	23
4.1.2	Penerapan Filter Spasial menggunakan Prosesor ARM dan FPGA .....	27
4.1.3	Proses Evaluasi Kinerja .....	28
4.2	Analisis Kinerja .....	31
4.2.1	Waktu Komputasi .....	31
4.2.2	Frame Rate (FPS) .....	33
4.2.3	Penggunaan CPU .....	34
4.2.4	Penggunaan Memory .....	35
4.2.5	Resident Memory (RES) .....	36
4.2.6	Shared Memory (SHR) .....	38
4.2.7	Virtual Memory (VIRT) .....	39
<b>BAB V KESIMPULAN DAN SARAN</b>	.....	<b>41</b>
5.1	Kesimpulan .....	41
5.2	Saran .....	41
<b>DAFTAR PUSTAKA</b>	.....	<b>42</b>
<b>LAMPIRAN</b>	.....	<b>43</b>

## **DAFTAR TABEL**

4.1	Tabel perbandingan waktu komputasi dengan menggunakan 50 frame.	32
4.2	Tabel perbandingan waktu komputasi dengan menggunakan 200 frame.	33
4.3	Tabel perbandingan FPS dengan menggunakan prosesor ARM dan FPGA. . . . .	34
4.4	Tabel perbandingan penggunaan CPU dengan menggunakan prosesor ARM dan FPGA. . . . .	35
4.5	Tabel perbandingan penggunaan memory dengan menggunakan prosesor ARM dan FPGA. . . . .	36
4.6	Tabel perbandingan penggunaan resident memory (RES) dengan menggunakan prosesor ARM dan FPGA. . . . .	37
4.7	Tabel perbandingan penggunaan shared memory (SHR) dengan menggunakan prosesor ARM dan FPGA. . . . .	38
4.8	Tabel perbandingan penggunaan virtual memory (VIRT) dengan menggunakan prosesor ARM dan FPGA. . . . .	39

## DAFTAR GAMBAR

2.1 (a) Contoh citra biner, (b) contoh citra grayscale, (c) contoh citra warna. . . . .	5
2.2 Ilustrasi konvolusi pada citra. Sumber: <a href="https://indoml.com">https://indoml.com</a> . . . . .	10
2.3 Struktur FPGA. . . . .	12
2.4 FPGA Board Xilinx PYNQ-Z2. . . . .	13
2.5 Tampilan program top pada Linux. . . . .	14
2.6 Kuadran pembagian memory pada Linux. . . . .	15
3.1 Flowchart tahapan penelitian. . . . .	20
3.2 Rancangan sistem. . . . .	21
4.1 Contoh Frame Grayscale. . . . .	24
4.2 Hasil filter Average Blur. . . . .	25
4.3 Hasil filter Gaussian Blur. . . . .	25
4.4 Hasil filter Laplacian. . . . .	26
4.5 Hasil filter Sharpening. . . . .	26
4.6 Hasil filter Sobel Horizontal. . . . .	27
4.7 Hasil filter Sobel Vertical. . . . .	27
4.8 Menghitung waktu komputasi dengan library time di Python. . . . .	29
4.9 Tampilan program <b>top</b> . . . . .	29
4.10 Menampilkan PID sebuah proses dengan bahasa pemrograman Python. . . . .	30
4.11 Menjalankan program <b>top</b> kemudian menyimpan hasilnya pada file arm-laplacian1.txt. . . . .	30
4.12 Potongan isi file arm-laplacian1.txt. . . . .	31
4.13 Isi file arm-laplacian1.csv. . . . .	32
4.14 Grafik perbandingan waktu komputasi dengan 50 frame dan 200 frame . . . . .	32
4.15 Grafik perbandingan waktu komputasi dengan 50 frame dan 200 frame . . . . .	33
4.16 Grafik perbandingan FPS dengan menggunakan prosesor ARM dan FPGA. . . . .	34
4.17 Grafik perbandingan penggunaan CPU dengan menggunakan prosesor ARM dan FPGA. . . . .	35

4.18 Grafik perbandingan penggunaan memory dengan menggunakan prosesor ARM dan FPGA. . . . .	36
4.19 Grafik perbandingan penggunaan resident memory (RES) dengan menggunakan prosesor ARM dan FPGA. . . . .	37
4.20 Grafik perbandingan penggunaan shared memory (SHR) dengan menggunakan prosesor ARM dan FPGA. . . . .	38
4.21 Grafik perbandingan penggunaan virtual memory (VIRT) dengan menggunakan prosesor ARM dan FPGA. . . . .	39

# BAB I

## PENDAHULUAN

### 1.1 Latar Belakang

Citra digital merupakan citra yang dihasilkan dari pengolahan secara digital dengan merepresentasikan citra secara numerik dengan nilai-nilai diskrit. Suatu citra digital dapat direpresentasikan dalam bentuk matriks dengan fungsi  $f(x,y)$  yang terdiri dari M kolom dan N baris. Perpotongan antara baris dan kolom disebut sebagai piksel (*pixel*) (Gonzalez and Woods, 2001).

Pengolahan citra digital merupakan proses mengolah piksel di dalam citra secara digital untuk tujuan tertentu. Berdasarkan tingkat pemrosesannya pengolahan citra digital dikelompokkan menjadi tiga kategori, yaitu: *low-level*, *mid-level* dan pemrosesan *high-level*. Pemrosesan *low-level* dilakukan dengan operasi primitif seperti *image preprocessing* untuk mengurangi derau (*noise*), memperbaiki kontras citra dan mempertajam citra (*sharpening*). Pemrosesan *mid-level* melibatkan tugas-tugas seperti segmentasi atau mempartisi gambar menjadi beberapa bagian atau objek, deskripsi objek untuk dilakukan pemrosesan lanjutan, dan klasifikasi objek yang terdapat dalam citra digital. Pemrosesan *high-level* merupakan proses tingkat lanjut dari dua proses sebelumnya, dilakukan untuk mendapat informasi lebih yang terkandung dalam citra seperti *pattern recognition*, *template matching*, *image analysis* dan sebagainya (Gonzalez and Woods, 2001).

Konsep filter spasial pada pengolahan citra digital berasal dari penerapan transformasi Fourier untuk pemrosesan sinyal pada domain frekuensi. Istilah filter spasial ini digunakan untuk membedakan proses ini dengan filter pada domain frequensi. Proses filter dilakukan dengan cara menggeser filter kernel dari titik ke titik dalam citra digital. Istilah *mask*, *kernel*, *template*, dan *window* merupakan istilah yang sama dan sering digunakan dalam pengolahan citra digital. Dalam penelitian ini penulis menggunakan istilah kernel untuk istilah tersebut. Konsep filter spasial linear mirip seperti konsep konvolusi pada domain frekuensi, dengan alasan tersebut filter spasial linear biasa disebut juga konvolusi sebuah kernel dengan citra digital (Gonzalez and Woods, 2001). Proses filter dalam pengolahan citra digital dilakukan

dengan memanipulasi sebuah citra menggunakan kernel untuk menghasilkan citra yang baru, sehingga dengan kernel yang berbeda maka citra hasil yang didapat juga akan berbeda.

Video stream dapat dipandang sebagai serangkaian citra digital berturut-turut (Zhao, 2015). Berbeda dengan format video lainnya, video stream ini tidak disimpan pada media penyimpanan sebagai file video melainkan setiap *frame* langsung disalurkan dari sumber (*source*) ke penerima, dalam hal ini FPGA. Dengan menganggap Video stream adalah kumpulan citra digital (*frame*) maka dapat dilakukan metode pengolahan seperti pada citra digital, termasuk filter spasial. Setiap citra yang ditangkap dari source disebut sebagai *frame*, setiap *frame* ini dilakukan metode filter spasial kemudian hasilnya ditampilkan secara berkesinambungan sehingga tampak seperti video yang telah difilter.

Frame per second (*fps*) atau *frame rate* adalah banyaknya *frame* yang ditampilkan per detik. Semakin tinggi *fps* sebuah video maka semakin halus pula gerakan yang dapat ditampilkan karena dibentuk dari *frame* yang lebih banyak, namun dengan jumlah *frame* yang lebih besar tentu dibutuhkan juga *resource* yang lebih besar dalam pengolahan video tersebut (Kowalczyk, Przewlocka, and Krvjak, 2018).

Field Programmable Gate Arrays atau FPGA adalah perangkat semikonduktor yang berbasis *matriks configurable logic block* (CLBs) yang terhubung melalui interkoneksi yang dapat diprogram. FPGA dapat diprogram ulang dengan aplikasi atau fungsi yang diinginkan setelah *manufacturing*. Fitur ini yang membedakan FPGA dengan *Application Specific Integrated Circuits* (ASICs), yang dibuat khusus untuk tugas tertentu saja (Xilinx, 2020).

FPGA Xilinx PYNQ-Z2 adalah FPGA *Board* yang digunakan pada penelitian ini secara *official* dapat menerima input video stream dengan resolusi 720p. Setiap *frame* yang diterima dari *source* akan dilakukan proses filter spasial, kemudian hasilnya disalurkan melalui HDMI output untuk kemudian ditampilkan. Video hasil filter spasial yang ditampilkan akan mengalami penurunan *fps*, hal ini disebabkan adanya penambahan jeda waktu komputasi untuk proses filter yang dilakukan pada setiap *frame*. Pada kesempatan ini penulis ingin melakukan "Implementasi Filter Spasial Linear pada Video Stream menggunakan FPGA Hardware Accelerator".

## **1.2 Rumusan Masalah**

Adapun rumusan masalah dalam penelitian ini yaitu:

1. Bagaimana cara implementasi filter spasial linear pada video stream menggunakan FPGA?
2. Bagaimana kinerja FPGA dalam mengimplementasikan filter spasial linear pada video stream?

## **1.3 Tujuan Penelitian**

Adapun tujuan dari penelitian ini yaitu:

1. Mampu melakukan implementasi filter spasial linear pada video stream menggunakan FPGA.
2. Mengetahui kinerja FPGA dalam mengimplementasikan filter spasial linear pada video stream.

## **1.4 Batasan Masalah**

Berikut ini merupakan beberapa batasan dalam penelitian ini.

1. Filter kernel yang digunakan berukuran 3x3.
2. Video stream yang digunakan dalam penelitian ini beresolusi 720p.
3. Setiap frame dari video stream diubah menjadi citra grayscale sebelum dilakukan penerapan filter spasial.
4. FPGA *Board* yang digunakan adalah Xilinx PYNQ-Z2 dengan processor 650MHz dual-core ARM Cortex-A9.

## **1.5 Manfaat Penelitian**

Hasil dari penelitian ini diharapkan dapat memberikan pemahaman tentang penerapan filter spasial pada video stream. Selain itu, penelitian ini juga diharapkan dapat menjadi rujukan untuk melihat kinerja FPGA dalam mengimplementasikan filter spasial linear pada video stream.

## **BAB II**

### **TINJAUAN PUSTAKA**

#### **2.1 Landasan Teori**

##### **2.1.1 Citra Digital**

Citra digital dapat didefinisikan sebagai fungsi  $f(x,y)$  berukuran M baris dan N kolom, dengan  $x$  dan  $y$  adalah kordinat spasial, dan amplitudo  $f$  di titik kordinat  $(x,y)$  dinamakan intensitas atau tingkat keabuan dari citra pada citra tersebut (Putra, 2010). Pada umumnya warna dasar dalam citra RGB menggunakan penyimpanan 8 bit untuk menyimpan data warna, yang berarti setiap warna mempunyai gradasi sebanyak 255 warna . Dewasa ini, citra digital dapat menggunakan 16 bit untuk menyimpan data warna dasarnya, hal ini menyebabkan semakin banyak gradasi warnanya sehingga citra yang dihasilkan memiliki tingkat warna yang jauh lebih banyak. Namun tentu saja hal ini mengakibatkan ukuran file citra digital yang dihasilkan juga menjadi semakin besar walaupun dengan ukuran yang sama. Berdasarkan jenis warnanya citra digital dibagi menjadi 3 jenis yaitu citra biner, citra grayscale dan citra warna.

###### **2.1.1.1 Citra Biner (monokrom)**

Citra biner adalah citra yang hanya memiliki dua warna saja yaitu hitam dan putih. Warna hitam direpresentasikan dengan 1 dan warna putih direpresentasikan dengan 0. Dibutuhkan 1 bit di memori untuk menyimpan nilai warna pada setiap piksel citra biner. Contoh citra biner dapat dilihat pada gambar 2.1(a).

###### **2.1.1.2 Citra Grayscale**

Banyaknya warna pada citra *grayscale* tergantung pada jumlah bit yang disediakan di memori untuk menampung kebutuhan warna pada citra ini. Citra *grayscale* yang 2 bit memiliki 4 gradasi warna, citra *grayscale* 3 bit memiliki 8 gradasi warna, citra *grayscale* dengan 8 bit memiliki 256 gradasi warna dan seterusnya. Semakin besar jumlah bit warna yang disediakan di memori, semakin banyak dan semakin halus gradasi warna yang terbentuk. Pada umumnya citra digital *grayscale*

menggunakan 8 bit memori dengan derajat keabuan dari 0 sampai 255. Contoh citra *grayscale* dapat dilihat pada gambar 2.1(b).

### 2.1.1.3 Citra Warna

Setiap piksel pada citra warna mewakili warna yang merupakan kombinasi dari tiga warna dasar (RGB = red, green, blue). Pada umumnya setiap warna dasar menggunakan penyimpanan 8 bit, yang berarti setiap warna mempunyai gradasi sebanyak 255 warna. Berarti setiap piksel mempunyai kombinasi warna sebanyak  $255 \times 255 \times 255 = 16$  juta warna lebih. Contoh citra warna dapat dilihat pada gambar 2.1(c).



Gambar 2.1: (a) Contoh citra biner, (b) contoh citra grayscale, (c) contoh citra warna.

### 2.1.2 Pengolahan Citra Digital

Pengolahan citra digital merupakan proses mengolah piksel-piksel di dalam citra secara digital untuk tujuan tertentu. Berdasarkan tingkat pemrosesannya pengolahan citra digital dikelompokkan menjadi tiga kategori, yaitu: *low-level*, *mid-level* dan pemrosesan *high-level*. Pemrosesan *low-level* dilakukan dengan operasi primitif seperti *image preprocessing* untuk mengurangi derau (*noise*), memperbaiki kontras citra dan mempertajam citra (*sharpening*). Karakteristik dari pemrosesan *low-level* yaitu keluaran atau hasil dari pemrosesannya berupa citra digital. Pemrosesan *mid-level* melibatkan tugas-tugas seperti segmentasi (mempartisi gambar menjadi beberapa bagian atau objek), deskripsi objek untuk dilakukan pemrosesan lanjutan, dan klasifikasi objek dalam citra digital. Karakteristik dari pemrosesan

*mid-level* yaitu keluaran atau hasilnya berupa atribut atau fitur seperti, kontur, tepi, atau objek yang terdapat dalam citra tersebut. Pemrosesan *high-level* merupakan proses tingkat lanjut dari dua proses sebelumnya, dilakukan untuk mendapat informasi lebih yang terkandung dalam citra (Gonzalez and Woods, 2001).

Berdasarkan tujuannya pengolahan citra juga dapat dibagi menjadi beberapa bagian yaitu: *image enhancement*, *image restoration*, *image analysis* dan *image compression* (Silva and Mendonca, 2005).

#### **2.1.2.1 Image Enhancement**

*Image Enhancement* adalah metode pengolahan citra digital untuk membuat citra tampak lebih baik atau dilakukan peningkatan untuk analisis tertentu. Namun hal ini dapat menyebabkan pengorbanan aspek lain dari citra tersebut. Penerapan filter, penghalusan citra, memperbaiki kontras dan morfologi citra adalah contoh *image enhancement*.

#### **2.1.2.2 Image Restoration**

*Image restoration* adalah metode pengolahan citra untuk memulihkan citra dari penurunan kualitas atau citra yang rusak karena derau (*noise*). Pada dasarnya metode ini berbeda dengan *image enhancement* yang berkaitan dengan ekstraksi fitur pada citra.

#### **2.1.2.3 Image Analysis**

*Image analysis* adalah metode pengolahan citra untuk menghitung besaran kuantitatif dari citra untuk menghasilkan deskripsinya. Metode ini dilakukan dengan mengekstraksi ciri-ciri tertentu yang membantu dalam identifikasi objek (Silva and Mendonca, 2005).

#### **2.1.2.4 Image Compression**

Metode ini dilakukan agar citra dapat direpresentasikan dalam bentuk yang lebih kompak sehingga memerlukan memori yang lebih sedikit. Metode ini dapat

dilakukan dengan mengurangi redundansi dari data-data yang terdapat dalam citra sehingga dapat disimpan atau ditransmisikan secara efisien.

### 2.1.3 Filter Spasial

Konsep filter spasial pada pengolahan citra digital berasal dari penerapan transformasi Fourier untuk pemrosesan sinyal pada domain frekuensi. Istilah filter spasial ini digunakan untuk membedakan proses ini dengan filter pada domain frequensi. Proses filter dilakukan dengan cara menggeser filter kernel dari titik ke titik dalam citra digital. Istilah *mask*, *kernel*, *template*, dan *window* merupakan istilah yang sama dan sering digunakan dalam pengolahan citra digital (Gonzalez and Woods, 2001). Dalam penelitian ini penulis menggunakan istilah kernel untuk istilah tersebut.

Proses filter dalam pengolahan citra digital dilakukan dengan memanipulasi sebuah citra menggunakan kernel untuk menghasilkan citra yang baru, sehingga dengan kernel yang berbeda maka citra hasil yang didapat juga akan berbeda.

#### 2.1.3.1 Operator Linear dan Non-linear

Didefinisikan  $H$  sebuah operator dengan *input* dan *output* adalah citra digital.  $H$  dikatakan operator linear jika untuk sembarang gambar  $f$  dan  $g$ , dan untuk sembarang skalar  $a$  dan  $b$  berlaku,

$$H(af + bg) = aH(f) + bH(g) \quad (2.1)$$

Dengan kata lain hasil dari operator linear dengan jumlahan dua buah citra (yang telah dikali dengan konstanta  $a$  dan  $b$ ) identik dengan hasil operator linear pada masing-masing gambar, dikali dengan konstanta yang sama, kemudian hasilnya dijumlahkan. Sebagai contoh, sebuah operator dengan fungsi yang menjumlahkan K citra adalah operator linear. Operator yang menghitung nilai mutlak dari perbedaan dua gambar adalah tidak linear. Operator yang tidak memenuhi persamaan (2.1) dikatakan non-linear (Gonzalez and Woods, 2001).

## 2.1.4 Kernel

### 2.1.4.1 Average Blur

*Average blur* atau biasa juga disebut *box filter* adalah salah satu filter yang digunakan untuk menghaluskan citra dan mengurangi derau. Secara sederhana nilai sebuah piksel yang baru adalah nilai rata-rata dari nilai piksel tersebut dengan nilai piksel tetangganya (Kowalczyk, Przewlocka, and Krvjak, 2018). Berikut kernel *Average blur* yang digunakan dalam penelitian ini:

$$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad (2.2)$$

### 2.1.4.2 Gaussian Blur

Filter ini juga digunakan untuk menghaluskan citra dan mengurangi derau. Idenya mirip seperti *Average blur*, nilai piksel yang baru dibentuk dari nilai piksel tetangganya, tepat dengan memberikan bobot yang lebih kuat pada nilai pikselnya sendiri diikuti dengan bobot yang lebih rendah pada piksel atas, bawah dan sampingnya (Ustyukov, Efimov, and Kolchaev, 2019). Berikut kernel gaussian blur filter yang digunakan dalam penelitian ini:

$$\frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \quad (2.3)$$

### 2.1.4.3 Sobel

Filter Sobel termasuk *high-pass* filter yang umum digunakan untuk deteksi tepi pada citra. Sobel memiliki dua kernel untuk deteksi tepi yaitu kernel sobel vertikal untuk mendeteksi tepi secara vertikal dan kernel sobel horizontal yang mendeteksi tepi secara horizontal (Kowalczyk, Przewlocka, and Krvjak, 2018). Berikut kernel

Sobel vertikal dan horizontal yang digunakan dalam penelitian ini:

$$\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \quad \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \quad (2.4)$$

#### 2.1.4.4 Laplacian

Filter ini dapat digunakan untuk deteksi tepi pada citra karena sifatnya yang sensitif dengan perubahan intensitas yang cepat (Jingbo dkk., 2011). Tidak seperti Sobel yang menggunakan dua kernel untuk mendeteksi tepi secara vertikal dan horizontal, disini hanya digunakan sebuah kernel yang dapat digunakan untuk deteksi tepi secara vertikal dan horizontal sekaligus. Berikut kernel Laplacian yang digunakan dalam penelitian ini:

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \quad (2.5)$$

#### 2.1.4.5 Sharpening

Sharpening filter digunakan untuk memperjelas detail halus dalam citra atau untuk meningkatkan detail pada citra yang *blur*, baik karena kesalahan ataupun karena efek dari metode akuisisi citra tertentu (Yang, 2013). Berikut kernel untuk filter *sharpening* yang digunakan dalam penelitian ini:

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{bmatrix} \quad (2.6)$$

#### 2.1.5 Konvolusi

Konsep filter spasial linear mirip seperti konsep konvolusi pada domain frekuensi, dengan alasan tersebut filter spasial linear biasa disebut juga konvolusi sebuah kernel dengan citra digital (Gonzalez and Woods, 2001). Konvolusi pada

fungsi  $f(x)$  dan  $g(x)$  didefinisikan pada persamaan 2.7.

