

*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**

**2020**

# **DAFTAR ISI**

<b>DAFTAR ISI . . . . .</b>	<b>iii</b>
<b>DAFTAR TABEL . . . . .</b>	<b>iv</b>
<b>DAFTAR GAMBAR . . . . .</b>	<b>vi</b>
<b>BAB I PENDAHULUAN . . . . .</b>	<b>1</b>
1.1 Latar Belakang . . . . .	1
1.2 Rumusan Masalah . . . . .	3
1.3 Batasan Masalah . . . . .	3
1.4 Tujuan Penelitian . . . . .	4
1.5 Manfaat Penelitian . . . . .	4
<b>BAB II TINJAUAN PUSTAKA . . . . .</b>	<b>5</b>
2.1 Landasan Teori . . . . .	5
2.1.1 Citra Digital . . . . .	5
2.1.2 Pengolahan Citra Digital . . . . .	6
2.1.3 Filter Spasial . . . . .	7
2.1.4 Kernel . . . . .	8
2.1.5 Konvolusi . . . . .	10
2.1.6 Video Streaming . . . . .	11
2.1.7 FPGA . . . . .	12
2.1.8 Evaluasi Performa . . . . .	13
2.2 Penelitian Terkait . . . . .	14
2.2.1 Spatial Filtering Based Boundary Extraction in Underwater Images for Pipeline Detection: FPGA Implementation . . . . .	14
2.2.2 FPGA Implementation of Spatial Filtering techniques for 2D Images . . . . .	14
2.2.3 Features of Image Spatial Filters Implementation on FPGA . . . . .	15

2.2.4	An FPGA-Oriented Algorithm for Real-Time Filtering of Poisson Noise in Video Streams, with Application to X-Ray Fluoroscopy . . . . .	15
2.2.5	A real-time video denoising algorithm with FPGA implementation for Poisson-Gaussian noise . . . . .	16
<b>BAB III METODE PENENILITIAN</b>	. . . . .	<b>17</b>
3.1	Tahapan Penelitian . . . . .	17
3.2	Waktu dan Lokasi Penelitian . . . . .	18
3.3	Rancangan Sistem . . . . .	18
3.4	Instrumen Penelitian . . . . .	19
<b>BAB IV HASIL DAN PEMBAHASAN</b>	. . . . .	<b>20</b>
4.1	Implementasi pada FPGA Development Board . . . . .	20
4.1.1	Penerapan Filter Spasial . . . . .	20
4.1.2	Penerapan Filter Spasial menggunakan Prosesor ARM dan FPGA . . . . .	24
4.1.3	Proses Evaluasi Kinerja . . . . .	25
4.2	Analisis Kinerja . . . . .	28
4.2.1	Waktu Komuptasi . . . . .	28
4.2.2	Frame Rate (FPS) . . . . .	28
4.2.3	Penggunaan CPU . . . . .	29
4.2.4	Penggunaan Memory . . . . .	30
4.2.5	Resident Memory (RES) . . . . .	32
4.2.6	Shared Memory (SHR) . . . . .	32
4.2.7	Virtual Memory (VIRT) . . . . .	33
<b>BAB V KESIMPULAN DAN SARAN</b>	. . . . .	<b>38</b>
5.1	Kesimpulan . . . . .	38
5.2	Saran . . . . .	38
<b>DAFTAR PUSTAKA</b>	. . . . .	<b>38</b>
<b>LAMPIRAN</b>	. . . . .	<b>40</b>

7.3	Data Hasil Percobaan ARM Average Blur . . . . .	41
7.3.1	Percobaan 1 . . . . .	41
7.3.2	Percobaan 2 . . . . .	43
7.3.3	Percobaan 3 . . . . .	46
7.3.4	Percobaan 4 . . . . .	47
7.3.5	Percobaan 5 . . . . .	47

## **DAFTAR TABEL**

4.1	Tabel perbandingan waktu komputasi dengan menggunakan 50 frame.	29
4.2	Tabel perbandingan FPS dengan menggunakan prosesor ARM dan FPGA. . . . .	30
4.3	Tabel perbandingan penggunaan CPU dengan menggunakan prosesor ARM dan FPGA. . . . .	31
4.4	Tabel perbandingan penggunaan memory dengan menggunakan prosesor ARM dan FPGA. . . . .	33
4.5	Tabel perbandingan penggunaan resident memory dengan menggunakan prosesor ARM dan FPGA. . . . .	33
4.6	Tabel perbandingan penggunaan shared memory dengan menggunakan prosesor ARM dan FPGA. . . . .	35
4.7	Tabel perbandingan penggunaan virtual memory dengan menggunakan prosesor ARM dan FPGA. . . . .	36

## DAFTAR GAMBAR

2.1 (a) Contoh citra biner, (b) contoh citra grayscale, (c) contoh citra warna.	6
2.2 Ilustrasi konvolusi pada citra. Sumber: <a href="https://indoml.com">https://indoml.com</a>	11
2.3 Struktur FPGA.	12
2.4 FPGA Board Xilinx PYNQ-Z2.	13
3.1 Flowchart tahapan penelitian.	17
3.2 Rancangan sistem.	18
4.1 Contoh Frame Grayscale.	21
4.2 Hasil filter Average Blur.	21
4.3 Hasil filter Gaussian Blur.	22
4.4 Hasil filter Laplacian.	23
4.5 Hasil filter Sharpening.	23
4.6 Hasil filter Sobel Horizontal.	24
4.7 Hasil filter Sobel Vertical.	24
4.8 Menghitung waktu komputasi dengan library time di Python.	25
4.9 Tampilan program <b>top</b> .	26
4.10 Menampilkan PID sebuah proses dengan bahasa pemrograman Python.	27
4.11 Menjalankan program <b>top</b> kemudian menyimpan hasilnya pada file arm-laplacian1.txt.	27
4.12 Potongan isi file arm-laplacian1.txt.	28
4.13 Isi file arm-laplacian1.csv.	29
4.14 Grafik perbandingan waktu komputasi dengan menggunakan 50 frame.	30
4.15 Grafik perbandingan FPS dengan menggunakan 50 frame.	31
4.16 Grafik perbandingan penggunaan CPU dengan menggunakan prosesor ARM dan FPGA.	32
4.17 Grafik perbandingan penggunaan memory dengan menggunakan prosesor ARM dan FPGA.	34
4.18 Grafik perbandingan penggunaan resident memory dengan menggunakan prosesor ARM dan FPGA.	35

