NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

FYP PROJECT C009

# Development of Human-Machine Interfaces for Stair Climbing Robot

Presenter: Bhatt Swapneel Dipak

Supervisor: Prof. I-Ming Chen

# Table of Contents



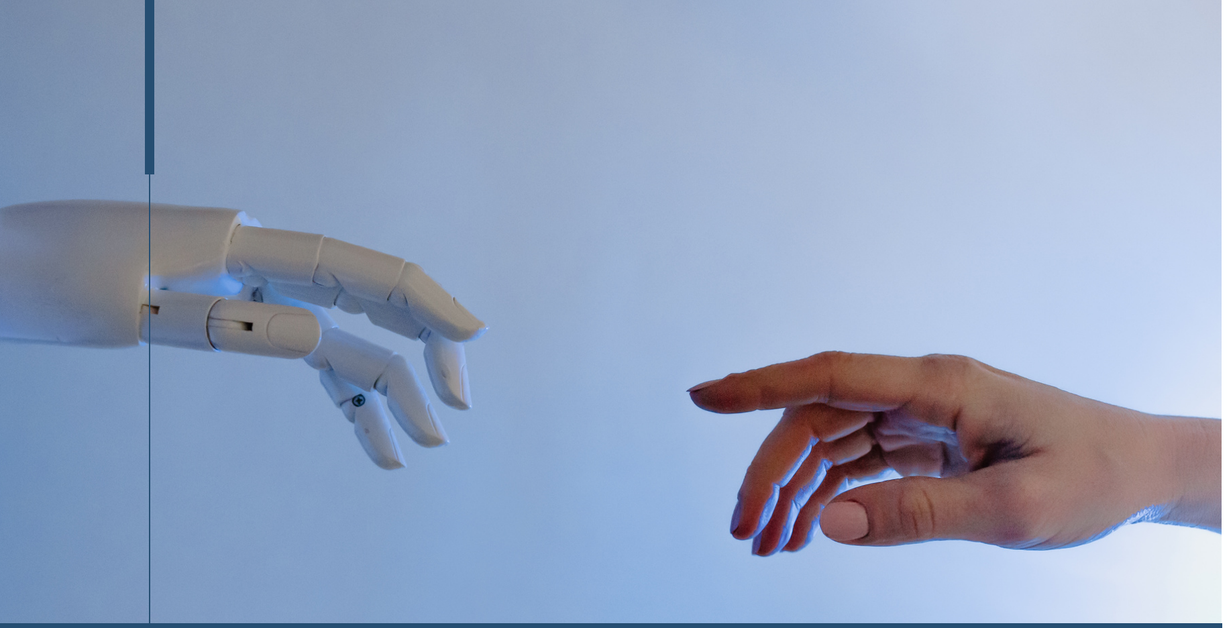
# Introduction

# Background

## INDUSTRY 4.0 IS ON THE RISE

The construction industry is no exception to the advent of the fourth industrial revolution

Robotics plays a key role: carrying out dangerous and labor-intensive tasks previously carried out by humans



# Technological Enablers

## MobileMan

Construction Site Robot on which cameras are mounted for tele-operation

## Human-Machine Interface

A functional and efficient Human-Machine Interface (UI) for tele-operation

## Computer Vision

Fish eye lenses used for top-view and wide-angle view

# Objectives

1. Create a system design integrating hardware and software

2. Camera Selection and Procurement

3. Design and Fabrication of Camera Mounts

4. Software Design and Testing

5. Integration with UI and Implementation on Robot

# MobileMan and Key CV Concepts

# MobileMan

## PRIMARY OBJECTIVE

Navigate in a construction site, carrying heavy materials or clearing obstacles, and climbing up stairs

## BODY

A heavy chassis with low centre of gravity.
Caterpillar Tracks to move on uneven surfaces
Robotic arm/forklift/cage attachment to carry/move items

## CONTROL

Tele-Operation control is done via a joystick, which is integrated with the ROS (Robotics Operating System) framework of the robot.

## COMPUTER VISION

Through this project, fish-eye cameras were mounted on the frame shown in the photo, to allow computer vision

# Key Computer Vision Concepts Used

- **Fish-Eye Lens**

  Wide Angle Lens used to capture the environment

- **Camera Calibration**

  Calculating Intrinsic and Extrinsic Parameters of a fish-eye camera

- **Lens Distortion**

  Change in image shape and perspective due to lens properties

- **Image Homography**

  Projective Mapping between two different image planes

- **Feature Extraction**

  Extracting unique feature points and keypoints for image analysis

- **Image Stitching**

  Combining images from multiple cameras to get wide-angle image

- **Perspective Transformation**

  Transformation applied on an image that changes locations of all pixels

# Past Work and Important Developments

### Zhengyou Zhang's Flexible Camera Calibration Method

Flexible camera calibration technique, used as a basis of many calibration functions nowadays

### SIFT and SURF Algorithms

SIFT Algorithm by David G. Lowe, and SURF by Bay, Tuytelaars, and van Gool used for feature extraction

### Wang et al. Perspective Transformation

Developed a car-parking application using perspective transformation to get bird's eye view of car wheels

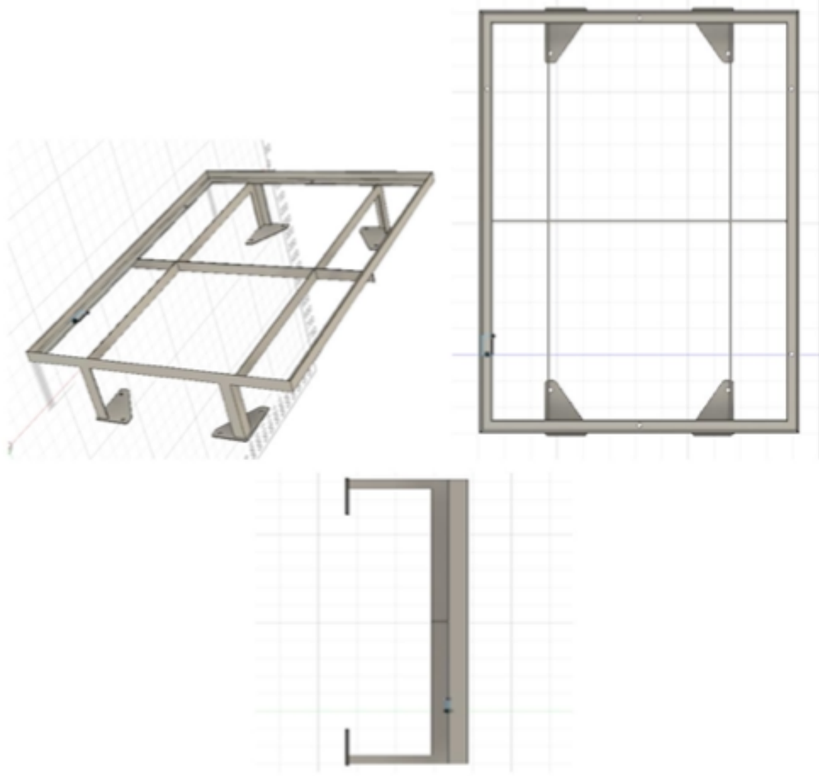
# Hardware Design and Fabrication

# Camera Selection

## STAR LIGHT SONY IMX291

- Camera selection was done on basis of field of view, cost and form factor
- The chosen camera provides a 180-degree field of view, with maximum resolution of 1920x1080p at 30fps
- USB2.0 interface, and image compression in MJPEG, YUV2 and H.264 format

# Camera Mount Design



## 3D PRINTED MOUNTS TO ATTACH CAMERAS TO THE MOBILEMAN TOP FRAME

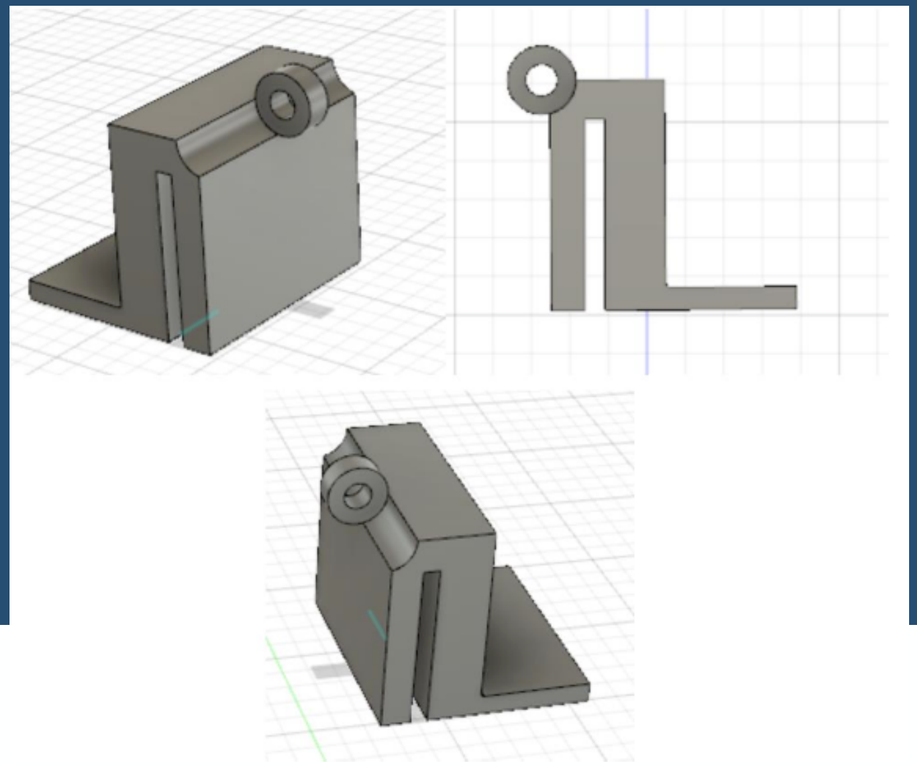
The camera module provides little to no protective casing, and needs to be protected in rugged environments

