Pengolahan Citra Digital

Konvolusi Citra

Disusun Oleh :

Ridho Surya Pangestu (226201033)

PROGRAM STUDI TEKNIK KOMPUTER

JURUSAN TEKNOLOGI INFORMASI

POLITEKNIK NEGERI SAMARINDA

Konvolusi Citra Gambar

1. **Kode dan penjelasan pada setiap baris**

```
import numpy as np  # Menyertakan library numpy sebagai np
import cv2  # Menyertakan library opencv-python

# Mengambil gambar dari direktori lalu ubah menjadi warna abu-abu
image1 = cv2.imread("download.jpeg", 0)
image2 = cv2.imread("NSP.jpg", 0)

# Definisi fungsi konvolusi manual dengan parameter citra dan kernel
def konvolusi(image, kernel):
    row, col = image.shape  # Mengambil nilai lebar dan tinggi dari citra
    mrow, mcol = kernel.shape  # Mengambil nilai lebar dan tinggi dari kernel
    h = int(mrow / 2)  # Mengambil setengah tinggi dari kernel

    # Membuat citra kosong atau citra penuh dengan piksel hitam yang setiap
    # pikselnya berukuran 8bit
    canvas = np.zeros((row, col), np.uint8)
    for i in range(0, row):  # Perulangan untuk membaca setiap baris dari citra
        for j in range(0, col):  # Perulangan untuk membaca setiap kolom dari citra
            # Memeriksa apakah piksel saat ini berada di tepi gambar
            if i == 0 or i == row - 1 or j == 0 or j == col - 1:
                canvas.itemset((i, j), 0)  # Ubah piksel tersebut menjadi hitam
            else:  # Memeriksa apakah piksel saat ini bukan berada di tepi citra
                imgsum = 0
                # Perulangan untuk membaca setiap baris dari kernel
                for k in range(-h, mrow - h):
                    # Perulangan untuk membaca setiap kolom dari kernel
```

```
                for l in range(-h, mcol - h):
                    # Perkalian piksel citra dengan piksel kernel
                    res = image[i + k, j + l] * kernel[h + k, h + l]
                    # Tambahkan hasil perkalian ke variable imgsum
                    imgsum += res
            # Ubah piksel tersebut menjadi hasil konvolusi
            canvas.itemset((i, j), imgsum)
    return canvas  # Mengembalikan hasil konvolusi

# Definisi fungsi kernel High Pass Filter dengan parameter citra
def kernel1(image):
    # Kernel High Pass Filter dengan setiap pikselnya merupakan nilai float 32-bit
    kernel = np.array([[-1 / 9, -1 / 9, -1 / 9], [-1 / 9, 8 / 9, -1 / 9], [-1 / 9, -1 / 9, -1 / 9]],
np.float32)
    # Lakukan proses konvolusi lalu simpan ke variable canvas
    canvas = konvolusi(image, kernel)
    print("Hasil konvolusi kernel1 = ", canvas)  # Cetak piksel hasil konvolusi
    return canvas  # Mengembalikan hasil konvolusi

# Definisi fungsi kernel Low Pass Filter dengan parameter citra
def kernel2(image):
    # Kernel Low Pass Filter dengan setiap pikselnya merupakan nilai float 32-bit
    kernel = np.array([[0, 1 / 8, 0], [1 / 8, 1 / 2, 1 / 8], [0, 1 / 8, 0]], np.float32)
    # Lakukan proses konvolusi lalu simpan ke variable canvas2
    canvas2 = konvolusi(image, kernel)
    print("Hasil konvolusi kernel2 = ", canvas2)  # Cetak piksel hasil konvolusi
    return canvas2

test1 = kernel1(image1)  # Lakukan proses High Pass Filter
print("gambar1 ordo = ", image1.shape)  # Cetak ukuran citra image1
print("gambar1 ori = ", image1)  # Cetak piksel citra image1
```

```
# Cetak ukuran citra setelah di filter dengan High Pass Filter
print("gambar1 HPF ordo = ", test1.shape)
# Cetak piksel citra setelah di filter dengan High Pass Filter
print("gambar1 HPF = ", test1)
cv2.imshow("gambar1", image1)  # Tampilkan citra image1 di jendela
cv2.imshow("High pass", test1)  # Tampilkan citra image1 High Pass Filter di jendela

test2 = kernel2(image2)  # Melakukan proses Low Pass Filter pada image2
print("gambar2 ori ordo = ", image2.shape)  # Mencetak ukuran image2
print("gambar2 ori = ", image2)  # Mencetak seluruh piksel dari image2

row, col = image2.shape  # Mengambil baris dan kolom dari image2
for i in range(0, row):  # Perulangan untuk membaca setiap baris dari image2
    print("pixel = ", image2[i])  # Mencetak piksel dari image2
    # Mencetak nilai min dari piksel image2
    print("nilai min = ", min(image2[i]))
    # Mencetak nilai max dari piksel image2
    print("nilai max = ", max(image2[i]))

# Mencetak ukuran low pass filter image2
print("gambar1 LPF ordo = ", test2.shape)
print("gambar2 LPF = ", test2)  # Mencetak piksel low pass filter image2

cv2.imshow("gambar2", image2)  # Menampilkan image2 pada jendela
cv2.imshow("low pass", test2)  # Menampilkan image2 low pass filter pada jendela
cv2.waitKey(0)  # Menunda selama 0 detik atau menunggu interupsi dari keyboard
cv2.destroyAllWindows()  # Tutup semua jendela gambar
```