4.19 Grafik perbandingan penggunaan shared memory dengan menggunakan prosesor ARM dan FPGA. . . . .	36
4.20 Grafik perbandingan penggunaan virtual memory dengan menggunakan prosesor ARM dan FPGA. . . . .	37

# 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). Setiap piksel mewakili sebuah warna, pada citra biner sebuah piksel hanya berwarna hitam atau putih saja. Pada citra grayscale warna sebuah piksel mewakili tingkat keabuannya. Sedangkan pada citra warna (RGB) setiap piksel mewakili warna yang merupakan kombinasi dari tiga warna dasar ,yaitu merah, hijau dan biru. Pada umumnya warna dasar dalam citra RGB menggunakan penyimpanan 8 bit untuk menyimpan data warna atau tingkat keabuannya, yang berarti setiap warna mempunyai gradasi sebanyak 255 warna . Dewasa ini, citra digital dapat menggunakan 16 bit untuk menyimpan data warna atau tingkat keabuannya, 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.

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 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.

## **1.4 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.

## **1.5 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:

###### **a. Citra Biner (monokrom)**

Citra biner 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 warna ini. Contoh citra biner dapat dilihat pada gambar 2.1(a).

###### **b. Citra Grayscale**

Banyaknya warna tergantung pada jumlah bit yang disediakan di memori untuk menampung kebutuhan warna ini. Citra *grayscale* 2 bit memiliki 4 gradasi warna, citra *grayscale* 3 bit memiliki 8 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).

### c. Citra Warna

Setiap piksel pada citra warna mewakili warna yang merupakan kombinasi dari tiga warna dasar (RGB = Red Green Blue). 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).

#### a. **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*.

#### b. **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.

#### c. **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.

#### d. **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 et al., 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 sebagai berikut:

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

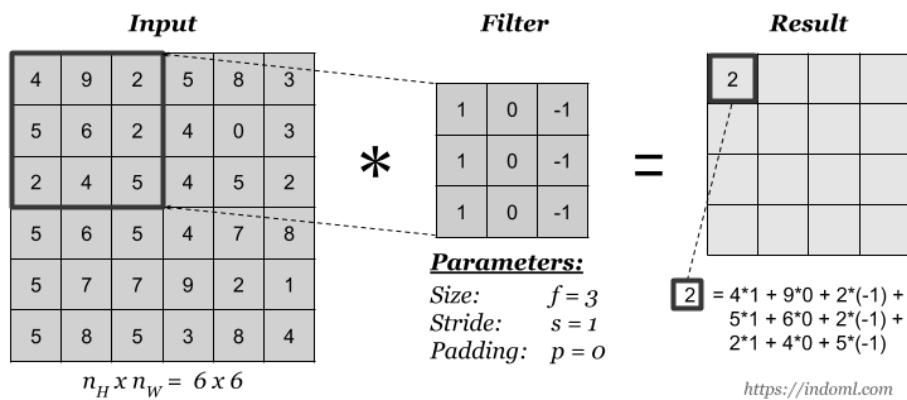
dimana tanda  $*$  menyatakan operator konvolusi, dan peubah  $a$  adalah peubah bantu. Untuk fungsi diskrit, konvolusi didefinisikan sebagai berikut:

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

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  $6 \times 6$  dengan filter berukuran  $3 \times 3$ . 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 (Sutoyo et al., 2009).

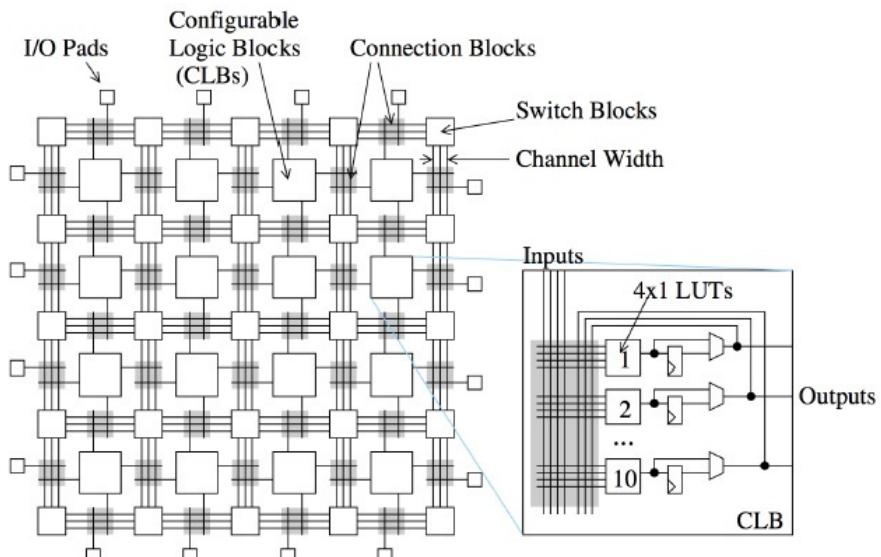
### 2.1.6 Video Streaming

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 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 *Develelment 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)



Gambar 2.4: FPGA Board Xilinx PYNQ-Z2.