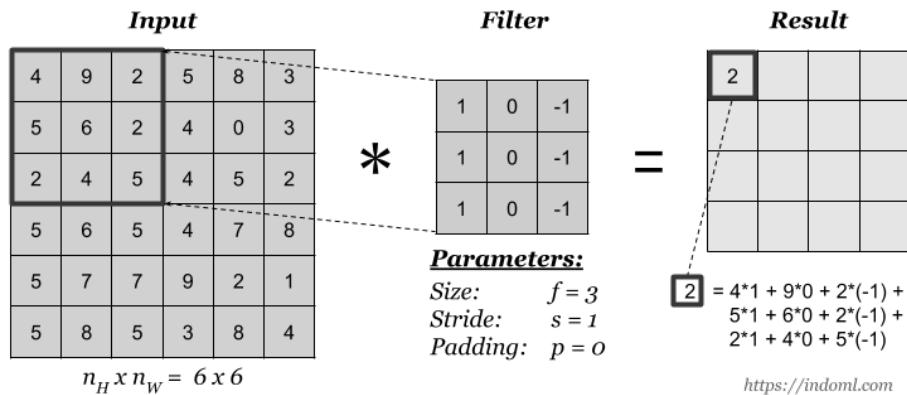
$$h(x) = f(x) * g(x) = \int_{-\infty}^{\infty} f(a)g(x-a)da \quad (2.7)$$

dimana tanda  $*$  menyatakan operator konvolusi, dan peubah  $a$  adalah peubah bantu. Untuk fungsi diskrit, konvolusi didefinisikan pada persamaan 2.8.

$$h(x) = f(x) * g(x) = \sum_{a=-\infty}^{\infty} f(a)g(x-a) \quad (2.8)$$

Pada operasi konvolusi diatas,  $g(x)$  disebut kernel konvolusi atau filter kernel. Kernel  $g(x)$  dioperasikan secara bergeser pada sinyal masukan  $f(x)$ . Jumlah perkalian kedua fungsi pada setiap titik merupakan hasil konvolusi yang dinyatakan dengan keluaran  $h(x)$  (Rinaldi, 2004).

Pada gambar (2.2) diilustrasikan bagaimana proses konvolusi pada citra digital yang direpresentasikan dalam bentuk matriks. Operasi konvolusi dilakukan pada matriks input berukuran  $6x6$  dengan filter berukuran  $3x3$ . Hasil konvolusinya ditampilkan pada matriks *result*.



Gambar 2.2: Ilustrasi konvolusi pada citra. Sumber: <https://indoml.com>

Jika hasil konvolusi menghasilkan nilai piksel negatif, maka nilai tersebut dijadikan 0, sebaliknya jika hasil konvolusi menghasilkan nilai piksel yang melebihi nilai keabuan maksimum, maka nilai tersebut dijadikan ke nilai keabuan maksimum pada citra tersebut (Sutoyo dkk., 2009).

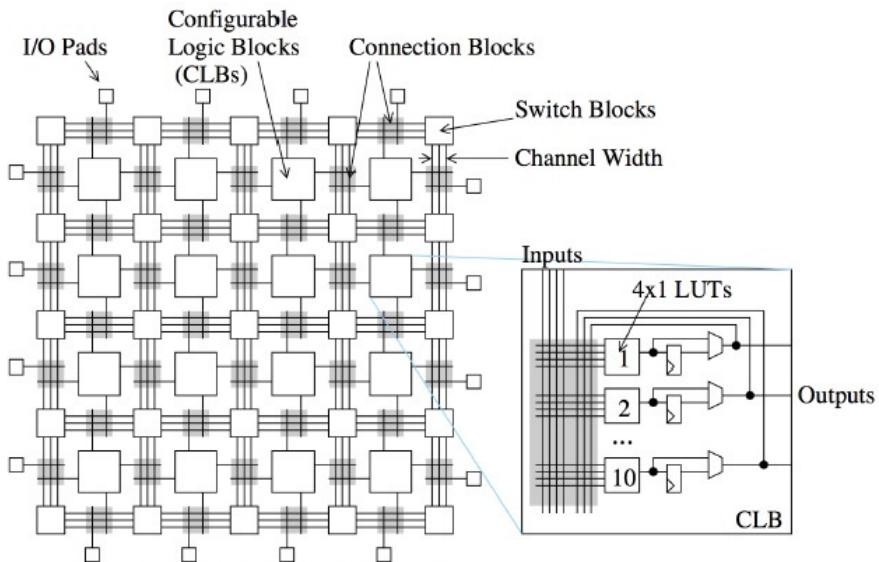
### **2.1.6 Video Stream**

Video *stream* dapat dipandang sebagai serangkaian citra digital berturut-turut (Zhao, 2015). Berbeda dengan format video lainnya, video *stream* ini tidak disimpan pada media penyimpanan sebagai file dengan format video melainkan langsung disalurkan setiap framenya dari sumber (*source*) ke penerima, dalam hal ini FPGA. Dengan menganggap Video *stream* adalah kumpulan citra digital (*frame*) maka dapat dilakukan metode pengolahan seperti pada citra digital, termasuk penerapan filter spasial.

### **2.1.7 FPGA**

*Field Programmable Gate Arrays* atau FPGA adalah perangkat semikonduktor yang berbasis *matriks configurable logic block* (CLBs) yang terhubung melalui interkoneksi yang dapat diprogram. FPGA dapat diprogram ulang ke aplikasi atau fungsi yang diinginkan setelah *manufacturing*. Fitur ini yang membedakan FPGA dengan *Application Specific Integrated Circuits* (ASICs), yang dibuat khusus untuk tugas tertentu saja (Xilinx, 2020).

Sebuah *microprocessor* menerima instruksi berupa kode 1 atau 0, kode-kode ini selanjutnya diinterpretasikan oleh komputer untuk menjalankan perintah yang diberikan. *Microprocessor* ini membutuhkan intruksi berupa kode secara terus menerus untuk menjalankan fungsinya. Sedangkan pada FPGA hanya dibutuhkan sekali konfigurasi *chip* setiap kali dinyalakan. Membuat atau mengunduh *bitstream* yang menentukan fungsi logika dilakukan oleh *logic elements* (LEs), sebuah sirkuit dapat dibuat dengan mengabungkan beberapa LEs menjadi satu kesatuan. Setelah *bitstream* dipasang, FPGA tidak perlu lagi membaca instruksi berupa 1 dan 0, berbeda dengan *microprocessor* yang selalu membutuhkan instruksi (Cheung, 2019). Secara tradisional, untuk membuat sebuah desain FPGA, aplikasi dideskripsikan menggunakan *Hardware Description Language* (HDL) seperti Verilog atau VHDL sehingga menghasilkan sebuah *bitstream* FPGA.



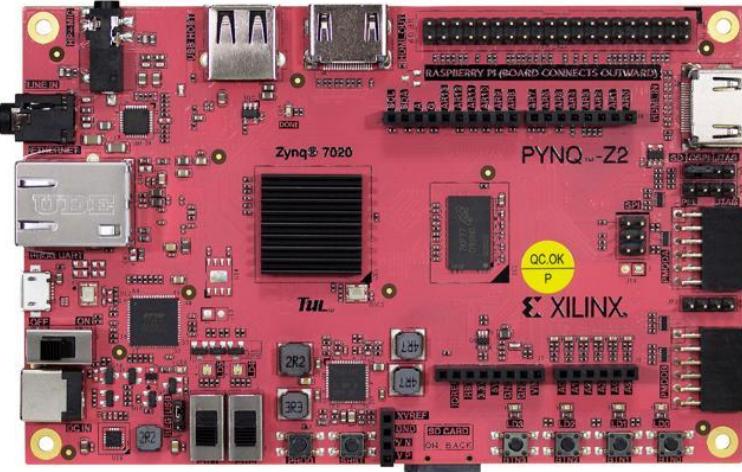
*Gambar 2.3: Struktur FPGA.*

#### 2.1.7.1 FPGA Development Board

Pada FPGA terdahulu tidak terdapat *processor* (CPU) untuk menjalankan software apapun, sehingga ketika ingin mengimplementasikan aplikasi haruslah merancang sirkuit dari awal, seperti mengonfigurasi FPGA sesederhana gerbang logika OR atau serumit *multi-core processor* (Biswas, 2019). Dewasa ini telah dikembangkan *FPGA Development Board* atau biasa disebut juga *FPGA Board* yaitu teknologi FPGA yang dirangkai dalam sebuah *board* dan dilengkapi dengan *microprocessor* dan beberapa *interface IO* untuk menjankan tugas tertenu. Umumnya *FPGA Board* telah dilengkapi dengan interface untuk mengakses dan menerapkan desain sirkuitnya. Xilinx, Altera dan Intel adalah produsen *FPGA Board* yang terkenal. *FPGA Board* yang digunakan dalam penelitian ini yaitu Xilinx PYNQ-Z2 dengan Jupyter Notebook sebagai *interface* untuk mengakses dan menjalankan program pada penelitian ini. Bentuk *FPGA Board* Xilinx PYNQ-Z2 dapat dilihat pada gambar (2.4)

#### 2.1.8 Evaluasi Performa

Pada penelitian ini peneliti menggunakan waktu komputasi, *frame rate* (FPS), penggunaan CPU, penggunaan memory, penggunaan resident memory (RES), shared memory (SHR), dan virtual memory (VIRT) untuk mengukur performa pada penerapan filter spasial linear dengan prosesor ARM dan FPGA.



Gambar 2.4: FPGA Board Xilinx PYNQ-Z2.

#### 2.1.8.1 Waktu Komputasi

Waktu komputasi yang dimaksud oleh peneliti adalah durasi yang dibutuhkan sebuah kernel untuk melakukan filter spasial linear terhadap beberapa *frame* input. Waktu komputasi ini diperoleh dengan cara menghitung selisih waktu selesai dengan waktu dimulai penerapan filter spasial linear pada *frame* input.

$$waktukompuasi = waktuselesai - waktumulai \quad (2.9)$$

#### 2.1.8.2 Frame Rate (FPS)

*Frame rate* atau *frame per second* (fps) adalah banyaknya *frame* yang ditampilkan per detik pada video ataupun video straem. Semakin tinggi fps sebuah video maka semakin halus pula gerakan yang dihasilkan. Sebaliknya video dengan fps rendah akan menghasilkan gerakan yang kurang baik. *Frame rate* atau fps dapat dihitung dengan cara membagi jumlah *frame* dengan waktu komputasinya seperti pada persamaan 2.10 (Madhusudana dkk., 2020).

$$fps = \frac{jumlah\ frame}{waktukomputasi} \quad (2.10)$$

### 2.1.8.3 Penggunaan CPU

Pada penelitian ini peneliti menggunakan fitur yang tersedia pada sistem operasi Linux yang berjalan di FPGA Development Board untuk melihat persentase penggunaan CPU pada proses penerapan filter spasial linear. Tampilan dari program ini dapat dilihat pada gambar 2.5. Program ini menampilkan informasi tentang proses-proses yang berjalan pada sistem operasi seperti ID sebuah proses, user yang menjalankan proses tersebut, memory yang digunakan, status sebuah proses, persentase CPU yang digunakan dan lainnya.

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
9528	dir	20	0	391668	47340	40212	S	9.3	1.2	0:00.28	flameshot
3896	dir	20	0	1707712	77500	23748	R	1.0	2.0	1:47.73	plasmashell
3448	root	20	0	866028	57936	26712	S	0.7	1.5	3:00.94	Xorg

Gambar 2.5: Tampilan program top pada Linux.

Status sebuah proses pada program ini dinyatakan dengan beberapa singkatan, diantaranya yaitu:

I = *idle*

R = *running*

S = *sleeping*

Z = *zombie*

D = *uninterruptible sleep*

T = *stopped by job control signal*

t = *stopped by debugger during trace*

Pada CPU yang multi-core sebuah proses akan ditampilkan persentase penggunaan CPUnya berdasarkan core yang digunakan oleh proses tersebut. Jika mode Irix pada program *top* dimatikan, maka program akan berjalan pada mode Solaris di mana penggunaan CPU sebuah proses yang ditampilkan akan dibagi dengan jumlah total core yang ada pada CPU (Linux, ).

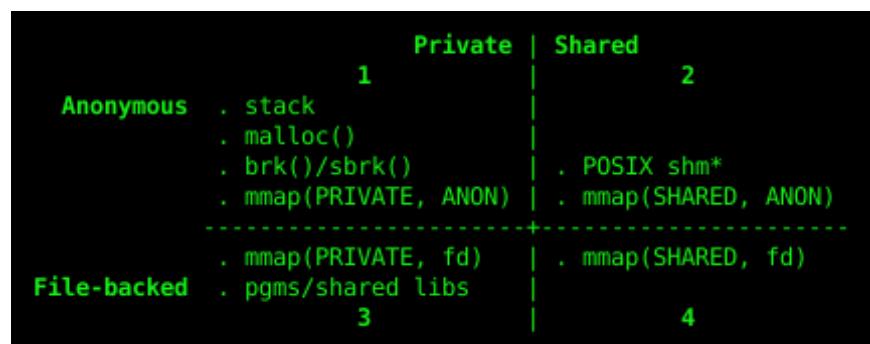
### 2.1.8.4 Penggunaan Memory

Pada sistem operasi linux *memory* dibagi menjadi tiga jenis (Linux, ). Pertama yaitu *memory* fisik, sumber daya terbatas di mana kode dan data harus berada saat

dijalankan atau direferensikan. Berikutnya adalah *memory swap*, yaitu *memory* yang berguna untuk membantu kerja *memory* fisik, data dari *memory* fisik akan disimpan pada *swap* dan kemudian diambil kembali jika terlalu banyak permintaan pada *memory* fisik. Ketiga yaitu virtual *memory*, sumber daya yang hampir tidak terbatas yang digunakan untuk tujuan berikut (Silberschatz, Galvin, and Gagne, 2009):

- *abstraction*, bebas dari alamat / batas *memory* fisik
- *isolation*, setiap proses dalam ruang alamat terpisah
- *sharing*, pemetaan tunggal dapat memenuhi banyak kebutuhan
- *flexibility*, menetapkan alamat virtual ke data

Terlepas dari bentuk *memory* mana yang mungkin digunakan, semua dikelola sebagai *pages* (biasanya 4096 byte). Penggunaan *memory* berhubungan dengan *memory* fisik dan *swap* untuk sistem secara keseluruhan. Untuk setiap proses yang berjalan, setiap *memory* page dibatasi ke satu kuadran seperti pada gambar 2.6. Baik *memory* fisik dan *memory* virtual dapat menyertakan salah satu dari empat kuadran, sementara file *swap* hanya mencakup kuadran 1 sampai 3. Memori di kuadran 4, bertindak sebagai file *swap* khusus ketika dimodifikasi (Linux, ).



Gambar 2.6: Kuadran pembagian *memory* pada Linux.

#### 2.1.8.5 Virtual Memory (VIRT)

Virtual *memory* menggunakan disk sebagai perpanjangan dari RAM sehingga ukuran efektif *memory* yang dapat digunakan bertambah secara bersamaan. Kernel akan menulis konten dari blok *memory* yang saat ini tidak digunakan ke hard disk sehingga *memory* dapat digunakan untuk tujuan lain. Ketika konten asli dibutuhkan lagi, mereka dibaca kembali ke dalam *memory*. Ini semua dibuat transparan

sepenuhnya bagi pengguna. Program yang berjalan di Linux hanya melihat jumlah *memory* yang tersedia lebih besar dan tidak memperhatikan bahwa sebagian dari program tersebut berada di disk dari waktu ke waktu. Tentu saja, membaca dan menulis hard disk lebih lambat daripada menggunakan *memory* fisik, sehingga program tidak berjalan secepat itu. Bagian dari hard disk yang digunakan sebagai *memory* virtual disebut ruang swap (S dkk., 2020).

Kolom VIRT pada program **top** menunjukkan jumlah total *memory* virtual yang digunakan oleh proses. Ini mencakup semua kode, data dan *shared libraries* ditambah dengan *pages* yang telah ditukar dan *pages* yang telah dipetakan tetapi tidak digunakan (Linux, ).

#### 2.1.8.6 Resident Memory (RES)

*Resident memory* adalah bagian dari ruang alamat virtual (VIRT) yang mewakili *memory* fisik yang tidak ditukar yang sedang digunakan tugas. Resident *memory* ini juga merupakan penjumlahan dari RSan, RSfd dan Bidang RSsh. Ini dapat mencakup private anonymous *pages*, halaman pribadi yang dipetakan ke file (termasuk *program images* dan *shared libraries*) ditambah shared anonymous *pages*. Semua *memory* tersebut didukung oleh file *swap* yang direpresentasikan secara terpisah pada SWAP. Resident *memory* ini juga dapat menyertakan *pages* yang didukung *shared file-backed* yang apabila dimodifikasi, maka akan bertindak sebagai file *swap* khusus dan karenanya tidak akan pernah memengaruhi SWAP (Linux, ).

#### 2.1.8.7 Shared Memory (SHR)

*Shared memory* adalah bagian dari resident *memory* (RES) yang dapat digunakan oleh proses lain. Termasuk *anonymous pages* dan shared file-backed *pages*. Ini juga termasuk private *pages* dipetakan ke file yang mewakili *program images* dan *shared libraries* (Linux, ).

## 2.2 Penelitian Terkait

### 2.2.1 Spatial Filtering Based Boundary Extraction in Underwater Images for Pipeline Detection: FPGA Implementation

Pipa bawah air diletakkan di dasar laut untuk tujuan pengangutan minyak bumi dan gas menyebrangi lautan. Pipa perlu terus dipantau untuk menghindari gangguan dalam proses transportasi. Gambar dasar laut dapat diperoleh dengan menggunakan kamera dan dengan memproses gambar yang diperoleh dapat membantu dalam mendeteksi pipa. Penelitian ini membahas tentang metode pemrosesan citra untuk deteksi pipa bawah laut dari gambar bawah laut yang diambil oleh kendaraan bawah laut yang dapat digunakan sebagai langkah awal untuk melacak saluran pipa. Implementasinya berhasil dilakukan pada *Field Programmable Gate Array* (FPGA) berbasis *development board* (Raj., Abraham, and Supriya, 2016).

### 2.2.2 FPGA Implementation of Spatial Filtering techniques for 2D Images

Berbagai teknik filter telah menjadi inti dari pemrosesan citra sejak awal teknik peningkatan citra (*image enhancement*). Filter spasial pada pemrosesan citra digital digunakan dalam banyak kepentingan seperti mempertajam citra, menghaluskan citra, menghilangkan derau dan sebagainya. Fleksibilitas dari metode filter spasial sering dibandingkan dengan domain transformasi karena dapat digunakan untuk filter linear dan filter non-linear. Penghalusan citra dilakukan dengan langsung memanipulasi nilai intensitas dari citra asli dengan sebuah kernel filter. Hasilnya yaitu berkurangnya detail kecil dan derau pada citra. Penelitian ini tentang penerapan berbagai macam operator filter spasial. Hasilnya didasarkan pada konsumsi perangkat keras, kecepatan desain masing-masing arsitektur. Ukuran kualitas citra didapat dengan membandingkan output dari Matlab dan output dari Xilinx FPGA dan dengan menghitung MSE (Sadangi dkk., 2017).

### **2.2.3 Features of Image Spatial Filters Implementation on FPGA**

Penelitian ini menyajikan fitur-fitur implementasi filter spasial pada citra dengan *Programmable Logic Integrated Circuits* (FPGA). Solusi yang disajikan memungkinkan untuk membuat arsitektur kristal dengan performa tinggi untuk algoritma filter spasial. Hasilnya menunjukkan kelebihan menggunakan *programmable logic* dalam tugas pemrosesan citra digital (Ustyukov, Efimov, and Kolchaev, 2019).

### **2.2.4 An FPGA-Oriented Algorithm for Real-Time Filtering of Poisson Noise in Video Streams, with Application to X-Ray Fluoroscopy**

Pada penelitian ini dibahas tentang algoritma baru untuk *real-time filtering* pada video yang rusak karena *poison noise*. Algoritma yang disajikan efektif dalam penanganan derau, dan ini secara ideal cocok dengan implementasi *hardware*, dan dapat diimplementasikan pada FPGA kecil yang memiliki sumber daya *hardware* yang terbatas. Pada penelitian ini penerapan algoritma menggunakan hasil *X-ray fluoroscopy* sebagai studi kasus. Hasil implementasi menggunakan yang StratixIV FPGA menunjukkan bahwa sistem hanya menggunakan, paling banyak, 22% dari sumber daya perangkat, dalam implementasi *real-time filtering* pada video *stream* 1024x1024 @49fps. Sebagai perbandingan, implementasi filter berbasis FIR pada FPGA yang sama dan dengan video *stream* yang serupa, dibutuhkan 80% *resource logic* pada FPGA (Castellano dkk., 2019).

