**Tugas 4 Pengolahan Citra Digital Praktek –   
Aplikasi Pengolahan Citra Wajah Berbasis FASTAPI**

****

**Disusun oleh :**

Muhammad Rafli Fadhilah (231524057)

**Kelas :**

**D4 – 3B Teknik Informatika**

**JURUSAN TEKNIK KOMPUTER DAN INFORMATIKA**

**PROGRAM STUDI D4-TEKNIK INFORMATIKA**

**POLITEKNIK NEGERI BANDUNG**

**2025**

# DAFTAR ISI

[DAFTAR ISI 2](#_Toc208835912)

[1. Pendahuluan 3](#_Toc208835913)

[1.1 Pembuatan Website Sederhana dengan FastAPI 3](#_Toc208835914)

[1.2 Tujuan 3](#_Toc208835915)

[1.3 Tujuan 3](#_Toc208835916)

[2. Metodologi 3](#_Toc208835917)

[3. Implementasi Fitur Pengolahan Citra 4](#_Toc208835918)

[4. Hasil dan Pembahasan 7](#_Toc208835919)

[5. Kesimpulan 10](#_Toc208835920)

# Deskripsi Aplikasi

Aplikasi ini adalah sistem pengolahan citra wajah berbasis web yang dikembangkan menggunakan **FastAPI** dan **OpenCV**. Aplikasi memungkinkan pengguna untuk:

**Fitur Utama:**

* **Pengumpulan Dataset Wajah**: Menambahkan wajah baru ke dataset menggunakan webcam
* **Pengolahan Citra Wajah**: Melakukan tiga tahap pengolahan citra (noise salt & pepper, penghilangan noise, penajaman)
* **Operasi Citra Umum**: Operasi aritmatika, logika, histogram, konvolusi, dan transformasi Fourier

**Teknologi yang Digunakan:**

* **Backend**: FastAPI (Python)
* **Image Processing**: OpenCV, NumPy, PIL
* **Frontend**: HTML, CSS, Bootstrap, JavaScript
* **Webcam Access**: JavaScript Web API
* **Template Engine**: Jinja2

**Struktur Aplikasi**

fastapi-opencv-26agustus/

├── main.py # Aplikasi utama FastAPI

├── templates/ # Template HTML

│ ├── base.html

│ ├── face\_dataset.html # Form penambahan wajah

│ ├── face\_processing.html # Form pengolahan citra

│ └── ...

├── static/ # File statis (CSS, JS, uploads)

├── dataset/ # Dataset wajah original

└── processed\_dataset/ # Dataset hasil pengolahan

## Pembuatan Website Sederhana dengan FastAPI

Pengolahan citra digital merupakan salah satu bidang yang sangat penting dalam computer vision dan artificial intelligence. Dalam proyek ini, kami mengkonversi algoritma pengolahan citra yang awalnya dikembangkan di Google Colab menjadi aplikasi web berbasis FastAPI yang dapat diakses melalui browser.

## Tujuan

1. Mengkonversi kode pengolahan citra dari Google Colab ke aplikasi web FastAPI
2. Mengimplementasikan fitur-fitur pengolahan citra digital meliputi convolution, zero padding, filtering, Fourier transform, dan noise reduction
3. Menyediakan interface web yang user-friendly untuk pengolahan citra
4. Menguji dan memvalidasi hasil implementasi dengan berbagai gambar sample

## Tujuan

Laporan ini berfokus pada implementasi lima fitur utama pengolahan citra digital:

* **Convolution** (Average, Sharpen, Edge Detection)
* **Zero Padding**
* **Filtering** (Low Pass, High Pass, Band Pass)
* **Fourier Transform**
* **Noise Reduction** (FFT, Bilateral, Gaussian)

