CSS-587 Final Project Proposal:
LP-SIFT Implementation and Optimization

David Li, Ben Schipunov, Kris Yu

November 3, 2025

1. Selected Paper

Title: Local-peak scale-invariant feature transform for fast and random image stitching
Authors: Hao Li, Lipo Wang, Tianyun Zhao, Wei Zhao
ArXiv: https://arxiv.org/pdf/2405.08578
MATLAB source:
https://github.com/haohaohao111222/LP-SIFT-Algorithm-Combined-with-RANSAC

2. Project Description

Background:
LP-SIFT replaces SIFT's computationally expensive Gaussian pyramid with multi-scale local peak detection, achieving 10-100× speedup on large images while maintaining stitching quality. The paper demonstrates successful stitching of nine 2600×1600 pixel images in under 160 seconds.

Proposal Summary:
Validate paper results by replicating work using the provided dataset and MATLAB implementation on our hardware. Discuss and report results. Port work to C++, this will include reimplementing the paper's algorithm in C++ using OpenCV and optimizations on the Local Peak (LP) implementation. We'll rerun the benchmarking with the new C++ implementations and compare results, noting performance improvements. If time permits, our stretch goal would be further optimization leveraging GPU acceleration.

In conclusion for our project, we will provide a demo application that uses LP-SIFT to show first hand how fast the image stitching process improves, specifically performing best with large resolutions.

Goals
1. Validate paper results using provided MATLAB implementation
2. Port LP-SIFT to C++ with OpenCV (library + CLI tool)
3. Benchmark against SIFT, ORB, BRISK, SURF on small/medium/large images
4. Apply CPU optimizations for local peak algorithm
5. Implement multi-image mosaic stitching for random fragments (demo application)
6. Stretch: GPU acceleration with CUDA

3. Input and Output

Input: Image pairs/sets in JPEG/PNG format across three size categories: small (~600×400), medium (1080×1920), and large (3072×4096). Parameters include interrogation window sizes L and matching threshold ∆s.
Output: Stitched panoramic images, performance metrics (timing, feature counts, match quality), comparison visualizations, and benchmark results in CSV format.

4. Evaluation Methodology

Replicate paper benchmarks using identical datasets (mountain, street, terrain, building, campus scenes). Measure processing time, feature point counts, matching accuracy, and visual quality.
Compare LP-SIFT against OpenCV's SIFT/ORB/BRISK/SURF implementations + RANSAC.

5. Schedule of Work

Phase 0 (Week 1: Nov 3-10): Run MATLAB baseline, validate paper results
Phase 1 (Weeks 2-3: Nov 11-24): C++ implementation of LP-SIFT algorithm
Owner: Algorithms

- Image preprocessing, local-peak detection, SIFT descriptors, Feature matching, RANSAC homography, multi-image mosaic stitching
- **Deliverables**: `liblpsift.lib`/`lpsift_cli.exe` (pairwise & folder modes). Config file with parameters.

Phase 2 (Week 4: Nov 25 - Dec 1): Benchmark replication
Owner: Evaluation

- Test on paper's datasets, compare with SIFT/ORB/BRISK/SURF
- **Deliverables**: `results.csv,` plots, side-by-side comparisons (C++ vs. MATLAB); table summarized by size bucket (small/medium/large).

Phase 3 (Week 5: Dec 2-8): CPU optimizations
Owner: Optimization

- Implement C++ specific optimizations on the local peak algorithm (to be determined)
- **Deliverables**: Performance comparison with baseline C++ implementation

Phase 3+ (Week 6: Dec 9-10): Final Report + GPU acceleration (stretch goal)

- CUDA kernels for gradient/histogram computation (The paper notes GPU/C++ would markedly improve efficiency over their MATLAB baseline.
- **Deliverables**: Final report, demo video and/or live demonstration

6. Publicly Available Code and Data

Reference Implementation: MATLAB code from paper co-author Wei Zhao
Baseline Comparisons: OpenCV's cv::SIFT, cv::ORB, cv::BRISK, cv::SURF implementations

Datasets: Public benchmark images from paper references, custom 6MP smartphone captures
Our Contribution: Optimized C++ port with SIMD/parallel optimizations and optional GPU acceleration, exceeding the original MATLAB implementation's performance

7. GitHub Repository

https://github.com/davecyli/css587project
Repository to include: C++ source code, library/CLI tool, test datasets, benchmark scripts/results, configuration files, documentation, and final report.

8. Preferred Dates

Paper Presentation: November 10 or 12, 2025
Design Review: November 24, 2025

9. Bibliography

[1] Li, H., Wang, L., Zhao, T., & Zhao, W. (2024). Local-peak scale-invariant feature transform for fast and random image stitching. arXiv:2405.08578v2.

[2] Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision, 60, 91-110.

[3] Rublee, E., Rabaud, V., Konolige, K., & Bradski, G. (2011). ORB: An efficient alternative to SIFT or SURF. 2011 International Conference on Computer Vision, 2564-2571.

[4] Bay, H., Ess, A., Tuytelaars, T., & Van Gool, L. (2008). Speeded-up robust features (SURF). Computer Vision and Image Understanding, 110(3), 346-359.

[5] Leutenegger, S., Chli, M., & Siegwart, R. Y. (2011). BRISK: Binary robust invariant scalable keypoints. 2011 International Conference on Computer Vision, 2548-2555.

[6] Fischler, M. A., & Bolles, R. C. (1981). Random sample consensus: A paradigm for model fitting. Communications of the ACM, 24(6), 381-395.