Final Project Report: OpenCV

by

Nikhil Kumar G and Raj Palival

Course Instructor: David Darian Muresan

Course Name: Software Architecture SSW565

STEVENS INSTITUTE OF TECHNOLOGY

Castle Point on Hudson

Hoboken, NJ 07030

May 15, 2023

Final Project Report: OpenCV

Nikhil Kumar G and Raj Palival
Course Instructor: David Darian Muresan
Course Name: Software Architecture SSW565
STEVENS INSTITUTE OF TECHNOLOGY
Castle Point on Hudson
Hoboken, NJ 07030

This document provides the requirements and design details of MyProject. The following table (Table 1) should be updated by authors whenever changes are made to the architecture design or new components are added.

Table 1: Document Update History

Date	**Updates**
05/15/2023	Architecture: • Added Architecture Introduction (Chapter 11) (Section 11.1). • Added Main Modules (Sub-Section 11.2.1). • Added Extra Modules (Sub-Section 11.2.2). • Added 15 Most Important Modules (Section 11.4). • Added 4+1 View of Architecture (Section 11.5). • Added Implementation View (Sub-Section 11.5.1). • Added Use Case View (Sub-Section 11.5.2). • Added Logical View (Sub-Section 11.5.3). • Added Process View (Sub-Section 11.5.4). • Added Deployment View (Sub-Section 11.5.5). • Added Brief Overview of 4+1 Architecture(Section 11.6).

Table 1: Document Update History

Date	**Updates**
04/25/2023	Risks: • Added Risks Introduction (Chapter 10) (Section 10.1). • Added Security Risks (Sub-Section 10.2.1). • Added Performance Risks (Sub-Section 10.2.2). • Added Usability Risks (Sub-Section 10.2.3). • Added Compatibility and Portability Risks (Sub-Section 10.2.4). • Added Documentation Risks (Sub-Section 10.2.5). • Added Accuracy and Robustness Risks (Sub-Section 10.2.6). • Added Scalability Risks (Sub-Section 10.2.7). • Added Data Privacy Risks (Sub-Section 10.2.8). • Added Community Engagement Risks (Sub-Section 10.2.9). • Added Integration Risks (Sub-Section 10.2.10).
04/15/2023	Quality Attributes: • Added Quality Attributes Introduction (Chapter 9) (Section 9.1). • Added Quality Attributes Scenario Introduction (Section 9.2). • Added a list of Significant Quality Attribute Scenarios (Section 9.3). • Added Availability QA (Sub-Section 9.3.1). • Added Deployability QA (Sub-Section 9.3.2). • Added Energy Efficiency QA (Sub-Section 9.3.3). • Added Integrability QA (Sub-Section 9.3.4). • Added Modifiability QA (Sub-Section 9.3.5). • Added Performance QA (Sub-Section 9.3.6). • Added Safety QA (Sub-Section 9.3.7). • Added Security QA (Sub-Section 9.3.8). • Added Testability QA (Sub-Section 9.3.9). • Added Usability QA (Sub-Section 9.3.10). • Added Evaluation of Business risk and Architectural Significance (Section 9.4). • Added Utility Tree (Section 9.5).
04/12/2023	User Requirements: • Added User Requirements Introduction (Chapter 8) (Section 8.1). • Added User Requirements (Section 8.2).
04/08/2023	Business Requirements: • Added Business Requirements Introduction (Chapter 7) (Section 7.1). • Added Business Requirements (Section 7.2).

Table 1: Document Update History

Date	**Updates**
04/07/2023	User Stories: • Added User Stories Introduction (Chapter 6) (Section 6.1). • Added User Stories (Section 6.2).
04/05/2023	Stakeholders: • Added Introduction (Chapter 5) (Section 5.1). • Added Types of Stakeholders (Section 5.2).
03/25/2023	Objective: • Added the objective (Chapter 4).
03/15/2023	Project Summary: • Added Project Summary (Chapter 3).
03/01/2023	Introduction: • Added the introduction (Chapter 1). • Added project proposal and its description (Chapter 2).

Table of Contents

List of Tables

List of Figures

Chapter 1

Team Members Introduction

– Nikhil Kumar G, Raj Palival

1.1 Nikhil Kumar G

I'm pursuing my masters in Software Engineering at SSE. I've always had a natural tendency to think very logically which has driven me to pursue this degree. Considering all the major ongoing technological advancements, I believe we can offer the future a lot more than we can imagine.
I have worked as an intern on ”Home automation and Security Systems” at a startup business ”Li2 Technologies”. After finishing my undergrad studies, I started working at ”Accenture”, where I spent more than a year creating Java, OIC, and SOA services.

1.2 Raj Palival

I am currently pursuing my masters in Software Engineering from Stevens Institute of Technology. My background work experience was working as a software devloper in Health sector domain, During my time here I have helped automate the front-end queries of our web application called 'GuidingCare' using Structured Rule Language in FICO Blaze system . I am looking forward to finish this degree with high honors and continue to work in computer science division as a software devloper and create meaningful applications.

Chapter 2

Project Proposal

– Nikhil Kumar G, Raj Palival

2.1 Open Source Project: Open CV

2.1.1 Description

The open source project that we have chosen for the project evaluation is ”OpenCV.org” [4]. As we have personally used Open CV for projects before, we recognize its importance and we are familiar with its functionality to go ahead with it.

Figure 2.1: Open CV Logo

OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. Being an Apache 2 licensed product, OpenCV makes it easy for businesses to utilize and modify the code [5].

Chapter 3

Summary

– Nikhil Kumar G, Raj Palival

The purpose of this document is to provide a overview of openCV's architecture and a comprehensive overview of the software architecture for OpenCV's Image Stitching module [6]. The document outlines the key stakeholders, user requirements, business requirements, quality attributes [7], and risks involved in the system, and provides a detailed description of the architecture from multiple perspectives. The document is intended to serve as a reference for developers, project managers, and other stakeholders who need to understand the system and its architecture. By providing a clear and detailed description of the system's architecture, the document helps ensure that the system is designed and implemented in a way that meets the needs and requirements of its stakeholders, while also addressing potential risks and quality issues. Overall, the purpose of this document is to facilitate effective communication and collaboration among stakeholders, and to ensure the successful development and deployment of the OpenCV Image Stitching module.

Overall, the 4+1 view model provides a comprehensive understanding of the OpenCV Image Stitching module's architecture from multiple perspectives, helping stakeholders to understand the system's functionality, behavior, development environment, physical architecture, and user requirements. This understanding is essential for ensuring that the system is designed and implemented in a way that meets the needs and requirements of its stakeholders, while also addressing potential risks and quality issues.

Chapter 4

Objective

– Nikhil Kumar G, Raj Palival

4.1 OpenCV

The objective of OpenCV architecture is to provide a flexible and modular framework for computer vision applications [8]. The architecture is designed to be highly customizable and extensible, allowing developers to easily integrate new functionalities and algorithms into their applications. The architecture also provides a set of core modules that cover a wide range of computer vision tasks, such as image processing, feature detection, object recognition, and machine learning.
These modules are designed to be efficient and optimized for performance, making them suitable for real-time applications. Additionally, the architecture provides a user-friendly interface and intuitive APIs that make it easy for developers to interact with the library and implement their computer vision projects. Overall, the objective of OpenCV architecture is to provide a powerful and flexible tool for computer vision applications that can be tailored to meet the specific needs and requirements of each project.

4.2 OpenCV Image-Stitching Module

The objective of the OpenCV image stitching module is to provide a set of tools and algorithms for stitching multiple images together to create a panoramic image. The module is designed to be efficient and optimized for performance, making it suitable for real-time applications. The module provides a set of core functionalities, such as feature detection, feature matching, and image blending, that are used to align and blend the input images into a seamless panoramic image.

The module also provides various stitching methods, such as pairwise stitching, multi-row stitching, and multi-band blending, that can be used to achieve different levels of quality and performance. Additionally, the module provides a user-friendly interface and intuitive APIs that make it easy for developers to interact with the library and implement their image stitching projects. Overall, the objective of the OpenCV image stitching module is to provide a powerful and flexible tool for creating panoramic images that can be tailored to meet the specific needs and requirements of each project.

Chapter 5

Stakeholders

– Nikhil Kumar G, Raj Palival

5.1 Introduction

In this chapter, we are going to learn about the stakeholders of Open CV. Stakeholders in general for OpenCV can be classified as people or roles who are affected, in some way, by OpenCV. Also, it's worth noting that stakeholders may vary over time as the OpenCV community and ecosystem evolve.

5.2 Types of Stakeholders

We will try to group the stakeholders of OpenCV into two broad categories:

1. OpenCV Builders
2. OpenCV Users

5.2.1 OpenCV Builders

These stakeholders are involved in developing, maintaining, and enhancing the OpenCV library. They contribute to its code-base, research and implement new algorithms, and ensure its continuous development. The potential stakeholders involved in building OpenCV are:

1. Developers
2. Contributors
3. OpenCV Foundation
4. Open-Source community

5.2.2 OpenCV Users

These stakeholders utilize the OpenCV library for various computer vision tasks and applications. They leverage the functionality provided by OpenCV in their projects, products, or services. They can be sub-divided into an exhaustive list. Hence, the potential stakeholders involved in using OpenCV are:

1. End-Users
2. Researchers
3. Academic institutions
4. Industry partners
5. Government agencies

STAKEHOLDERS	**DESCRIPTION**
Developers	Developers are the core team responsible for actively developing, maintaining, and enhancing the OpenCV library's source code and features.
Contributors	They are individuals or organizations who support the project by providing bug reports, feedback, documentation, tutorials, and code changes, helping to improve and expand its functionality.
OpenCV Foundation	The organization responsible for managing and overseeing the development and maintenance of the OpenCV library, ensuring its long-term sustainability and growth.
Open-Source Community	The broader community of developers, enthusiasts, and users who provide feedback, support, and contribute to the improvement and expansion of OpenCV.
End-Users	The ultimate beneficiaries or users of the project's outcome, who will interact with the OpenCV-based functionalities directly or indirectly.
Researchers	Professionals and academics who employ OpenCV in their research work, leveraging its computer vision algorithms and functionalities.
Academic Institutions	Universities and research institutes that incorporate OpenCV into their computer vision courses and projects, helping educate future computer vision professionals.
Industry Partners	Companies and organizations that integrate OpenCV into their products, services, or platforms, enabling computer vision capabilities.
Government Agencies	Public sector organizations that may utilize OpenCV for various purposes, such as surveillance, image analysis, or research initiatives.

Table 5.1: Stakeholders' description

Chapter 6

User Stories

– Nikhil Kumar G, Raj Palival

6.1 What is a User Story ?

A user story is a small, self-contained unit of development work designed to accomplish a specific goal within a product. A user story is usually written from the user's perspective and follows the format: "As [a user persona], I want [to perform this action] so that [I can accomplish this goal]."

User stories:

- Are easy for anyone to understand
- Represent bite-sized deliverables that can fit in sprints, whereas not all full features can.
- Help the team focus on real people, rather than abstract features
- Build momentum by giving development teams a feeling of progress

6.2 User Stories

1. As a developer, I want to use OpenCV to easily integrate computer vision capabilities into my software applications, so that I can create more advanced and intelligent applications that can analyze and interpret visual data.
2. As a contributor, I want to contribute to the development of OpenCV by submitting bug reports, fixing issues, and adding new features, so that I can help improve the quality and functionality of the software for the benefit of the community.
3. As the OpenCV foundation, we want to promote the use and development of OpenCV by providing resources, support, and guidance to developers, researchers, and industry partners, so that we can advance the field of computer vision and make it more accessible to everyone.

4. As a member of the open source community, I want to collaborate with other developers and contributors to improve OpenCV, share my knowledge and expertise, and help make computer vision more accessible and useful to everyone.

5. As an end-user, I want to use OpenCV to solve real-world problems, such as object recognition, face detection, and image processing, so that I can improve my productivity, efficiency, and quality of life.

6. As a researcher, I want to use OpenCV to conduct experiments, analyze data, and develop new algorithms and techniques in the field of computer vision, so that I can advance the state of the art and contribute to the scientific community [9].

7. As an academic institution, we want to use OpenCV to teach computer vision concepts and techniques to our students, conduct research, and collaborate with other institutions and industry partners, so that we can prepare our students for careers in this rapidly growing field [10].

8. As an industry partner, we want to use OpenCV to develop innovative products and services that leverage computer vision technology, so that we can improve our competitiveness, create new business opportunities, and provide value to our customers [11].

9. As a government agency, we want to use OpenCV to develop solutions that can help us address various challenges, such as public safety, transportation, and environmental monitoring, so that we can improve the quality of life for our citizens and promote economic growth.

Chapter 7

Business Requirements

– Nikhil Kumar G, Raj Palival

7.1 Introduction

Business requirements outline a project's needs and prerequisites for success while taking the target audience into account. It explains the rationale behind developing a certain project, who will use it, what advantages users will experience, and how the project's success will be measured. Business requirements do not specify how the project is to be developed [12].

7.1.1 What does a business requirement include ?

- Key objectives and identification of a problem.
- Benefits of the proposed solution.
- Project scope.
- Rules, regulations, and policies.
- Key features of the project.
- Performance and security features.
- Metrics to measure the success of the project.

7.1.2 What does a business requirement not include ?

- Details of the functional requirements of the system.
- Details of the implementation of functional or performance requirements.
- Details of how to implement policies and regulations.

7.2 Business Requirements

1. Open source: Develop and maintain an open source computer vision and machine learning software library that provides a common infrastructure for computer vision applications, which will allow developers to contribute to the project and make it better. This will also make OpenCV more accessible to a wider range of developers.

2. Real-time performance: OpenCV must be able to process images and videos in real time, which is essential for many applications such as facial recognition, object detection, and video surveillance.

3. Documentation: OpenCV must have comprehensive documentation, which will help developers learn how to use the library. This will make it easier for developers to get started with OpenCV and build their applications.

4. Image Processing: OpenCV should be able to process images of various formats, including JPEG, PNG, and BMP. It should be able to perform basic image processing operations such as resizing, cropping, and rotating images. Additionally, it should be able to apply filters such as blur, sharpen, and edge detection to images.

5. Hardware acceleration: OpenCV must support hardware acceleration, such as CUDA and OpenCL, to improve performance. This will allow developers to take advantage of the latest hardware to speed up their applications.

6. Support: OpenCV should have a strong support system that provides timely and effective assistance to users. It should have a large and active community forum where users can ask questions and share their experiences. It should also have a dedicated support team that can help users troubleshoot issues and provide solutions to problems.

7. Cross-platform compatibility: Develop and maintain a computer vision and machine learning software library that can be used on various platforms, including Windows, Linux, Android, and Mac OS. This will allow developers to use OpenCV on a variety of devices and systems.

8. Lowering costs: Enable businesses to utilize and modify the code for their specific needs, without incurring high licensing costs.

