IMAGE PROCESSING USING GRAPH-BASED ALGORITHMS

Presented by:

Usman Rasool

Supervisor: ________________________________

Department of Computer Engineering

Table of Contents

1. PROBLEM DEFINITION:

Visual data analysis through image processing starts with identifying important structures throughout images as its first step. The two essential operations which enable this task include edge detection and image segmentation. The analysis becomes computationally difficult for complex or noisy images during these processing tasks.

This study makes use of graph theory, a mathematical framework that models pictures as graphs, to overcome these issues. Every pixel is regarded as a node, and the edges show connections between nearby pixels, such as variations in intensity. We can efficiently detect edges and separate pictures into relevant pieces by using graph-based techniques like Graph Cuts and the Minimum Spanning Tree (MST).

Digital image processing, edge detection, and image segmentation are key operations to understand visuals. They form a key part of different applications, including medical imaging, identifying objects, interpreting scenes, and watching over video recordings.

Traditional methods such as the **Sobel**, **Canny**, and **threshold-based segmentation** techniques often struggle in complex scenarios, such as images with:

- Low contrast between the object and the background
- Noise and lighting variability
- Irregular object boundaries

1.1.PROBLEM STATEMENT

How can graph theory be used instead of pixel-based methods to do reliable edge detection and accurate image segmentation?

This project proposes a novel approach using **graph-based algorithms**, specifically:

- **Minimum Spanning Tree (MST)** for identifying significant edge transitions in images.
- **Graph Cut (GrabCut)** for automatic background removal and object segmentation.

Additionally, a **graphical user interface (GUI)** is developed to:

- Allow user interaction (image upload and processing).

- ❖ Visually compare original, edge-detected, and segmented results.

1.2.OBJECTIVES

- ❖ Model images as graphs to capture pixel connectivity and similarity.
- ❖ Detect region boundaries using MST to highlight areas of high intensity change.
- ❖ Extract foreground objects using Graph Cut segmentation.
- ❖ Develop a user-friendly GUI to demonstrate graph theory applications in image analysis.

1.3.PURPOSE OF THE PROJECT:

This project aims to apply graph theory concepts to:

- ❖ Detect edges using the **MST algorithm** by identifying regions with high contrast or boundaries between objects.
- ❖ Segment images using **Graph Cuts**, which separate the image into foreground and background (or multiple regions) based on pixel similarity.

These approaches offer precise, flexible, and scalable solutions for image analysis problems.

1.4.APPLICATIONS:

These Graph-based edge detection and segmentation are widely used in various fields, including:

Field	**Application**
Medical Imaging	Identifying tumors, organs, and tissues from scans like MRIs and CTs
Autonomous Vehicles	Detecting road lanes, pedestrians, and obstacles
Security & Surveillance	Face detection, object tracking
Photo Editing & Graphics	Background removal, image enhancement
Agriculture	Analyzing satellite images for crop health or land use
Robotics & AI	Vision systems for navigation and object recognition

Industrial Inspection	Identifying defects in manufacturing processes

2. ALGORITHM USED TO SOLVE THE PROBLEM:

This project applies two advanced graph-based algorithms, each suited to a specific image processing task.

2.1. MINIMUM SPANNING TREE (MST) FOR EDGE DETECTION

2.1.1. TECHNIQUE:

Kruskal's Algorithm applied on an image-modeled graph.

2.1.2. CONCEPT:

- An image is modeled as a graph:
    - Each pixel = node
    - Connections to neighboring pixels (4 or 8 directions) = edges
    - Edge weight = difference in pixel intensity
- The **Minimum Spanning Tree (MST)** of this graph is computed using **Kruskal's Algorithm**, which finds the minimum set of edges that connect all nodes without cycles and with minimal total weight.

2.1.3. STEPS:

1. Convert the image to **grayscale**.
2. Create a graph where each pixel is connected to its neighbors.
3. Assign edge weights based on intensity differences.
4. Apply **Kruskal's algorithm** to form the MST.
5. Identify and highlight edges with **high weights**, which correspond to significant transitions—i.e., **edges** in the image.

2.2. GRAPH CUT (GRABCUT) FOR IMAGE SEGMENTATION

2.2.1. TECHNIQUE:

GrabCut – an iterative algorithm based on **Max-Flow / Min-Cut** graph partitioning.

