# **Analisis Teoritis dan Matematis Kode Program Bilateral Filter dengan Korelasi ke Jurnal dan Materi Kuliah (Grayscale)**

## **1. Konsep Dasar Bilateral Filter**

### **1.1 Definisi dan Tujuan**

**Bilateral Filter** adalah teknik penghalusan citra yang bertujuan untuk:

* Mengurangi **noise** tanpa mengaburkan **tepi (edges)**.
* Mempertahankan **warna alami** pada gambar berwarna melalui ruang warna **CIE-Lab**.

### **1.2 Persamaan Matematis**

Dari jurnal ICCV 1998, persamaan bilateral filter adalah:

Dimana:

* : Nilai pixel hasil filter di lokasi .
* : Nilai pixel tetangga di lokasi .
* : Fungsi kedekatan spasial (domain kernel).
* : Fungsi kesamaan intensitas/warna (range kernel).
* : Faktor normalisasi untuk memastikan bobot jumlahnya 1.

## **2. Implementasi Kode Program**

### **2.1 Struktur Kode Utama**

Kode program mencakup:

1. **Precompute Similarity LUT**: Optimasi perhitungan kesamaan intensitas.
2. **Bilateral Filter Manual**: Menghitung nilai pixel baru dengan kombinasi domain dan range.
3. **Dinamisasi Sigma\_r**: Penyesuaian parameter berdasarkan histogram.
4. **Iterative Filtering**: Peningkatan efek “cartoon-like”.
5. **Konversi ke CIE-Lab**: Preservasi warna perceptual.

### **2.2 Contoh Perhitungan Matematis**

#### **Langkah 1: Domain Kernel (Gaussian)**

Untuk domain kernel, kita menggunakan fungsi Gaussian:

Dimana adalah jarak Euclidean antara lokasi pixel dan .

**Contoh Numerik**: Misal , dan kernel size :

ax = [-1, 0, 1] # Koordinat relatif dari pixel tengah  
xx, yy = np.meshgrid(ax, ax) # Membuat grid 3x3  
d = np.sqrt(xx\*\*2 + yy\*\*2)  
domain\_kernel = np.exp(-d\*\*2 / (2 \* sigma\_d\*\*2))

Hasil domain\_kernel:

[[0.3679 0.6065 0.3679]  
 [0.6065 1.0000 0.6065]  
 [0.3679 0.6065 0.3679]]

#### **Langkah 2: Range Kernel (Similarity Function)**

Fungsi kesamaan intensitas:

**Contoh Numerik**: Jika , , dan :

diff = np.array([10, 0, 10])  
range\_weights = np.exp(-diff\*\*2 / (2 \* sigma\_r\*\*2))

Hasil range\_weights:

[0.6065, 1.0000, 0.6065]

#### **Langkah 3: Gabungan Domain dan Range**

Kombinasi bobot:

**Contoh Numerik**: Dengan domain\_kernel dan range\_weights di atas:

combined\_weights = domain\_kernel \* range\_weights

Hasil combined\_weights:

[[0.2231 0.6065 0.2231]  
 [0.3679 1.0000 0.3679]  
 [0.2231 0.6065 0.2231]]

#### **Langkah 4: Normalisasi dan Hasil Akhir**

Normalisasi:

**Contoh Numerik**: Jika dan bobot combined\_weights = [0.6, 1.0, 0.6]:

k\_x = np.sum(combined\_weights)  
h\_x = np.sum(f\_xi \* combined\_weights) / k\_x

Hasil:

## **3. Korelasi dengan Materi Kuliah**

### **3.1 Domain Spasial (Spatial Domain)**

Dari materi kuliah:

* Operasi langsung pada pixel citra: .
* Termasuk **point processing** (transformasi intensitas) dan **mask processing** (filtering dengan kernel).

**Korelasi dengan Kode**:

* **Mask Processing**: Kode menggunakan kernel domain (spasial) dan range (intensitas) untuk menghitung nilai pixel baru.
* **Point Processing**: Dinamisasi sigma\_r berdasarkan histogram (fungsi auto\_sigma\_r).

### **3.2 Histogram dan Point Processing**

Dari materi kuliah:

* Histogram digunakan untuk analisis distribusi intensitas.
* Transformasi intensitas seperti contrast stretching dan log transformation.

**Korelasi dengan Kode**:

* Fungsi auto\_sigma\_r menghitung standar deviasi histogram untuk menyesuaikan parameter filter.
* Analisis histogram (analyze\_histogram) untuk validasi hasil filtering.

### **3.3 Edge Preservation**

Dari jurnal dan materi kuliah:

* Bilateral filter mempertahankan tepi dengan menggabungkan domain dan range filtering.

**Contoh Studi Kasus**:

* **Input**: Gambar dengan tepi tajam dan noise (misal: foto kucing dengan latar belakang buram).
* **Output**: Hasil bilateral filter mengurangi noise pada area datar (smooth region) tetapi mempertahankan tepi (contour kucing).

## **4. Penggunaan CIE-Lab untuk Gambar Warna**

### **4.1 Alasan Penggunaan CIE-Lab**

Dari jurnal:

* Ruang warna CIE-Lab mempertahankan persepsi warna manusia (perceptually meaningful).
* Filter hanya diterapkan pada channel Luminance (L) untuk preservasi warna alami.

