

# **Indian Institute of Technology Gandhinagar**

## Computer Vision, Imaging, and Graphics (CVIG) Lab



**Project Course Report**  
Indian Institute of Technology Gandhinagar  
Palaj, Gandhinagar – 382355

---

# **Under Vehicle Surveillance System**

---

**Submitted by**  
**Bhavay Goyal** (24110070)  
**Chaitanya Bhoite** (24110086)  
**Divyansh Sharma** (24110113)  
**Yash Goyal** (24110399)

**Under the guidance of**  
**Prof. Shanmuganathan Raman**

Professor and Head, Department of Computer Science and Engineering  
Professor, Department of Electrical Engineering  
Indian Institute of Technology Gandhinagar

# Abstract

Vehicle security inspection is an essential component of safety at critical checkpoints such as airports, military bases, embassies, borders, malls, and metro stations. The underside of a vehicle is a natural hiding place for contraband, explosives, drugs, suspicious packages, and unauthorized modifications. Traditionally, guards rely on handheld mirrors, flashlights, and time-pressured human judgment. This is slow, error-prone, and potentially dangerous in high-risk environments.

This report presents a complete Under Vehicle Surveillance System (UVSS) pipeline that automates undercarriage imaging. Our system captures multiple images of the vehicle underside using both planar and fisheye cameras, undistorts them using Scaramuzza's omnidirectional camera model, preprocesses them using cropping and histogram equalization, and stitches them into a seamless panoramic mosaic using the highly optimized OpenPano C++ package. We also outline a future anomaly detection stage based on a convolutional autoencoder that will produce pixel-wise heatmaps and alerts for security personnel. The goal is to take a collection of distorted, raw undercarriage images and convert them into a single, geometrically accurate, analysis-ready mosaic that forms the backbone of next-generation automated security systems.

# Contents

<b>Abstract</b>	<b>1</b>	
<b>1</b>	<b>Introduction</b>	<b>4</b>
1.1	Background: Why Under-Vehicle Surveillance Matters . . . . .	4
1.2	Project Goal and System Functionality . . . . .	5
1.3	End-to-End System Pipeline Overview . . . . .	5
1.4	Narrative System Description . . . . .	5
<b>2</b>	<b>Imaging Setup and Distortion</b>	<b>7</b>
2.1	Camera Configurations: Planar vs. Fisheye . . . . .	7
2.1.1	Planar Camera (Normal Lens) . . . . .	7
2.1.2	Fisheye Camera (Omnidirectional Lens) . . . . .	7
2.2	Example Images from the System . . . . .	8
2.3	Understanding Fisheye Image Distortion . . . . .	10
<b>3</b>	<b>The Science of Undistortion</b>	<b>11</b>
3.1	Image Distortion: Intuitive View . . . . .	11
3.2	Scaramuzza OCamCalib Model . . . . .	11
3.2.1	Why OCamCalib? . . . . .	11
3.2.2	Calibration Procedure . . . . .	11
3.3	Python Implementation for Undistortion . . . . .	12
3.3.1	Pixel-wise Inverse Projection . . . . .	12
3.3.2	Computational Considerations . . . . .	12
<b>4</b>	<b>Preprocessing</b>	<b>13</b>
4.1	Cropping Strategy . . . . .	13
4.2	Histogram Equalization . . . . .	13
<b>5</b>	<b>The Stitching Pipeline</b>	<b>15</b>
5.1	Stitching Experiments and Limitations . . . . .	15
5.1.1	OpenCV Stitcher Module . . . . .	15
5.1.2	SIFT / ORB + Homography (Manual Pipeline) . . . . .	15
5.1.3	OmniCV-Lib . . . . .	16
5.2	The Chosen Solution: OpenPano . . . . .	16
5.2.1	Overview of OpenPano . . . . .	16

5.2.2	Why OpenPano Worked Better . . . . .	16
5.2.3	Multi-Band Blending . . . . .	16
5.3	Final Mosaic Output . . . . .	17
<b>6</b>	<b>Future Work: Anomaly Detection</b>	<b>18</b>
6.1	Dataset Creation . . . . .	18
6.2	Convolutional Autoencoder Model . . . . .	18
6.3	Anomaly Detection Workflow . . . . .	19
6.4	Advantages of the CAE Approach . . . . .	19
<b>7</b>	<b>Presentation Poster</b>	<b>20</b>
	<b>Presentation Poster</b>	<b>20</b>
<b>8</b>	<b>Conclusion and Key Technical Highlights</b>	<b>21</b>
8.1	Conclusion . . . . .	21
8.2	Key Points for Visitors and Graders . . . . .	21
	<b>References</b>	<b>23</b>