2.2.2. CONCEPT:

- ❖ The image is treated as a **flow network**:
    - Pixels = **nodes**
    - Edges between pixels reflect **color similarity**
    - Pixels are connected to **foreground (source)** and **background (sink)** based on initial probability models
- ❖ The algorithm computes the **optimal binary partition** (foreground vs. background) by finding the **minimum cut** that separates the graph into two parts with minimum cost.

2.2.3. STEPS:

1. Input image is loaded in color (RGB).
2. A **bounding rectangle** is drawn to provide an initial guess of the foreground.
3. Construct a graph where:
    - Edges between pixels are weighted by color similarity (stronger = more similar).
    - Pixels are linked to source/sink based on probability of being object/background.
4. Run **Max-Flow / Min-Cut algorithm** to find the best separation.
5. Result: only **foreground object** remains, background is removed.

3. <u>REQUIREMENT LIST:</u>

3.1. FUNCTIONAL REQUIREMENTS:

- ❖ Load and display an input image.
- ❖ Represent the image as a graph (pixels as nodes, intensity difference as edge weights).
- ❖ Apply **Minimum Spanning Tree (MST)** to detect edges.
- ❖ Apply **Graph Cuts** to perform image segmentation.
- ❖ Visualize:
    - Original image.

    - Edge-detected output.
    - Segmented image.

- Demonstrate the graph theory logic behind each algorithm.

3.2. NON-FUNCTIONAL REQUIREMENTS:

- User-friendly output visualization (clear, labeled comparisons).
- Efficient execution time for small to medium-sized images.
- Maintainable and well-documented code.

3.3. SOFTWARE REQUIREMENTS:

- **Operating System:** Windows 11.
- **IDE:** Visual Studio Code (VSCode).
- **Python Version**: Preferably 3.8 or above.

3.4. HARDWARE REQUIREMENTS:

- A computer with:
    - Minimum 4 GB RAM (8 GB recommended).
    - Windows 11 Operating System (or compatible).
    - Processor: Intel i5 or equivalent and above.
    - Sufficient disk space for Python libraries and image datasets.

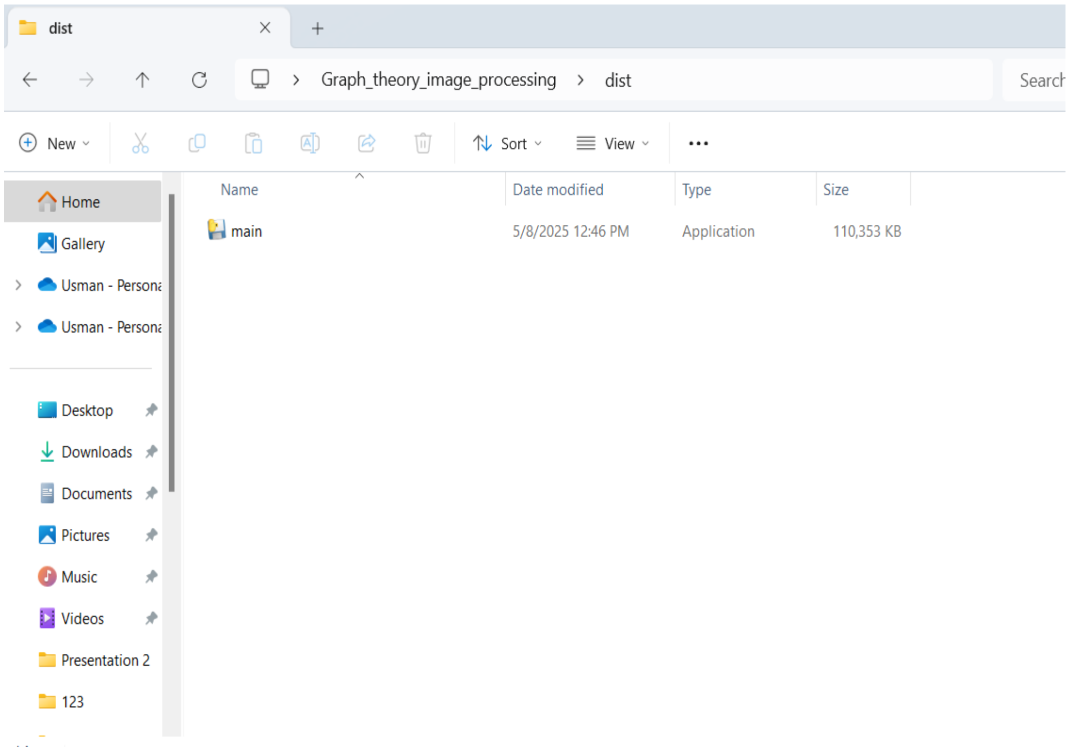
3.5. LIBRARIES AND FRAMEWORKS:

- **Python Libraries:**
    - **OpenCV:** Image handling and processing.
    - **NetworkX:** Graph creation and graph algorithms.
    - **NumPy**: Numerical operations and image matrix handling.
    - **Matplotlib**: Visualization of results.
    - **TKinter**: Create Graphical User Interfaces (GUIs) easily and efficiently.

4. OUTPUT SCREEN OF THE PROGRAM:

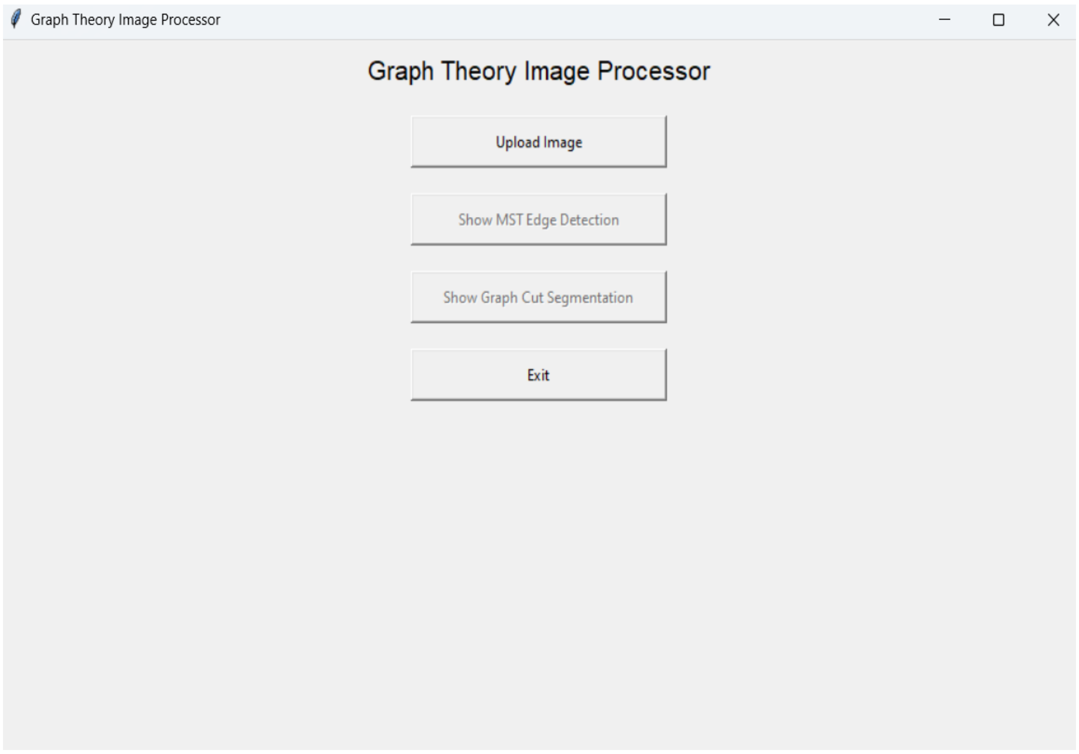
4.1. STEPS:

- Click on the executable file.