# Langkah-langkah Penggunaan Aplikasi

1. **Persiapan dan Instalasi**
2. **Install Dependencies**
3. **Struktur Aplikasi**

fastapi-opencv-26agustus/

├── main.py # File utama aplikasi FastAPI

├── templates/ # Template HTML

│ ├── base.html

│ ├── convolution.html

│ ├── filters.html

│ ├── fourier.html

│ ├── noise\_reduction.html

│ └── zero\_padding.html

└── static/ # File statis dan hasil upload

├── uploads/ # Gambar hasil processing

└── histograms/ # Histogram yang dihasilkan

# Implementasi Fitur Pengolahan Citra

1. **Convolution**

Konvolusi adalah operasi matematika fundamental dalam pengolahan citra yang digunakan untuk menerapkan berbagai efek seperti blur, sharpen, dan edge detection melalui kernel matrix.

def apply\_convolution(image, kernel\_type="average"):

"""Apply convolution with different kernel types"""

if kernel\_type == "average":

kernel = np.ones((3, 3), np.float32) / 9

elif kernel\_type == "sharpen":

kernel = np.array([[0, -1, 0], [-1, 5, -1], [0, -1, 0]])

elif kernel\_type == "edge":

kernel = np.array([[-1, -1, -1], [-1, 8, -1], [-1, -1, -1]])

output\_img = cv2.filter2D(image, -1, kernel)

return output\_img

**Jenis Kernel yang diimplementasikan:**

A. Average Kernel (Smoothing)

* Matrix: 3×3 dengan nilai 1/9
* Fungsi: Menghaluskan gambar dengan rata-rata pixel tetangga
* Efek: Mengurangi noise dan detail halus

B. Sharpen Kernel

* Matrix: [[0, -1, 0], [-1, 5, -1], [0, -1, 0]]
* Fungsi: Meningkatkan ketajaman tepi dan detail
* Efek: Membuat gambar lebih tajam dan kontras

C. Edge Detection Kernel

* Matrix: [[-1, -1, -1], [-1, 8, -1], [-1, -1, -1]]
* Fungsi: Mendeteksi tepi dan kontur objek
* Efek: Menghasilkan outline objek dalam gambar

1. **Zero Padding**

Zero padding adalah teknik menambahkan border dengan nilai pixel 0 (hitam) di sekitar gambar. Teknik ini sering digunakan dalam preprocessing untuk:

* Mempertahankan ukuran gambar setelah konvolusi
* Mencegah kehilangan informasi di tepi gambar
* Menyiapkan gambar untuk operasi FFT

def apply\_zero\_padding(image, padding\_size=10):

"""Apply zero padding to image"""

padded\_img = cv2.copyMakeBorder(

image,

padding\_size, padding\_size, padding\_size, padding\_size,

cv2.BORDER\_CONSTANT,

value=[0, 0, 0]

)

return padded\_img

**Parameter dan Validasi:**

* **Range padding**: 1-200 pixel
* **Validasi input**: Memastikan ukuran padding dalam batas wajar
* **Efek visual**: Border hitam mengelilingi gambar asli

1. **Filtering**

Filtering dalam domain spasial digunakan untuk memodifikasi karakteristik frekuensi gambar. Tiga jenis filter utama yang diimplementasikan:

1. **Low Pass Filter**

if filter\_type == "low":

filtered\_img = cv2.GaussianBlur(image, (15, 15), 0)

* **Fungsi**: Mempertahankan frekuensi rendah, menghilangkan noise
* **Efek**: Gambar menjadi lebih halus dan blur
* **Aplikasi**: Noise reduction, preprocessing

1. **High Pass Filter**

elif filter\_type == "high":

low\_pass = cv2.GaussianBlur(image, (15, 15), 0)

filtered\_img = cv2.subtract(image, low\_pass)

filtered\_img = cv2.add(filtered\_img, np.full(image.shape, 128, dtype=np.uint8))

* **Fungsi**: Mempertahankan detail halus dan tepi
* **Efek**: Menampilkan outline dan texture
* **Teknik**: Subtrasi gambar asli dengan low pass + offset 128

1. **Band Pass Filter**

elif filter\_type == "band":

low\_pass = cv2.GaussianBlur(image, (15, 15), 0)

very\_low\_pass = cv2.GaussianBlur(image, (31, 31), 0)

filtered\_img = cv2.subtract(low\_pass, very\_low\_pass)

filtered\_img = cv2.add(filtered\_img, np.full(image.shape, 128, dtype=np.uint8))

