cs5330_project06_landmark

CS5330 Pattern Recognition & Computer Vision

NEU 2023 Fall

Instructor: Bruce Maxwell

Team: Shi Zhang, ZhiZhou Gu

3D Scene Reconstruction Pipeline and its Optimization

Overview

This project aims to reconstruct 3D scenes from a collection of 2D images. It involves several steps from preprocessing images to visualizing the reconstructed 3D scene. Below shows the development process, from a basic local pipeline, to the integration with Structure from Motion library, and eventually setting up a remote pipeline taking advantage from high quality image dataset and advanced 3d reconstruction library.

Tasks (Part 1: Local Pipeline for 3D Reconstruction)

1. Data Preprocessing:

- Select a subset of images from the dataset.
- Preprocess images using OpenCV.
    - Requirements:
        - Resize images for uniformity.
        - Convert images to grayscale.

2. Feature Detection and Matching:

- Implement feature detection algorithms to find key points in images.

- Match these features across different images of the same landmark.
    - Techniques:
        - Use SIFT algorithms for feature detection.
        - Employ feature matching methods like FLANN.

3. Estimating Camera Parameters:

- Estimate the camera parameters and pose for each image.
    - Requirements:
        - Determine intrinsic parameters (like focal length, principal point).
        - Estimate extrinsic parameters (like rotation, translation vectors).

4. 3D Reconstruction:

- Use triangulation methods to estimate 3D points from matched 2D points.
- Generate a point cloud representing the 3D scene.
    - Techniques:
        - Implement triangulation using the camera parameters and feature matches.
        - Assemble the 3D points into a point cloud.

5. Optimization and Refinement:

- Apply bundle adjustment to refine camera parameters and 3D point estimates.
- Clean up the point cloud for noise and outliers.
    - Techniques:
        - Use optimization libraries like OpenCV or g2o for bundle adjustment.
        - Implement noise reduction and outlier removal in the point cloud.

6. Visualization and Analysis:

- Visualize the 3D reconstruction using appropriate tools.
    - Use Open3d Library for visualization.

Tasks (Part 2: Local Pipeline with OpenSfM Integration)

1. Data Preparation:

- Load and read images using OpenCV.
- Perform initial feature detection using OpenSfM's Python bindings.

2. 3D Scene Reconstruction as Sparse Point Cloud

- Run feature extraction and matching using OpenSfM commands ('extract_metadata', 'detect_features', 'match_features', 'create_tracks').
- Execute the Structure from Motion reconstruction process to generate a sparse point cloud.

3. 3D Scene Dense Reconstruction as Mesh Model

- Apply dense reconstruction processes like 'undistort' and 'compute_depthmaps' using OpenSfM commands.
- Integrate the steps to form a complete pipeline for dense 3D reconstruction, resulting in a detailed mesh model of the scene.

Tasks (Part 3: Remote Pipeline on Kaggle with Pycolmap)

0: Import Libraries

- Import necessary libraries for handling images, feature extraction, and visualization.

1: Preprocess the Images

- Load and preprocess the images from the dataset：image-matching-challenge-2023.
- Perform initial processing like resizing, normalization, or format conversion.

2: Feature Matching

- Implement feature detection and matching algorithms.
- Utilize advanced feature matching techniques to establish correspondences between different images.

3: Track Construction

- Construct 2D tracks by linking feature points across multiple images.
- Ensure the consistency and accuracy of these tracks for reliable 3D reconstruction.

4: SfM Model Solution

- Solve for the Structure from Motion (SfM) model using the constructed 2D tracks.
- This involves deriving the 3D structure from 2D image data.