Mounts for the cameras were needed to be designed to attach the cameras to the frame shown in the picture, as well as provide protection from the elements

Design consists of two parts, a mount base and camera holder, and are connected by a hinge joint to vary the camera angle

Designed using AutoDesk Fusion360

# Camera Mount Base



## Objectives

To design a base that can

- Tightly fit to the MobileMan frame
- Distributes Camera Load over a wide surface area
- Connects to the camera holder as a hinge joint

## Features

Slot in the middle that snap fits onto the MobileMan frame

Wide support structure at the back to carry load, and reduce bending moment on the holder

Hinge Joint slot at the top to connect to camera holder

# Camera Holder

## Objectives

To design a camera holder that can

- Hold the camera securely, and provide a stable support for the camera
- Protect the camera from the elements
- Connect with the base holder in a hinge joint connection

## Iterative Design

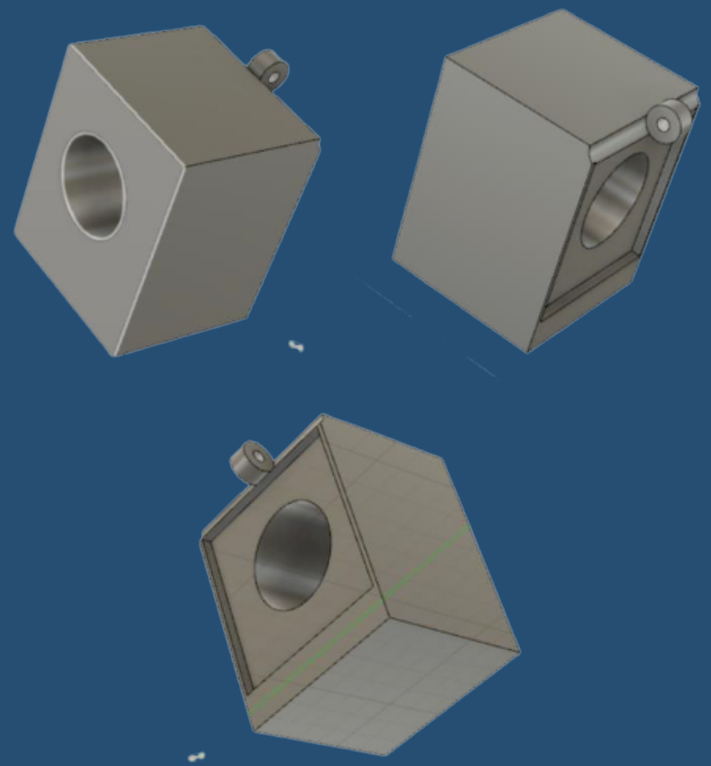
The camera holder design took place in two design iterations.
The factors considered in the holder design were : air flow to prevent camera from overheating, secure protection for the camera circuitry, and amount of 3D print material used to reduce cost

# Camera Holder Iteration 1

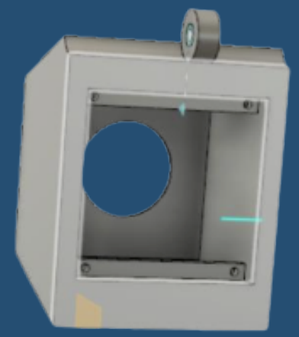
## A BULKY, BUT PROTECTIVE DESIGN

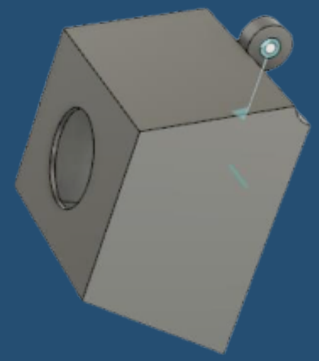
The first iteration provided maximum protection to the camera, however had plenty of defects

- Inefficient bulky structure, wastes a lot of 3D print material and has long fabrication time
- Box shape does not provide significantly more structural support than a sleeker design
- Although it has good elemental protection, it has no air flow considerations and will  cause camera to heat up



# Camera Holder Iteration 2

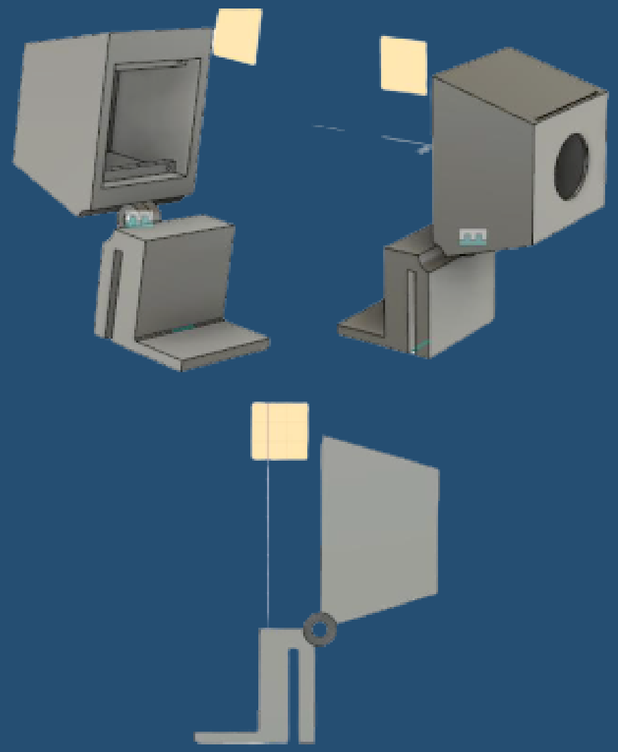




## A SLEEKER, MORE EFFICIENT DESIGN

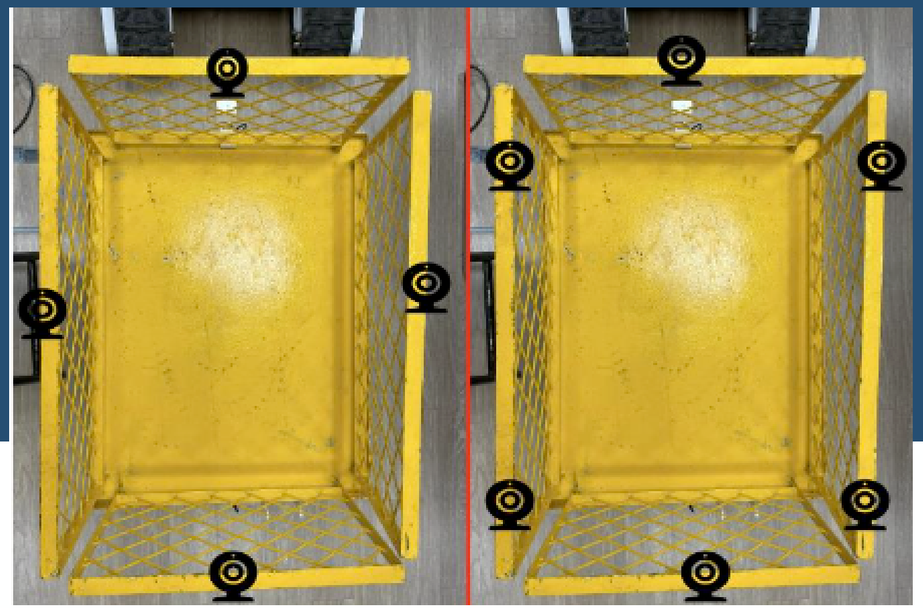
- A more efficient design, that provides similar protection without wasting extra material
- Has extra slits cut into the bottom to facilitate airflow, and prevent the camera from overheating

# Final Assembly



A hinge jointed assembly that provides support and protection to the camera, and allows for changes to the viewing angle