### 2.1.8 Evaluasi Performa

#### 2.1.8.1 Waktu Komputasi

(Chang and Culurciello, 2017)

### **2.1.8.2 Frame Rate (FPS)**

Frame rate (Madhusudana et al., 2020)

$$fps = \frac{jumlah\ frame}{waktukomputasi} \quad (2.9)$$

### **2.1.8.3 Penggunaan CPU**

(Linux, )

### **2.1.8.4 Penggunaan Memory**

#### **2.1.8.5 Resident Memory (RES)**

#### **2.1.8.6 Shared Memory (SHR)**

#### **2.1.8.7 Virtual Memory (VIRT)**

## **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 pengangkutan 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 et al., 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 et al., 2019).

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

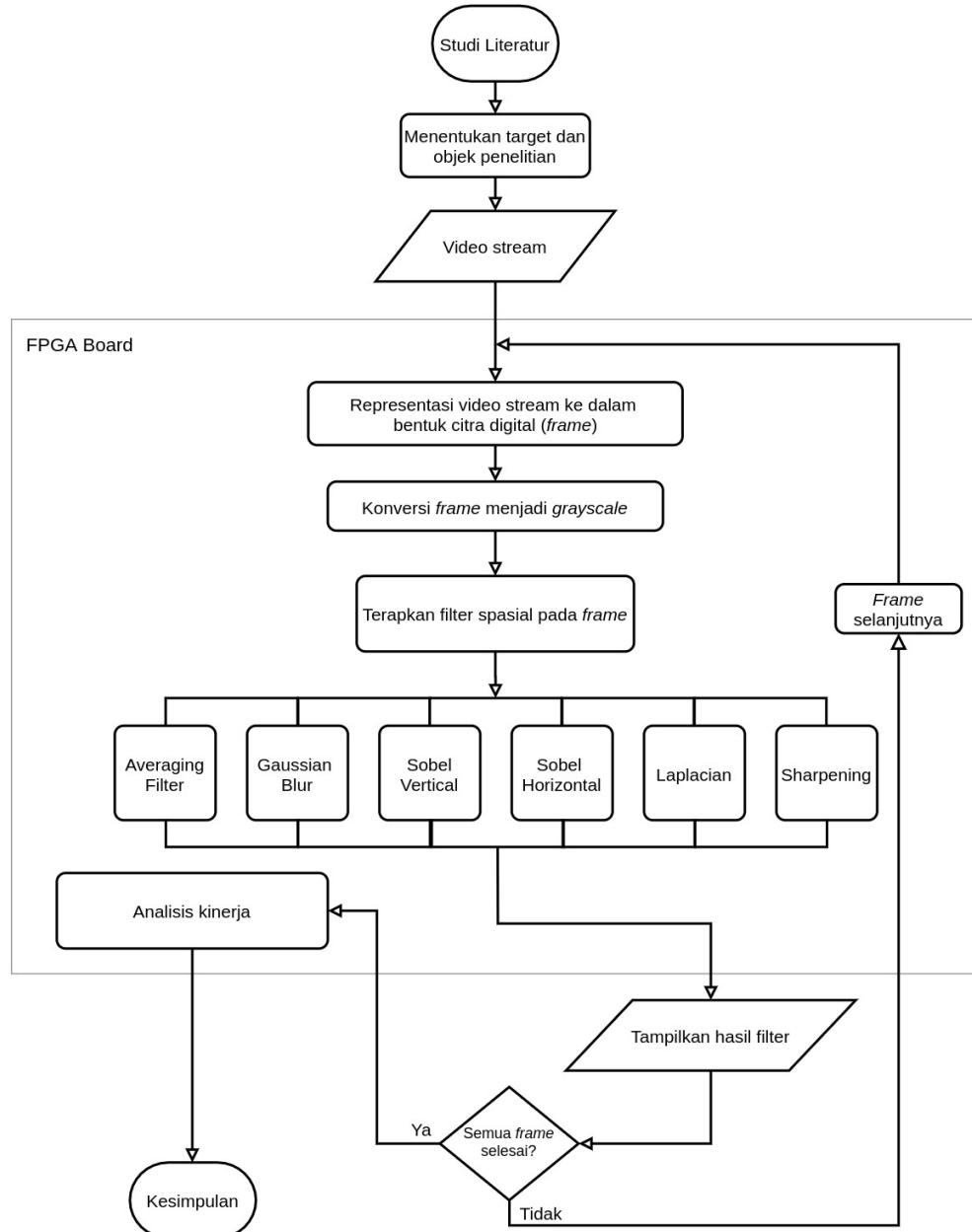
Pada penggunaan umum metode denoising yaitu *Pixel Similarity Weighted Frame Average* (PSWFA). Pada penelitian ini, dilakukan peningkatan kemampuan denoising dari PSWFA menggunakan pre-filter yang mengandung operator *downsampling* dan small Gaussian filter. Transformasi citra dapat mengalami gangguan 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 et al., 2014).

## BAB III

### METODE PENENILITIAN

#### 3.1 Tahapan Penelitian

Tahapan dalam penelitian ini dapat dilihat pada gambar 3.1.



