Comprehensive Documentation of the Image Stitching Application

Ken Van Laer, Toon Smets, Ahmad Shakleya

April 2024

Contents

1 Introduction

This documentation provides a comprehensive guide to the Python Tkinter GUI application designed for image processing tasks, specifically focusing on feature detection, feature matching, and image stitching techniques. Developed as a project for the university course II-Industrial Vision Technology, this application serves as a robust tool for users requiring advanced capabilities in handling and manipulating multiple images to produce high-quality panoramas or detailed feature analyses.

1.1 Purpose

The primary purpose of this application is to offer an intuitive and efficient graphical user interface (GUI) that allows users, ranging from amateur photographers to professional image analysts, to perform complex image processing operations with ease. By integrating sophisticated algorithms and providing real-time visual feedback, the application aims to streamline the workflow involved in image editing, enhancement, and analysis tasks.

1.2 Scope

The application covers a wide range of functionalities:

- **Feature Detection**: Utilizes state-of-the-art algorithms to identify and mark key points in images, which are crucial for further processing steps such as matching and stitching.
- **Feature Matching**: Employs advanced matching techniques to find correspondences between features in different images, facilitating accurate alignments in the stitching process.
- **Image Stitching**: Combines multiple overlapping images into a single seamless panorama, adjusting for discrepancies and enhancing the overall image quality.
- **User Interaction**: Provides various GUI tabs and tools for editing, inserting, viewing, and logging information, enhancing usability and accessibility.

This software is particularly suited for applications in fields such as digital photography, surveillance, and scientific research where precise image analysis and manipulation are required.

1.3 General Functionality

The GUI is designed to be user-friendly and responsive, accommodating users with varying levels of expertise in image processing. It features multiple tabs, each dedicated to specific functions:

- **Edit Tab**: Allows users to apply various modifications to the images.
- **Help Tab**: Offers guidelines and documentation to assist users in utilizing the software effectively.
- **Insert Tab**: Facilitates the addition of new images into the application.
- **Log Tab**: Captures and displays log information related to the operations performed, aiding in troubleshooting and performance monitoring.
- **View Tab**: Provides a platform to preview images and the results of the processing tasks.

2 GUI Design

2.1 Main Window

The main window of "CellBlend App" is the core of the user interface, designed to provide a comprehensive and intuitive environment for image processing. Upon launching, the application briefly displays a splash screen, which transitions to the main window once initial setup processes are complete.

The window is structured around a tabbed interface, facilitating easy navigation between different functional areas:

- **File Tab**: Used for importing and managing image files.
- **Edit Tab**: Allow users to change text settings.
- **View Tab**: Allows users to view to stitched images.
- **Help Tab**: Provides help resources and documentation.
- **Log Tab**: Logs and displays application activities and events.

This interface is housed within a resizable window, ensuring that the application can adapt to various screen sizes and user preferences. The window's dimensions are initially set to 600x400 pixels, but can be adjusted to accommodate more extensive interfaces or more detailed views of images.

The architecture is also designed for operational efficiency. For example, the application logs important events, aiding in troubleshooting and user support. The design emphasizes usability with features like:

- **Centering on Screen**: The main window is automatically centered on the screen, enhancing the user experience by positioning the application optimally regardless of screen resolution.
- **Runnability**: The application is packaged as a runnable executable file, making it easily distributable and accessible for end-users without the need for additional installation procedures.

Functionality and aesthetics are balanced to provide a user-friendly interface that is both powerful and accessible, making "CellBlend App" suitable for a range of users from amateurs to professional image analysts.

2.2 File Tab

The File Tab in "CellBlend App" is designed as the initial point of interaction for users engaging in image processing tasks. This tab is divided into three key areas: Images, Stitcher, and Matcher, each tailored to specific parts of the image processing workflow.

2.2.1 Images Tab

The Images Tab serves as the primary interface for importing, displaying, and managing image files within "CellBlend App". It supports essential tasks for initiating the image processing workflow.

- **Import Images**: Users can select multiple images through a standard file dialog, setting the stage for subsequent image manipulation and analysis.