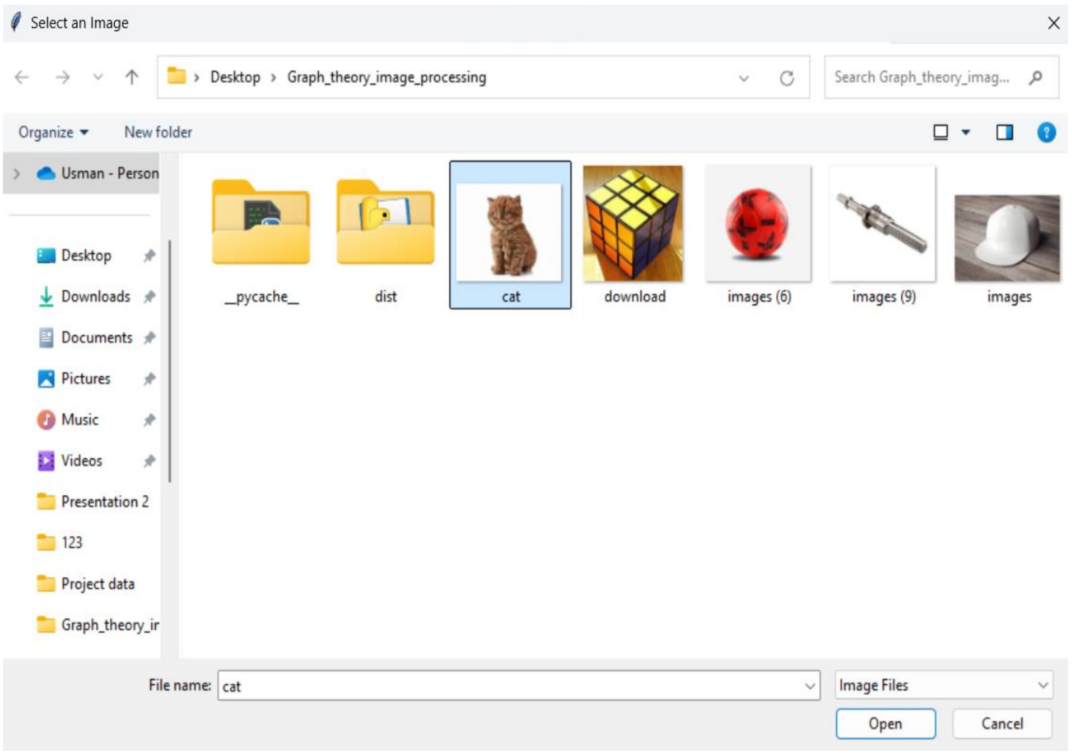


Name	Date modified	Type	Size
main	5/8/2025 12:46 PM	Application	110,353 KB

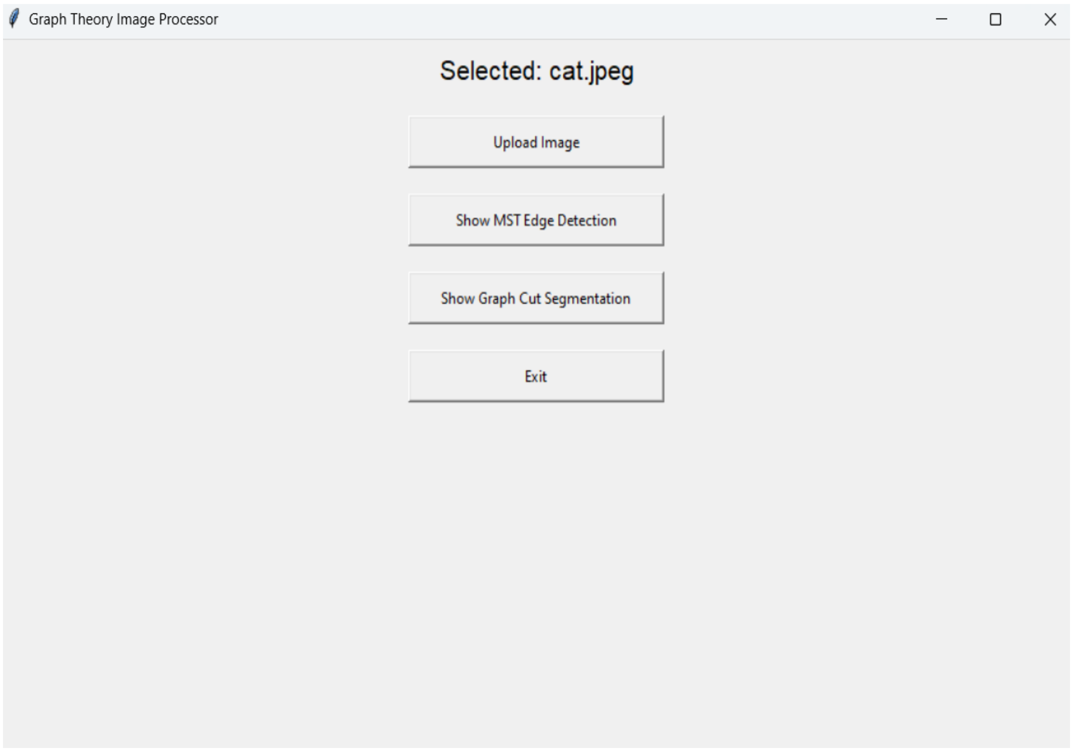
❖ Click on the upload image button.