### **2.2.5 A real-time video denoising algorithm with FPGA implementation for Poisson-Gaussian noise**

Pada penggunaan umum metode denoising yaitu *Pixel Similarity Weighted Frame Average* (PSWFA). Pada penelitian ini, dilakukan peningkatan kemampuan denoising dari PSWFA menggunakan pre-filter yang mengandung operator *downsampling* dan small Gaussian filter. Transformasi citra dapat mengalami gangguan oleh derau Gaussian. Untuk memasang algoritma ini pada perangkat keras, sebelumnya diimplementasikan algoritma ini pada Spartan-6 FPGA untuk evaluasi. Dilakukan juga perbandingan dengan beberapa metode denoising yang sudah ada. Evaluasi selanjutnya untuk kemampuan denoising, algoritma ini dibandingkan dengan

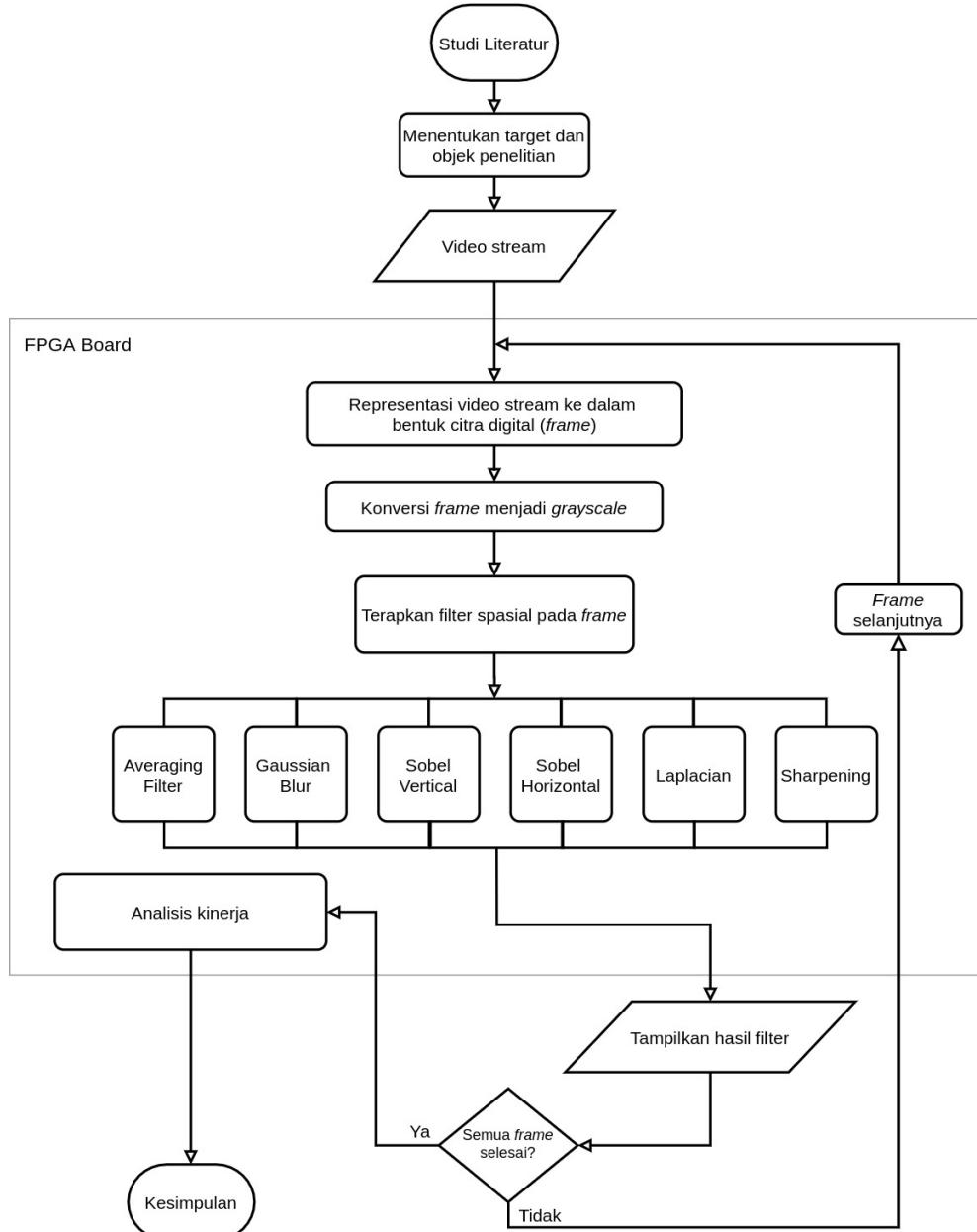
beberapa algoritma *state-of-art* yang tidak diimplementasikan pada FPGA tetapi memiliki performa yang baik pada personal komputer. Hasil eksperimen pada kedua simulasi video berderau dan video yang ditangkap pada pencahayaan yang kurang menunjukkan tingkat keefektifan pada algoritma ini, terkhusus pada pemrosesan derau berskala besar (Tan dkk., 2014).

## BAB III

### METODE PENENILITIAN

#### 3.1 Tahapan Penelitian

Tahapan dalam penelitian ini dapat dilihat pada gambar 3.1.



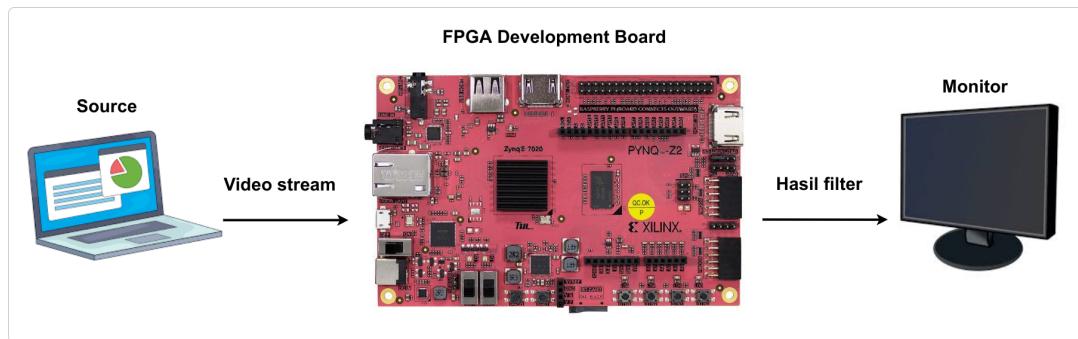
Gambar 3.1: Flowchart tahapan penelitian.

### 3.2 Waktu dan Lokasi Penelitian

Penelitian ini dilaksanakan dari bulan Juni 2020 sampai dengan bulan Agustus 2020. Lokasi penelitian dilakukan di Laboratorium Rekayasa Perangkat Lunak Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Hasanuddin Makassar.

### 3.3 Rancangan Sistem

Pada penelitian ini akan dibangun suatu sistem untuk mengimplementasikan filter spasial linear pada FPGA Development Board, dapat dilihat pada gambar 3.2.



Gambar 3.2: Rancangan sistem.

Video *stream* dari *source* disalurkan melalui port HDMI Input pada FPGA Development Board, kemudian video *stream* tersebut akan diolah dengan menerapkan filter spasial linear pada setiap framenya. Setiap *frame* yang telah diterapkan filter spasial akan dialirkan ke monitor untuk kemudian ditampilkan. Selanjutnya dilakukan analisis kinerja pada FPGA. FPGA Development Board yang digunakan dalam penelitian ini dapat diakses dengan *ssh* pada port 22 atau dengan *Jupyter Notebook* melalui *web browser*.

### **3.4 Instrumen Penelitian**

1. Kebutuhan perangkat lunak:
  - a. Linux Ubuntu 18, sebagai OS pada FPGA Development Board.
  - b. Python 3.6, dengan library OpenCV, Numpy, Pynq 5.2, dan Xilinx xfOpenCV.
  - c. Jupyter Notebook pada FPGA Development Board.
  - d. Web Browser untuk mengakses Jupyter Notebook pada FPGA Development Board.
2. Kebutuhan perangkat keras:
  - a. FPGA Development Board.
  - b. Micro SD Card 16 GB, sebagai media penyimpanan OS pada FPGA Development Board.
  - c. Monitor Eksternal, untuk menampilkan hasil penerapan filter spasial pada FPGA Development Board.
  - d. Laptop Lenovo Ideapad 320 (sebagai *source video stream*).

Spesifikasi FPGA Development Board yang digunakan:

- Model : Xilinx PYNQ-Z2.
- Processor : Dual-Core ARM Cortex A9, 650 MHz
- FPGA : 1,3M reconfigurable gates
- Memory : 512 MB DDR3 / Flash
- Storage : Micro SD card slot
- Power : DC 7V-15V
- Dimension : 3,44" x 5,39" (87mm x 137mm)

## **BAB IV**

### **HASIL DAN PEMBAHASAN**

#### **4.1 Implementasi pada FPGA Development Board**

Pada penelitian ini digunakan 6 kernel berbeda berukuran  $3 \times 3$  untuk penerapan filter spasial linear pada video *stream* 720p 60 FPS. Penerapan ini dilakukan pada FPGA Development Board dengan menggunakan prosesor ARM dan FPGA. Kemudian dilakukan analisis kinerja dari keduanya untuk menunjukkan masing-masing waktu komputasi, FPS, persentase penggunaan CPU, penggunaan *memory*, *resident memory* (RES), *shared memory* (SHR), dan *virtual memory* (VIRT) pada masing-masing kernel.

FPGA Development Board dirangkai seperti yang telah disebutkan pada Bab sebelumnya, pada gambar 3.2. HDMI Output pada FPGA dihubungkan ke monitor dan HDMI Input dihubungkan ke *source* dalam hal ini laptop Lenovo Ideapad 320. FPGA Development Board juga dihubungkan ke *router* menggunakan kabel UART agar dapat diakses menggunakan protokol *ssh*. Selanjutnya memasang semua *library* yang dibutuhkan untuk melakukan penerapan filter spasial linear pada video *stream* menggunakan FPGA Development Board.

##### **4.1.1 Penerapan Filter Spasial**

Setiap *frame* dari *source* video *stream* dibaca sebagai citra digital yang direpresentasikan sebagai matriks berukuran 1280x720 dengan rentang nilai keabuan pada masing-masing piksel yaitu dari 0 sampai 255. Setiap piksel pada matriks tersebut hanya memiliki satu lapis warna sebab citra yang diterima dari *source* adalah citra *grayscale*, tidak seperti citra warna yang memiliki tiga lapis warna pada setiap pikselnya.

Proses filter spasial dilakukan dengan operasi konvolusi pada setiap matriks dengan kernel yang telah ditentukan sebelumnya. Operasi konvolusi ini menghasilkan matrix baru dengan ukuran 1280x720. Matriks hasil tersebut selanjutnya direpresentasikan kembali sebagai citra digital yang selanjutnya disebut sebagai hasil filter. Hasil filter dari setiap *frame* ini ditampilkan ke monitor melalui HDMI Output

pada FPGA Development Board secara berkesinambungan sehingga tampak seperti video.



*Gambar 4.1: Contoh Frame Grayscale.*

Salah satu contoh *frame* dari *source* dapat dilihat pada gambar 4.1. Selanjutnya dilakukan filter spasial menggunakan 6 kernel yang telah ditentukan sebelumnya.

#### **4.1.1.1 Average Blur**

Penerapan filter spasial pada *frame grayscale* yang berukuran 1280x720 pixel dengan kernel *average blur* (2.2) yang berukuran 3x3 menghasilkan citra blur yang berukuran 1280x720. Hasil filter *average blur* dapat dilihat pada gambar 4.2. Filter seperti ini dapat digunakan untuk mengurangi derau pada citra.

#### **4.1.1.2 Gaussian Blur**

Penerapan filter spasial dengan kernel *gaussian blur* (2.3) yang berukuran 3x3 menghasilkan citra blur yang secara kasat mata mirip dengan filter *average blur*. Namun apabila diperhatikan nilai masing-masing pixel pada gambar 4.3 akan terlihat berbeda dengan nilai masing-masing pixel pada gambar 4.2. Hal ini disebabkan oleh nilai bobot pada kernel *gaussian blur* yang berbeda dengan kernel *average blur* sehingga hasil konvolusinya juga berbeda.



Gambar 4.2: Hasil filter Average Blur.



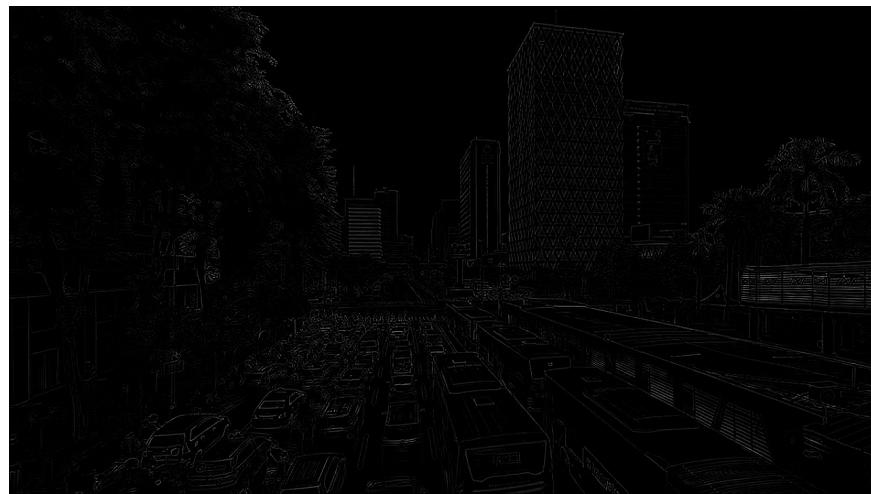
Gambar 4.3: Hasil filter Gaussian Blur.

#### 4.1.1.3 Laplacian

Penerapan filter spasial dengan kernel *laplacian* (2.5) menghasilkan citra biner yang hanya direpresentasikan dengan warna hitam dan putih saja, dapat dilihat pada gambar 4.4. Filter seperti ini dapat digunakan pada metode deteksi tepi dalam proses pengolahan citra digital.

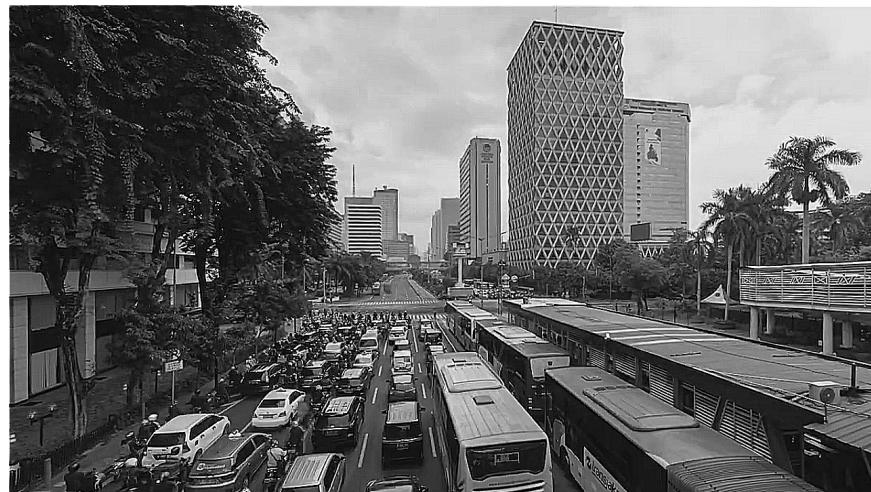
#### 4.1.1.4 Sharpening

Penerapan filter spasial dengan kernel *sharpening* (2.6) dapat meningkatkan detail (seperti garis) pada citra, namun dapat juga dapat menimbulkan derau pada citra



Gambar 4.4: Hasil filter Laplacian.

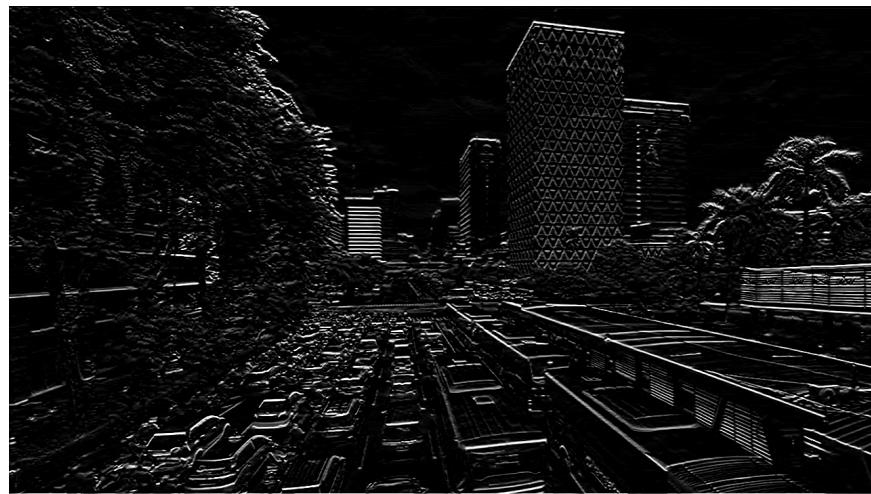
apabila bobot kernelnya tidak sesuai. Filter seperti ini lebih tepat digunakan untuk memperbaiki kualitas citra (dengan nilai kernel yang sesuai). Hasil filter *sharpening* ini dapat dilihat pada gambar 4.5.



Gambar 4.5: Hasil filter Sharpening.

#### 4.1.1.5 Sobel Horizontal

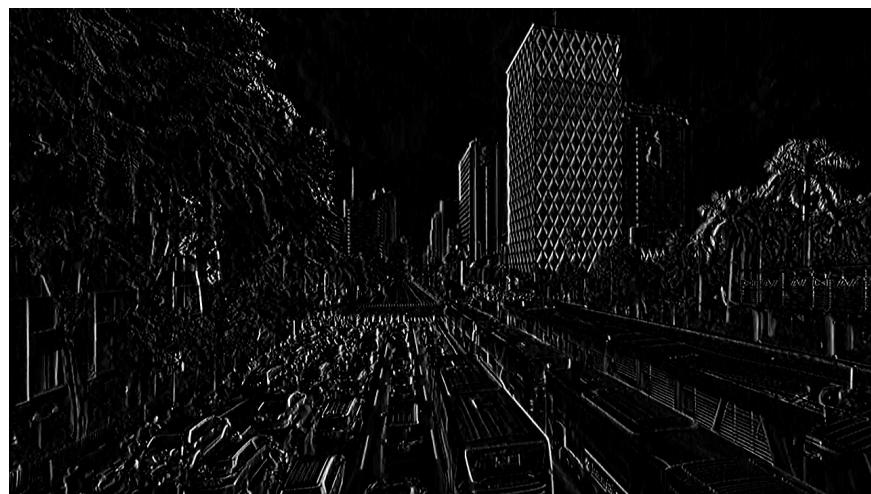
Penerapan filter spasial dengan kernel *sobel horizontal* (2.4) menghasilkan citra biner, dapat dilihat pada gambar 4.6. Filter seperti lebih tepat digunakan pada metode deteksi tepi dengan citra yang banyak mengandung garis horizontal.



Gambar 4.6: Hasil filter Sobel Horizontal.

#### 4.1.1.6 Sobel Vertical

Penerapan filter spasial dengan kernel *sobel vertical* (2.4) menghasilkan citra biner, dapat dilihat pada gambar 4.7. Sama halnya dengan filter *sobel horizontal*, filter *sobel vertical* juga dapat digunakan untuk metode deteksi tepi, terutama pada citra yang banyak mengandung garis vertikal.



Gambar 4.7: Hasil filter Sobel Vertical.

### 4.1.2 Penerapan Filter Spasial menggunakan Prosesor ARM dan FPGA

Penerapan filter spasial menggunakan prosesor ARM dilakukan dengan menggunakan *library* OpenCV dengan bahasa pemrograman Python yang dijalankan

pada FPGA Development Board. Sedangkan untuk penerapan filter spasial menggunakan FPGA dilakukan dengan menggunakan *library* xfOpenCV dari Xilinx. *Library* xfOpenCV ini adalah library khusus yang dimodifikasi dari OpenCV sehingga proses komputasinya dapat dilakukan dengan FPGA, bukan dengan prosesor ARM yang pada FPGA Development Board ini. Lebih lanjut mengenai program yang digunakan dapat dilihat pada lampiran *pynq-filter-spasial-arm* (7.3.2) dan *pynq-filter-spasial-fpga* (7.3.1).

#### 4.1.3 Proses Evaluasi Kinerja

Pada penelitian ini proses dalam evaluasi kinerja dilakukan dalam dua tahap. Proses pertama untuk menghitung waktu komputasi dan FPS dari masing-masing kernel dengan prosesor ARM dan FPGA. Kemudian proses kedua untuk mencatat persentase penggunaan CPU dan penggunaan *memory* termasuk *resident memory*, *shared memory* dan *virtual memory*.

##### 4.1.3.1 Menghitung Waktu Komputasi dan FPS

Pada proses ini peneliti melakukan percobaan dengan menggunakan 50 *frame* dan 200 *frame* dari *source* untuk dihitung FPS dan waktu komputasinya. Percobaan dilakukan sebanyak 5 kali menggunakan 50 *frame* dengan masing-masing kernel pada prosesor ARM dan FPGA, dan 5 kali percobaan menggunakan 200 *frame* dengan masing-masing kernel. Hasil masing-masing percobaan ini dapat dilihat pada daftar lampiran (7.4).

Menghitung waktu komputasi dilakukan dengan cara mencatat waktu mulai dan waktu semua *frame* selesai difilter pada setiap percobaan. Waktu komputasi diperoleh dari selisih antara waktu selesai dengan waktu mulai, seperti pada persamaan 2.9. Dilakukan dengan 50 *frame* dan 200 *frame* pada masing-masing kernel dengan menggunakan prosesor ARM dan FPGA. Proses ini dilakukan dengan menggunakan *library time* pada bahasa pemrograman Python, dapat dilihat pada gambar 4.8.

Hasil waktu komputasi ini dicatat dan kemudian digunakan untuk menghitung FPS dari masing-masing percobaan. Selanjutnya FPS dari masing-masing percobaan ini dihitung dengan menggunakan persamaan 2.10.

```

import time
...
startSW=time.time()
# implementasi
stopSW=time.time()
result = stopSW - startSW

```

Gambar 4.8: Menghitung waktu komputasi dengan library time di Python.

#### 4.1.3.2 Mencatat Penggunaan Memory dan CPU (Resource)

Pada proses ini peneliti menggunakan fitur yang tersedia pada sistem operasi Linux yang berjalan di FPGA Development Board untuk membantu mencatat penggunaan CPU dan *memory (resource)* pada saat proses penerapan filter spasial. Program ini menampilkan data tentang proses yang berjalan seperti ID sebuah proses, persentase *memory* yang digunakan oleh sebuah proses, persentase CPU yang digunakan, berapa lama sebuah proses berjalan, *resident memory*, *shared memory* dan *virtual memory*. Contoh output program tersebut dapat dilihat pada gambar 4.9.