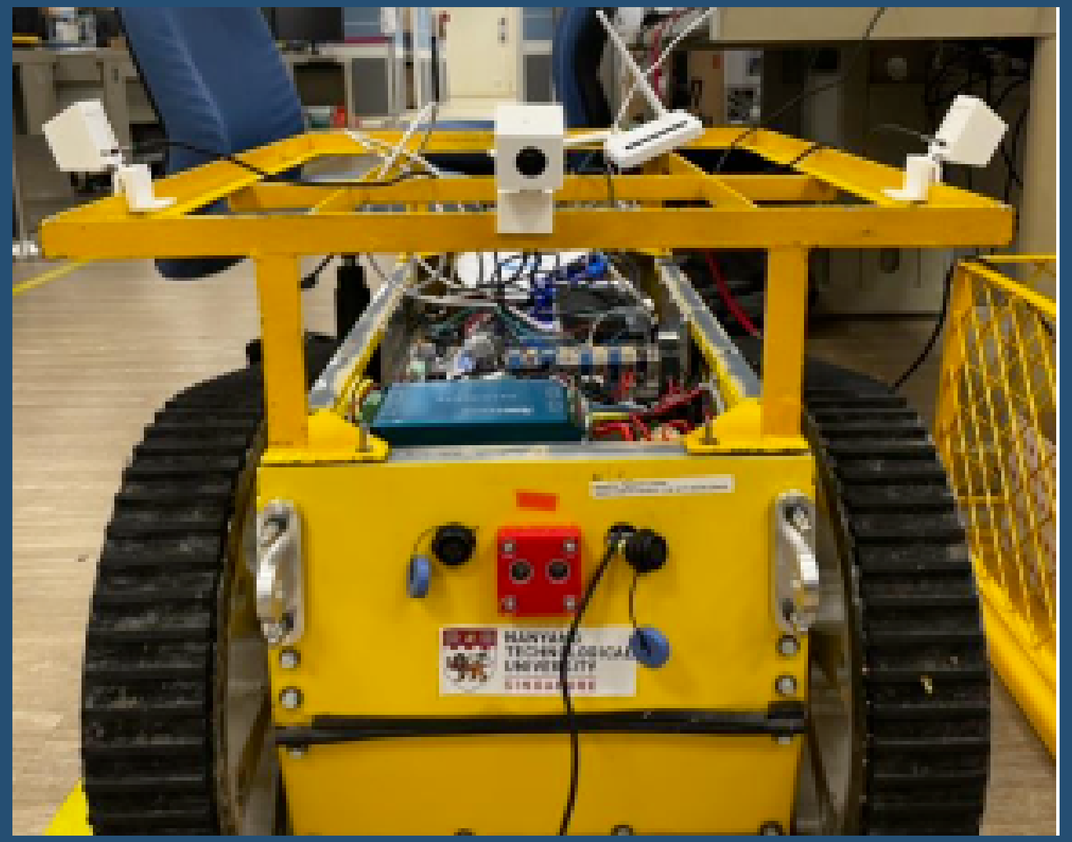
# Camera Placement and System Setup



## TWO SYSTEM LAYOUTS, 4-CAMERA AND 6-CAMERA

4 camera system is cheaper, and has lower computational load.

However, 6-camera setup was chosen to provide better field-of-view overlap that will help in wide angle view creation.



Cameras mounted on the front of MobileMan

# Software Design and Programming

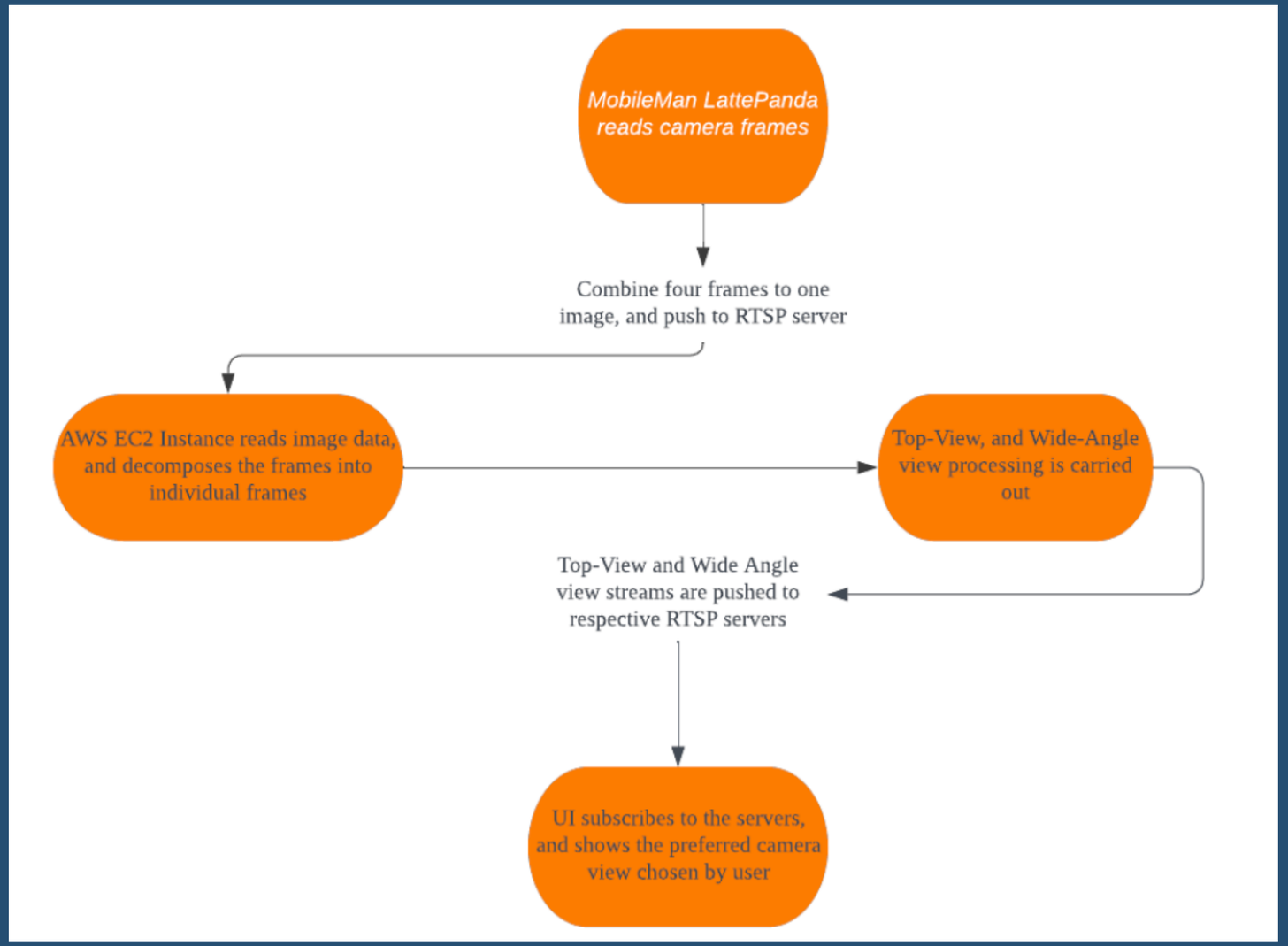
# Key Functionalities of the Software

Reading camera frames, and streaming them to the cloud server

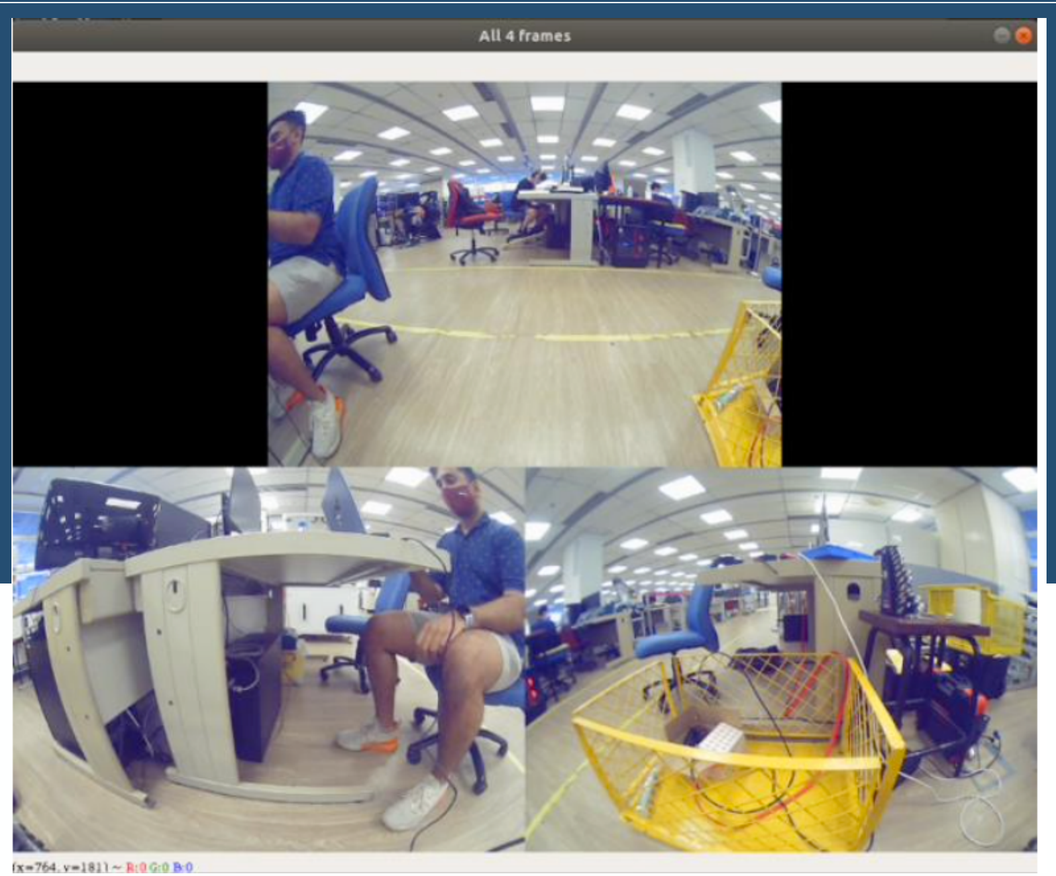
Processing camera frames from the cloud server, and creating the required views for the tele-operation user

