Overview

The current project focuses on image matching using the ORB (Oriented FAST and Rotated BRIEF) feature detection and matching algorithm. While ORB is a robust method for various computer vision tasks, it might not be the most accurate choice for satellite image processing due to the distinctive characteristics of such images. This report suggests improvements to the project by transitioning to a more advanced and accurate solution, specifically by training a neural network model using Keras.

Issues with ORB for Satellite Image Processing

1. Limited Discriminative Power: ORB relies on keypoint matching, which might not provide sufficient discriminative power for satellite images. Satellite images often contain complex patterns and textures that may not be adequately captured by keypoint-based methods.
2. Insensitive to Scale and Rotation: ORB is less effective when dealing with scale and rotation variations in satellite images. Satellite images may have objects or features at different scales and orientations, making it challenging for ORB to handle such variations.
3. Global Context Ignored: ORB operates on local keypoints and descriptors, neglecting the global context of the entire image. In satellite imagery, understanding the overall scene and context is crucial, and a model capable of learning global patterns may yield better results.

Proposed Solution: Keras Model Training

To enhance the accuracy of satellite image matching, consider implementing a machine learning approach using deep learning techniques. Specifically, use Keras, a high-level neural networks API in Python.

Steps to Improve the Project:

1. ***Data Collection:***

Gather a diverse dataset of satellite images with corresponding ground truth annotations for image matching.

2. ***Data Preprocessing:***

Augment the dataset with variations like rotations, flips, and brightness adjustments.

3. ***Model Architecture:***

Design a convolutional neural network (CNN) architecture suitable for image matching.

Utilize layers like Conv2D, MaxPooling2D, and Dense.

Incorporate siamese network architecture for comparing pairs of images.

4. ***Loss Function:***

Define a suitable loss function for image matching, such as contrastive loss or triplet loss, to train the model on pairs of matching and non-matching images.

Training:

5. ***Split the dataset into training and validation sets.***

Train the model on the training set and monitor performance on the validation set.

Experiment with different hyperparameters and model architectures for optimal results.

6. ***Evaluation:***

Evaluate the trained model on a separate test set to assess its generalization to new, unseen data.

7. ***Integration:***

Implement the trained model into the console-based application, enabling users to load images and perform accurate image matching.