5: Bundle Adjustment

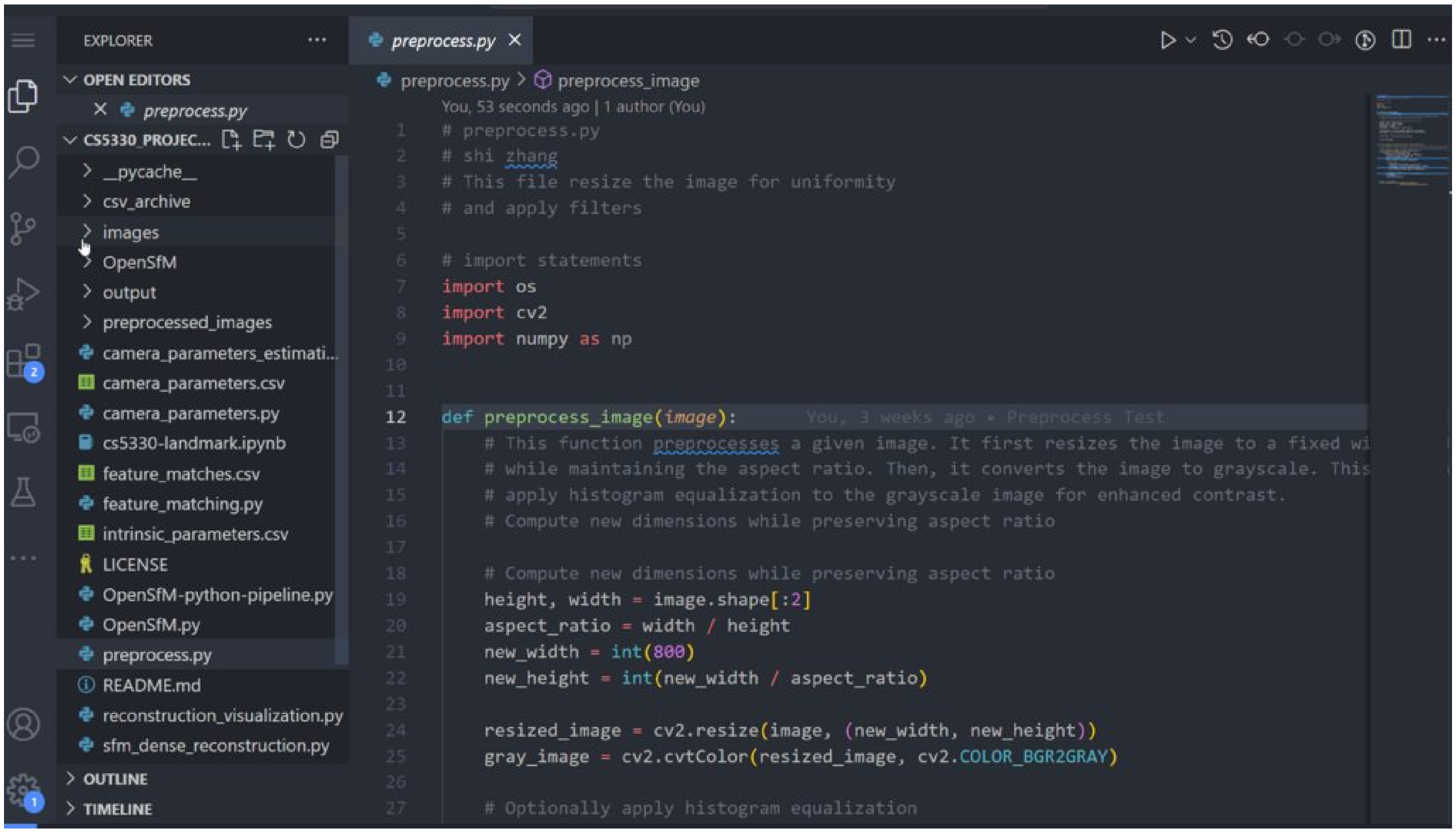
- Apply bundle adjustment techniques to refine the SfM model.
- Optimize camera parameters and 3D point positions for enhanced reconstruction accuracy.

6: Visualization and Analysis

- Visualize the reconstructed 3D scene using suitable tools.
- Analyze the reconstruction's quality and make necessary adjustments for improvement.