Graph Theory Image Processor
Graph Theory Image Processor
Upload Image
Show MST Edge Detection
Show Graph Cut Segmentation
Exit

- Select image.

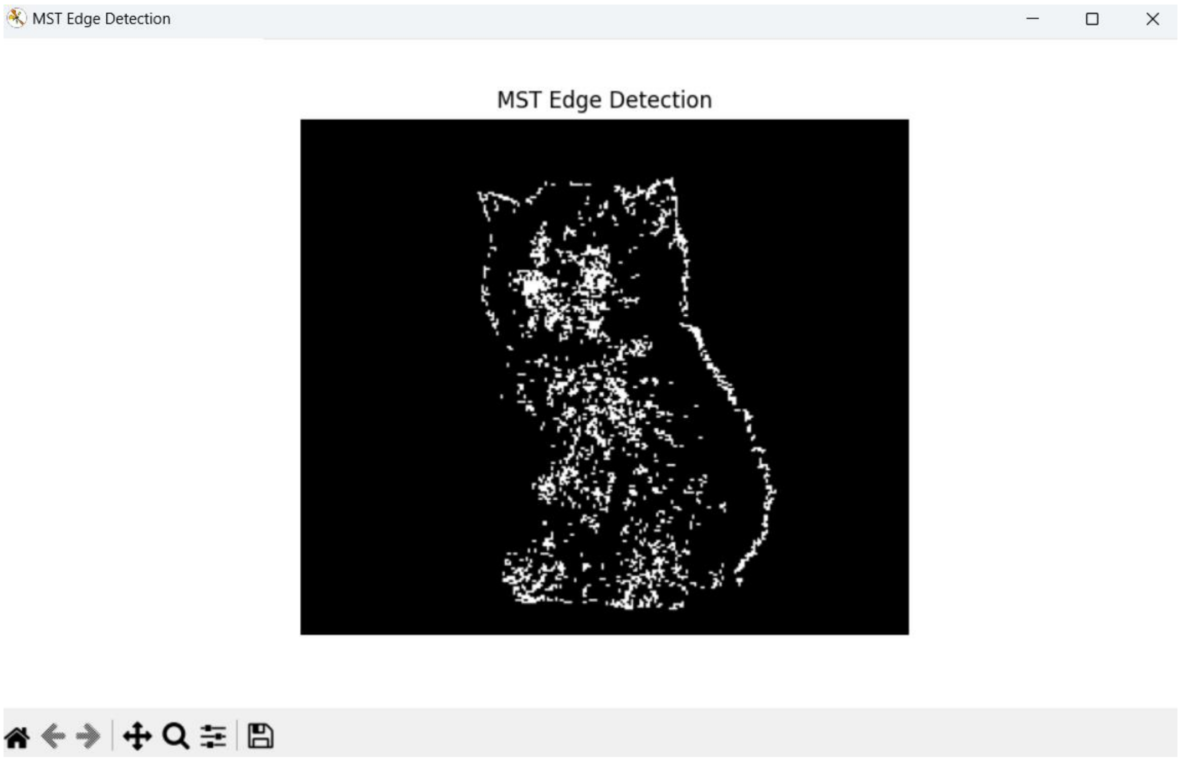


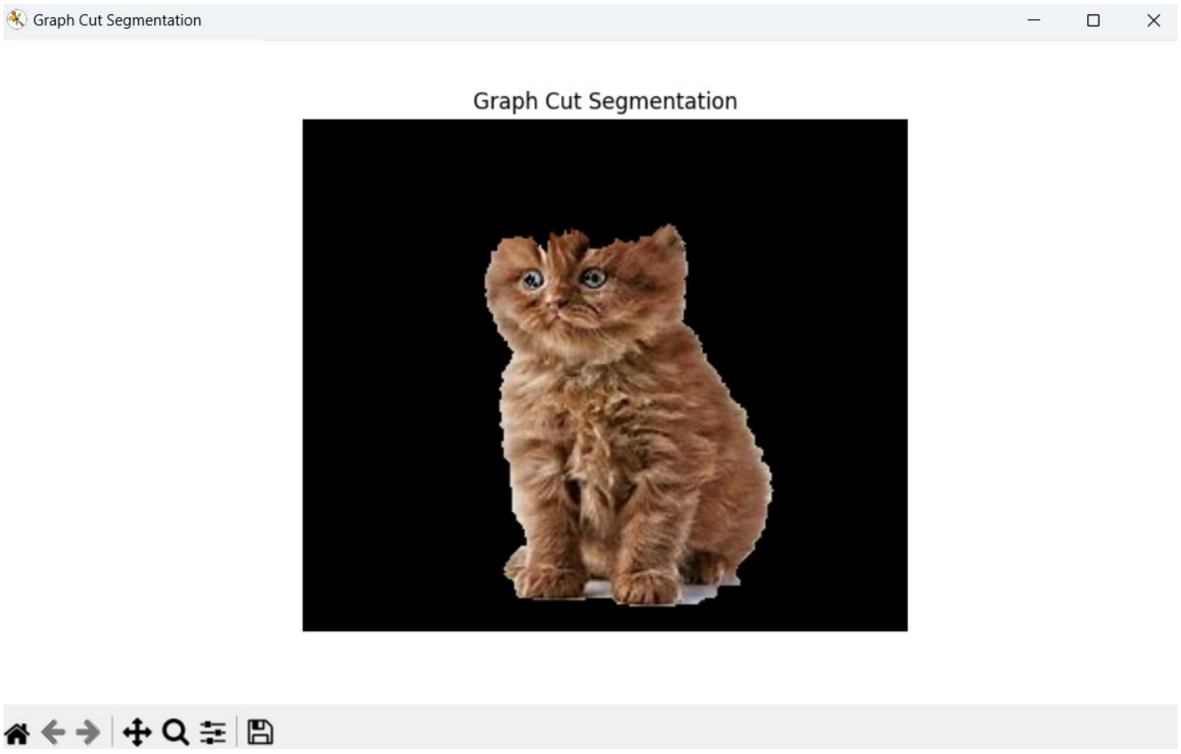
- Image uploaded.