- **Thumbnail Display and Interaction**: Images are displayed as thumbnails within the tab for immediate visual feedback. This interaction allows users to:
  - Select images for further processing actions like stitching or feature matching.
  - Quickly assess which images to process or discard, based on the thumbnail previews.
- **Image Details**: Displays essential information such as file type, size, and dimensions next to each thumbnail, helping users assess the suitability of images for specific processing tasks.
- **Image Management Tools**:
  - **Delete Images**: Enables users to remove unwanted images, refining the dataset for processing.
  - **Toggle Keypoints Display**: Allows users to toggle the display of keypoints on images, crucial for evaluating feature detection quality. This function uses advanced feature detectors (such as ORB, SIFT) to process images and visually mark keypoints directly on the image canvas. This visual aid is instrumental in pre-processing stages, especially for tasks requiring precise feature alignment in stitching or feature matching processes.
- **Progress and Status Feedback**: A progress bar and status label provide dynamic feedback during operations like image loading or processing, enhancing the user interaction by keeping them informed of ongoing processes.
- **Advanced Interaction**:
  - **Canvas for Detailed Viewing**: An integrated canvas allows for in-depth inspection and manipulation of images. Features such as zooming and panning are supported, enabling detailed examination of selected images.
  - **Save Functionality**: Users can save modified or selected images directly from the canvas to the filesystem, which offers flexibility in handling outputs.

2.2.2 Stitcher Tab

The Stitcher Tab in "CellBlend App" is intricately designed to handle all aspects of image stitching, providing users with a robust set of tools and settings to control and execute image stitching processes effectively.

- **Stitching Settings**:
  - **Feature Detector Type**: Users can select from various feature detectors such as SIFT, ORB, BRISK, and AKAZE. Each detector has unique characteristics suitable for different types of images and requirements.
  - **Number of Features**: Adjusting the number of features allows users to balance between detection accuracy and computational performance.
  - **Matcher Type**: Options like Homography or Affine provide flexibility in how images are aligned and stitched.
  - **GPU Acceleration**: Toggles the use of GPU resources to speed up the stitching process, beneficial for handling high-resolution images or complex stitching scenarios.
  - **Warper Type**: Users can choose from various warping methods (e.g., Plane, Cylindrical, Spherical) to optimize how images are projected during stitching.

  - **Confidence Threshold and Cropping**: These settings allow for fine-tuning the stitching output, adjusting how tightly images should match and whether the result should be cropped for a cleaner look.

- **Stitching Execution**:
  - **Start Stitching**: A dedicated button initiates the stitching process of the selected images. Users can observe the progress through a visual progress bar that updates in real-time.
  - **Interactive Feedback**: During stitching, the tab displays dynamic messages about the process status, keeping users informed and engaged.

- **Outcome Feedback**:
  - **Success or Failure Indicators**: Upon completion, the tab provides clear feedback on the success or failure of the stitching operation. Successful stitches display the resulting image, while failures prompt troubleshooting tips or error details.
  - **Display Results**: The stitched image can be viewed directly within the tab, allowing immediate assessment and further actions if needed.

2.2.3 Matcher Tab

The Matcher Tab within "CellBlend App" specializes in the feature matching aspect of image processing, equipping users with advanced tools to perform detailed comparisons between two selected images.

- **Image Selection for Matching**:
  - **Dual Image Selection**: Users can select two images from the imported list through intuitive comboboxes. This selection is critical as it defines the pair of images whose features will be matched.

- **Matching Parameters**:
  - **Matcher Type**: Provides options such as "Homography" and "Affine" for the type of geometric transformation used in matching. This choice affects how accurately the images are aligned based on detected features.
  - **Confidence Threshold**: A slider allows users to set a threshold that controls the robustness of match acceptance, influencing the strictness of feature correspondence validation.

- **Execute Matching**:
  - **Match Features Button**: Initiates the feature matching process once both images and desired settings are selected. This button is dynamically enabled only when valid image pairs are selected, preventing erroneous operations.
  - **Visual Results Display**: The results of the matching process are visually displayed on a canvas within the tab. Users can see lines connecting matched features across the images, providing a clear and intuitive display of how well the selected images align based on the chosen parameters.
  - **Error Handling and Feedback**: In cases where matching fails to find sufficient correspondences or if an error occurs, appropriate feedback is displayed, guiding users to adjust parameters or select different images.

2.3 Edit Tab

The Edit Tab in "CellBlend App" is a versatile environment equipped with a broad array of image and text editing tools, allowing users to modify and personalize their workspace and media. This tab is essential for enhancing user engagement and creativity through intuitive and accessible controls.

