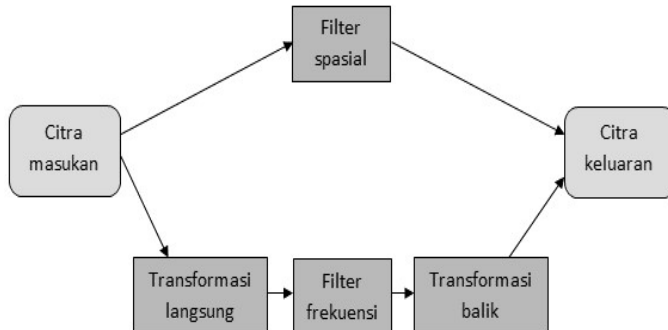


# MODUL TRANSFORMASI CITRA

## I. Pengolahan Citra di Kawasan Frekuensi

### 1. Transformasi Fourier

Transformasi Citra: Spasial vs Frekuensi



Spasial = berkaitan dengan posisi pixel

Frekuensi = berkaitan dengan kuantitas pixel

Transformasi Fourier dimanfaatkan untuk memetakan citra dari kawasan spasial ke dalam kawasan frekuensi. Citra dapat diamati sebagai kumpulan gelombang sinusoid dengan frekuensi, amplitudo, dan fase yang berbeda-beda.

### 2. Fourier 1-D (Discrete Fourier Transform)

$$F(u) = \frac{1}{N} \sum_{x=0}^{N-1} f(x) \left( \cos \left[ \frac{2\pi ux}{N} \right] - j \sin \left[ \frac{2\pi ux}{N} \right] \right)$$

atau

$$F(u) = \frac{1}{N} \sum_{x=0}^{N-1} f(x) \exp \left[ -j \frac{2\pi ux}{N} \right], \text{ dengan } u = 0, 1, 2, \dots, N-1$$

Invers Discrete Fourier Transform

$$f(x) = \sum_{x=0}^{N-1} F(u) \exp \left[ j \frac{2\pi ux}{N} \right], \text{ dengan } u = 0, 1, 2, \dots, N-1$$

Contoh Perhitungan:

f(x)		DFT	F(u)	
Real			Real	Imajiner
2	<div style="display: flex; align-items: center; justify-content: center;"><div style="margin-right: 10px;">→</div><div style="text-align: center;">DFT</div><div style="margin-left: 10px;">→</div></div> <div style="display: flex; align-items: center; justify-content: center;"><div style="margin-right: 10px;">←</div><div style="text-align: center;">DFT<sup>-1</sup></div><div style="margin-left: 10px;">←</div></div>	3	0	
4		0,25	0,25	
1		-1,50	0	
5		0,25	-0,25	

### Fourier 2-D

$$F(u, v) = \sum_{y=0}^{M-1} \sum_{x=0}^{N-1} f(x, y) \left( \cos \left( 2\pi \left( \frac{ux}{N} + \frac{vy}{M} \right) \right) - j \sin \left( 2\pi \left( \frac{ux}{N} + \frac{vy}{M} \right) \right) \right)$$

### Invers 2-D Fourier Transform

$$f(y, x) = \frac{1}{MN} \sum_{y=0}^{M-1} \sum_{x=0}^{N-1} F(v, u) \left( \cos \left( 2\pi \left( \frac{ux}{N} + \frac{vy}{M} \right) \right) + j \sin \left( 2\pi \left( \frac{ux}{N} + \frac{vy}{M} \right) \right) \right)$$

### 3. Fast Fourier Transform (FFT)

Metode Fast Fourier Transform (FFT) dibuat untuk mempercepat komputasi transformasi Fourier. Jika kompleksitas DFT untuk mentransformasikan sebuah piksel seperti yang tertuang dalam implementasi di depan sebesar  $O(N^2)$ , FFT memiliki kompleksitas sebesar  $O(N \log_2 N)$ .

Sebagai pembanding, jika  $N$  sama dengan 256 maka  $N^2 = 65.536$ , sedangkan  $N \log_2 N = 256 \times 8 = 2048$ . Jadi, FFT lebih cepat 32 kali dibandingkan DFT.