9. Multi-Language Support: Ensure compatibility with popular programming languages, including C++, Python, Java, and MATLAB.

10. Customization: OpenCV should be customizable to meet the specific needs of the business. It should allow users to create custom filters, algorithms, and models. It should also allow users to modify the user interface to suit their preferences and workflows.

Chapter 8

User Requirements

– Nikhil Kumar G, Raj Palival

8.1 What is a User Requirement?

In this chapter, we are going to learn about the 'User Requirements' of OpenCV. User requirements are detailed and specific statements that outline the functionality, constraints, and qualities expected from a system or product.

8.2 User Requirements of OpenCV

Users often have certain expectations and needs when utilizing OpenCV in their projects. Some common user requirements of OpenCV can include:

User requirements	**Description**
Versatile Image and Video Processing	Users expect OpenCV to provide a wide range of image and video processing capabilities, such as image filtering, resizing, color space conversions, object detection, and tracking.
Efficient and Optimized Algorithms	Users require efficient and optimized algorithms implemented in OpenCV, enabling fast and accurate computer vision operations even on resource-constrained devices.
Robust and Accurate Functions	Users depend on OpenCV to deliver reliable and accurate computer vision functions, such as feature extraction, pattern recognition, image segmentation, and optical flow analysis.
Documentation and Support	Users look for comprehensive documentation, tutorials, and examples that help them understand and effectively use the OpenCV library. They also appreciate active community support, forums, and resources that assist in troubleshooting and problem-solving.
Cross-Platform Compatibility	Users expect OpenCV to work seamlessly across different platforms and operating systems, including Windows, macOS, Linux, Android, and iOS.

Integration with Libraries and Frameworks	Users often require OpenCV to integrate smoothly with other libraries and frameworks commonly used in computer vision and machine learning, such as NumPy, TensorFlow, and PyTorch.
Flexibility and Customization	Users appreciate the ability to customize and extend OpenCV to suit their specific project requirements, including the option to add new algorithms, modify existing functions, or interface with specialized hardware.
Performance Optimization Tools	Users often require tools and techniques to optimize the performance of OpenCV-based applications, including multi-threading, GPU acceleration, and code profiling.
License Compatibility	Users may have specific requirements regarding the licensing of OpenCV to ensure compliance with their project's open-source or commercial license obligations.
Real-Time Camera	Users require OpenCV to provide seamless integration with cameras and enable real-time capture and processing of camera input.

Table 8.1: User Requirements

These user stories demonstrate the diverse range of users and their specific needs when utilizing OpenCV for various applications, spanning from research and development to commercial projects and personal interests.

Chapter 9

Quality Attributes

– Nikhil Kumar G, Raj Palival

9.1 Introduction

Quality is never an accident; it is always the result of high intention, sincere effort, intelligent direction and skillful execution.

- William A. Foster

Non-functional requirements (NFRs) define the criteria that are used to evaluate the whole system, but not for a specific behavior, and are also called quality attributes and described in detail in architectural specifications.
All NFRs can be divided into two main categories:

1. NFRs that affect system behavior, design, and user interface during work.

2. NFRs that affect the development and support of the system.

9.2 Quality Attribute Scenario

A common form to specify all QA requirements as scenarios. The common form is testable and unambiguous; Thus it provides regularity in how we treat all quality attributes.
Quality attribute scenarios have six parts:

1. Source of stimulus: This is some entity (a human, a computer system, or any other actuator) that generated the stimulus.

2. Stimulus: The stimulus is a condition that needs to be considered when it arrives at a system.

3. Artifact: Some artifact is stimulated. This may be the whole system or some pieces of it.

4. Environment: The stimulus occurs within certain conditions. The system may be in an overload condition or may be running when the stimulus occurs, or some other condition may be true.

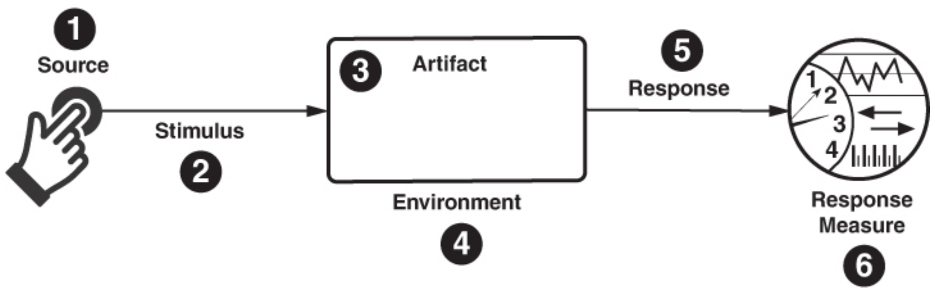


Figure 9.1: The parts of a quality attribute scenario.

5. Response: The response is the activity undertaken after the arrival of the stimulus.
6. Response measure: When the response occurs, it should be measurable in some fashion so that the requirement can be tested.

9.3 List of Significant Quality Attribute Scenarios

9.3.1 Availability

1. Source of stimulus: A user requests to stitch two images together.
2. Stimulus: The user clicks on a button to start the stitching process.
3. Artifact: The two images that the user wants to stitch together.
4. Environment: The OpenCV Image Stitcher module is running on a computer.
5. Response: The Image Stitcher module stitches the two images together and displays the result.
6. Response measure:
   - The response measure is the time it takes to stitch the two images together.
   - The percentage of time that the library is able to process requests and return results without any errors or downtime.
   - This can be measured using metrics such as uptime, response time, and error rate.

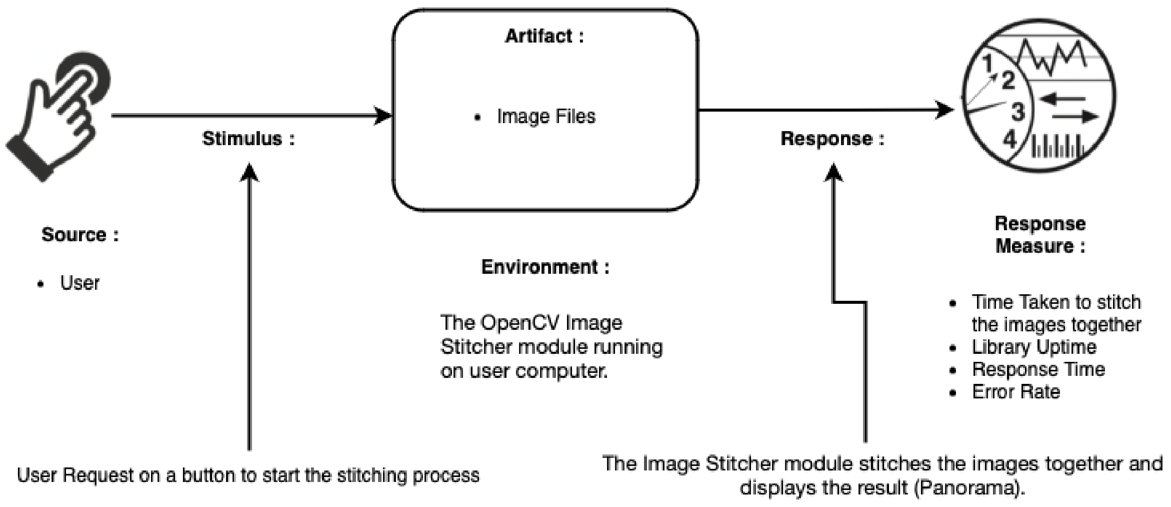

Artifact :
Image Files
Stimulus :
Response :
Source :
User
Environment :
The OpenCV Image Stitcher module running on user computer.
Response Measure :
Time Taken to stitch the images together
Library Uptime
Response Time
Error Rate
User Request on a button to start the stitching process
The Image Stitcher module stitches the images together and displays the result (Panorama).

Figure 9.2: Quality Attribute Scenario for Availability [1].

A high availability score indicates that OpenCV is able to handle a large volume of requests and provide reliable results to users, ensuring that the application is always available and responsive. In this scenario, the availability quality attribute is met if the Image Stitcher module is able to stitch the two images together in a timely manner. If the Image Stitcher module is not able to stitch the images together, or if it takes a long time to do so, then the availability quality attribute is not met.

9.3.2 Deployability

1. Source of stimulus: A software developer wants to deploy the OpenCV Image Stitcher module in a new application.
2. Stimulus: The developer downloads the OpenCV Image Stitcher module from the OpenCV website.
3. Artifact: The OpenCV Image Stitcher module is a C++ library.
4. Environment: The developer's development environment must have a C++ compiler and the OpenCV libraries installed.
5. Response: The developer compiles the OpenCV Image Stitcher module and links it to their application.
6. Response measure: The response measure is the time it takes to deploy the OpenCV Image Stitcher module in the new application.

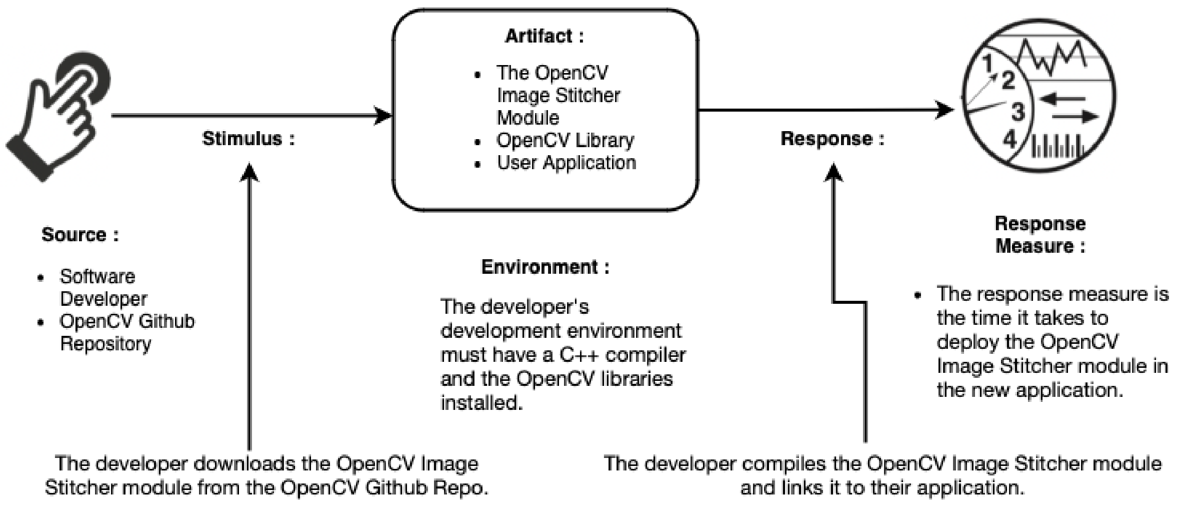

Artifact :
The OpenCV Image Stitcher Module
OpenCV Library
User Application
Stimulus :
Response :
Source :
Software Developer
OpenCV Github Repository
Environment :
The developer's development environment must have a C++ compiler and the OpenCV libraries installed.
Response Measure :
The response measure is the time it takes to deploy the OpenCV Image Stitcher module in the new application.
The developer downloads the OpenCV Image Stitcher module from the OpenCV Github Repo.
The developer compiles the OpenCV Image Stitcher module and links it to their application.

Figure 9.3: Quality Attribute Scenario for Deployability.

In this scenario, the deployability quality attribute is met if the OpenCV Image Stitcher module can be deployed in the new application in a timely manner. If the OpenCV Image Stitcher module cannot be deployed, or if it takes a long time to do so, then the deployability quality attribute is not met.

9.3.3 Energy Efficiency

1. Source of stimulus: A user wants to use the OpenCV Image Stitcher module to stitch two images together on a mobile device.
2. Stimulus: The user opens the application that uses the OpenCV Image Stitcher module.
3. Artifact: The two images that the user wants to stitch together.
4. Environment: The mobile device has a limited battery life.
5. Response: The OpenCV Image Stitcher module stitches the two images together and displays the result.
6. Response measure: The response measure is the amount of energy consumed by the mobile device while the OpenCV Image Stitcher module is running.

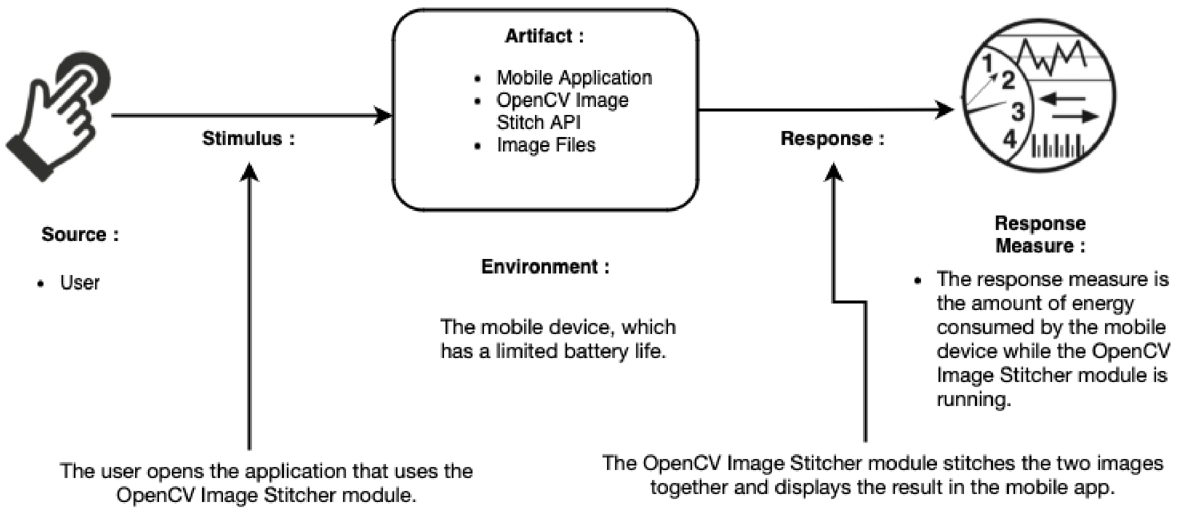

Artifact :
Mobile Application
OpenCV Image Stitch API
Image Files
Stimulus :
Response :
Source :
User
Environment :
The mobile device, which has a limited battery life.
Response Measure :
The response measure is the amount of energy consumed by the mobile device while the OpenCV Image Stitcher module is running.
The user opens the application that uses the OpenCV Image Stitcher module.
The OpenCV Image Stitcher module stitches the two images together and displays the result in the mobile app.

Figure 9.4: Quality Attribute Scenario for Energy Efficiency.

In this scenario, the energy efficiency quality attribute is met if the OpenCV Image Stitcher module is able to stitch the two images together while consuming as little energy as possible. If the OpenCV Image Stitcher module consumes a lot of energy, then the battery life of the mobile device will be reduced.