2.3.1 Image Editing Tools

The core functionality of the Edit Tab is to facilitate image adjustments with tools that manipulate brightness, contrast, saturation, and the application of various filters like grayscale, sepia, or sharpen effects. Users benefit from:

- **Real-Time Previews**: Every adjustment provides immediate feedback in the preview pane, enabling users to see the effects of their modifications in real-time.
- **Sliders and Input Fields**: These components offer precise control over the degree of each adjustment, ensuring users can fine-tune images to their exact preferences.
- **Undo/Redo Functions**: These features support experimental changes by allowing users to easily revert or reapply modifications, fostering a flexible editing process.

2.3.2 Text and Application Customization

Beyond image processing, the Edit Tab includes advanced customization features for the application interface and text elements, enhancing the user's interaction with the software:

- **Color Customization**:
  - **Text Color**: Users can choose text colors via a color picker, applying changes across labels and buttons within the application, thus personalizing the visual theme.
- **Font Customization**:
  - **Font Type and Size**: Users can select from a variety of fonts and adjust the font size through a combination of a dropdown menu and a slider, respectively. This allows for significant personalization of the application's textual elements.
- **Apply Settings Button**: Confirms and applies all selected customizations, ensuring that changes are not only previewed but also enacted within the user interface.

2.3.3 User Experience Enhancements

The design of the Edit Tab prioritizes ease of use and accessibility:

- **Organized Layout**: Tools are grouped logically and presented in a user-friendly manner, facilitating quick access and intuitive operation.
- **Responsive Interactions**: The interface responds dynamically to user inputs, ensuring a smooth and efficient editing experience.

2.4 View Tab

The View Tab within "CellBlend App" is meticulously designed to facilitate an interactive and efficient analysis of processed images. It integrates advanced image manipulation tools such as zooming, panning, and image rotation, tailored to enhance the user's ability to examine details and assess the quality of images.

2.4.1 Canvas and Image Display

At the heart of the View Tab is a flexible canvas area where images are displayed. This canvas supports:

- **Dynamic Image Resizing**: Users can zoom in and out of images using dedicated "Zoom In" and "Zoom Out" buttons, adjusting the image scale interactively to focus on details or to view the image in its entirety.
- **Panning Functionality**: By clicking and dragging, users can navigate across large or zoomed-in images, allowing for meticulous examination of specific image regions.
- **Initial Image Setup**: Upon loading an image onto the canvas, it is centered and scaled appropriately to fit within the current view, ensuring optimal visibility.

2.4.2 Image Manipulation

The image displayed on the canvas can be manipulated through user interactions, providing a hands-on approach to viewing:

- **Zoom Adjustments**: Each zoom action recalibrates the image resolution, maintaining clarity as the user zooms in for a closer look or out for a broader perspective.
- **Drag to Navigate**: As users explore different parts of an image, the canvas updates dynamically to reflect the new area of focus.

2.4.3 Saving Images

A key feature of the View Tab is the ability to save the currently displayed image:

- **Save Functionality**: Users can save the image being viewed in formats such as PNG, JPEG, or other supported file types via a save dialog that allows specifying the file path and format.
- **File Management**: This feature supports archiving images for future use, presentations, or reports, enhancing the application's utility in professional settings.
- **Operational Feedback**: The application logs the success or cancellation of each save operation, providing feedback via the system's logging service.

2.4.4 User Experience

The View Tab is designed with an emphasis on usability and performance:

- **Responsive Interface**: Despite the high degree of interactivity required for image manipulation, the canvas remains responsive and efficient, ensuring a seamless user experience even during intensive operations.
- **Intuitive Controls**: The zoom and pan controls are intuitively placed and easy to use, requiring minimal learning curve for new users.

2.5 Help Tab

The Help Tab of "CellBlend App" serves as an interactive user manual and a critical resource for support within the application. This tab provides users with direct access to a comprehensive PDF manual, complete with FAQs, troubleshooting guides, and detailed documentation of all features.

2.5.1 PDF Viewer Integration

The Help Tab incorporates a dynamic PDF viewer that displays the application's manual. This viewer allows for:

- **Page Navigation**: Users can move through the document using "Previous" and "Next" navigation buttons.
- **Zoom Functionality**: Zoom in and out buttons enable users to adjust the view according to their preference, enhancing readability and detail visibility.
- **Document Download**: A "Download PDF" button allows users to save the document locally, facilitating offline access.