Allowing user interface to subscribe to the cloud server, and display the camera views requested by the user

# Process Flow Chart



# Reading Camera Frames

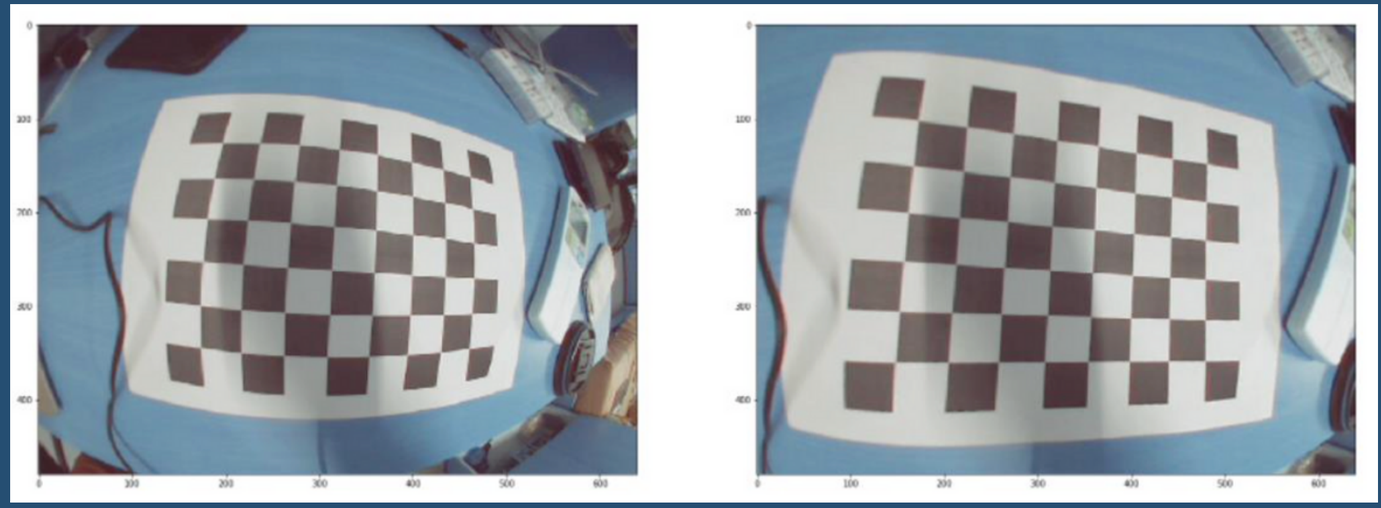



The onboard LattePanda computer reads each individual camera frame.

The individual frames are combined into a single frame ready for streaming.

The combined frame is streamed to the rtsp server in the cloud.

# Calibration and Image Undistortion



Before the frames are streamed, the distortion caused by the fish-eye lens has to be removed.

We use Zhang's flexible  camera calibration technique to obtain intrinsic camera parameters.

Using these parameters, we use in-built functions in OpenCV to undistort the images.

Note the pixel loss in the first image, which has been accounted for in the second image.

# KeyPoint Detection and Matching

Keypoints are points of interest, or features in an image that help in image analysis.

Used for Image Stitching to generate the wide angle view.

SIFT algorithm is used for keypoint detection, and Brute Force Matcher is used to match the keypoints.

A checkerboard is placed in the overlap region to increase the number of good matches.

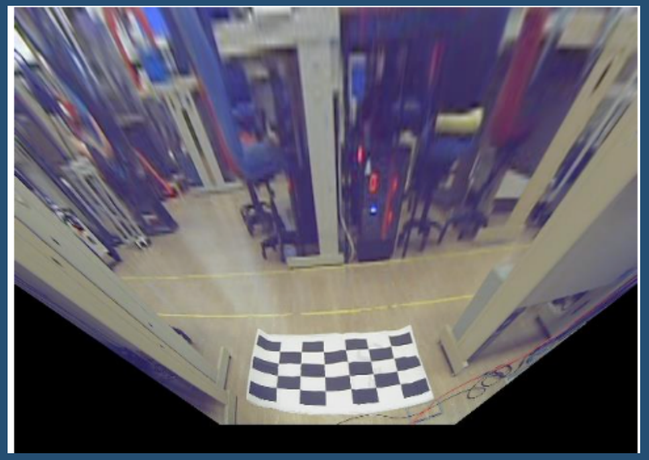
# Image Stitching

After obtaining the keypoints and their matches, we can stitch images together to generate a wide-angle image. Images are stitched from left to right

The stitching process took place in the following steps:

- The left image, and front image were first inverted, then stitched from front to left, and the final image was inverted again to obtain correct orientation.
- The front and the right image were stitched together
- The left-front image, and the front-right image were stitched together to obtain the final result

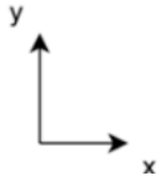
# Bird's Eye View Generation

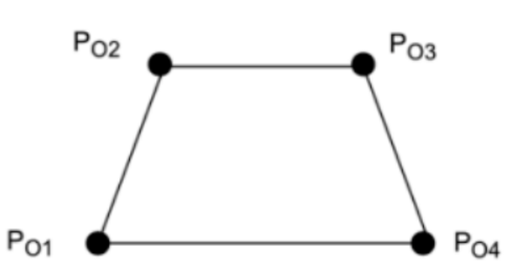


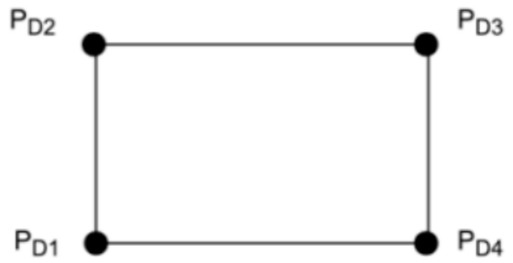
Perspective transformation is carried out on the camera images to obtain a "bird's eye view" of the environment.

This view provides a much more intuitive feeling for tele-operators to guage obstacles close to the robot body.

The checkerboard in the images above is used to calibration the transformation matrix.




$$Chessboard\_size = 5 \text{ x } 7 \text{ squares}$$

$$P_{01} = corner1 \qquad P_{D1} = P_{01}$$

$$P_{02} = corner2 \qquad P_{D2} = (P_{D1}.X, P_{D1}.Y + 5)$$

$$P_{03} = corner3 \qquad P_{D3} = (P_{D1}.X + 7, P_{D1}.Y + 5)$$

$$P_{04} = corner4 \qquad P_{D4} = (P_{D1}.X + 7, P_{D1})$$

# Bird's Eye View Frame

Each individual bird's eye view frame is generated on the cloud server.

The images are cropped to focus on the regions of interest.

The final frame is generated by placing the bird's eye views along the top view of the MobileMan in their respective positions

# Implementation – Integrating with the UI and Performance

# Hardware Performance

The hardware and streaming performance was tested over long field tests where the robot was driven across the NTU campus via tele-operation.

Camera Mounts held the camera stable, leading to a smooth video stream.