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
28663	root	20	0	104940	28980	9128	S	52.8	5.7	1:45.05	python3
1663	root	20	0	65800	39936	9840	S	1.8	7.9	0:20.01	jupyter-notebook
<b>28659</b>	xilinx	<b>20</b>	<b>0</b>	<b>6724</b>	<b>2468</b>	<b>2064</b>	R	<b>0.3</b>	<b>0.5</b>	<b>0:03.16</b>	<b>top</b>
1	root	20	0	27460	5756	4488	S	0.0	1.1	0:02.99	systemd
2	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kthreadd
4	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/0:0H
5	root	20	0	0	0	0	I	0.0	0.0	0:00.04	kworker/u4:0
6	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	mm_percpu_wq
7	root	20	0	0	0	0	S	0.0	0.0	0:00.15	ksoftirqd/0
8	root	20	0	0	0	0	I	0.0	0.0	0:01.07	rcu_preempt
9	root	20	0	0	0	0	I	0.0	0.0	0:00.00	rcu_sched
10	root	20	0	0	0	0	I	0.0	0.0	0:00.00	rcu_bh
11	root	rt	0	0	0	0	S	0.0	0.0	0:00.00	migration/0
12	root	20	0	0	0	0	S	0.0	0.0	0:00.00	cpuhp/0

Gambar 4.9: Tampilan program top.

Peneliti melakukan 5 kali percobaan untuk mencatat penggunaan *memory* dan CPU dari masing-masing kernel dengan prosesor ARM dan FPGA. Percobaan ini dilakukan dengan menggunakan *Jupyter Notebook* yang berjalan pada FPGA Development Board yang dapat diakses menggunakan *web browser* dari perangkat lain. Serta digunakan protokol SSH untuk mengakses FPGA Development Board

dan menjalankan program untuk menampilkan penggunaan *resource* dari proses yang sedang berjalan.

```
import os  
print(os.getpid())
```

Gambar 4.10: Menampilkan PID sebuah proses dengan bahasa pemrograman Python.

Pertama peneliti menampilkan ID proses dari *Jupyter Notebook* yang digunakan untuk mencatat penggunaan *memory* dan CPU dari penerapan filter menggunakan prosesor ARM dan FPGA. Cara menampilkan ID proses (PID) menggunakan bahasa pemrograman Python dapat dilihat pada gambar 4.10. ID proses tersebut kemudian digunakan pada program **top** untuk menampilkan *resource* yang digunakan oleh proses tersebut.

```
$ top -d 0.1 -p 4382 -b >> arm-laplacian1.txt
```

Gambar 4.11: Menjalankan program **top** kemudian menyimpan hasilnya pada file *arm-laplacian1.txt*.

Pada gambar 4.11 ditunjukan cara menggunakan program **top** untuk mencatat penggunaan *resource* dari proses dengan ID 4382 kemudian *outputnya* disimpan pada file *arm-laplacian1.txt*. Program **top** tersebut dijalankan sesaat sebelum proses penerapan filter dijalankan pada *Jupyter Notebook*. Output dari program **top** setiap 0.1 detik disimpan pada file text yang telah ditentukan. Setelah seluruh frame selesai difilter maka program **top** juga dihentikan.

Isi file *arm-laplacian1.txt* setelah program selesai dijalankan dapat dilihat pada gambar 4.12. Isi dari file hasil ini masih banyak mengandung informasi yang tidak dibutuhkan pada penelitian ini, sehingga perlu dilakukan proses ekstraksi informasi yang dibutuhkan saja. Kemudian file yang telah diekstraksi tersebut dibuat menjadi file CSV agar lebih mudah dalam proses pengolahan selanjutnya, seperti pada gambar 4.13. Proses tersebut dilakukan sebanyak 5 kali percobaan pada masing-masing kernel menggunakan prosesor ARM dan FPGA.

```

top - 18:03:01 up 1:24, 2 users, load average: 0.35, 0.54, 0.47
Tasks: 1 total, 0 running, 1 sleeping, 0 stopped, 0 zombie
%Cpu(s): 3.3 us, 2.5 sy, 0.0 ni, 94.3 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
KiB Mem : 508556 total, 23816 free, 239548 used, 245192 buff/cache
KiB Swap: 1048572 total, 1047804 free, 768 used. 261604 avail Mem

PID USER      PR NI    VIRT    RES    SHR S %CPU %MEM     TIME+ COMMAND
4382 root      20  0 388124 125980 54808 S 0.0 24.8 0:24.96 python3

top - 18:03:01 up 1:24, 2 users, load average: 0.35, 0.54, 0.47
Tasks: 1 total, 1 running, 0 sleeping, 0 stopped, 0 zombie
%Cpu(s): 47.1 us, 1.7 sy, 0.0 ni, 51.3 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
KiB Mem : 508556 total, 23816 free, 239548 used, 245192 buff/cache
KiB Swap: 1048572 total, 1047804 free, 768 used. 261604 avail Mem

PID USER      PR NI    VIRT    RES    SHR S %CPU %MEM     TIME+ COMMAND
4382 root      20  0 392624 127384 56120 R 77.0 25.0 0:25.43 python3

top - 18:03:02 up 1:24, 2 users, load average: 0.35, 0.54, 0.47
Tasks: 1 total, 1 running, 0 sleeping, 0 stopped, 0 zombie
%Cpu(s): 52.9 us, 0.8 sy, 0.0 ni, 46.2 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
KiB Mem : 508556 total, 18668 free, 244684 used, 245204 buff/cache
KiB Swap: 1048572 total, 1047804 free, 768 used. 256476 avail Mem

PID USER      PR NI    VIRT    RES    SHR S %CPU %MEM     TIME+ COMMAND
4382 root      20  0 393524 127384 56120 R 100.0 25.0 0:26.62 python3

...

```

*Gambar 4.12: Potongan isi file arm-laplacian1.txt.*

## 4.2 Analisis Kinerja

### 4.2.1 Waktu Komputasi

Data waktu komputasi dengan menggunakan 50 frame pada masing-masing kernel dapat dilihat pada tabel 4.1 dan grafik pada gambar 4.14. Secara umum waktu komputasi dengan menggunakan prosesor ARM lebih lambat daripada waktu komputasi dengan menggunakan FPGA. Rata-rata waktu komputasi dengan prosesor ARM menggunakan 50 frame adalah 7,26 detik, sedangkan rata-rata waktu komputasi dengan FPGA menggunakan 50 frame hanya 0,82 detik.

```

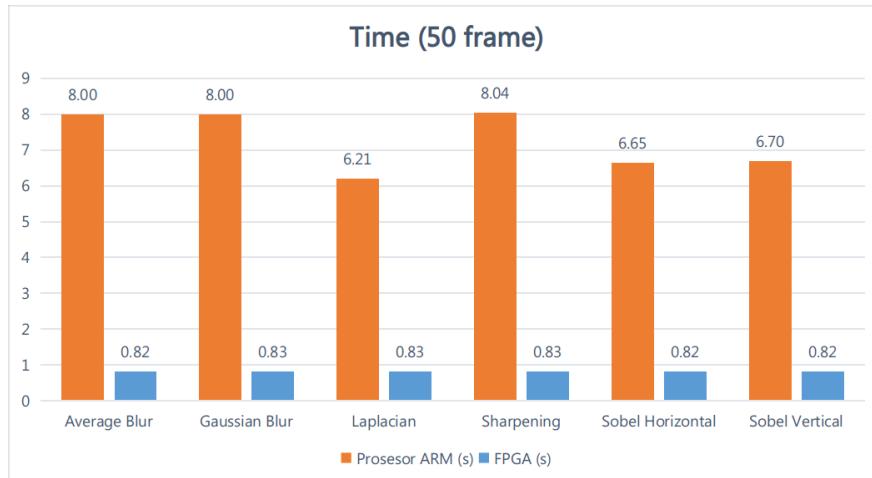
PID, USER, PR, NI, VIRT, RES, SHR, S, CPU, MEM, TIME, COMMAND
4382, root, 20, 0, 388124, 125980, 54808, S, 0.0, 24.8, 0:24.96, python3
4382, root, 20, 0, 392624, 127384, 56120, R, 77.0, 25.0, 0:25.43, python3
4382, root, 20, 0, 393524, 127384, 56120, R, 100.0, 25.0, 0:26.62, python3
4382, root, 20, 0, 393524, 127384, 56120, R, 96.7, 25.0, 0:27.20, python3
4382, root, 20, 0, 393524, 127384, 56120, R, 100.0, 25.0, 0:27.81, python3
4382, root, 20, 0, 393524, 127384, 56120, R, 100.0, 25.0, 0:29.00, python3
4382, root, 20, 0, 393524, 127384, 56120, R, 98.3, 25.0, 0:29.59, python3
4382, root, 20, 0, 393524, 127384, 56120, R, 100.0, 25.0, 0:34.93, python3
4382, root, 20, 0, 393524, 127384, 56120, R, 100.0, 25.0, 0:35.53, python3
4382, root, 20, 0, 393524, 127384, 56120, S, 0.0, 25.0, 0:35.67, python3

```

*Gambar 4.13: Isi file arm-laplacian1.csv.*

*Tabel 4.1: Tabel perbandingan waktu komputasi dengan menggunakan 50 frame.*

Filter	Prosesor ARM (s)	FPGA (s)
Average Blur	8.003612137	0.823560476
Gaussian Blur	7.998623228	0.827384996
Laplacian	6.210968781	0.827101517
Sharpening	8.041392469	0.825223923
Sobel Horizontal	6.646468353	0.823914003
Sobel Vertical	6.696593809	0.823559713
Rata-rata	7.266276463	0.825124105



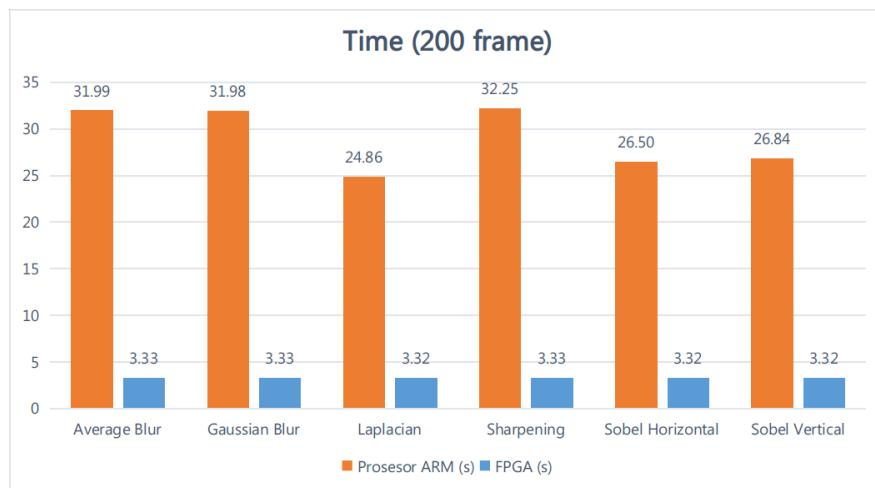
*Gambar 4.14: Grafik perbandingan waktu komputasi dengan 50 frame dan 200 frame*

Data waktu komputasi menggunakan 200 frame pada masing-masing kernel dapat dilihat pada tabel 4.2 dan grafik pada gambar 4.15. Rata-rata waktu komputasi dengan prosesor ARM menggunakan 200 frame adalah 29,06 detik, sedangkan

rata-rata waktu komputasi dengan FPGA menggunakan 200 frame hanya 3,32 detik.

Tabel 4.2: Tabel perbandingan waktu komputasi dengan menggunakan 200 frame.

Filter	Prosesor ARM (s)	FPGA (s)
Average Blur	31.9899159	3.325525188
Gaussian Blur	31.97958055	3.325351763
Laplacian	24.85993662	3.323753452
Sharpening	32.24906039	3.328786802
Sobel Horizontal	26.49739237	3.32414484
Sobel Vertical	26.84196448	3.324060822
Rata-rata	29.06964172	3.325270478



Gambar 4.15: Grafik perbandingan waktu komputasi dengan 50 frame dan 200 frame

Waktu komputasi tercepat dengan menggunakan prosesor ARM terdapat pada filter *laplacian* yaitu 6,21 detik dengan 50 frame dan 24,85 detik dengan 200 frame. Sedangkan waktu komputasi paling lambat ketika menggunakan prosesor ARM terdapat pada filter *sharpening* yaitu 8,04 detik dengan 50 frame dan 32,24 detik dengan 200 frame.

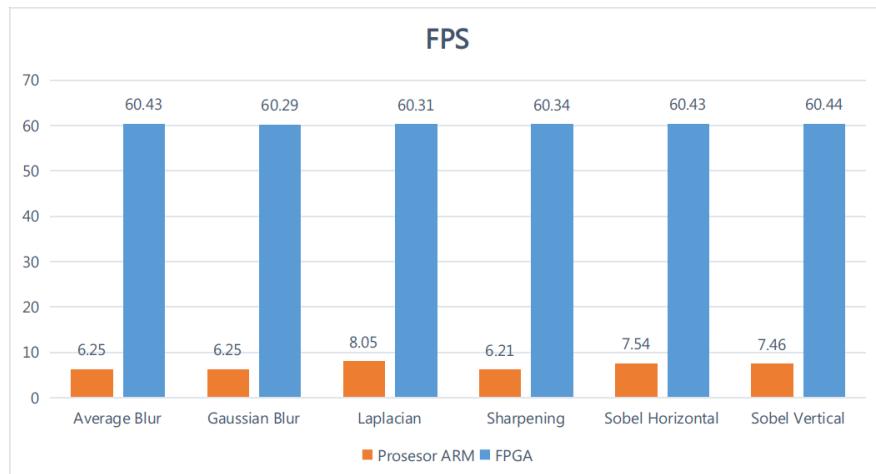
#### 4.2.2 Frame Rate (FPS)

Dengan mengetahui waktu komputasi dan jumlah frame maka frame rate atau FPS dapat dihitung menggunakan persamaan 2.10. Data FPS dari masing-masing

kernel dengan prosesor ARM dan FPGA dapat dilihat pada tabel 4.3 dan grafik pada gambar 4.16.

*Tabel 4.3: Tabel perbandingan FPS dengan menggunakan prosesor ARM dan FPGA.*

Filter	Prosesor ARM	FPGA
Average Blur	6.249583533	60.42684593
Gaussian Blur	6.25253493	60.28874396
Laplacian	8.047839131	60.31306253
Sharpening	6.209782985	60.33633034
Sobel Horizontal	7.53538058	60.42633377
Sobel Vertical	7.459019618	60.44047449
Rata-rata	6.959023463	60.37196517



*Gambar 4.16: Grafik perbandingan FPS dengan menggunakan prosesor ARM dan FPGA.*

Pada tabel 4.3 terlihat dengan menggunakan prosesor ARM diperoleh rata-rata 6.95 frame per detik (FPS), sedangkan ketika menggunakan FPGA diperoleh rata-rata 60.37 frame per detik. Terlihat pada grafik 4.16 nilai FPS dengan FPGA jauh lebih tinggi daripada dengan prosesor ARM.

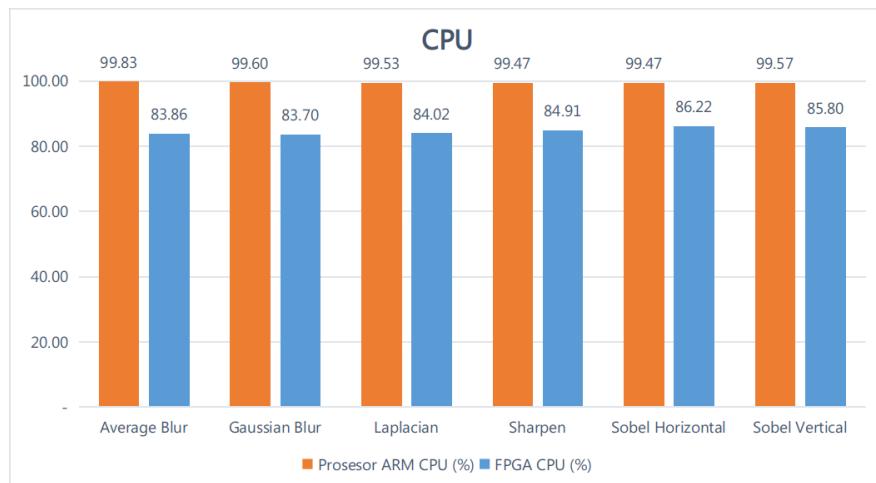
#### 4.2.3 Penggunaan CPU

Data perbandingan penggunaan CPU pada masing-masing kernel dengan prosesor ARM dan FPGA dapat dilihat pada tabel 4.4 dan grafik pada gambar 4.17. Rata-rata penggunaan CPU dengan prosesor ARM adalah 99.58% sedangkan dengan

FPGA diperoleh 84,75%. Data ini menunjukkan bahwa penggunaan CPU dengan prosesor ARM sedikit lebih besar daripada dengan FPGA.

*Tabel 4.4: Tabel perbandingan penggunaan CPU dengan menggunakan prosesor ARM dan FPGA.*

Filter	Prosesor ARM (%)	FPGA (%)
Average Blur	99.83	83.86
Gaussian Blur	99.60	83.70
Laplacian	99.53	84.02
Sharpening	99.47	84.91
Sobel Horizontal	99.47	86.22
Sobel Vertical	99.57	85.80
Rata-rata	99.58	84.75



*Gambar 4.17: Grafik perbandingan penggunaan CPU dengan menggunakan prosesor ARM dan FPGA.*

Penggunaan CPU terbesar dengan prosesor ARM yaitu pada kernel *average blur* (99,83%) dan dengan FPGA pada kernel *sobel horizontal* (86,22%). Penggunaan CPU terkecil dengan prosesor ARM yaitu pada kernel *sharpening* dan *sobel horizontal* (99,47%) dan dengan FPGA pada kernel *gaussian blur* (83,70%).

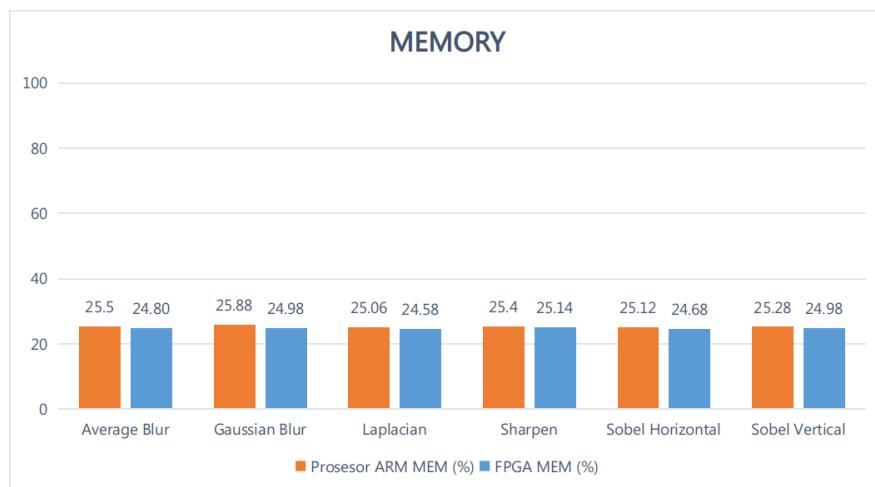
#### 4.2.4 Penggunaan Memory

Data penggunaan *memory* dengan prosesor ARM dan FPGA dapat dilihat pada tabel 4.5 dan grafik pada gambar 4.18. Data ini menunjukkan persentase *memory*

yang digunakan pada masing-masing kernel. Rata-rata penggunaan *memory* dengan prosesor ARM adalah 25,37% dan 24,86% dengan FPGA.

*Tabel 4.5: Tabel perbandingan penggunaan memory dengan menggunakan prosesor ARM dan FPGA.*

Filter	Prosesor ARM (%)	FPGA (%)
Average Blur	25.50	24.80
Gaussian Blur	25.88	24.98
Laplacian	25.06	24.58
Sharpening	25.40	25.14
Sobel Horizontal	25.12	24.68
Sobel Vertical	25.28	24.98
Rata-rata	25.37	24.86



*Gambar 4.18: Grafik perbandingan penggunaan memory dengan menggunakan prosesor ARM dan FPGA.*

Walaupun FPGA lebih baik daripada prosesor ARM pada segi waktu komputasi dan FPS namun penggunaan *memory* pada penerapan filter ini terlihat tidak jauh berbeda. Penggunaan *memory* FPGA ini hanya 0,51% lebih rendah dari penggunaan *memory* dengan prosesor ARM.

#### 4.2.5 Resident Memory (RES)

Data penggunaan *resident memory* atau RES dengan prosesor ARM dan FPGA dapat dilihat pada tabel 4.6 dan grafik pada gambar 4.19. Data ini menunjukkan

banyaknya RES (dalam satuan *kilobyte*) yang digunakan pada saat penerapan filter spasial pada video *stream* dengan masing-masing kernel.