2. Hasil

- **Gambar yang digunakan**

Gambar pertama atau image1 awalnya memiliki warna. Setelah dibaca menggunakan fungsi cv2.imread() dengan indeks 0, gambar tersebut berubah menjadi berwarna abu-abu. Gambar asli ditampilkan pada Gambar 1.1, sementara hasil setelah dibaca oleh cv2 ditampilkan pada Gambar 1.2.

Gambar 1.1 Original Image1

Gambar 1.2 image1 setelah dibaca oleh cv2 dengan indeks 0

Gambar kedua atau image2 awalnya memiliki warna. Setelah dibaca menggunakan fungsi cv2.imread() dengan indeks 0, gambar tersebut berubah menjadi berwarna abu-abu. Gambar asli ditampilkan pada Gambar 1.3, sementara hasil setelah dibaca oleh cv2 ditampilkan pada Gambar 1.4.

Gambar 1.3 Original Image2

Gambar 1.4 image2 setelah dibaca oleh cv2 dengan indeks 0

- **Ukuran dan piksel gambar sebelum di filter**

Sebelum image1 dan image2 di filter, ukuran awal image1 adalah (258x195) dan image2 adalah (664x670). Isi dari piksel image1 dan image2 yaitu image1 [[85 85 86 … 1 19 20] ..] dan image2 [[79 79 80 … 44 44 44] …]. Ukuran dan piksel dari image1 dan image2 dapat dilihat pada Gambar 1.5 untuk image1 dan Gambar1.6 untuk image1.

```
gambar1 ordo =  (258, 195)
gambar1 ori =  [[ 85  85  86 ...   1  19   2]
 [ 86  86  87 ...  13  16   6]
 [ 87  87  88 ...  18   2   0]
 ...
 [104 122 139 ...  83  56  52]
 [122 102  96 ...  73  73  72]
 [124 100  87 ...  55  55  55]]
```

Gambar 1 5 ukuran dan piksel image1

```
gambar2 ori ordo =  (664, 670)
gambar2 ori =  [[ 79  79  80 ...  44  44  44]
 [ 67  66  64 ...  31  31  30]
 [ 66  67  65 ...  33  33  32]
 ...
 [165 165 167 ...  94 102  83]
 [159 158 157 ...  78  84  92]
 [163 159 156 ...  78 157  95]]
```