* **Fungsi**: Mempertahankan frekuensi menengah
* **Efek**: Kombinasi smoothing dan edge detection
* **Teknik**: Selisih antara dua level Gaussian blur

1. **Fourier Transform**

Fourier Transform mengkonversi gambar dari domain spasial ke domain frekuensi, memungkinkan analisis komponen frekuensi dalam gambar.

def apply\_fourier\_transform(image):

"""Apply Fourier Transform and return magnitude spectrum"""

gray = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)

# Apply FFT

f = np.fft.fft2(gray)

fshift = np.fft.fftshift(f)

# Calculate magnitude spectrum

magnitude\_spectrum = np.abs(fshift)

# Apply log transform to enhance visibility

magnitude\_spectrum = np.log(magnitude\_spectrum + 1)

# Normalize to 0-255 range for display

magnitude\_spectrum = cv2.normalize(magnitude\_spectrum, None, 0, 255, cv2.NORM\_MINMAX)

return magnitude\_spectrum

**Proses Implementasi:**

1. **Konversi ke Grayscale**: Simplifikasi untuk analisis frekuensi
2. **FFT 2D**: Transformasi ke domain frekuensi
3. **FFT Shift**: Memindahkan komponen DC ke pusat
4. **Magnitude Spectrum**: Menghitung amplitudo setiap frekuensi
5. **Log Transform**: Meningkatkan visibilitas spektrum
6. **Normalisasi**: Konversi ke range 0-255 untuk display
7. **Noise Reduction**

Noise reduction bertujuan mengurangi gangguan pada gambar sambil mempertahankan detail penting. Tiga metode yang diimplementasikan:

1. **FFT-based Noise Reduction**

def reduce\_periodic\_noise(image):

"""Reduce periodic noise using FFT with improved algorithm"""

gray = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)

# Apply FFT

f = np.fft.fft2(gray)

fshift = np.fft.fftshift(f)

# Advanced noise detection and suppression

magnitude\_spectrum = np.abs(fshift)

magnitude\_threshold = np.percentile(magnitude\_spectrum, 99.5)

noise\_locations = magnitude\_spectrum > magnitude\_threshold

# Create adaptive mask

mask = np.ones((rows, cols), np.float32)

mask[noise\_locations] = 0.1 # Suppress instead of eliminate

# Apply inverse FFT

fshift\_filtered = fshift \* mask

f\_ishift = np.fft.ifftshift(fshift\_filtered)

img\_back = np.real(np.fft.ifft2(f\_ishift))

return processed\_image

**Teknik yang digunakan:**

* **Deteksi Noise Selektif**: Identifikasi frekuensi dengan magnitude tinggi (top 0.5%)
* **Preservasi DC Component**: Mempertahankan komponen frekuensi rendah
* **Gradual Suppression**: Atenuasi bertahap (10%) daripada eliminasi total
* **Mask Smoothing**: Gaussian blur pada mask untuk menghindari ringing artifacts

1. **Bilateral Filter**

def reduce\_noise\_bilateral(image):

"""Reduce noise using bilateral filter"""

result = cv2.bilateralFilter(image, 15, 80, 80)

return result

* **Karakteristik**: Mengurangi noise sambil mempertahankan tepi
* **Parameter**: Diameter=15, sigmaColor=80, sigmaSpace=80
* **Kelebihan**: Efektif untuk noise reduction dengan preservasi detail

1. **Gaussian Filter**

def reduce\_noise\_gaussian(image):

"""Reduce noise using Gaussian blur"""

result = cv2.GaussianBlur(image, (5, 5), 0)

return result

* **Karakteristik**: Simple smoothing dengan Gaussian kernel
* **Parameter**: Kernel size 5×5
* **Aplikasi**: General purpose noise reduction

# Hasil dan Pembahasan

1. **Convolutional Results**
   1. **Average Kernel (Blur)**

A collage of a person

AI-generated content may be incorrect.

