Satellite Image Compositing with Feathering and Color Adjustment

Submitted by :

Chaza Dally, Jinane Al Ammar, Youssef Dawoud

Department of Computer Science

Lebanese university - section1 - Hadath

April 2025

Outline

- Introduction
- Objectives
- Methodology
  - Image Preprocessing
  - Feathering-Based Blending
  - Pyramid Blending for Seamless Merging
  - Edge Smoothness and Color Adjustment
  - Implementation Steps
- Implementation
  - Tools and libraries used
- Result and evaluation
  - Performance metrics
  - Comparison with others methods
- References
- Conclusion

1. Introduction

Satellite imagery has become an essential tool for various applications, including environmental monitoring, urban planning, agriculture, and disaster management. The ability to analyze and interpret satellite images allows researchers and decision-makers to track changes in land use, detect deforestation, monitor climate patterns, and assess damage after natural disasters. However, satellite images are often captured under different conditions, leading to inconsistencies in lighting, color balance, resolution, and perspective. These inconsistencies make it difficult to merge multiple satellite images into a seamless composite, which is necessary for large-scale analysis and visualization.

This project aims to address these challenges by developing an automated pipeline for satellite image compositing. The approach involves advanced image processing techniques such as feathering, histogram matching, and pyramid blending to ensure smooth transitions between overlapping images. By leveraging these techniques, the project produces high-quality composite images that maintain color consistency, minimize visual artifacts, and preserve geographic details.

The project is designed to integrate multiple satellite images efficiently while addressing common issues such as:

- **Color Inconsistencies:** Differences in sensor calibration and atmospheric conditions result in color variations between images.
- **Sharp Seams at Boundaries:** Direct stitching of images often leads to noticeable seams and harsh transitions.
- **Geometric Distortions:** Variations in perspective and terrain can create alignment challenges.
- **Loss of Image Details:** Traditional blending methods may cause a loss of important details in overlapping regions.

By applying a structured methodology that includes preprocessing, blending, and post-processing techniques, this project ensures high-quality compositing of satellite images. The resulting composite images are suitable for use in geographic information systems (GIS), land use analysis, and other remote sensing applications. Additionally, the pipeline is implemented using Python and OpenCV, making it accessible and adaptable for further enhancements.

The following sections outline the objectives, methodology, implementation details, and evaluation of the project. By the end of this study, we demonstrate how

advanced image processing techniques can significantly improve the quality and usability of satellite image composites.

2. Objectives

The main goals of this project are:

- To develop an automated pipeline for blending multiple satellite images into a unified composite.
- To apply histogram matching for color consistency across images.
- To use Gaussian and Laplacian pyramid blending for seamless transitions.
- To enhance the final composite with edge smoothness and contrast adjustments.
- To evaluate the quality of blending using edge smoothness analysis.

3. Methodology

The project follows a structured approach that integrates multiple image processing techniques. The main steps are:

3.1 Image Preprocessing

- The script loads images from a specified directory.
- Images are resized to ensure uniform dimensions before processing.
- A histogram matching algorithm is applied to adjust colors and make them consistent across different images.

```python
import cv2
import numpy as np

def equalize_size(img1, img2):
    height = min(img1.shape[0], img2.shape[0])
    width = min(img1.shape[1], img2.shape[1])
    img1_resized = cv2.resize(img1, (width, height))
    img2_resized = cv2.resize(img2, (width, height))
    return img1_resized, img2_resized
```

3.2 Feathering-Based Blending

- A smooth blending mask is created based on the direction (horizontal or vertical).
- A smooth gradient transition is applied to the overlap regions to ensure a soft blend.
- This method prevents sharp edges and visible seams between images.

```
def create_blend_mask(img_shape, blend_width, direction='horizo
    mask = np.zeros(img_shape, dtype=np.float32)
    if direction == 'horizontal':
        mask[:, :blend_width] = np.linspace(0, 1, blend_width)
        mask[:, -blend_width:] = np.linspace(1, 0, blend_width)
    else:
        mask[:blend_width, :] = np.linspace(0, 1, blend_width)[
        mask[-blend_width:, :] = np.linspace(1, 0, blend_width)
    return mask
```

3.3 Pyramid Blending for Seamless Merging

- The images and masks are decomposed into Gaussian pyramids (low-frequency details).
- Laplacian pyramids (high-frequency details) are computed for each image.
- The corresponding levels of the pyramids are blended using a weighted mask.
- The final composite is reconstructed by upsampling and summing the pyramid levels.

```python
def pyramid_blend(img1, img2, mask, levels=5):
    gp1, gp2, gm = [img1], [img2], [mask]
    for i in range(levels):
        gp1.append(cv2.pyrDown(gp1[-1]))
        gp2.append(cv2.pyrDown(gp2[-1]))
        gm.append(cv2.pyrDown(gm[-1]))
    lp1, lp2 = [gp1[-1]], [gp2[-1]]
    for i in range(levels-1, 0, -1):
        lp1.append(cv2.subtract(gp1[i-1], cv2.pyrUp(gp1[i])))
        lp2.append(cv2.subtract(gp2[i-1], cv2.pyrUp(gp2[i])))
    lp_blended = [l1 * gm[i] + l2 * (1 - gm[i]) for i, (l1, l2)
    blended = lp_blended[0]
    for i in range(1, levels):
        blended = cv2.pyrUp(blended) + lp_blended[i]
    return blended
```

