Team Name: Cosmo Coders
Name of College(s)/University(s): Kalinga Institute of Industrial Technology

Team Members Details:

1. Ankit Mahapatra
2. Jyotika Jayani
3. Tanisha Basu
4. Pradipto

Approach Brief (50 words)

Develop software to handle different spatial resolutions between Chandrayaan-2 TMC and LRO WAC images. Implement template matching to locate craters in the mosaic. Use efficient data storage for quick search and visualization. Ensure accurate latitude and longitude extraction and crater identification with high correlation.

Detailed solution and Approach (250-300 words)

To find the location of a crater from a lunar mosaic, start by downloading the LRO WAC global mosaic (100m resolution) from the USGS website and the higher resolution Chandrayaan-2 TMC nadir images (5m resolution). Use OpenCV to preprocess both image sets, adjusting for resolution differences through scaling and feature enhancement. Implement template matching algorithms, such as cross-correlation, to identify crater images from Chandrayaan-2 TMC within the LRO WAC mosaic, ensuring the algorithm handles variations in lighting, shadows, and scale. Upon a successful match, extract the crater's coordinates by mapping the pixel coordinates from the template match to geographical coordinates using the mosaic's metadata. To enhance efficiency, develop a data storage strategy utilizing indexing techniques or spatial databases for quicker retrieval and search within the LRO WAC mosaic. Use QGIS for visualization, enabling a clear representation of the identified crater locations. Validate the accuracy of the extracted coordinates against known locations, ensuring a correlation value of ≥0.70 for reliable template matching. This approach results in software capable of handling resolution disparities, performing accurate template matching, and extracting precise latitude and longitude coordinates of craters from the LRO WAC mosaic using Chandrayaan-2 TMC images.

Tools and Technology Used (50 words)

- OpenCV : For image preprocessing and template matching
- QGIS : For visualization of crater locations
- Spatial Databases : For efficient data storage and retrieval
- GeoTIFF Format : For handling the LRO WAC global mosaic
- Python : For implementing algorithms and managing data processing