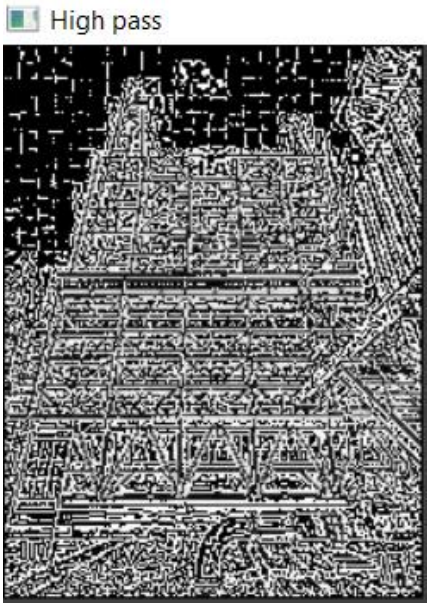
Gambar 1 6 ukuran dan piksel image2

```
pixel =  [ 79  79  80  83  84  80  78  80  88  80  78  86  90  86  83  84  82  80
  79  74  76  80  77  79  82  83  85  88  82  81  90  88  95  94  95 100
 104 101  93  86  87  86  85  86  88  89  88  86  88  87  83  88  89  85
  90  90  88  88  87  86  88  91  93  93  90  93  95  96  96  96  97  99
  97  98 100 101 100  98  98 100  98 101 103 103 104 103 102 101 106 110
 113 113 113 114 112 108 110 110 110 109 107 108 112 116 108 108 110 112
 111 109 109 110 113 111 111 109 110 111 108 110 108 109 107 108 107 105
 108 108 106 104 101 100 100 101 101 101 100 102 103 103 100  98 100 102
 102  99  98 101 104 103 100  98 100 102 105 107 106 103  99  96 101  96
  95  99  95  93  98  92  94  95  92  90  93  91  86  85  88  86  86  89
  89  86  84  84  82  86  87  83  81  82  83  82  78  77  78  81  79  75
  75  78  75  78  74  78  75  75  75  74  72  78  75  77  77  72  73  74
  74  75  77  74  71  75  78  73  72  74  75  72  73  75  76  74  71  68
  67  67  69  70  71  72  65  60  71  64  63  73  65  67  61  60  62  64
  63  61  61  60  61  59  59  61  61  59  57  57  58  59  57  57  59  60
  56  53  55  57  57  56  56  57  58  57  54  55  61  55  51  56  56  54
  63  62  62  62  60  60  65  70  64  61  60  62  64  64  63  63  65  64
  63  64  66  66  65  63  64  68  66  61  61  63  63  62  65  61  58  59
  61  63  62  61  65  68  67  61  62  65  63  65  63  63  63  63  61  60
  61  63  60  62  62  66  62  59  64  57  56  56  55  55  54  57  61  56
  55  60  64  63  60  56  58  65  59  58  54  49  48  51  51  49  57  50
  49  54  55  54  56  58  62  58  51  49  55  58  57  59  57  57  58  58
  58  58  61  64  63  62  63  65  65  64  65  66  73  66  60  61  63  63
  65  68  67  64  62  64  61  60  63  60  63  64  61  65  63  63  70  64
  64  68  68  64  67  67  69  62  60  66  68  67  67  67  66  65  67  72
  73  67  62  62  66  68  61  68  70  63  57  58  62  63  72  64  60  63
  67  66  65  65  65  66  68  69  65  59  61  70  68  61  62  66  63  60
  62  62  60  56  55  62  68  63  60  64  58  56  62  61  58  57  58  67
  65  64  66  60  59  63  62  66  65  62  64  66  67  63  59  63  69  66
  63  62  62  58  64  68  61  63  64  62  60  61  62  59  59  62  58  58
  66  63  57  60  66  64  61  58  58  59  60  61  58  58  58  58  60  60
  56  52  56  63  60  57  57  52  49  56  55  52  56  60  63  61  55  54
  57  55  55  57  58  55  52  51  52  53  55  56  55  57  60  60  58  55
  56  59  56  52  54  57  54  54  54  55  56  57  54  51  51  55  55  54
  48  50  48  49  54  57  55  49  54  50  49  49  50  50  49  48  47  47
  48  49  49  49  50  51  49  45  45  48  47  47  49  50  49  47  47  49
  47  50  50  46  46  48  48  46  47  44  44  42  44  47  46  47  46  45
  45  44  44  44]
nilai min =  42
nilai max =  116
```

Gambar 1 7 piksel pada baris pertama beserta nilai min dan max

- **Hasil dari filter**

Setelah di filter menggunakan High Pass Filter, gambat image1 memiliki edge atau tepi dan gambar memiliki warna hitam dan putih. Tepi ini menunjukkan bahwa tepi tersebut merupakan bagian dari sebuah objek. Hasil filter dapat dilihat pada Gambar 1.8.



Gambar 1 8 image1 setelah di filter menggunakan High Pass Filter

Gambar image1 setelah di filter dengan High Pass Filter tetap memiliki ukuran yang sama seperti sebelumnya yaiut (664x670). Namun ada perbedaan pada nilai piksel. Gambar yang digunakan saat ini, piksel hitam ini dapat terlihat sebagai garis hitam horizontal yang terletak di atas gambar. Ukuran dan piksel image1 dapat dilihat pada Gamabr 1.9.

```
gambar1 LPF ordo =  (664, 670)
gambar1 LPF =  [[  0   0   0 ...   0   0   0]
 [  0  67  66 ...  33  32   0]
 [  0  66  65 ...  32  32   0]
 ...
 [  0 165 166 ...  85  90   0]
 [  0 159 158 ...  81  95   0]
 [  0   0   0 ...   0   0   0]]
```

Gambar 1 9 ukuran dan piksel image1 setelah di filter dengan Low Pass Filter

Gambar image2 kelihatan sedikit blur atau kabur setelah di filter dengan Low Pass Filter dibandingkan dengan gambar image2 sebelum di filter. Hasil filter image2 dapat dilihat pada Gambar 1.10.

