Submitter: Shira Broner
ID: 207375841

CV - Project

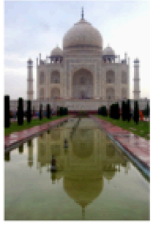
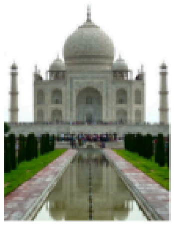
Project Goal

Given pairs of real-world outdoor images, compute their fundamental matrix.
[Kaggle]

Data Images

Submitter: Shira Broner
ID: 207375841

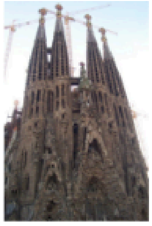
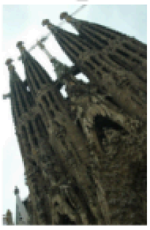
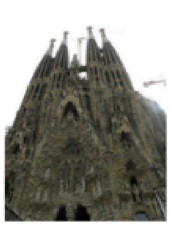
taj_mahal

taj_mahal

taj_mahal


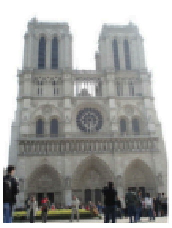
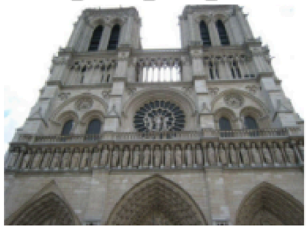
temple_nara_japan

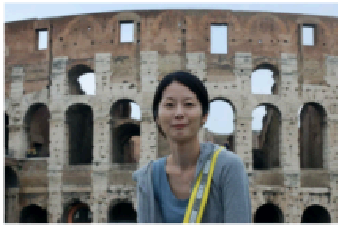
temple_nara_japan

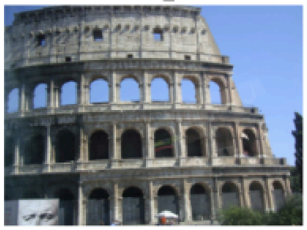
temple_nara_japan


sagrada_familia

sagrada_familia

sagrada_familia


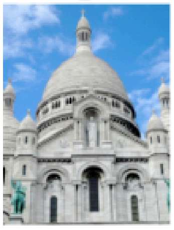
notre_dame_front_facade

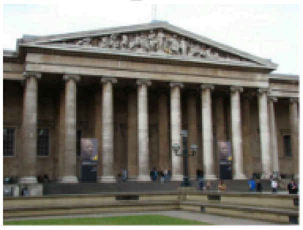
notre_dame_front_facade

notre_dame_front_facade


colosseum_exterior

colosseum_exterior

colosseum_exterior


sacre_coeur

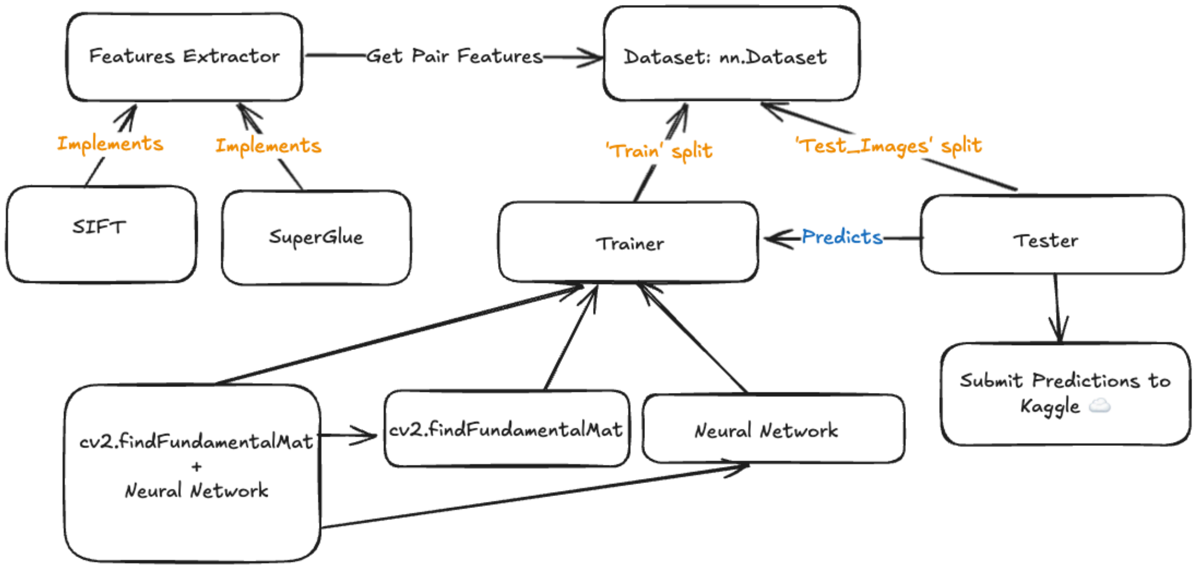
sacre_coeur

sacre_coeur


british_museum

british_museum

british_museum


Submitter: Shira Broner
ID: 207375841

Project Structure



I structured the project to facilitate experimentation with various features and algorithms. First, I outline the key components of my code, followed by the experiments conducted, challenges encountered, mistakes made, and results obtained..