The streaming quality was also quite optimal, apart from few spots where the 4G network was weak

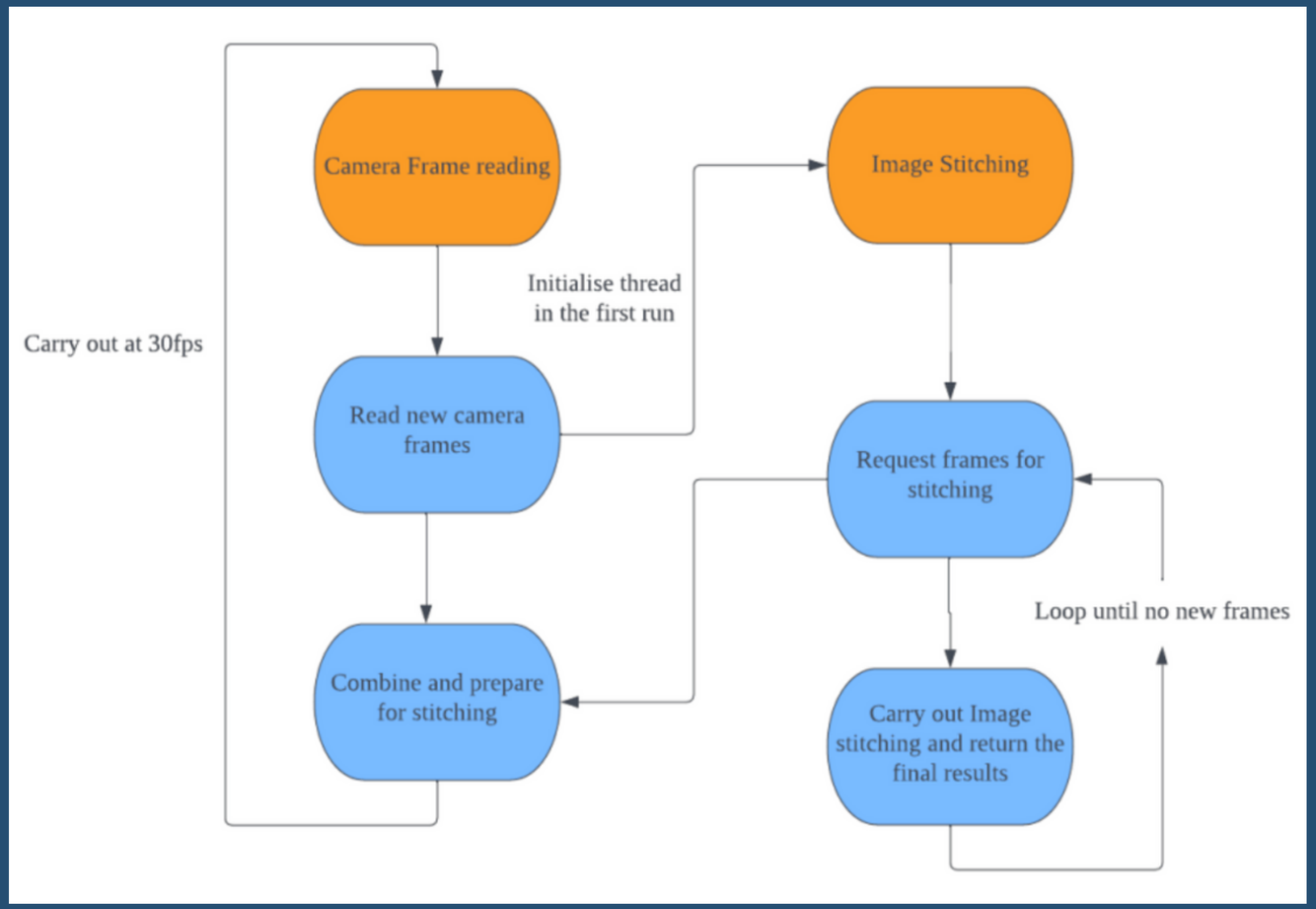
# Real Time Wide Angle View Processing

After calibrating the image stitching parameters on still images, the software was tested on videos to generate real-time wide-angle views.

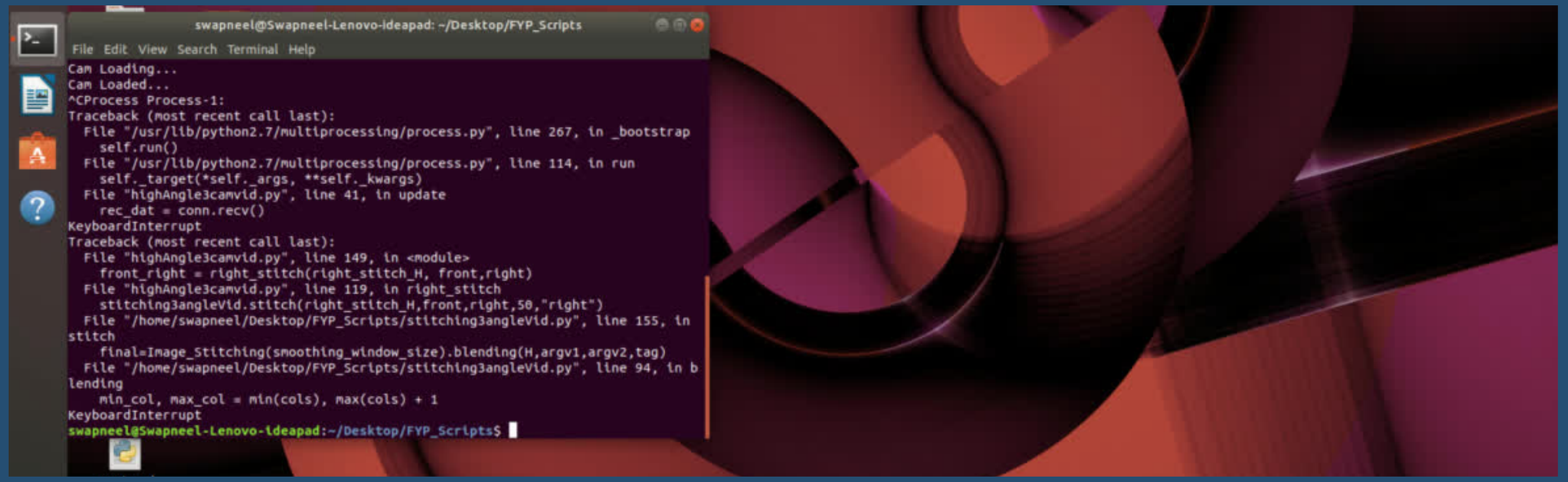
Single thread performance was laggy, hence multithreading was utilised. However, multithreading did not perform optimally as well.

Processing time creating an increasing delay, and result framerate was <10fps.

Due to this, the wide-angle view was not implemented in the final UI.



# Real-Time Wide Angle Video Result



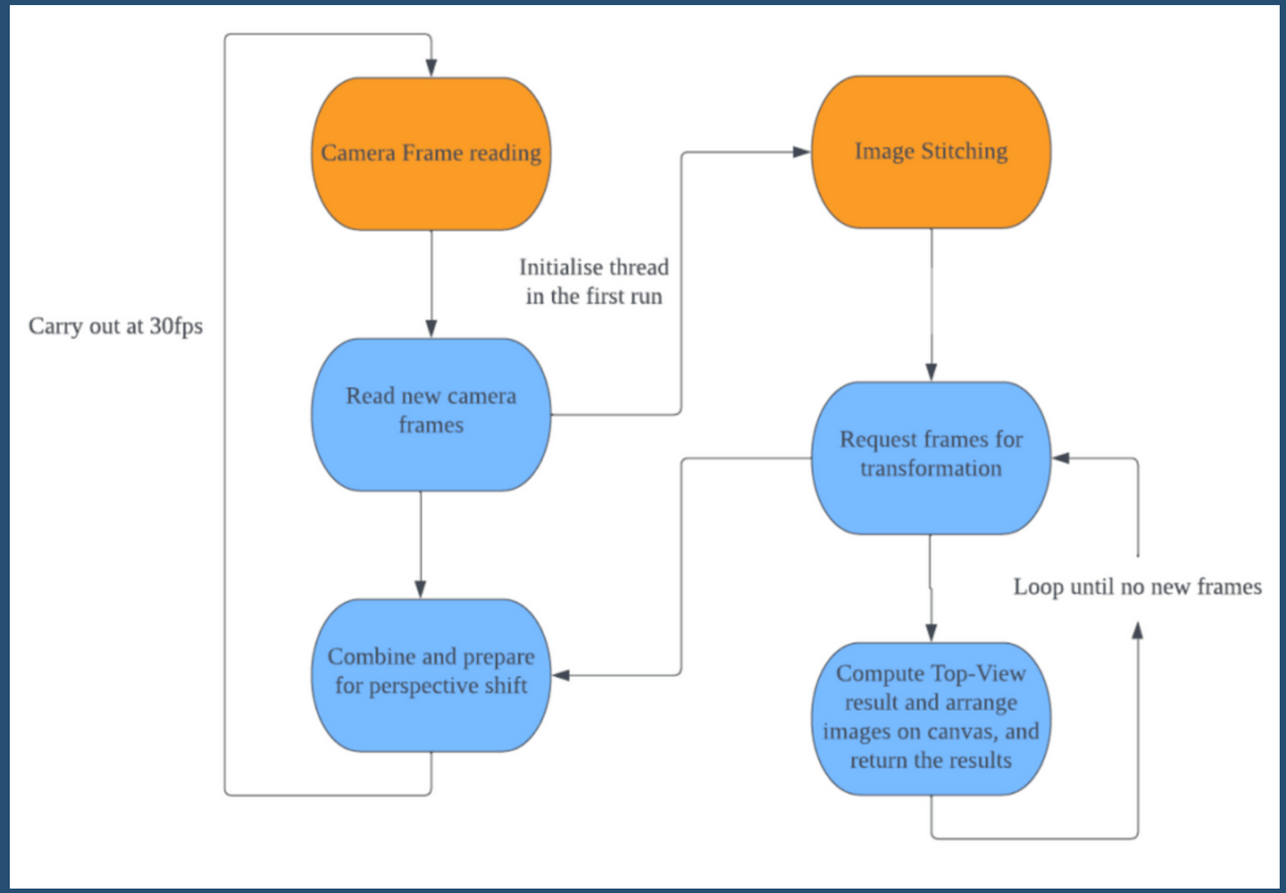
# Wide Angle Video Result After Processing (not Real-Time)

# Real Time Bird's Eye View Processing

Similar to wide-angle, a real-time video for top view was computed.