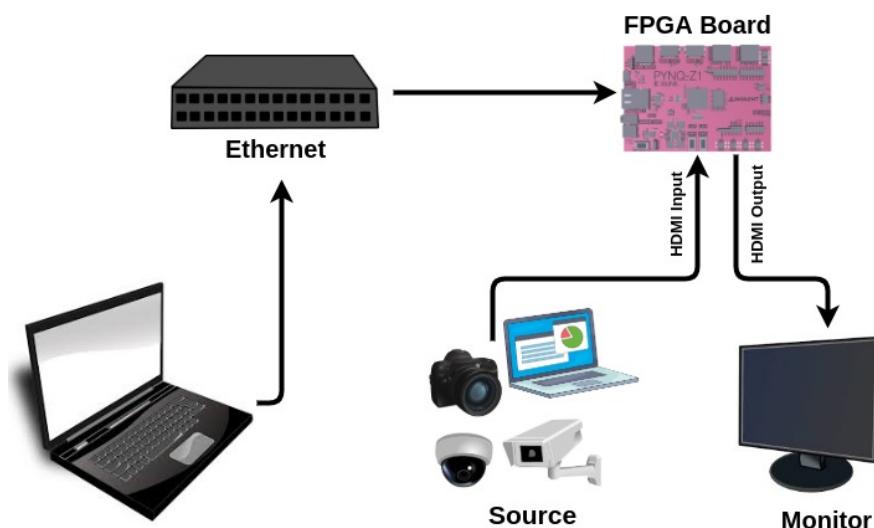
Gambar 3.1: Flowchart tahapan penelitian.

### 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, 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 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 Xilinx PYNQ-Z2.
  - 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:

- 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 tersebut dilakukan pada FPGA Development Board yang menggunakan prosesor ARM dan pada FPGA Development Board yang sama menggunakan FPGA. Kemudian dilakukan analisis kinerja dari keduanya untuk menunjukkan masing-masing waktu komputasi, FPS, penggunaan CPU, penggunaan *memory*, *resident memory* (RES), *shared memory* (SHR), dan *virtual memory* (VIRT).

FPGA Development Board dirangkai seperti yang telah disebutkan di Bab sebelumnya, dapat dilihat 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 di FPGA Development Board.

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

Setiap frame dari *source* video *stream* dibaca sebagai matriks berukuran  $1280 \times 720$  dengan rentang nilai keabuan pada masing-masing piksel yaitu dari 0 sampai 255. Setiap piksel pada matriks tersebut hanya memiliki satu lapis warna karena 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  $1280 \times 720$ . 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.



*Gambar 4.2: Hasil filter Average Blur.*

#### 4.1.1.2 Gaussian Blur

Penerapan filter spasial dengan kernel *gaussian blur* (2.3) yang berukuran  $3 \times 3$  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 perbedaan dengan nilai masing-masing pixel pada gambar 4.2. Hal ini disebabkan oleh nilai bobot pada kernel *gaussian blur* (2.3) yang berbeda dengan kernel *average blur* sehingga hasil konvolusinya juga berbeda.



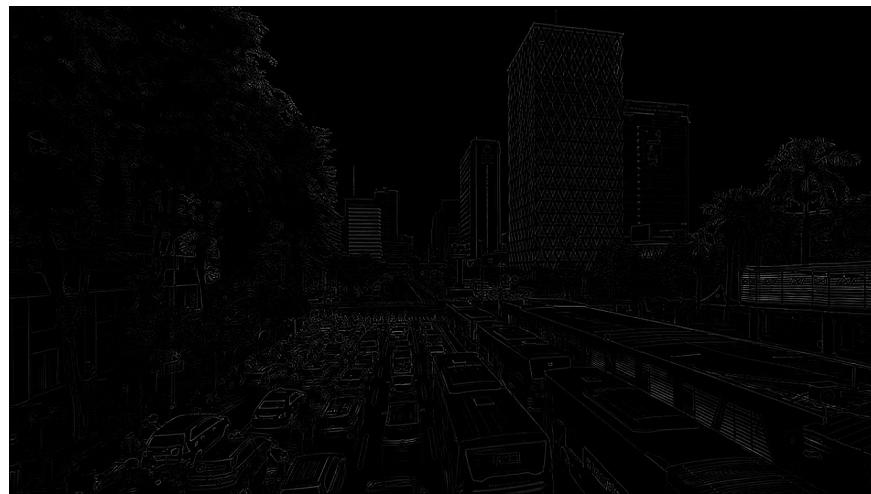
Gambar 4.3: Hasil filter Gaussian Blur.

#### 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 apabila nilai 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 4.4: Hasil filter Laplacian.



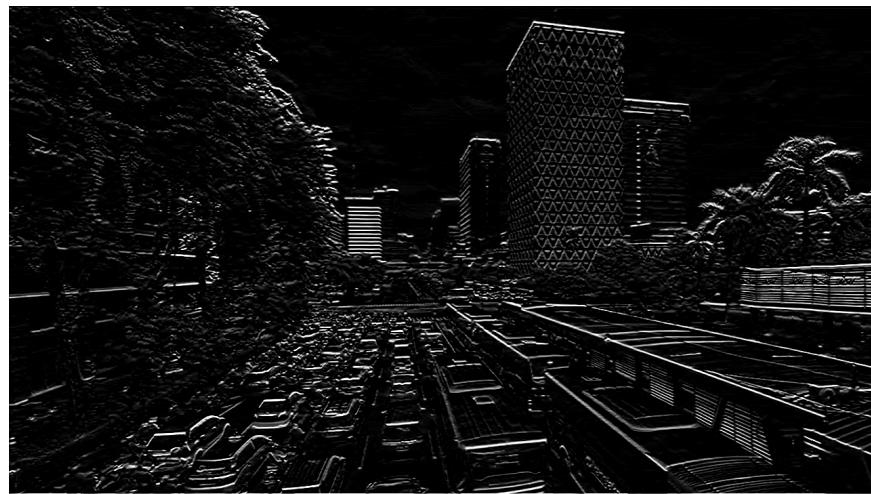
Gambar 4.5: Hasil filter Sharpening.