### **4.2 Proses Konversi**

1. Konversi RGB ke CIE-Lab:

* lab = cv2.cvtColor(img\_rgb, cv2.COLOR\_RGB2Lab)

1. Pisahkan channel:

* l, a, b = cv2.split(lab)

1. Terapkan bilateral filter hanya pada channel L:

* l\_filtered = bilateral\_filter(Image.fromarray(l), sigma\_d, sigma\_r)

1. Gabung kembali channel:

* enhanced\_lab = cv2.merge([np.array(l\_filtered), a, b])  
  result = cv2.cvtColor(enhanced\_lab, cv2.COLOR\_Lab2RGB)

**Contoh Studi Kasus**:

* **Input**: Foto bunga dengan warna cerah.
* **Output**: Warna tetap alami meskipun telah dilakukan filtering untuk mengurangi noise.

## **5. Iterative Filtering untuk Efek “Cartoon-Like”**

### **5.1 Konsep**

Iterasi berturut-turut dari bilateral filter menghasilkan efek flattening warna tanpa mengaburkan tepi (seperti gambar kartun).

### **5.2 Proses Matematis**

Setiap iterasi:

for \_ in range(iterations):  
 result = bilateral\_filter(result, sigma\_d, sigma\_r)

**Contoh Studi Kasus**:

* **Input**: Foto alam (pegunungan).
* **Output**: Setelah 5 iterasi, warna menjadi lebih flat (monoton) tetapi tepi tetap tajam.

## **6. Validasi dengan Histogram**

### **6.1 Analisis Histogram**

Fungsi analyze\_histogram membandingkan distribusi intensitas sebelum dan sesudah filtering.

**Contoh Output**:

* Histogram input: Distribusi intensitas lebar (ada noise).
* Histogram output: Distribusi lebih sempit (noise berkurang).

### **6.2 Korelasi dengan Materi Kuliah**

Dari materi kuliah:

* Histogram equalization dan contrast stretching digunakan untuk meningkatkan kontras.

**Korelasi dengan Kode**:

* Dinamisasi sigma\_r berdasarkan histogram mirip dengan contrast stretching adaptif.

## **7. Kesimpulan**

### **7.1 Aspek Teoritis**

* Kode program mengimplementasikan **bilateral filter** sesuai jurnal ICCV 1998 dengan:
  + Kombinasi **domain** (spasial) dan **range** (intensitas/warna).
  + Penggunaan **CIE-Lab** untuk preservasi warna perceptual.
  + Dinamisasi parameter berdasarkan histogram.

### **7.2 Aspek Matematis**

* Persamaan integral bilinear diubah menjadi operasi diskret dengan kernel Gaussian.
* Normalisasi bobot memastikan konsistensi hasil.

### **7.3 Relevansi dengan Materi Kuliah**

* Sesuai dengan konsep **spatial domain filtering** dan **point processing**.
* Mengintegrasikan **histogram analysis** untuk optimasi parameter.
* Memenuhi tujuan **peningkatan kualitas citra** dengan preservasi tepi dan pengurangan noise.

### **8. Studi Kasus Sederhana: Penerapan Bilateral Filter pada Citra Grayscale dengan Noise**

Berikut adalah **analisis langkah-demi-langkah** penerapan **Bilateral Filter** pada **citra grayscale dengan noise**, dilengkapi dengan **perhitungan matematis manual** dan **korelasi ke teori**. Studi kasus ini menggunakan **matriks 3×3** sebagai input dan menjelaskan proses penghalusan pada **pixel tengah** (koordinat ) dengan nilai intensitas .

## **Input Citra**

Diberikan citra grayscale berikut (nilai intensitas 0–255):

[90, 100, 110]  
[80, 100, 120]  
[70, 100, 130]

**Tujuan**: Menghitung nilai pixel tengah setelah diterapkan Bilateral Filter dengan parameter:

* (domain kernel).
* (range kernel).

## **Langkah 1: Domain Kernel (Spasial)**

### **1.1 Definisi**

Domain kernel mengukur kedekatan spasial antara pixel tengah () dan tetangganya () menggunakan fungsi Gaussian:

Di mana adalah jarak Euclidean antara lokasi pixel.

### **1.2 Perhitungan Manual**

Untuk kernel size , kita definisikan grid koordinat relatif ():

Koordinat relatif:  
xx = [[-1, 0, 1],  
 [-1, 0, 1],  
 [-1, 0, 1]]  
yy = [[-1, -1, -1],  
 [0, 0, 0],  
 [1, 1, 1]]

**Jarak Euclidean** :

Hasil :

[[1.414, 1.0, 1.414],  
 [1.0, 0.0, 1.0],  
 [1.414, 1.0, 1.414]]

**Domain Kernel ()**:

Hitung untuk setiap posisi:

1. :
2. :
3. :

Hasil domain\_kernel:

[[0.7788, 0.8825, 0.7788],  
 [0.8825, 1.0000, 0.8825],  
 [0.7788, 0.8825, 0.7788]]

### **1.3 Eksekusi Kode**

ax = [-1, 0, 1]  
xx, yy = np.meshgrid(ax, ax)  
d = np.sqrt(xx\*\*2 + yy\*\*2)  
domain\_kernel = np.exp(-d\*\*2 / (2 \* sigma\_d\*\*2))

### **1.4 Korelasi dengan Teori**

* **Jurnal ICCV 1998**:
  + Domain kernel sesuai Persamaan 5–6:
  + Fungsi Gaussian untuk domain filtering (halaman 2.1).
* **Materi Kuliah**:
  + Spatial domain filtering (Slide 5–6):  
    , di mana adalah operator berbasis kernel.

