Reflection and Traceability Report on Image Feature Correspondences for Camera Calibration Software

Kiran Singh

1 Changes in Response to Feedback

1.1 SRS and Hazard Analysis

A hazard Analysis was determined to exceed the scope of work for the development of the Image Feature Correspondences for Camera Calibration software. This software will not be used in real-time or safety critical applications.

Version 1.0 of the **SRS** was published under commit **8776a2b**. Version 2.0 of the **SRS** was published under commit **d139755**.

Software Requirements Specification Revision Summary

- **Source of Feedback:** GitHub Issue #2
  Issue Identified: The *Table of Units* was not applicable to the software system.
  Action Taken: The table was removed from the document.
  Commit: c7d6830
- **Source of Feedback:** GitHub Issue #2
  Issue Identified: Inconsistent symbol definitions across theoretical models, instance models, and data definitions.
  Action Taken: The *Table of Symbols* was updated for consistency across all sections.
  Commit: 6c49512
- **Source of Feedback:** GitHub Issue #4
  Issue Identified: Expected user inputs and responsibilities were unclear in the System Context Diagram.
  Action Taken: Diagram and descriptive text were revised to clarify user input responsibilities.
  Commit: 86e483c
- **Source of Feedback:** Instructor (verbal) during VnV Plan discussion
  Issue Identified: Lack of documentation for OpenCV-based constraints.

Action Taken: A new system constraint based on OpenCV limitations was added.
Commit: c7d6830

- **Source of Feedback:** GitHub Issue #2
  Issue Identified: Definition of extrinsic parameters was vague.
  Action Taken: Definition of extrinsic parameters was clarified.
  Commit: e0236d3

- **Source of Feedback:** GitHub Issue #2
  Issue Identified: Requirement GS5 was outdated and unclear.
  Action Taken: Replaced GS5 with new requirement R15 for feature correspondence reporting.
  Commit: 6c9a58a

- **Source of Feedback:** GitHub Issue #2
  Issue Identified: Requirements GS1 and GS2 were no longer needed.
  Action Taken: GS1 and GS2 were removed.
  Commit: e0236d3

- **Source of Feedback:** GitHub Issue #2
  Issue Identified: Theoretical and instance models used inconsistent or unclear notation.
  Action Taken: TM01–TM04 and IM01–IM04 were revised for consistent mathematical presentation.
  Commit: 6c49512

- **Source of Feedback:** GitHub Issue #2
  Issue Identified: The definition of the XOR operation in TM4 was unclear.
  Action Taken: Improved definition and symbolic notation for XOR was added.
  Commits: c7d6830, 6c49512

- **Source of Feedback:** GitHub Issue #2
  Issue Identified: IM01 lacked detailed steps for computing the Gaussian-smoothed image.
  Action Taken: IM01 was expanded to include a full description of the required computation steps.
  Commit: 6c49512

- **Source of Feedback:** GitHub Issue #2
  Issue Identified: No non-functional requirements for software quality were provided.
  Action Taken: Added five non-functional requirements (NFR1–NFR5) covering reliability, usability, maintainability, and performance.
  Commit: 6c9a58a

- **Source of Feedback:** GitHub Issue #5
  Issue Identified: Table of Symbols was not consistent with mathematical models.
  Action Taken: Table of Symbols was updated to align with definitions and notation in theoretical and instance models.
  Commit: 6c49512

1.2 Design and Design Documentation

Version 1.0 of the **MG** was published under commit **75b68fc**. Version 2.0 of the **MG** was published under commit **f7b81b4**.

Module Guide Revision Summary

- **Source of Feedback:** GitHub Issue #17
  Issue Identified: Minor grammatical and formatting errata were present in Revision 1.0 of the Module Guide.
  Action Taken: All minor grammar and formatting issues were corrected.
  Commit: b71d13a

- **Source of Feedback:** GitHub Issue #18 (AC12) and GitHub Issue #19 (AC13)
  Issue Identified: Both issues requested clarification on how anticipated changes (AC12 and AC13) impact software design.
  Action Taken:

  - AC12 was rewritten to emphasize the design implications rather than user interaction.
  - AC13 was removed due to significant overlap with the revised AC12.