2.5.2 Handling Missing Documents

If the PDF manual is not found locally upon accessing the Help Tab, the application proactively prompts the user with the option to download it from a pre-configured URL. This ensures that all users have access to the help resources, regardless of their initial setup:

- If the user agrees to download, the application attempts to fetch the document from the specified internet source. On successful download, the document is loaded into the viewer for immediate use.
- In case of download failure, due to network issues or access errors, the user is informed of the failure through an error message, and options to retry the download or check their network settings are suggested.

2.5.3 Technical Details

The Help Tab leverages the PyMuPDF library for efficient rendering of PDF documents within a Tkinter environment. The PDF is displayed on a Canvas, with attached horizontal and vertical scrollbars that adjust according to the zoom level and document size. This setup supports a responsive and user-friendly interaction with the document.

2.5.4 User Experience Features

- **Responsive Design**: The PDF viewer adjusts the displayed content dynamically to fit the changing window sizes and user zoom preferences.
- **Immediate Access and Backup**: Ensures users always have access to the help manual, either locally or via download, minimizing disruptions in accessing support information.
- **Interactive Assistance**: By integrating the manual directly within the application, users receive guided, context-aware support, enhancing their ability to utilize the application effectively.

This Help Tab is designed not only as a static resource but as an interactive, fail-safe component that actively works to ensure users have access to necessary documentation at all times.

2.6 Log Tab

The Log Tab is an integral component of "CellBlend App," designed to provide robust logging functionality that tracks user actions and system responses. This detailed log helps in diagnosing issues, understanding user interactions, and ensuring that the application runs as intended.

2.6.1 Interface

The user interface of the Log Tab features a scrollable text area where log messages are displayed. This widget is designed to be read-only to users, ensuring that log data cannot be altered from the GUI. A dedicated "Export Logs" button enables users to save the logs to a file, supporting various formats for different use cases.

2.6.2 Logging Mechanics

Logging within the application is managed using a structured format that includes the time of the log entry, its severity, and the detailed message. The format—`%(asctime)s - %(levelname)s - %(message)s`—is specifically chosen to provide clarity and immediate insight into the operational status of the application. The default logging level is set to INFO, capturing essential operational data along with any warnings and errors that may occur.

2.6.3 TextHandler Functionality

The `TextHandler` class is a custom logging handler that extends `logging.Handler`, specifically designed to output log messages directly to a Tkinter `ScrolledText` widget. This handler makes it possible to integrate the application's logging framework with the GUI, providing real-time logging feedback directly in the Log Tab.

Here is how the `TextHandler` operates:

- Upon receiving a log record, the `TextHandler` formats the message according to the predefined format and schedules it to be appended to the text widget.
- The `emit` method of the handler ensures that the log message is added to the end of the text widget, automatically scrolling to the latest message. This is particularly useful during extensive operations where real-time updates are crucial.
- To prevent interference with the GUI's responsiveness, the log message insertion is performed asynchronously via the `after` method of the widget, which schedules the update to be processed when the GUI is idle.

2.6.4 Export Functionality

The ability to export logs is a key feature of the Log Tab. By clicking the "Export Logs" button, users can save the current log content to a local file for later review or external use. The save dialog allows selection of the file format and destination, providing flexibility in how logs are archived.

2.6.5 Operational Benefits

The Log Tab, coupled with the `TextHandler`, offers significant benefits:

- **Transparency**: Users have immediate access to detailed logs of all actions performed and responses generated by the application.
- **Troubleshooting**: The detailed logs facilitate quick identification and resolution of issues as they arise.
- **Accountability**: Recording all operations ensures that actions can be traced and audited if necessary.

This tab exemplifies how effective logging practices can be integrated within a software application, enhancing its usability, reliability, and maintainability.

3 Image Processing Capabilities

3.1 Feature Detection

Feature detection plays a pivotal role in image processing tasks like image matching, stitching, and recognition. It involves pinpointing unique points or patterns within an image that remain consistent across various viewpoints, making them essential for applications that require reliable identification and matching of image features.

3.1.1 Process Overview

The feature detection process typically entails several critical steps, each designed to enhance the quality and reliability of the detected features:

1. **Preprocessing**: This initial step involves preparing the image to improve the detectability of features. Common preprocessing techniques include converting images to grayscale to simplify analysis, normalizing illumination to reduce the impact of varying lighting conditions, and applying filters to reduce noise or enhance contrast.