## **Langkah 2: Range Kernel (Intensitas/Warna)**

### **2.1 Definisi**

Range kernel mengukur kesamaan intensitas antara pixel tengah dan tetangganya :

### **2.2 Perhitungan Manual**

Diketahui dan . Hitung selisih intensitas () untuk setiap tetangga:

Δf = f(ξ) - f(x) =   
[[90-100, 100-100, 110-100] → [-10, 0, 10]  
 [80-100, 100-100, 120-100] → [-20, 0, 20]  
 [70-100, 100-100, 130-100]] → [-30, 0, 30]

**Range Weights ()**:

Hitung untuk setiap :

1. :
2. :
3. :
4. :
5. :
6. :
7. :

Hasil range\_weights:

[[0.6065, 1.0000, 0.6065],  
 [0.1353, 1.0000, 0.1353],  
 [0.0111, 1.0000, 0.0111]]

### **2.3 Eksekusi Kode**

diff = np.array([10, 0, 10])  
range\_weights = np.exp(-diff\*\*2 / (2 \* sigma\_r\*\*2))

### **2.4 Korelasi dengan Teori**

* **Jurnal ICCV 1998**:
  + Persamaan 5–6:  
     adalah fungsi Gaussian berbasis kesamaan intensitas (halaman 2.1).
* **Materi Kuliah**:
  + Point processing (Slide 10–11):  
    Transformasi intensitas berbasis histogram (contrast stretching adaptif).

## **Langkah 3: Gabungan Domain dan Range**

### **3.1 Definisi**

Bobot gabungan () adalah hasil perkalian domain dan range kernel:

### **3.2 Perhitungan Manual**

Gabungkan domain\_kernel dan range\_weights:

1. **Baris 1 (Top)**:
   * Hasil: [0.472, 0.8825, 0.472]
2. **Baris 2 (Middle)**:
   * Hasil: [0.119, 1.0000, 0.119]
3. **Baris 3 (Bottom)**:
   * Hasil: [0.0086, 0.8825, 0.0086]

Hasil combined\_weights:

[[0.472, 0.8825, 0.472 ],  
 [0.119, 1.0000, 0.119 ],  
 [0.0086, 0.8825, 0.0086]]

### **3.3 Eksekusi Kode**

combined\_weights = domain\_kernel \* range\_weights

### **3.4 Korelasi dengan Teori**

* **Jurnal ICCV 1998**:
  + Kombinasi domain dan range kernel (halaman 2.1).
* **Materi Kuliah**:
  + Mask processing (Slide 6):  
    Operator menggabungkan kedekatan spasial dan kesamaan intensitas.

## **Langkah 4: Normalisasi Bobot**

### **4.1 Definisi**

Faktor normalisasi () memastikan jumlah bobot = 1:

### **4.2 Perhitungan Manual**

Jumlahkan semua bobot:

### **4.3 Eksekusi Kode**

k\_x = np.sum(combined\_weights)

### **4.4 Korelasi dengan Teori**

* **Jurnal ICCV 1998**:
  + Normalisasi bobot (Persamaan 6):  
    .
* **Materi Kuliah**:
  + Histogram analysis (Slide 10–11):  
    Normalisasi bobot serupa dengan contrast stretching adaptif.

## **Langkah 5: Hasil Akhir (Filtered Pixel)**

### **5.1 Definisi**

Nilai pixel hasil filter () adalah rata-rata tertimbang intensitas tetangga:

### **5.2 Perhitungan Manual**

Kalikan intensitas tetangga dengan bobot gabungan , lalu jumlahkan:

1. **Baris 1 (Top)**:
   * Jumlah:
2. **Baris 2 (Middle)**:
   * Jumlah:
3. **Baris 3 (Bottom)**:
   * Jumlah:

**Total Weighted Sum**:

**Hasil Akhir**:

Namun, karena nilai dibulatkan ke bilangan bulat terdekat, hasil akhir tetap **100**.

### **5.3 Eksekusi Kode**

f\_xi = np.array([[90, 100, 110],  
 [80, 100, 120],  
 [70, 100, 130]])  
weighted\_sum = np.sum(f\_xi \* combined\_weights)  
h\_x = weighted\_sum / k\_x

### **5.4 Korelasi dengan Teori**

* **Jurnal ICCV 1998**:
  + Persamaan 5–6:  
    .
* **Materi Kuliah**:
  + Edge Preservation (Slide 21–22):  
    Bobot gabungan mengurangi pengaruh tetangga yang kontras.

## 6. **Hasil dan Analisis**

### **6.1 Hasil**

Pixel tengah () **tetap bernilai 100** setelah filtering, meskipun ada noise di sekitarnya (misal: dan ).

### **6.2 Interpretasi Teoretis**

* **Preservasi Tepi**: Pixel tengah tidak terpengaruh oleh tetangga yang intensitasnya sangat berbeda (misal: atau ) karena bobot gabungan () sangat kecil ().
* **Reduksi Noise**: Pixel dengan intensitas ekstrem ( dan ) memiliki bobot kecil, sehingga dampak noise berkurang.

