

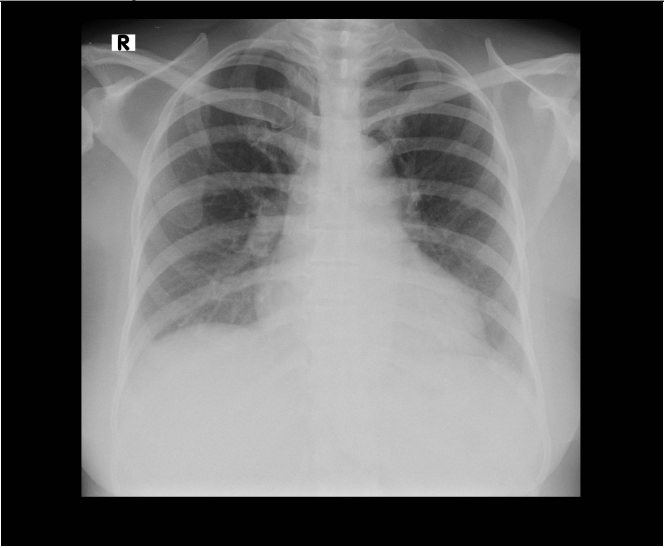
**TUGAS 4-ADVANCED IMAGE PROCESSING (T04)**

NIM : 14230018  
Nama : ILHAM MAULANA

**SOAL**

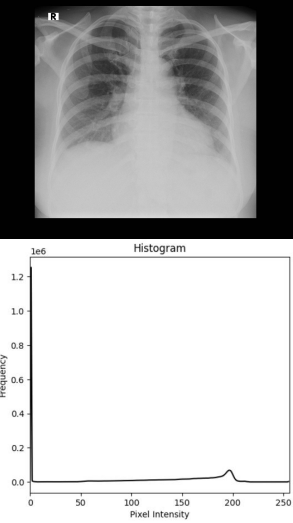
Buat coding dengan mengembangkan algoritma filter dan edge detection terhadap citra paru-paru dalam mengidentifikasi objek selain tulang rusuk, spt **Infiltrate** Rujukan: <http://telkomnika.uad.ac.id/index.php/TELKOMNIKA/article/view/3163/5264>

**Citra Input**

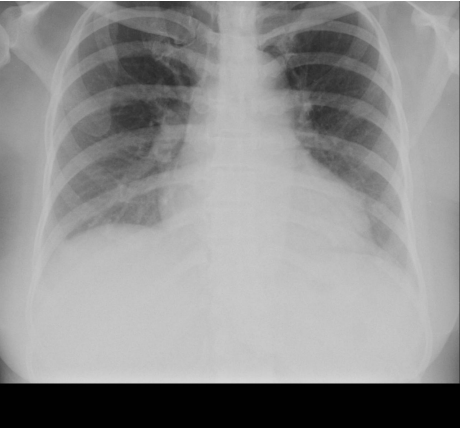
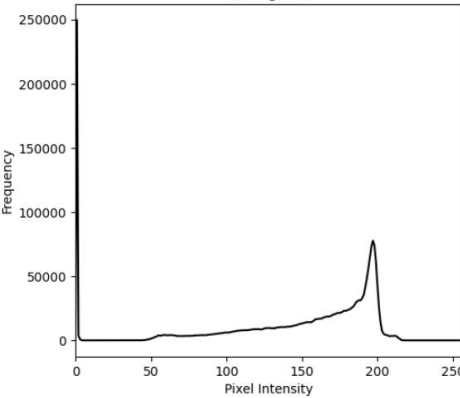