### J. Filter Lolos-Rendah (Low-pass Filter)

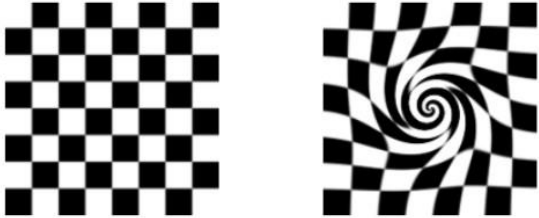
Filter lolos-bawah (low-pass filter) adalah filter yang mempunyai sifat dapat meloloskan yang berfrekuensi rendah dan menghilangkan yang berfrekuensi tinggi.

Efek filter ini membuat perubahan level keabuan menjadi lebih lembut. Filter ini berguna untuk menghaluskan

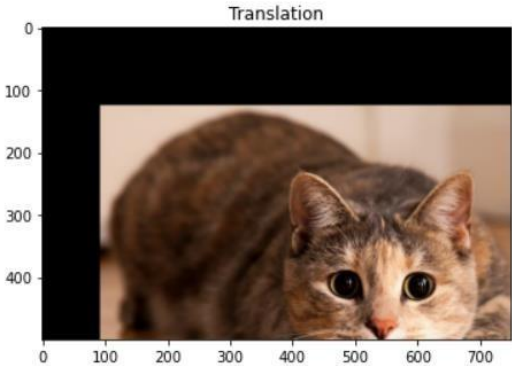
derau atau untuk kepentingan interpolasi tepi objek dalam citra.

## Latihan:

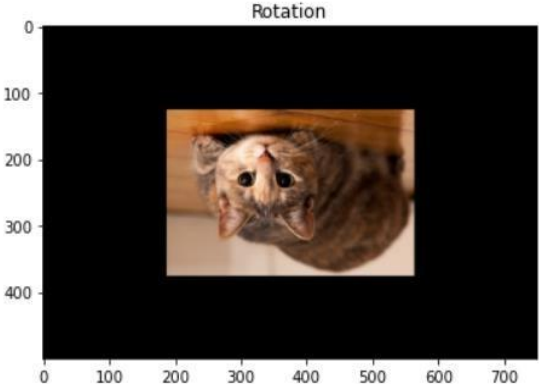
### 1. Transformasi Twirl

	<pre>import matplotlib.pyplot as plt  from skimage import data from skimage.transform import swirl  image = data.checkerboard() swirled = swirl(image, rotation=0, strength=10, radius=120)  fig, (ax0, ax1) = plt.subplots(nrows=1, ncols=2, figsize=(8, 3), sharex=True, sharey=True)  ax0.imshow(image, cmap=plt.cm.gray) ax0.axis('off') ax1.imshow(swirled, cmap=plt.cm.gray) ax1.axis('off') plt.show()</pre>
	

### 2. Translation

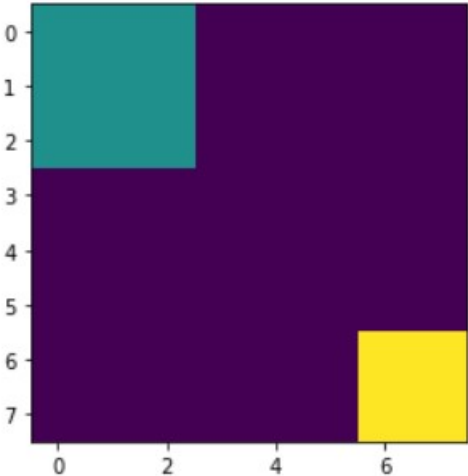
[in]	<pre> import cv2 as cv import numpy as np import matplotlib.pyplot as plt  image = cv.imread("kucing.jpg") h, w = image.shape[:2] half_height, half_width = h//4, w//8 transition_matrix = np.float32([[1, 0, half_width], [0, 1, half_height]]) img_transition = cv.warpAffine(image, transition_matrix, (w, h))  plt.imshow(cv.cvtColor(img_transition, cv.COLOR_BGR2RGB)) plt.title("Translation") plt.show() </pre>
[out]	 <p>The output is a plot titled "Translation". It displays a cat image that has been translated to the bottom-right corner of a black frame. The x-axis is labeled from 0 to 700 in increments of 100, and the y-axis is labeled from 0 to 400 in increments of 100. The cat image is positioned such that its top-left corner is at approximately (100, 100) in the plot's coordinate system.</p>

