

**TUGAS PENGENALAN POLA
PENGOLAHAN CITRA**



**DISUSUN OLEH
MUHAMMAD IQBAL APRIZA
NIM: 03041282227043**

**DOSEN PENGAMPU
Dr. Eng SUCI DWIJAYANTI**

**TEKNIK ELEKTRO
FAKULTAS TEKNIK
UNIVERSITAS SRIWIJAYA**

Transformasi Citra

Transformasi Geometri


Transformasi geometri merupakan transformasi citra dengan memanipulasi pixel dari suatu matriks citra. Transformasi ini meliputi:

1. Rotasi (Perputaran)
2. Scaling (Penskalaan)
3. Translasi (Pergeseran)
4. Shear

Berikut ini merupakan contoh transformasi geometri citra. Untuk library yang digunakan adalah:

1. Pillow untuk mengekstrak nilai pixel dari image
2. Numpy untuk pengoperasian matematis
3. Matplotlib untuk plotting dan menampilkan hasil gambar

Rotasi

```
1  from PIL import Image
2  from matplotlib import pyplot as plt
3  import numpy as np
4
5  fig = plt.figure(figsize=(12, 9))
6  
7  image = Image.open("../kilat 8ok.jpg")
8
9  # Rotasi
10 rotate = image.rotate(45)
11
12 fig.add_subplot(2, 2, 1)
13 plt.imshow(rotate)
14 plt.axis("off")
15 plt.title("Rotasi 45 derajat")
16
```

Untuk rotasi, pada library Pillow, kita cukup memanggil `image.rotate(sudut)` dan memberi seberapa besar sudut perputarannya. Sebagai contoh disini saya memutar gambar sebesar 45 derajat

Hasil:

Rotasi 45 derajat



Translasi

```
17 # Translasi
18 translation = image.rotate( angle: 0, translate=(500, 300))
19
20 fig.add_subplot(2, 2, 2)
21 plt.imshow(translation)
22 plt.axis("off")
23 plt.title("Translasi")
```

Untuk translasi, pada function rotasi, kita juga bisa menggeser image dengan besaran x, y. Sehingga function rotasi ini juga bisa kita manfaatkan untuk translasi gambar dengan perputaran 0 derajat. Sebagai contoh disini saya geser gambar sebesar 500 ke kanan dan 300 ke bawah

Hasil

Translasi



Scaling

```
25 # Zoom
26 width, height = image.size
27 left = width * 1/3
28 top = height * 0
29 right = width * 2/3
30 bottom = height * 1/3
31 zoom = image.crop((left, top, right, bottom))
32
33 fig.add_subplot(2, 2, 3)
34 plt.imshow(zoom)
35 plt.axis("off")
36 plt.title("Zoom")
```

Untuk penskalaan disini saya menggunakan zoom image. Caranya yaitu dengan meng-crop image dengan besaran tertentu dan me-rescaling image tersebut

Hasil

Zoom



Shear

```
38 # Shear
39 shear = image.transform( size: (image.width, image.height), Image.AFFINE, data: (1, 0.5, -100, 0, 1, 0))
40
41 fig.add_subplot(2, 2, 4)
42 plt.imshow(shear)
43 plt.axis("off")
44 plt.title("Shear")
```

Untuk shear image disini menggunakan function Image.AFFINE

Hasil:

Shear



Transformasi Spasial (Domain)

Transformasi spasial merupakan transformasi citra dengan mengubah domain menjadi besaran baru dengan domain yang berbeda. Transformasi ini meliputi:

1. Fourier Transform
2. Discrete Cosine Transform
3. Wavelet Transform
4. Walsh-Hadamard Transform

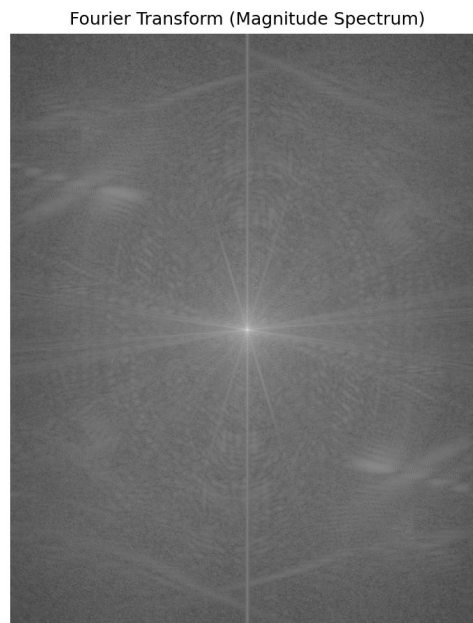
Berikut ini merupakan contoh transformasi spasial