#### 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 di citra yang banyak mengandung garis horizontal.

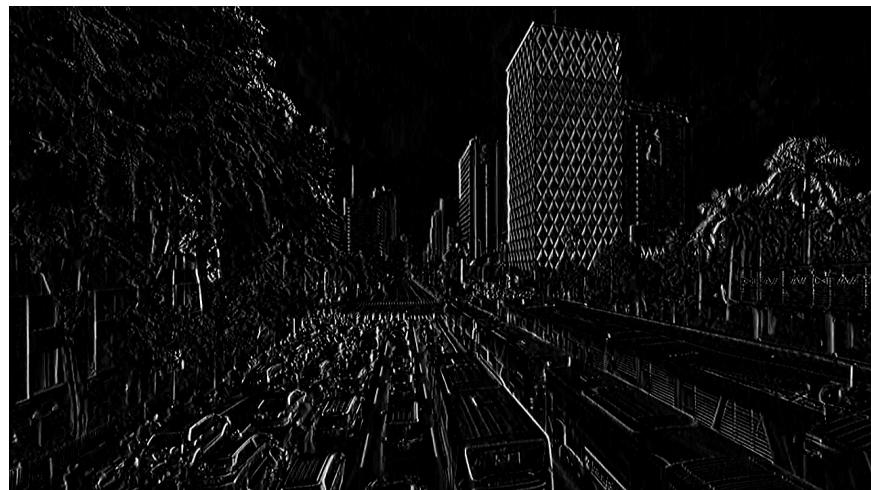
#### 4.1.1.6 Sobel Vertical

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



Gambar 4.6: Hasil filter Sobel Horizontal.

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* dan *pynq-filter-spasial-fpga*.

#### 4.1.3 Proses Evaluasi Kinerja

Pada penelitian ini peneliti melakukan dua proses dalam evaluasi kinerja. Proses pertama untuk menghitung waktu komputasi dan FPS dari masing-masing kernel dengan prosesor ARM dan FPGA. Kemudian proses kedua untuk mencatat 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 untuk masing-masing kernel pada prosesor ARM dan FPGA, dan 5 kali percobaan menggunakan 200 frame untuk masing-masing kernel. Hasil masing-masing percobaan ini dapat dilihat pada lampiran.

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. Dilakukan dengan 50 frame dan 200 frame pada masing-masing kernel menggunakan prosesor ARM dan FPGA. Proses ini dilakukan dengan menggunakan *library time* pada bahasa pemrograman Python, dapat dilihat pada gambar 4.8.

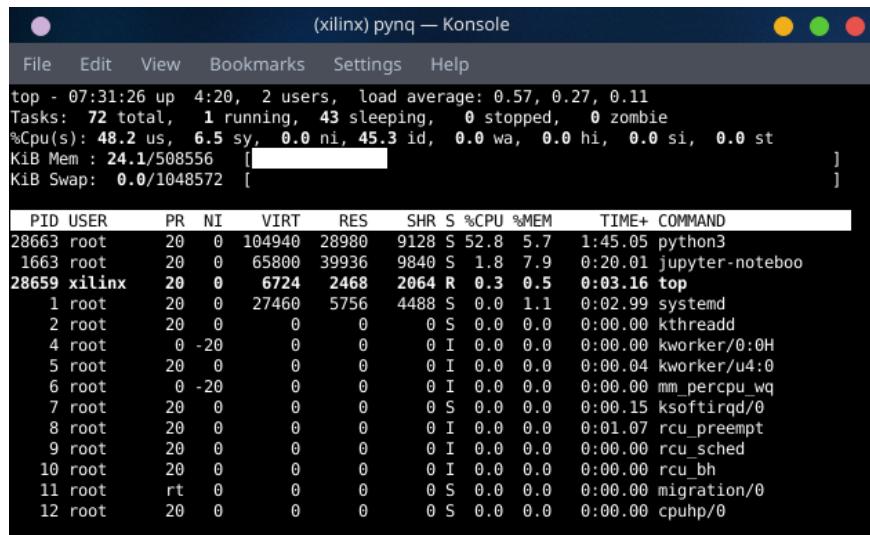
```
import time
...
startSW=time.time()
# implementasi
stopSW=time.time()
result = stopSW - startSW
```

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

Hasil waktu komputasi ini dicatat dan kemudian digunakan untuk menghitung FPS dari masing-masing percobaan. FPS masing-masing percobaan ini dihitung dengan menggunakan persamaan 2.9.

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

Pada proses ini peneliti menggunakan program bawaan dari 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.



```
(xilinx) pynq — Konsole
File Edit View Bookmarks Settings Help
top - 07:31:26 up 4:20, 2 users, load average: 0.57, 0.27, 0.11
Tasks: 72 total, 1 running, 43 sleeping, 0 stopped, 0 zombie
%Cpu(s): 48.2 us, 6.5 sy, 0.0 ni, 45.3 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
KiB Mem : 24.1/508556 [██████] ] KiB Swap: 0.0/1048572 [ ]
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-noteboo
28659 xilinx 20 0 6724 2468 2064 R 0.3 0.5 0:03.16 top
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_premempt
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 implementasi ini menggunakan Jupyter Notebook yang berjalan di 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 percobaan 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 pada saat 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 di 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

...
...