9.3.4 Integrability

1. Source of stimulus: Development team
2. Stimulus: The development team needs to integrate openCV with a new software component.
3. Artifact: openCV software package, new component
4. Environment: Development environment with the new software component and its associated hardware and software.
5. Response: The openCV software package is successfully integrated with the new software component and is fully functional.
6. Response Measure: The development team measures the time it takes to integrate openCV with the new software component, the number of errors encountered during integration, and the ease of using openCV with the new software component.

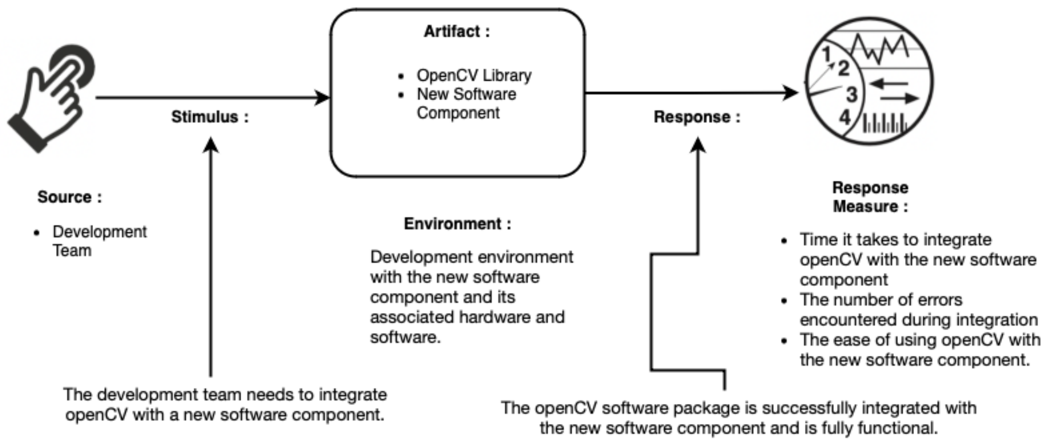

Artifact :
• OpenCV Library
• New Software Component
Stimulus :
Response :
Source :
• Development Team
Environment :
Development environment with the new software component and its associated hardware and software.
Response Measure :
• Time it takes to integrate openCV with the new software component
• The number of errors encountered during integration
• The ease of using openCV with the new software component.
The development team needs to integrate openCV with a new software component.
The openCV software package is successfully integrated with the new software component and is fully functional.

Figure 9.5: Quality Attribute Scenario for Integrability.

Additionally, the team may also measure the impact of the integration on the performance of the new software component and the overall system. The goal is to minimize integration time, errors, and impact on performance, while maximizing ease of use and compatibility with the new software component.

9.3.5 Modifiability

1. Source of stimulus: Product owner
2. Stimulus: The product owner requests a new feature to be added to openCV.
3. Artifact: openCV software package, new feature
4. Environment: Development environment with the necessary hardware and software.
5. Response: The development team modifies the openCV software package to include the new feature and ensures that it is fully functional.
6. Response measure: The development team measures the time it takes to modify openCV to include the new feature, the number of errors encountered during modification, and the impact of the modification on the performance of openCV.

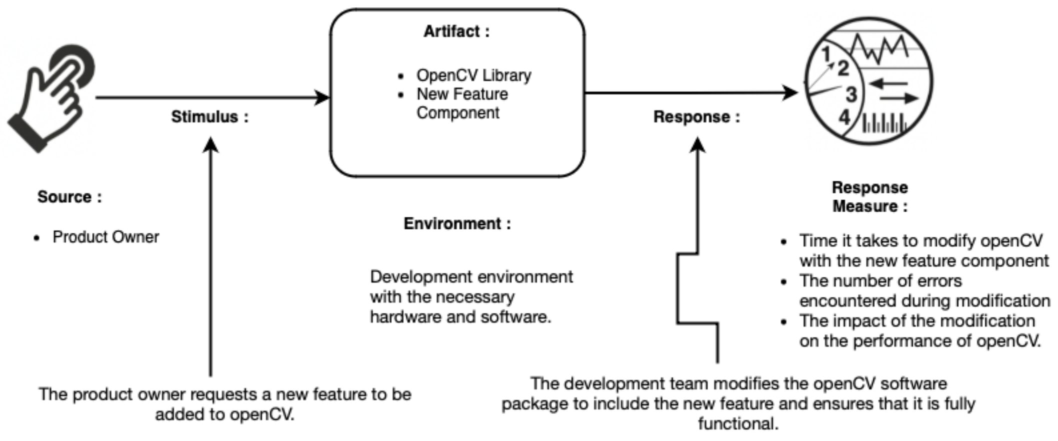

Artifact :
• OpenCV Library
• New Feature Component
Stimulus :
Response :
Source :
• Product Owner
Environment :
Development environment with the necessary hardware and software.
Response Measure :
• Time it takes to modify openCV with the new feature component
• The number of errors encountered during modification
• The impact of the modification on the performance of openCV.
The product owner requests a new feature to be added to openCV.
The development team modifies the openCV software package to include the new feature and ensures that it is fully functional.

Figure 9.6: Quality Attribute Scenario for Modifiability.

Additionally, the team may also measure the ease of maintaining the modified code and the impact of the modification on the overall system. The goal is to minimize modification time, errors, and impact on performance, while maximizing ease of maintenance and compatibility with the overall system.

9.3.6 Performance

1. Source of stimulus: User
2. Stimulus: The user requests openCV to process a large image dataset.
3. Artifact: openCV software package, image dataset, image library
4. Environment: Production environment with the necessary hardware and software.
5. Response: openCV processes the large image dataset within an acceptable time frame and with minimal errors.
6. Response measure: The user measures the time it takes for openCV to process the large image dataset, the number of errors encountered during processing, and the resource utilization of the system during processing.

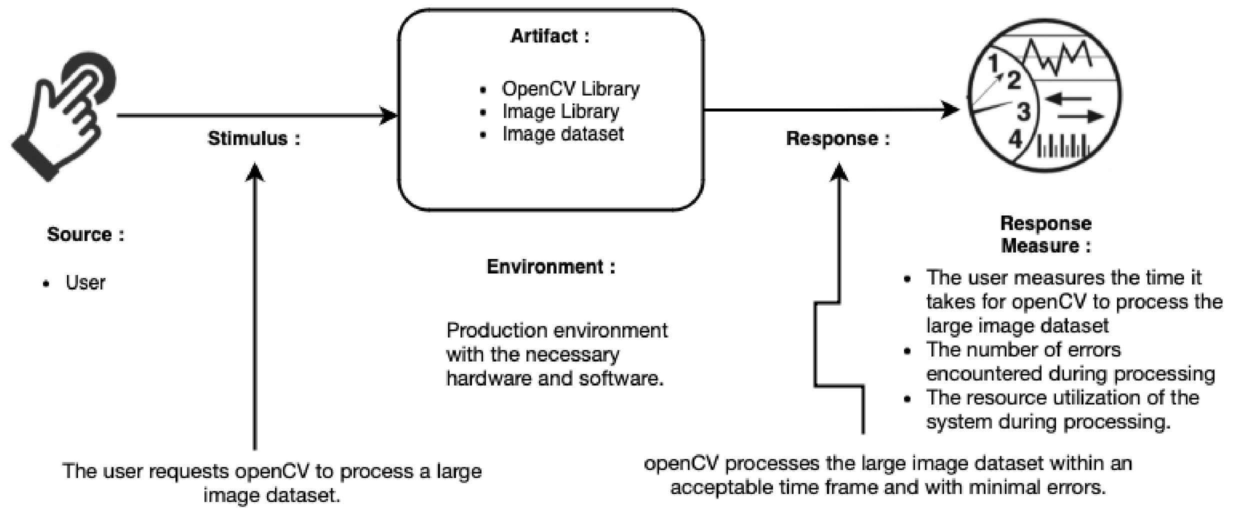

Artifact :
• OpenCV Library
• Image Library
• Image dataset
Stimulus :
Response :
Source :
• User
Environment :
Production environment with the necessary hardware and software.
Response Measure :
• The user measures the time it takes for openCV to process the large image dataset
• The number of errors encountered during processing
• The resource utilization of the system during processing.
The user requests openCV to process a large image dataset.
openCV processes the large image dataset within an acceptable time frame and with minimal errors.

Figure 9.7: Quality Attribute Scenario for Performance.

Additionally, the user may also measure the accuracy of the processing results and the ease of integrating openCV with other software components in the production environment. The goal is to minimize processing time, errors, and resource utilization, while maximizing accuracy and ease of integration with other software components.

9.3.7 Safety

1. Source of stimulus: Self-driving car system.
2. Stimulus: The self-driving car system requests openCV to detect objects on the road and make decisions based on the detected objects.
3. Artifact: openCV software package, car camera.
4. Environment: Production environment in a self-driving car with the necessary hardware and software.
5. Response: openCV accurately detects objects on the road and provides the self-driving car system with the necessary information to make safe driving decisions.
6. Response measure: The self-driving car system measures the accuracy of openCV's object detection on the road, the number of false positives or false negatives encountered, and the impact of openCV's operation on the safety of the passengers and other vehicles on the road.

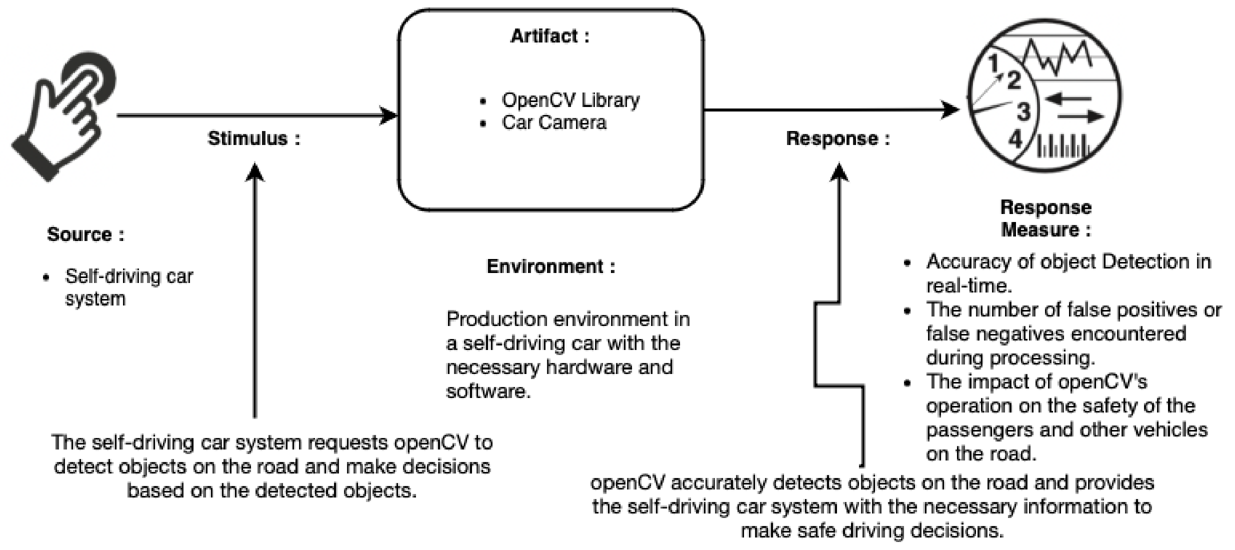

Artifact :
• OpenCV Library
• Car Camera
Stimulus :
Response :
Source :
• Self-driving car system
Environment :
Production environment in a self-driving car with the necessary hardware and software.
Response Measure :
• Accuracy of object Detection in real-time.
• The number of false positives or false negatives encountered during processing.
• The impact of openCV's operation on the safety of the passengers and other vehicles on the road.
The self-driving car system requests openCV to detect objects on the road and make decisions based on the detected objects.
openCV accurately detects objects on the road and provides the self-driving car system with the necessary information to make safe driving decisions.

Figure 9.8: Quality Attribute Scenario for Safety.

Additionally, the system may also measure the ease of integrating openCV with other safety systems in the self-driving car. The goal is to maximize the accuracy of object detection while minimizing any safety hazards or risks associated with openCV's operation in the self-driving car.

9.3.8 Security

1. Source of stimulus: Unauthorized user
2. Stimulus: The unauthorized user attempts to gain access to sensitive information or disrupt the operation of openCV.
3. Artifact: openCV software package, Security log report
4. Environment: Production environment with the necessary hardware and software.
5. Response: openCV detects and prevents the unauthorized user's attempts to gain access to sensitive information or disrupt the system's operation and does not compromise the security of the system or any sensitive information.
6. Response measure: The system measures the number of successful and unsuccessful attempts by the unauthorized user to gain access to sensitive information or disrupt the system's operation, the impact of the unauthorized user's attempts on the security of the system and any sensitive information, and the effectiveness of openCV's security measures in preventing the unauthorized user's attempts.

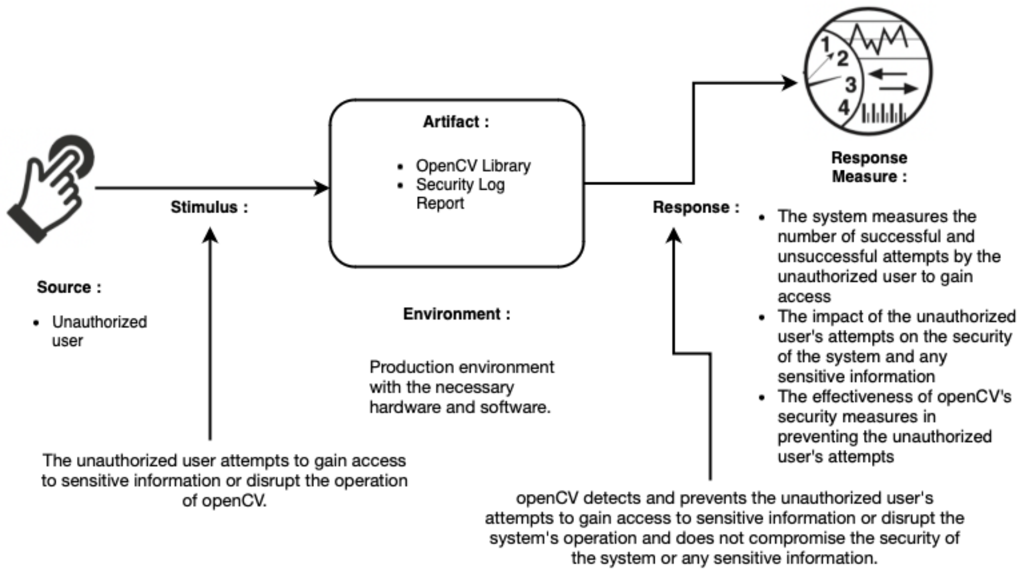

Artifact :
• OpenCV Library
• Security Log Report
Stimulus :
Source :
• Unauthorized user
Environment :
Production environment with the necessary hardware and software.
The unauthorized user attempts to gain access to sensitive information or disrupt the operation of openCV.
Response :
openCV detects and prevents the unauthorized user's attempts to gain access to sensitive information or disrupt the system's operation and does not compromise the security of the system or any sensitive information.
Response Measure :
• The system measures the number of successful and unsuccessful attempts by the unauthorized user to gain access
• The impact of the unauthorized user's attempts on the security of the system and any sensitive information
• The effectiveness of openCV's security measures in preventing the unauthorized user's attempts
1
2
3
4

Figure 9.9: Quality Attribute Scenario for Security.