# Chapter 1

## Introduction

### 1.1 Background: Why Under-Vehicle Surveillance Matters

Security checkpoints at airports, military bases, embassies, borders, malls, and metro stations routinely inspect the underside of vehicles. The undercarriage is a prime location to conceal:

- contraband,
- explosives and improvised explosive devices (IEDs),
- narcotics and other illegal substances,
- suspicious packages or tampered components,
- unauthorized mechanical or electronic modifications.

The most common traditional method is manual inspection using:

- a mirror mounted on a stick,
- a flashlight,
- human judgment under time pressure.

This procedure is:

- **slow** — each vehicle requires time-consuming manual checking,
- **inaccurate** — small or hidden threats are easy to miss,
- **inconsistent** — results vary across guards and conditions,
- **unsafe** — personnel must stand close to potentially dangerous vehicles.

Recent incidents and threat scenarios, such as suspicious objects found near sensitive installations (e.g., locations around the Red Fort mentioned in news reports), have further emphasized the need for reliable, automated under-vehicle inspection systems that can generate high-fidelity visual evidence and support algorithmic anomaly detection.

## 1.2 Project Goal and System Functionality

Our Under Vehicle Surveillance System (UVSS) aims to automate undercarriage inspection by building a robust imaging and processing pipeline. At a high level, our system:

- 1) captures multiple raw images of the vehicle undercarriage (using planar and fisheye cameras),
- 2) undistorts the images (especially crucial for fisheye inputs),
- 3) preprocesses the images to improve clarity and feature consistency,
- 4) stitches the images into a single, large panoramic mosaic,
- 5) (future) feeds the mosaic into a deep learning model to detect anomalies,
- 6) (future) generates an anomaly heatmap and sends alerts to a security console.

In simple terms, we convert a set of raw, distorted camera images taken from below a car into a clean, geometrically accurate panoramic view of the entire undercarriage. This mosaic can then be inspected visually by security operators or processed further for automatic anomaly detection.

## 1.3 End-to-End System Pipeline Overview

Figure 1.1 summarizes the complete pipeline in conceptual form:

$$\text{Capture Images} \rightarrow \text{Undistort} \rightarrow \text{Crop} \rightarrow \text{Histogram Equalization} \rightarrow \text{Feature Detection} \rightarrow \text{Feature Matching}$$

(1.1)

In the following chapters, we explain each component of this pipeline in detail, from camera choice and distortion modeling to stitching, blending, and the planned anomaly detection module.

## 1.4 Narrative System Description

From a narrative perspective, our UVSS addresses a critical limitation of current security practice: the dependence on slow, manual mirror-based inspection. The underside of a vehicle is highly vulnerable because it is largely out of sight and easy to access for an attacker. Our system replaces this manual process with an automated, reproducible pipeline.

We started by experimenting with both planar (normal) and fisheye (omnidirectional) cameras. While planar cameras are easy to handle and have minimal distortion, they require many images to cover the entire undercarriage. Fisheye cameras, in contrast, can capture up to  $180^\circ$  or more field of view in a single frame, drastically reducing the number of shots required. However, this comes at the cost of severe geometric distortion, where straight lines in the world become curved in the image.

To handle this distortion, we adopted the Scaramuzza OCamCalib model, which is widely recognized as a gold standard in robotics and computer vision for omnidirectional camera calibration. After calibrating the camera using checkerboard images, we implemented a

custom C++ undistortion module (`omni_cam_model.cpp` and `omni_cam_utilities.cpp`) that performs pixel-wise inverse projection and resampling to rectify the fisheye images.

These undistorted images undergo preprocessing steps such as cropping and histogram equalization to remove irrelevant regions and enhance contrast. We then experimented with multiple stitching strategies. Standard APIs like the OpenCV stitcher and basic SIFT/ORB+homography pipelines were not stable under low-light and high-shadow undercarriage conditions. We finally converged on OpenPano, an optimized C++ panorama stitching package that provides robust feature matching, RANSAC-based homography estimation, and multi-band blending. The result is a high-quality, seamless panoramic mosaic of the entire vehicle underside.