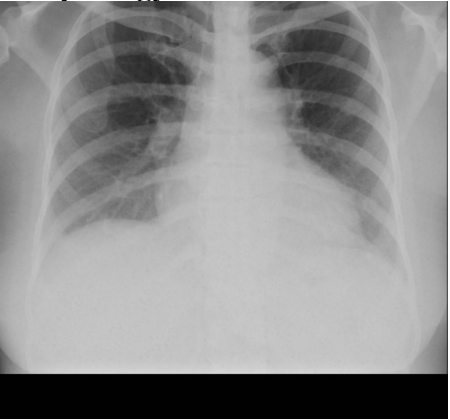
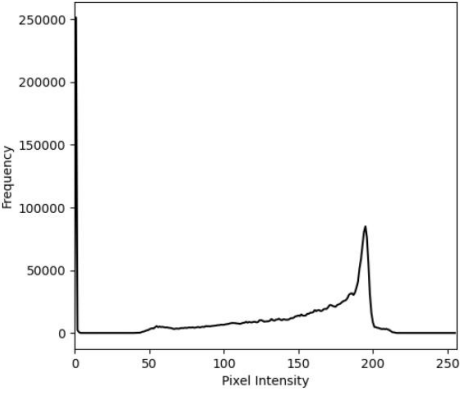
**Jawaban Pengolahan Citra**

Citra hasil pengolahan	Coding yang digunakan	Pseudocode	Discussion (penjelasan ttg hasil yang didapatkan)
	<pre>import cv2 import numpy as np import urllib import matplotlib.pyplot as plt  def display_img(img, title=None):     fig, ax = plt.subplots(1, 2, figsize=(12, 5))</pre>	<ol style="list-style-type: none"><li>1. Mengimport paket yang dibutuhkan</li><li>2. Membuat Prosedur <b>display_img</b></li><li>3. Buka URL Gambar</li><li>4. Konversi konten ke array byte</li></ol>	<p><b>Original Image</b> ditampilkan untuk melihat perbedaan sebelum melakukan image processing.</p>

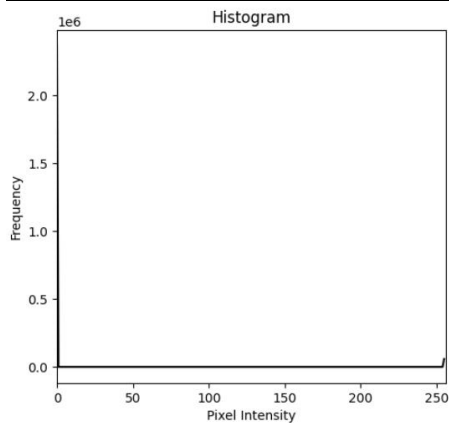
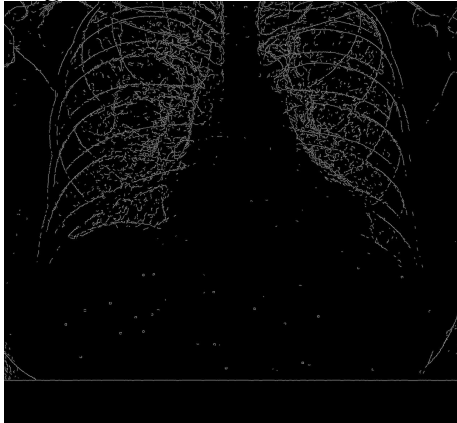
<p><b>Original Image</b></p> 	<pre>ax[0].imshow(img, cmap='gray') ax[0].set_title('Image') ax[0].axis('off')  hist = cv2.calcHist([img], [0], None, [256], [0, 256])  ax[1].plot(hist, color='black') ax[1].set_title('Histogram') ax[1].set_xlim([0, 256]) ax[1].set_xlabel('Pixel Intensity') ax[1].set_ylabel('Frequency')  if title:     plt.suptitle(title)  plt.show()  url = 'https://raw.githubusercontent.com/k4ilham/Detecti on-of-Infiltrate-on-Infant-Chest-X- Ray/main/14230018.jpg'  req = urllib.request.urlopen(url) arr = np.asarray(bytearray(req.read()), dtype=np.uint8) img = cv2.imdecode(arr, -1)  display_img(img, 'Original Image')</pre>	<p>dengan tipe data uint8</p> <p>5. Dekode array byte menjadi gambar menggunakan <code>cv2.imdecode</code></p> <p>6. Tampilkan gambar menggunakan <code>display_img</code> dengan judul "Original Image"</p>	
	<pre>cropped_image = img[260:1874,260:1874]  display_img(cropped_image, 'Cropped Image') cv2.imwrite("cropped_Image.jpg", cropped_image)</pre>	<p>1. <code>cropped_image</code> &lt;- potong gambar dari baris 260 hingga 1874 dan kolom 260 hingga 1874</p> <p>2. Tampilkan gambar menggunakan</p>	<p>Proses pertama adalah <code>cropping</code>. Proses ini bertujuan untuk memotong area yang tidak diperlukan dalam proses selanjutnya untuk mendapatkan</p>