Additionally, the system may also measure the ease of maintaining and updating openCV's security measures to address any new vulnerabilities or threats. The goal is to minimize the number of successful unauthorized access attempts and their impact on the security of the system and any sensitive information.

9.3.9 Testability

1. Source of stimulus: Testing team
2. Stimulus: The testing team needs to test the accuracy and performance of the openCV Image Stitching module in a real-world scenario.
3. Artifact: openCV Image Stitching module, Image files
4. Environment: Real-world environment with the necessary hardware and software on tester's computer.
5. Response: The openCV Image Stitching module accurately stitches images in the real-world scenario and performs within acceptable performance parameters.
6. Response measure: The testing team measures the accuracy of the openCV Image Stitching module in stitching images in the real-world scenario, the number of errors encountered during stitching, and the performance of the module in terms of processing time and resource utilization. Test-cases pass percentage for unique scenarios. Test Coverage percentage of openCV Image Stitching module.

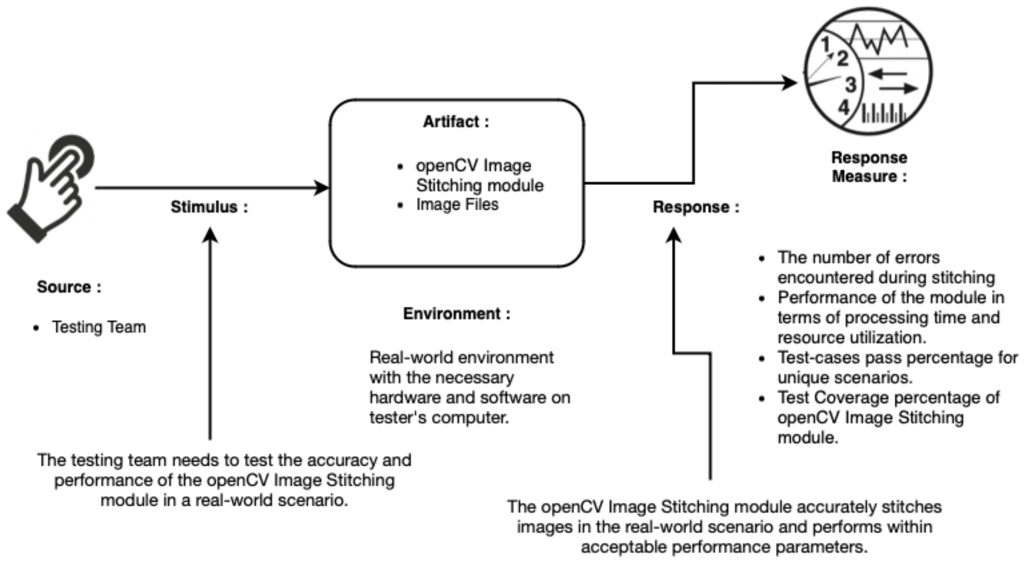

Artifact :
openCV Image Stitching module
Image Files
Stimulus :
Response :
Response Measure :
Source :
Testing Team
Environment :
Real-world environment with the necessary hardware and software on tester's computer.
The number of errors encountered during stitching
Performance of the module in terms of processing time and resource utilization.
Test-cases pass percentage for unique scenarios.
Test Coverage percentage of openCV Image Stitching module.
The testing team needs to test the accuracy and performance of the openCV Image Stitching module in a real-world scenario.
The openCV Image Stitching module accurately stitches images in the real-world scenario and performs within acceptable performance parameters.

Figure 9.10: Quality Attribute Scenario for Testability.

Additionally, the team may also measure the ease of testing the openCV Image Stitching module in the real-world scenario and the effectiveness of any testing tools or frameworks used. The goal is to maximize the accuracy and performance of the openCV Image Stitching module in the real-world scenario while minimizing any testing time and effort required.

9.3.10 Usability

1. Source of stimulus: User
2. Stimulus: The user needs to use openCV to perform image processing tasks.
3. Artifact: openCV software package, image files
4. Environment: User's environment with the necessary hardware and software.
5. Response: The user is able to use openCV to perform image processing tasks with ease and efficiency.
6. Response measure: The user measures the ease of learning and using openCV to perform image processing tasks, the time it takes to complete the tasks, and the user satisfaction with the user interface and overall usability of openCV.

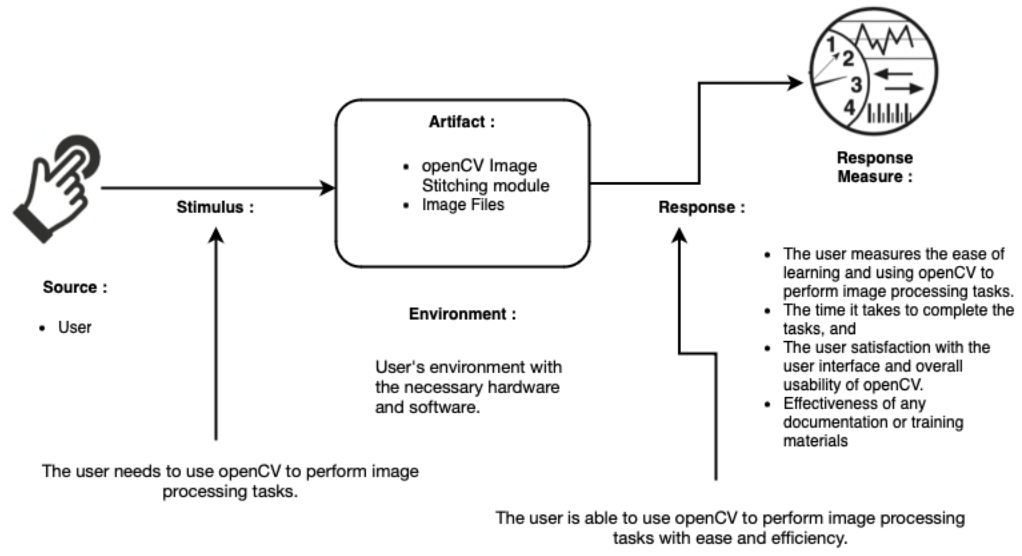

Artifact :
• openCV Image Stitching module
• Image Files
Stimulus :
Source :
• User
Environment :
User's environment with the necessary hardware and software.
Response :
Response Measure :
• The user measures the ease of learning and using openCV to perform image processing tasks.
• The time it takes to complete the tasks, and
• The user satisfaction with the user interface and overall usability of openCV.
• Effectiveness of any documentation or training materials
The user needs to use openCV to perform image processing tasks.
The user is able to use openCV to perform image processing tasks with ease and efficiency.

Figure 9.11: Quality Attribute Scenario for Usability.

Additionally, the user may also measure the effectiveness of any documentation or training materials provided with openCV. The goal is to maximize the ease of use and efficiency of openCV for image processing tasks, while minimizing the time and effort required to learn and use the software.

9.4 Evaluation of Business risk and Architectural Significance

Quality Attribute	Business Value (a)	Effect on Architecture (b)
Availability	H	M
Deployability	M	M
Energy Efficiency	L	M
Integrability	M	M
Modifiability	H	M
Performance	H	H
Safety	H	H
Security	H	H
Testability	H	M
Usability	M	M

Table 9.1: Quality Attribute Evaluation

9.4.1 Understanding the values

Grading Scale Used:

Rating: (a, b):
a: the ASR's business value(importance)
b: the effect on the architecture(difficulty)

Key:
H=high
M=medium
L=low

1. Availability: openCV is a mature library with a large community of users and developers. There are many different ways to install and use openCV, which makes it easy to deploy in a variety of environments. This is why we rated availability as high.

2. Deployability: openCV is available for a variety of platforms, including Windows, Linux, and macOS. There are also many different ways to deploy openCV, such as as a static library, a shared library, or a Docker image. This is why we rated deployability as medium.

3. Energy Efficiency: openCV uses a variety of techniques to optimize its performance, such as using hardware acceleration and image compression. This is why we rated energy efficiency as low.

4. Integrability: openCV provides a variety of APIs, such as C++, Python, and Java, which make it easy to integrate with other languages and frameworks. This is why we rated integrabilty as medium.

5. Modifiability: openCV is a open source library, which means that the source code is available for anyone to modify. This makes it easy to customize openCV to meet the specific needs of a project. This is why we rated modifiability as high.

6. Performance: openCV is designed to be fast and efficient, even on resource-constrained devices. This is why we rated performance as high.

7. Safety: openCV is designed to be secure and reliable, even in critical applications. This is why I rated safety as high.

8. Security: openCV is designed to be resistant to attack and intrusion. This is why we rated security as high.

9. Testability: openCV provides a variety of tools and resources to help developers test their code. This makes it easy to ensure that openCV is working properly before it is deployed in production. This is why we rated testability as high.

10. Usability: openCV provides a variety of documentation and tutorials to help developers get started. This makes it easy for anyone to use openCV, regardless of their experience level. This is why we rated usability as medium.

9.5 Utiltity Tree

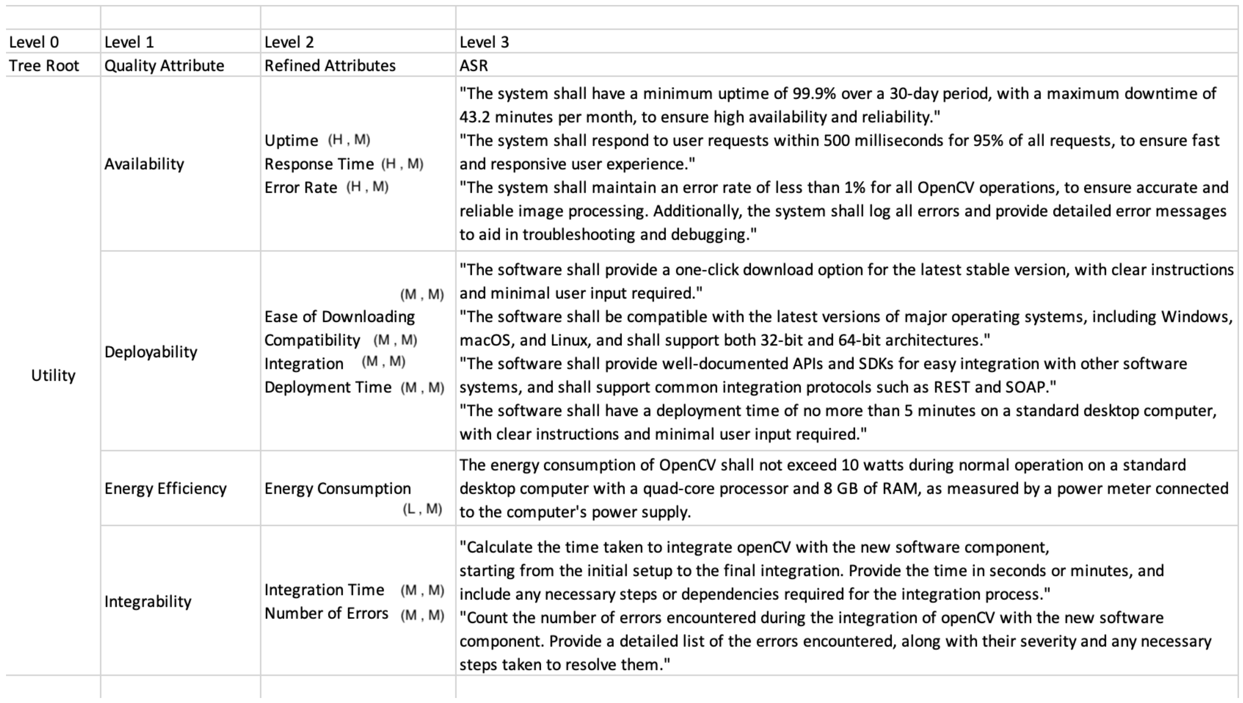
9.5.1 Introduction

A utility tree starts with the root node ”Utility” representing the overall ”goodness” of the system. Major Quality Attributes (QAs) are listed under the root node. Each QA is then further refined with specific aspects relevant to the system. These refinements can be broken down into specific Quality Attribute Scenarios (ASRs).

ASRs are placed as leaves on the tree and evaluated based on business value and technical risk. Business value is assessed as high, medium, or low, indicating the importance of meeting the requirement. Technical risk is evaluated as high, medium, or low, reflecting the level of concern and confidence in meeting the ASR.

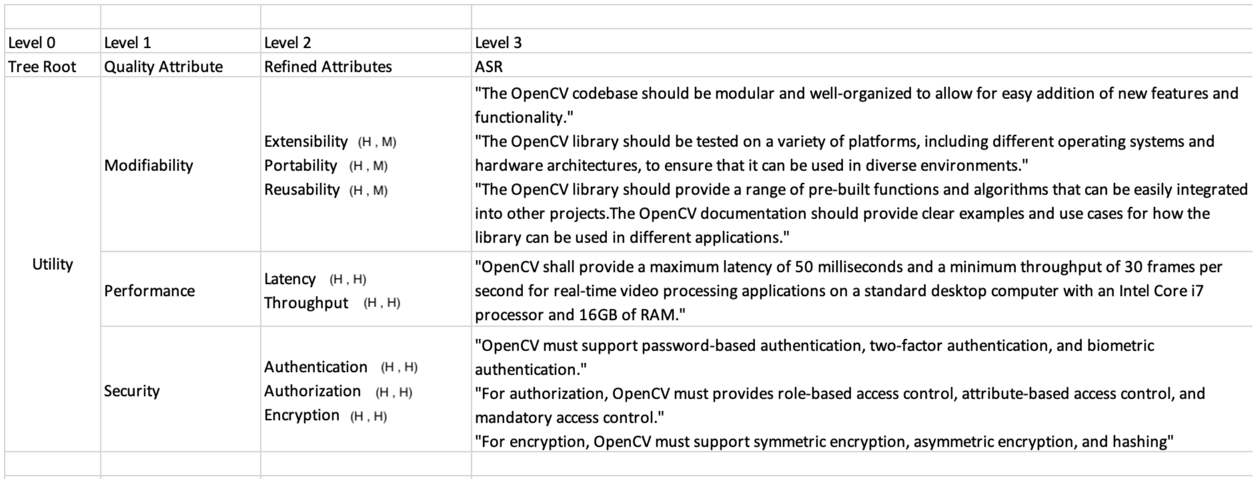
In summary, a utility tree outlines the system’s QAs, their refinements, and ASRs as scenarios. These scenarios are then evaluated based on business value and technical risk.