This mosaic is the foundation of our future anomaly detection work, where we plan to train a convolutional autoencoder purely on “normal” mosaics. During deployment, the reconstruction error will be used as an anomaly score, and a pixel-wise heatmap will highlight suspicious regions. An alert will be sent to a security console whenever the anomaly score crosses a threshold, enabling rapid, focused human verification.

# Chapter 2

## Imaging Setup and Distortion

### 2.1 Camera Configurations: Planar vs. Fisheye

We experimented with two broad categories of cameras to design a practical UVSS:

#### 2.1.1 Planar Camera (Normal Lens)

A planar camera behaves like a standard phone or DSLR camera with a relatively narrow field of view.

- **Characteristics:** Captures visually familiar images; straight lines remain straight, and distortion is relatively mild.
- **Pros:**
  - Simple geometric model (pinhole approximation).
  - Well-supported by standard calibration tools.
  - Minimal distortion makes stitching easier.
- **Cons:**
  - Limited field of view means we need many images to cover the full undercarriage.
  - More images imply higher processing time and potential gaps if coverage is inconsistent.

#### 2.1.2 Fisheye Camera (Omnidirectional Lens)

A fisheye camera is designed to capture an extremely wide field of view, which is ideal for surveillance.

- **Characteristics:** Can capture up to or beyond 180° FoV in one shot, producing circular or heavily curved images.
- **Pros:**
  - Requires significantly fewer shots to cover the entire undercarriage.
  - Simplifies mechanical setup since the camera can be placed at a fixed position along a track or pit.

- **Cons:**

- Strong radial distortion — straight lines become curved.
  - Raw outputs are not directly suitable for stitching.
  - Requires a robust omnidirectional calibration and undistortion pipeline.

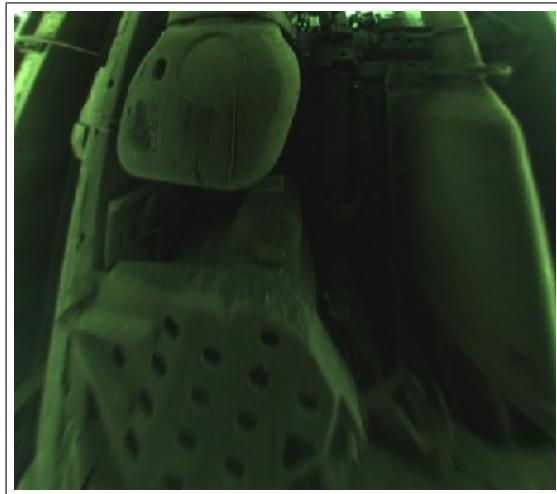
## 2.2 Example Images from the System

To visualize the impact of distortion and undistortion, our pipeline uses the following representative images:

- a raw fisheye undercarriage image,
- the corresponding undistorted fisheye image,
- a raw planar undercarriage image,
- the corresponding undistorted planar image.



(a) Raw fisheye image



(b) Undistorted fisheye image



(c) Raw planar image



(d) Undistorted planar image

Figure 2.1: Representative raw and undistorted images for fisheye and planar cameras.  
Replace filenames with actual project images.

These images clearly show how fisheye distortion warps straight beams and chassis elements, and how the undistortion step restores straightness, making the images compatible with standard stitching algorithms.

## 2.3 Understanding Fisheye Image Distortion

In a normal pinhole model, light rays from the world pass through a small aperture and are projected onto a flat sensor. Straight lines in 3D map to straight lines in the image (up to perspective effects). In a fisheye lens, however, the mapping is non-linear: the lens intentionally bends light rays to squeeze a very wide field of view onto the sensor.

Qualitatively:

- **Normal lens:** light → lens → straight lines remain straight.
- **Fisheye lens:** light → curved lens → straight lines become arcs.

For human viewing, fisheye images are acceptable and sometimes even desirable. For algorithms that rely on geometric consistency, such as stitching, these distortions are catastrophic. It is analogous to trying to assemble a puzzle where each piece is bent differently. Therefore, a robust undistortion stage is mandatory before stitching.

# Chapter 3

## The Science of Undistortion

### 3.1 Image Distortion: Intuitive View

The core challenge is to recover a geometrically faithful representation of the undercarriage from a heavily distorted fisheye image. Distortion arises from the choice of projection model used by the lens. In a fisheye camera, angular distance from the optical axis is mapped non-linearly to radial distance on the sensor, producing characteristic circular or ring-like patterns.

