Discrepancy Analysis of OpenCV

Authors

Gabriel Rincon - 218624734 - gabrielenrrincon@gmail.com
Arash Saffari - 218791632 - arashrt@my.yorku.ca
Nargis Rafie - 220785903 - nargis.rafie00@gmail.com
Joshua Zuker - 218135574 - joshz@my.yorku.ca
Leonard Schickedanz - 222634661 - leon.schickedanz@gmail.com
Shaheer Lone - 217277807 - shaheerlone@gmail.com

Abstract

This report analyzes the software architecture of OpenCV by comparing its conceptual design against its concrete implementation using the Reflexion Model. The conceptual architecture was derived from system documentation and manual source code analysis, while the concrete architecture was found from the use of analysis tools such as SciTools Understand and LSEdit. This comparison revealed many discrepancies, specifically divergences where the concrete system exhibited higher interconnectivity than the conceptual model. To investigate the validity of these dependencies, a W4 (Which, Who, When, Why) analysis was applied to key subsystems including calib3d, features2d, and imgproc. The analysis confirmed that the majority of these divergences were not architectural erosion, but intentional, practical decisions driven by code reuse and optimization. As a result, this report proposes a remediation strategy to update the conceptual architecture, formalizing valid functional dependencies to accurately reflect the system's evolution.

Introduction and Overview

In the previous architecture analyses, various models were formed utilizing different tools; the conceptual architecture was derived from module documentation, while the concrete architecture was generated using tools such as LSEdit and Understand. Comparing these models reveals discrepancies at both the top-level and subsystem layers, establishing a baseline for accuracy. However, because OpenCV is a large-scale, constantly evolving system, new features and optimizations inevitably create dependency divergences. Therefore, simply identifying where discrepancies occur is insufficient; understanding the underlying reasons for this architectural drift is critical for a reliable understanding of the system.

To address this, the discrepancies were analyzed using specific techniques, primarily the reflexion analysis, which compares the conceptual and concrete architectures to discover divergences. To investigate individual discrepancies further, a W4 approach (StickyNotes) was applied to discover their specific sources in the code. This method revealed the specific functions responsible for unexpected dependencies in the concrete model, while also highlighting relationships that were less substantial than initially assumed. By analyzing the two architectural models in this manner, we can move beyond simple comparison to the creation of a final, refined model that accurately reflects the true state of OpenCV's architecture.

Top Level Architecture Analysis

At the top-level view of OpenCV, Figure 1 shows the original conceptual architecture diagram, which showed a simple layered architecture in which higher level modules build on lower level ones, all of which also depend on the Core. However, Figure 2 shows that once the concrete architecture was generated, a much more interconnected system formed. Green arrows highlight convergences, which are dependencies that did not change from the original conceptual architecture. The red arrows highlight divergences that were not found in the conceptual diagram. Finally, the blue arrows highlight absences, which represent dependencies that were in the conceptual diagram but not in the concrete. Many modules have lots of dependencies, while others are more like utility hubs that are relied on by other modules.

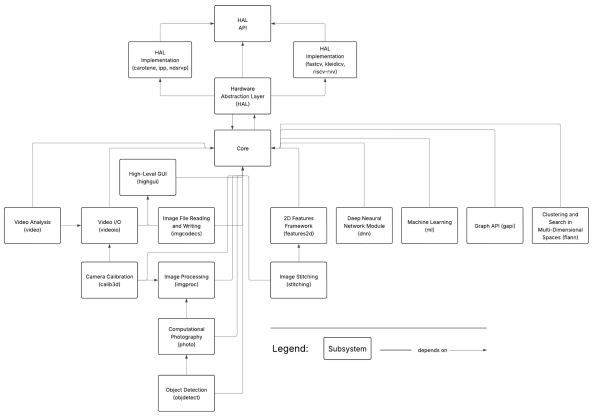

Figure 1 - Conceptual Architecture

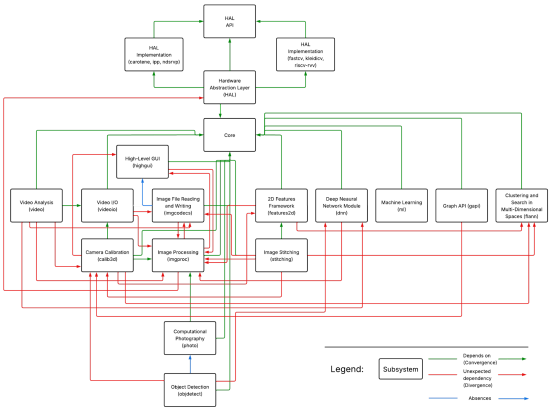

Figure 2 - Concrete Architecture

A detailed LSEdit concrete architecture visualization can be seen in Figure 3, which further relates to the concrete architecture diagram shown in Figure 2.