9.5.2 Tabular form of Utility Tree



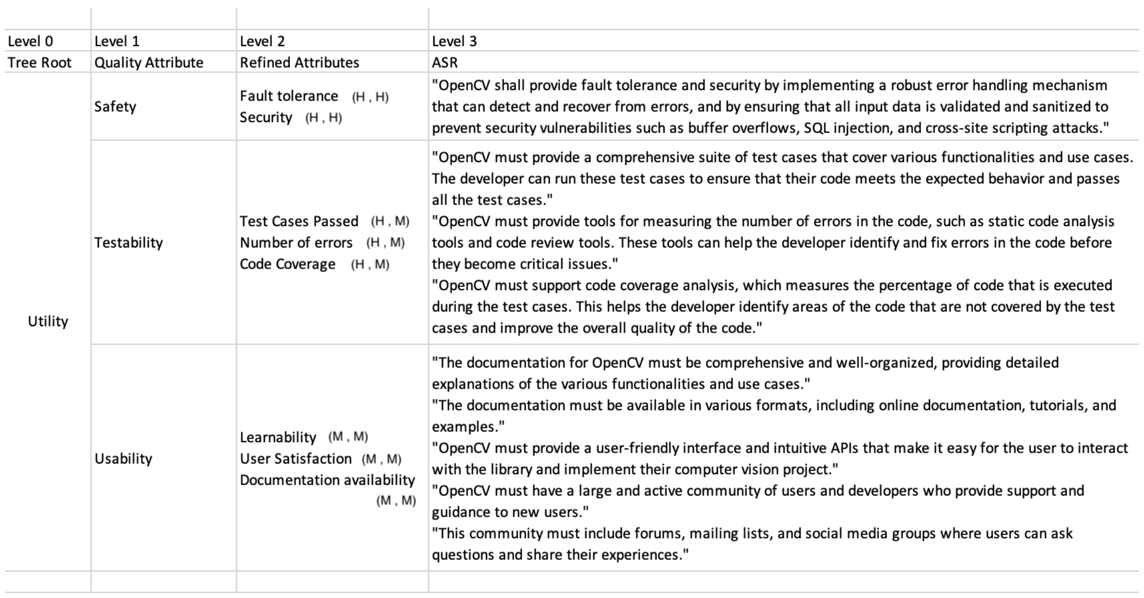
Level 0	Level 1	Level 2	Level 3
Tree Root	Quality Attribute	Refined Attributes	ASR
Utility	Availability	Uptime (H , M) Response Time (H , M) Error Rate (H , M)	"The system shall have a minimum uptime of 99.9% over a 30-day period, with a maximum downtime of 43.2 minutes per month, to ensure high availability and reliability." "The system shall respond to user requests within 500 milliseconds for 95% of all requests, to ensure fast and responsive user experience." "The system shall maintain an error rate of less than 1% for all OpenCV operations, to ensure accurate and reliable image processing. Additionally, the system shall log all errors and provide detailed error messages to aid in troubleshooting and debugging."
	Deployability	(M , M) Ease of Downloading Compatibility (M , M) Integration (M , M) Deployment Time (M , M)	"The software shall provide a one-click download option for the latest stable version, with clear instructions and minimal user input required." "The software shall be compatible with the latest versions of major operating systems, including Windows, macOS, and Linux, and shall support both 32-bit and 64-bit architectures." "The software shall provide well-documented APIs and SDKs for easy integration with other software systems, and shall support common integration protocols such as REST and SOAP." "The software shall have a deployment time of no more than 5 minutes on a standard desktop computer, with clear instructions and minimal user input required."
	Energy Efficiency	Energy Consumption (L , M)	The energy consumption of OpenCV shall not exceed 10 watts during normal operation on a standard desktop computer with a quad-core processor and 8 GB of RAM, as measured by a power meter connected to the computer's power supply.
	Integrability	Integration Time (M , M) Number of Errors (M , M)	"Calculate the time taken to integrate openCV with the new software component, starting from the initial setup to the final integration. Provide the time in seconds or minutes, and include any necessary steps or dependencies required for the integration process." "Count the number of errors encountered during the integration of openCV with the new software component. Provide a detailed list of the errors encountered, along with their severity and any necessary steps taken to resolve them."

Figure 9.12: Utility Tree - 1.



Level 0	Level 1	Level 2	Level 3
Tree Root	Quality Attribute	Refined Attributes	ASR
Utility	Modifiability	Extensibility (H , M) Portability (H , M) Reusability (H , M)	"The OpenCV codebase should be modular and well-organized to allow for easy addition of new features and functionality." "The OpenCV library should be tested on a variety of platforms, including different operating systems and hardware architectures, to ensure that it can be used in diverse environments." "The OpenCV library should provide a range of pre-built functions and algorithms that can be easily integrated into other projects.The OpenCV documentation should provide clear examples and use cases for how the library can be used in different applications."
	Performance	Latency (H , H) Throughput (H , H)	"OpenCV shall provide a maximum latency of 50 milliseconds and a minimum throughput of 30 frames per second for real-time video processing applications on a standard desktop computer with an Intel Core i7 processor and 16GB of RAM."
	Security	Authentication (H , H) Authorization (H , H) Encryption (H , H)	"OpenCV must support password-based authentication, two-factor authentication, and biometric authentication." "For authorization, OpenCV must provides role-based access control, attribute-based access control, and mandatory access control." "For encryption, OpenCV must support symmetric encryption, asymmetric encryption, and hashing"

Figure 9.13: Utility Tree - 2.



Level 0	Level 1	Level 2	Level 3
Tree Root	Quality Attribute	Refined Attributes	ASR
Utility	Safety	Fault tolerance (H , H) Security (H , H)	"OpenCV shall provide fault tolerance and security by implementing a robust error handling mechanism that can detect and recover from errors, and by ensuring that all input data is validated and sanitized to prevent security vulnerabilities such as buffer overflows, SQL injection, and cross-site scripting attacks."
	Testability	Test Cases Passed (H , M) Number of errors (H , M) Code Coverage (H , M)	"OpenCV must provide a comprehensive suite of test cases that cover various functionalities and use cases. The developer can run these test cases to ensure that their code meets the expected behavior and passes all the test cases." "OpenCV must provide tools for measuring the number of errors in the code, such as static code analysis tools and code review tools. These tools can help the developer identify and fix errors in the code before they become critical issues." "OpenCV must support code coverage analysis, which measures the percentage of code that is executed during the test cases. This helps the developer identify areas of the code that are not covered by the test cases and improve the overall quality of the code."
	Usability	Learnability (M , M) User Satisfaction (M , M) Documentation availability (M , M)	"The documentation for OpenCV must be comprehensive and well-organized, providing detailed explanations of the various functionalities and use cases." "The documentation must be available in various formats, including online documentation, tutorials, and examples." "OpenCV must provide a user-friendly interface and intuitive APIs that make it easy for the user to interact with the library and implement their computer vision project." "OpenCV must have a large and active community of users and developers who provide support and guidance to new users." "This community must include forums, mailing lists, and social media groups where users can ask questions and share their experiences."

Figure 9.14: Utility Tree - 3.

Chapter 10

Risks

– Nikhil Kumar G, Raj Palival

10.1 Introduction

OpenCV, like any software, is not immune to risks and challenges. While it is a powerful and widely-used open-source computer vision library, it's important to recognize that there may be potential pitfalls that can impact its usage and effectiveness. These risks can manifest in various areas, such as functionality, performance, security, compatibility, and documentation. Understanding these risks allows developers to be proactive in addressing them and ensuring the smooth integration and deployment of OpenCV in their applications. By staying informed, keeping up with updates, seeking community support, and following best practices, developers can mitigate these risks and harness the full potential of OpenCV for their computer vision needs.

10.2 Risks

Based on observations and investigations, there are a few anticipated risks and areas that can be improved in OpenCV:

10.2.1 Security Risks

1. OpenCV may be vulnerable to security attacks such as buffer overflows or injection attacks, which can lead to unauthorized access or control of systems.

2. Malicious users could exploit vulnerabilities in the library, potentially causing damage or compromising sensitive data.

Tactics for Security Risks

The tactics that can be applied to above risks are:

- Regularly update OpenCV to the latest version to incorporate security patches and fixes.

- Implement secure coding practices when using OpenCV, such as input validation and proper memory management.
- Perform security audits and vulnerability assessments to identify and mitigate potential risks.
- Encourage the OpenCV community to report and address security issues promptly.

10.2.2 Performance Risks

1. OpenCV may face performance limitations, especially when processing large-scale or real-time video streams or when working with resource-intensive algorithms.
2. Inefficient use of system resources or suboptimal algorithms could result in slow execution or inadequate responsiveness.

Tactics for Performance Risks

The tactics that can be applied to above risks are:

- Optimize algorithms and data structures to enhance computational efficiency.
- Leverage hardware acceleration technologies like GPU (Graphics Processing Unit) or dedicated AI accelerators when applicable.
- Employ parallel processing techniques, such as multi-threading or distributed computing, to distribute computational load and improve performance.
- Continuously benchmark and profile OpenCV applications to identify bottlenecks and areas for optimization.

10.2.3 Usability Risks

1. OpenCV has a steep learning curve for newcomers and may be challenging for developers without a strong background in computer vision.
2. Documentation may be insufficient or not beginner-friendly, making it difficult to understand and utilize the library effectively.

Tactics for Usability Risks

The tactics that can be applied to above risks are:

- Enhance the official OpenCV documentation with comprehensive and well-structured tutorials, examples, and explanations.
- Foster an active community that provides support, answers questions, and shares best practices.

- Develop user-friendly APIs and higher-level abstractions to simplify common computer vision tasks.
- Provide educational resources, such as online courses or workshops, to help developers acquire the necessary skills to use OpenCV effectively.

10.2.4 Compatibility and Portability Risks

1. OpenCV may face compatibility issues with different operating systems, hardware architectures, or versions of dependencies.
2. Lack of portability can limit the usage of OpenCV in diverse environments or on specific platforms.

Tactics for Compatibility and Portability Risks

The tactics that can be applied to above risks are:

- Conduct extensive testing on different platforms and environments to identify and resolve compatibility issues.
- Follow best practices for cross-platform development and ensure proper handling of platform-specific features or dependencies.
- Provide pre-compiled binaries or packages for popular operating systems to simplify installation and deployment.
- Encourage community contributions and feedback to address compatibility issues specific to certain environments.

10.2.5 Documentation Risks

1. OpenCV's documentation may lack clarity, completeness, or up-to-date information, leading to confusion and difficulties for developers.
2. Inadequate learning resources can make it challenging for beginners to grasp the concepts and effectively utilize the library.

Tactics for Documentation Risks

The tactics that can be applied to above risks are:

- Establish clear and well-structured documentation that covers all aspects of OpenCV, including detailed explanations, API references, and usage examples.
- Maintain an active and accessible community forum or support channel to address questions and provide guidance.

- Collaborate with educational institutions or organizations to develop comprehensive learning resources, tutorials, and courses for OpenCV.
- Encourage community contributions to enhance documentation and create additional learning materials.

10.2.6 Accuracy and Robustness Risks

1. OpenCV's computer vision algorithms may exhibit inaccuracies or lack robustness in certain scenarios or edge cases.
2. Performance degradation or incorrect results can occur when the algorithms fail to handle specific image variations or conditions.

Tactics for Accuracy and Robustness Risks

The tactics that can be applied to above risks are:

- Continuously evaluate and benchmark the performance and accuracy of OpenCV algorithms across a wide range of test cases and datasets.
- Collect and analyze user feedback and reported issues to identify areas where algorithms need improvement.
- Encourage collaboration and contributions from the computer vision research community to refine and enhance existing algorithms.
- Maintain a well-organized repository of sample images and datasets to facilitate testing and evaluation of algorithmic performance.

10.2.7 Scalability and Concurrency Risks

1. OpenCV may encounter challenges in scaling and efficiently utilizing multiple processing cores or distributed computing resources.
2. In scenarios where high concurrency is required, such as real-time video processing or large-scale computer vision pipelines, performance bottlenecks or resource contention can arise.

Tactics for Scalability and Concurrency Risks

The tactics that can be applied to above risks are:

- Implement parallel processing techniques, such as multi-threading or task-based concurrency, to distribute workloads and maximize resource utilization.
- Utilize frameworks or libraries that offer higher-level concurrency abstractions, such as OpenMP or Intel TBB, to simplify parallel programming in OpenCV.

- Optimize critical sections of code and minimize thread synchronization overhead to improve scalability and reduce contention.
- Explore distributed computing frameworks, such as Apache Spark or Hadoop, for processing large-scale computer vision tasks across multiple machines.

10.2.8 Data Privacy Risks

1. OpenCV applications may process and analyze sensitive data, such as images containing personally identifiable information or proprietary content.
2. Inadequate data privacy measures can lead to privacy breaches or unauthorized access to confidential information.

Tactics for Data Privacy Risks

The tactics that can be applied to above risks are:

- Adhere to relevant data privacy regulations and standards, such as GDPR or HIPAA, when handling sensitive data with OpenCV.
- Implement data anonymization techniques to remove or obfuscate personally identifiable information from images or other data inputs.
- Employ encryption mechanisms to protect data at rest or in transit during OpenCV processing.
- Educate developers on best practices for data privacy and encourage the use of secure coding principles when working with OpenCV.

10.2.9 Community Engagement Risks

1. OpenCV may face challenges in actively engaging the community and effectively incorporating user feedback and contributions.
2. Limited community involvement can result in slower resolution of issues, missed opportunities for improvement, and reduced adoption.

Tactics for Community Engagement Risks

The tactics that can be applied to above risks are:

- Foster an inclusive and collaborative community environment where users feel encouraged to provide feedback, report issues, and contribute to the development of OpenCV.
- Establish transparent communication channels, such as mailing lists, forums, or dedicated feedback platforms, to gather and address user input effectively.

- Regularly acknowledge and appreciate community contributions through recognition, documentation credits, or code attribution.
- Actively participate in computer vision conferences, workshops, and hackathons to engage with the broader community and promote OpenCV's advancements.

10.2.10 Integration Risks

1. OpenCV's integration with machine learning frameworks and libraries may face challenges in terms of compatibility, ease of use, or performance.
2. Inadequate integration can hinder the adoption of state-of-the-art machine learning techniques within OpenCV applications.

Tactics for Integration Risks

The tactics that can be applied to above risks are:

- Establish seamless integration with popular machine learning frameworks, such as TensorFlow or PyTorch, by providing dedicated APIs or interoperability layers.
- Offer pre-trained models or model conversion tools that enable users to easily leverage deep learning models within OpenCV.
- Collaborate with the machine learning community to identify and address integration issues, ensure compatibility, and keep up with the latest advancements in the field.
- Provide extensive documentation and examples showcasing the integration of OpenCV with machine learning, covering common use cases and best practices.