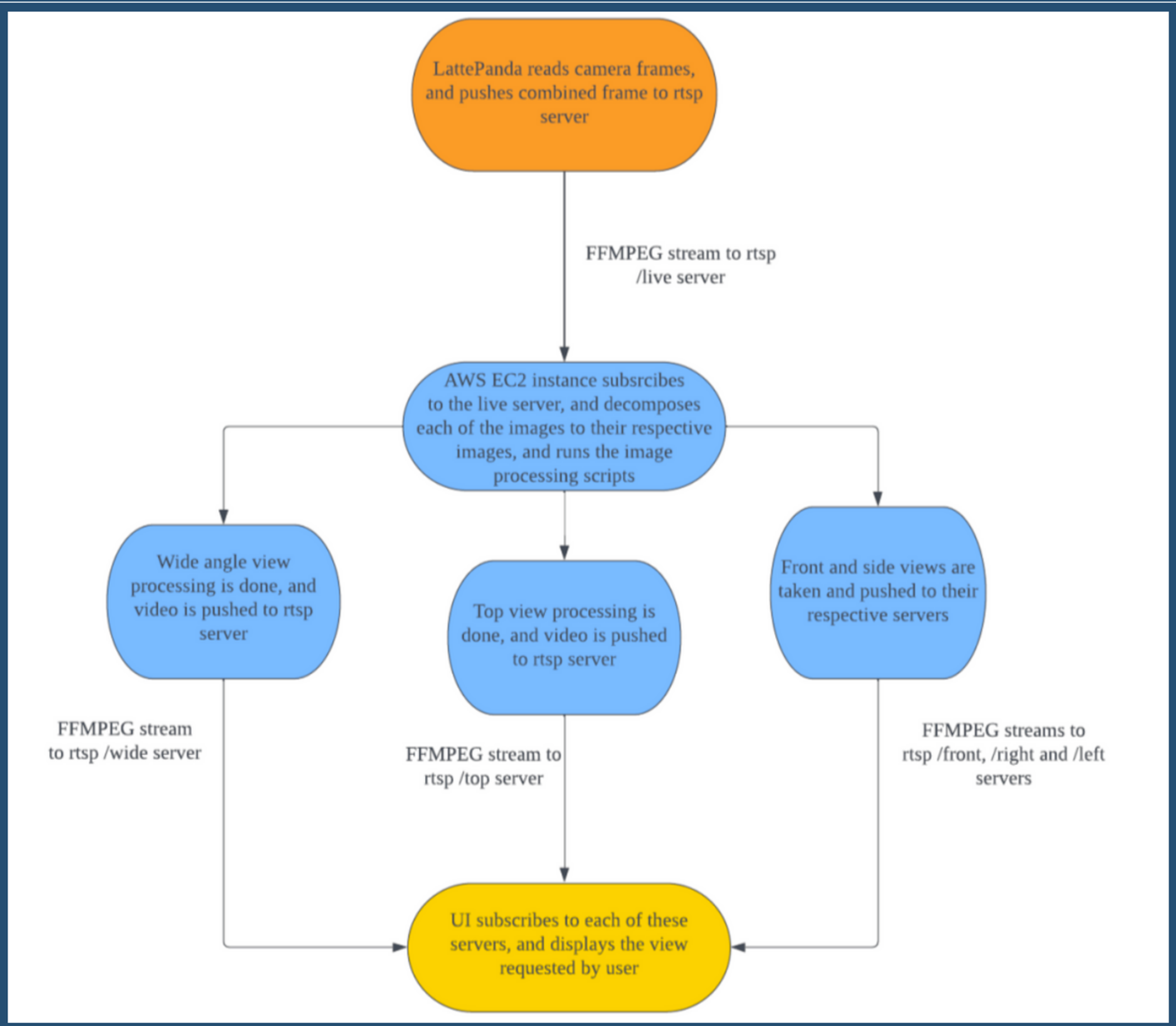
Multithreading was used to improve performance and reduce delays.

Results were near-perfect, and deemed operationally ready for implementation into the UI.



# Cross Platform Streaming

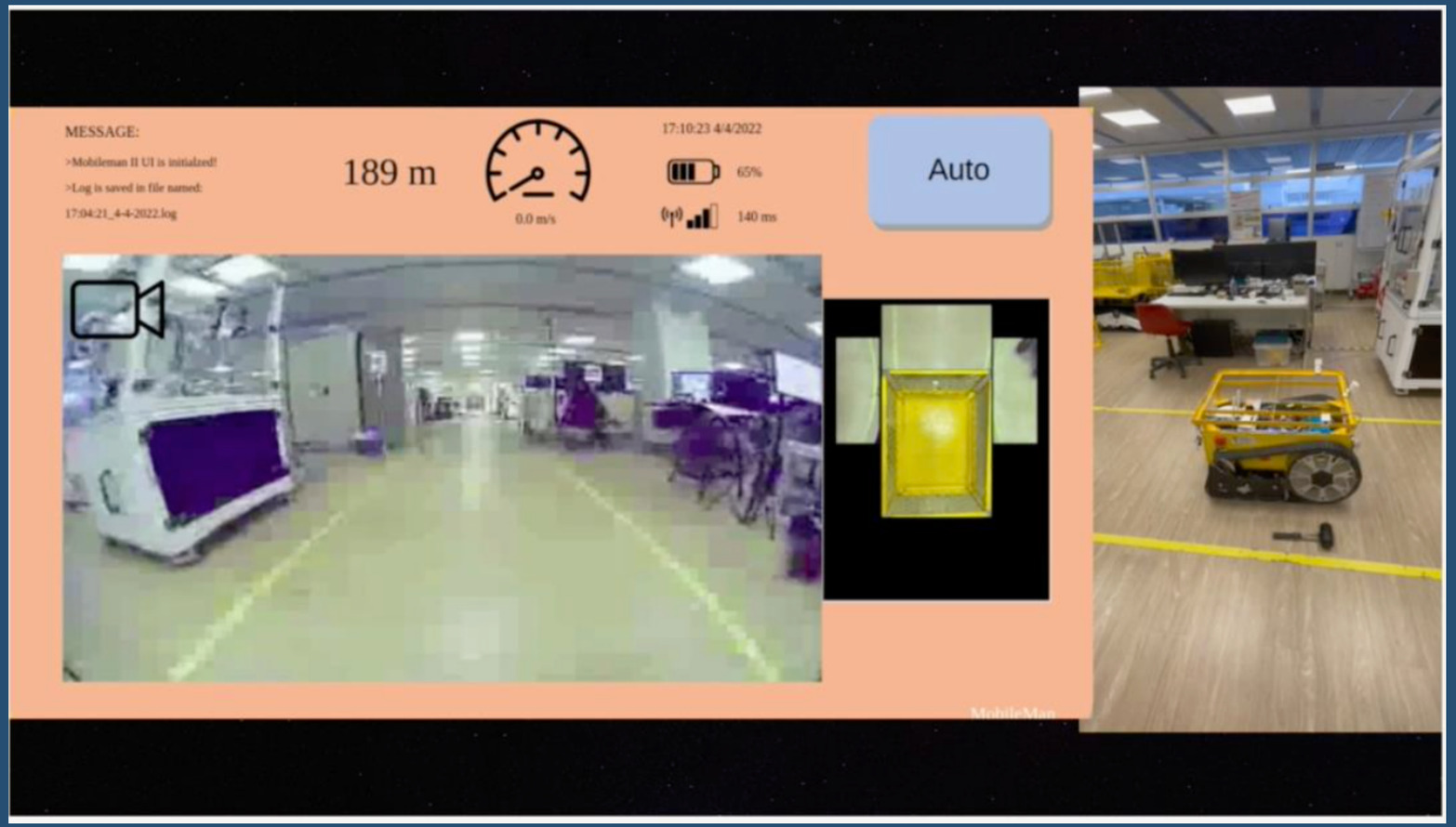
RTSP streaming using FFMPEG was used to stream videos between on-board computer, AWS EC2 server, and the UI computer.