```

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

## 4.2 Analisis Kinerja

### 4.2.1 Waktu Komputasi

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

### 4.2.2 Frame Rate (FPS)

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi

```

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.*

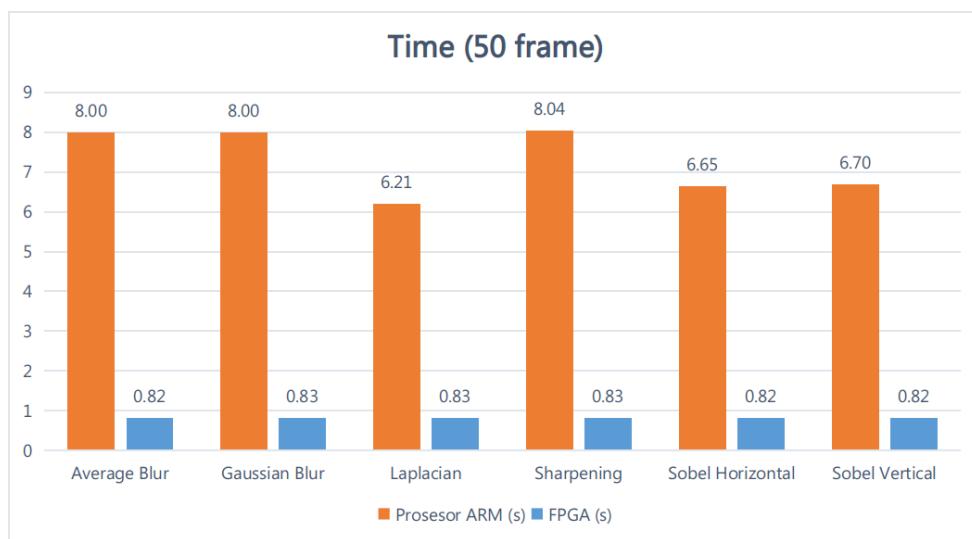
*Tabel 4.1: Tabel perbandingan waktu komputasi dengan menggunakan 50 frame.*

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

nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

#### 4.2.3 Penggunaan CPU

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies



*Gambar 4.14: Grafik perbandingan waktu komputasi dengan menggunakan 50 frame.*

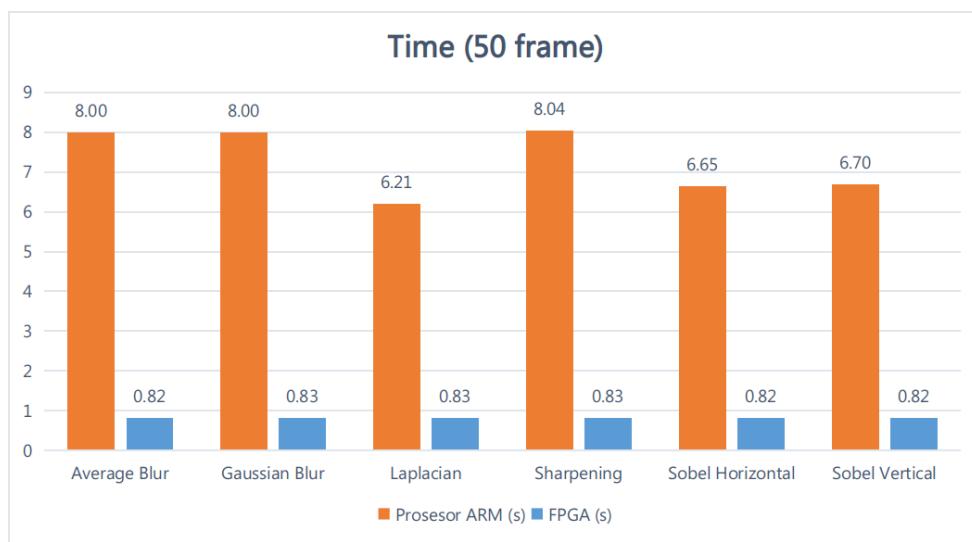
*Tabel 4.2: Tabel perbandingan FPS dengan menggunakan prosesor ARM dan FPGA.*

Filter	Tanpa FPGA	Dengan FPGA
Average Blur	6.24958353344702	60.4268459267488
Gaussian Blur	6.25253493030523	60.2887439620844
Laplacian	8.04783913050973	60.3130625296822
Sharpening	6.20978298450373	60.3363303411042
Sobel Horizontal	7.53538057993614	60.4263337664952
Sobel Vertical	7.45901961777084	60.4404744910869

vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

#### 4.2.4 Penggunaan Memory

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue,

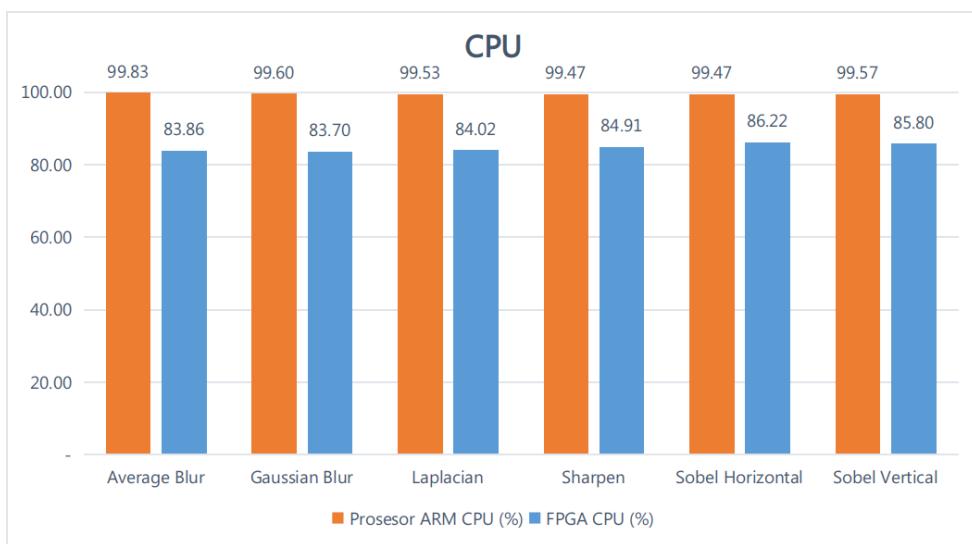


*Gambar 4.15: Grafik perbandingan FPS dengan menggunakan 50 frame.*

*Tabel 4.3: 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
Sharpen	99.47	84.91
Sobel Horizontal	99.47	86.22
Sobel Vertical	99.57	85.80
Rata-rata	99.58	84.75

a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.



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

#### 4.2.5 Resident Memory (RES)

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

#### 4.2.6 Shared Memory (SHR)

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet,

*Tabel 4.4: 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
Sharpen	25.40	25.14
Sobel Horizontal	25.12	24.68