  Commit: f7b81b4

- **Source of Feedback:** GitHub Issue #20
  Issue Identified: The Control Module was described as an abstract data type rather than an abstract object.
  Action Taken: The description of the Control Module was revised to reflect it as an abstract object.
  Commit: b71d13a

- **Source of Feedback:** GitHub Issue #21
  Issue Identified: A link to the VnV Plan was not provided in the Module Guide.
  Action Taken: An explicit reference to the VnV Plan was added to the Module Guide.
  Commit: b71d13a

- **Source of Feedback:** GitHub Issue #15
  Issue Identified: It was suggested that a graphical user interface (GUI)

be added to support image display in M11 (Image Plot Module).
Action Taken: No change was made. It was determined that image generation using OpenCV functions like `imwrite()`, `drawKeypoints()`, and `drawMatches()` was sufficient, and GUI development was outside the project scope.
Commit: Not applicable (no changes made).

Version 1.0 of the **MIS** was published under commit **75b68fc**. Version 2.0 of the **MIS** was published under commit **9f9e07a**.

Module Interface Specification (MIS) Revision Summary

- **Source of Feedback:** GitHub Issue #15
Issue Identified: OpenCV class data in Section 4 duplicated content from Section 16.
Action Taken: Redundant content in Section 4 was removed to avoid duplication.
Commit: 9f9e07a

- **Source of Feedback:**GitHub Issue #15
Issue Identified: Question raised about enumerating available image processing methods in Section 6.4.1.
Action Taken: No action taken. Justification was provided: all available methods are presented to the user dynamically via Specification Parameters Module and validated in the Input Format Module.
Commit: 9f9e07a

- **Source of Feedback:**GitHub Issue #15
Issue Identified: Path strings used as environment variables were not defined as state variables.
Action Taken: Path strings were reclassified as state variables, including similar updates in Sections 7.4.2 and 9.4.2.
Commit: 9f9e07a

- **Source of Feedback:** GitHub Issue #21
Issue Identified: Method calls lacked explicit module references in Section 6.4.4.
Action Taken: All method calls were updated to include explicit module prefixes (e.g., `config.get_active_methods()`).
Commit: 9f9e07a

- **Source of Feedback:**GitHub Issue #15
Issue Identified: `getcwd()` was incorrectly attributed to the Python `Path` class in Section 7.4.4.
Action Taken: Corrected to state that `getcwd()` belongs to the `os` module and clarified its output semantics.
Commit: 9f9e07a

- **Source of Feedback:** GitHub Issue #21
  Issue Identified: The `check_limits()` method in Section 7.4.4 had undefined arguments.
  Action Taken: Method signature was updated to include the appropriate tuning parameter arguments.
  Commit: 9f9e07a

- **Source of Feedback:**GitHub Issue #15
  Issue Identified: Tuning parameters in Sections 8.3 and 8.4 were redundantly defined as both exported constants and state variables.
  Action Taken: Removed state variable definitions to retain only exported constants.
  Commit: 9f9e07a

- **Source of Feedback:**GitHub Issue #15
  Issue Identified: Suggested that method variables (e.g., `mthd_...`) in Section 8.3.1 should be booleans.
  Action Taken: Suggestion was kindly rejected. Method identifiers remain as integers to support enumerated selections for future extensibility.
  Commit: Not applicable (no change implemented)

- **Source of Feedback:**GitHub Issue #15
  Issue Identified: `make_directory(folder_name)` in Section 9.4.4 did not clarify its effect on environment variables.
  Action Taken: Semantics were updated to specify that this function causes an environment variable transition.
  Commit: 9f9e07a

1.3 VnV Plan and Report

Version 1.0 of the **VnV Plan** was published under commit **f73ef27**. Version 2.0 of the **VnV Plan** was published under commit **d139755**.