<p><b>Cropping</b></p>  <p>Histogram</p> 		<p><code>display_img</code> dengan judul "Cropped Image"</p> <p>3. Simpan gambar sebagai "cropped_Image.jpg"</p>	<p>area infiltrasi. Teknik <b>cropping</b> mengambil area gambar masukan dari titik koordinat kiri atas hingga titik koordinat kanan bawah secara manual.</p> <p>Hasil <b>cropping</b> menunjukkan bahwa area yang tidak diperlukan untuk proses selanjutnya telah dihapus. Namun, gambar masih memiliki banyak noise. Noise adalah piksel yang mengganggu kualitas gambar dalam bentuk titik hitam atau putih yang tersebar secara acak. Untuk menghilangkan noise ini, maka akan dilakukan proses filter.</p>
	<pre>cropped_image = cv2.imread('cropped_Image.jpg')  gray = cv2.cvtColor(cropped_image, cv2.COLOR_BGR2GRAY)  filtered = cv2.GaussianBlur(gray, (3, 3), 0)  display_img(filtered, 'Filtered Image')  cv2.imwrite("filtered_image.jpg", filtered)</pre>	<p>1. Baca gambar "cropped_Image.jpg" menggunakan <code>cv2.imread</code></p> <p>2. Konversi gambar menjadi citra keabuan menggunakan <code>cv2.cvtColor</code> dengan parameter <code>cv2.COLOR_BGR2GRAY</code></p> <p>3. Lakukan filter Gaussian pada</p>	<p>Proses selanjutnya setelah melakukan cropping adalah <b>menghilangkan noise</b> pada hasil cropping gambar. Noise adalah nilai piksel yang mengganggu kualitas gambar. Nilai piksel noise bisa terlalu rendah atau terlalu tinggi di mana nilainya terdistribusi secara acak. Noise dapat dihilangkan</p>

<p><b>Gaussian Filter</b></p>  <p>Histogram</p> 		<p>citra keabuan menggunakan <code>cv2.GaussianBlur</code> dengan kernel 3x3 dan sigma 0</p> <p>4. Tampilkan gambar hasil filter menggunakan <code>display_img</code> dengan judul <code>"Filtered Image"</code></p> <p>5. Simpan gambar hasil penyaringan sebagai <code>"filtered_image.jpg"</code> menggunakan <code>cv2.imwrite</code></p>	<p>dengan melakukan proses filter. <b>Filter</b> adalah aktivitas untuk menghindari noise dalam proses penyesuaian nilai piksel pada tingkat nilai histogram.</p>
	<pre>kernel = np.ones((5, 5), np.uint8) erosion = cv2.erode(filtered, kernel, iterations=1) display_img(erosion, 'erosion Image') cv2.imwrite("erosion_image.jpg", erosion)</pre>	<p>1. <code>kernel</code> &lt;- buat kernel dengan ukuran 5x5 dan tipe data <code>uint8</code>, di mana semua elemen memiliki nilai 1</p> <p>2. <code>erosion</code> &lt;- lakukan operasi erosi pada gambar yang telah disaring menggunakan <code>cv2.erode</code></p>	<p>Hasil fiter gambar masih belum terlihat dengan jelas untuk deteksi tepi objek yang terdapat dalam sinar-X dada. Untuk membentuk deteksi tepi, proses pertama yang dilakukan adalah erosi morfologi. <b>Erosi morfologi</b> adalah meminimalkan total piksel yang membentuk deteksi tepi objek.</p>