Chapter 11

Architecture

– Nikhil Kumar G, Raj Palival

11.1 Description

OpenCV (Open Source Computer Vision Library [4]) is an open-source library that includes several hundreds of computer vision algorithms. It has a modular structure architecture, which means that the package includes several modules with shared or static libraries.

11.2 Modules

There are two types of modules available:

1. Main Modules: These modules are the core components of the OpenCV library that provide essential computer vision functionalities.

2. Extra modules: These are additional modules that offer extended features and algorithms not included in the main modules.

Below is a list of all the modules available in each category:

11.2.1 Main Modules

1. core. Core functionality: a compact module defining basic data structures, including the dense multi-dimensional array Mat and basic functions used by all other modules.

2. imgproc. Image Processing: an image processing module that includes linear and non-linear image filtering, geometrical image transformations (resize, affine and perspective warping, generic table-based remapping), color space conversion, histograms, and so on.

3. imgcodecs. Image file reading and writing: This module in OpenCV provides functions for reading and writing image files in various formats

4. videoio. Video I/O: This module in OpenCV provides functions for reading and writing video files and accessing video streams from cameras or other devices.

5. highgui. High-level GUI: This module in OpenCV provides high-level functions for creating graphical user interfaces (GUI) to display and interact with images, videos, and other visual data.

6. video. Video Analysis: This module in OpenCV involves performing various operations on video frames, such as object detection, tracking, motion analysis, and frame manipulation.

7. calib3d. Camera Calibration and 3D Reconstruction: This module in OpenCV provides functions for camera calibration and 3D reconstruction.

8. features2d. 2D Features Framework: This module in OpenCV provides a framework for detecting and describing 2D image features. These features can be used for tasks such as image matching, object recognition, and image stitching.

9. objdetect. Object Detection: This module in OpenCV provides functions and pre-trained models for object detection. It allows you to detect and localize objects within images or video frames.

10. dnn. Deep Neural Network module: This module in OpenCV provides a flexible and efficient framework for working with deep neural networks (DNNs). It allows you to load pre-trained models, perform inference on images or videos, and fine-tune models.

11. ml. Machine Learning: This module in OpenCV provides a set of machine learning algorithms and tools for various tasks, including classification, regression, clustering, and dimensionality reduction.

12. flann. Clustering and Search in Multi-Dimensional Spaces: This module in OpenCV provides efficient algorithms for clustering and searching in multi-dimensional spaces. FLANN stands for Fast Library for Approximate Nearest Neighbors.

13. photo. Computational Photography: This module refers to the use of computational techniques and algorithms to enhance or modify digital photographs.

14. stitching. Images stitching: Image stitching is the process of combining multiple overlapping images to create a larger, seamless panorama.

15. gapi. Graph API: This module is a computational graph-based framework provided by OpenCV that allows for efficient parallel execution of image processing and computer vision algorithms.

11.2.2 Extra Modules

1. alphamat. Alpha Matting

2. aruco. Aruco markers, module functionality was moved to objdetect module

3. barcode. Barcode detecting and decoding methods

4. bgsegm. Improved Background-Foreground Segmentation Methods
5. bioinspired. Biologically inspired vision models and derivated tools
6. ccalib. Custom Calibration Pattern for 3D reconstruction
7. cudaarithm. Operations on Matrices
8. cudabgsegm. Background Segmentation
9. cudacodec. Video Encoding/Decoding
10. cudafeatures2d. Feature Detection and Description
11. cudafilters. Image Filtering
12. cudaimgproc. Image Processing
13. cudalegacy. Legacy support
14. cudaobjdetect. Object Detection
15. cudaoptflow. Optical Flow
16. cudastereo. Stereo Correspondence
17. cudawarping. Image Warping
18. cudev. Device layer
19. cvv. GUI for Interactive Visual Debugging of Computer Vision Programs
20. datasets. Framework for working with different datasets
21. dnn objdetect. DNN used for object detection
22. dnn superres. DNN used for super resolution
23. dpm. Deformable Part-based Models
24. face. Face Analysis
25. freetype. Drawing UTF-8 strings with freetype/harfbuzz
26. fuzzy. Image processing based on fuzzy mathematics
27. hdf. Hierarchical Data Format I/O routines
28. hfs. Hierarchical Feature Selection for Efficient Image Segmentation
29. img hash. The module brings implementations of different image hashing algorithms.

30. intensity transform. The module brings implementations of intensity transformation algorithms to adjust image contrast.
31. julia. Julia bindings for OpenCV
32. line descriptor. Binary descriptors for lines extracted from an image
33. mcc. Macbeth Chart module
34. optflow. Optical Flow Algorithms
35. ovis. OGRE 3D Visualiser
36. phase unwrapping. Phase Unwrapping API
37. plot. Plot function for Mat data
38. quality. Image Quality Analysis (IQA) API
39. rapid. silhouette based 3D object tracking
40. reg. Image Registration
41. rgbd. RGB-Depth Processing
42. saliency. Saliency API
43. sfm. Structure From Motion
44. shape. Shape Distance and Matching
45. stereo. Stereo Correspondance Algorithms
46. structured light. Structured Light API
47. superres. Super Resolution
48. surface matching. Surface Matching
49. text. Scene Text Detection and Recognition
50. tracking. Tracking API
51. videostab. Video Stabilization
52. viz. 3D Visualizer
53. wechat qrcode. WeChat QR code detector for detecting and parsing QR code.
54. xfeatures2d. Extra 2D Features Framework
55. ximgproc. Extended Image Processing
56. xobjdetect. Extended object detection
57. xphoto. Additional photo processing algorithms

11.3 Grouping Modules

The modules don't have a clear order or hierarchy. Core, python, and java are commonly used modules, but there isn't a definite separation between them. Often, a module that uses the core module also depends on another module, which makes the structure very complex.

However, we can group all the modules in three categories:

1. Main Functionality
2. Extended Functionality
3. High Level GUI

11.4 15 Most Important Modules

Below is a figure of the 15 most important modules in the three categories discussed above:

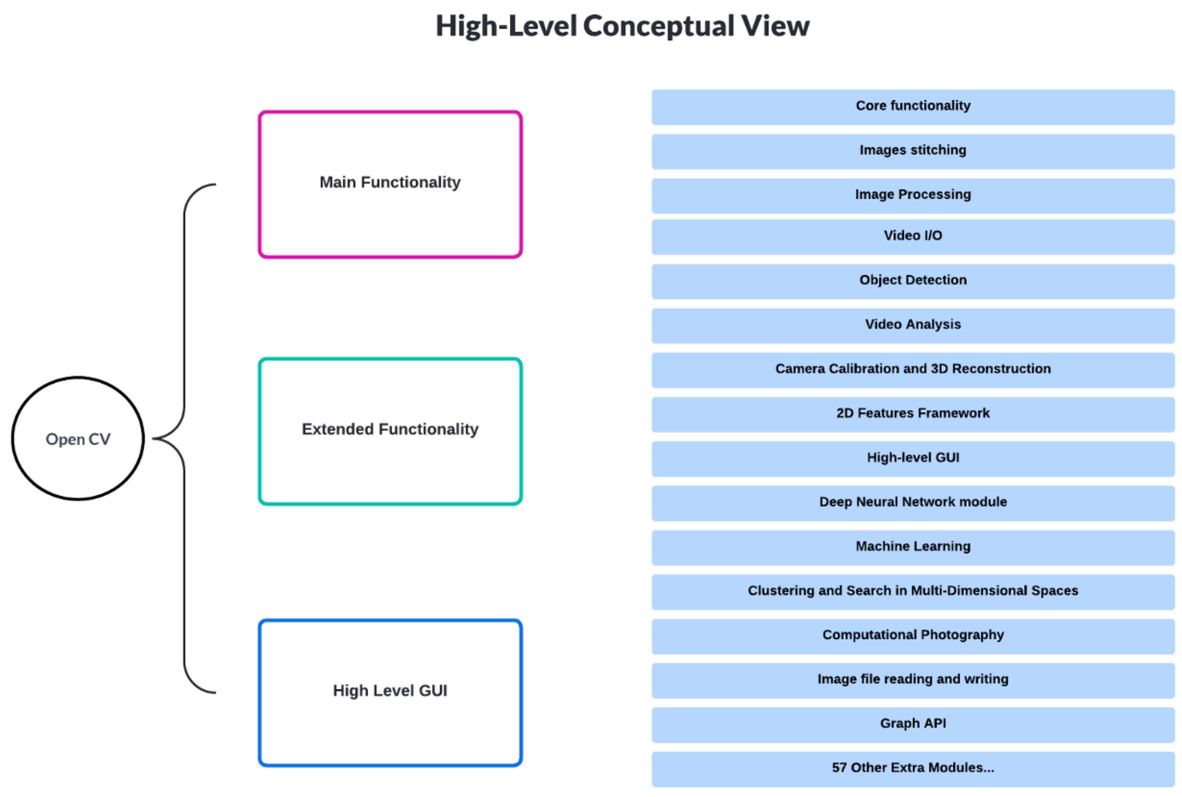

High-Level Conceptual View
Open CV
Main Functionality
Extended Functionality
High Level GUI
Core functionality
Images stitching
Image Processing
Video I/O
Object Detection
Video Analysis
Camera Calibration and 3D Reconstruction
2D Features Framework
High-level GUI
Deep Neural Network module
Machine Learning
Clustering and Search in Multi-Dimensional Spaces
Computational Photography
Image file reading and writing
Graph API
57 Other Extra Modules...

Figure 11.1: High Level Conceptual View

11.5 4+1 View of Architecture

The 4+1 architectural view model provides multiple perspectives on the architecture of a system [13]. Such as:

1. Implementation View
2. Use Case View
3. Logical View
4. Process View
5. Deployment View

We will use an example of one module to explain the rest of the OpenCV modules in OpenCV's architecture. More on this later.

11.5.1 Implementation View

The implementation view is an architectural view that provides a detailed perspective on how the system is implemented and how the implementation artifacts are organized and interact with each other.

In our case we will consider the following:

- Perspective: Developers, Proj. mngs.
- Stage: Design
- Focus: Subsystem decomposition
- Concerns: Software management
- Artefacts: Modules structure, Package diagram

Implementation View Artefacts

The modules structure below shows how all the modules in OpenCV interact with each other. It is clear that the modules in OpenCV are interconnected in a very complex way.

Modules Structure: Diagram taken from source [14]

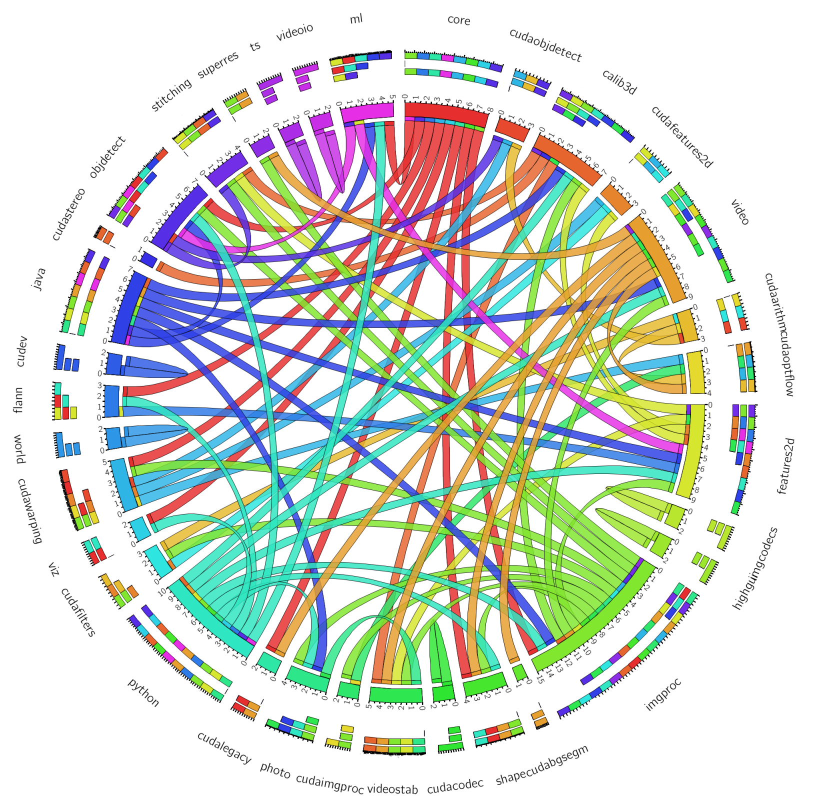


Figure 11.2: Modules Interconnection Structure

Because of its complex nature, we will zoom into the architecture a little bit and understand one such important module of OpenCV among the 15 most important modules through all the 4+1 views' from here on:

- Image Stitching Module (stitching.): Famously used to create panorama images [15].

The idea is to understand this one module which in-turn sort of explains the rest of the OpenCV modules in OpenCV's architecture.

Below is a Package Diagram of Image Stitching Module:

The diagram includes 3 levels:

1. 1st Level: It is the OpenCV package that contains all modules.
2. 2nd Level: It shows how the Image Stitching module is Interacting with other modules in OpenCV package.
3. 3rd Level: Shows the functions present in Image Stitching module and the other two modules.

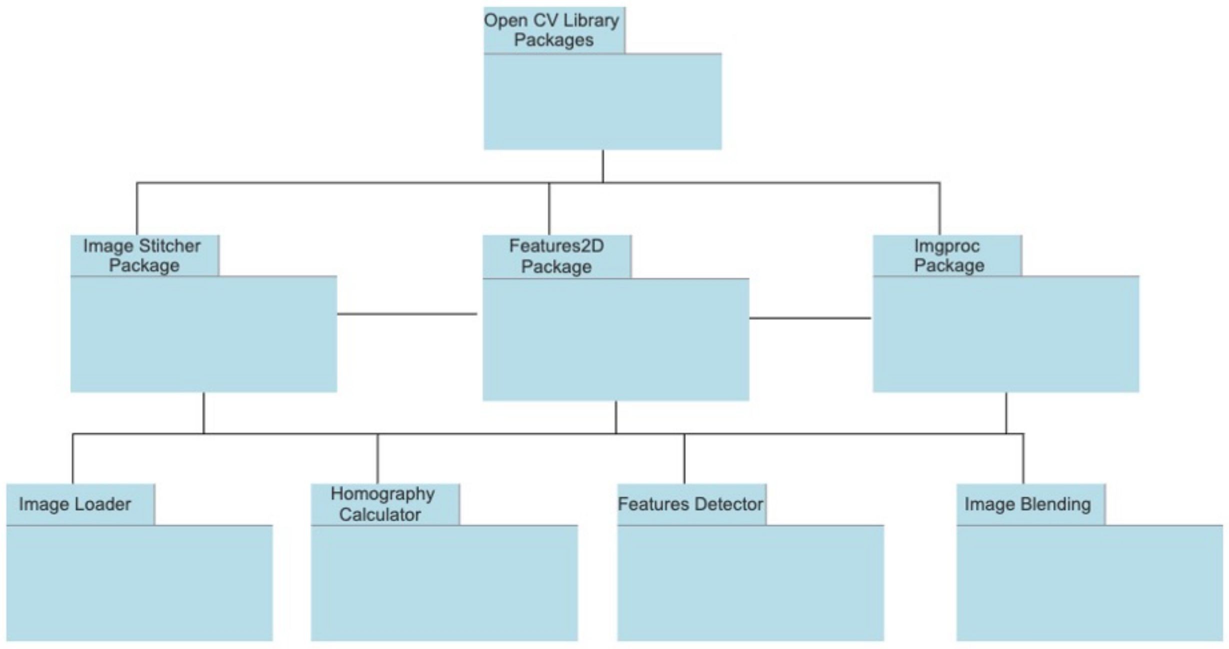


Figure 11.3: Image Stitcher Package Diagram

Functions

1. Image Loader: It is used to load and prepare the images in Image Stitching package for stitching process.

2. Homography Calculator: It is used to calculate a homography matrix between two sets of corresponding points in an image.

