

Parallel Cellular Algorithm

Application : Image Edge Detection

Code:

```
import cv2

import numpy as np

from multiprocessing import Pool, cpu_count

from google.colab.patches import cv2_imshow # Import cv2_imshow for displaying images in Colab

# Function to apply Sobel operator to a small image chunk

def apply_sobel(chunk):

    # Sobel kernels

    sobel_x = np.array([[ -1,  0,  1], [-2,  0,  2], [ -1,  0,  1]])

    sobel_y = np.array([[ -1, -2, -1], [ 0,  0,  0], [ 1,  2,  1]])

    # Pad chunk to handle edge cases

    padded_chunk = np.pad(chunk, ((1, 1), (1, 1)), mode='constant')

    edge_chunk = np.zeros_like(chunk)

    # Apply Sobel operator

    for i in range(1, padded_chunk.shape[0] - 1):

        for j in range(1, padded_chunk.shape[1] - 1):

            region = padded_chunk[i-1:i+2, j-1:j+2]

            gx = np.sum(region * sobel_x)

            gy = np.sum(region * sobel_y)

            edge_chunk[i-1, j-1] = min(255, np.sqrt(gx**2 + gy**2)) # Gradient magnitude
```

```

return edge_chunk

# Function to split the image into chunks

def split_image(image, num_chunks):

    h, w = image.shape

    chunk_height = h // num_chunks

    # If image height is not divisible by num_chunks, ensure the last chunk gets the remaining rows
    chunks = [image[i * chunk_height:(i + 1) * chunk_height] for i in range(num_chunks - 1)]

    chunks.append(image[(num_chunks - 1) * chunk_height:]) # Add the last chunk with remaining rows

    return chunks

# Function to combine chunks back into a single image

def combine_chunks(chunks):

    return np.vstack(chunks)

# Main function to process the image

def parallel_edge_detection(image_path, num_workers=None):

    if num_workers is None:

        num_workers = cpu_count()

    # Load image in grayscale

    image = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)

    if image is None:

        raise FileNotFoundError(f"Image file not found: {image_path}")

    # Split the image into chunks for parallel processing

    chunks = split_image(image, num_workers)

    # Process each chunk in parallel

    with Pool(num_workers) as pool:

```

```
processed_chunks = pool.map(apply_sobel, chunks)

# Combine the processed chunks

edge_image = combine_chunks(processed_chunks)

return image, edge_image

# Example usage

if __name__ == "__main__":

    print("Chaitanya N 1BM22CS076")

    input_image_path = "/content/image.jpeg" # Replace with your image path

    output_image_path = "output_edge_detected.jpg"

    # Run edge detection

    original_image, edge_detected_image = parallel_edge_detection(input_image_path)

    # Save the edge-detected image

    cv2.imwrite(output_image_path, edge_detected_image)

    # Combine original and edge-detected images side by side

    combined_image = np.hstack((original_image,

    edge_detected_image)) # Display the combined image in Colab

    cv2.imshow(combined_image)

    print(f"Edge-detected image saved as: {output_image_path}")
```

Output :