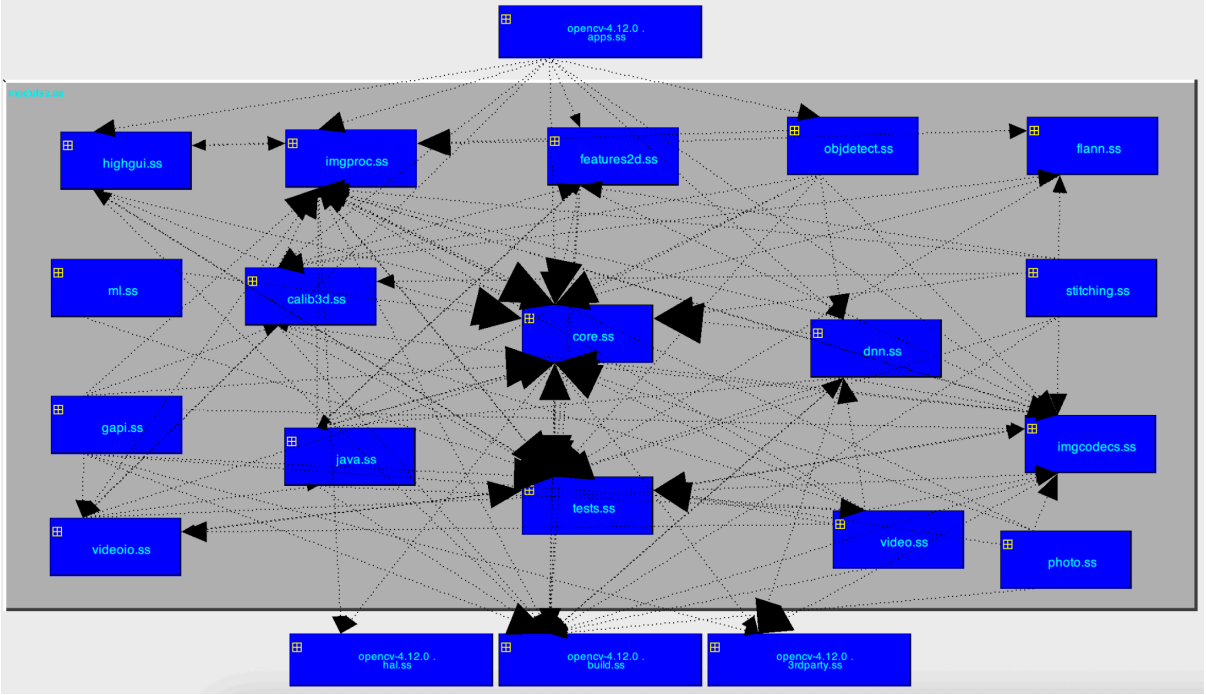

Figure 3 - LSEdit Concrete Architecture

When examining our chosen subsystem, Calib3d, we found in Figure 3 that its top-level behavior matches the conceptual expectations: it depends heavily on Core for mathematical operations such as linear algebra, matrix manipulation, and optimization routines. Yet at the concrete level, Figure 4 reveals Calib3d's internal structure; a linear, staged flow of events, reflecting a Pipe-and-Filter style that is not easily visible in the conceptual architecture.