3. Features Detector: It is used to identify and extract distinctive features or points of interest from an image.

4. Image Blending: It is the process of combining two or more images to create a single composite image that preserves the important information from each input image.

11.5.2 Use Case View

The use case view is an architectural view that focuses on describing the system's functionality from the perspective of its users or external actors.

In our case we will consider the following:

- Perspective: End users
- Stage: Putting it alltogether
- Concerns: Understandability, usability
- Focus: Feature decomposition
- Artefact: Use-case diagram

Use Case Diagram

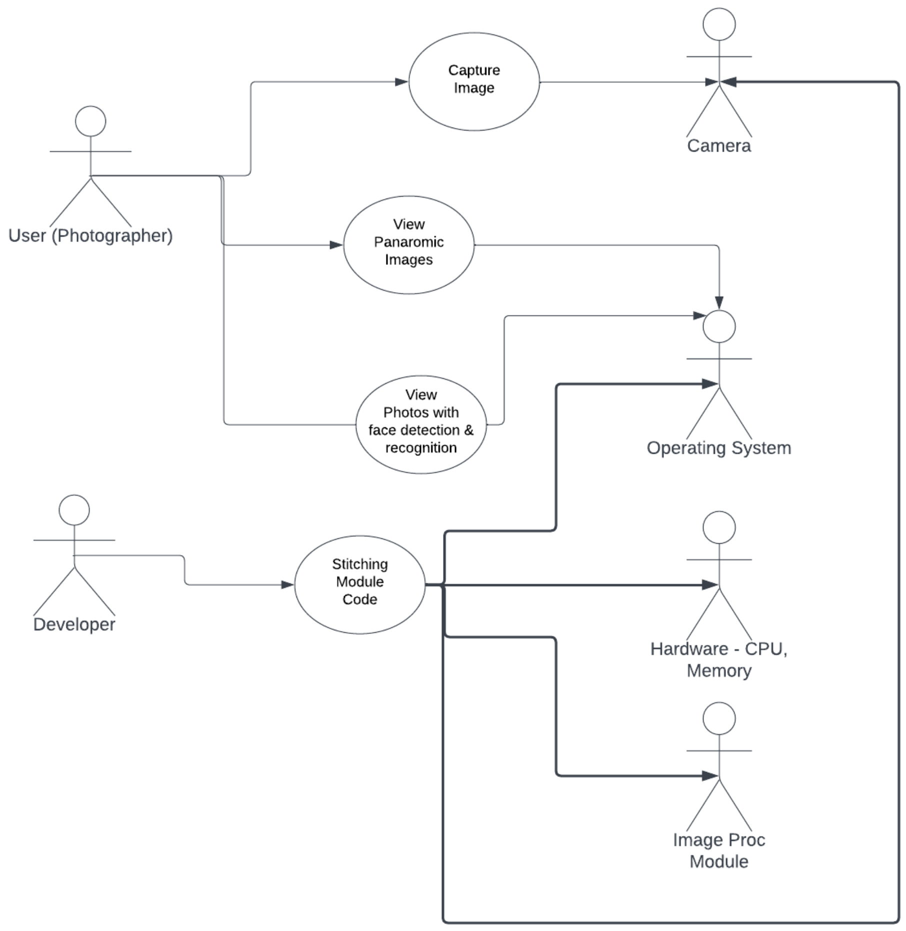


Figure 11.4: Use Case Diagram for Image Stiching Module [2]

We have the following in our diagram:

1. Actors: Two actors
   (a) User: Photographer has 3 use cases:-
      i. To capture images
      ii. View panaromic images
      iii. View photos with face detection and recognition.
   (b) Developer has 1 use case:-
      i. Develop stitching module code
2. Use Cases: Four use cases
   (a) To capture images
   (b) View panaromic images
   (c) View photos with face detection and recognition.
   (d) Develop stitching module code
3. External Actors: Four external actors
   (a) Camera: Facilitates in capturing images
   (b) Operating System: Used by both User and developer for their use cases
   (c) Hardware Components: CPU, Memory
   (d) Image Proc Module: Provides a wide range of functions and algorithms for manipulating and enhancing images.

11.5.3 Logical View

The logical view is an architectural view that focuses on representing the high-level structure and organization of the system's functionality and behavior. It provides an abstraction of the system's functionality and its relationships, without delving into implementation details.

In our case we will consider the following:

- Perspective: Analysts, Designers Stage: Requirement analysis
- Focus: Object oriented decomposition
- Concerns: Functionality
- Artefacts: Class diagram [16]

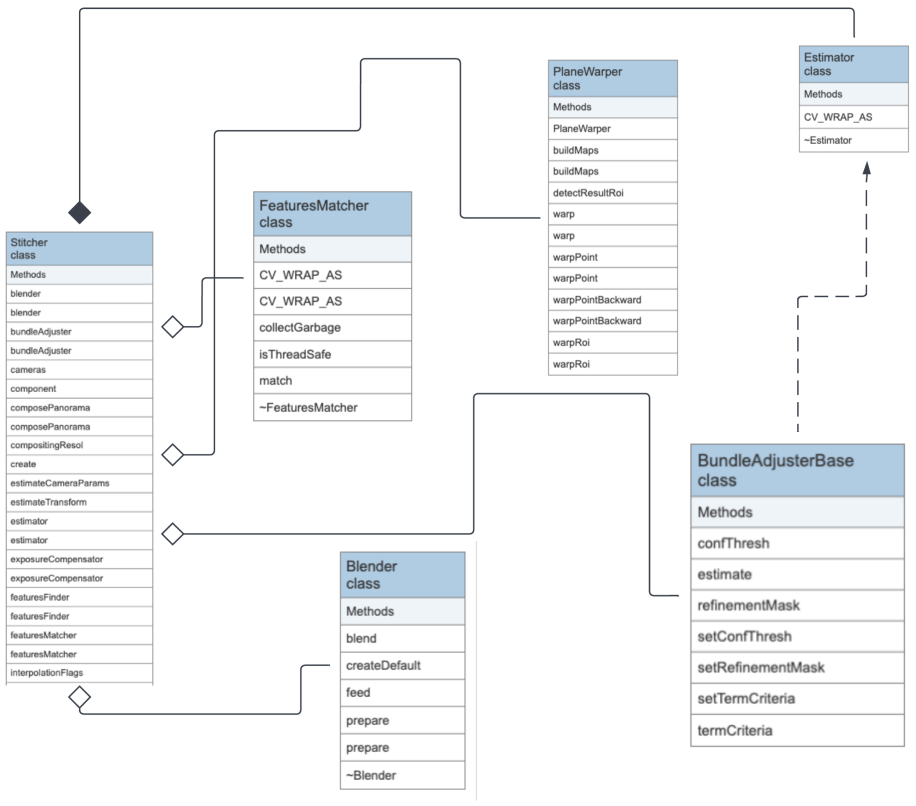


Figure 11.5: Class Diagram of Image Stitching Module [3]

The logical view of OpenCV's Image Stitching module describes the system's functionality and the relationships between its components. The module consists of several classes, including the Stitcher class, Features Matcher class, Plane Warper class, Blender class, BundleAdjusterBase-Class, and Estimator class. Here is a brief description of each class and how they depend on each other:

1. Stitcher class: The Stitcher class is the main class in the Image Stitching module. It provides the functionality to stitch multiple images together to create a panoramic image. The Stitcher class uses other classes in the module to perform various tasks, such as feature detection, feature matching, and image blending.

2. Features Matcher class: The Features Matcher class is responsible for detecting and matching features in the input images. It uses various algorithms to detect and match features, such as the SIFT algorithm and the SURF algorithm.

3. Plane Warper class: The Plane Warper class is responsible for warping the input images to a common plane. It uses various algorithms to perform the warping, such as the Homography algorithm and the Affine algorithm.

4. Blender class: The Blender class is responsible for blending the warped images together to create a seamless panoramic image. It uses various algorithms to perform the blending, such as the Multi-band blending algorithm and the Feather blending algorithm.

5. BundleAdjusterBaseClass: The BundleAdjusterBaseClass is responsible for adjusting the parameters of the stitching process to improve the quality of the output image. It uses various algorithms to adjust the parameters, such as the Levenberg-Marquardt algorithm and the Gauss-Newton algorithm.

6. Estimator class: The Estimator class is responsible for estimating the camera parameters of the input images. It uses various algorithms to estimate the parameters, such as the RANSAC algorithm and the Least Squares algorithm.

These classes depend on each other in a hierarchical manner. The Stitcher class is the main class that uses the other classes to perform various tasks. The Features Matcher class and the Estimator class are used by the Stitcher class to detect and match features and estimate camera parameters, respectively. The Plane Warper class is used by the Stitcher class to warp the input images to a common plane. The Blender class is used by the Stitcher class to blend the warped images together to create a seamless panoramic image. Finally, the BundleAdjusterBaseClass is used by the Stitcher class to adjust the parameters of the stitching process to improve the quality of the output image. Overall, these classes work together to provide the functionality of the Image Stitching module in OpenCV.

11.5.4 Process View

The process view is an architectural view that focuses on illustrating the runtime behavior and concurrency aspects of a system. It provides a perspective on how the system's functionality is executed and how different components or processes interact and collaborate during runtime.

In our case we will consider the following:

- Perspective: System Integrators
- Stage: Design
- Focus: Process decomposition
- Concerns: Performance, scalability, throughput
- Artefacts: Sequence diagram

Sequence Diagram

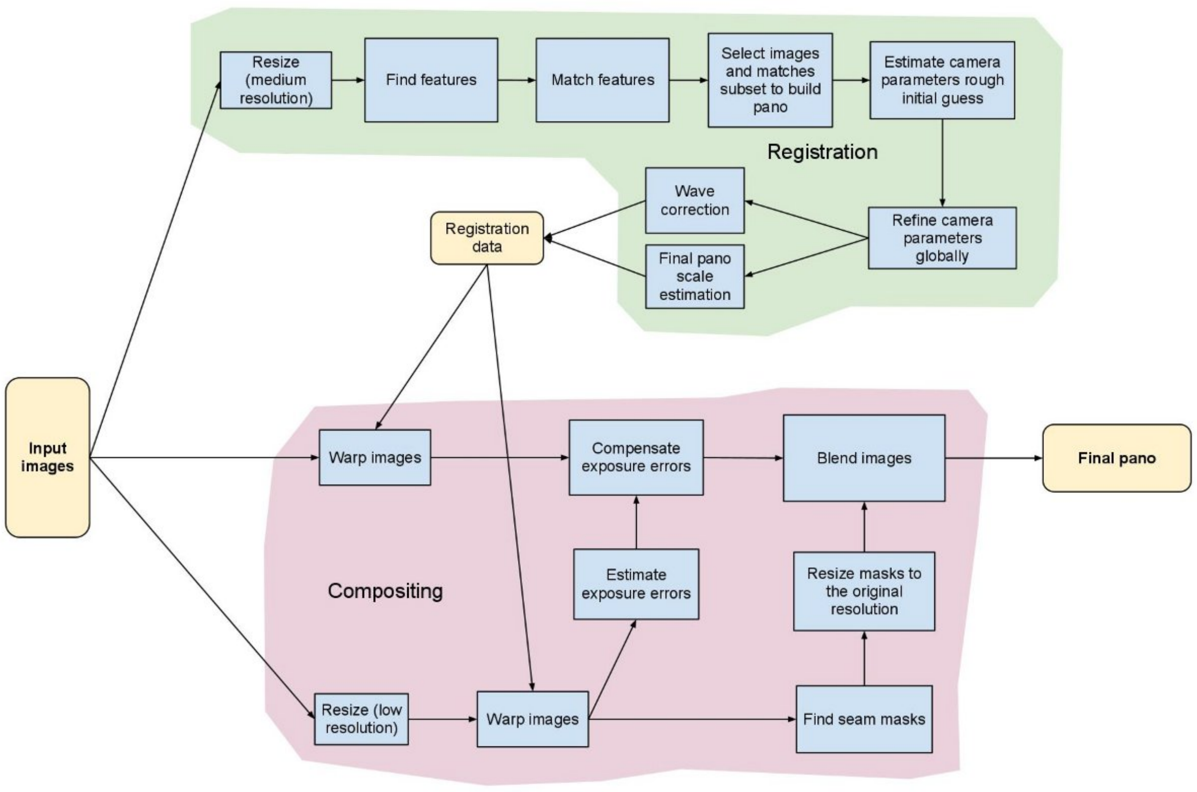

Resize (medium resolution)
Find features
Match features
Select images and matches subset to build pano
Estimate camera parameters rough initial guess
Registration
Wave correction
Registration data
Refine camera parameters globally
Final pano scale estimation
Input images
Warp images
Compensate exposure errors
Blend images
Final pano
Compositing
Estimate exposure errors
Resize masks to the original resolution
Resize (low resolution)
Warp images
Find seam masks

Figure 11.6: Sequence Diagram of Image Stitching Module

The process view of OpenCV's Image Stitching module describes the system's behavior and how it responds to external events. The module performs several operations to stitch multiple images together to create a panoramic image. Here is a brief description of each operation and the steps involved:

1. Step 1 - Input Images:
   The first operation is to input the images that need to be stitched together. These images are typically captured using a camera or downloaded from a source.

2. Step 2 - Registration:
   The second operation is registration, which involves several steps:

   - Resize (medium resolution): The input images are resized to a medium resolution to reduce the computational complexity of the feature detection and matching algorithms.
   - Find features: The feature detection algorithm is used to detect features in the input images, such as corners, edges, and blobs.
   - Match features: The feature matching algorithm is used to match the features between the input images.
   - Select images and matches subset to build panorama: The input images and their corresponding feature matches are selected to build the panorama.
   - Estimate camera parameters rough initial guess: The camera parameters of the input images are estimated using a rough initial guess.
   - Refine camera parameters globally: The camera parameters are refined globally using a bundle adjustment algorithm.
   - Wave correction: The wave correction algorithm is used to correct the distortion caused by the camera lens.
   - Final panorama scale estimation: The final scale of the panorama is estimated based on the camera parameters and the input images.