2. **Detector Application**: At this stage, a feature detection algorithm is applied to the preprocessed image. The choice of algorithm can vary based on the specific requirements of the task:

   - SIFT (Scale-Invariant Feature Transform) is renowned for its robustness against changes in scale and rotation.
   - ORB (Oriented FAST and Rotated BRIEF) offers a good balance between speed and performance, particularly in real-time applications.
   - BRISK and AKAZE are other options that provide unique advantages in certain scenarios.

   The detector type and parameters such as the number of features can be dynamically adjusted based on user input or specific application needs.

3. **Key Points Localization**: This step refines the position of each detected feature to sub-pixel accuracy, ensuring precise localization that is crucial for the next steps of image analysis.

4. **Descriptor Extraction**: Following the detection of key points, descriptors are computed for each feature. These descriptors encapsulate the local image appearance around each feature in a way that allows for robust matching across different images.

3.1.2 Feature Matching

Post detection, the identified feature points can be employed to establish correspondences between different images, a process that is fundamental to complex applications such as panorama stitching and 3D reconstruction.

3.2 Feature Matching

Feature matching is a critical process in computer vision that involves comparing features extracted from different images to establish correspondences. This process is pivotal in applications such as image stitching, object recognition, and 3D reconstruction, where aligning multiple images accurately is essential.

3.2.1 Process Overview

The feature matching process involves several key steps:

1. **Feature Extraction**: First, features are extracted from each image using a feature detection algorithm, as outlined in the Feature Detection section. Each feature is described by a set of descriptors that uniquely identify it.

2. **Initialization of Feature Matcher**: A feature matcher object is initialized based on the desired matching technique, which could be Homography, Affine, or others depending on the application requirements.

3. **Matching Execution**: The matcher compares descriptors from different images to find matches. This involves calculating distances between descriptors and identifying the closest matches according to the specified matcher parameters.

4. **Filtering Matches**: Matches are filtered based on a confidence threshold to ensure only the most reliable matches are kept. This step is crucial for improving the accuracy of the matching process.

5. **Visualization**: Optionally, the matches can be visualized by drawing lines between matched features across images. This visual representation helps in assessing the quality and accuracy of the matches.

3.2.2 Error Handling

The feature matching function is designed to handle exceptions robustly, logging errors and returning ‘None‘ if the process fails. This ensures that the application can gracefully manage unexpected issues during execution without crashing.

3.2.3 Practical Usage

In practice, the feature matching process can be invoked after features have been detected in multiple images. For example, in a scenario where three images are used, features are detected in each image, and the feature matching process attempts to find correspondences between these images. The matches are then either used directly for further processing, such as image stitching, or displayed to the user for verification.

3.3 Image Stitching

Image stitching is a complex process that combines multiple overlapping images to produce a single seamless panorama. This technique is widely used in photography, surveillance, and various fields of research where large visual scopes are required.

3.3.1 Stitching Process Overview

The stitching process involves several key stages, each critical to ensuring the quality and coherence of the final panorama:

1. **Image Loading and Preprocessing**: Initially, images are loaded from storage and may be resized to improve processing speed and efficiency. This step is crucial for handling large datasets or high-resolution images.

2. **Feature Detection**: Each image undergoes feature detection where specific points or patterns, known as features, are identified. These features are crucial for aligning and merging images accurately.

3. **Feature Matching**: Using the detected features, the algorithm identifies corresponding features across different images. This step utilizes various matching techniques, with the matcher type (e.g., 'homography', 'affine') chosen based on the application requirements.

4. **Image Alignment and Merging**: After matching, images are aligned based on the matched features. This alignment is critical to ensure that overlapping regions between images blend seamlessly.

5. **Blending and Mosaic Creation**: Finally, aligned images are blended to minimize visible seams and create a continuous image mosaic, often involving techniques to adjust color and brightness for uniformity across the panorama.

3.3.2 Technical Details and Settings

The Python implementation utilizes the 'Stitcher' class, which allows detailed customization through settings such as:

- **Detector Type**: Type of feature detector (e.g., SIFT, ORB).
- **Number of Features**: Influences the granularity of feature detection.
- **Matcher Type**: Determines the algorithm for matching features across images.
- **Confidence Threshold**: Sets the threshold for accepting matches, affecting the robustness and accuracy of the alignment.
- **GPU Utilization**: Option to use GPU acceleration to speed up the process.
- **Cropping**: Optional cropping of the panorama to remove areas with no image data.