<p><b>Morphology Erosion</b></p>  		<p>dengan kernel yang telah dibuat dan satu iterasi</p> <ol style="list-style-type: none"> <li>3. Tampilkan gambar hasil erosi menggunakan <code>display_img</code> dengan judul <code>"erosion Image"</code></li> <li>4. Simpan gambar hasil erosi sebagai <code>"erosion_image.jpg"</code> menggunakan <code>cv2.imwrite</code></li> </ol>	
	<pre>edges = cv2.Canny(erosion, 20, 30) display_img(edges, 'edges Image') cv2.imwrite("edges_image.jpg", edges)</pre>	<ol style="list-style-type: none"> <li>1. <code>edges &lt;-</code> gunakan metode <code>Canny</code> untuk mendeteksi tepi pada gambar hasil erosi dengan nilai threshold 20 dan 30</li> <li>2. Tampilkan gambar hasil deteksi tepi menggunakan <code>display_img</code> dengan judul <code>"edges Image"</code></li> <li>3. Simpan gambar hasil deteksi</li> </ol>	<p><code>Edge Detection</code> adalah proses pembentukan tepi suatu objek. Tahap ini mengurangi citra hasil penyaringan dengan citra hasil erosi morfologi.</p>

## Edge Detection



tepi sebagai "edges\_image.jpg"  
menggunakan cv2.imwrite

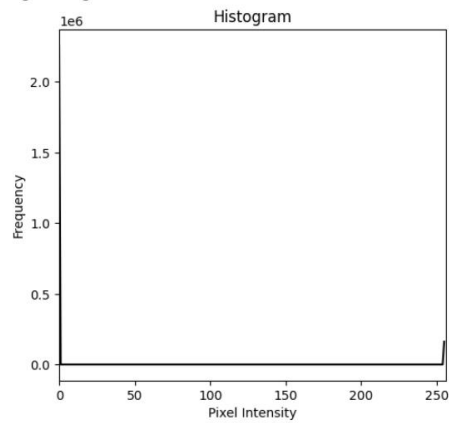
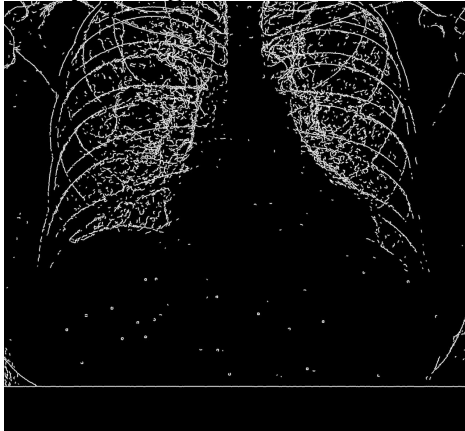
```
laplacian = cv2.Laplacian(edges, cv2.CV_64F)
sharpened_edges = cv2.convertScaleAbs(laplacian)
display_img(sharpened_edges, 'sharpened_edges  
Image')
cv2.imwrite("sharpened_edges_image.jpg",  
sharpened_edges)
```

1. `laplacian` <- Terapkan operator Laplacian pada gambar tepi menggunakan `cv2.Laplacian` dengan tipe data `CV_64F`
2. `sharpened_edges` <- Konversi gambar Laplacian menjadi gambar skala absolut menggunakan

Proses **Sharpen Edge** bertujuan untuk mempertajam tepi setiap objek dalam gambar. Nilai piksel tepi ingin ditingkatkan atau lebih tinggi dari piksel lainnya

Berdasarkan tahapan proses sinar-X dada citra medis di atas, dapat

## Sharpen Edge



`cv2.convertScaleAbs`

3. Tampilkan gambar tepi yang telah diperjelas menggunakan `display_img` dengan judul `"sharpened_edges Image"`
4. Simpan gambar tepi yang telah diperjelas sebagai `"sharpened_edges_image.jpg"` menggunakan `cv2.imwrite`

dihasilkan gambar yang lebih jelas.

Hasil pemrosesan gambar menampilkan deteksi tepi objek yang lebih jelas, sehingga informasi yang terkandung dalam gambar lebih mudah dikenali. Objek yang dapat diidentifikasi dengan jelas adalah infiltrat dan tulang dada. Ketelitian deteksi tepi objek akan membantu dokter dalam mengatasi keraguan dalam mengidentifikasi infiltrat pada sinar-X dada.

Link Source Code

<https://github.com/k4ilham/Detection-of-Infiltrate-on-Infant-Chest-X-Ray>