### **6.3 Eksekusi Kode**

def analyze\_histogram(input\_path, output\_path):  
 img\_input = np.array(Image.open(input\_path))  
 img\_output = np.array(Image.open(output\_path))  
 plt.figure(figsize=(12, 5))  
 for i, title in enumerate(['Input', 'Output']):  
 plt.subplot(1, 2, i+1)  
 plt.hist(img\_output.ravel(), bins=256, alpha=0.5, color='gray')  
 plt.savefig("histogram\_comparison.png")

### **6.4 Korelasi dengan Materi Kuliah**

#### **6.4.1 Spatial Domain Filtering**

* **Teori**: Operasi langsung pada pixel dengan kernel (Slide 5: ).
* **Kode**: Domain kernel menggabungkan intensitas tetangga dengan bobot berbasis jarak (Slide 6: Operator ).

#### **6.4.2 Histogram dan Point Processing**

* **Teori**: Dinamisasi parameter berdasarkan histogram (Slide 10–11: Contrast Stretching dan Log Transformasi).
* **Kode**: Dinamisasi berdasarkan standar deviasi histogram (fungsi auto\_sigma\_r).

#### **6.4.3 Edge Preservation**

* **Teori**: Preservasi tepi dengan menggabungkan domain dan range filtering (Slide 21–22: Contrast Stretching).
* **Kode**: Bobot gabungan () mempertahankan tepi dengan mengurangi pengaruh tetangga yang kontras.

## **7. Flowchart Proses Studi Kasus**

Input Citra:  
[90, 100, 110]  
[80, 100, 120]  
[70, 100, 130]  
Proses:  
1. Hitung domain\_kernel (Gaussian) → Bobot berbasis jarak spasial.  
2. Hitung range\_weights (Gaussian) → Bobot berbasis kesamaan intensitas.  
3. Gabung domain dan range → Bobot gabungan.  
4. Normalisasi bobot → Pastikan jumlah bobot = 1.  
5. Hitung nilai pixel tengah → Rata-rata tertimbang.  
Output:  
[90, 100, 110]  
[80, 100, 120]  
[70, 100, 130]

## **8. Kesimpulan Studi Kasus**

### **9.1 Efektivitas Bilateral Filter**

* **Noise Reduction**: Tetangga dengan intensitas ekstrem ( dan ) diabaikan karena bobot kecil.
* **Edge Preservation**: Pixel tengah tetap bernilai , tidak terpengaruh oleh tetangga yang kontras.

### **9.2 Relevansi dengan Jurnal dan Materi Kuliah**

* **Jurnal ICCV 1998**: Sesuai dengan prinsip “domain + range filtering” (Persamaan 5–6).
* **Materi Kuliah**:
  + **Spatial Domain**: Menggunakan kernel untuk operasi langsung pada pixel (Slide 5–6).
  + **Point Processing**: Dinamisasi berdasarkan histogram (Slide 10–11).
  + **Edge Preservation**: Preservasi tepi melalui bobot berbasis kesamaan intensitas (Slide 21–22).

### **9.3 Validasi dengan Histogram**

Histogram input:

Intensitas: 70, 80, 90, 100, 110, 120, 130  
Frekuensi: 1, 1, 1, 3, 1, 1, 1

Histogram output:

Intensitas: 70, 80, 90, 100, 110, 120, 130  
Frekuensi: 1, 1, 1, 3, 1, 1, 1

Tidak ada perubahan frekuensi karena pixel tengah tetap , menunjukkan efek smoothing tanpa distorsi.

## **Studi Kasus Efek “Cartoon-Like”**

### **Input Citra**

Citra grayscale dengan tepi vertikal tajam:

[100, 100, 130]  
[100, 100, 130]  
[100, 100, 130]

**Tujuan**: Menerapkan **5 iterasi Bilateral Filter** dengan parameter:

* (domain kernel).
* (range kernel).

### **Langkah 1: Iterasi Pertama**

#### **1.1 Domain Kernel (Spasial)**

Domain kernel dihitung dengan fungsi Gaussian:

Untuk , hasil domain\_kernel:

[[0.3679, 0.6065, 0.3679],  
 [0.6065, 1.0000, 0.6065],  
 [0.3679, 0.6065, 0.3679]]

#### **1.2 Range Kernel (Intensitas)**

Range kernel dihitung berdasarkan kesamaan intensitas:

Untuk , hasil range\_weights untuk baris tengah ():

Baris 2 (Middle):  
[80, 100, 120] → Δf = [-20, 0, 20]  
range\_weights = [0.1353, 1.0000, 0.1353]

#### **1.3 Gabungan Domain dan Range**

Bobot gabungan adalah hasil perkalian domain dan range kernel:

combined\_weights = domain\_kernel \* range\_weights

Hasil combined\_weights untuk baris 2 (Middle):

[0.0819, 1.0000, 0.0819]

#### **1.4 Normalisasi Bobot**

Faktor normalisasi memastikan jumlah bobot = 1:

Contoh perhitungan:

#### **1.5 Hasil Pixel Tengah (Middle Center)**

Nilai pixel hasil filter adalah rata-rata tertimbang:

Contoh perhitungan:

* Intensitas tetangga : [80, 100, 120].
* Bobot gabungan : [0.0819, 1.0000, 0.0819].
* Weighted sum:
* Hasil akhir:

**Output Iterasi 1**:

[100, 100, 125]  
[100, 100, 125]  
[100, 100, 125]

### **Langkah 2: Iterasi Kedua**

#### **2.1 Update Range Kernel**

Setelah iterasi pertama, intensitas tetangga berubah. Contoh:

* Tetangga kanan () → .
* Range weights baru:

#### **2.2 Bobot Gabungan Baru**

Gabung domain\_kernel dan range\_weights baru:

Baris 2 (Middle):  
[0.0819, 1.0000, 0.044 \cdot 0.6065 = 0.0266]

#### **2.3 Normalisasi Bobot**

#### **2.4 Hasil Pixel Tengah**

* Intensitas tetangga : [80, 100, 125].
* Bobot gabungan : [0.0819, 1.0000, 0.0266].
* Weighted sum:
* Hasil akhir:

**Output Iterasi 2**:

[100, 100, 120]  
[100, 99, 120]  
[100, 100, 120]

### **Langkah 3: Iterasi Ketiga**

#### **3.1 Update Range Kernel**

Tetangga kanan () → :

#### **3.2 Bobot Gabungan Baru**

Baris 2 (Middle):  
[0.0819, 1.0000, 0.1353 \cdot 0.6065 = 0.0819]

#### **3.3 Normalisasi Bobot**

#### **3.4 Hasil Pixel Tengah**

* Intensitas tetangga : [80, 99, 120].
* Bobot gabungan : [0.0819, 1.0000, 0.0819].
* Weighted sum:
* Hasil akhir:

**Output Iterasi 3**:

[100, 100, 115]  
[100, 99, 115]  
[100, 100, 115]

### **Langkah 4: Iterasi Keempat**

#### **4.1 Update Range Kernel**

Tetangga kanan () → :

#### **4.2 Bobot Gabungan Baru**

Baris 2 (Middle):  
[0.0819, 1.0000, 0.3247 \cdot 0.6065 = 0.1970]

#### **4.3 Normalisasi Bobot**

#### **4.4 Hasil Pixel Tengah**

* Intensitas tetangga : [80, 99, 115].
* Bobot gabungan : [0.0819, 1.0000, 0.1970].
* Weighted sum:
* Hasil akhir:

**Output Iterasi 4**:

[100, 100, 110]  
[100, 100, 110]  
[100, 100, 110]

### **Langkah 5: Iterasi Kelima**

#### **5.1 Update Range Kernel**

Tetangga kanan () →

#### **5.2 Bobot Gabungan Baru**

Baris 2 (Middle):  
[0.0819, 1.0000, 0.6065 \cdot 0.6065 = 0.3679]

#### **5.3 Normalisasi Bobot**

#### **5.4 Hasil Pixel Tengah**

* Intensitas tetangga : [80, 100, 110].
* Bobot gabungan (): [0.0819, 1.0000, 0.3679].
* Weighted sum:
* Hasil akhir:

**Output Iterasi 5**:

[100, 100, 105]  
[100, 101, 105]  
[100, 100, 105]

### 6. **Hasil dan Analisis**

#### **6.1 Output Akhir**

Setelah 5 iterasi, intensitas citra menjadi lebih flat:

[100, 100, 105]  
[100, 101, 105]  
[100, 100, 105]

* Area kiri tetap bernilai **100**.
* Area kanan menjadi **105** (flat).
* Tepi tetap tajam (tidak ada blending antara 100 dan 105).

#### **6.2 Interpretasi Teoretis**

* **Flattening Warna**: Setiap iterasi mengurangi variasi intensitas dalam area seragam.
* **Preservasi Tepi**: Bobot gabungan sangat kecil untuk tetangga yang kontras (misal: dan ) → :
* Tetapi tetangga dari area berbeda memiliki bobot rendah karena jarak spasial jauh dari kernel.

#### **6.3 Korelasi dengan Teori**

* **Jurnal ICCV 1998**:
  + Iterasi menghasilkan efek “cartoon-like” (Gambar 7(c)).
  + Area seragam terbentuk tanpa mengaburkan tepi.
* **Materi Kuliah**:
  + Mask processing berturut-turut (Slide 6–7):  
    Operator diterapkan berulang kali untuk efek visual spesifik.

### **Langkah 7: Validasi dengan Histogram**

#### **6.1 Histogram Input**

Intensitas: 100, 130  
Frekuensi: 6, 3

### **6.2 Histogram Output (Iterasi 5)**

Intensitas: 100, 101, 105  
Frekuensi: 6, 1, 2

* Distribusi intensitas menjadi lebih sempit → **reduksi noise**.
* Tidak ada nilai intensitas ekstrem (misal: 130) → **flattening warna**.

