**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

1. **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.



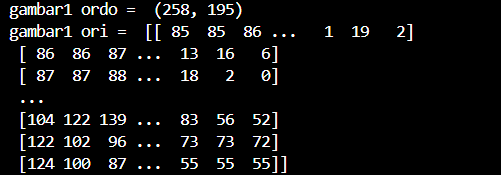
Gambar 1.3 Original Image2



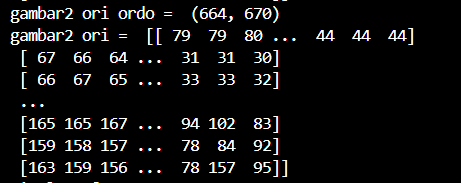
Gambar 1.4 image2 setelah dibaca oleh cv2 dengan indeks 0

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

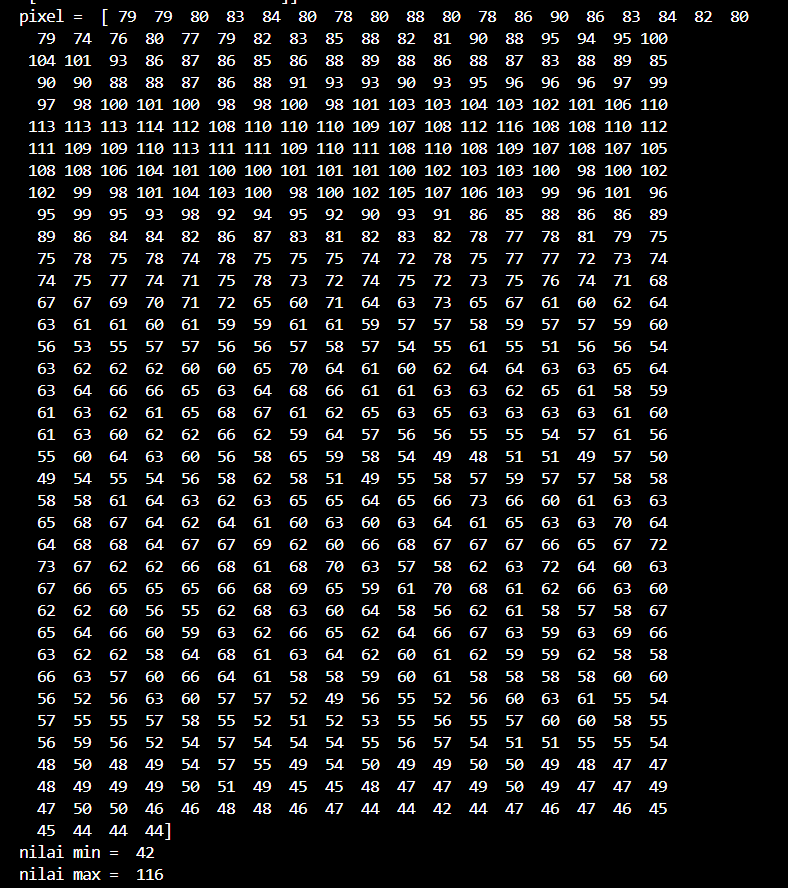
Sebelum image1 dan image2 di filter, ukuran awal image1 adalah (258x195) dan image2 adalah (664x670). Isi dari piksel image1 dan image2 yaitu image1 [[85 85 86 … 1 19 20 ] ..] dan image2 [[79 79 80 … 44 44 44] … ]. Ukuran dan piksel dari image1 dan image2 dapat dilihat pada Gambar 1.5 untuk image1 dan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



Gambar 1 5 ukuran dan piksel image1



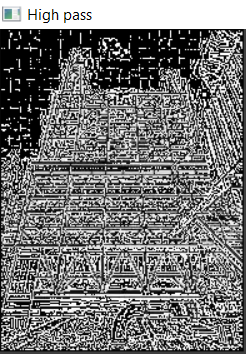
Gambar 1 6 ukuran dan piksel image2



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

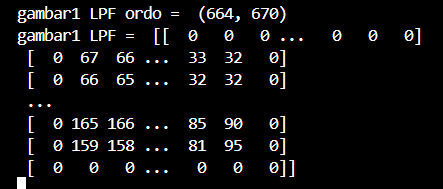
* **Hasil dari filter**

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.