*Tabel 4.6: Tabel perbandingan penggunaan resident memory (RES) dengan menggunakan prosesor ARM dan FPGA.*

Filter	Prosesor ARM (KiB)	FPGA (KiB)
Average Blur	129794.80	126097.60
Gaussian Blur	131804.40	127135.20
Laplacian	127493.60	125005.60
Sharpening	129117.22	127931.20
Sobel Horizontal	127830.40	125420.00
Sobel Vertical	128611.20	127033.60
Rata-rata	129108.60	126437.20



*Gambar 4.19: Grafik perbandingan penggunaan resident memory (RES) dengan menggunakan prosesor ARM dan FPGA.*

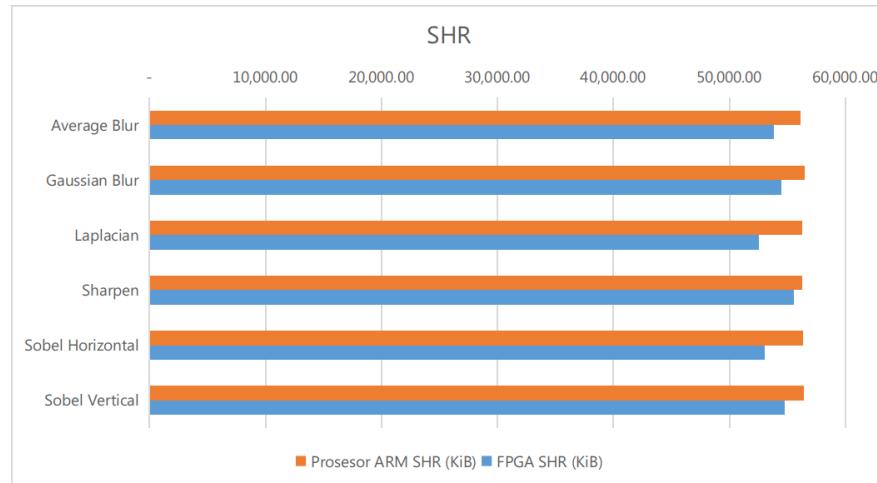
Rata-rata RES yang digunakan pada prosesor ARM adalah 129108,60 KiB dan 126437,20 KiB pada FPGA. Terlihat bahwa penggunaan RES pada prosesor ARM dan FPGA juga tidak jauh berbeda. Penggunaan RES terbesar dengan prosesor ARM yaitu pada kernel *gaussian blur* 131804,40 KiB, sedangkan dengan FPGA yaitu pada kernel *sharpening* 127931,20 KiB.

#### 4.2.6 Shared Memory (SHR)

Data penggunaan *shared memory* dengan prosesor ARM dan FPGA dapat dilihat pada tabel 4.7 dan grafik pada gambar 4.20. Data ini menunjukkan banyaknya *shared memory* (dalam satuan *kilobyte*) yang digunakan pada saat penerapan filter spasial pada video *stream* dengan masing-masing kernel.

Tabel 4.7: Tabel perbandingan penggunaan *shared memory* (SHR) dengan menggunakan prosesor ARM dan FPGA.

Filter	Prosesor ARM (KiB)	FPGA (KiB)
Average Blur	56,157.40	53,840.00
Gaussian Blur	56,503.60	54,504.80
Laplacian	56,248.00	52,568.00
Sharpen	56,298.82	55,528.80
Sobel Horizontal	56,349.60	53,015.20
Sobel Vertical	56,396.00	54,728.00
Rata-rata	56,325.57	54,030.80



Gambar 4.20: Grafik perbandingan penggunaan *shared memory* (SHR) dengan menggunakan prosesor ARM dan FPGA.

Rata-rata penggunaan *shared memory* pada prosesor ARM adalah 56325,57 KiB dan 54030,80 KiB pada FPGA. Terlihat bahwa penggunaan *shared memory* pada prosesor ARM sedikit lebih besar daripada FPGA. Penggunaan *shared memory* terbesar pada prosesor ARM yaitu pada kernel *gaussian blur* 56503,60 KiB dan

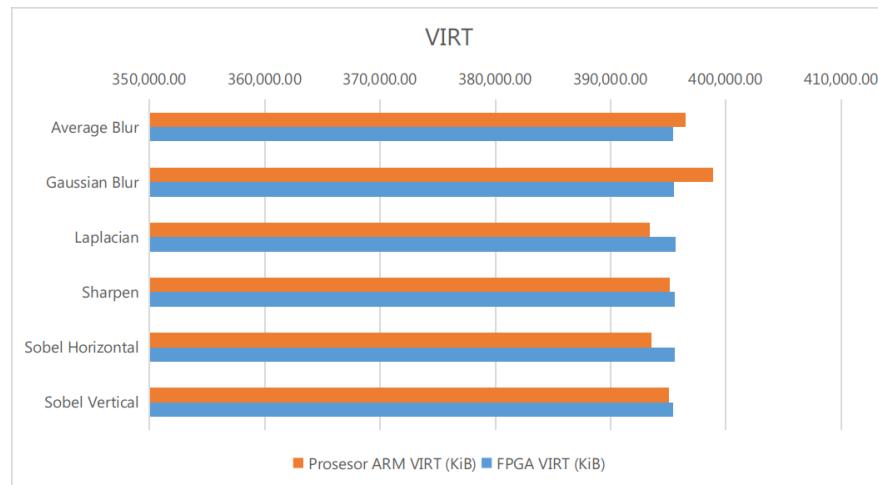
kernel *laplacian* 55528,80 KiB pada FPGA. Penggunaan *shared memory* terkecil pada prosesor ARM yaitu pada kernel *average blur* 56157,40 KiB dan kernel *laplacian* 42568 KiB pada FPGA.

#### 4.2.7 Virtual Memory (VIRT)

Data penggunaan *virtual memory* atau VIRT dapat dilihat pada tabel 4.8 dan grafik pada gambar 4.21. Data ini menunjukkan banyaknya *virtual memory* (dalam satuan *kilobyte*) yang digunakan pada saat penerapan filter spasial pada video *stream* dengan masing-masing kernel.

*Tabel 4.8: Tabel perbandingan penggunaan virtual memory (VIRT) dengan menggunakan prosesor ARM dan FPGA.*

Filter	Prosesor ARM (KiB)	FPGA (KiB)
Average Blur	396,474.64	395,396.00
Gaussian Blur	398,862.80	395,488.80
Laplacian	393,388.00	395,638.40
Sharpen	395,126.77	395,575.20
Sobel Horizontal	393,544.80	395,524.00
Sobel Vertical	395,048.80	395,385.60
Rata-rata	395,407.64	395,501.33



*Gambar 4.21: Grafik perbandingan penggunaan virtual memory (VIRT) dengan menggunakan prosesor ARM dan FPGA.*

Rata-rata penggunaan VIRT pada prosesor ARM adalah 395407,64 KiB dan

395501,33 KiB pada FPGA. Rata-rata penggunaan VIRT pada FPGA sedikit lebih tinggi dari pada prosesor ARM. Penggunaan VIRT terbesar dengan prosesor ARM yaitu pada kernel *gaussian blur* 398862,80 KiB dan kernel *laplacian* 395638,40 KiB pada FPGA. Penggunaan VIRT terkecil dengan prosesor ARM yaitu pada kernel *laplacian* 393388,00 KiB dan kernel *sobel vertical* 395385,60 KiB pada FPGA.

## **BAB V**

### **KESIMPULAN DAN SARAN**

#### **5.1 Kesimpulan**

Berdasarkan hasil penerapan filter spasial linear pada FPGA Development Board dengan menggunakan 6 kernel, peneliti dapat menarik beberapa kesimpulan sebagai berikut:

1. Proses implementasi filter spasial linear pada video *stream* dengan FPGA Development Board dilakukan dengan library OpenCV python dan library xfOpenCV Xilinx. Setiap *frame* dari *source* video *stream* direpresentasikan sebagai citra digital kemudian dilakukan filter spasial linear, selanjutnya hasil filter ini ditampilkan secara berkesinambungan sehingga tampak seperti video.
2. Waktu komputasi dan FPS dengan menggunakan FPGA secara umum lebih baik dibandingkan dengan menggunakan prosesor ARM. Penggunaan CPU pada FPGA sedikit lebih rendah dibandingkan penggunaan CPU pada prosesor ARM. Secara umum penggunaan *memory*, *shared memory*, *virtual memory*, dan *resident memory* pada FPGA tidak jauh berbeda dengan yang digunakan pada prosesor ARM.

#### **5.2 Saran**

Setelah melakukan penelitian ini, peneliti dapat memberikan beberapa saran sebagai berikut:

1. Pada penelitian ini peneliti hanya menggunakan 6 kernel dengan ukuran 3x3 yaitu *average blur*, *gaussian blur*, *laplacian*, *sharpening*, *sobel horizontal* dan *sobel vertical*. Untuk kedepannya, disarankan untuk menggunakan kernel lain dengan ukuran yang lebih beragam.
2. Pada penelitian ini peneliti hanya menggunakan metode pemrosesan citra *low-level* yaitu filter spasial linear, kedepannya disarankan menggunakan metode pemrosesan citra *mid-level* atau pemrosesan *high-level*.
3. Melakukan pengolahan citra digital pada jenis citra warna, tidak terbatas hanya pada citra *grayscale* saja.

## DAFTAR PUSTAKA

- Biswas, Priyabrata (2019). *Introduction to FPGA and its Architecture*. <https://towardsdatascience.com/introduction-to-fpga-and-its-architecture-20a62c14421c>. Accessed on 2020-06-18.
- Castellano, G. dkk. (Jan. 2019). “An FPGA-Oriented Algorithm for Real-Time Filtering of Poisson Noise in Video Streams, with Application to X-Ray Fluoroscopy”. In: *Circuits, Systems, and Signal Processing*. doi: 10.1007/s00034-018-01020-x.
- Cheung, Peter (2019). *Introduction to FPGAs*. [http://www.ee.ic.ac.uk/pcheung/teaching/ee2\\_digital/Lecture2-IntroductiontoFPGAs.pdf](http://www.ee.ic.ac.uk/pcheung/teaching/ee2_digital/Lecture2-IntroductiontoFPGAs.pdf). Accessed on 2020-04-19.
- Gonzalez, Rafael C. and Richard E. Woods (2001). *Digital Image Processing*. 2nd. ISBN-13: 978-0201180756. Upper Saddle River, New Jersey 07458: Prentice Hall.
- Jingbo, Xu dkk. (Aug. 2011). “A New Method for Realizing LOG Filter in Image Edge Detection”. In: *The 6th International Forum on Strategic Technology*. doi: 10.1109/IFOST.2011.6021127.
- Kowalczyk, Marcin, Dominika Przewlocka, and Tomasz Krvjak (Oct. 2018). “Real-Time Implementation of Contextual Image Processing Operations for 4K Video Stream in Zynq UltraScale+ MPSoC”. In: *2018 Conference on Design and Architectures for Signal and Image Processing (DASIP)*. doi: 10.1109/DASIP.2018.8597105.
- Linux (n.d.). *Top Manual page*. Linux Version. Ubuntu.
- Madhusudana, P. C. dkk. (2020). “Capturing Video Frame Rate Variations via Entropic Differencing”. In: *IEEE Signal Processing Letters* 27, pp. 1809–1813. doi: 10.1109/LSP.2020.3028687.
- Putra, Darma (2010). *Pengolahan Citra Digital*. ISBN-13: 978-979-29-1443-6. Jl. Beo 38-40, Yogyakarta 55281: Penerbit Andi.
- Raj., S.M. Alex, Rita Maria Abraham, and M.H. Supriya (Sept. 2016). “Spatial Filtering Based Boundary Extraction in Underwater Images for Pipeline

- Detection: FPGA Implementation”. In: *International Journal of Computer Science and Information Security (IJCSIS)*. Vol. 14, No. 9.
- Rinaldi, Munir (2004). *Pengolahan Citra Digital dengan Pendekatan Algoritmik*. ISBN: 979-3338296. Bandung: Penerbit Informatika.
- S, Lars dkk. (2020). *The Linux System Administrator's Guide Chapter 6. Memory Management*. <https://tldp.org/LDP/sag/html/vm-intro.html>. Accessed on 2021-02-22.
- Sadangi, Sushant dkk. (May 2017). “FPGA Implementation of Spatial Filtering techniques for 2D Images”. In: *IEEE International Conference On Recent Trends in Electronics Information & Communication Technology (RTEICT)*.
- Silberschatz, Avi, Peter Baer Galvin, and Greg Gagne (2009). *Operationg System Concepts*. ISBN: 978-0-470-12872-5. John Wiley and Sons, Inc.
- Silva, Eduardo A.B. da and Gelson V. Mendonca (2005). “4 - Digital Image Processing”. In: *The Electrical Engineering Handbook*. Ed. by Wai-Kai Chen. Burlington: Academic Press, pp. 891–910. ISBN: 978-0-12-170960-0. doi: <https://doi.org/10.1016/B978-012170960-0/50064-5>.
- Sutoyo, T. dkk. (2009). *Teori Pengolahan Citra Digital*. ISBN-13: 978-979-29-0974-6. Jl. Beo 38-40, Yogyakarta 55281: Penerbit Andi.
- Tan, Xin dkk. (Feb. 2014). “A Real-time Video Denoising Algorithm with FPGA Implementation for Poisson-Gaussian Noise”. In: *J Real-Time Image Proc.* doi: 10.1007/s11554-014-0405-2.
- Ustyukov, Dmitry I., Alex I. Efimov, and Dmitry A. Kolchaev (June 2019). “Features of Image Spatial Filters Implementation on FPGA”. In: *Mediterranean Conference On Embedded Computing (Meco)*.
- Xilinx (2020). *Field Programmable Gate Array (FPGA)*. <https://www.xilinx.com/products/silicon-devices/fpga/what-is-an-fpga.html>. Accessed on 2020-04-17.
- Yang, Ching-Chung (Sept. 2013). “Finest Image Sharpening by Sse of the Modified Mask Filter Dealing with Highest Spatial Frequencies”. In: *OPTIK*. doi: 10.1016/j.ijleo.2013.09.070.
- Zhao, Jin (Apr. 2015). “Video/Image Processing on FPGA”. Master thesis. Worcester Polytechnic Institute.

## LAMPIRAN

### 7.3 Sourcede Program

#### 7.3.1 ARM Prosesor

```
import os
os.getpid()

# Load filter2D + dilate overlay
from pynq import Overlay
bareHDMI = Overlay("/usr/local/lib/python3.6/dist-packages/
    pynq_cv/overlays/xv2Filter2DDilate.bit")
bareHDMI.download()
import pynq_cv.overlays.xv2Filter2DDilate as xv2

# Load xlnk memory manager
from pynq import Xlnk
Xlnk.set_allocator_library("/usr/local/lib/python3.6/
    dist-packages/pynq_cv/overlays/xv2Filter2DDilate.so")
mem_manager = Xlnk()

hdmi_in = bareHDMI.video.hdmi_in
hdmi_out = bareHDMI.video.hdmi_out

mymode = hdmi_in.mode
print("My mode: "+str(mymode))

height = hdmi_in.mode.height
width = hdmi_in.mode.width
bpp = hdmi_in.mode.bits_per_pixel

hdmi_in.tie(hdmi_out)
```

```

import numpy as np
import cv2
import time

kernel = {
    'average blur': np.array([
        [1.0, 1.0, 1.0],
        [1.0, 1.0, 1.0],
        [1.0, 1.0, 1.0]],np.float32)/9,
    'gaussian blur': np.array([
        [1.0, 2.0, 1.0],
        [2.0, 4.0, 2.0],
        [1.0, 2.0, 1.0]],np.float32)/16,
    'sobel ver': np.array([
        [1.0,0.0,-1.0],
        [2.0,0.0,-2.0],
        [1.0,0.0,-1.0]],np.float32),
    'sobel hor': np.array([
        [1.0,2.0,1.0],
        [0.0,0.0,0.0],
        [-1.0,-2.0,-1.0]],np.float32),
    'laplacian': np.array([
        [0.0, 1.0, 0],
        [1.0, -4, 1.0],
        [0, 1.0, 0.0]],np.float32),
    'sharpen': np.array([
        [-1,-1, -1],
        [-1, 9, -1],
        [-1, -1, -1]],np.float32),
}

kernel_name = 'sobel hor'
numberOfIterations=85

```

```

startSW=time.time()
for i in range(numberOfIterations):
    inframe = hdmi_in.readframe()
    outframe = hdmi_out.newframe()
    cv2.filter2D(inframe, -1, kernel.get(kernel_name),dst=outframe,
                 borderType=cv2.BORDER_CONSTANT) #filter2D on ARM
    hdmi_out.writeframe(outframe)
stopSW=time.time()
print("Start SW loop = ", (stopSW - startSW))
print("SW frames per second: ",
      ((numberOfIterations) / (stopSW - startSW)))

hdmi_out.close()
hdmi_in.close()

```

### 7.3.2 FPGA Prosesor

```

import os
os.getpid()

# Load filter2D + dilate overlay
from pynq import Overlay
bareHDMI = Overlay("/usr/local/lib/python3.6/dist-packages/
                    pynq_cv/overlays/xv2Filter2DDilate.bit")
bareHDMI.download()
import pynq_cv.overlays.xv2Filter2DDilate as xv2

# Load xlkn memory manager
from pynq import Xlnk
Xlnk.set_allocator_library("/usr/local/lib/python3.6/
                            dist-packages/pynq_cv/overlays/xv2Filter2DDilate.so")
mem_manager = Xlnk()

```

```

hdmi_in = bareHDMI.video.hdmi_in
hdmi_out = bareHDMI.video.hdmi_out

from pynq.lib.video import *
hdmi_in.configure(PIXEL_GRAY)
hdmi_out.configure(hdmi_in.mode)

hdmi_in.cacheable_frames = False
hdmi_out.cacheable_frames = False

hdmi_in.start()
hdmi_out.start()

mymode = hdmi_in.mode
print("My mode: "+str(mymode))

height = hdmi_in.mode.height
width = hdmi_in.mode.width
bpp = hdmi_in.mode.bits_per_pixel

hdmi_in.tie(hdmi_out)

import numpy as np
import time

dstSW = np.ones((height,width),np.uint8);
xFdst = mem_manager.cma_array((height,width),np.uint8)

kernel = {
    'average blur': np.array([
        [1.0, 1.0, 1.0],
        [1.0, 1.0, 1.0],
        [1.0, 1.0, 1.0],
    ])
}

```

```

[1.0, 1.0, 1.0]],np.float32)/9,
'gaussian blur': np.array([
[1.0, 2.0, 1.0],
[2.0, 4.0, 2.0],
[1.0, 2.0, 1.0]],np.float32)/16,
'sobel ver': np.array([
[1.0,0.0,-1.0],
[2.0,0.0,-2.0],
[1.0,0.0,-1.0]],np.float32),
'sobel hor': np.array([
[1.0,2.0,1.0],
[0.0,0.0,0.0],
[-1.0,-2.0,-1.0]],np.float32),
'laplacian': np.array([
[0.0, 1.0, 0],
[1.0, -4, 1.0],
[0, 1.0, 0.0]],np.float32),
'sharpen': np.array([
[-1,-1, -1],
[-1, 9, -1],
[-1, -1, -1]],np.float32),
}

kernel_name = 'sharpen'
numberOfIterations=300

startPL=time.time()
for i in range(numberOfIterations):
    inframe = hdmi_in.readframe()
    outframe = hdmi_out.newframe()
    xv2.filter2D(inframe, -1, kernel.get(kernel_name),
                 dst=outframe,borderType=cv2.BORDER_CONSTANT)
    hdmi_out.writeframe(outframe)

```

```

stopPL=time.time()
print("Start HW loop = ", (stopPL - startPL))
print("PL frames per second: ",
((numberOfIterations) / (stopPL - startPL)))

hdmi_out.close()
hdmi_in.close()

```

## 7.4 Data Hasil Percobaan Waktu Komputasi dan FPS

### 7.4.1 Average Blur

Frame	ARM-time(s)	ARM-fps	FPGA-time(s)	FPGA-fps
50	8.024973154	6.230550438	0.821577311	60.85854533
50	7.99996233	6.25002943	0.823753595	60.69776239
50	7.998621464	6.251077167	0.82895875	60.31663218
50	7.993776083	6.254866221	0.822452068	60.79381635
50	8.000727654	6.249431572	0.821060658	60.89684062
200	32.04833198	6.240574397	3.320029736	60.24042431
200	31.9667685	6.256497274	3.321809292	60.20815237
200	31.97968745	6.253969814	3.329262733	60.07336038
200	31.99306297	6.251355182	3.325505972	60.14122413
200	31.96172857	6.257483839	3.331018209	60.04170119

### 7.4.2 Gaussian Blur

Frame	ARM-time(s)	ARM-fps	FPGA-time(s)	FPGA-fps
50	8.001603127	6.248747808	0.8199687	60.9779373
50	7.993735313	6.254898122	0.835051775	59.87652682
50	7.99848032	6.251187476	0.826747656	60.47794589
50	8.005704403	6.245546611	0.829063416	60.30901746
50	7.993592978	6.255009498	0.826093435	60.5258411

Frame	ARM-time(s)	ARM-fps	FPGA-time(s)	FPGA-fps
200	31.97753382	6.254391009	3.326043844	60.13149837
200	31.97605824	6.254679625	3.326021433	60.13190355
200	31.97065854	6.255736013	3.321739435	60.20941856
200	31.96669817	6.25651104	3.325276613	60.14537233
200	32.00695395	6.248642101	3.327677488	60.10197824

#### 7.4.3 Laplacian

Frame	ARM-time(s)	ARM-fps	FPGA-time(s)	FPGA-fps
50	6.254891157	7.993744215	0.82506156	60.60153865
50	6.168808699	8.105292682	0.826088667	60.52619047
50	6.242358208	8.009793468	0.822906017	60.76027997
50	6.161360025	8.11509144	0.82921052	60.29831847
50	6.227425814	8.028999702	0.83224082	60.07876423
200	24.89130092	8.034935605	3.326535225	60.12261602
200	24.88935256	8.035564586	3.317910194	60.27890699
200	24.87290311	8.040878828	3.316281796	60.30850583
200	24.86412835	8.043716521	3.330517769	60.050723
200	24.78199816	8.070374258	3.327522278	60.10478167

#### 7.4.4 Sharpening

Frame	ARM-time(s)	ARM-fps	FPGA-time(s)	FPGA-fps
50	8.043478489	6.21621604	0.830551624	60.20095384
50	8.045930862	6.214321358	0.824851274	60.61698823
50	8.042553186	6.216931221	0.825832129	60.54499247
50	8.034646034	6.223049502	0.820035696	60.9729555
50	8.040353775	6.218631841	0.82484889	60.61716344
200	32.23359871	6.204705897	3.323871374	60.1708001
200	32.25315762	6.20094325	3.324242353	60.16408515
200	32.30215144	6.191538058	3.328672647	60.08400981