Verification and Validation Plan Revision Summary

- **Source of Feedback:** GitHub Issue #9
  Issue Identified: The course name "CAS 741" was mentioned within the VnV Plan, which is not appropriate.
  Action Taken: All references to "CAS 741" were removed from the VnV Plan.
  Commit: e67619f

- **Source of Feedback:** GitHub Issue #10
  Issue Identified: Relevant documents were missing from Section 2.4 (Relevant Documentation).

Action Taken: Section 2.4 was updated to include descriptions of the SRS, MG, and MIS.
Commits: bec8167, 66fa0a4

- **Source of Feedback:** GitHub Issue #11
  Issue Identified: Minor errata were present, including incorrect table placement, inconsistent acronym/symbol usage, and test numbering errors.
  Action Taken: All identified errata were corrected throughout the VnV Plan.
  Commit: e67619f

- **Source of Feedback:** GitHub Issue #12
  Issue Identified: Reference to ArUco markers created the impression that only ArUco was acceptable.
  Action Taken: Reference to ArUco was removed and replaced with a general term: "generated binary markers".
  Commit: f73ef27

- **Source of Feedback:** GitHub Issue #13
  Issue Identified: System tests STFR-KP-01 and STFR-DC-01 lacked clear traceability to R6 and R8.
  Action Taken: Method flags were added to both tests to identify that corner detection and Hamming distance comparisons were used, respectively.
  Commit: 60dac24

- **Source of Feedback:** GitHub Issue #14
  Issue Identified: Requirement R4 and its connection to STFR-FD-01 lacked a clear definition of descriptor bin size.
  Action Taken: Descriptor bin size was explicitly defined in the Table of Symbols and the STFR-FD-01 input section.
  Commit: f298373

- **Source of Feedback:** GitHub Issue #7 (Software Design Instructor)
  Issue Identified: Symbols table contained unused entries not referenced in the VnV Plan.
  Action Taken: Symbols table was reduced to only include symbols and abbreviations used in the document.
  Commit: bec8167

- **Source of Feedback:** GitHub Issue #7
  Issue Identified: Section 2.3 lacked specification of the challenge level of the project.
  Action Taken: Section 2.3 was updated to state that the project is a research-level front-end to a camera calibration pipeline.
  Commit: 86e483c

- **Source of Feedback:** GitHub Issue #7
  Issue Identified: Reference documents PS, SRS, MG, and MIG were not mentioned in the VnV Plan.
  Action Taken: These documents were listed and described in Section 2.4.
  Commits: bec8167, 66fa0a4

- **Source of Feedback:** GitHub Issue #7
  Issue Identified: The proposed Software Validation Plan simply duplicated verification activities and did not reflect actual validation.
  Action Taken: Validation was identified as out of scope and the section was updated accordingly.
  Commit: 86e483c

- **Source of Feedback:** GitHub Issue #7
  Issue Identified: System tests lacked specific parameter values for each scenario.
  Action Taken: STFR-IS-01, STFR-KP-01, STFR-FD-01, and STFR-FM-01 were each updated to include well-defined test inputs and parameters.
  Commit: 86e483c

- **Source of Feedback:** GitHub Issue #7
  Issue Identified: Definition of a "match" was missing in STFR-FM-01.
  Action Taken: A clear definition of a match was added to STFR-FM-01.
  Commit: 86e483c

1.4 Verification and Validation Report

Version 1.0 of the **VnV Report** was published under commit **1304f5e**. No feedback has been received to propose additional revisions.

2 Challenge Level and Extras

2.1 Challenge Level

This project is defined as an advanced research project. The output of the Image Feature Correspondences for Camera Calibration software form the front end of a robust optimization algorithm for extrinsic camera calibration. The framework of this software will be iterated upon during the Summer 2025 development period and beyond to collect meaningful input data to test the backend of the optimization software.

2.2 Extras

No extra deliverables will be delivered with this project.

3 Design Iteration (LO11 (PrototypeIterate))

The design of the IFCS system evolved substantially over the course of the development cycle. Although the image processing techniques employed—such as keypoint detection, smoothing, and feature description—are well-established, the process of integrating and validating these components within a cohesive pipeline proved more complex and tacit than initially anticipated. It became evident that the diversity of image capture conditions—ranging from indoor to outdoor environments, varying lighting conditions, sensor noise, and differences in feature density—necessitated a modular architecture.