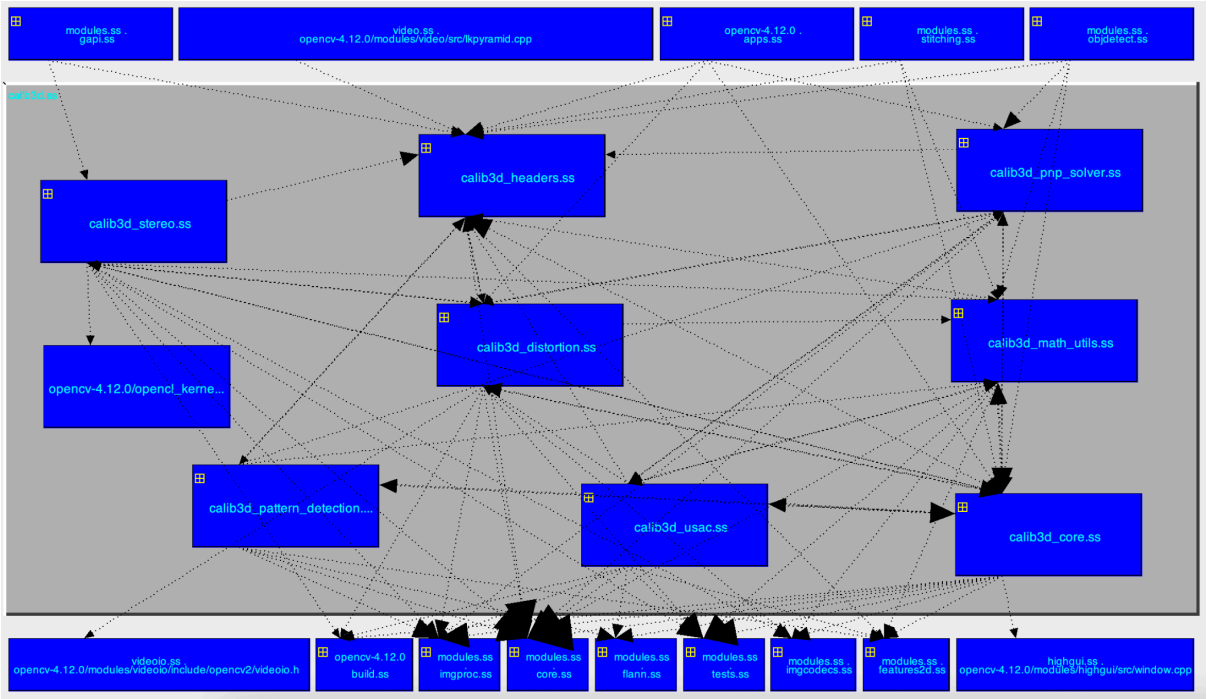

Figure 4 - LSEdit Calib3d Visualization

Reflexion Analysis Process

A reflexion model provides a structured approach for comparing a system's expected (conceptual) architecture with its recovered (concrete) architecture from implementation. For systems as large as OpenCV that are constantly evolving, this technique is especially valuable as understanding the system's architecture and complexity can be difficult [1].

Our team applied the reflexion model using the artifacts developed in the propose and compare phases. The conceptual architecture stemmed from the propose phase, and helped get a general idea of how relationships and modules in OpenCV are connected and work together. The concrete architecture formed as a result of the compare phase, where using tools such as SciTools Understand and LSEdit produced dependency graphs giving a deeper understanding of the systems infrastructure. To connect these two, a mapping file was built that associated OpenCV's source files with their corresponding conceptual subsystems. Overlaying the conceptual and concrete architecture allowed us to generate the reflexion analysis.

In the reflexion model, absences occur when a dependency in the conceptual model does not appear in the concrete one. Dependencies that appear both in the conceptual and concrete models are called convergences. Dependencies that don't appear in the conceptual and only appear when a thorough analysis of the code is done are called divergences. In the concrete

architecture diagram, the green arrows represent the original conceptual convergences, and the red arrows represent the divergences found.

To analyze these divergences, the W4 approach was applied. This includes the following dependency investigation questions:

- Which code entity is responsible for the dependency?
- Who added/removed the dependency?
- When was the dependency modified?
- Why was the dependency added/removed?

Understand was used to locate the files that are responsible for the dependency, and then examined the GitHub repository to determine when it was introduced and by whom. The commit messages provided further details about what the goal of the change was and what the developers intentions' may have been.

Discrepancy Analysis

After reviewing the discrepancies across the selected OpenCV modules, there were many patterns that emerged. First, there were several divergences that were a result of the evolution of different modules or subsystems over time. Some of these evolutions focused on improving the subsystems' performance, particularly in modules such as dnn, objdetect, and stitching. These modules are ones that rely on additional components such as the Hardware Abstraction Layer (HAL), calib3d, and flann to improve their performance and accuracy. Second, we saw that many conceptual dependencies underestimated the complexity of subsystem interactions. For example, stitching was originally thought to be dependent on core and features2d, but the concrete analysis showed that flann and calib3d revealed additional dependencies. Lastly, the W4 method showed that most of the divergences were intentional and introduced by developers to support system improvements such as speed and accuracy. These findings demonstrate the need to update conceptual architecture to better reflect OpenCV's implementation requirements.

2D Features Framework (features2d)

- Conceptual dependency: core
- Concrete dependencies: core, flann, imgproc

Discrepancy	flann
Which?	**Entity**: FlannbasedMatcher (opencv/modules/features2d/src/matchers.cpp) **Dependency**: IndexParams & SearchParams (opencv/modules/flann/include/opencv2/flann/miniflann.hpp)
Who?	Maria Dimashova
When?	Oct 29, 2010
Why?	To configure the algorithm and search depth (IndexParams and SearchParams) needed by the Flann-based matcher for efficient keypoint search [2].