To correct this, we must reverse the projection: for each pixel in the desired undistorted image, we determine which pixel in the original fisheye image it corresponds to, and then sample that location. This requires an accurate mathematical model of the camera's intrinsic parameters and projection equations.

### 3.2 Scaramuzza OCamCalib Model

To achieve high-accuracy correction of omnidirectional images, we use the Scaramuzza OCamCalib model. This model is particularly suitable for wide-angle and fisheye cameras and is widely used in robotics and drone applications.

#### 3.2.1 Why OCamCalib?

- **Designed for omnidirectional cameras:** It accurately models extreme wide-angle lenses beyond the capability of simple pinhole or basic fisheye models.
- **Higher accuracy:** For very distorted inputs, it is more precise than the standard OpenCV fisheye model.
- **Compatible toolchain:** It integrates naturally with MATLAB for calibration and can be exported for C++ use.
- **Proven in practice:** It has been used extensively in robotic navigation and 3D reconstruction tasks.

#### 3.2.2 Calibration Procedure

The calibration pipeline is as follows:

- 1) **Image Acquisition:** Capture approximately 30–40 images of a checkerboard pattern using the fisheye camera, with the board at varying orientations, tilts, and positions.
- 2) **Corner Detection:** Use OCamCalib in MATLAB to detect the checkerboard corners across all images.
- 3) **Parameter Estimation:** OCamCalib estimates the intrinsic camera parameters, including:
  - distortion coefficients,
  - projection polynomial,
  - inverse projection polynomial.
- 4) **Export:** The computed parameters are saved as MATLAB structures and then converted into a C++ friendly format.
- 5) **Integration:** These parameters are integrated into our C++ codebase.

### 3.3 Python Implementation for Undistortion

Our repository contains a single key Python file:

- `ocam_model.py`

This file implements the core undistortion logic based on the calibrated OCamCalib parameters.

#### 3.3.1 Pixel-wise Inverse Projection

The undistortion algorithm effectively performs the following for each output pixel  $(u_{\text{out}}, v_{\text{out}})$  in the desired rectified image:

- 1) Use the inverse projection polynomial to compute the corresponding 3D ray direction in camera coordinates.
- 2) Intersect this ray with the original fisheye image plane to find fractional coordinates  $(u_{\text{in}}, v_{\text{in}})$ .
- 3) Sample the intensity at  $(u_{\text{in}}, v_{\text{in}})$  using bilinear interpolation.
- 4) Write the sampled value into the undistorted output image at  $(u_{\text{out}}, v_{\text{out}})$ .

#### 3.3.2 Computational Considerations

This process is computationally intensive because it involves:

- evaluating polynomials for each pixel,
- performing floating-point ray projections,
- executing interpolation operations.

However, the payoff is high: the resulting images are geometrically accurate, with straight lines in the world appearing straight in the image. This correctness is crucial for downstream tasks such as feature detection and matching.

# Chapter 4

## Preprocessing

Once images are undistorted, they are not yet ready for stitching. They may still contain empty regions, black borders, and low contrast due to shadows and uneven lighting. Our preprocessing pipeline focuses on two key tasks: cropping and histogram equalization.

### 4.1 Cropping Strategy

Undistortion often introduces large black regions around the useful content, especially near the corners. Retaining these regions can:

- confuse feature detectors,
- increase the chance of incorrect feature matches,
- slow down processing without adding useful information.

Therefore, we crop the undistorted images along the dimension perpendicular to the direction of vehicle motion. This:

- removes black margins and irrelevant areas,
- focuses on the undercarriage region that actually contains informative structures,
- improves stitch quality and stability by reducing outlier keypoints.

### 4.2 Histogram Equalization

The undercarriage environment typically suffers from:

- low and uneven illumination,
- strong shadows cast by components,
- highly non-uniform brightness due to directional lighting.

To tackle this, we apply histogram equalization to enhance contrast and make features more prominent:

- **Function:** Redistributes pixel intensities so that dark regions become visible and

bright regions maintain details.

- **Impact:** Increases the number of stable keypoints detected by feature detectors.
- **Benefit:** Leads to more reliable feature matching and homography estimation during stitching.