Frame	ARM-time(s)	ARM-fps	FPGA-time(s)	FPGA-fps
200	32.21903443	6.207510669	3.334614992	59.97693901
200	32.23735976	6.20398201	3.332532644	60.01441587

#### 7.4.5 Sobel Horizontal

Frame	ARM-time(s)	ARM-fps	FPGA-time(s)	FPGA-fps
50	6.6412673	7.52868357	0.824651003	60.63170944
50	6.666188955	7.500537464	0.824263334	60.66022583
50	6.635772943	7.534917247	0.819379091	61.02181583
50	6.6434834	7.526172188	0.822708607	60.77485952
50	6.645629168	7.523742108	0.828567982	60.34507862
200	26.39209056	7.578027953	3.319519758	60.24967904
200	26.52648592	7.539634183	3.320359707	60.23443773
200	26.5723536	7.526619697	3.328252077	60.09160225
200	26.48957157	7.550140985	3.328671694	60.08402702
200	26.50646019	7.545330405	3.323920965	60.16990238

#### 7.4.6 Sobel Vertical

Frame	ARM-time(s)	ARM-fps	FPGA-time(s)	FPGA-fps
50	6.715774775	7.445157361	0.81825161	61.10589872
50	6.589876413	7.587395706	0.825090408	60.59941977
50	6.717816114	7.442895005	0.824631214	60.63316443
50	6.746128082	7.411658864	0.82959795	60.27015858
50	6.713373661	7.447820206	0.820227385	60.95870601
200	26.9272964	7.42740738	3.325917721	60.13377864
200	26.74835086	7.477096478	3.325239897	60.14603644
200	26.82612038	7.45542021	3.324659824	60.15653046
200	26.8522346	7.448169695	3.315615892	60.3206181
200	26.85582018	7.447175274	3.328870773	60.08043376

## 7.5 Data Hasil Percobaan ARM Average Blur

### 7.5.1 Percobaan 1

PID	USER	PR	NI	VIRT	RES	SHR	S	CPU	MEM	TIME
1842	root	20	0	403380	134772	54944	R	100.0	26.5	3:48.28
1842	root	20	0	403380	134772	54944	R	100.0	26.5	3:48.78
1842	root	20	0	403380	134772	54944	R	100.0	26.5	3:49.29
1842	root	20	0	403380	134772	54944	R	100.0	26.5	3:49.79
1842	root	20	0	403380	134772	54944	R	100.0	26.5	3:50.30
1842	root	20	0	403380	134772	54944	R	100.0	26.5	3:50.80
1842	root	20	0	403380	134772	54944	R	100.0	26.5	3:51.30
1842	root	20	0	403380	134772	54944	R	100.0	26.5	3:51.80
1842	root	20	0	403380	134772	54944	R	100.0	26.5	3:52.30
1842	root	20	0	403380	134772	54944	R	100.0	26.5	3:52.81
1842	root	20	0	403380	134772	54944	R	100.0	26.5	3:53.31
1842	root	20	0	403380	134772	54944	R	100.0	26.5	3:53.81
1842	root	20	0	403380	134772	54944	R	100.0	26.5	3:54.31
1842	root	20	0	403380	134772	54944	R	100.0	26.5	3:54.81
1842	root	20	0	403380	134772	54944	R	100.0	26.5	3:55.32
1842	root	20	0	403380	134772	54944	R	100.0	26.5	3:55.82
1842	root	20	0	403380	134772	54944	R	100.0	26.5	3:56.32
1842	root	20	0	403380	134772	54944	R	100.0	26.5	3:56.83
1842	root	20	0	403380	134772	54944	R	100.0	26.5	3:57.33
1842	root	20	0	403380	134772	54944	R	100.0	26.5	3:57.83
1842	root	20	0	403380	134772	54944	R	100.0	26.5	3:58.33
1842	root	20	0	403380	134772	54944	R	100.0	26.5	3:58.83
1842	root	20	0	403380	134772	54944	R	100.0	26.5	3:59.34
1842	root	20	0	403380	134772	54944	R	100.0	26.5	3:59.84
1842	root	20	0	403380	134772	54944	R	100.0	26.5	4:00.35
1842	root	20	0	403380	134772	54944	R	100.0	26.5	4:00.85
1842	root	20	0	403380	134772	54944	R	100.0	26.5	4:01.35

PID	USER	PR	NI	VIRT	RES	SHR	S	CPU	MEM	TIME
1842	root	20	0	403380	134772	54944	R	100.0	26.5	4:01.85
1842	root	20	0	403380	134772	54944	R	100.0	26.5	4:02.35
1842	root	20	0	403380	134772	54944	R	100.0	26.5	4:02.86
1842	root	20	0	403380	134772	54944	R	100.0	26.5	4:03.36
1842	root	20	0	403380	134772	54944	R	100.0	26.5	4:03.86
1842	root	20	0	403380	134772	54944	R	100.0	26.5	4:04.36
1842	root	20	0	403380	134772	54944	R	100.0	26.5	4:04.86
1842	root	20	0	403380	134772	54944	R	100.0	26.5	4:05.37
1842	root	20	0	403380	134772	54944	R	100.0	26.5	4:05.87
1842	root	20	0	403380	134772	54944	R	100.0	26.5	4:06.37
1842	root	20	0	403380	134772	54944	R	100.0	26.5	4:06.88
1842	root	20	0	403380	134772	54944	R	100.0	26.5	4:07.38
1842	root	20	0	403380	134772	54944	R	100.0	26.5	4:07.88
1842	root	20	0	403380	134772	54944	R	100.0	26.5	4:08.38
1842	root	20	0	403380	134772	54944	R	100.0	26.5	4:08.88
1842	root	20	0	403380	134772	54944	R	100.0	26.5	4:09.39
1842	root	20	0	403380	134772	54944	R	100.0	26.5	4:09.89
1842	root	20	0	403380	134772	54944	R	100.0	26.5	4:10.39
1842	root	20	0	403380	134772	54944	R	100.0	26.5	4:10.90
1842	root	20	0	403380	134772	54944	R	100.0	26.5	4:11.40
1842	root	20	0	403380	134772	54944	R	100.0	26.5	4:11.90
1842	root	20	0	403380	134772	54944	R	100.0	26.5	4:12.40
1842	root	20	0	403380	134772	54944	R	100.0	26.5	4:12.90
1842	root	20	0	403380	134772	54944	R	100.0	26.5	4:13.41
1842	root	20	0	403380	134772	54944	R	100.0	26.5	4:13.91
1842	root	20	0	403380	134772	54944	R	100.0	26.5	4:14.41
1842	root	20	0	403380	134772	54944	R	100.0	26.5	4:14.91
1842	root	20	0	403380	134772	54944	R	100.0	26.5	4:15.42
1842	root	20	0	403380	134772	54944	R	100.0	26.5	4:15.92
1842	root	20	0	403380	134772	54944	R	100.0	26.5	4:16.42

### 7.5.2 Percobaan 2

PID	USER	PR	NI	VIRT	RES	SHR	S	CPU	MEM	TIME
2484	root	20	0	390056	127984	55876	S	0.0	25.2	2:26.41

PID	USER	PR	NI	VIRT	RES	SHR	S	CPU	MEM	TIME
2484	root	20	0	390056	127984	55876	S	0.0	25.2	2:26.41
2484	root	20	0	390056	127984	55876	S	0.0	25.2	2:26.41
2484	root	20	0	390056	127984	55876	S	0.0	25.2	2:26.41
2484	root	20	0	390956	128396	56192	R	16.0	25.2	2:26.49
2484	root	20	0	394556	128832	56628	R	98.0	25.3	2:26.98
2484	root	20	0	395456	128832	56628	R	100.0	25.3	2:27.49
2484	root	20	0	395456	128832	56628	R	100.0	25.3	2:27.99
2484	root	20	0	395456	128832	56628	R	100.0	25.3	2:28.49
2484	root	20	0	395456	128832	56628	R	100.0	25.3	2:28.99
2484	root	20	0	395456	128832	56628	R	100.0	25.3	2:29.49
2484	root	20	0	395456	128832	56628	R	100.0	25.3	2:30.00
2484	root	20	0	395456	128832	56628	R	100.0	25.3	2:30.50
2484	root	20	0	395456	128832	56628	R	100.0	25.3	2:31.00
2484	root	20	0	395456	128832	56628	R	100.0	25.3	2:31.50
2484	root	20	0	395456	128832	56628	R	100.0	25.3	2:32.01
2484	root	20	0	395456	128832	56628	R	100.0	25.3	2:32.51
2484	root	20	0	395456	128832	56628	R	100.0	25.3	2:33.01
2484	root	20	0	395456	128832	56628	R	100.0	25.3	2:33.52
2484	root	20	0	395456	128832	56628	R	100.0	25.3	2:34.02
2484	root	20	0	395456	128832	56628	R	100.0	25.3	2:34.52
2484	root	20	0	395456	128832	56628	R	100.0	25.3	2:35.02
2484	root	20	0	395456	128832	56628	R	100.0	25.3	2:35.52
2484	root	20	0	395456	128832	56628	R	100.0	25.3	2:36.03
2484	root	20	0	395456	128832	56628	R	100.0	25.3	2:36.53
2484	root	20	0	395456	128832	56628	R	100.0	25.3	2:37.04
2484	root	20	0	395456	128832	56628	R	100.0	25.3	2:37.54
2484	root	20	0	395456	128832	56628	R	100.0	25.3	2:38.04
2484	root	20	0	395456	128832	56628	R	100.0	25.3	2:38.54
2484	root	20	0	395456	128832	56628	R	100.0	25.3	2:39.04
2484	root	20	0	395456	128832	56628	R	100.0	25.3	2:39.55

PID	USER	PR	NI	VIRT	RES	SHR	S	CPU	MEM	TIME
2484	root	20	0	395456	128832	56628	R	100.0	25.3	2:40.05
2484	root	20	0	395456	128832	56628	R	100.0	25.3	2:40.55
2484	root	20	0	395456	128832	56628	R	100.0	25.3	2:41.05
2484	root	20	0	395456	128832	56628	R	100.0	25.3	2:41.55
2484	root	20	0	395456	128832	56628	R	100.0	25.3	2:42.06
2484	root	20	0	395456	128832	56628	R	100.0	25.3	2:42.56
2484	root	20	0	395456	128832	56628	R	100.0	25.3	2:43.06
2484	root	20	0	395456	128832	56628	R	102.0	25.3	2:43.58
2484	root	20	0	395456	128832	56628	R	100.0	25.3	2:44.08
2484	root	20	0	395456	128832	56628	R	100.0	25.3	2:44.58
2484	root	20	0	395456	128832	56628	R	100.0	25.3	2:45.08
2484	root	20	0	395456	128832	56628	R	102.0	25.3	2:45.59
2484	root	20	0	395456	128832	56628	R	98.0	25.3	2:46.09
2484	root	20	0	395456	128832	56628	R	100.0	25.3	2:46.59
2484	root	20	0	395456	128832	56628	R	102.0	25.3	2:47.10
2484	root	20	0	395456	128832	56628	R	98.0	25.3	2:47.60
2484	root	20	0	395456	128832	56628	R	100.0	25.3	2:48.10
2484	root	20	0	395456	128832	56628	R	100.0	25.3	2:48.60
2484	root	20	0	395456	128832	56628	R	100.0	25.3	2:49.10
2484	root	20	0	395456	128832	56628	R	102.0	25.3	2:49.61
2484	root	20	0	395456	128832	56628	R	98.0	25.3	2:50.11
2484	root	20	0	395456	128832	56628	R	100.0	25.3	2:50.61
2484	root	20	0	395456	128832	56628	R	100.0	25.3	2:51.11
2484	root	20	0	395456	128832	56628	R	100.0	25.3	2:51.61
2484	root	20	0	395456	128832	56628	R	100.0	25.3	2:52.12
2484	root	20	0	395456	128832	56628	R	100.0	25.3	2:52.62
2484	root	20	0	395456	128832	56628	R	100.0	25.3	2:53.12
2484	root	20	0	395456	128832	56628	R	100.0	25.3	2:53.63
2484	root	20	0	395456	128832	56628	R	100.0	25.3	2:54.13
2484	root	20	0	395456	128832	56628	R	100.0	25.3	2:54.63

PID	USER	PR	NI	VIRT	RES	SHR	S	CPU	MEM	TIME
2484	root	20	0	395456	128832	56628	R	100.0	25.3	2:55.13
2484	root	20	0	395456	128832	56628	R	100.0	25.3	2:55.63
2484	root	20	0	395456	128832	56628	R	100.0	25.3	2:56.14
2484	root	20	0	395456	128832	56628	R	100.0	25.3	2:56.64
2484	root	20	0	395456	128832	56628	R	100.0	25.3	2:57.14
2484	root	20	0	395456	128832	56628	R	100.0	25.3	2:57.65
2484	root	20	0	395456	128832	56628	R	100.0	25.3	2:58.15
2484	root	20	0	395456	128832	56628	S	70.0	25.3	2:58.50
2484	root	20	0	395456	128832	56628	S	2.0	25.3	2:58.51
2484	root	20	0	395456	128832	56628	S	0.0	25.3	2:58.51
2484	root	20	0	395456	128832	56628	S	0.0	25.3	2:58.51
2484	root	20	0	395456	128832	56628	S	0.0	25.3	2:58.51
2484	root	20	0	395456	128832	56628	S	0.0	25.3	2:58.51

### 7.5.3 Percobaan 3

PID	USER	PR	NI	VIRT	RES	SHR	S	CPU	MEM	TIME
2581	root	20	0	395454	128830	56631	S	0.0	25.3	2:58.51
2581	root	20	0	395454	128830	56631	R	46.5	25.3	2:59.91
2581	root	20	0	395454	128830	56631	R	100.0	25.3	3:02.91
2581	root	20	0	395454	128830	56631	R	100.0	25.3	3:05.92
2581	root	20	0	395454	128830	56631	R	100.0	25.3	3:08.92
2581	root	20	0	395454	128830	56631	R	100.0	25.3	3:11.93
2581	root	20	0	395454	128830	56631	R	100.0	25.3	3:14.93
2581	root	20	0	395454	128830	56631	R	100.0	25.3	3:17.94
2581	root	20	0	395454	128830	56631	R	100.0	25.3	3:20.94
2581	root	20	0	395454	128830	56631	R	99.7	25.3	3:23.94
2581	root	20	0	395454	128830	56631	R	100.0	25.3	3:26.94
2581	root	20	0	395454	128830	56631	R	100.0	25.3	3:29.95

#### 7.5.4 Percobaan 4

PID	USER	PR	NI	VIRT	RES	SHR	S	CPU	MEM	TIME
2916	root	20	0	388276	126496	54860	S	0.0	24.9	1:32.89
2916	root	20	0	393676	127828	56100	R	42.0	25.1	1:34.15
2916	root	20	0	393676	127828	56100	R	99.7	25.1	1:37.15
2916	root	20	0	393676	127828	56100	R	100.0	25.1	1:40.15
2916	root	20	0	393676	127828	56100	R	100.0	25.1	1:43.16
2916	root	20	0	393676	127828	56100	R	100.0	25.1	1:46.16
2916	root	20	0	393676	127828	56100	S	94.7	25.1	1:49.01

#### 7.5.5 Percobaan 5

PID	USER	PR	NI	VIRT	RES	SHR	S	CPU	MEM	TIME
3629	root	20	0	389184	127288	55152	S	0.0	25.0	0:24.78
3629	root	20	0	389184	127288	55152	S	0.0	25.0	0:24.78
3629	root	20	0	389184	127288	55152	S	0.0	25.0	0:24.78
3629	root	20	0	390084	128256	56028	R	15.0	25.2	0:24.87
3629	root	20	0	393684	128712	56484	R	98.3	25.3	0:25.46
3629	root	20	0	394584	128712	56484	R	100.0	25.3	0:26.06
3629	root	20	0	394584	128712	56484	R	98.3	25.3	0:26.65
3629	root	20	0	394584	128712	56484	R	100.0	25.3	0:27.26
3629	root	20	0	394584	128712	56484	R	100.0	25.3	0:27.86
3629	root	20	0	394584	128712	56484	R	98.3	25.3	0:28.45
3629	root	20	0	394584	128712	56484	R	100.0	25.3	0:29.05
3629	root	20	0	394584	128712	56484	R	98.4	25.3	0:29.65
3629	root	20	0	394584	128712	56484	R	100.0	25.3	0:30.25
3629	root	20	0	394584	128712	56484	R	100.0	25.3	0:30.85
3629	root	20	0	394584	128712	56484	R	100.0	25.3	0:31.45
3629	root	20	0	394584	128712	56484	R	100.0	25.3	0:32.05
3629	root	20	0	394584	128712	56484	R	100.0	25.3	0:32.66
3629	root	20	0	394584	128712	56484	R	100.0	25.3	0:33.26

PID	USER	PR	NI	VIRT	RES	SHR	S	CPU	MEM	TIME
3629	root	20	0	394584	128712	56484	R	100.0	25.3	0:33.86
3629	root	20	0	394584	128712	56484	R	100.0	25.3	0:34.46
3629	root	20	0	394584	128712	56484	R	100.0	25.3	0:35.06
3629	root	20	0	394584	128712	56484	R	100.0	25.3	0:35.67
3629	root	20	0	394584	128712	56484	S	73.3	25.3	0:36.11
3629	root	20	0	394584	128712	56484	S	0.0	25.3	0:36.11

## 7.6 Data Hasil Percobaan ARM Gaussian Blur

### 7.6.1 Percobaan 1

PID	USER	PR	NI	VIRT	RES	SHR	S	CPU	MEM	TIME
2448	root	20	0	399900	136990	56630	S	0.0	26.9	3:42.74
2448	root	20	0	405302	136990	56630	R	26.0	26.9	3:43.52
2448	root	20	0	405302	136990	56630	R	100.0	26.9	3:46.53
2448	root	20	0	405302	136990	56630	R	100.0	26.9	3:49.53
2448	root	20	0	405302	136990	56630	R	100.0	26.9	3:52.54
2448	root	20	0	405302	136990	56630	R	99.7	26.9	3:55.53
2448	root	20	0	405302	136990	56630	R	100.0	26.9	3:58.54
2448	root	20	0	405302	136990	56630	R	100.0	26.9	4:01.54
2448	root	20	0	405302	136990	56630	R	100.0	26.9	4:04.55
2448	root	20	0	405302	136990	56630	R	100.0	26.9	4:07.55
2448	root	20	0	405302	136990	56630	R	100.0	26.9	4:10.56
2448	root	20	0	405302	136990	56630	R	100.0	26.9	4:13.56

### 7.6.2 Percobaan 2

PID	USER	PR	NI	VIRT	RES	SHR	S	CPU	MEM	TIME
2484	root	20	0	405300	136996	56628	S	0.0	26.9	4:14.83
2484	root	20	0	405300	136996	56628	R	41.7	26.9	4:16.08
2484	root	20	0	405300	136996	56628	R	100.0	26.9	4:19.09

PID	USER	PR	NI	VIRT	RES	SHR	S	CPU	MEM	TIME
2484	root	20	0	405300	136996	56628	R	100.0	26.9	4:22.09
2484	root	20	0	405300	136996	56628	R	100.0	26.9	4:25.10
2484	root	20	0	405300	136996	56628	R	100.0	26.9	4:28.10
2484	root	20	0	405300	136996	56628	S	92.7	26.9	4:30.89

### 7.6.3 Percobaan 3

PID	USER	PR	NI	VIRT	RES	SHR	S	CPU	MEM	TIME
3002	root	20	0	390048	128116	56004	S	0	25.2	1:31.00
3002	root	20	0	395448	128860	56696	R	100	25.3	1:35.45
3002	root	20	0	395448	128860	56696	R	100	25.3	1:38.45
3002	root	20	0	395448	128860	56696	R	100.3	25.3	1:41.47
3002	root	20	0	395448	128860	56696	R	99.7	25.3	1:44.46
3002	root	20	0	395448	128860	56696	S	88	25.3	1:47.11

### 7.6.4 Percobaan 4

PID	USER	PR	NI	VIRT	RES	SHR	S	CPU	MEM	TIME
3726	root	20	0	389292	127056	55060	S	0	25	0:27.50
3726	root	20	0	389292	127056	55060	S	0	25	0:27.50
3726	root	20	0	389292	127056	55060	S	0	25	0:27.50
3726	root	20	0	390192	128400	56312	R	25	25.2	0:27.65
3726	root	20	0	394692	128400	56312	R	98.3	25.2	0:28.24
3726	root	20	0	394692	128400	56312	R	100	25.2	0:28.85
3726	root	20	0	394692	128400	56312	R	98.3	25.2	0:29.44
3726	root	20	0	394692	128400	56312	R	100	25.2	0:30.04
3726	root	20	0	394692	128400	56312	R	98.3	25.2	0:30.63
3726	root	20	0	394692	128400	56312	R	100	25.2	0:31.23
3726	root	20	0	394692	128400	56312	R	98.4	25.2	0:31.83
3726	root	20	0	394692	128400	56312	R	100	25.2	0:32.43
3726	root	20	0	394692	128400	56312	R	98.3	25.2	0:33.02

PID	USER	PR	NI	VIRT	RES	SHR	S	CPU	MEM	TIME
3726	root	20	0	394692	128400	56312	R	100	25.2	0:33.62
3726	root	20	0	394692	128400	56312	R	100	25.2	0:34.22
3726	root	20	0	394692	128400	56312	R	98.4	25.2	0:34.82
3726	root	20	0	394692	128400	56312	R	98.3	25.2	0:35.41
3726	root	20	0	394692	128400	56312	R	100	25.2	0:36.01
3726	root	20	0	394692	128400	56312	R	100	25.2	0:36.61
3726	root	20	0	394692	128400	56312	R	98.4	25.2	0:37.21
3726	root	20	0	394692	128400	56312	R	100	25.2	0:37.81
3726	root	20	0	394692	128400	56312	R	98.3	25.2	0:38.40
3726	root	20	0	394692	128400	56312	S	71.7	25.2	0:38.83
3726	root	20	0	394692	128400	56312	S	0	25.2	0:38.83
3726	root	20	0	394692	128400	56312	S	0	25.2	0:38.83
3726	root	20	0	394692	128400	56312	S	0	25.2	0:38.83

### 7.6.5 Percobaan 5

PID	USER	PR	NI	VIRT	RES	SHR	S	CPU	MEM	TIME
3809	root	20	0	388172	126520	55092	S	0	24.9	4:34.19
3809	root	20	0	388172	126520	55092	S	0	24.9	4:34.19
3809	root	20	0	388172	126520	55092	S	0	24.9	4:34.19
3809	root	20	0	390872	127776	56252	R	43.3	25.1	4:34.45
3809	root	20	0	393572	127776	56252	R	100	25.1	4:35.05
3809	root	20	0	393572	127776	56252	R	98.3	25.1	4:35.64
3809	root	20	0	393572	127776	56252	R	98.3	25.1	4:36.23
3809	root	20	0	393572	127776	56252	R	100	25.1	4:36.84
3809	root	20	0	393572	127776	56252	R	100	25.1	4:37.44
3809	root	20	0	393572	127776	56252	R	98.3	25.1	4:38.03
3809	root	20	0	393572	127776	56252	R	90	25.1	4:38.57
3809	root	20	0	393572	127776	56252	R	98.4	25.1	4:39.17
3809	root	20	0	393572	127776	56252	R	100	25.1	4:39.77
3809	root	20	0	393572	127776	56252	R	98.3	25.1	4:40.36

PID	USER	PR	NI	VIRT	RES	SHR	S	CPU	MEM	TIME
3809	root	20	0	393572	127776	56252	R	100	25.1	4:40.96
3809	root	20	0	393572	127776	56252	R	98.3	25.1	4:41.55
3809	root	20	0	393572	127776	56252	R	100	25.1	4:42.16
3809	root	20	0	393572	127776	56252	R	98.3	25.1	4:42.75
3809	root	20	0	393572	127776	56252	R	100	25.1	4:43.35
3809	root	20	0	393572	127776	56252	R	100	25.1	4:43.95
3809	root	20	0	393572	127776	56252	R	98.3	25.1	4:44.54
3809	root	20	0	393572	127776	56252	R	100	25.1	4:45.15
3809	root	20	0	393572	127776	56252	R	100	25.1	4:45.75
3809	root	20	0	393572	127776	56252	R	98.3	25.1	4:46.34
3809	root	20	0	393572	127776	56252	R	98.3	25.1	4:46.93
3809	root	20	0	393572	127776	56252	R	100	25.1	4:47.54
3809	root	20	0	393572	127776	56252	S	66.7	25.1	4:47.94
3809	root	20	0	393572	127776	56252	S	0	25.1	4:47.94
3809	root	20	0	393572	127776	56252	S	0	25.1	4:47.94

## 7.7 Data Hasil Percobaan ARM Laplacian

### 7.7.1 Percobaan 1

PID	USER	PR	NI	VIRT	RES	SHR	S	CPU	MEM	TIME
4382	root	20	0	388124	125980	54808	S	0	24.8	0:24.96
4382	root	20	0	388124	125980	54808	S	0	24.8	0:24.96
4382	root	20	0	388124	125980	54808	S	0	24.8	0:24.96
4382	root	20	0	392624	127384	56120	R	77	25	0:25.43
4382	root	20	0	393524	127384	56120	R	98.3	25	0:26.02
4382	root	20	0	393524	127384	56120	R	100	25	0:26.62
4382	root	20	0	393524	127384	56120	R	96.7	25	0:27.20
4382	root	20	0	393524	127384	56120	R	100	25	0:27.81
4382	root	20	0	393524	127384	56120	R	98.3	25	0:28.40
4382	root	20	0	393524	127384	56120	R	100	25	0:29.00

PID	USER	PR	NI	VIRT	RES	SHR	S	CPU	MEM	TIME
4382	root	20	0	393524	127384	56120	R	98.3	25	0:29.59
4382	root	20	0	393524	127384	56120	R	100	25	0:30.19
4382	root	20	0	393524	127384	56120	R	98.4	25	0:30.79
4382	root	20	0	393524	127384	56120	R	100	25	0:31.39
4382	root	20	0	393524	127384	56120	R	96.7	25	0:31.97
4382	root	20	0	393524	127384	56120	R	93.3	25	0:32.53
4382	root	20	0	393524	127384	56120	R	101.7	25	0:33.14
4382	root	20	0	393524	127384	56120	R	96.7	25	0:33.73
4382	root	20	0	393524	127384	56120	R	100	25	0:34.33
4382	root	20	0	393524	127384	56120	R	100	25	0:34.93
4382	root	20	0	393524	127384	56120	R	100	25	0:35.53
4382	root	20	0	393524	127384	56120	S	23	25	0:35.67
4382	root	20	0	393524	127384	56120	S	0	25	0:35.67
4382	root	20	0	393524	127384	56120	S	0	25	0:35.67
4382	root	20	0	393524	127384	56120	S	0	25	0:35.67

### 7.7.2 Percobaan 2

PID	USER	PR	NI	VIRT	RES	SHR	S	CPU	MEM	TIME
4687	root	20	0	388372	126748	55008	S	0	24.9	1:24.81
4687	root	20	0	388372	126748	55008	S	0	24.9	1:24.81
4687	root	20	0	388372	126748	55008	S	0	24.9	1:24.81
4687	root	20	0	388372	126748	55008	S	0	24.9	1:24.81
4687	root	20	0	391972	128152	56320	R	65	25.2	1:25.20
4687	root	20	0	393772	128152	56320	R	100	25.2	1:25.80
4687	root	20	0	393772	128152	56320	R	98.3	25.2	1:26.39
4687	root	20	0	393772	128152	56320	R	100	25.2	1:27.00
4687	root	20	0	393772	128152	56320	R	100	25.2	1:27.60
4687	root	20	0	393772	128152	56320	R	100	25.2	1:28.20
4687	root	20	0	393772	128152	56320	R	100	25.2	1:28.80
4687	root	20	0	393772	128152	56320	R	100	25.2	1:29.41

PID	USER	PR	NI	VIRT	RES	SHR	S	CPU	MEM	TIME
4687	root	20	0	393772	128152	56320	R	100	25.2	1:30.01
4687	root	20	0	393772	128152	56320	R	100	25.2	1:30.61
4687	root	20	0	393772	128152	56320	R	100	25.2	1:31.21
4687	root	20	0	393772	128152	56320	R	100	25.2	1:31.81
4687	root	20	0	393772	128152	56320	R	100	25.2	1:32.42
4687	root	20	0	393772	128152	56320	R	100	25.2	1:33.02
4687	root	20	0	393772	128152	56320	R	100	25.2	1:33.62
4687	root	20	0	393772	128152	56320	R	100	25.2	1:34.22
4687	root	20	0	393772	128152	56320	R	100	25.2	1:34.83
4687	root	20	0	393772	128152	56320	R	100	25.2	1:35.43
4687	root	20	0	393772	128152	56320	S	8.3	25.2	1:35.48
4687	root	20	0	393772	128152	56320	S	0	25.2	1:35.48
4687	root	20	0	393772	128152	56320	S	0	25.2	1:35.48

### 7.7.3 Percobaan 3

PID	USER	PR	NI	VIRT	RES	SHR	S	CPU	MEM	TIME
4764	root	20	0	387640	125868	54940	S	0	24.8	0:36.38
4764	root	20	0	387640	125868	54940	S	0	24.8	0:36.38
4764	root	20	0	387640	125868	54940	S	0	24.8	0:36.38
4764	root	20	0	391240	127228	56208	R	60	25	0:36.74
4764	root	20	0	393040	127228	56208	R	100	25	0:37.35
4764	root	20	0	393040	127228	56208	R	100	25	0:37.95
4764	root	20	0	393040	127228	56208	R	98.3	25	0:38.54
4764	root	20	0	393040	127228	56208	R	98.3	25	0:39.13
4764	root	20	0	393040	127228	56208	R	100	25	0:39.73
4764	root	20	0	393040	127228	56208	R	100	25	0:40.34
4764	root	20	0	393040	127228	56208	R	100	25	0:40.94
4764	root	20	0	393040	127228	56208	R	98.3	25	0:41.53
4764	root	20	0	393040	127228	56208	R	100	25	0:42.13
4764	root	20	0	393040	127228	56208	R	100	25	0:42.74

PID	USER	PR	NI	VIRT	RES	SHR	S	CPU	MEM	TIME
4764	root	20	0	393040	127228	56208	R	100	25	0:43.34
4764	root	20	0	393040	127228	56208	R	100	25	0:43.94
4764	root	20	0	393040	127228	56208	R	100	25	0:44.54
4764	root	20	0	393040	127228	56208	R	98.3	25	0:45.13
4764	root	20	0	393040	127228	56208	R	100	25	0:45.74
4764	root	20	0	393040	127228	56208	R	100	25	0:46.34
4764	root	20	0	393040	127228	56208	R	100	25	0:46.94
4764	root	20	0	393040	127228	56208	S	21.7	25	0:47.07
4764	root	20	0	393040	127228	56208	S	0	25	0:47.07
4764	root	20	0	393040	127228	56208	S	0	25	0:47.07
4764	root	20	0	393040	127228	56208	S	0	25	0:47.07
4764	root	20	0	393040	127228	56208	S	0	25	0:47.07

#### 7.7.4 Percobaan 4

PID	USER	PR	NI	VIRT	RES	SHR	S	CPU	MEM	TIME
4796	root	20	0	388160	126196	54992	S	0	24.8	1:23.26
4796	root	20	0	388160	126196	54992	S	0	24.8	1:23.26
4796	root	20	0	388160	126196	54992	S	0	24.8	1:23.26
4796	root	20	0	388160	126196	54992	S	0	24.8	1:23.26
4796	root	20	0	392660	127644	56348	R	95	25.1	1:23.83
4796	root	20	0	393560	127644	56348	R	98.3	25.1	1:24.42
4796	root	20	0	393560	127644	56348	R	100	25.1	1:25.02
4796	root	20	0	393560	127644	56348	R	100	25.1	1:25.62
4796	root	20	0	393560	127644	56348	R	100	25.1	1:26.23
4796	root	20	0	393560	127644	56348	R	100	25.1	1:26.83
4796	root	20	0	393560	127644	56348	R	100	25.1	1:27.43
4796	root	20	0	393560	127644	56348	R	100	25.1	1:28.03
4796	root	20	0	393560	127644	56348	R	98.3	25.1	1:28.62
4796	root	20	0	393560	127644	56348	R	100	25.1	1:29.23
4796	root	20	0	393560	127644	56348	R	100	25.1	1:29.83

PID	USER	PR	NI	VIRT	RES	SHR	S	CPU	MEM	TIME
4796	root	20	0	393560	127644	56348	R	100	25.1	1:30.43
4796	root	20	0	393560	127644	56348	R	98.3	25.1	1:31.02
4796	root	20	0	393560	127644	56348	R	101.6	25.1	1:31.64
4796	root	20	0	393560	127644	56348	R	98.3	25.1	1:32.23
4796	root	20	0	393560	127644	56348	R	100	25.1	1:32.83
4796	root	20	0	393560	127644	56348	R	100	25.1	1:33.43
4796	root	20	0	393560	127644	56348	S	86.7	25.1	1:33.95
4796	root	20	0	393560	127644	56348	S	0	25.1	1:33.95
4796	root	20	0	393560	127644	56348	S	0	25.1	1:33.95
4796	root	20	0	393560	127644	56348	S	0	25.1	1:33.95
4796	root	20	0	393560	127644	56348	S	0	25.1	1:33.95
4796	root	20	0	393560	127644	56348	S	0	25.1	1:33.95

### 7.7.5 Percobaan 5

PID	USER	PR	NI	VIRT	RES	SHR	S	CPU	MEM	TIME
4848	root	20	0	387644	125812	55088	S	0	24.7	1:27.47
4848	root	20	0	387644	125812	55088	S	0	24.7	1:27.47
4848	root	20	0	387644	125812	55088	S	0	24.7	1:27.47
4848	root	20	0	387644	125812	55088	S	0	24.7	1:27.47
4848	root	20	0	393044	127060	56244	R	98.3	25	1:28.19
4848	root	20	0	393044	127060	56244	R	100	25	1:28.80
4848	root	20	0	393044	127060	56244	R	100	25	1:29.40
4848	root	20	0	393044	127060	56244	R	100	25	1:30.00
4848	root	20	0	393044	127060	56244	R	100	25	1:30.60
4848	root	20	0	393044	127060	56244	R	100	25	1:31.20
4848	root	20	0	393044	127060	56244	R	100	25	1:31.81
4848	root	20	0	393044	127060	56244	R	100	25	1:32.41
4848	root	20	0	393044	127060	56244	R	100	25	1:33.01
4848	root	20	0	393044	127060	56244	R	100	25	1:33.61
4848	root	20	0	393044	127060	56244	R	100	25	1:34.22

PID	USER	PR	NI	VIRT	RES	SHR	S	CPU	MEM	TIME
4848	root	20	0	393044	127060	56244	R	100	25	1:34.82
4848	root	20	0	393044	127060	56244	R	100	25	1:35.42
4848	root	20	0	393044	127060	56244	R	100	25	1:36.02
4848	root	20	0	393044	127060	56244	R	100	25	1:36.62
4848	root	20	0	393044	127060	56244	R	100	25	1:37.23
4848	root	20	0	393044	127060	56244	R	100	25	1:37.83
4848	root	20	0	393044	127060	56244	S	0	25	1:38.11
4848	root	20	0	393044	127060	56244	S	0	25	1:38.11
4848	root	20	0	393044	127060	56244	S	0	25	1:38.11
4848	root	20	0	393044	127060	56244	S	0	25	1:38.11
4848	root	20	0	393044	127060	56244	S	0	25	1:38.11

## 7.8 Data Hasil Percobaan ARM Sharpening

### 7.8.1 Percobaan 1

PID	USER	PR	NI	VIRT	RES	SHR	S	CPU	MEM	TIME
3884	root	20	0	388268	126400	55208	S	0	24.9	0:34.21
3884	root	20	0	388268	126400	55208	S	0	24.9	0:34.21
3884	root	20	0	388268	126400	55208	S	0	24.9	0:34.21
3884	root	20	0	392768	127604	56316	R	96.7	25.1	0:34.87
3884	root	20	0	393668	127604	56316	R	100	25.1	0:35.48
3884	root	20	0	393668	127604	56316	R	100	25.1	0:36.08
3884	root	20	0	393668	127604	56316	R	100	25.1	0:36.68
3884	root	20	0	393668	127604	56316	R	100	25.1	0:37.28
3884	root	20	0	393668	127604	56316	R	100	25.1	0:37.89
3884	root	20	0	393668	127604	56316	R	100	25.1	0:38.49
3884	root	20	0	393668	127604	56316	R	100	25.1	0:39.09
3884	root	20	0	393668	127604	56316	R	100	25.1	0:39.69
3884	root	20	0	393668	127604	56316	R	100	25.1	0:40.29
3884	root	20	0	393668	127604	56316	R	100	25.1	0:40.90

PID	USER	PR	NI	VIRT	RES	SHR	S	CPU	MEM	TIME
3884	root	20	0	393668	127604	56316	R	100	25.1	0:41.50
3884	root	20	0	393668	127604	56316	R	100	25.1	0:42.10
3884	root	20	0	393668	127604	56316	R	100	25.1	0:42.70
3884	root	20	0	393668	127604	56316	R	100	25.1	0:43.30
3884	root	20	0	393668	127604	56316	R	100	25.1	0:43.91
3884	root	20	0	393668	127604	56316	R	100	25.1	0:44.51
3884	root	20	0	393668	127604	56316	R	100	25.1	0:45.11
3884	root	20	0	393668	127604	56316	R	100	25.1	0:45.71
3884	root	20	0	393668	127604	56316	R	100	25.1	0:46.32
3884	root	20	0	393668	127604	56316	R	100	25.1	0:46.92
3884	root	20	0	393668	127604	56316	R	100	25.1	0:47.52
3884	root	20	0	393668	127604	56316	S	88.3	25.1	0:48.05
3884	root	20	0	393668	127604	56316	S	0	25.1	0:48.05
3884	root	20	0	393668	127604	56316	S	0	25.1	0:48.05

### 7.8.2 Percobaan 2

PID	USER	PR	NI	VIRT	RES	SHR	S	CPU	MEM	TIME
4052	root	20	0	394836	132740	54952	S	0	26.1	0:41.85
4052	root	20	0	394836	132740	54952	S	0	26.1	0:41.85
4052	root	20	0	394836	132740	54952	S	0	26.1	0:41.85
4052	root	20	0	400236	134168	56288	R	98.3	26.4	0:42.90
4052	root	20	0	400236	134168	56288	R	100	26.4	0:43.50
4052	root	20	0	400236	134168	56288	R	98.3	26.4	0:44.09
4052	root	20	0	400236	134168	56288	R	100	26.4	0:44.70
4052	root	20	0	400236	134168	56288	R	100	26.4	0:45.30
4052	root	20	0	400236	134168	56288	R	98.3	26.4	0:45.89
4052	root	20	0	400236	134168	56288	R	98.3	26.4	0:46.48
4052	root	20	0	400236	134168	56288	R	100	26.4	0:47.09
4052	root	20	0	400236	134168	56288	R	100	26.4	0:47.69
4052	root	20	0	400236	134168	56288	R	98.3	26.4	0:48.28

PID	USER	PR	NI	VIRT	RES	SHR	S	CPU	MEM	TIME
4052	root	20	0	400236	134168	56288	R	100	26.4	0:48.88
4052	root	20	0	400236	134168	56288	R	100	26.4	0:49.48
4052	root	20	0	400236	134168	56288	R	98.4	26.4	0:50.08
4052	root	20	0	400236	134168	56288	R	100	26.4	0:50.68
4052	root	20	0	400236	134168	56288	R	100	26.4	0:51.28
4052	root	20	0	400236	134168	56288	R	100	26.4	0:51.88
4052	root	20	0	400236	134168	56288	R	98.4	26.4	0:52.48
4052	root	20	0	400236	134168	56288	R	100	26.4	0:53.08
4052	root	20	0	400236	134168	56288	R	100	26.4	0:53.68
4052	root	20	0	400236	134168	56288	R	98.3	26.4	0:54.27
4052	root	20	0	400236	134168	56288	R	100	26.4	0:54.87
4052	root	20	0	400236	134168	56288	R	100	26.4	0:55.48
4052	root	20	0	400236	134168	56288	S	0	26.4	0:55.74

### 7.8.3 Percobaan 3

PID	USER	PR	NI	VIRT	RES	SHR	S	CPU	MEM	TIME
4226	root	20	0	388268	126452	55252	S	0	24.9	0:33.83
4226	root	20	0	388268	126452	55252	S	0	24.9	0:33.83
4226	root	20	0	388268	126452	55252	S	0	24.9	0:33.83
4226	root	20	0	392768	127684	56392	R	95	25.1	0:34.45
4226	root	20	0	393668	127684	56392	R	100	25.1	0:35.05
4226	root	20	0	393668	127684	56392	R	100	25.1	0:35.66
4226	root	20	0	393668	127684	56392	R	100	25.1	0:36.26
4226	root	20	0	393668	127684	56392	R	98.3	25.1	0:36.85
4226	root	20	0	393668	127684	56392	R	100	25.1	0:37.45
4226	root	20	0	393668	127684	56392	R	100	25.1	0:38.05
4226	root	20	0	393668	127684	56392	R	100	25.1	0:38.66
4226	root	20	0	393668	127684	56392	R	100	25.1	0:39.26
4226	root	20	0	393668	127684	56392	R	100	25.1	0:39.86
4226	root	20	0	393668	127684	56392	R	100	25.1	0:40.46

PID	USER	PR	NI	VIRT	RES	SHR	S	CPU	MEM	TIME
4226	root	20	0	393668	127684	56392	R	100	25.1	0:41.07
4226	root	20	0	393668	127684	56392	R	98.3	25.1	0:41.66
4226	root	20	0	393668	127684	56392	R	100	25.1	0:42.26
4226	root	20	0	393668	127684	56392	R	100	25.1	0:42.86
4226	root	20	0	393668	127684	56392	R	100	25.1	0:43.46
4226	root	20	0	393668	127684	56392	R	100	25.1	0:44.07
4226	root	20	0	393668	127684	56392	R	100	25.1	0:44.67
4226	root	20	0	393668	127684	56392	R	100	25.1	0:45.27
4226	root	20	0	393668	127684	56392	R	100	25.1	0:45.87
4226	root	20	0	393668	127684	56392	R	98.3	25.1	0:46.46
4226	root	20	0	393668	127684	56392	R	100	25.1	0:47.07
4226	root	20	0	393668	127684	56392	R	100	25.1	0:47.67
4226	root	20	0	393668	127684	56392	S	0	25.1	0:47.70
4226	root	20	0	393668	127684	56392	S	0	25.1	0:47.70
4226	root	20	0	393668	127684	56392	S	0	25.1	0:47.70

#### 7.8.4 Percobaan 4

PID	USER	PR	NI	VIRT	RES	SHR	S	CPU	MEM	TIME
3941	root	20	0	388124	126232	54928	S	0	24.8	0:23.33
3941	root	20	0	388124	126232	54928	S	0	24.8	0:23.33
3941	root	20	0	388124	126232	54928	S	0	24.8	0:23.33
3941	root	20	0	388124	126232	54928	S	0	24.8	0:23.33
3941	root	20	0	391724	127644	56248	R	95.1	25.1	0:23.91
3941	root	20	0	393524	127608	56212	R	98.3	25.1	0:24.50
3941	root	20	0	393524	127608	56212	R	100	25.1	0:25.10
3941	root	20	0	393524	127608	56212	R	98.3	25.1	0:25.69
3941	root	20	0	393524	127608	56212	R	100	25.1	0:26.30
3941	root	20	0	393524	127608	56212	R	100	25.1	0:26.90
3941	root	20	0	393524	127608	56212	R	98.3	25.1	0:27.49
3941	root	20	0	393524	127608	56212	R	100	25.1	0:28.09

PID	USER	PR	NI	VIRT	RES	SHR	S	CPU	MEM	TIME
3941	root	20	0	393524	127608	56212	R	100	25.1	0:28.69
3941	root	20	0	393524	127608	56212	R	98.4	25.1	0:29.29
3941	root	20	0	393524	127608	56212	R	100	25.1	0:29.89
3941	root	20	0	393524	127608	56212	R	100	25.1	0:30.49
3941	root	20	0	393524	127608	56212	R	98.3	25.1	0:31.08
3941	root	20	0	393524	127608	56212	R	103.3	25.1	0:31.70
3941	root	20	0	393524	127608	56212	R	96.7	25.1	0:32.29
3941	root	20	0	393524	127608	56212	R	100	25.1	0:32.89
3941	root	20	0	393524	127608	56212	R	100	25.1	0:33.49
3941	root	20	0	393524	127608	56212	R	98.3	25.1	0:34.08
3941	root	20	0	393524	127608	56212	R	100	25.1	0:34.69
3941	root	20	0	393524	127608	56212	R	100	25.1	0:35.29
3941	root	20	0	393524	127608	56212	R	98.3	25.1	0:35.88
3941	root	20	0	393524	127608	56212	R	100	25.1	0:36.48
3941	root	20	0	393524	127608	56212	R	98.3	25.1	0:37.07
3941	root	20	0	393524	127608	56212	S	0	25.1	0:37.20
3941	root	20	0	393524	127608	56212	S	0	25.1	0:37.20

### 7.8.5 Percobaan 5

PID	USER	PR	NI	VIRT	RES	SHR	S	CPU	MEM	TIME
4262	root	20	0	389400	127112	54968	S	0	25	2:34.42
4262	root	20	0	389400	127112	54968	S	0	25	2:34.42
4262	root	20	0	389400	127112	54968	S	0	25	2:34.42
4262	root	20	0	394800	128520	56284	R	98.4	25.3	2:35.18
4262	root	20	0	394800	128520	56284	R	100	25.3	2:35.78
4262	root	20	0	394800	128520	56284	R	98.3	25.3	2:36.37
4262	root	20	0	394800	128520	56284	R	100	25.3	2:36.97
4262	root	20	0	394800	128520	56284	R	100	25.3	2:37.57
4262	root	20	0	394800	128520	56284	R	100	25.3	2:38.18
4262	root	20	0	394800	128520	56284	R	98.3	25.3	2:38.77

PID	USER	PR	NI	VIRT	RES	SHR	S	CPU	MEM	TIME
4262	root	20	0	394800	128520	56284	R	100	25.3	2:39.37
4262	root	20	0	394800	128520	56284	R	100	25.3	2:39.97
4262	root	20	0	394800	128520	56284	R	101.7	25.3	2:40.58
4262	root	20	0	394800	128520	56284	R	100	25.3	2:41.19
4262	root	20	0	394800	128520	56284	R	100	25.3	2:41.79
4262	root	20	0	394800	128520	56284	R	98.3	25.3	2:42.38
4262	root	20	0	394800	128520	56284	R	100	25.3	2:42.98
4262	root	20	0	394800	128520	56284	R	100	25.3	2:43.59
4262	root	20	0	394800	128520	56284	R	98.3	25.3	2:44.18
4262	root	20	0	394800	128520	56284	R	100	25.3	2:44.78
4262	root	20	0	394800	128520	56284	R	100	25.3	2:45.38
4262	root	20	0	394800	128520	56284	R	100	25.3	2:45.98
4262	root	20	0	394800	128520	56284	R	100	25.3	2:46.59
4262	root	20	0	394800	128520	56284	R	100	25.3	2:47.19
4262	root	20	0	394800	128520	56284	R	100	25.3	2:47.79
4262	root	20	0	394800	128520	56284	S	80	25.3	2:48.27
4262	root	20	0	394800	128520	56284	S	0	25.3	2:48.27
4262	root	20	0	394800	128520	56284	S	0	25.3	2:48.27
4262	root	20	0	394800	128520	56284	S	0	25.3	2:48.27

## 7.9 Data Hasil Percobaan ARM Sobel Horizontal

### 7.9.1 Percobaan 1

PID	USER	PR	NI	VIRT	RES	SHR	S	CPU	MEM	TIME
4892	root	20	0	393556	127756	56308	S	0	25.1	0:39.52
4892	root	20	0	393556	127756	56308	S	0	25.1	0:39.52
4892	root	20	0	393556	127756	56308	R	98.3	25.1	0:40.50
4892	root	20	0	393556	127756	56308	R	100	25.1	0:41.10
4892	root	20	0	393556	127756	56308	R	100	25.1	0:41.71
4892	root	20	0	393556	127756	56308	R	98.3	25.1	0:42.30

PID	USER	PR	NI	VIRT	RES	SHR	S	CPU	MEM	TIME
4892	root	20	0	393556	127756	56308	R	100	25.1	0:42.90
4892	root	20	0	393556	127756	56308	R	100	25.1	0:43.50
4892	root	20	0	393556	127756	56308	R	98.4	25.1	0:44.10
4892	root	20	0	393556	127756	56308	R	100	25.1	0:44.70
4892	root	20	0	393556	127756	56308	R	100	25.1	0:45.30
4892	root	20	0	393556	127756	56308	R	100	25.1	0:45.90
4892	root	20	0	393556	127756	56308	R	98.3	25.1	0:46.49
4892	root	20	0	393556	127756	56308	R	100	25.1	0:47.10
4892	root	20	0	393556	127756	56308	R	101.7	25.1	0:47.71
4892	root	20	0	393556	127756	56308	R	98.3	25.1	0:48.30
4892	root	20	0	393556	127756	56308	R	100	25.1	0:48.90
4892	root	20	0	393556	127756	56308	R	100	25.1	0:49.51
4892	root	20	0	393556	127756	56308	R	100	25.1	0:50.11
4892	root	20	0	393556	127756	56308	R	98.3	25.1	0:50.70
4892	root	20	0	393556	127756	56308	S	0	25.1	0:50.89
4892	root	20	0	393556	127756	56308	S	0	25.1	0:50.89
4892	root	20	0	393556	127756	56308	S	0	25.1	0:50.89
4892	root	20	0	393556	127756	56308	S	0	25.1	0:50.89
4892	root	20	0	393556	127756	56308	S	0	25.1	0:50.89

### 7.9.2 Percobaan 2

PID	USER	PR	NI	VIRT	RES	SHR	S	CPU	MEM	TIME
5001	root	20	0	387644	125860	55160	S	0	24.7	1:22.23
5001	root	20	0	387644	125860	55160	S	0	24.7	1:22.23
5001	root	20	0	387644	125860	55160	S	0	24.7	1:22.23
5001	root	20	0	393044	127228	56436	R	95.1	25	1:22.89
5001	root	20	0	393044	127228	56436	R	96.7	25	1:23.47
5001	root	20	0	393044	127228	56436	R	101.7	25	1:24.08
5001	root	20	0	393044	127228	56436	R	98.3	25	1:24.67
5001	root	20	0	393044	127228	56436	R	100	25	1:25.27

PID	USER	PR	NI	VIRT	RES	SHR	S	CPU	MEM	TIME
5001	root	20	0	393044	127228	56436	R	98.4	25	1:25.87
5001	root	20	0	393044	127228	56436	R	100	25	1:26.47
5001	root	20	0	393044	127228	56436	R	100	25	1:27.07
5001	root	20	0	393044	127228	56436	R	100	25	1:27.67
5001	root	20	0	393044	127228	56436	R	98.3	25	1:28.26
5001	root	20	0	393044	127228	56436	R	100	25	1:28.87
5001	root	20	0	393044	127228	56436	R	100	25	1:29.47
5001	root	20	0	393044	127228	56436	R	100	25	1:30.07
5001	root	20	0	393044	127228	56436	R	100	25	1:30.67
5001	root	20	0	393044	127228	56436	R	98.4	25	1:31.27
5001	root	20	0	393044	127228	56436	R	100	25	1:31.87
5001	root	20	0	393044	127228	56436	R	100	25	1:32.47
5001	root	20	0	393044	127228	56436	R	100	25	1:33.07
5001	root	20	0	393044	127228	56436	S	95	25	1:33.64
5001	root	20	0	393044	127228	56436	S	0	25	1:33.64
5001	root	20	0	393044	127228	56436	S	0	25	1:33.64
5001	root	20	0	393044	127228	56436	S	0	25	1:33.64

### 7.9.3 Percobaan 3

PID	USER	PR	NI	VIRT	RES	SHR	S	CPU	MEM	TIME
5037	root	20	0	388372	126416	54956	S	0	24.9	1:37.14
5037	root	20	0	388372	126416	54956	S	0	24.9	1:37.14
5037	root	20	0	388372	126416	54956	S	0	24.9	1:37.14
5037	root	20	0	388372	126416	54956	S	0	24.9	1:37.14
5037	root	20	0	392872	127844	56292	R	91.8	25.1	1:37.70
5037	root	20	0	393772	127844	56292	R	100	25.1	1:38.30
5037	root	20	0	393772	127844	56292	R	100	25.1	1:38.90
5037	root	20	0	393772	127844	56292	R	100	25.1	1:39.50
5037	root	20	0	393772	127844	56292	R	100	25.1	1:40.10
5037	root	20	0	393772	127844	56292	R	98.4	25.1	1:40.70

PID	USER	PR	NI	VIRT	RES	SHR	S	CPU	MEM	TIME
5037	root	20	0	393772	127844	56292	R	98.3	25.1	1:41.29
5037	root	20	0	393772	127844	56292	R	100	25.1	1:41.89
5037	root	20	0	393772	127844	56292	R	100	25.1	1:42.49
5037	root	20	0	393772	127844	56292	R	100	25.1	1:43.10
5037	root	20	0	393772	127844	56292	R	98.3	25.1	1:43.69
5037	root	20	0	393772	127844	56292	R	100	25.1	1:44.29
5037	root	20	0	393772	127844	56292	R	100	25.1	1:44.89
5037	root	20	0	393772	127844	56292	R	98.3	25.1	1:45.48
5037	root	20	0	393772	127844	56292	R	100	25.1	1:46.09
5037	root	20	0	393772	127844	56292	R	100	25.1	1:46.69
5037	root	20	0	393772	127844	56292	R	100	25.1	1:47.29
5037	root	20	0	393772	127844	56292	R	100	25.1	1:47.89
5037	root	20	0	393772	127844	56292	R	100	25.1	1:48.50
5037	root	20	0	393772	127844	56292	S	0	25.1	1:48.57
5037	root	20	0	393772	127844	56292	S	0	25.1	1:48.57
5037	root	20	0	393772	127844	56292	S	0	25.1	1:48.57

#### 7.9.4 Percobaan 4

PID	USER	PR	NI	VIRT	RES	SHR	S	CPU	MEM	TIME
5114	root	20	0	388308	127024	55200	S	0	25	1:30.19
5114	root	20	0	388308	127024	55200	S	0	25	1:30.19
5114	root	20	0	388308	127024	55200	S	0	25	1:30.19
5114	root	20	0	388308	127024	55200	S	0	25	1:30.19
5114	root	20	0	393708	128348	56428	R	98.3	25.2	1:31.06
5114	root	20	0	393708	128348	56428	R	100	25.2	1:31.67
5114	root	20	0	393708	128348	56428	R	100	25.2	1:32.27
5114	root	20	0	393708	128348	56428	R	93.3	25.2	1:32.83
5114	root	20	0	393708	128348	56428	R	100	25.2	1:33.43
5114	root	20	0	393708	128348	56428	R	98.3	25.2	1:34.02
5114	root	20	0	393708	128348	56428	R	100	25.2	1:34.63

PID	USER	PR	NI	VIRT	RES	SHR	S	CPU	MEM	TIME
5114	root	20	0	393708	128348	56428	R	100	25.2	1:35.23
5114	root	20	0	393708	128348	56428	R	100	25.2	1:35.83
5114	root	20	0	393708	128348	56428	R	100	25.2	1:36.43
5114	root	20	0	393708	128348	56428	R	100	25.2	1:37.03
5114	root	20	0	393708	128348	56428	R	98.4	25.2	1:37.63
5114	root	20	0	393708	128348	56428	R	101.7	25.2	1:38.24
5114	root	20	0	393708	128348	56428	R	98.3	25.2	1:38.83
5114	root	20	0	393708	128348	56428	R	100	25.2	1:39.43
5114	root	20	0	393708	128348	56428	R	100	25.2	1:40.04
5114	root	20	0	393708	128348	56428	R	100	25.2	1:40.64
5114	root	20	0	393708	128348	56428	R	100	25.2	1:41.24
5114	root	20	0	393708	128348	56428	S	0	25.2	1:41.61
5114	root	20	0	393708	128348	56428	S	0	25.2	1:41.61
5114	root	20	0	393708	128348	56428	S	0	25.2	1:41.61

### 7.9.5 Percobaan 5

PID	USER	PR	NI	VIRT	RES	SHR	S	CPU	MEM	TIME
5229	root	20	0	388244	126724	55124	S	0	24.9	0:24.91
5229	root	20	0	388244	126724	55124	S	0	24.9	0:24.91
5229	root	20	0	388244	126724	55124	S	0	24.9	0:24.91
5229	root	20	0	393644	127976	56284	R	98.3	25.2	0:25.80
5229	root	20	0	393644	127976	56284	R	100	25.2	0:26.40
5229	root	20	0	393644	127976	56284	R	98.3	25.2	0:26.99
5229	root	20	0	393644	127976	56284	R	100	25.2	0:27.60
5229	root	20	0	393644	127976	56284	R	100	25.2	0:28.20
5229	root	20	0	393644	127976	56284	R	98.3	25.2	0:28.79
5229	root	20	0	393644	127976	56284	R	100	25.2	0:29.39
5229	root	20	0	393644	127976	56284	R	100	25.2	0:29.99
5229	root	20	0	393644	127976	56284	R	100	25.2	0:30.60
5229	root	20	0	393644	127976	56284	R	98.3	25.2	0:31.19

PID	USER	PR	NI	VIRT	RES	SHR	S	CPU	MEM	TIME
5229	root	20	0	393644	127976	56284	R	100	25.2	0:31.79
5229	root	20	0	393644	127976	56284	R	98.3	25.2	0:32.38
5229	root	20	0	393644	127976	56284	R	100	25.2	0:32.99
5229	root	20	0	393644	127976	56284	R	100	25.2	0:33.59
5229	root	20	0	393644	127976	56284	R	100	25.2	0:34.19
5229	root	20	0	393644	127976	56284	R	98.3	25.2	0:34.78
5229	root	20	0	393644	127976	56284	R	100	25.2	0:35.38
5229	root	20	0	393644	127976	56284	R	100	25.2	0:35.99
5229	root	20	0	393644	127976	56284	S	0	25.2	0:36.33
5229	root	20	0	393644	127976	56284	S	0	25.2	0:36.33

## 7.10 Data Hasil Percobaan ARM Sobel Vertical

### 7.10.1 Percobaan 1

PID	USER	PR	NI	VIRT	RES	SHR	S	CPU	MEM	TIME
3160	root	20	0	389932	128000	55724	S	0	25.2	0:31.48
3160	root	20	0	395332	128844	56508	R	33.9	25.3	0:32.50
3160	root	20	0	395332	128844	56508	R	99.7	25.3	0:35.49
3160	root	20	0	395332	128844	56508	R	100	25.3	0:38.50
3160	root	20	0	395332	128844	56508	R	100	25.3	0:41.50
3160	root	20	0	395332	128844	56508	R	100	25.3	0:44.50
3160	root	20	0	395332	128844	56508	S	18.9	25.3	0:45.07

### 7.10.2 Percobaan 2

PID	USER	PR	NI	VIRT	RES	SHR	S	CPU	MEM	TIME
3302	root	20	0	395576	129188	56444	R	100	25.4	1:31.32
3302	root	20	0	395576	129188	56444	R	100	25.4	1:34.33
3302	root	20	0	395576	129188	56444	R	99.7	25.4	1:37.32
3302	root	20	0	395576	129188	56444	R	100	25.4	1:40.33

PID	USER	PR	NI	VIRT	RES	SHR	S	CPU	MEM	TIME
3302	root	20	0	395576	129188	56444	S	86.3	25.4	1:42.92

### 7.10.3 Percobaan 3

PID	USER	PR	NI	VIRT	RES	SHR	S	CPU	MEM	TIME
3365	root	20	0	390052	127956	55848	S	0	25.2	2:29.25
3365	root	20	0	395452	128632	56472	R	43	25.3	2:30.54
3365	root	20	0	395452	128632	56472	R	100	25.3	2:33.55
3365	root	20	0	395452	128632	56472	R	100	25.3	2:36.55
3365	root	20	0	395452	128632	56472	R	100	25.3	2:39.56
3365	root	20	0	395452	128632	56472	R	100	25.3	2:42.56

### 7.10.4 Percobaan 4

PID	USER	PR	NI	VIRT	RES	SHR	S	CPU	MEM	TIME
4551	root	20	0	388788	126840	55264	S	0	24.9	0:30.02
4551	root	20	0	388788	126840	55264	S	0	24.9	0:30.02
4551	root	20	0	388788	126840	55264	S	0	24.9	0:30.02
4551	root	20	0	388788	126840	55264	S	0	24.9	0:30.02
4551	root	20	0	394188	128096	56428	R	100	25.2	0:31.09
4551	root	20	0	394188	128096	56428	R	98.3	25.2	0:31.68
4551	root	20	0	394188	128096	56428	R	100	25.2	0:32.28
4551	root	20	0	394188	128096	56428	R	100	25.2	0:32.89
4551	root	20	0	394188	128096	56428	R	100	25.2	0:33.49
4551	root	20	0	394188	128096	56428	R	98.3	25.2	0:34.08
4551	root	20	0	394188	128096	56428	R	100	25.2	0:34.68
4551	root	20	0	394188	128096	56428	R	98.3	25.2	0:35.27
4551	root	20	0	394188	128096	56428	R	100	25.2	0:35.88
4551	root	20	0	394188	128096	56428	R	100	25.2	0:36.48
4551	root	20	0	394188	128096	56428	R	98.3	25.2	0:37.07
4551	root	20	0	394188	128096	56428	R	100	25.2	0:37.67

PID	USER	PR	NI	VIRT	RES	SHR	S	CPU	MEM	TIME
4551	root	20	0	394188	128096	56428	R	100	25.2	0:38.27
4551	root	20	0	394188	128096	56428	R	98.4	25.2	0:38.87
4551	root	20	0	394188	128096	56428	R	100	25.2	0:39.47
4551	root	20	0	394188	128096	56428	R	100	25.2	0:40.07
4551	root	20	0	394188	128096	56428	R	98.3	25.2	0:40.66
4551	root	20	0	394188	128096	56428	R	100	25.2	0:41.27
4551	root	20	0	394188	128096	56428	S	0	25.2	0:41.55
4551	root	20	0	394188	128096	56428	S	0	25.2	0:41.55

### 7.10.5 Percobaan 5

PID	USER	PR	NI	VIRT	RES	SHR	S	CPU	MEM	TIME
4349	root	20	0	389296	126920	54848	S	0	25	0:23.35
4349	root	20	0	389296	126920	54848	S	0	25	0:23.35
4349	root	20	0	389296	126920	54848	S	0	25	0:23.35
4349	root	20	0	389296	126920	54848	S	0	25	0:23.35
4349	root	20	0	394696	128296	56128	R	98.3	25.2	0:24.38
4349	root	20	0	394696	128296	56128	R	100	25.2	0:24.98
4349	root	20	0	394696	128296	56128	R	98.4	25.2	0:25.58
4349	root	20	0	394696	128296	56128	R	100	25.2	0:26.18
4349	root	20	0	394696	128296	56128	R	100	25.2	0:26.78
4349	root	20	0	394696	128296	56128	R	93.3	25.2	0:27.34
4349	root	20	0	394696	128296	56128	R	100	25.2	0:27.94
4349	root	20	0	394696	128296	56128	R	98.4	25.2	0:28.54
4349	root	20	0	394696	128296	56128	R	100	25.2	0:29.14
4349	root	20	0	394696	128296	56128	R	100	25.2	0:29.74
4349	root	20	0	394696	128296	56128	R	100	25.2	0:30.34
4349	root	20	0	394696	128296	56128	R	98.4	25.2	0:30.94
4349	root	20	0	394696	128296	56128	R	100	25.2	0:31.54
4349	root	20	0	394696	128296	56128	R	98.3	25.2	0:32.13
4349	root	20	0	394696	128296	56128	R	91.7	25.2	0:32.68

PID	USER	PR	NI	VIRT	RES	SHR	S	CPU	MEM	TIME
4349	root	20	0	394696	128296	56128	R	100	25.2	0:33.28
4349	root	20	0	394696	128296	56128	R	100	25.2	0:33.89
4349	root	20	0	394696	128296	56128	R	98.3	25.2	0:34.48
4349	root	20	0	394696	128296	56128	S	0	25.2	0:34.93
4349	root	20	0	394696	128296	56128	S	0	25.2	0:34.93