These settings are configurable, allowing the stitcher to adapt to different types of imagery and specific requirements of the stitching task.

4 Compiling and Running the Application

This section outlines the necessary steps to set up and run the Image Stitcher application, ensuring users can start using the application with minimal setup.

4.1 Installation

- Ensure Python 3.x is installed on your system. You can download it from the official Python website.
- Dependencies are listed in the ‘requirements.txt‘ file located in the project directory.

4.2 Installing Dependencies

To install the required libraries, open a terminal in the project directory and run:

```
pip install -r requirements.txt
```

4.3 Running the Application

Navigate to the project directory and execute the following command in the terminal:

```
python gui.py
```

4.4 Usage Steps

1. Launch the application via the command line or through your integrated development environment (IDE).
2. Upload the images you intend to stitch using the graphical user interface.
3. Adjust the necessary settings for feature detection, matching, and stitching.
4. Use the 'Stitch Images' button to process the images and view or save the resulting panorama.

5 Appendices

5.1 Additional Resources

For further reading and additional resources, refer to the course materials provided by Prof. dr. Steve Vanlanduit or consult the extensive online documentation available for the Python libraries used in this project.

5.2 Acknowledgments

Special thanks to Prof. dr. Steve Vanlanduit for his invaluable guidance and support throughout the development of this project. Appreciation is also extended to all contributors, including Ahmad Shakleya, Ken Van Laer, and Toon Smets, whose efforts made this project possible. Acknowledgment is given to all open source software contributors whose tools and libraries have been employed in the development of this image stitcher.