Empirically, we observed that applying histogram equalization significantly improved both the quantity and quality of keypoints found along repetitive undercarriage structures such as beams, pipes, and chassis frames.

# Chapter 5

## The Stitching Pipeline

### 5.1 Stitching Experiments and Limitations

Before arriving at the final stitching solution, we conducted several experiments using conventional tools and libraries. These attempts helped us understand the specific challenges of undercarriage imagery.

#### 5.1.1 OpenCV Stitcher Module

Our first attempt used the built-in OpenCV Stitcher module, which internally uses feature detection, matching, and warping.

- **Pros:** Easy to use, minimal additional code, widely documented.
- **Cons Observed:**
  - Too slow for our multi-image sequences.
  - Failed frequently under low-light conditions typical of undercarriage scenes.
  - Sensitive to low texture and repeated patterns, resulting in misalignments.

#### 5.1.2 SIFT / ORB + Homography (Manual Pipeline)

Next, we implemented a custom pipeline with:

- SIFT or ORB feature detection,
- descriptor matching,
- RANSAC-based homography estimation,
- warping and blending using OpenCV primitives.

While this gave us more control, it exposed several practical limitations:

- performance degraded for large image sequences,
- accuracy suffered when the field of view varied significantly between frames,
- strong shadows and low-contrast regions led to unstable homography estimates,
- the amount of manual tuning required was high.

### 5.1.3 OmniCV-Lib

We also experimented with OmniCV-Lib, which can transform fisheye images into equirectangular or planar formats.

- **Pros:** Provided a neat way to convert omnidirectional views into a more stitching-friendly representation.
- **Cons:**
  - Introduced artifacts at the poles and edges.
  - These artifacts propagated into the stitching and often degraded the final mosaic.
  - Overall, not sufficiently robust for production use in our UVSS context.

From these experiments, we concluded that standard, out-of-the-box tools were not stable enough for our specific undercarriage imaging conditions. We needed a more optimized and robust stitching engine.

## 5.2 The Chosen Solution: OpenPano

### 5.2.1 Overview of OpenPano

OpenPano is a highly optimized panorama stitching framework implemented in C++. It is designed for speed, robustness, and high-quality output. Compared to basic OpenCV methods, OpenPano provides:

- fast, robust feature extraction (SIFT-like),
- efficient feature matching across multiple overlapping frames,
- RANSAC-based homography or transformation estimation,
- high-quality warping,
- advanced multi-band blending.

### 5.2.2 Why OpenPano Worked Better

OpenPano addressed many of the issues we faced with previous attempts:

- **Stability under low texture:** Its robust matching and RANSAC-based estimation handle repetitive undercarriage structures better.
- **Performance:** The implementation is optimized for speed, making it suitable even when multiple frames must be stitched.
- **Blending quality:** Multi-band blending removes harsh seams and reduces ghosting, which is crucial when some parts of the undercarriage are only partially visible in different frames.

### 5.2.3 Multi-Band Blending

Multi-band blending operates in the frequency domain, splitting images into different frequency bands and blending each band separately:

- low frequencies ensure smooth global transitions,
- high frequencies preserve sharp edges and details.

For UVSS, this leads to mosaics where:

- seams are practically invisible,
- overlapping beams, pipes, and other components appear continuous,
- ghosting and double edges are significantly reduced.

### 5.3 Final Mosaic Output

After undistortion, preprocessing, feature extraction, matching, and blending, we obtain a final panoramic undercarriage mosaic. This mosaic has the following properties:

- **Geometric correctness:** structural elements such as beams and axles are straight and properly aligned.
- **Continuity:** overlapping regions from multiple images are merged seamlessly.
- **Readiness for analysis:** the mosaic is suitable for visual inspection by security personnel, as well as for future algorithmic anomaly detection.

In practice, this means a security operator can look at a single image to understand the entire undercarriage instead of interpreting multiple fragmented views.

# Chapter 6

## Future Work: Anomaly Detection

The stitched mosaic forms a natural input for automated anomaly detection. Our proposed approach uses a convolutional autoencoder (CAE) trained on normal undercarriage mosaics.

### 6.1 Dataset Creation

The first step is building a dataset of “normal” vehicle mosaics:

- capture many vehicles with no anomalies,
- run them through the full pipeline (undistortion, preprocessing, stitching),
- store the resulting mosaics along with metadata (vehicle type, lighting conditions, etc.).

Data augmentation (e.g., brightness and contrast changes) can be used to:

- make the model robust to lighting variations,
- simulate different environmental conditions.

### 6.2 Convolutional Autoencoder Model

A convolutional autoencoder consists of:

- an encoder that compresses the input into a lower-dimensional latent representation,
- a decoder that reconstructs the input from the latent code.

We train the CAE only on normal mosaics. As a result:

- it learns to reconstruct typical undercarriage structures very well,
- it fails to reconstruct regions it has never seen (e.g., foreign objects, tampered parts).

## 6.3 Anomaly Detection Workflow

The anomaly detection workflow is:

- 1) **Training:** Train the CAE on normal mosaics until reconstruction is stable and accurate.
- 2) **Inference:** For a new undercarriage mosaic, pass it through the trained CAE to obtain a reconstructed version.
- 3) **Reconstruction Error:** Compute the pixel-wise reconstruction error:

$$\text{Reconstruction Error} = |\text{Input Mosaic} - \text{Reconstructed Mosaic}|.$$

- 4) **Anomaly Score:** Aggregate the error into a single anomaly score (e.g., mean or max error).
- 5) **Thresholding:** If the score exceeds a pre-computed threshold (from validation data), mark the mosaic as suspicious.
- 6) **Heatmap Generation:** Visualize the pixel-wise error as a heatmap overlaid on the original mosaic.
- 7) **Alert System:** If an anomaly is detected, send the mosaic and heatmap to the security console for immediate human verification.

## 6.4 Advantages of the CAE Approach

- **Unsupervised:** No need for labeled anomaly data, which is hard to collect.
- **Spatial awareness:** The convolutional architecture captures spatial patterns of the undercarriage.
- **Explainability:** Heatmaps highlight *where* an anomaly is, not just that something is wrong.

# Chapter 7

## Presentation Poster

### Under Vehicle Surveillance System

Bhavay Goyal (24110070) Yash Goyal (24110399)  
Divyansh Sharma (24110113) Chaitanya Bhoite (24110086)

Advisor Prof. Shanmuganathan Raman

IIT Gandhinagar



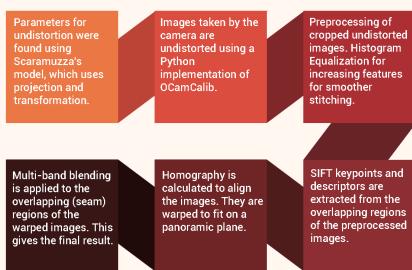
#### Introduction

Vehicle security inspections are critical for detecting hidden contraband or unauthorised modifications hidden underneath a vehicle. This has become much more relevant currently following the unfortunate events near the Red Fort. This project presents an Under Vehicle Surveillance System (UVSS) designed to automate and enhance the process of inspection of the undercarriage of cars and trucks. The aim of this project is to create a stitched image from several images of a vehicle's undercarriage, which can be checked for anomalies.

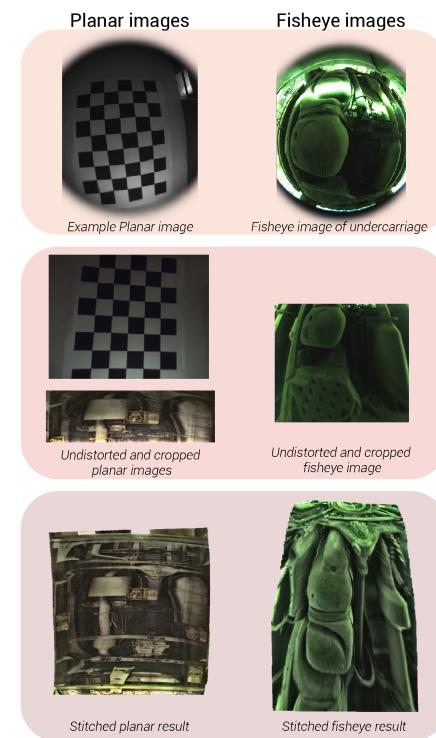
#### Pipeline Architecture

- 1 The camera (Name of Camera) is placed normal to the plane of the road. Multiple images of the undercarriage are taken in quick succession. These images can either be fisheye or planar.
- 2 The images taken by the camera are then undistorted using Scaramuzza's Omnidirectional Camera Model. The undistorted images are cropped from the sides, perpendicular to the direction of travel of the car, to remove artefacts from undistortion.
- 3 Undistorted and cropped images are processed. Histogram Equalization is applied on these images, to make stitching smoother. Images are then stitched and blended together using OpenPano library. The result will be used for anomaly detection.