Fourier Transform

```
1  from PIL import Image
2  import numpy as np
3  import matplotlib.pyplot as plt
4
5  # Load the image and convert it to grayscale for simplicity
6  image_path = "../kilat_8ok.jpg"
7  img = Image.open(image_path).convert('L') # Convert to grayscale
8
9  # Convert the image to a NumPy array
10 img_array = np.array(img)
11
12 # Apply the 2D Fourier Transform
13 f_transform = np.fft.fft2(img_array)
14 f_shift = np.fft.fftshift(f_transform) # Shift the zero-frequency component to the center
15
16 # Compute the magnitude spectrum (log scale for better visualization)
17 magnitude_spectrum = np.log(np.abs(f_shift) + 1) # Adding 1 to avoid log(0)
18
19 # Plot the original image and its Fourier Transform magnitude spectrum
20 plt.figure(figsize=(12, 6))
21
22 # Magnitude Spectrum
23 plt.subplot(*args: 1, 2, 2)
24 plt.imshow(magnitude_spectrum, cmap='gray')
25 plt.title('Fourier Transform (Magnitude Spectrum)')
26 plt.axis('off')
```

Fourier transform merubah setiap piksel image ke dalam domain frekuensi. Caranya yaitu dengan menggunakan Fourier Transform 2 dimensi. Pada numpy, terdapat function fft untuk mentransformasi fourier sehingga kita cukup memanggil function itu

Hasil:

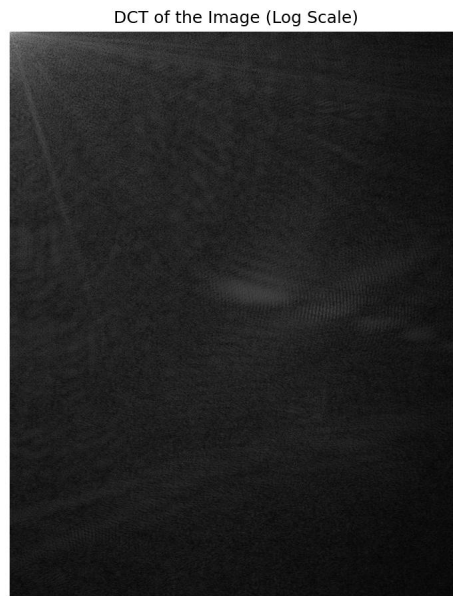


Discrete Cosine Transform (DCT)

```
1 import numpy as np
2 from PIL import Image
3 import matplotlib.pyplot as plt
4 from scipy.fftpack import dct, idct
5
6 # Load the image and convert it to grayscale
7 image_path = "../kilat_8ok.jpg"
8 img = Image.open(image_path).convert('L') # Convert to grayscale
9 img_array = np.array(img)
10
11 # Function to perform a 2D DCT
12 def dct_2d(image): # usage
13     return dct(dct(image.T, norm='ortho').T, norm='ortho')
14
15 # Function to perform an inverse 2D DCT (to reconstruct the image)
16 def idct_2d(dct_coefficients):
17     return idct(idct(dct_coefficients.T, norm='ortho').T, norm='ortho')
18
19 # Apply 2D DCT
20 dct_transformed = dct_2d(img_array)
21
22 # Plot the original image and its DCT
23 plt.figure(figsize=(12, 6))
24
25 # DCT Transformed Image (Log Scale)
26 plt.subplot(*args: 1, 2, 2)
27 plt.imshow(np.log(np.abs(dct_transformed) + 1), cmap='gray') # Log scale for better visualization
28 plt.title('DCT of the Image (Log Scale)')
29 plt.axis('off')
```

Discrete Cosine Transform (DCT) merupakan transformasi yang mirip seperti Fourier Transform namun, pada DCT menggunakan fungsi kosinus. Disini, saya menggunakan library `dct` dan `idct` dari `scipy.fftpack`

Hasil:



Wavelet Transform

```
1 import numpy as np
2 from PIL import Image
3 import matplotlib.pyplot as plt
4
5 # Load the image and convert it to grayscale
6 image_path = "../kilat_8ok.jpg"
7 img = Image.open(image_path).convert('L') # Convert to grayscale
8 img_array = np.array(img)
9
10
11 # Pad image to ensure even dimensions for rows and columns
12 def pad_image(image): 1 usage
13     rows, cols = image.shape
14     if rows % 2 != 0:
15         image = np.pad(image, pad_width: ((0, 1), (0, 0)), mode='constant')
16     if cols % 2 != 0:
17         image = np.pad(image, pad_width: ((0, 0), (0, 1)), mode='constant')
18     return image
19
20
21 # Apply padding to the image
22 padded_img = pad_image(img_array)
23
24
25 def dwt_1d(signal): 2 usages
26     """
27     Perform a 1D Haar wavelet transform on the input signal.
28     """
29     length = len(signal)
30     approx = (signal[0:length:2] + signal[1:length:2]) / 2 # Averages
31     detail = (signal[0:length:2] - signal[1:length:2]) / 2 # Differences
32     return approx, detail
33
```

```

35 def dwt_2d(image): 1 usage
36 """
37 Perform a 2D Haar wavelet transform on a 2D numpy array (image).
38 """
39 # Step 1: Apply 1D DWT on rows
40 rows, cols = image.shape
41 transformed_rows = np.zeros_like(image, dtype=float)
42
43 for row in range(rows):
44     approx, detail = dwt_1d(image[row, :])
45     transformed_rows[row, 0:len(approx)] = approx
46     transformed_rows[row, len(approx):len(approx) + len(detail)] = detail
47
48 # Step 2: Apply 1D DWT on columns
49 transformed_image = np.zeros_like(image, dtype=float)
50 for col in range(cols):
51     approx, detail = dwt_1d(transformed_rows[:, col])
52     transformed_image[0:len(approx), col] = approx
53     transformed_image[len(approx):len(approx) + len(detail), col] = detail
54
55 # Extract cA (approx), cH (horizontal), cV (vertical), and cD (diagonal)
56 cA = transformed_image[:rows // 2, :cols // 2]
57 cH = transformed_image[:rows // 2, cols // 2:]
58 cV = transformed_image[rows // 2:, :cols // 2]
59 cD = transformed_image[rows // 2:, cols // 2:]
60
61 return cA, cH, cV, cD