Sobel Vertical	25.28	24.98
Rata-rata	25.37	24.86

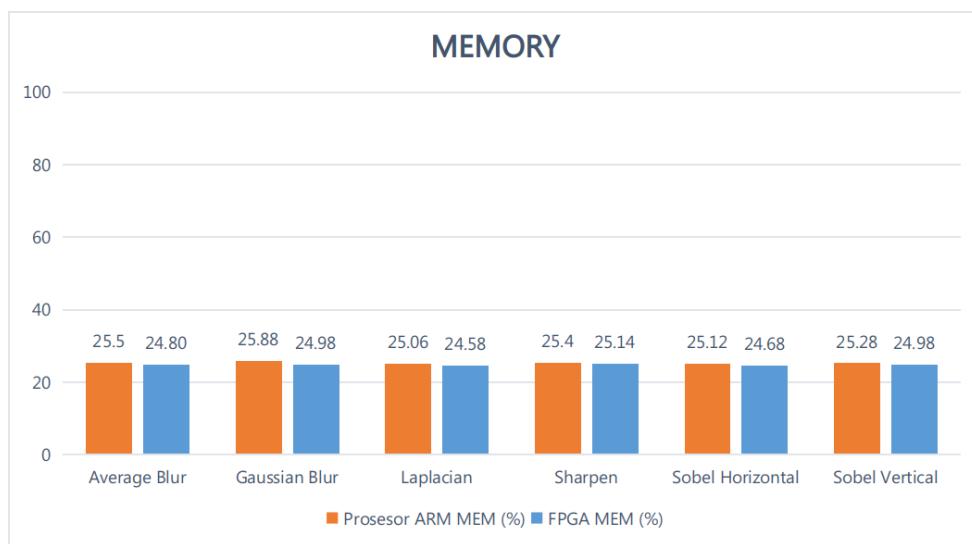
*Tabel 4.5: Tabel perbandingan penggunaan resident memory dengan menggunakan prosesor ARM dan FPGA.*

Filter	Prosesor ARM (KiB)	FPGA (KiB)
Average Blur	129,794.80	126,097.60
Gaussian Blur	131,804.40	127,135.20
Laplacian	127,493.60	125,005.60
Sharpen	129,117.22	127,931.20
Sobel Horizontal	127,830.40	125,420.00
Sobel Vertical	128,611.20	127,033.60
Rata-rata	129,108.60	126,437.20

consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

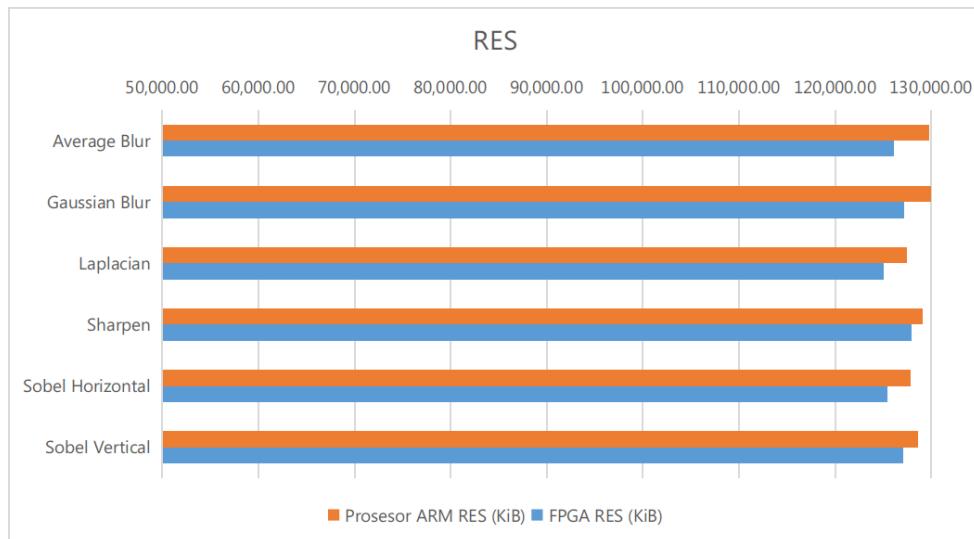
#### 4.2.7 Virtual Memory (VIRT)

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue,



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

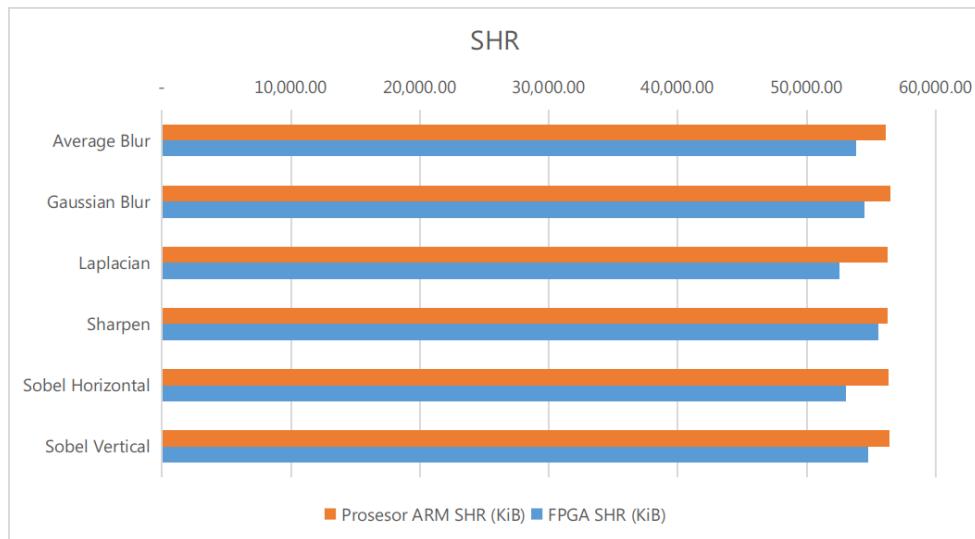
a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.



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

*Tabel 4.6: Tabel perbandingan penggunaan shared memory dengan menggunakan prosesor ARM dan FPGA.*

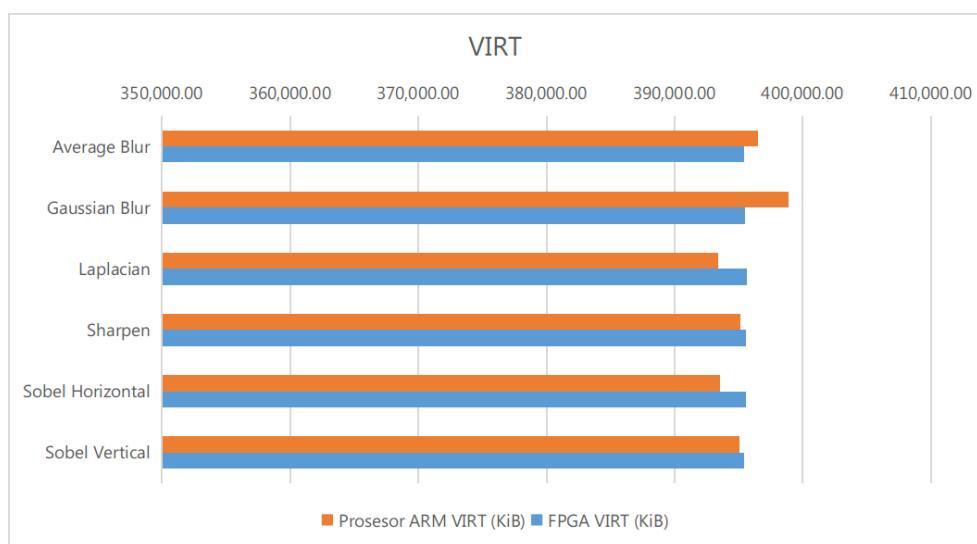
Filter	Prosesor ARM (KiB)	FPGA (KiB)
Average Blur	56,157.40	53,840.00
Gaussian Blur	56,503.60	54,504.80
Laplacian	56,248.00	52,568.00
Sharpen	56,298.82	55,528.80
Sobel Horizontal	56,349.60	53,015.20
Sobel Vertical	56,396.00	54,728.00
Rata-rata	56,325.57	54,030.80



*Gambar 4.19: Grafik perbandingan penggunaan shared memory dengan menggunakan prosesor ARM dan FPGA.*

*Tabel 4.7: Tabel perbandingan penggunaan virtual memory dengan menggunakan prosesor ARM dan FPGA.*

Filter	Prosesor ARM (KiB)	FPGA (KiB)
Average Blur	396,474.64	395396
Gaussian Blur	398,862.80	395488.8
Laplacian	393,388.00	395638.4
Sharpen	395,126.77	395575.2
Sobel Horizontal	393,544.80	395524
Sobel Vertical	395,048.80	395385.6
Rata-rata	395,407.64	395501.3333



*Gambar 4.20: Grafik perbandingan penggunaan virtual memory dengan menggunakan prosesor ARM dan FPGA.*

## **BAB V**

### **KESIMPULAN DAN SARAN**

#### **5.1 Kesimpulan**

  Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