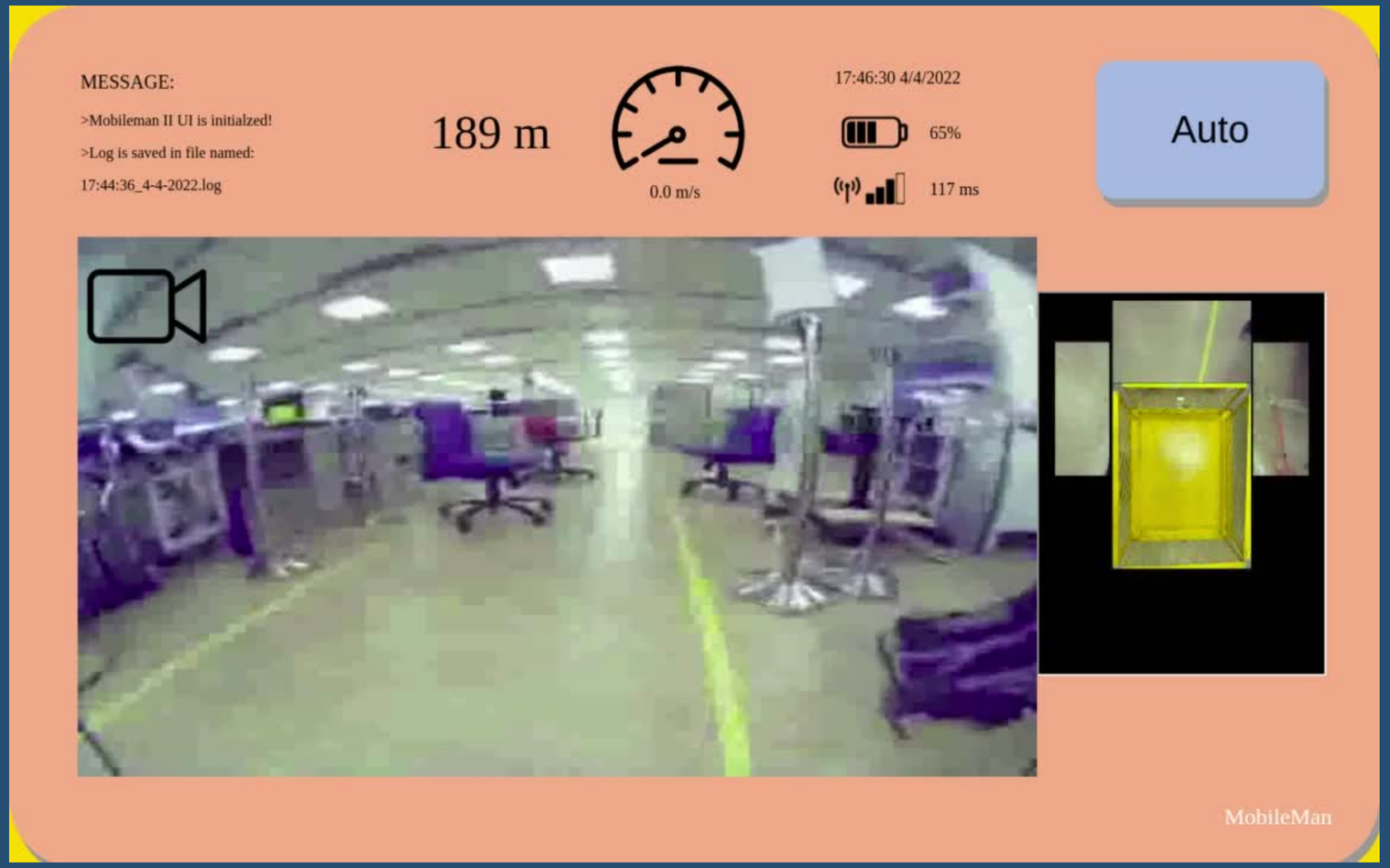
# User Interface

We can see the front view, and top view from the robot.

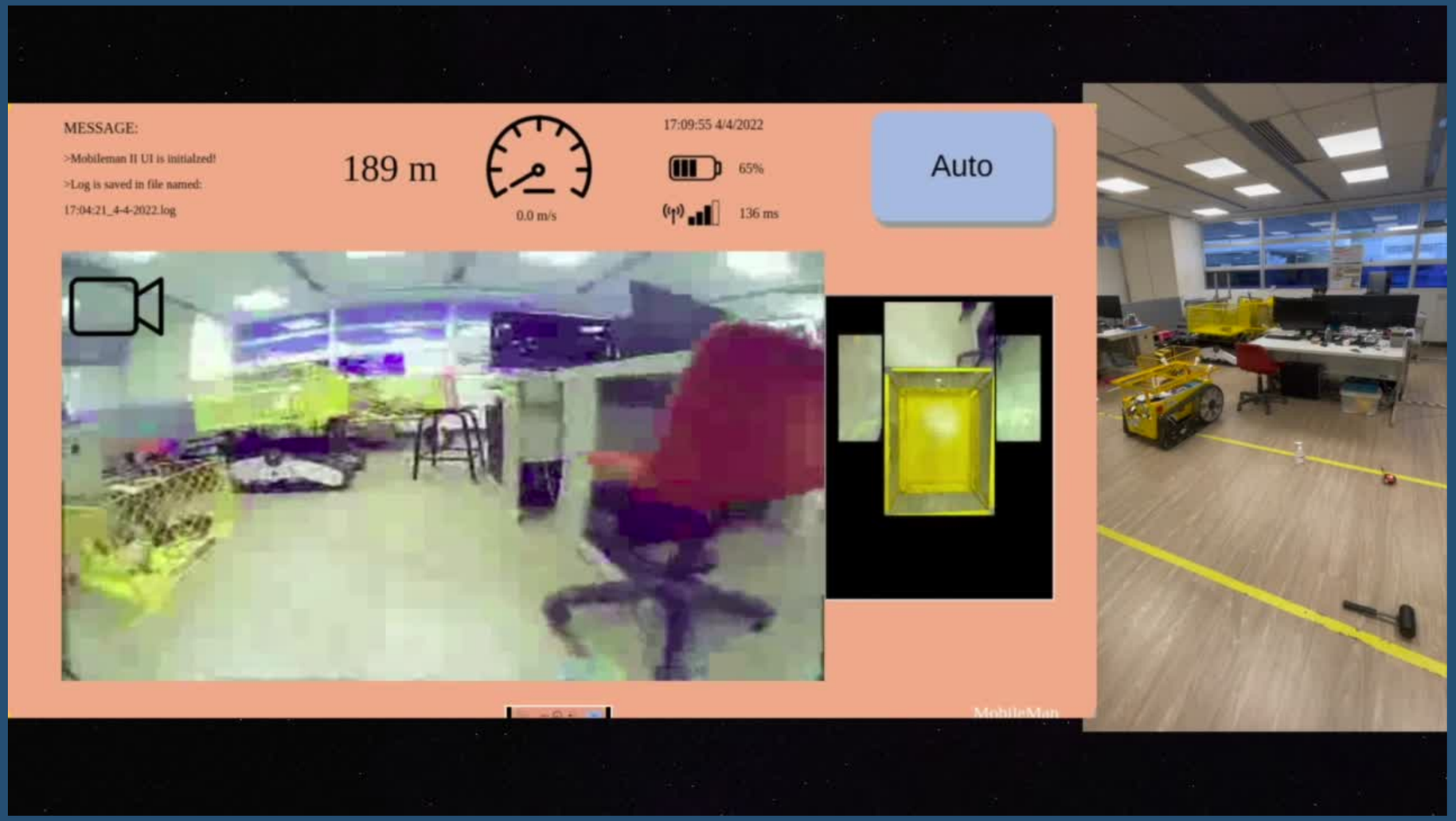
Wide-Angle view is not included due to performance issues.



# Real Time UI Integration  Result



# Moving In and Out of Parked Location



# Future Works and Conclusion

# Scope for Future Works

## Real Time Wide Angle View Processing

The problem of the delay prevented this view from being included in the final UI. This performance issue could be solved using more powerful computation hardware, or by developing a more efficient algorithm

## Top View Processing Improvement

Though real-time top view was sufficiently efficient, a small delay still built up. This could be solved by improving the hardware, or via better algorithms. Programming in C++ could also help, as OpenCV is natively written in C++.

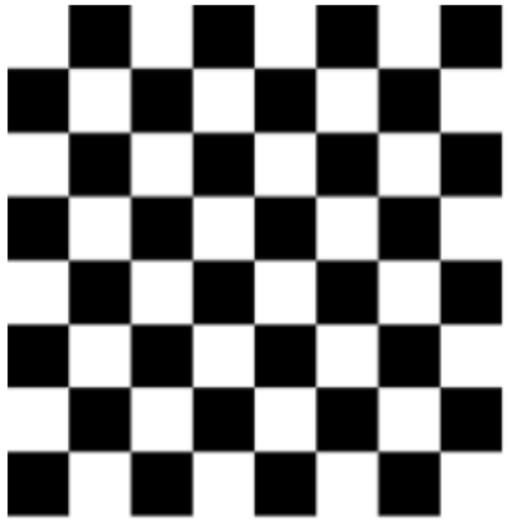
## RTSP Stream Quality

Currently, the RTSP stream could not handle a 720p, or full 1080p HD resolution. This prevents the user from having a completely pleasant experience when tele-operating. This problem could be solved by using better streaming algorithms, or using a faster network.