Discrepancy	imgproc
Which?	**Entity**: FastFeatureDetector (opencv/modules/features2d/src/fast.cpp) **Dependency**: cvtColor (opencv/modules/imgproc/src/color.cpp)
Who?	Vadim Pisarevsky
When?	Oct 17, 2014
Why?	Provides color space conversion (cvtColor) which is necessary for the feature detector to efficiently identify corners [3, 4].

Deep Neural Network (dnn)

- Conceptual dependency: core
- Concrete dependency: core, imgproc

Discrepancy	imgproc
Which?	**Entity**: blobFromImage (opencv/modules/dnn/src/dnn_utils.cpp) **Dependency**: resize (opencv/modules/imgproc/src/resize.cpp)
Who?	Zhiming Zheng
When?	Nov 29, 2020
Why?	Utility reuse for preprocessing: resize() is needed to format the image tensor to the specific input size required by the DNN models [5].

Image Stitching (stitching)

- Conceptual dependency: core, features2d
- Concrete dependencies: core, features2d, imgproc, calib3d, flann

Discrepancy	imgproc
Which?	**Entity**: SeamFinders (opencv/modules/stitching/src/seam_finders.cpp) **Dependency**: GcGraph (opencv2/imgproc/detail/gcgraph.hpp)
Who?	Gaetano Checinski
When?	Sep 17, 2018
Why?	It uses the detail attribute in GCGraph in imageproc to define and create graphs [6].

Discrepancy	calib3d
Which?	**Entity**: MotionEstimators (modules/stitching/src/motion_estimators.cpp) **Dependency**: CameraCalibration (opencv2/calib3d/calib3d_c.h)
Who?	Andrey Kamaev
When?	Apr 11, 2013
Why?	Needs to calibrate the camera first and know its position in order to properly estimate motion [7].

Discrepancy	flann
Which?	**Entity**: PerformanceMatchers (modules/stitching/perf/perf_matchers.cpp) **Dependency**: Flann (opencv2/flann.hpp)
Who?	Alexander Alekhin
When?	Oct 22, 2016
Why?	Uses the nearest neighbor approximation algorithm (best algorithm for the problem) to compare the performance of OpenCV to said algorithm [8].

Camera Calibration (calib3d)

- Conceptual dependency: core, imgproc, videoio
- Concrete dependencies: core, imgproc, videoio, features2d, highgui

Discrepancy	features2d
Which?	**Entity**: RadiusSearchNeighborhoodGraphImpl (opencv/modules/calib3d/src/usac/utils.cpp) **Dependency**: FlannBasedMatcher (opencv/modules/features2d/src/matchers.cpp)
Who?	Maksym Ivashechkin
When?	Aug 14, 2020
Why?	Used to provide the Flann-based search algorithm for efficiently finding nearest neighbors within the neighborhood graph, avoiding brute force [9].

Discrepancy	highGUI
Which?	**Entity**:CalibProcessor

	(opencv/apps/interactive-calibration/frameProcessor.cpp) **Dependency**: imShow (opencv/modules/highgui/src/window.cpp)
Who?	rogday (Full name not public)
When?	Jun 30, 2022
Why?	Used to help show a captured message for debugging [10].

Discrepancy	calib3d
Which?	**Entity**: ShowPoints (modules/calib3d/test/test_chesscorners.cpp) **Dependency**: imshow (modules/highgui/src/window.cpp)
Who?	Vadim Pisarevsky
When?	Feb 9, 2011
Why?	Debugging/testing - reworked tests [11].

Object Detection (objdetect)

- Conceptual dependency: core, photo
- Concrete dependencies: core, dnn, calib3d

Discrepancy	dnn
Which?	**Entity**: FaceRecognize (modules/objdetect/src/face_recognize.cpp) **Dependency**: dnn (opencv2/dnn.hpp)
Who?	Yuantao Feng
When?	Oct 8, 2021
Why?	To modernize the face analysis toolkit by incorporating more powerful and accurate deep learning algorithms from the dnn module [12].

Discrepancy	calib3d
Which?	**Entity**: QRCode (modules/objdetect/src/qrcode.cpp) **Dependency**: calib3d (in opencv2/calib3d.hpp)
Who?	Nesterov Alexander

When?	Oct 10, 2018
Why?	Requires camera calibration to "undistort" the image, which corrects the QR code's perspective, making detection and decoding easier and more reliable [13].

Image Processing (imgproc)

- Conceptual dependency: core
- Concrete dependencies: core, hal, highui, imgcodecs

Discrepancy	hal
Which?	**Entity**: ippiGrayToRGB_C1C3R (opencv/modules/imgproc/src/color_rgb.dispatch.cpp) **Dependency**: CV_INSTRUMENT_FUN_IPP (opencv/hal/ipp/include/ipp_utils.hpp)
Who?	Rostislav Vasilikhin
When?	Mar 15, 2018
Why?	Part of a major code refactoring to split color conversion implementations and utilize the HAL for better organization and CPU/OpenCL optimization [14].