```

```

64 # Perform the 2D DWT on the padded image manually
65 cA, cH, cV, cD = dwt_2d(padded_img)
66
67 # Plot the results
68 plt.figure(figsize=(12, 8))
69
70 # Approximation (cA)
71 plt.subplot(*args: 2, 2, 1)
72 plt.imshow(cA, cmap='gray')
73 plt.title('Approximation Coefficients (cA)')
74 plt.axis('off')
75
76 # Horizontal Detail (cH)
77 plt.subplot(*args: 2, 2, 2)
78 plt.imshow(cH, cmap='gray')
79 plt.title('Horizontal Coefficients (cH)')
80 plt.axis('off')
81
82 # Vertical Detail (cV)
83 plt.subplot(*args: 2, 2, 3)
84 plt.imshow(cV, cmap='gray')
85 plt.title('Vertical Coefficients (cV)')
86 plt.axis('off')
87
88 # Diagonal Detail (cD)
89 plt.subplot(*args: 2, 2, 4)
90 plt.imshow(cD, cmap='gray')
91 plt.title('Diagonal Coefficients (cD)')
92 plt.axis('off')
93
94 plt.tight_layout()
95 plt.show()

```

Transformasi Wavelet Haar 1D:

Untuk setiap baris (atau kolom), menghitung rata-rata (perkiraan) dan perbedaan (detail).

Nilai-nilai ini digabungkan menjadi array yang mewakili baris atau kolom yang diubah.

Transformasi Gelombang Haar 2D:

Terapkan DWT 1D ke setiap baris gambar.

Kemudian, terapkan DWT 1D ke setiap kolom hasil antara.

Hasil akhir dibagi menjadi empat bagian:

cA (Kiri atas): Nilai rata-rata dari transformasi baris dan kolom, mewakili komponen frekuensi rendah (perkiraan).

cH (Kanan atas): Detail horizontal, menangkap perubahan di sepanjang baris.

cV (Kiri Bawah): Detail vertikal, menangkap perubahan di sepanjang kolom.

cD (Kanan bawah): Detail diagonal, menangkap perubahan frekuensi tinggi.

Hasil:

Approximation Coefficients (cA)



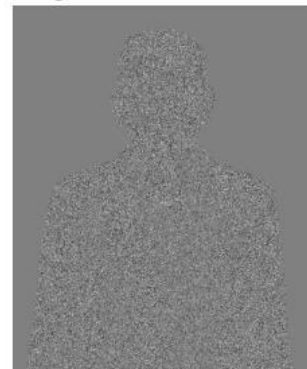
Vertical Coefficients (cV)



Horizontal Coefficients (cH)



Diagonal Coefficients (cD)



Transformasi Walsh Hadamard

```
1 import cv2
2 import numpy as np
3 from matplotlib import pyplot as plt
4 from scipy.linalg import hadamard
5
6 # Baca citra
7 img = cv2.imread('../kilat_8ok.jpg')
8
9 img_array = np.array(img)
10
11 plt.figure(figsize=(12, 6))
12
13 img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
14
15 img_resized = cv2.resize(img_gray, dsize=(128, 128))
16 H = hadamard(128)
17 wht_img = np.dot(np.dot(H, img_resized), H)
18
19 # Walsh Hadamard Image
20 plt.subplot(*args: 1, 2, 2)
21 plt.imshow(wht_img, cmap='gray')
22 plt.title('Walsh Hadamard Image')
23 plt.axis('off')
```

Transformasi walsh-adamard, Mirip dengan Fourier, namun bekerja dalam bidang binari dan digunakan untuk menganalisis sinyal diskrit. Lebih ringan dalam komputasi dan cocok untuk citra biner atau dengan sedikit variasi intensitas.

Hasil:

