# Final Report and Requirements

## Final Report

### Feature Extraction

**SIFT**

- **Number of Features Extracted**: 15986 (training set), 3994 (testing set)

**HOG**

- **Number of Features Extracted**: 16000 (training set), 4000 (testing set)

### Keypoint Matching

**SIFT**

- **Number of Matches Found**: 1485

### Classifier Accuracy

**SIFT**

- **SVM Accuracy**: 0.12

**HOG**

- **SVM Accuracy with HOG Features**: 0.09

### Summary

- The SIFT feature extraction process resulted in 15986 features for the training set and 3994 features for the testing set.
- The HOG feature extraction process resulted in 16000 features for the training set and 4000 features for the testing set.
- The keypoint matching function found 1485 matches between the sample images using SIFT features.
- The SVM classifier trained on SIFT features achieved an accuracy of 12% on the test set.
- The SVM classifier trained on HOG features achieved an accuracy of 9% on the test set.

## Requirements

### Keypoint Detectors

1. **Feature Extraction**:
    - **SIFT**:
        - Implement the extract_sift_features function to extract SIFT features from images.
        - Save the extracted features in a file named processed_cifar10_sift.npz containing X_train, y_train, X_test, and y_test.
    - **HOG**:
        - Implement the extract_hog_features function to extract HOG features from images.
        - Save the extracted features in a file named processed_cifar10_hog.npz containing X_train, y_train, X_test, and y_test.
2. **Feature Matching**:
    - Create a function custom_match_descriptors that matches keypoint descriptors between two images.
    - Implement a function plot_keypoint_matches to visualize the matched keypoints.

3. **Bag of Visual Words**:
    - Implement functions build_visual_vocabulary and build_histograms to create a visual vocabulary using KMeans clustering and build histograms of visual words for each image.
    - Use TF-IDF to adjust the frequency of visual words using the adjust_frequency_vector function.
4. **Classification**:
    - **SIFT**:
        - Implement evaluate_sift.py to train and evaluate an SVM classifier using SIFT features.
        - Report the accuracy of the classifier on the test set.
    - **HOG**:
        - Implement evaluate_hog.py to train and evaluate an SVM classifier using HOG features.
        - Report the accuracy of the classifier on the test set.

**Image Stitching**

1. **Transformation Estimation**:
    - Implement compute_affine_transform to estimate the affine transformation matrix.
    - Implement compute_projective_transform to estimate the projective transformation matrix.
2. **RANSAC**:
    - Implement a function ransac to estimate transformation matrices and validate them using RANSAC.
3. **Image Stitching**:
    - Implement a function stitch_images to stitch two images together using the computed transformation matrices.
    - Use the provided sample images (yosemite1.jpg and yosemite2.jpg) to test the implementation.
    - Save the code in a file named stitch_images.py.

**Commands to Run the Code**

1. **Setting Up the Environment**:

poetry install

2. **Running Feature Extraction**:

poetry run python feature_matching/keypoint_detectors/feature_extraction.py

3. **Evaluating SIFT Classifier**:

poetry run python feature_matching/keypoint_detectors/evaluate_sift.py

4. **Evaluating HOG Classifier**:

poetry run python feature_matching/keypoint_detectors/evaluate_hog.py

5. **Running Image Stitching**:

poetry run python feature_matching/image_stitching/stitch_images.py