Dataset *(dataset.py)*:

- Extracts zip file to a temp folder
- Loads pair_covis.csv and all the images paths and filters out all the pairs with co-visibility (> 0.1)
- Configures the number of keypoints required (> 2048) and maximum images per scene (1000).
- Initializes the required feature extractor (SuperGlue) that is a GNN with attention mechanism.

Frameworks and models for feature matching (feature_extractor.py):
Given a pair of images, performs feature matching: extract keypoints and descriptor and match them.

- **SIFT**: the standard way to extract keypoints and descriptors and match them using OpenCV
- **SuperGlue** : a more novel way to extract and match keypoints and descriptors using GNN and attention. Here is the recommended configuration for outdoor setting:
  - SuperPoint → nms_radius: 3
  - SuperPoint → keypoint_threshold: 0.05
  - SuperPoint → max_keypoints: 2048
  - Superglue → weights: outdoor
  - SuperGlue → sinkhorn_iterations: 20
  - SuperGlue → match_threshold: 0.2

Submitter: Shira Broner
ID: 207375841

- **LoFTR:** LoFTR is a transformer based model to extract and match keypoints (not implemented in my code)
- **RoMA**: Robust dense feature matching (not implemented in my code)
- **MambaGlue:** which is a hybrid Mamba-Transformer feature matching framework that just released on Feb25 (no code yet)

  The extracted features for each pair are saved to a file to reduce execution time during experimentation.

- **Trainer:**
  First, the training data was randomly split 80% for train and 20% for validation. Given features extracted that contain: keypoints 0, keypoints 1, matches, matched keypoints 0, matched keypoints 1, training in a supervised manner using the fundamental matrix assigned to each pair.
    - **cv2.findFundamentalMat**: no training, just using the built-in function with matched keypoints from the feature extractor while performing hyper-parameters tuning.
    - **Neural Network**: using the raw features (keypoints, descriptors and matches) to predict the fundamental matrix in a supervised manner
    - **cv2.findFundamentalMat + Neural Network:** a combination of the two algorithms above, using the raw features + the predicted fundamental matrix by OpenCV and calibrating it.
- **Tester**:
  Loads *test.csv* and initializes a Dataset instance with split '*test_images*'. For each pair, extract features and predict the fundamental matrix by the trainer's predictor. Generates a submission CSV file for Kaggle

Experiments

1. **SIFT → cv2.findFundamentalMat:** similar code as suggested baseline code. Extracting keypoints using SIFT, then matching with KNN. The matched keypoints are then used to find the fundamental matrix
   Result: score of 0.043
2. **SuperGlue → cv2.findFundamentalMat → DL:** SuperPoint and SuperGlue were used to find matches. The fundamental matrix from OpenCV was calibrated with RMSE loss
   Result: The model failed to converge..
3. **SuperGlue → custom head:** Attempted to build a custom head on top of SuperGlue's Matching module. Finding an appropriate loss function was challenging due to the geometrical nature of the problem (Epipolar Geometry).
   Result: Some OpenCV functions, such as cv.recoverPose, were not backward-compatible with PyTorch and could not be used as a loss function.

Submitter: Shira Broner
ID: 207375841

4. **SuperGlue → cv2.findFundamentalMat:** I went back to focusing on performing feature matching with SuperGlue and then used the matched features as an input for cv2.findFundamentalMat with RANSAC method (threshold 0.3, confidence 0.7, maxIters= 2000).
   Result: test score of **0.15441**
5. **SuperGlue → cv2.findFundamentalMat (scaled images):** Initially used SuperGlue's recommended image resizing ([840, 840]), but realized it affected geometric relationships. Instead, images were resized while preserving resolution by adding borders.
   Result: test score of **0.33108**
6. **SuperGlue → cv2.findFundamentalMat (no resize):** Modified SuperGlue's preprocessing to remove the resizing step
   Result: test score of **0.31214**
7. **SuperGlue → cv2.findFundamentalMat (no resize + hyper-parameter tuning)**: I Performed hyperparameter tuning for cv2.findFundamentalMat. Best parameters:
   a. method=cv2.USAC_MAGSAC (that is more robust to outliers)
   b. confidence=0.5
   c. maxIters=2000

   Result: test score of **0.49548**

Notes

1. **SuperGlue occasionally overmatches keypoints**, leading to errors due to insufficient keypoints (< 8). If co-visibility ≤ 0.1, the mean fundamental matrix from training data is used as a fallback.
2. **SuperGlue has likely been fully optimized** within the given constraints. Exploring alternative libraries like LoFTR could yield better results but was not pursued due to GPU limitations.
3. **Key lesson learned**: Avoid delving into deep learning prematurely. Simpler methods should be maximally leveraged before exploring complex models.

Model and Code

Attached model file called 'model.pkl', which is the SuperGlue model.

Code entry point: **main.py**

Submitter: Shira Broner
ID: 207375841

Final Prediction Flow