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

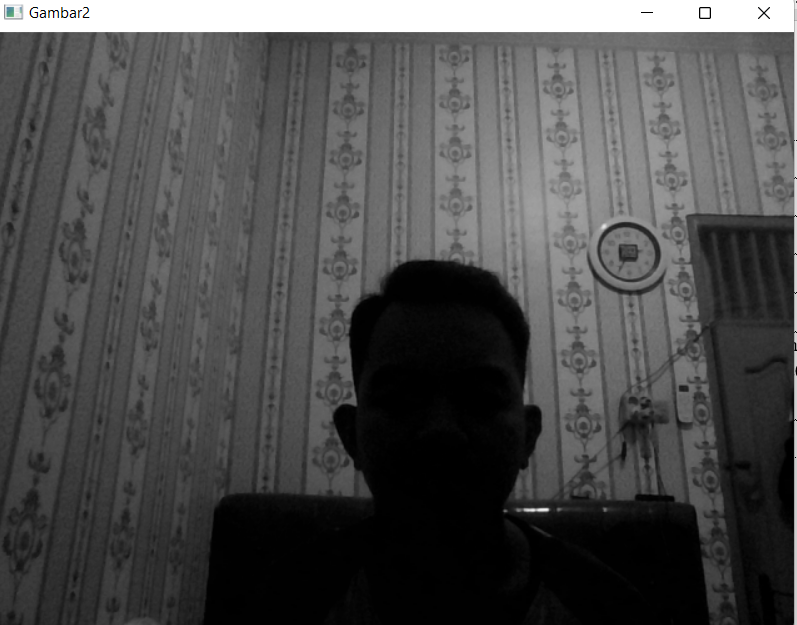
camera.release() # Menutup kamera

cv2.destroyAllWindows() # Menutup semua jendela frame

**2. Hasil**

* **Video yang digunakan**

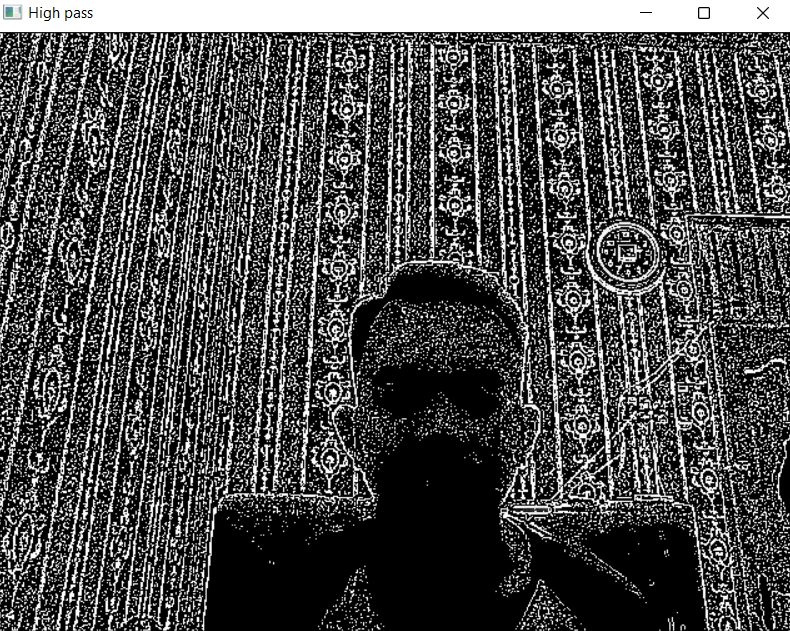
Video yang digunakan berdurasi 21 detik dan dibaca menggunakan objek cv2.VideoCapture(<videopath>) dengan fungsi camera.read(). Lalu, frame ini diubah menjadi warna abu-abu. Frame dapat dilihat pada Gambar 2.1.



Gambar 2 1 video yang digunakan

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

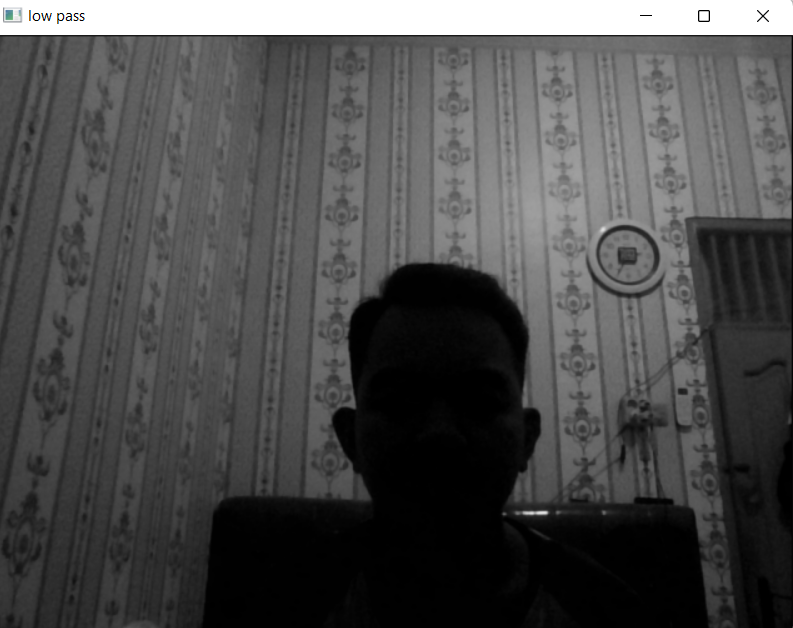
Setelah di filter dengan High Pass Filter, frame memiliki tepi atau edge berwarna putih disekitar objek dan frame menjadi warna hitam dan putih. Hasil frame di filter dengan High Pass Filter dapat dilihat pada Gambar 2.2.



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