Gambar 1 10 image2 setelah di filter dengan Low Pass Filter

Konvolusi Citra Video

1. **Kode dan penjelasan pada setiap baris**

```
import numpy as np  # Menyertakan library numpy sebagai np
import cv2  # Menyertakan library opencv-python

# Mengambil video dari direktori
camera = cv2.VideoCapture("Ridho Surya Pangestu_TK5B_226201033_Tugas2_KonvolusiCitra.mp4")  # konvolusi manual

# Definisi fungsi konvolusi manual dengan parameter citra dan kernel
def konvolusi(image, kernel):
    row, col = image.shape  # Mengambil nilai lebar dan tinggi dari citra
    mrow, mcol = kernel.shape  # Mengambil nilai lebar dan tinggi dari kernel
    h = int(mrow / 2)  # Mengambil setengah tinggi dari kernel

    # Membuat citra kosong atau citra penuh dengan piksel hitam yang setiap pikselnya berukuran 8bit
    canvas = np.zeros((row, col), np.uint8)

    for i in range(0, row):  # Perulangan untuk membaca setiap baris dari citra
        for j in range(0, col):  # Perulangan untuk membaca setiap kolom dari citra
            # Memeriksa apakah piksel saat ini berada di tepi gambar
            if i == 0 or i == row - 1 or j == col - 1:
                canvas.itemset((i, j), 0)  # Ubah piksel tersebut menjadi hitam
            else:
                # Deklarasi variabel untuk penempatan hasil penjumlahan dari hasil perkalian citra dengan kernel
                imgsum = 0

                # Perulangan untuk membaca setiap baris dari kernel
```

```
            for k in range(-h, mrow - h):
                # Perulangan untuk membaca setiap kolom dari kernel
                for l in range(-h, mcol - h):
                    # Perkalian piksel citra dengan piksel kernel
                    res = image[i + k, j + l] * kernel[h + k, h + l]
                    # Tambahkan hasil perkalian ke variabel imgsum
                    imgsum += res

            # Ubah piksel tersebut menjadi hasil konvolusi
            canvas.itemset((i, j), imgsum)

    return canvas  # Mengembalikan hasil konvolusi

# Definisi fungsi kernel High Pass Filter dengan parameter citra
def kernel1(image):
    # Kernel High Pass Filter dengan setiap pikselnya merupakan nilai float 32-bit
    kernel = np.array([[-1 / 9, -1 / 9, -1 / 9], [-1 / 9, 8 / 9, -1 / 9], [-1 / 9, -1 / 9, -1 / 9]],
np.float32)
    # Lakukan proses konvolusi lalu simpan ke variabel canvas
    canvas = konvolusi(image, kernel)
    print("Hasil konvolusi kernel1 = ", canvas)  # Cetak piksel hasil konvolusi
    return canvas  # Mengembalikan hasil konvolusi

# Definisi fungsi kernel Low Pass Filter dengan parameter citra
def kernel2(image):
    # Kernel Low Pass Filter dengan setiap pikselnya merupakan nilai float 32-bit
    kernel = np.array([[0, 1 / 8, 0], [1 / 8, 1 / 2, 1 / 8], [0, 1 / 8, 0]], np.float32)
    # Lakukan proses konvolusi lalu simpan ke variabel canvas2
    canvas2 = konvolusi(image, kernel)
    print("Hasil konvolusi kernel2 = ", canvas2)  # Cetak piksel hasil konvolusi
    return canvas2
```