To accommodate this variability, the baseline implementation of IFCS was designed to support interchangeable image processing methods, enabling users to tailor the pipeline to their specific operating environments. This architectural flexibility was solidified during the development of the Module Guide and Module Interface Specification, where decomposition-by-secrets was applied to isolate critical design concerns. In particular, this approach decoupled the implementation of image processing routines from the formatting and storage of output data.

The benefits of this modularity were twofold: it facilitated parallel development and testing of the image processing components, and it enabled the use of synthetic test images to independently validate the behavior of modules responsible for smoothing, keypoint detection, feature description, and matching. This strategy not only improved test coverage but also ensured that design decisions could be iterated with minimal disruption to the overall system architecture.

4 Design Decisions (LO12)

Schedule and developer availability were the most significant factors influencing the design of the IFCS system. All development activities had to be scoped to align with the constraints of the Winter 2025 development schedule. This necessitated the implementation of a minimally viable product (MVP) and required deferring several non-essential features and test cases. For instance, advanced capabilities such as homography-based alignment, Harris corner detection, and FLANN-based feature matching were excluded from the development scope to ensure that the core functionality of the IFCS pipeline could be delivered, validated, and documented within the allotted timeline.

One of the most impactful design assumptions was the requirement that imagery must originate from at least one camera. While seemingly minimal, this assumption provided significant architectural flexibility. It allowed the system to remain agnostic to the number of cameras involved, thereby supporting both single-camera and multi-camera workflows without additional code branching or conditional logic. This abstraction enabled users to formulate their calibra-

tion and correspondence tasks based on their application context, without being constrained by the internal structure of the software.

The final design decision with long-term implications was the selection of `OpenCV` as the foundational image processing library. Although it was theoretically possible to implement all operations using custom Python code, OpenCV was chosen for its maturity, efficiency, and extensive support for standard computer vision tasks. This decision significantly accelerated development and shifted testing priorities from unit-level verifications of low-level routines to higher-level functional validation. Moreover, reliance on OpenCV highlighted specific areas where custom implementations could add value, such as in the `drawMatches()` visualization function, where certain parameters like line thickness or Hamming distance thresholds are not directly configurable. These limitations were flagged for future refinement and served to guide subsequent decomposition efforts where customization and extensibility are prioritized.

5 Economic Considerations (LO23)

The camera calibration pipeline developed for this project is not intended for commercial sale but rather as an open-source contribution to the academic and research community. The objective is to provide a reliable and extensible tool that supports robust feature detection and matching, with direct application to multi-camera calibration tasks such as extrinsic pose estimation, hand-eye calibration, and SLAM systems.

There is a strong and growing user base for such tools, particularly among robotics researchers, computer vision engineers, and graduate students. Many of these users currently rely on closed-source libraries or partial solutions that lack transparency or flexibility. By providing an open implementation with clear modularity—especially in the feature detection and matching components—we aim to attract users who require more control over detection thresholds, descriptor tuning, or match filtering logic than what is available in black-box tools such as those bundled in proprietary suites.

To attract users, dissemination efforts would focus on:

- Publishing documentation and usage examples via GitHub and ReadTheDocs.
- Demonstrating the accuracy and repeatability of the pipeline through reproducible benchmarks (e.g., using ArUco tags or synthetic datasets).
- Participating in relevant online communities and forums (e.g., ROS Discourse, OpenCV forums, or academic mailing lists).
- Linking to the tool in academic publications or theses where it is applied.

The feature detection module in particular plays a key role in attracting users. Many calibration failures or suboptimal results stem from poor keypoint localization or insufficient match coverage. By offering an interface that allows users to customize detector parameters (e.g., FAST threshold, ORB patch size, descriptor binning strategy), users can tailor the pipeline to different environments, whether operating on textured outdoor scenes or structured indoor setups.