High-level GUI (highgui)

- Conceptual dependency: core
- Concrete dependencies: core, imgproc

Discrepancy	imgproc
Which?	**Entity**: (/modules/highgui/include/opencv2/highgui/highgui_c.h) **Dependency**: imgproc (opencv2/imgproc/imgproc_c.h)
Who?	Vadim Pisarevsky
When?	Aug 14, 2014
Why?	Maintenance - trying to fix builds [15].

Video I/O (videoio)

- Conceptual dependency: core
- Concrete dependencies: core, imgcodecs, imgproc

Discrepancy	imgcodecs

Which?	**Entity**: imread (opencv/modules/videoio/src/cap_images.cpp) **Dependency**: imread (opencv/modules/imgcodecs/src/loadsave.cpp)
Who?	Vadim Pisarevsky
When?	Nov 8, 2018
Why?	A practical layering violation to reuse existing image codec functions from imgcodecs for the CAP_IMAGES backend, avoiding duplication [16].

Video Analysis (video)

- Conceptual dependency: core, videoio
- Concrete Dependency: core, videoio, imgproc, calib3d

Discrepancy	imgproc
Which?	**Entity**: meanShift (opencv/modules/video/src/camshift.cpp) **Dependency**: moments (opencv/modules/imgproc/src/moments.cpp)
Who?	Ilya Lavrenov
When?	Dec 21, 2013
Why?	Refactoring - The commit was made to refactor the code in the file [17].

Key Findings & Rationale

Our findings showed that OpenCV's architecture has gone through natural expansion over time, well beyond what the conceptual diagram had exhibited. The application of the W4 analysis confirmed that many of these concrete dependencies are functional necessities driven by code reuse. For example, the features2d module depends heavily on the imgproc module; this is not accidental, but required for the cvtColor function used in the FastFeatureDetector, as well as the resize() function used in the FAST algorithm. Furthermore, we found that valid dependencies exist even when a module relies on a single function from another subsystem. A key example is calib3d, which maintains a dependency on features2d solely due to the usage of the FlannBasedMatcher within the RadiusSearchNeighborhoodGraphImpl function. These discoveries highlight that the concrete architecture is driven by pragmatic utility usage that connects modules more densely than the conceptual model anticipated.

Discrepancy Mitigation Proposal

To bridge the gap between the conceptual architecture and the actual implemented system, we propose a remediation strategy to align the models. There are various ways these discrepancies can be mitigated to create a more refined architecture:

1. Adding Strong Dependencies: Dependencies revealed during the concrete analysis that are functional requirements, such as the link between features2d and imgproc, must be added as they are intentional architectural decisions.
2. Trimming Trivial Dependencies: Less substantial dependencies, such as those arising from circular references in test files or temporary utility calls that do not define the subsystem's core behavior, should be removed from the high-level architecture.
3. Reconciling Conceptual Absences: Dependencies that were present in the original conceptual diagram but were not found in the concrete implementation (absences) should be removed.

By applying these changes, the resulting diagram Figure 5 represents a sample remediation that preserves readability of the architectural logic and OpenCVs complexity.

