```
import cv2
import numpy as np
from scipy.spatial import distance
import networkx as nx
import matplotlib.pyplot as plt

# Load the Chandrayaan image
img_path = 'chandrayaan_image.jpg'
img = cv2.imread(img_path)
if img is None:
    raise FileNotFoundError(f"Image file '{img_path}' not found.")

# Apply image preprocessing techniques
img = cv2.GaussianBlur(img, (5, 5), 0)
img = cv2.normalize(img, None, 0, 255, cv2.NORM_MINMAX)

# Convert the image to grayscale
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# Detect craters using edge detection and thresholding
edges = cv2.Canny(gray, 50, 150)
contours, _ = cv2.findContours(edges, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)

# Extract features from the detected craters
crater_features = []
for contour in contours:
    x, y, w, h = cv2.boundingRect(contour)
    aspect_ratio = float(w) / h
    crater_features.append((x, y, w, h, aspect_ratio))

# Filter out small craters
min_crater_size = 10
crater_features = [feature for feature in crater_features if feature[2] > min_crater_size]

# Create a graph representation of the lunar surface
G = nx.Graph()
for i, feature in enumerate(crater_features):
    G.add_node(i, pos=(feature[0], feature[1]))

# Add edges between nodes based on proximity
max_distance = 100
for i in range(len(crater_features)):
    for j in range(i + 1, len(crater_features)):
        distance_ij = distance.euclidean(crater_features[i][:2], crater_features[j][:2])
        if distance_ij < max_distance:
            G.add_edge(i, j, weight=distance_ij)

# Define the start and end points for the navigation route
```

```
if len(crater_features) < 2:
    raise ValueError("Not enough craters detected to determine a path.")
start_node = 0
end_node = len(crater_features) - 1

# Use Dijkstra's algorithm to find the shortest path
try:
    path = nx.shortest_path(G, source=start_node, target=end_node, weight='weight')
except nx.NetworkXNoPath:
    raise ValueError("No path found between the start and end nodes.")

# Visualize the navigation route
plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
plt.scatter([crater_features[node][0] for node in path], [crater_features[node][1] for node in
path], c='yellow', s=40)
plt.plot([crater_features[node][0] for node in path], [crater_features[node][1] for node in path],
'r-')
plt.title("Shortest Path Between Craters")
plt.show()
```