The cost to develop and maintain the pipeline is limited to developer time, version control, and documentation hosting—all of which can be managed at minimal or no monetary expense through public infrastructure. Since the software is not intended to generate revenue, success will be measured in terms of user adoption, citation count (in the case of academic use), and contributions from external users in the form of bug reports or feature enhancements.

6 Reflection on Project Management (LO24)

6.1 How Does Your Project Management Compare to Your Development Plan

Each milestone of this project required significantly more time that anticpated to refine its deliverables and incorporate feedback. The use of version control for task management was performed effectively as outlined in Section 1. The lead developer worked dilligently with the Domain Expert to address feedback in a timely manner, as reflected in the closure dates for each issue in GitHub. The lead developer also worked diligently to inform the project supervisor of critical updates with each of the key deliverables of the program, as well as a code demonstration and walkthrough. The biggest miss was that, in an effort to prevent over-constraining the design space, the lead developer did not impose the constraint of using OpenCV until midway through the development cycle. Once it was implemented, the rate of development notably increased and accelerated prototyping and testing activities.

6.2 What Went Well?

The use of behaviour hiding and a decomposition by secrets ellucidated several key factors that should be anticipated to change, such as the expected methods of image processing and the methods of plotting observed feature matches. This enabled the design of modules with loose coupling which yielded benefits in both modularity and parallel development.

6.3 What Went Wrong?

The implementation of unit and coverage testing using GitHub Actions presented several challenges, particularly related to the configuration of relative imports for modules located in the `projectFiles` directory. Initial attempts to

run the pipeline on GitHub failed due to incorrect module resolution, which required extensive troubleshooting and refactoring of the package structure. These issues were eventually resolved, but they introduced delays during critical stages of development.

In parallel, feedback for project milestones was provided by the instructor in PDF format rather than through inline comments on the corresponding `.tex` source files. This presented a logistical issue for tracking revisions. The development team had two primary options: manually recreate each comment as a separate GitHub issue, or incorporate changes while linking them back to the original milestone issue containing the instructor's feedback. The team chose the latter approach, prioritizing efficiency and development continuity. However, this decision reduced the traceability of "Was–Is" changes across the evolving specification documents.

6.4 What Would you Do Differently Next Time?

A number of key lessons emerged throughout the development of the IFCS software system, several of which pertain to early-stage planning and architectural decision-making.

First, identifying critical use cases at the outset of the project would have more effectively informed both the functional requirements and the software design decisions. Early enumeration of these use cases—such as multi-view feature matching, inter-frame correspondence validation, and camera calibration using ArUco patterns—would have clarified the required interfaces and processing capabilities. This would, in turn, have helped focus development efforts on ensuring robustness and flexibility where it was most needed, rather than giving equal priority to all aspects of the design.

Second, a structured evaluation of existing software tools could have better motivated the development of this pipeline. By cataloguing the limitations of existing tools early—such as lack of transparency in feature scoring, or rigid camera model assumptions—the motivation for a modular, customizable system becomes clearer and more compelling. Such an exercise would have aided in articulating the distinct value proposition of the IFCS pipeline.

Third, the decomposition-by-secrets methodology—used to partition responsibilities across modules based on hidden implementation details—would have been more impactful if conducted earlier. Applying this strategy at the onset of development would have enabled cleaner separation of concerns, better isolation of testable units, and improved traceability between modules and requirements. Its delayed application led to some restructuring late in the development cycle, though this ultimately proved to be of significant benefit for the overall design. However, those benefits would have yielded a greater return if they had been identified earlier in the development cycle.

Another key insight pertains to the evaluation of external dependencies, particularly OpenCV. While the library ultimately provided substantial benefits such as its offering reliable implementations of core image processing routines and accelerating development, an initial attempt to avoid its use resulted in wasted development time. Early efforts focused on replacing OpenCV functionality with custom implementations in the interest of transparency and control. However, these alternatives often introduced complexity, reduced efficiency, and required additional debugging effort. In retrospect, a more pragmatic approach would have been to adopt OpenCV as a baseline from the beginning, then selectively replace components only when customization was demonstrably necessary.