Diagrams

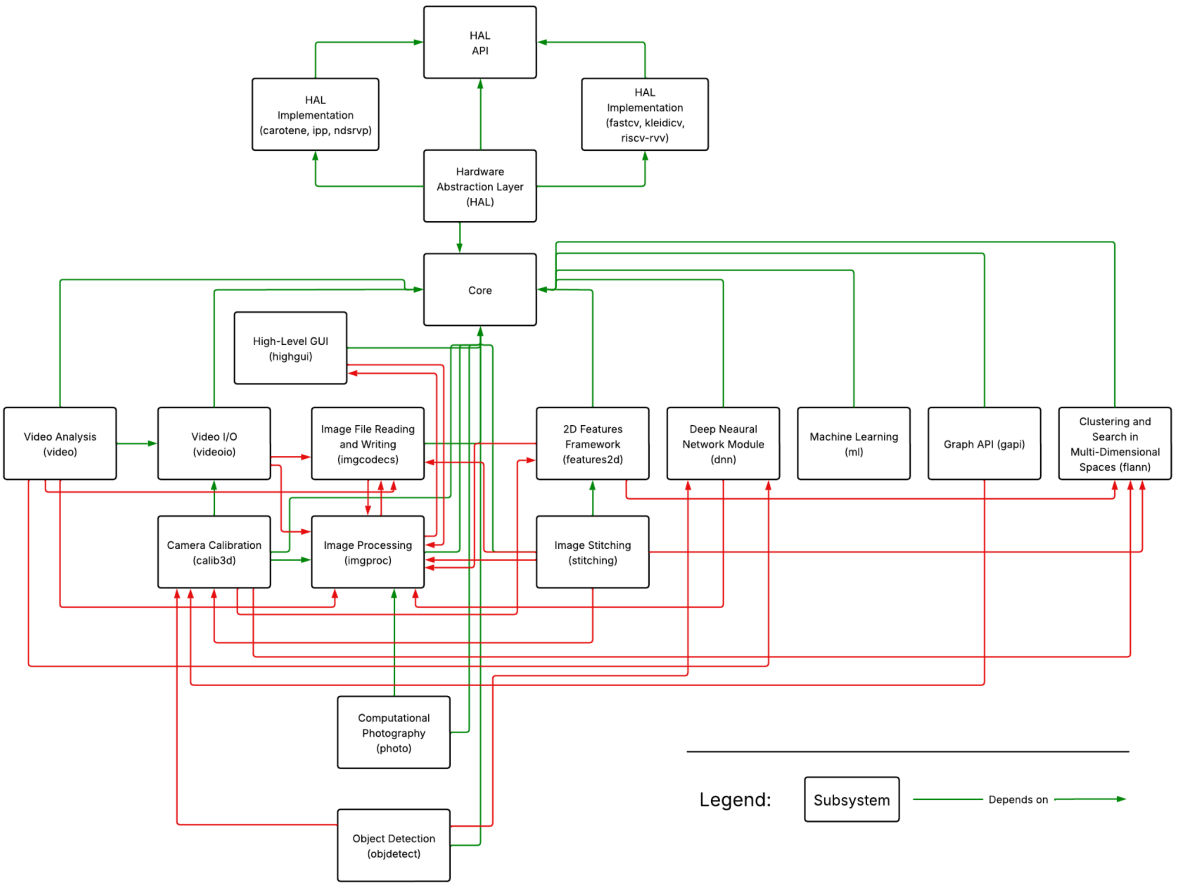

Figure 5 - Concrete Architecture Remediation

External Interfaces

The system's external interfaces remained consistent across the conceptual and concrete models.

Use Cases

Figure 6 illustrates a real-time face detection loop orchestrated by the Main Program. Initially, the system instantiates a CascadeClassifier (Object Detection) and a VideoCapture object (Video I/O) to access the camera. Inside the processing loop, the program captures a frame and passes it to the classifier, which converts it to grayscale via cvtColor before calling detectMultiScale to identify face coordinates. These coordinates are then passed to the Image Processing module to draw bounding boxes using rectangle(). Finally, the High-Level GUI renders the marked frame to the user via imshow().