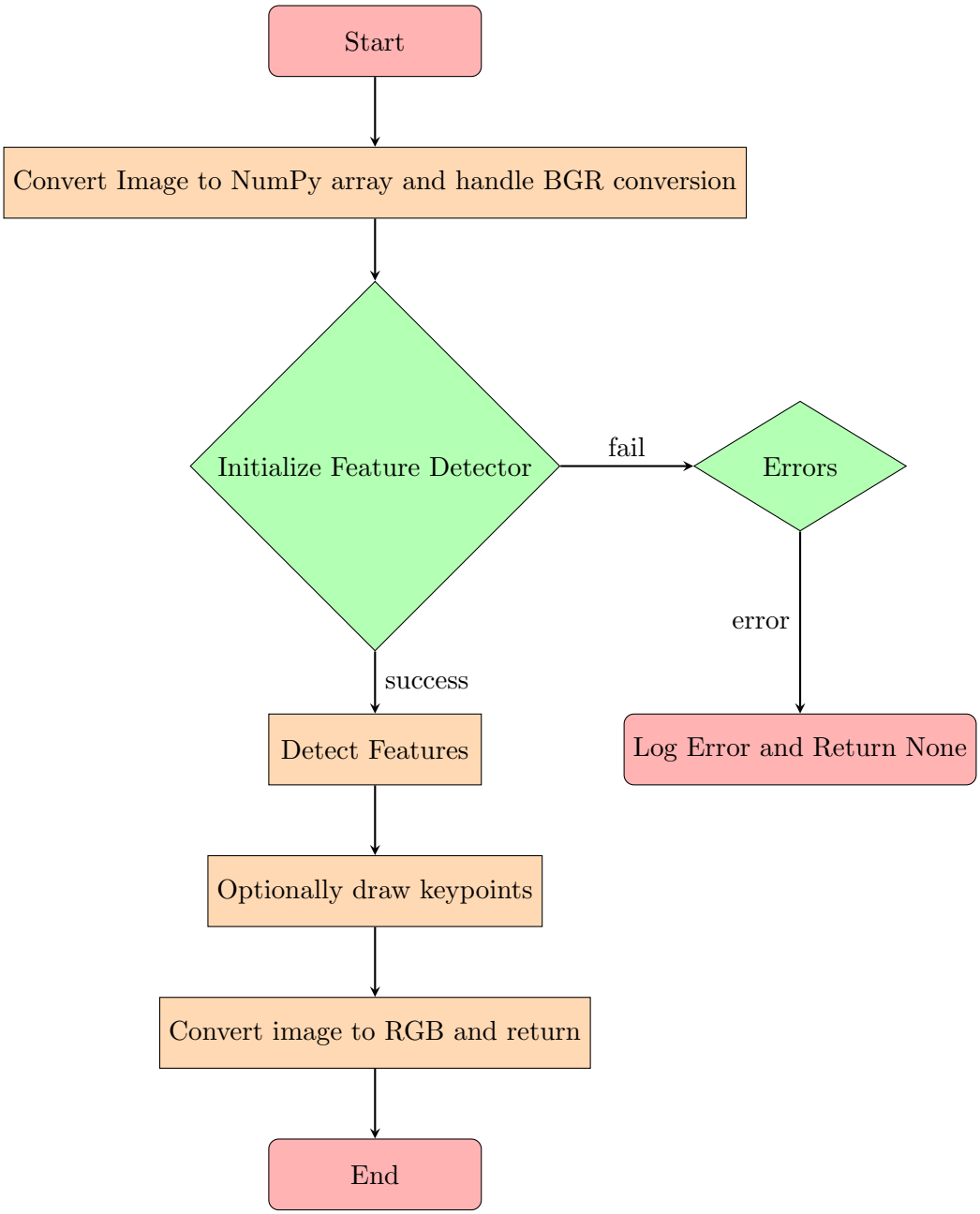

Figure 1: Flowchart of the Feature Detection Process

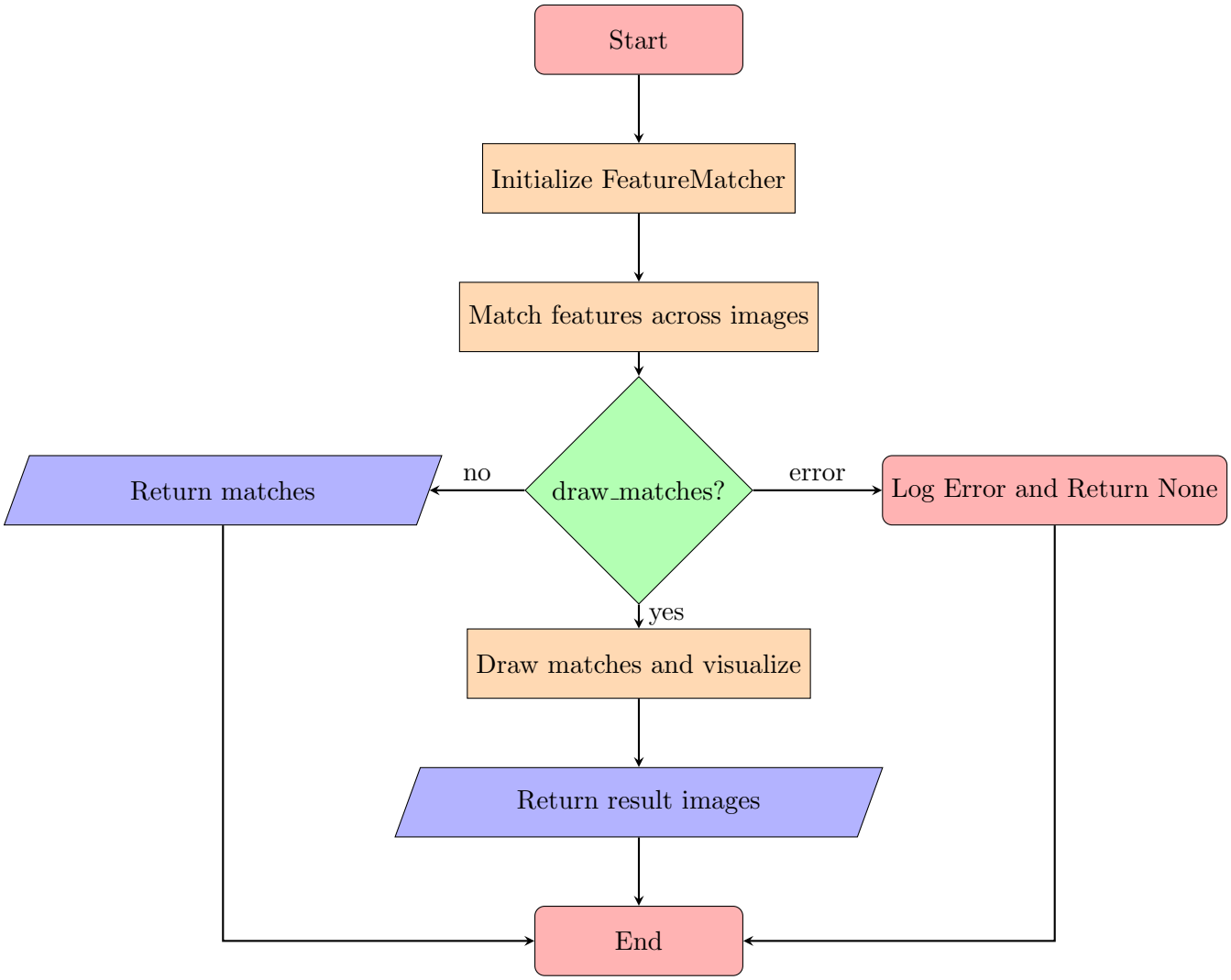