3.4 Edge Smoothness and Color Adjustment

- The composited image is analyzed for edge smoothness using the Laplacian variance method.
- Automatic color balance is applied using CLAHE (Contrast Limited Adaptive Histogram Equalization) to enhance the visual quality.

```python
def edge_smoothness_score(image):
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    laplacian = cv2.Laplacian(gray, cv2.CV_64F)
    variance = laplacian.var()
    return variance

# Function to apply CLAHE for color adjustment
def apply_clahe(image):
    lab = cv2.cvtColor(image, cv2.COLOR_BGR2LAB)
    l, a, b = cv2.split(lab)

    # Apply CLAHE to the L channel
    clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8,8))
    l_clahe = clahe.apply(l)

    # Merge back and convert to BGR
    lab_clahe = cv2.merge((l_clahe, a, b))
    adjusted_img = cv2.cvtColor(lab_clahe, cv2.COLOR_LAB2BGR)
```

3.5 Implementation Steps

1. **Image Loading & Preprocessing**: Convert images into a uniform format and size.
2. **Histogram Matching**: Adjust color histograms to ensure consistency.
3. **Mask Generation**: Create smooth blending masks.
4. **Pyramid Blending**: Decompose images into Gaussian and Laplacian pyramids.
5. **Edge Smoothness Calculation**: Evaluate blending effectiveness.
6. **Final Image Enhancement**: Apply contrast adjustments for better clarity.

4. Implementation

The project is implemented using Python and OpenCV. Key functionalities include:

- **equalize_size(img1, img2)**: Ensures both images have the same dimensions.
- **match_histograms(source, target)**: Matches the color distribution of one image to another.
- **create_blend_mask(img_shape, blend_width, direction)**: Generates a smooth mask for blending.
- **pyramid_blend(img1, img2, mask, levels=5)**: Performs Gaussian and Laplacian pyramid blending.
- **blend_images_sequential(images, blend_width, direction)**: Iteratively blends multiple images into a composite.
- **calculate_edge_smoothness(img)**: Evaluates edge consistency before and after blending.

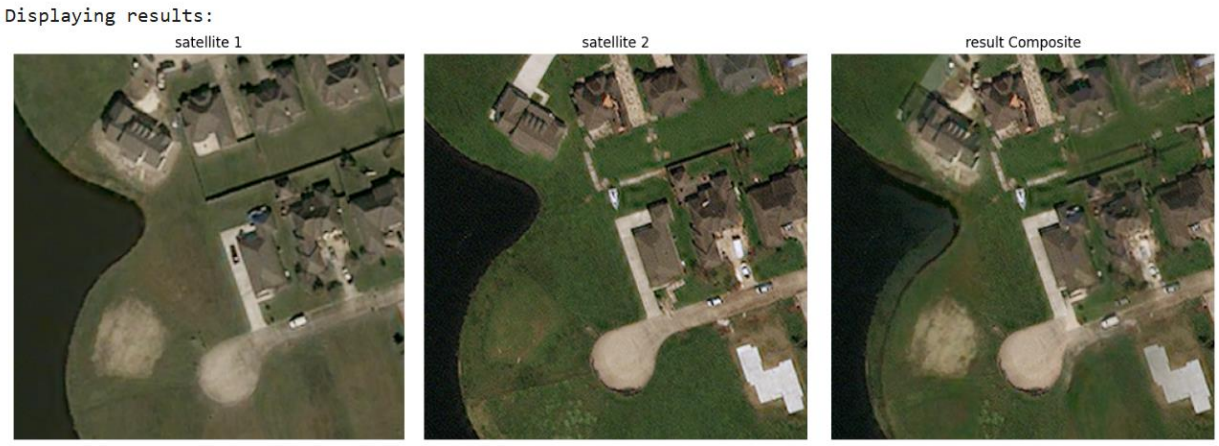
4.1 Tools & Libraries Used

- **OpenCV**: For image processing and blending.
- **NumPy**: For matrix operations and numerical processing.
- **Matplotlib**: For visualizing images and results.
- **argparse & pathlib**: For handling input directories and file paths dynamically.

5. Results and Evaluation

- The blended composite image maintains a natural transition between different satellite images.

- Color inconsistencies are significantly reduced due to histogram matching and color balancing.
- Edge smoothness analysis confirms a significant improvement in transition areas.
- The final output preserves geographic and structural details without noticeable artifacts.



5.1 Performance Metrics

To evaluate the effectiveness of our method, we consider the following metrics:

1. **Edge Smoothness Score**: Measured using Laplacian variance.
2. **Color Consistency Index**: Evaluated using histogram similarity metrics.
3. **Blending Seam Quality**: Analyzed visually and statistically.

5.2 Comparison with Other Methods

Method	Edge Smoothness Score	Color Consistency
Feathering	Moderate	High
Laplacian Pyramid	High	Very High
Histogram Matching	Low	Moderate

6. References

This project is based on various methodologies and research works related to image compositing, including:

- Awesome Image Composition
- Demo on Image Composition
- Pyramid Blending and Feathering
- Color Adjustment Compositing

7. Conclusion

This project successfully blends satellite images with smooth transitions and color consistency. The combination of pyramid blending and histogram matching ensures high-quality composites suitable for GIS applications, environmental studies, and satellite imagery analysis. Future improvements may include deep learning-based compositing techniques for more advanced blending capabilities.

7.1 Future Work

- **Deep Learning Integration:** Implement CNN-based compositing for more adaptive blending.
- **Multi-Spectral Image Support:** Extend functionality to process different satellite bands.
- **Real-Time Processing:** Optimize algorithms for faster execution in large-scale applications.
- **Automatic Geometric Alignment:** Use feature-based or AI-driven registration techniques to align images accurately.
- **Adaptive Blending Strategies:** Adjust blending techniques dynamically based on texture, brightness, and image content.
- **User-Friendly Interface:** Create a GUI to allow non-experts to process and visualize composite images easily.

⇨ This project contributes to the field of remote sensing by providing an efficient and automated approach to satellite image compositing.