#### Image Processing



#### Results



#### Future Work

We did not have a dataset to train a model for anomaly detection. The logical next step for this project is to incorporate anomaly detection as well, creating an entire Under Vehicle Surveillance System.

#### Acknowledgements and References

We sincerely thank Prof. Shanmuganathan Raman for giving us this wonderful opportunity to work on this project. We also extend our thanks to Mr. Arjun Boddu and Ms. Sayali Dutta for their continuous support throughout the semester. We also acknowledge IIT Gandhinagar for providing us resources and infrastructure to make this possible.  
[1] S. K. Nayar, "Foundations of Computer Vision," Department of Computer Science, Columbia University. [Online]. Available: <http://pov.cs.columbia.edu>.  
[2] D. Scaramuzza, A. Martinelli, and R. Siegwart, "A Toolbox for Easily Calibrating Omnidirectional Cameras," Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Oct. 2006, pp. 5695–5700.  
[3] Y. Wu, "How to Write a Panorama Stitcher," Blog Post, 2016. [Online]. Available: <https://pwwwyyxx.com/blog/2016/How-to-Write-a-Panorama-Stitcher/>.

# Chapter 8

## Conclusion and Key Technical Highlights

### 8.1 Conclusion

In this project, we implemented a complete Under Vehicle Surveillance System (UVSS) pipeline starting from raw image capture to a high-quality stitched mosaic of the vehicle undercarriage. The main contributions are:

- a robust imaging setup combining planar and fisheye cameras,
- high-accuracy fisheye undistortion using the Scaramuzza OCamCalib model,
- a carefully designed preprocessing pipeline with cropping and histogram equalization,
- a reliable stitching engine using the OpenPano C++ package with multi-band blending,
- a clearly defined roadmap for a convolutional autoencoder-based anomaly detection system.

The resulting system is stable across diverse lighting conditions and camera configurations. It replaces manual mirror-based inspection with an automated, reproducible, and scalable solution. The final panoramic mosaic is easy for security personnel to interpret and forms a solid foundation for machine learning–based anomaly detection.

### 8.2 Key Points for Visitors and Graders

For quick reference, Table 8.1 summarizes the most important technical aspects of our UVSS and why they matter.

Component	Technical Detail	Why It Matters / Why It Is Better
Problem / Goal	Full automation of undercarriage inspection.	Replaces slow, inaccurate, and potentially dangerous manual mirror checks.
Distortion Model	Scaramuzza OCamCalib omnidirectional camera model.	Superior accuracy for fish-eye lenses compared to standard OpenCV models; essential for true geometric correction.
Implementation	Custom C++ undistortion modules ( <code>omni_cam_model.cpp</code> , <code>omni_cam_utilities.cpp</code> )	Demonstrates understanding of camera projection theory and ability to implement inverse projection polynomials efficiently.
Preprocessing	Cropping and histogram equalization.	Enhances relevant features and reduces noise, making stitching robust even under poor lighting and shadows.
Stitching Engine	OpenPano optimized C++ panorama stitcher with multi-band blending.	Solves instability and ghosting issues seen with OpenCV Stitcher; produces seamless, high-quality mosaics.
Future Work	Convolutional coder-based anomaly detection.	State-of-the-art unsupervised approach: detects anomalies using reconstruction error without needing labeled anomaly examples.
Final Output	Single, geometrically accurate panoramic mosaic of the undercarriage.	Provides a comprehensive, easy-to-analyze view suitable for both human operators and automated inspection algorithms.

Table 8.1: Key technical highlights of the UVSS project.

# References

1. S. K. Nayar, “Foundations of Computer Vision,” Columbia University. [Online]. Available: <http://fpcv.cs.columbia.edu>
2. D. Scaramuzza, A. Martinelli, and R. Siegwart, “A Toolbox for Easily Calibrating Omnidirectional Cameras,” *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2006, pp. 5695–5700.
3. Y. Wu, “How to Write a Panorama Stitcher,” 2016. [Online]. Available: <https://ppwwyyxx.com/blog/2016/How-to-Write-a-Panorama-Stitcher/>
4. Project Source Code Repository: <https://github.com/BhavayGoyal/UVSS>