### **Korelasi dengan Jurnal dan Materi Kuliah**

| Aspek | Penghitungan Manual | Teori Jurnal | Materi Kuliah |
| --- | --- | --- | --- |
| **Flattening Warna** | Iterasi mengurangi variasi intensitas dalam area seragam (misal: area kanan → 105). | Jurnal ICCV 1998: Iterasi menghasilkan efek “cartoon-like” (Gambar 7(c)). | Slide 6–7: Operator diterapkan berulang kali untuk efek visual spesifik. |
| **Preservasi Tepi** | Tepi tetap tajam meskipun intensitas area berbeda (100 vs 105). | Jurnal ICCV 1998: Tidak ada “phantom colors” di tepi (halaman 4–5). | Slide 21–22: Edge preservation dengan contrast stretching adaptif. |
| **Normalisasi Bobot** | Jumlah bobot selalu = 1 setelah setiap iterasi. | Persamaan 5–6: |  |
| . | Slide 10–11: Normalisasi bobot mirip dengan contrast stretching. |  |  |

### **Flowchart Proses**

Input Citra  
 ↓  
Iterasi 1: Hasil = [100, 100, 125] dst.  
 ↓  
Iterasi 2: Hasil = [100, 100, 120] dst.  
 ↓  
Iterasi 3: Hasil = [100, 100, 115] dst.  
 ↓  
Iterasi 4: Hasil = [100, 100, 110] dst.  
 ↓  
Iterasi 5: Hasil = [100, 100, 105] dst.

### **Referensi**

* **Jurnal ICCV 1998**: Gambar 7(c) → Efek “cartoon-like” dengan iterasi.
* **Materi Kuliah**: Slide 6–7 (mask processing berturut-turut), Slide 21–22 (edge preservation).
* **Studi Kasus**: Efek filtering pada gambar bunga dan alam.

Dengan analisis ini, kita dapat memahami **bagaimana iterasi Bilateral Filter menghasilkan efek “cartoon-like”** melalui proses matematis manual dan korelasi langsung ke teori jurnal/materi kuliah.

### **10. Referensi**

* **Jurnal ICCV 1998**: Persamaan 5–6 untuk bilateral filter.
* **Materi Kuliah**: Slide 3–5 tentang domain spasial dan point processing.
* **Studi Kasus**: Efek filtering pada gambar bunga dan alam.

# **Analisis Teoritis, Matematis, dan Implementasi Kode Program Bilateral Filter dengan Studi Kasus Input (Berwarna dengan CIE-Lab)**

Berikut adalah penjelasan **terintegrasi** antara **teori jurnal ICCV 1998**, **materi kuliah “Peningkatan Kualitas Citra Spatial”**, dan **implementasi kode program**, dilengkapi dengan **studi kasus input sederhana** untuk memvalidasi proses penghitungan manual dan korelasi teori.

## **1. Konsep Dasar Bilateral Filter**

### **1.1 Definisi dan Tujuan**

**Bilateral Filter** adalah teknik penghalusan citra yang bertujuan untuk:

* Mengurangi **noise** tanpa mengaburkan **tepi (edges)**.
* Mempertahankan **warna alami** pada gambar berwarna melalui ruang warna **CIE-Lab**.

### **1.2 Persamaan Matematis**

Dari jurnal ICCV 1998, persamaan bilateral filter adalah:

Dimana:

* : Nilai pixel hasil filter di lokasi .
* : Nilai pixel tetangga di lokasi .
* : Fungsi kedekatan spasial (domain kernel).
* : Fungsi kesamaan intensitas/warna (range kernel).
* : Faktor normalisasi untuk memastikan bobot jumlahnya 1.

## **2. Implementasi Kode Program**

### **2.1 Struktur Kode Utama**

Kode program mencakup:

1. **Precompute Similarity LUT**: Optimasi perhitungan kesamaan intensitas.
2. **Bilateral Filter Manual**: Menghitung nilai pixel baru dengan kombinasi domain dan range.
3. **Dinamisasi Sigma\_r**: Penyesuaian parameter berdasarkan histogram.
4. **Iterative Filtering**: Peningkatan efek “cartoon-like”.
5. **Konversi ke CIE-Lab**: Preservasi warna perceptual.

### **2.2 Contoh Kode**

def bilateral\_filter(image, sigma\_d=5, sigma\_r=50, similarity\_lut=None):  
 """  
 Implementasi bilateral filter yang mendukung gambar grayscale dan warna.  
 """  
 img\_array = np.array(image).astype(np.float32)  
   
 # Jika gambar grayscale (2D), tambahkan dimensi channel (h, w) → (h, w, 1)  
 if len(img\_array.shape) == 2:  
 img\_array = img\_array[:, :, np.newaxis] # Tambahkan dimensi ke-3 (channel)  
   
 h, w, c = img\_array.shape # Sekarang selalu berhasil  
 result = np.zeros\_like(img\_array)  
   
 # Kernel domain (Gaussian)  
 kernel\_size = int(2 \* np.ceil(3 \* sigma\_d)) + 1  
 ax = np.arange(-kernel\_size//2 + 1, kernel\_size//2 + 1)  
 xx, yy = np.meshgrid(ax, ax)  
 domain\_kernel = np.exp(-(xx\*\*2 + yy\*\*2) / (2 \* sigma\_d\*\*2))  
 domain\_kernel /= np.sum(domain\_kernel)  
  
 # Gunakan LUT jika tersedia  
 if similarity\_lut is None:  
 similarity\_lut = precompute\_similarity\_lut(256, sigma\_r)  
  
 # Proses setiap pixel  
 for y in range(h):  
 for x in range(w):  
 i\_min = max(0, y - kernel\_size//2)  
 i\_max = min(h, y + kernel\_size//2 + 1)  
 j\_min = max(0, x - kernel\_size//2)  
 j\_max = min(w, x + kernel\_size//2 + 1)  
   
 for ch in range(c):  
 local\_window = img\_array[i\_min:i\_max, j\_min:j\_max, ch]  
 center\_val = img\_array[y, x, ch]  
   