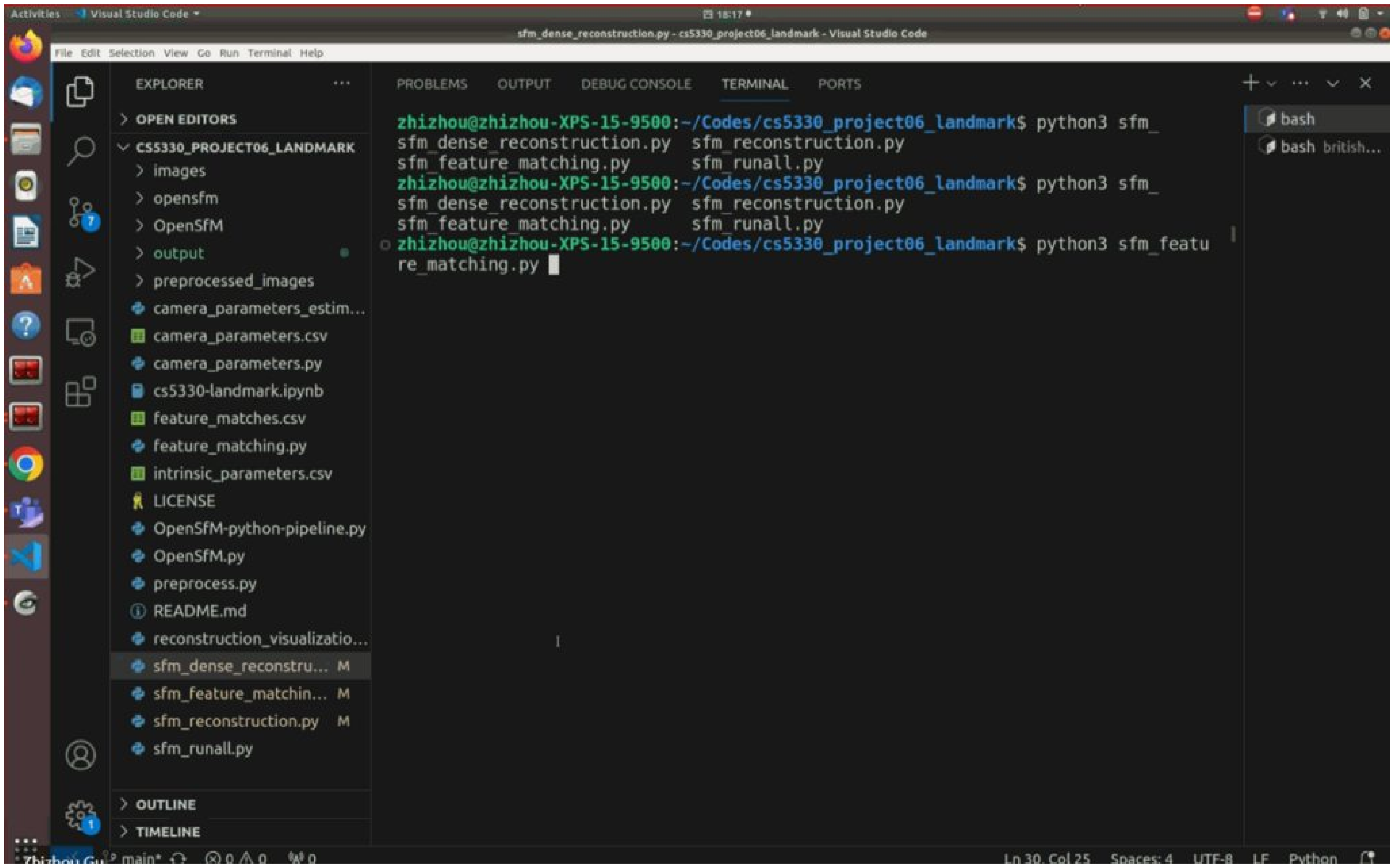
Live Demo

For Part 1: http://i.imgur.com/irmwEpzh.gif



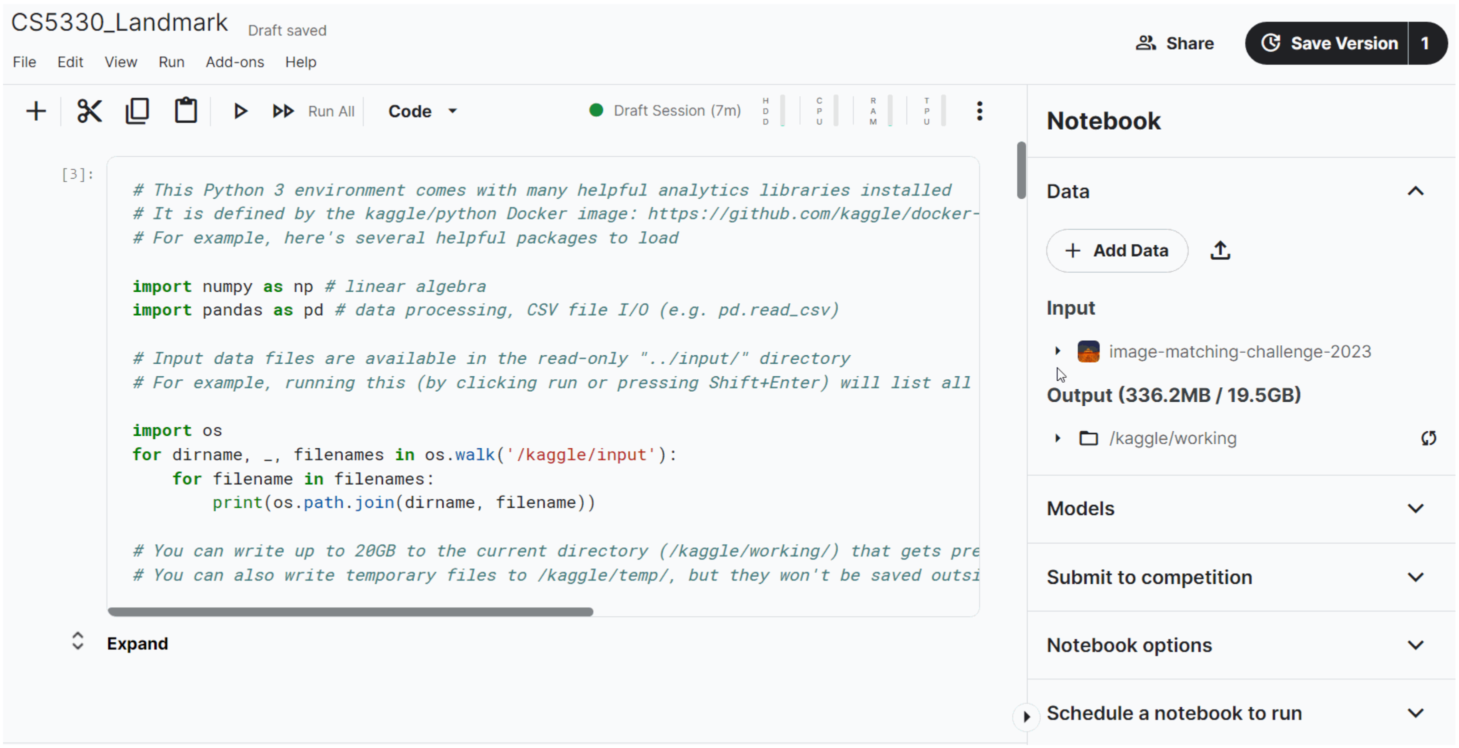
```python
# preprocess.py
# shi zhang
# This file resize the image for uniformity
# and apply filters

# import statements
import os
import cv2
import numpy as np


def preprocess_image(image):
    # This function preprocesses a given image. It first resizes the image to a fixed wi
    # while maintaining the aspect ratio. Then, it converts the image to grayscale. This
    # apply histogram equalization to the grayscale image for enhanced contrast.
    # Compute new dimensions while preserving aspect ratio

    # Compute new dimensions while preserving aspect ratio
    height, width = image.shape[:2]
    aspect_ratio = width / height
    new_width = int(800)
    new_height = int(new_width / aspect_ratio)

    resized_image = cv2.resize(image, (new_width, new_height))
    gray_image = cv2.cvtColor(resized_image, cv2.COLOR_BGR2GRAY)

    # Optionally apply histogram equalization
```

For Part 2: http://i.imgur.com/WfBpl1jh.gif



```
zhizhou@zhizhou-XPS-15-9500:~/Codes/cs5330_project06_landmark$ python3 sfm_
sfm_dense_reconstruction.py  sfm_reconstruction.py
sfm_feature_matching.py      sfm_runall.py
zhizhou@zhizhou-XPS-15-9500:~/Codes/cs5330_project06_landmark$ python3 sfm_
sfm_dense_reconstruction.py  sfm_reconstruction.py
sfm_feature_matching.py      sfm_runall.py
zhizhou@zhizhou-XPS-15-9500:~/Codes/cs5330_project06_landmark$ python3 sfm_featu
re_matching.py
```

For Part 3: https://i.imgur.com/xf8nuuO.gif



```
[3]:
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that gets pre
# You can also write temporary files to /kaggle/temp/, but they won't be saved outsi
```

Acknowledgement

Reference Tutorial

From 2D to 3D: 4 Ways to Make a 3D Reconstruction from Imagery

3D Scene Reconstruction from Multiple Views

3D Reconstruction Playlist

2D to 3D surface reconstruction from images and point clouds

Reference Paper

E.-K. Stathopoulou, M. Welponer, F. Remondino, "Open-Source Image-Based 3D Reconstruction Pipelines: Review, Comparison and Evaluation," The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, 2019.