#### **5.2 Saran**

  Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

## 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. et al. (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.
- Chang, A. X. M. and E. Culurciello (2017). “Hardware accelerators for recurrent neural networks on FPGA”. In: *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1–4. doi: 10.1109/ISCAS.2017.8050816.
- 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 et al. (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. et al. (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.
- Sadangi, Sushant et al. (May 2017). “FPGA Implementation of Spatial Filtering techniques for 2D Images”. In: *IEEE International Conference On Recent Trends in Electronics Information & Communication Technology (RTEICT)*.
- 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. et al. (2009). *Teori Pengolahan Citra Digital*. ISBN-13: 978-979-29-0974-6. Jl. Beo 38-40, Yogyakarta 55281: Penerbit Andi.
- Tan, Xin et al. (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 Data Hasil Percobaan ARM Average Blur

#### 7.3.1 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

PID	USER	PR	NI	VIRT	RES	SHR	S	CPU	MEM	TIME
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
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

### 7.3.2 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
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

PID	USER	PR	NI	VIRT	RES	SHR	S	CPU	MEM	TIME
2484	root	20	0	395456	128832	56628	R	100.0	25.3	2:39.55
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

PID	USER	PR	NI	VIRT	RES	SHR	S	CPU	MEM	TIME
2484	root	20	0	395456	128832	56628	R	100.0	25.3	2:54.63
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

### 7.3.3 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

### 7.3.4 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

### 7.3.5 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