 # Range weights (dari LUT)  
 diff = np.abs(local\_window.astype(int) - int(center\_val))  
 range\_weights = similarity\_lut[diff]  
   
 # Combined weights  
 combined\_weights = domain\_kernel[:i\_max-i\_min, :j\_max-j\_min] \* range\_weights  
 normalized\_weights = combined\_weights / np.sum(combined\_weights)  
   
 # Konvolusi terboboti  
 result[y, x, ch] = np.sum(local\_window \* normalized\_weights)  
   
 # Hapus dimensi channel jika grayscale  
 if c == 1:  
 result = np.squeeze(result, axis=2)  
   
 return Image.fromarray(result.astype(np.uint8))

## **3. Studi Kasus Input: Citra Grayscale dengan Noise**

### **3.1 Input Citra**

Diberikan citra grayscale berikut (nilai intensitas 0–255):

[90, 100, 110]  
[80, 100, 120]  
[70, 100, 130]

**Tujuan**: Menghitung nilai pixel tengah () setelah diterapkan Bilateral Filter dengan parameter:

* (domain kernel).
* (range kernel).

### **3.2 Proses Matematis Manual**

#### **Langkah 1: Domain Kernel (Spasial)**

Domain kernel mengukur kedekatan spasial antara pixel tengah () dan tetangganya () menggunakan fungsi Gaussian:

Di mana adalah jarak Euclidean antara lokasi pixel.

**Contoh Perhitungan**: Untuk , kernel size , hasil domain\_kernel:

[[0.3679, 0.6065, 0.3679],  
 [0.6065, 1.0000, 0.6065],  
 [0.3679, 0.6065, 0.3679]]

#### **Langkah 2: Range Kernel (Intensitas/Warna)**

Range kernel mengukur kesamaan intensitas antara pixel tengah dan tetangganya :

**Contoh Perhitungan**: Untuk dan , hasil range\_weights:

[[0.6065, 1.0000, 0.6065],  
 [0.1353, 1.0000, 0.1353],  
 [0.0111, 1.0000, 0.0111]]

#### **Langkah 3: Gabungan Domain dan Range**

Bobot gabungan adalah hasil perkalian domain dan range kernel:

**Contoh Perhitungan**: Gabungkan domain\_kernel dan range\_weights:

combined\_weights = domain\_kernel \* range\_weights

Hasil combined\_weights:

[[0.2231, 0.6065, 0.2231],  
 [0.0819, 1.0000, 0.0819],  
 [0.0041, 0.6065, 0.0041]]

#### **Langkah 4: Normalisasi Bobot**

Faktor normalisasi memastikan jumlah bobot = 1:

**Contoh Perhitungan**:

k\_x = np.sum(combined\_weights) = 3.2231

#### **Langkah 5: Hasil Akhir (Filtered Pixel)**

Nilai pixel hasil filter adalah rata-rata tertimbang intensitas tetangga:

**Contoh Perhitungan**: Kalikan intensitas tetangga dengan bobot gabungan , lalu jumlahkan:

f\_xi = np.array([[90, 100, 110],  
 [80, 100, 120],  
 [70, 100, 130]])  
weighted\_sum = np.sum(f\_xi \* combined\_weights)   
 = (90×0.2231) + (100×0.6065) + ... + (130×0.0041)   
 = 322.31

**Hasil Akhir**:

## **4. Studi Kasus Input: Citra Warna dengan Ruang Warna CIE-Lab**

### **4.1 Input Citra**

Diberikan citra warna sederhana dengan nilai RGB untuk pixel tengah dan tetangga:

Pixel Tengah (x): [R=120, G=80, B=60] → Warna: Merah Tua  
Tetangga Kiri (ξ1): [R=110, G=70, B=50]  
Tetangga Kanan (ξ2): [R=130, G=90, B=70]

**Tujuan**: Menghitung nilai pixel tengah setelah diterapkan Bilateral Filter dengan parameter:

* (domain kernel).
* (range kernel).

### **4.2 Proses Konversi ke CIE-Lab**

#### **Langkah 1: Konversi RGB ke XYZ**

Gunakan matriks transformasi:

**Contoh Perhitungan**: Untuk pixel tengah ():

Hasil XYZ:

X ≈ 0.4124×0.4706 + 0.3576×0.3137 + 0.1805×0.2353 ≈ 0.364  
Y ≈ 0.2126×0.4706 + 0.7152×0.3137 + 0.0722×0.2353 ≈ 0.289  
Z ≈ 0.0193×0.4706 + 0.1192×0.3137 + 0.9505×0.2353 ≈ 0.275

#### **Langkah 2: Konversi ke CIE-Lab**

Gunakan referensi white point D65: .

**Contoh Perhitungan**:

### **4.3 Filter Hanya pada Channel Luminance (L)**

Setelah konversi ke CIE-Lab, filter hanya diterapkan pada channel L.

**Contoh Perhitungan**:

* (tidak berubah karena kesamaan intensitas).
* Channel a dan b tetap tidak berubah.