### 3. Rotation

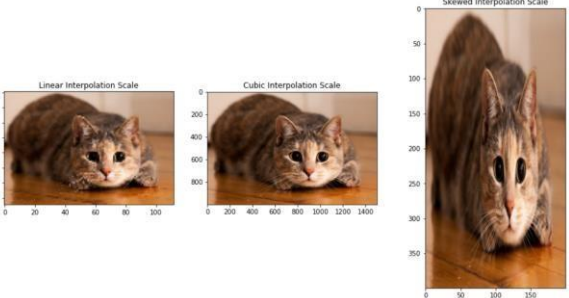
[in]	<pre> import cv2 as cv import numpy as np import matplotlib.pyplot as plt  image = cv.imread("kucing.jpg")  h, w = image.shape[:2]  rotation_matrix = cv.getRotationMatrix2D((w/2,h/2), -180, 0.5)  rotated_image = cv.warpAffine(image, rotation_matrix, (w, h))  plt.imshow(cv.cvtColor(rotated_image, cv.COLOR_BGR2RGB)) plt.title("Rotation") plt.show() </pre>
[out]	 <p>The output is a plot titled "Rotation". It displays a 180-degree rotated image of a cat, which appears upside down. The image is centered on a black background. The x-axis and y-axis are both labeled from 0 to 700, with major ticks every 100 units.</p>

#### 4. Interpolation

[in]	<pre> import mahotas as mh import numpy as np from pylab import imshow, show  regions = np.zeros((8,8), bool) regions[:3,:3] = 1 regions[6:,6:] = 1 labeled, nr_objects = mh.label(regions)  imshow(labeled, interpolation='nearest') show() </pre>
------	---

[out]	
[in]	<pre>labeled,nr_objects = mh.label(regions, np.ones((3,3), bool))</pre>
[in]	<pre>sizes = mh.labeled.labeled_size(labeled) print('Background size', sizes[0]) print('Size of first region: {}'.format(sizes[1]))</pre>
[out]	<pre>Background size 51 Size of first region: 9</pre>
[in]	<pre>array = np.random.random_sample(regions.shape) sums = mh.labeled_sum(array, labeled) print('Sum of first region: {}'.format(sums[1]))</pre>
[out]	<pre>Sum of first region: 5.760848502091579</pre>

5. Skala Interpolasi Miring

[in]	<pre> import cv2 as cv import numpy as np import matplotlib.pyplot as plt  image = cv.imread("kucing.jpg")  fig, ax = plt.subplots(1, 3, figsize=(16, 8))  # image size being 0.15 times of it's original size image_scaled = cv.resize(image, None, fx=0.15, fy=0.15) ax[0].imshow(cv.cvtColor(image_scaled, cv.COLOR_BGR2RGB)) ax[0].set_title("Linear Interpolation Scale")  # image size being 2 times of it's original size image_scaled_2 = cv.resize(image, None, fx=2, fy=2, interpolation=cv.INTER_CUBIC) ax[1].imshow(cv.cvtColor(image_scaled_2, cv.COLOR_BGR2RGB)) ax[1].set_title("Cubic Interpolation Scale")  # image size being 0.15 times of it's original size image_scaled_3 = cv.resize(image, (200, 400), interpolation=cv.INTER_AREA) ax[2].imshow(cv.cvtColor(image_scaled_3, cv.COLOR_BGR2RGB)) ax[2].set_title("Skewed Interpolation Scale") </pre>
[out]	<p>Text (0.5, 1.0, 'Skewed Interpolation Scale')</p>  <p>The output displays three subplots of a cat image, each with its own axes and title. The first subplot, titled 'Linear Interpolation Scale', shows a small, pixelated version of the cat. The second subplot, titled 'Cubic Interpolation Scale', shows a larger, smoother version of the cat. The third subplot, titled 'Skewed Interpolation Scale', shows a very large, distorted version of the cat, where the image is stretched and skewed, resulting in a loss of detail and a distorted shape.</p>

## TUGAS

- Lakukan proses seperti kode program di atas,  
Gunakan gambar yang ada di laptop masing-

masing menggunakan CV2 atau imageIo. display hasil boleh menggunakan cv2 dengan operator imshow atau imwrite atau menggunakan library matplotlib

- Bandingkan output masing masing

File yang dikumpulkan:

- Kode program ipynb/py
- File word berisi screenshoot, link github
- Format nama file: nim\_nama
- Agar drive LMS tidak penuh disarankan diupload di drive lalu attach link saat pengumpulan tugas. Pastikan link dapat diakses oleh semua orang.