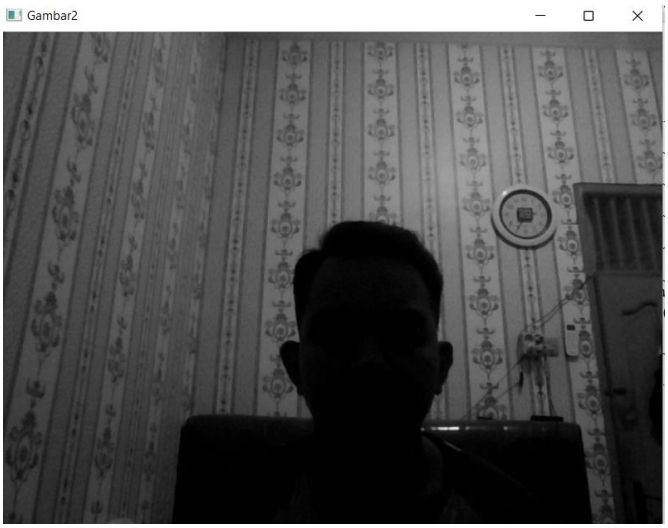
```
while True:
    ret, frame = camera.read()  # Membaca setiap frame dari video
    if not ret:
        break

    image1 = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)  # Mengubah frame video menjadi warna abu-abu
    test1 = kernel1(image1)  # Lakukan proses High Pass Filter
    print("gambar1 ordo = ", image1.shape)  # Cetak ukuran citra frame
    print("gambar1 ori = ", image1)  # Cetak piksel citra frame
    print("gambar1 HPF ordo = ", test1.shape)  # Cetak ukuran citra setelah di-filter dengan High Pass Filter
    print("gambar1 HPF = ", test1)  # Cetak piksel citra setelah di-filter dengan High Pass Filter

    cv2.imshow("gambar1", image1)  # Tampilkan citra frame di jendela
    cv2.imshow("High pass", test1)  # Tampilkan citra frame High Pass Filter di jendela

    test2 = kernel2(image1)  # Lakukan proses Low Pass Filter pada frame
    print("gambar2 ori ordo = ", image1.shape)  # Cetak ukuran frame
    print("gambar2 ori = ", image1)  # Cetak piksel frame
    print("gambar1 LPF ordo = ", test2.shape)  # Cetak ukuran frame Low Pass Filter
    print("gambar2 LPF = ", test2)  # Cetak piksel frame Low Pass Filter

    cv2.imshow("gambar2", image1)  # Tampilkan frame di jendela
    cv2.imshow("low pass", test2)  # Tampilkan frame Low Pass Filter di jendela

    # Menunda selama 1 detik dan menunggu interupsi dari keyboard jika pengguna menekan 'q'
    if cv2.waitKey(1) == ord('q'):
        break  # Hentikan program
```

```
camera.release()  # Menutup kamera
cv2.destroyAllWindows()  # Menutup semua jendela frame
```

2. Hasil

- **Video yang digunakan**

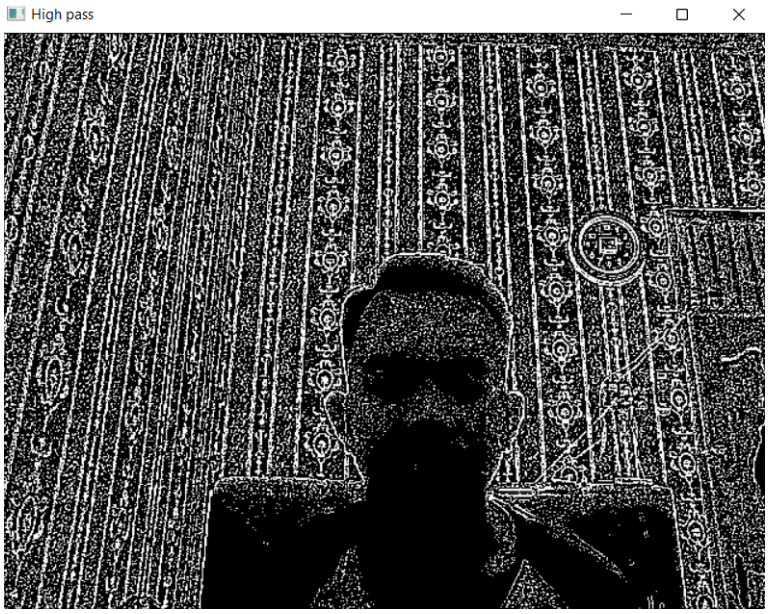
Video yang digunakan berdurasi 21 detik dan dibaca menggunakan objek cv2.VideoCapture(<videopath>) dengan fungsi camera.read(). Lalu, frame ini diubah menjadi warna abu-abu. Frame dapat dilihat pada Gambar 2.1.



Gambar 2 1 video yang digunakan

- **frame setelah di filter dengan High Pass Filter**

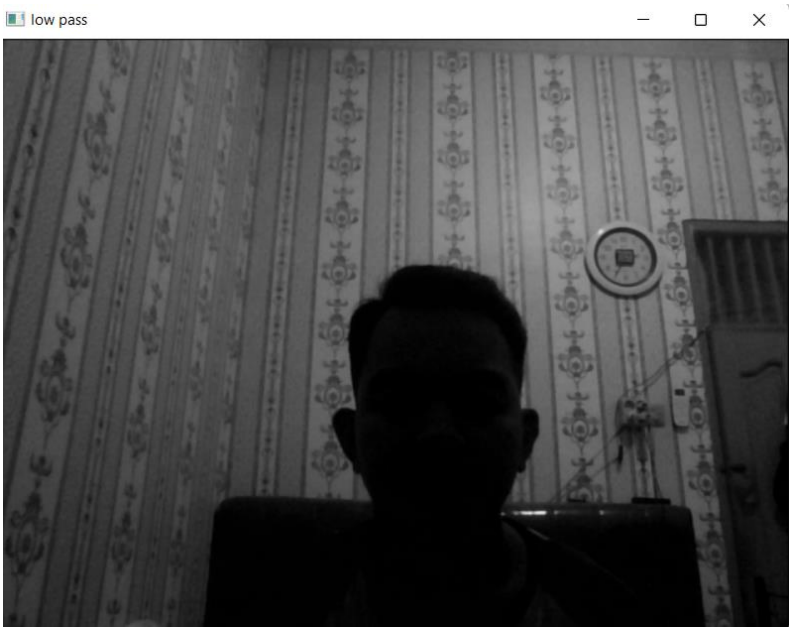
Setelah di filter dengan High Pass Filter, frame memiliki tepi atau edge berwarna putih disekitar objek dan frame menjadi warna hitam dan putih. Hasil frame di filter dengan High Pass Filter dapat dilihat pada Gambar 2.2.



Gambar 2 2 frame setelah di filter dengan High Pass Filter

- **frame setelah di filter dengan Low Pass Filter**

Frame terlihat sedikit kabur atau blur setelah di filter dengan Low Pass Filter dibandingkan frame pada Gambar 2.1. Frame setelah di filter dapat dilihat pada Gambar 2.3.



Gambar 2 3 frame setelah di filter dengan Low Pass Filter

Kesimpulan

Konvolusi pada citra adalah proses yang melibatkan operasi matematis antara dua matriks, yaitu citra input dan kernel konvolusi. Proses ini memerlukan waktu untuk mengeksekusi, terutama ketika citra yang diproses memiliki dimensi yang besar. Secara umum, lamanya waktu pemrosesan konvolusi bergantung pada beberapa faktor utama, salah satunya adalah ukuran citra atau frame video. Semakin besar dimensi atau resolusi dari citra tersebut, maka semakin lama waktu yang dibutuhkan untuk menyelesaikan proses konvolusi. Ini karena setiap elemen dari citra harus dioperasikan dengan kernel, yang pada gilirannya menghasilkan lebih banyak perhitungan untuk citra berukuran besar.

Konvolusi sering kali diimplementasikan menggunakan beberapa lapisan perulangan bersarang (nested for loops) yang bisa mencapai 2 hingga 4 tingkat dalam struktur program. Lapisan perulangan ini diatur untuk menavigasi setiap elemen citra dan menghitung hasil konvolusi untuk setiap piksel. Seiring bertambahnya dimensi matriks atau kernel yang digunakan, waktu pemrosesan konvolusi meningkat secara signifikan. Selain itu, ukuran dari setiap frame video yang digunakan juga menjadi salah satu faktor penentu lamanya waktu eksekusi.

ada beberapa faktor lain yang juga berkontribusi pada rendahnya framerate video. Salah satunya adalah penggunaan fungsi print() yang digunakan untuk mencetak nilai tertentu ke layar selama proses berlangsung. Fungsi print() ini, meskipun tampaknya sederhana, bisa menambah beban waktu eksekusi, terutama jika dijalankan berulang kali di dalam loop yang cepat. Selain itu, proses menampilkan setiap frame dari video secara visual, baik melalui GUI atau jendela tampilan terpisah, juga memerlukan waktu dan mempengaruhi kinerja secara keseluruhan.

Spesifikasi laptop

Processor : Intel Core i5-11400H @ 2.70GHz 2.69 GHz

RAM : 8 GB

VGA : NVIDIA GeForce RTX 3050