# Thank You

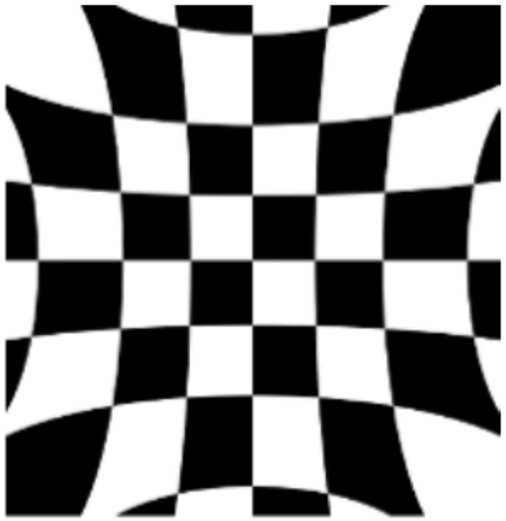
# Appendix

# Radial Distortion



No distortion

Positive radial distortion
(Barrel distortion)



Negative radial distortion
(Pincushion distortion)

# Feature Extraction

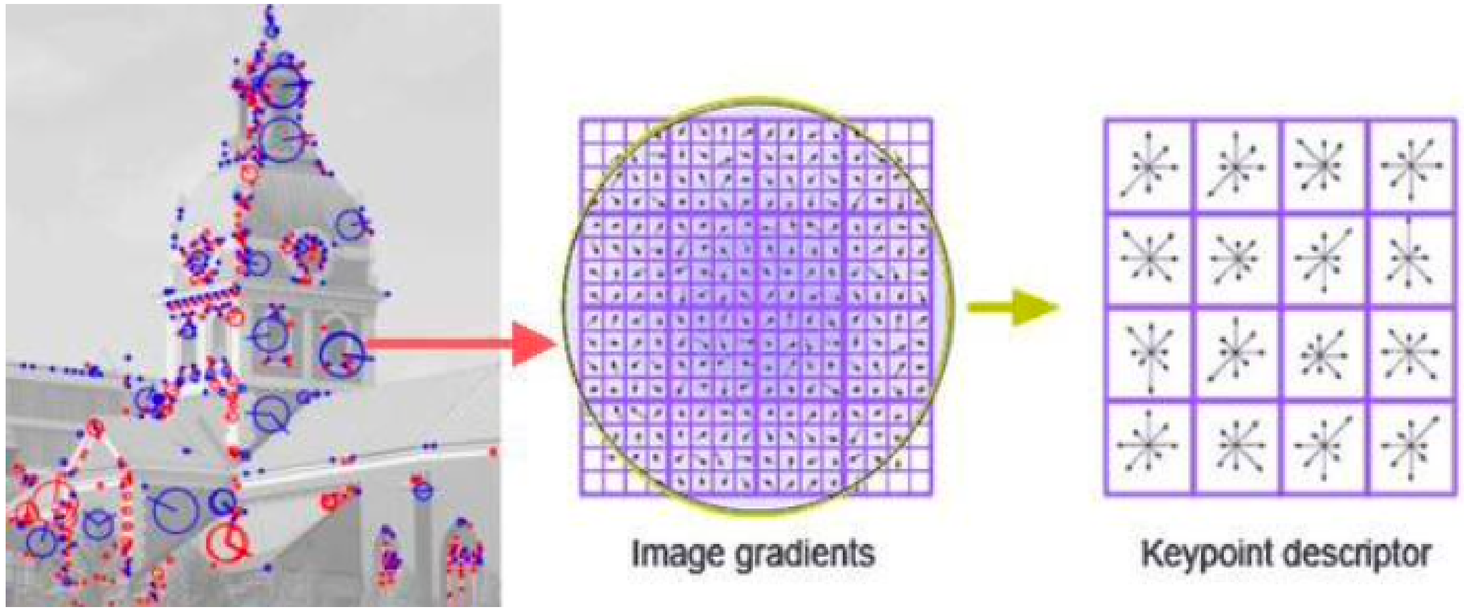


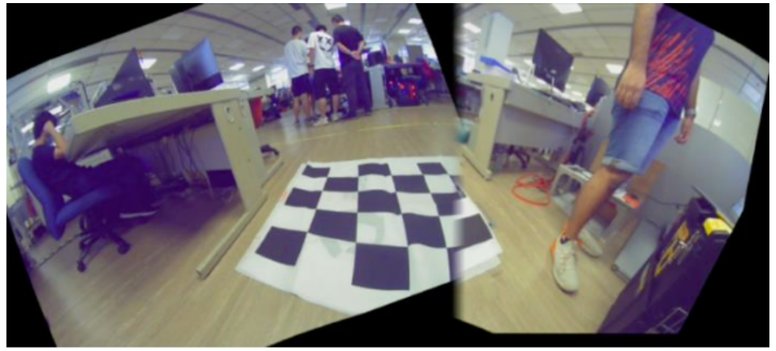

*Figure 4 An Example of Feature Extraction*

# Image Stitching

# Performance of Image Stitching with 4-camera Setup

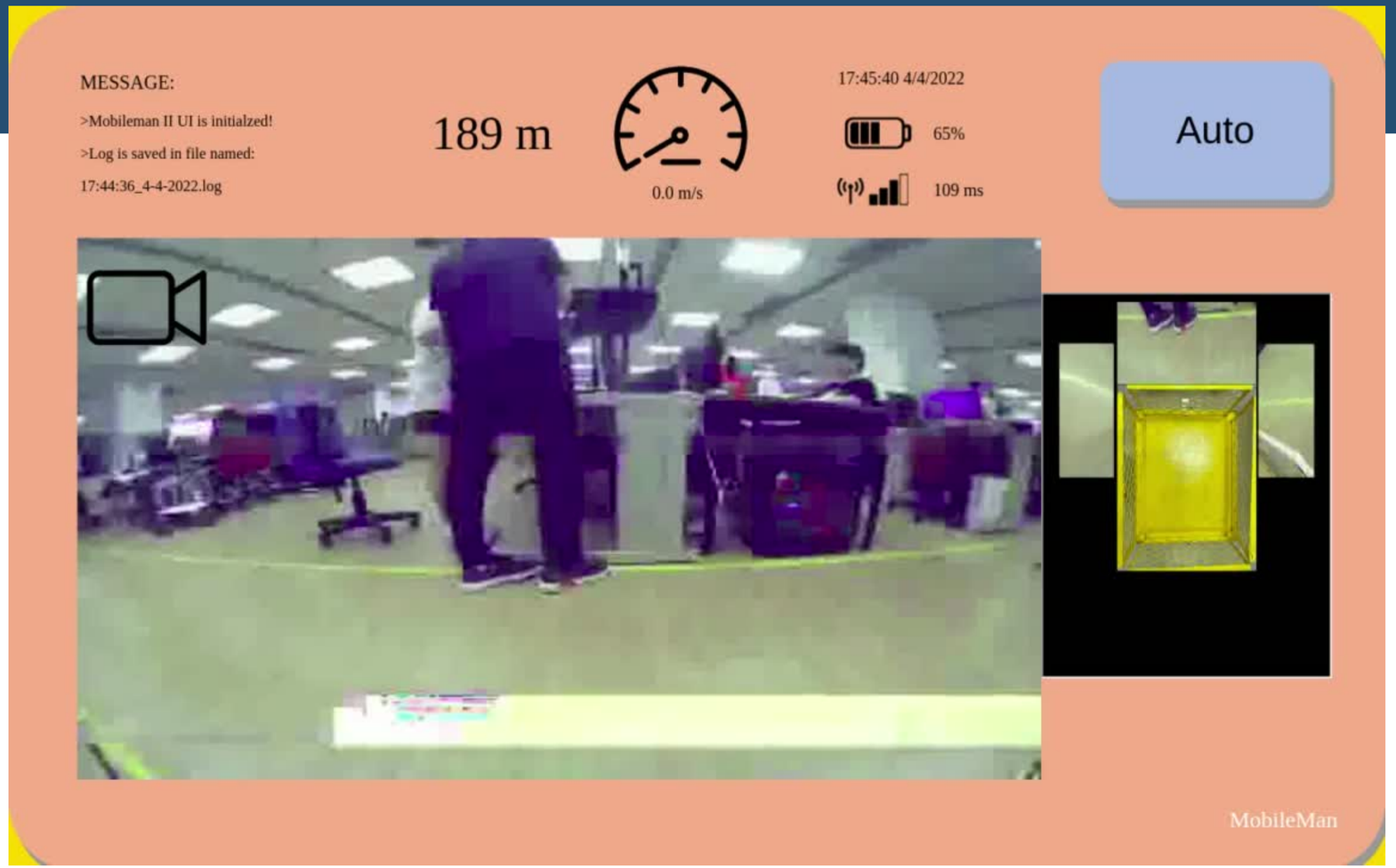
Front and Side Images for stitching

Images Rotated to match homography



Stitching Results

# Full UI Performance Vid



# Field Tests