3. Step 3 - Registration data :
   The registration data, including the camera parameters and the warped images, are passed to the compositing operation.

4. Step 4 - Compositing:
   The final operation is compositing, which involves several steps:

   - Warp images: The warped images are transformed to a common coordinate system using the estimated camera parameters.
   - Estimate exposure errors: The exposure errors in the input images are estimated using an exposure estimation algorithm.
   - Compensate exposure errors: The exposure errors in the input images are compensated using an exposure compensation algorithm.

- Resize (low resolution): The final panoramic image is resized to a low resolution for display or storage purposes.
- Warp images: The warped images are transformed again to the original resolution.
- Find seam masks: The seam masks are found using a seam finding algorithm.
- Resize masks to original resolution: The seam masks are resized to the original resolution.
- Blend images: The blended image is created by blending the warped images together using a blending algorithm.

5. Step 5 - Final panorama:
   The final panoramic image is created by combining the blended image and the seam masks.

Overall, the process view of OpenCV's Image Stitching module describes the system's behavior and how it responds to external events, such as input images and user requests. The module performs several operations, including registration and compositing, to stitch multiple images together to create a beautiful panoramic image.

11.5.5 Deployment View

The deployment view is an architectural view that focuses on illustrating how the software components of a system are deployed and distributed across hardware resources. It provides a representation of the physical infrastructure and deployment topology of the system.

In our case we will consider the following:

- Perspective: System Engineers
- Stage: Design
- Focus: Map software to hardware
- Concerns: System topology, delivery, installation, communication
- Artefacts: Deployment diagram

Deployment Diagram

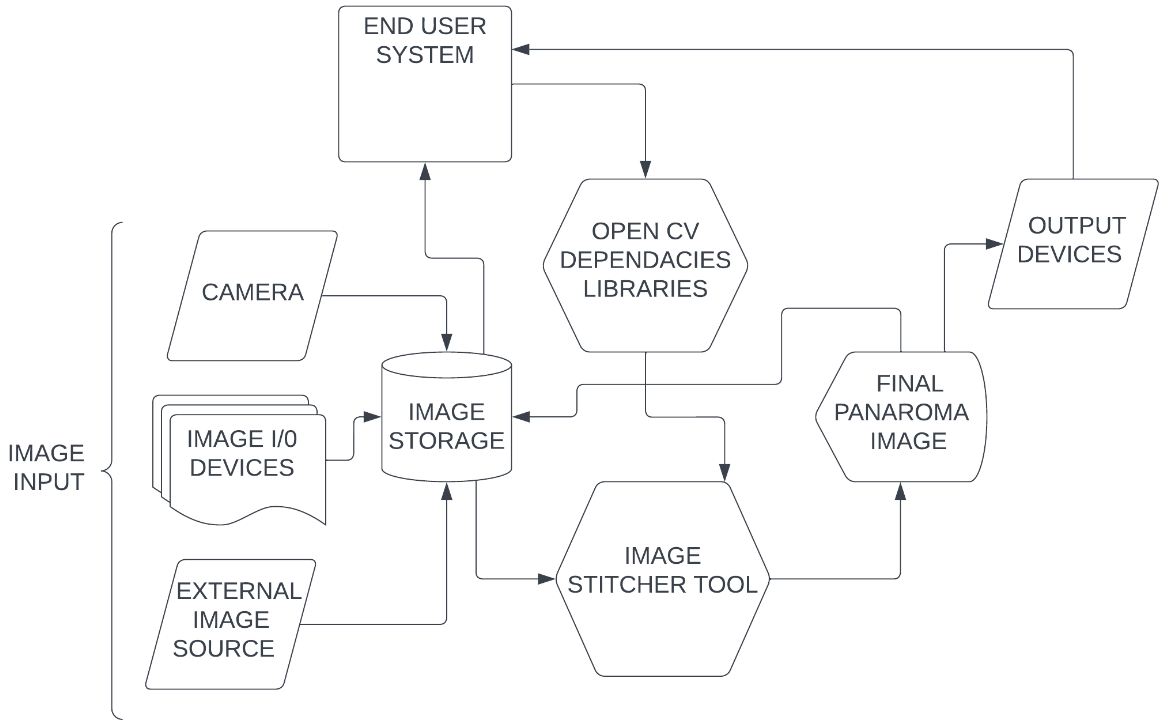


Figure 11.7: Deployment Diagram of Image Stitching Module

The deployment view of OpenCV's image stitching module involves several components that work together to produce a final panorama output image:

1. Image input: The images that need to be stitched together.
    - Camera: This is the device that captures the images that will be stitched together.
    - Image input/output devices: These are devices that can be used to input or output images, such as USB drives or SD cards.
    - External image source: This component refers to any external source of images that can be used as input for the stitching process.
2. Image storage database: This component is responsible for storing the images that will be used for stitching. It can be a local or remote database.
3. OpenCV libraries / dependencies: This component includes all the necessary libraries and dependencies required for the image stitching module to function properly.
4. Image Stitching module of OpenCV: This is the core component of the deployment view. It is responsible for stitching the input images together to produce a final panorama output image.
5. Final panorama output image: This is the output of the image stitching module. It is the final image that is produced after the input images have been stitched together.
6. Output devices: These are devices that can be used to output the final panorama image, such as a monitor or a printer.
7. End-user system: This component refers to the system that the end-user will be using to access the image stitching module. It can be a desktop computer, a laptop, or a mobile device.

Overall, the deployment view of OpenCV's image stitching module involves several components that work together to produce a final panorama output image. The input images can come from various sources, and the final output can be displayed on different output devices. The image stitching module is the core component that stitches the input images together, and the end-user system is the system that the end-user will be using to access the module.

OpenCV Installation Dependencies

To install OpenCV on a system, there are several dependencies that need to be installed first. The specific dependencies required may vary depending on the operating system and the version of OpenCV being installed. Here is a general list of dependencies that are commonly required:

1. CMake: This is a cross-platform build system that is used to build OpenCV from source.
2. Git: This is a version control system that is used to download the OpenCV source code.
3. GCC: This is a compiler that is used to compile the OpenCV source code.
4. Python: OpenCV has a Python interface, so Python needs to be installed on the system.

5. NumPy: This is a Python library that is used for numerical computing. It is required for the Python interface of OpenCV.
6. libjpeg: This is a library for reading and writing JPEG files. It is required for the imgcodecs module of OpenCV.
7. libpng: This is a library for reading and writing PNG files. It is required for the imgcodecs module of OpenCV.
8. libtiff: This is a library for reading and writing TIFF files. It is required for the imgcodecs module of OpenCV.
9. libjasper: This is a library for reading and writing JPEG-2000 files. It is required for the imgcodecs module of OpenCV.
10. libwebp: This is a library for reading and writing WebP files. It is required for the imgcodecs module of OpenCV.
11. OpenEXR: This is a library for reading and writing high dynamic range (HDR) images. It is required for the imgcodecs module of OpenCV.
12. TBB: This is the Intel Threading Building Blocks library, which is used for parallel processing. It is required for the performance optimization of OpenCV.
13. Eigen: This is a C++ template library for linear algebra. It is used for the computation of homography matrices in the image stitching process.
14. CUDA: This is a parallel computing platform and programming model developed by NVIDIA. It is used for GPU acceleration in the image stitching process.
15. OpenCL: This is an open standard for parallel programming of heterogeneous systems. It is used for GPU acceleration in the image stitching process.

11.6 Summary of 4+1 View:

Each view provides a distinct perspective on the OpenCV architecture, allowing stakeholders to understand and analyze the system from different angles. This 4+1 view model helps in capturing the functional, development, process, physical, and use case aspects of the OpenCV backend architecture.

Here is a diagram to showcase a brief overview of what we covered:

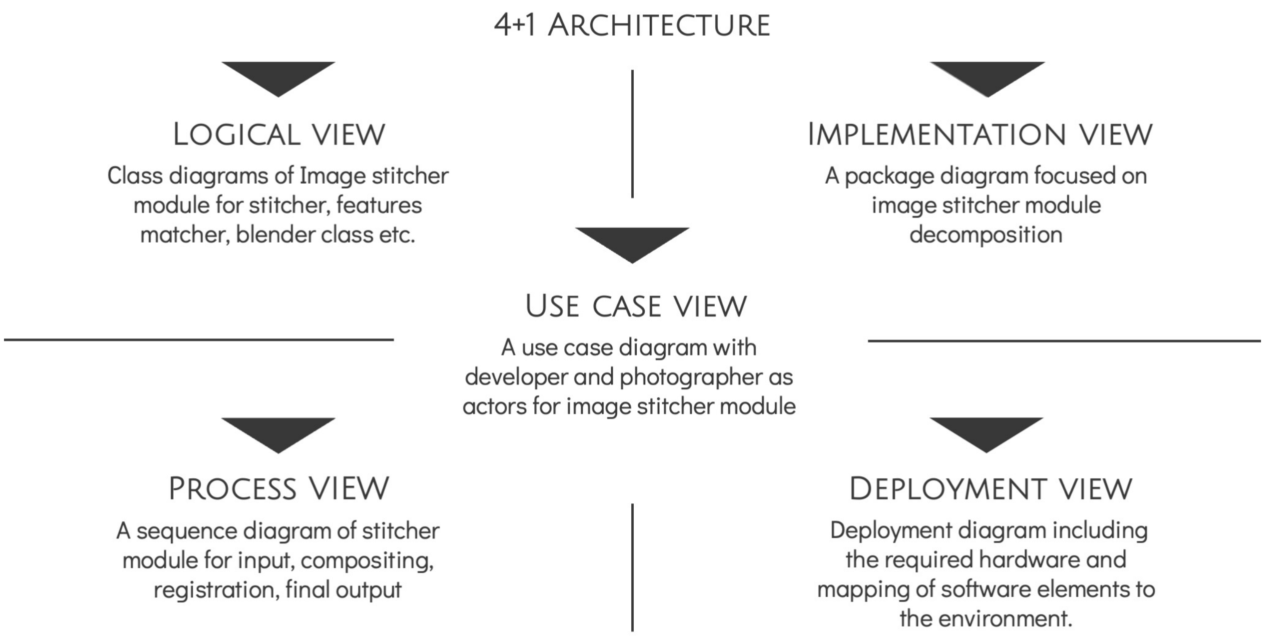

4+1 ARCHITECTURE
LOGICAL VIEW
Class diagrams of Image stitcher module for stitcher, features matcher, blender class etc.
IMPLEMENTATION VIEW
A package diagram focused on image stitcher module decomposition
USE CASE VIEW
A use case diagram with developer and photographer as actors for image stitcher module
PROCESS VIEW
A sequence diagram of stitcher module for input, compositing, registration, final output
DEPLOYMENT VIEW
Deployment diagram including the required hardware and mapping of software elements to the environment.

Figure 11.8: 4+1 Overview

The purpose of using the 4+1 view model for OpenCV's Image Stitching module is to provide a comprehensive understanding of the system's architecture from multiple perspectives. This approach helps to ensure that the system is designed and implemented in a way that meets the needs and requirements of its stakeholders, while also addressing potential risks and quality issues.

The Logical View provides a high-level overview of the system's functionality and the relationships between its components. This view focuses on the system's data and functionality, and how they are organized and structured. It helps stakeholders to understand the system's overall purpose and how it achieves its goals.

The Process View describes the system's behavior and how it responds to external events. This view focuses on the system's processes and how they interact with each other and with external systems. It helps stakeholders to understand how the system works and how it responds to different inputs and events.

The Implementation View describes the system's development environment and how it is organized. This view focuses on the system's development tools, processes, and methodologies. It helps stakeholders to understand how the system is developed and maintained, and how changes are made to the system over time.

The Deployment View describes the system's physical architecture and how it is deployed. This view focuses on the system's hardware and software components, and how they are connected and configured. It helps stakeholders to understand how the system is deployed and how it interacts with other systems and devices.

The Use Case View describes the system's functionality from the perspective of its users. This view focuses on the system's use cases and how they are implemented in the system. It helps stakeholders to understand how the system is used and how it meets the needs and requirements of its users.

Overall, the 4+1 view model provides a comprehensive understanding of the OpenCV Image Stitching module's architecture from multiple perspectives, helping stakeholders to understand the system's functionality, behavior, development environment, physical architecture, and user requirements. This understanding is essential for ensuring that the system is designed and implemented in a way that meets the needs and requirements of its stakeholders, while also addressing potential risks and quality issues.

Bibliography

[1] JGraph Ltd, “draw io,” 2005-2023. [Online]. Available: https://www.drawio.com

[2] Lucid Software Inc., “Lucid chart.” [Online]. Available: https://www.lucidchart.com/pages/

[3] SmartDraw, LLC, “Smartdraw tool,” 1994-2023. [Online]. Available: https://www.smartdraw.com

[4] (2000) Open cv. [Online]. Available: https://opencv.org/

[5] I. Culjak, D. Abram, T. Pribanic, H. Dzapo, and M. Cifrek, “A brief introduction to opencv,” pp. 1725–1730, 2012.

[6] (2020) Opencv project report by delft students. [Online]. Available: https://delftswa.gitbooks.io/desosa2016/content/opencv/Chapter.html

[7] (2018) Quality attributes requirements. [Online]. Available: https://medium.com/@nvashanin/quality-attributes-in-software-architecture-3844ea482732

[8] (2001) Modular architecture. [Online]. Available: https://www.ecs.csun.edu/~rlingard/COMP684/Example2SoftArch.htm

[9] Z. Zhang, “A flexible new technique for camera calibration,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 11, pp. 1330–1334, 2000.

[10] J. Sigut, M. Castro, R. Arnay, and M. Sigut, “Opencv basics: A mobile application to support the teaching of computer vision concepts,” *IEEE Transactions on Education*, vol. 63, no. 4, pp. 328–335, 2020.

[11] R. Botan, K. F. Coco, and K. S. Komati, “Implementation of an image stitching algorithm to a low-cost digital microscope,” pp. 285–291, 2017.

[12] (2023) Business requirements. [Online]. Available: https://www.educative.io/answers/what-are-business-requirements

[13] (2010) Software architecture - 4+1 view model. [Online]. Available: https://www.cs.unb.ca/~wdu/cs6075w10/sa2.htm

[14] (2020) Modular strucure. [Online]. Available: https://delftswa.gitbooks.io/desosa2016/content/opencv/Chapter.html

[15] K. Chen and M. Wang, “Image stitching algorithm research based on opencv,” pp. 7292–7297, 2014.

[16] (2023) Class diagram — Wikipedia, the free encyclopedia. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Class_diagram&oldid=1137357004

Index