* **Efek**: Gambar menjadi lebih halus, noise berkurang
* **Trade-off**: Detail halus hilang, gambar sedikit blur
* **Aplikasi**: Preprocessing untuk noise reduction
  1. **Sharpen Kernel**

A collage of a person

AI-generated content may be incorrect.

* **Efek**: Detail dan tepi lebih tajam dan kontras
* **Peningkatan**: Tekstur dan fine details lebih jelas
* **Aplikasi**: Enhancement untuk gambar yang blur
  1. **Edge Detection Kernel**

**A person's face and a black and white picture

AI-generated content may be incorrect.**

* **Efek**: Outline objek dan boundary terdeteksi
* **Hasil**: Struktur gambar dalam bentuk edges
* **Aplikasi**: Computer vision, object detection

1. **Zero Padding Results**

**A collage of a flower bed

AI-generated content may be incorrect.**

* **Padding Size**: 20 pixels
* **Efek Visual**: Border hitam mengelilingi gambar asli
* **Preservasi**: Gambar asli tetap utuh di tengah
* **Aplikasi**: Preprocessing untuk FFT dan convolution

1. **Filtering Results**
   1. **Low Pass Filter**

A collage of a stack of books

AI-generated content may be incorrect.

* Efek: Smoothing effect, noise reduction
* Karakteristik: Frekuensi tinggi dihilangkan
* Trade-off: Detail halus berkurang
  1. **High Pass Filter**

A collage of a picture of a building and a flower bed

AI-generated content may be incorrect.

* Efek: Edge enhancement, detail preservation
* Karakteristik: Frekuensi rendah dihilangkan
* Hasil: Outline dan texture lebih prominence
  1. **Band Pass Filter**

A collage of a picture of a building and a flower bed

AI-generated content may be incorrect.

* Efek: Kombinasi smoothing dan edge detection
* Karakteristik: Mempertahankan frekuensi menengah
* Hasil: Balance antara noise reduction dan detail preservation

1. **Fourier Transform Results**

**A collage of a picture of a building and a flower bed

AI-generated content may be incorrect.**

* Representasi: Magnitude spectrum dalam domain frekuensi
* Pusat Bright: Komponen DC (frekuensi rendah)
* Pattern Analysis: Struktur frekuensi dalam gambar
* Aplikasi: Analisis tekstur, noise detection

1. **Noise Reduction Results**
   1. **FFT-based Noise Reduction**

* Metode: Selective frequency suppression
* Efek: Periodic noise berkurang significantly
* Preservasi: Detail penting tetap terjaga
  1. **Bilateral Filter**
* Karakteristik: Edge-preserving smoothing
* Efek: Noise berkurang dengan tepi tetap tajam
* Kualitas: Superior untuk photographic images
  1. **Gaussian Filter**
* Karakteristik: General smoothing
* Efek: Uniform noise reduction
* Trade-off: Semua frekuensi tinggi berkurang

# Kesimpulan

Proyek konversi algoritma pengolahan citra digital dari Google Colab ke aplikasi web FastAPI telah berhasil diimplementasikan dengan sempurna, mencakup lima fitur utama yaitu **Convolution** (average, sharpen, edge detection), **Zero Padding**, **Filtering** (low pass, high pass, band pass), **Fourier Transform**, dan **Noise Reduction** (FFT-based, bilateral, gaussian). Implementasi ini menunjukkan transformasi yang sukses dari kode penelitian akademik menjadi aplikasi web yang praktis dan user-friendly dengan interface HTML yang intuitif, error handling yang robust, dan validasi input yang komprehensif.

Penggunaan FastAPI, OpenCV, NumPy, dan scikit-image memberikan foundation yang solid untuk pengembangan lebih lanjut. Proyek ini berhasil menjembatani gap antara research dan practical application dalam bidang computer vision, membuktikan bahwa algoritma akademik dapat ditransformasikan menjadi tools yang dapat diakses dan digunakan oleh pengguna umum melalui web interface yang responsif dan efisien.