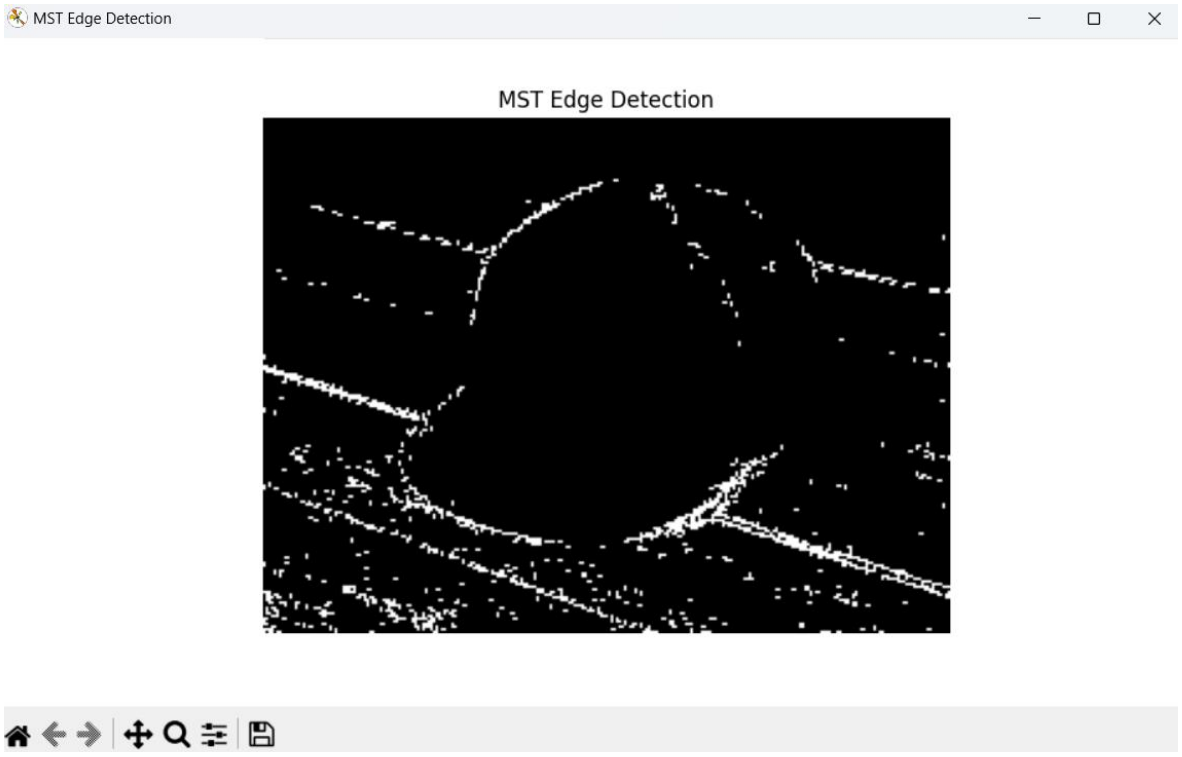


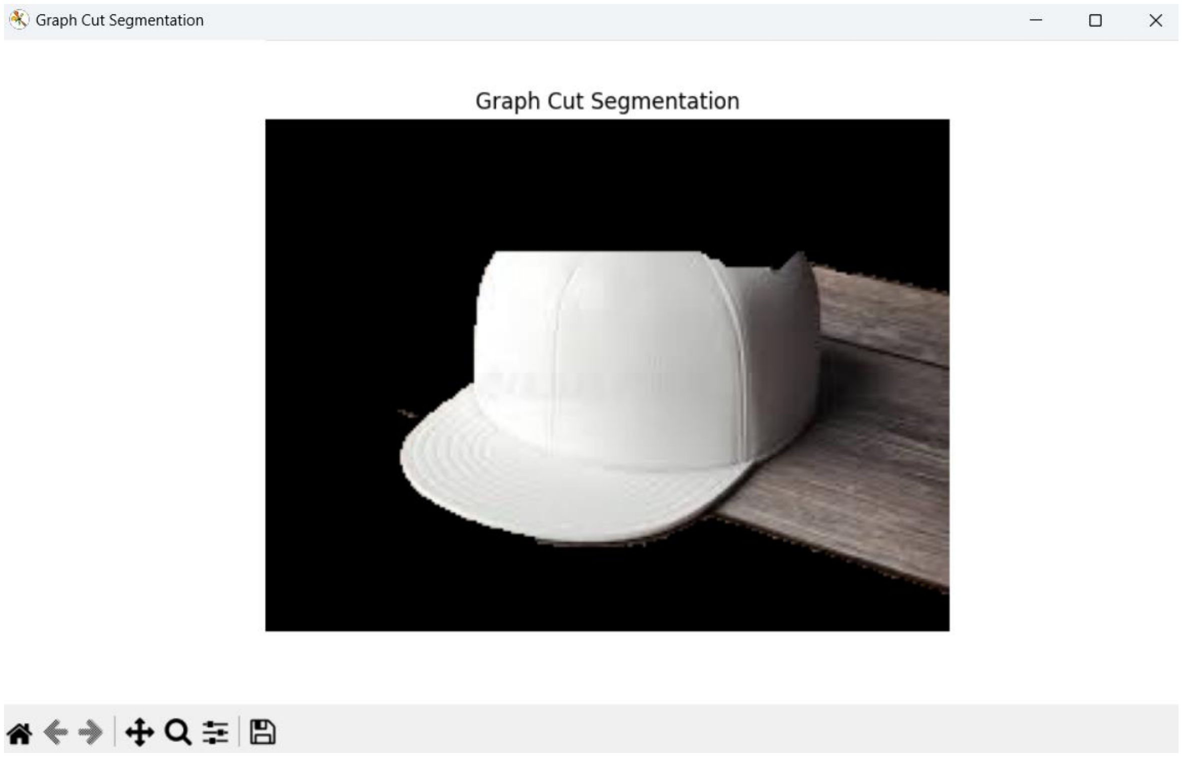
Selected: cat.jpeg

4.2. OUTPUTS.


MST Edge Detection
MST Edge Detection


Graph Cut Segmentation
Graph Cut Segmentation


MST Edge Detection
MST Edge Detection


Graph Cut Segmentation
Graph Cut Segmentation