**Hasil CIE-Lab**:

[60.8, 32.5, 5.8]

### **4.4 Konversi ke RGB**

Gunakan invers transformasi CIE-Lab → XYZ → RGB:

**Contoh Perhitungan**:

* , , → Warna tetap **Merah Tua**.

## **5. Korelasi dengan Teori Jurnal dan Materi Kuliah**

### **5.1 Preservasi Tepi (Edge Preservation)**

* **Teori Jurnal ICCV 1998**:
  + Bilateral filter menggabungkan domain dan range kernel untuk preservasi tepi (Persamaan 5–6).
* **Materi Kuliah**:
  + Operator menggabungkan kedekatan spasial dan kesamaan intensitas (Slide 6–7).
* **Implementasi Kode**:
  + Bobot gabungan sangat kecil untuk tetangga yang kontras → tepi tetap tajam.

### **5.2 Preservasi Warna Perceptual**

* **Teori Jurnal ICCV 1998**:
  + CIE-Lab mempertahankan persepsi warna manusia (halaman 5).
* **Materi Kuliah**:
  + Transformasi warna berbasis histogram (Slide 10–11).
* **Implementasi Kode**:
  + Filter diterapkan hanya pada channel L → warna tetap alami.

### **5.3 Histogram Analysis**

* **Teori Jurnal ICCV 1998**:
  + Histogram compression (Gambar 2(b)) menunjukkan noise reduction tanpa distorsi.
* **Materi Kuliah**:
  + Contrast stretching adaptif berbasis histogram (Slide 20–24).
* **Implementasi Kode**:
  + Dinamisasi berdasarkan standar deviasi histogram.

## **6. Validasi dengan Studi Kasus**

### **6.1 Input Citra**

Foto bunga dengan warna cerah:

* **Pixel Tengah**: Merah Muda (RGB: [200, 100, 100]).
* **Tetangga Kiri**: Merah Tua (RGB: [180, 90, 90]).
* **Tetangga Kanan**: Putih (RGB: [255, 255, 255]).

### **6.2 Proses**

1. **Konversi ke CIE-Lab**:
   * Merah Muda: .
   * Merah Tua: .
   * Putih: .
2. **Filter Channel L**:
   * ,  
     .
   * ,  
     .

* **Combined Weights**:
* [0.6065 \* 0.8825 ≈ 0.535, 1.000, 0.6065 \* 0.044 ≈ 0.027]

1. **Normalisasi Bobot**:  
   .
2. **Hasil Pixel Tengah**:

* .

1. **Konversi ke RGB**:
   * → RGB: [195, 95, 95] (Merah Muda sedikit lebih gelap).

### **6.3 Hasil dan Analisis**

* **Preservasi Warna**:
  + Tidak ada warna “pink/purple” yang muncul di tepi antara Merah Muda dan Putih (sesuai jurnal halaman 7).
* **Reduksi Noise**:
  + Intensitas L tetap seragam (70.0 → 69.0) meskipun ada tetangga Putih yang kontras.
* **Edge Preservation**:
  + Tetangga Putih diabaikan karena bobot kecil ().

## **7. Flowchart Proses**

Input Citra (RGB)  
 ↓  
Konversi ke CIE-Lab → Pisahkan L, a, b  
 ↓  
Terapkan Bilateral Filter hanya pada channel L  
 ↓  
Gabung kembali channel L, a, b  
 ↓  
Konversi ke RGB → Output Citra

## **8. Validasi dengan Histogram**

### **8.1 Histogram Input**

Intensitas: 70, 80, 90, 100, 110, 120, 130  
Frekuensi: 1, 1, 1, 3, 1, 1, 1

### **8.2 Histogram Output**

Intensitas: 70, 80, 90, 100, 110, 120, 130  
Frekuensi: 1, 1, 1, 3, 1, 1, 1

→ Tidak ada perubahan frekuensi karena pixel tengah tetap , menunjukkan efek smoothing tanpa distorsi.

## **9. Korelasi dengan Jurnal dan Materi Kuliah**

| Aspek | Penghitungan Manual | Teori Jurnal | Materi Kuliah |
| --- | --- | --- | --- |
| **Preservasi Tepi** | Bobot gabungan sangat kecil untuk tetangga kontras (misal: → ). | Jurnal ICCV 1998: Tidak ada “phantom colors” di tepi (Gambar 6(d)). | Slide 21–22: Edge preservation dengan contrast stretching adaptif. |
| **Preservasi Warna** | Channel a dan b tetap tidak berubah. Warna tetap Merah Tua (tidak ada blending warna ekstrem). | Jurnal ICCV 1998: CIE-Lab mempertahankan persepsi warna manusia (halaman 5). | Slide 10–11: Transformasi warna berbasis histogram. |
| **Normalisasi Bobot** | Jumlah bobot selalu = 1 . | Persamaan 5–6: . | Slide 10–11: Normalisasi bobot mirip dengan contrast stretching. |

## **10. Referensi**

* **Jurnal ICCV 1998**: Gambar 6(d) → Preservasi warna dengan CIE-Lab.
* **Materi Kuliah**: Slide 10–11 (contrast stretching), Slide 21–22 (edge preservation).
* **Studi Kasus**: Efek filtering pada gambar bunga dan alam.

Dengan korelasi ini, kita dapat memahami \*\*bagaimana Bilateral Filter bekerja pada citra berwarna (RGB) \*\* melalui ruang warna CIE-Lab.