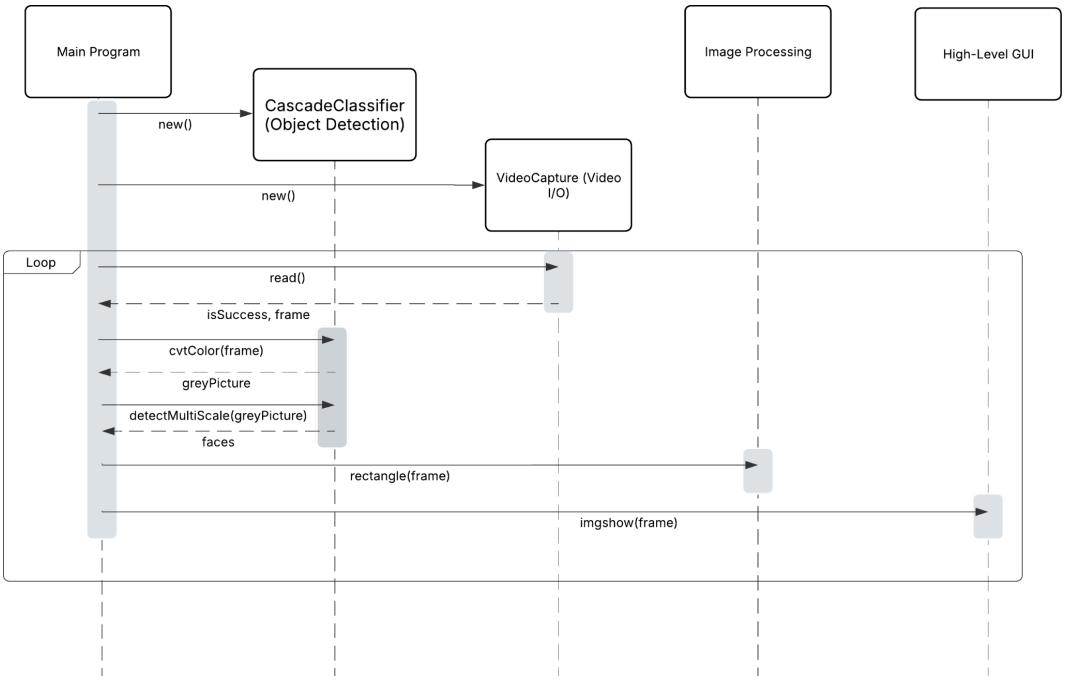

Figure 6 - Face Detection Workflow

Figure 7 depicts the transformation of raw architectural images into structured data. The Client Application first loads the image into a matrix using imread from the Image File Reading/Writing module. Control then shifts to the Image Processing module for a pipeline of operations: converting to grayscale (cvtColor), smoothing noise (GaussianBlur), and identifying structural edges (Canny). findContours then groups these edges into geometric shapes. Finally, the application processes this data for visualization via drawContours or exports the structured results back to disk using imwrite.

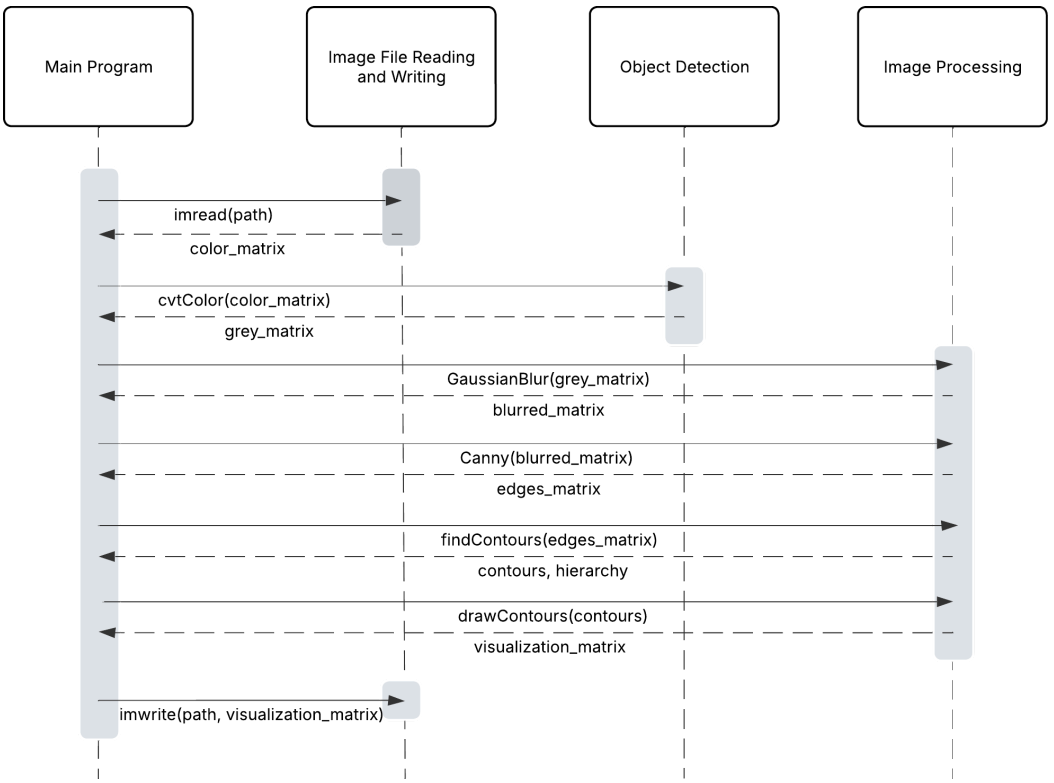

Figure 7 - Automated Architectural Feature Extraction via Contours

Data Dictionary

There are no terms needed to be defined that have not been defined in the report. This section is N/A.

Naming Conventions

There are no abbreviations or naming conventions used in this report that require further clarification. This section is N/A.

Conclusions

The comparison between the conceptual and concrete architectures revealed multiple discrepancies in how dependencies were originally understood. Through the use of reflexion analysis and the W4 method, our team identified which relationships were meaningful and which appeared less substantial upon closer inspection. This deeper investigation clarified dependency origins, exposed overlooked interactions, and showed that several assumed links in the conceptual model did not hold in practice. As a result, these insights led to necessary adjustments in the architectural model, ensuring that it more accurately reflects the system's real implementation.