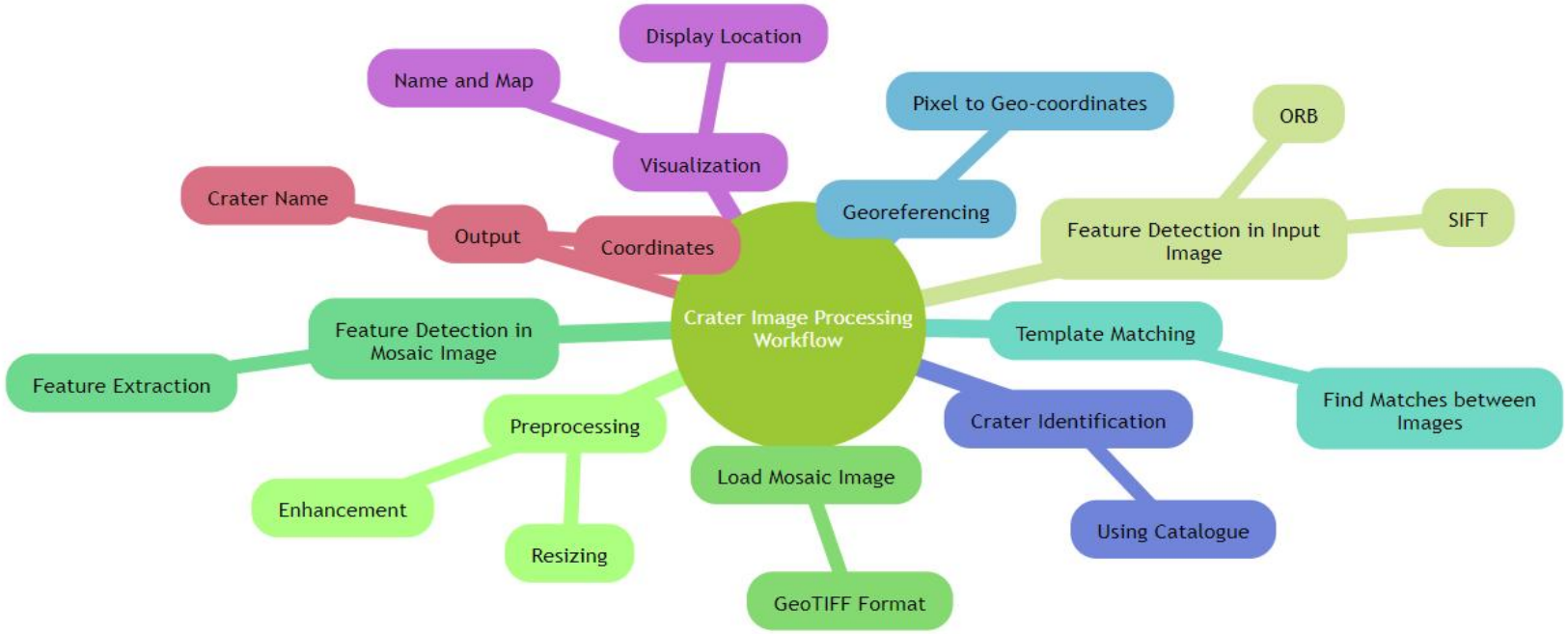
Datasets used : - lunar dataset

To view Project :-

GitHub Repository : - https://github.com/JyotikaJayani-08/ISRO-Hackathon

Proposed architecture/user diagram

figures 2.1: mindmap

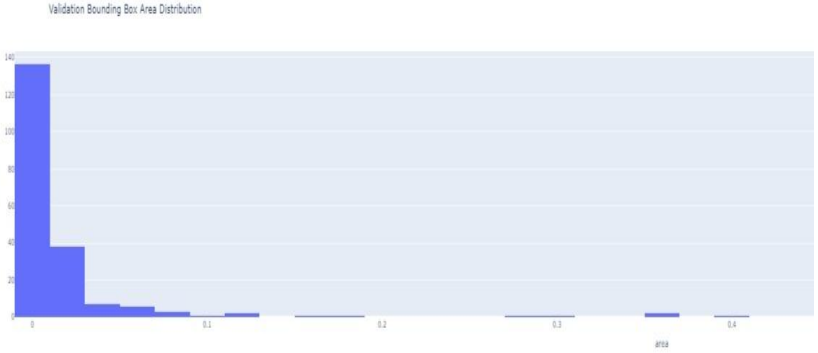


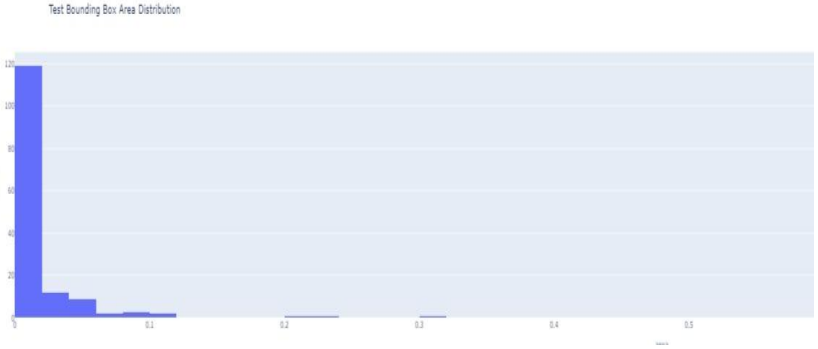
Solution Brief (Overall)

To locate a crater on the lunar surface using images from Chandrayaan-2 TMC and the LRO WAC mosaic, start by downloading the necessary datasets: the LRO WAC global mosaic (100m resolution) from the USGS website and Chandrayaan-2 TMC nadir images (5m resolution). Use OpenCV to preprocess both image sets, adjusting for differences in resolution through scaling and feature enhancement. Implement template matching algorithms, such as cross-correlation, to identify crater images from Chandrayaan-2 TMC within the LRO WAC mosaic. Ensure the algorithm accommodates variations in lighting, shadows, and scale. After a successful template match, map the pixel coordinates to geographical coordinates using the mosaic's metadata to extract the crater's latitude and longitude. Develop a data storage strategy, utilizing spatial databases for efficient management and quick retrieval of the LRO WAC mosaic. Visualize the identified crater locations using QGIS and validate the accuracy of the coordinates against known locations, ensuring a template matching correlation value of ≥0.70. This solution will produce software capable of accurate crater identification and localization on the lunar surface.

In [9]:

```python
# Image Preprocessing and Visualization
def visualize_sample_images(image_path, label_df, n_samples=5):
    image_files = os.listdir(image_path)[:n_samples]
    for img_file in image_files:
        img_path = os.path.join(image_path, img_file)
        img = cv2.imread(img_path)
        img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

        fig, ax = plt.subplots(1, 1, figsize=(10, 10))
        ax.imshow(img)

        labels = label_df[label_df['file'] == img_file]
        for _, label in labels.iterrows():
            x_center = int(label['x_center'] * img.shape[1])
            y_center = int(label['y_center'] * img.shape[0])
            width = int(label['width'] * img.shape[1])
            height = int(label['height'] * img.shape[0])
            x_min = x_center - width // 2
            y_min = y_center - height // 2

            rect = plt.Rectangle((x_min, y_min), width, height, edgecolor='red', facecolor='none', line
            ax.add_patch(rect)

        plt.title(f'Sample Image: {img_file}')
        plt.axis('off')
        plt.show()

visualize_sample_images(train_img_path, train_labels)
visualize_sample_images(valid_img_path, valid_labels)
visualize_sample_images(test_img_path, test_labels)
```






```
      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      1/50         0G      1.752      3.365      1.415         30        640: 100%|          | 7/7 [01:23<
00:00, 11.97s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|          |
1/1 [00:07<00:00,  7.84s/it]
                   all         26        202     0.0174      0.673     0.0897     0.0394

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      2/50         0G      1.505      2.391      1.249         19        640: 100%|          | 7/7 [01:15<
00:00, 10.76s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|          |
1/1 [00:07<00:00,  7.49s/it]
                   all         26        202     0.0205      0.792      0.216      0.125

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      3/50         0G      1.483      1.743      1.179         20        640: 100%|          | 7/7 [01:17<
00:00, 11.03s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|          |
1/1 [00:07<00:00,  7.52s/it]
                   all         26        202     0.0213      0.822      0.219      0.123

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      4/50         0G      1.462      1.676      1.141         10        640: 100%|          | 7/7 [01:17<
00:00, 11.13s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|          |
1/1 [00:07<00:00,  7.59s/it]
                   all         26        202     0.0224      0.866      0.255      0.156

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      5/50         0G      1.528      1.745      1.226          6        640: 100%|          | 7/7 [01:17<
00:00, 11.12s/it]
```

Conclusion

By combining high-resolution images from Chandrayaan-2 TMC and the comprehensive LRO WAC global mosaic, this project aims to accurately locate lunar craters and extract their geographical coordinates. Through preprocessing, template matching using OpenCV, and efficient data management with spatial databases, the solution ensures precise crater identification despite resolution differences. Visualization using QGIS and rigorous validation processes guarantee reliable results, making this software a valuable tool for lunar exploration and research. This approach not only bridges the gap between different spatial resolutions but also enhances the capability to map and study lunar surface features accurately.


BHARATIYA ANTARIKSH HACKATHON 2024
Powered by H2S
इसरो isro
THANK YOU