Figure 2: Flowchart of the Feature Matching Process

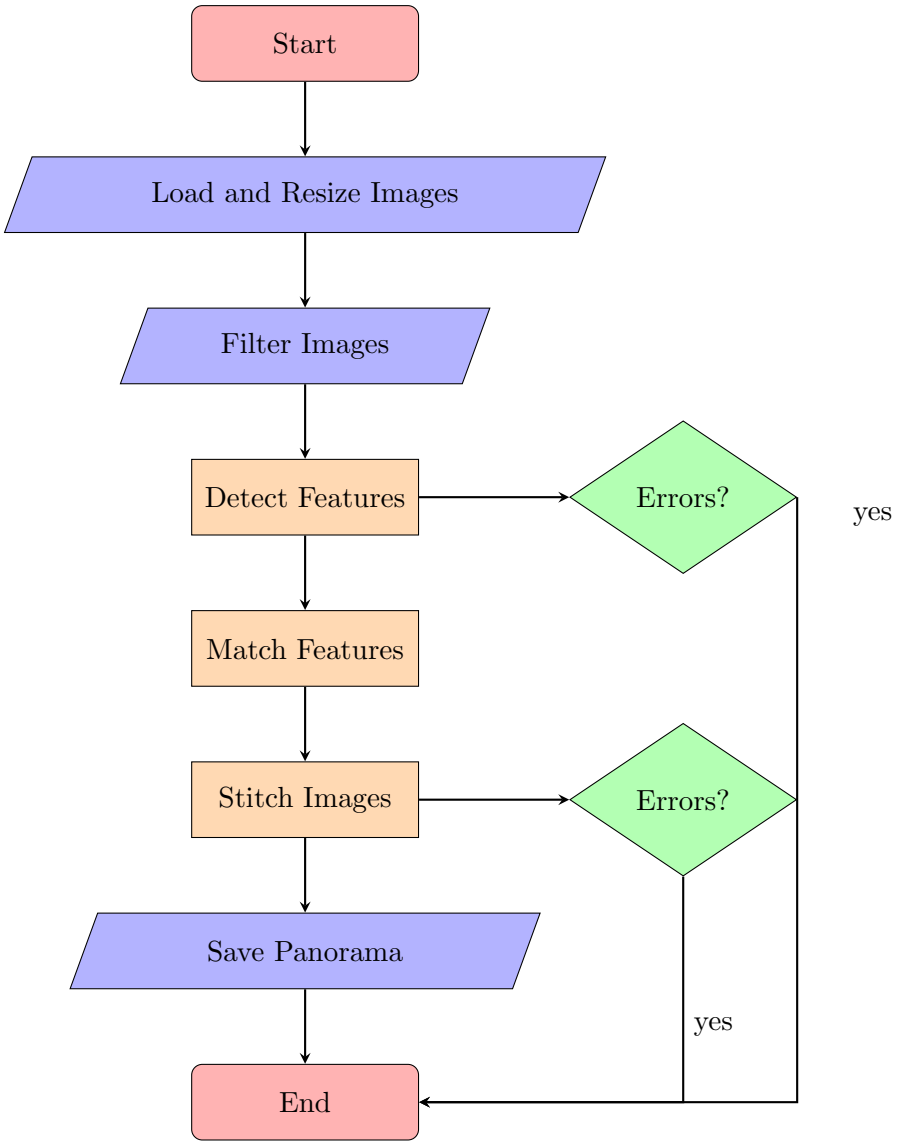

Figure 3: Flowchart of the Image Stitching Process