Y. Wang, Y. Yuan, and Z. Lei, "Fast SIFT Feature Matching Algorithm Based on Geometric Transformation," IEEE Access, vol. 8, pp. 88133-88140, Apr. 2020, doi: 10.1109/ACCESS.2020.2989157.

T. H. M. Siddique, Y. Rehman, T. Rafiq, M. Z. Nisar, M. S. Ibrahim, M. Usman, "3D Object Localization Using 2D Estimates for Computer Vision Applications," in 2021 Mohammad Ali Jinnah University

International Conference on Computing (MAJICC), 2021, doi: 10.1109/MAJICC53071.2021.9526270.

Y. Furukawa and C. Hernández, "Multi-View Stereo: A Tutorial," in Foundations and Trends® in Computer Graphics and Vision, vol. 9, no. 1-2, pp. 1–148, 2013, doi: 10.1561/0600000052.

Project Running Instructions

Project Setup

Our team used Visual Studio Code for this project.

Before running this project, you will need to install several libraries.

Open the command prompt, and navigate to the python folder in your local environment, normally like: C:\Users\AppData\Local\Programs\Python\Python310>

Then run following commands

```
pip install opencv-python numpy scipy matplotlib
pip install open3d
```

For Part 2, you will need to install OpenSfM Library and follow the instructions from here: OpenSfM Documentation

For Part 3, you can run the 'cs5330-landmark.ipynb' on Kaggle.

Project Running

Part 1: Local Pipeline for 3D Reconstruction

Task 1: Data Preprocessing

Run preprocess.py, it should go through image files in a sub-folder under 'images' folder and convert each image to the same size grayscale images saved under a sub-folder under 'preprocessed_images' folder.

Task 2: Feature Detection and Matching

Run feature_matching.py, it should load images from 'preprocessed_images' folder, implement feature detection algorithms to find key points in images, then save the image features as a csv file, like 'feature_matches.csv'.

Task 3: Estimating Camera Parameters

Run camera_parameters_estimation.py, it should first read features from the 'feature_matches.csv', and estimate the camera parameters and pose for each image, and save the camera parameters as a csv file, like 'camera_parameters.csv'.

Task 4 - 6: 3D Reconstruction and Visualization

Run reconstruction_visualization.py, it should load 'camera_parameters.csv', 'intrinsic_parameters.csv' and 'feature_matches.csv', triangulate points, then create and visualize the point cloud.

Part 2: Local Pipeline with OpenSfM Integration

Task 1: Feature Extraction and Matching

- Run `sfm_feature_matching.py` .
- This script will extract image features and match them using OpenSfM commands like 'extract_metadata', 'detect_features', 'match_features', 'create_tracks'.

Task 2: SfM Reconstruction

- Execute `sfm_reconstruction.py` .
- This will run the Structure from Motion reconstruction process to generate a sparse point cloud.

Task 3: Dense Reconstruction

- Run `sfm_dense_reconstruction.py` .
- This script applies dense reconstruction processes using commands like 'undistort' and 'compute_depthmaps'.

Task 4: Run All Steps

- Optionally, execute `sfm_runall.py` to run all the above steps in sequence.

Part 3: Remote Pipeline on Kaggle with Pycolmap

Task 1: Preprocess the Images

- Use the provided code block to preprocess images.
- This includes resizing images and converting them to grayscale.

Task 2: Feature Matching

- Implement the feature matching code block using SIFT and BFMatcher.
- Visualize and save the feature matches.

Task 3: Using SfM

- Use the Structure from Motion (SfM) code block.
- Parse the camera and image data for 3D reconstruction.

Task 4: 3D Reconstruction

- Run the 3D reconstruction code block.
- Visualize the reconstruction using the provided visualization tools.

Task 5: Save the 3D Scene

- Execute the code block to save the 3D scene.
- Use PyntCloud to export the point cloud.