Lessons Learned

When reflecting on the analysis process, several lessons became clear. Tools that generate concrete architectures often overestimate dependencies because they cannot determine the underlying reason for a connection in the code, which means a reported dependency may not represent a meaningful relationship at all. Similarly, modules that appear to be “leaf” components - those without incoming dependencies - are frequently more complex than expected once their internal structures are examined. Even dependencies involving only a small number of functions from another module can form strong connections, especially when those functions are used repeatedly or provide essential functionality required by the dependent module.

References

[1]“EECS4314_07_ReflexionModels” https://eclass.yorku.ca/pluginfile.php/7589465/mod_resource/content/1/EECS4314_07_ReflexionModels.pdf.

[2]“School Manager by Family Zone,” Github.com, 2025. https://github.com/opencv/opencv/commit/69e329c9fdf5807dc8edd15f2a587357a1d25e9b

[3]“OpenCV: Color Space Conversions,” *docs.opencv.org*. https://docs.opencv.org/4.x/d8/d01/group__imgproc__color__conversions.html

[4]opencv, “yet another attempt to refactor features2d; the first commit, feature… · opencv/opencv@2e91502,” *GitHub*, 2025. https://github.com/opencv/opencv/commit/2e915026a03e63acd6624d2f19e5876d6dbf4d5d

[5]opencv, “Blaming opencv/doc/js_tutorials/js_assets/js_dnn_example_helper.js at 5bec733bfb56ba39bda7b0de9430ae1f21f01342 · opencv/opencv,” *GitHub*, 2025. https://github.com/opencv/opencv/blame/5bec733bfb56ba39bda7b0de9430ae1f21f01342/doc/js_tutorials/js_assets/js_dnn_example_helper.js#L4

[6]opencv, “Merge pull request #12503 from nikhedonia:12500-move-gcgraph · opencv/opencv@e628fd7,” GitHub, 2025. https://github.com/opencv/opencv/commit/e628fd7bce2b5d64c36b5bdc55a37c0ae78bc907

[7]opencv, “Move C API of opencv_calib3d to separate file · opencv/opencv@e5a3372,” *GitHub*, 2025. https://github.com/opencv/opencv/commit/e5a33723fcabc6fc266c0ccf9b407de421da821f

[8]opencv, “Merge pull request #6933 from hrnr:gsoc_all · opencv/opencv@c17afe0,” *GitHub*, 2025. https://github.com/opencv/opencv/commit/c17afe0fab61bb11452b36fb94ea

[9]opencv, “Merge pull request #17683 from ivashmak:homography · opencv/opencv@a66f617,” *GitHub*, 2025. https://github.com/opencv/opencv/commit/a66f61748fc4861374b8dc458866a9614925350a

[10]opencv, “Merge pull request #22147 from rogday:zoom_factor · opencv/opencv@b91f173,” *GitHub,* 2025. https://github.com/opencv/opencv/commit/b91f173680af92498a8bd2517fc7c524c2eae640

[11]opencv, “reworked nearly all of the OpenCV tests (except for opencv_gpu tests)… · opencv/opencv@061b49e,” *GitHub*, 2025. https://github.com/opencv/opencv/commit/061b49e0b29b28a4d403fd6684c78ec63df272bf

[12]opencv, “Merge pull request #20422 from fengyuentau:dnn_face · opencv/opencv@34d359f,” *GitHub*, 2025. https://github.com/opencv/opencv/commit/34d359fe035a92d48a399c6e6975c77513bd5139

[13]opencv, “Added QR code decoding. · opencv/opencv@53ec8f2,” *GitHub*, 2025. https://github.com/opencv/opencv/commit/53ec8f286b7d475fa5a799023595bd4dc73e5c81

[14]opencv, “Merge pull request #10869 from savuor:color_cpp_split · opencv/opencv@64916d3,” *GitHub*, 2025. https://github.com/opencv/opencv/commit/64916d3d83595dbe075e4494cafe0d2c4b5744a6

[15]opencv, “trying to fix builds · opencv/opencv@4530c7a,” *GitHub*, 2025. https://github.com/opencv/opencv/commit/4530c7ad085bea62ec7f2262b0d7561c02db4336

[16]opencv, “removed C API in the following modules: photo, video, imgcodecs, vide… · opencv/opencv@11eafca,” *GitHub*, 2025. https://github.com/opencv/opencv/commit/11eafca3e2a4cbc62f1309d25db0ea3ed9a6ea8e#diff-8adb76e2da991cc09d60b6717da130069647e384a81d642ab6724bf5661350f9R53

[17]opencv, “refactored cv::CamShift and cv::meanShift · opencv/opencv@6083efc,” *GitHub*, 2025. https://github.com/opencv/opencv/commit/6083efc111ae8016500d3a1ad12c98ab768b270c