ECE 445
SENIOR DESIGN LABORATORY
DESIGN DOCUMENT

REMOTE ROBOT CAR CONTROL SYSTEM WITH RGBD CAMERA FOR 3D RECONSTRUCTION

Team Members:
Yuhao Ge, yuhaoge2
Hao Chen, haoc8
Junyan Li, junyanl3
Han Yang, hany6
Teaching assistant: Yiqun Niu
Sponsor: Prof. Pavel Loskot

Friday 24th March, 2023

Contents

1 Introduction

1.1 Problem and Solution Overview

1.1.1 Problem

The existing remote control systems utilized in applications such as robotics and drones encounter difficulties when operating in complex environments, resulting in inadequate control over the machine. Moreover, for remote control, operators usually lack a comprehensive view of their surroundings, thereby increasing the likelihood of accidents. Therefore, it is essential to design a remote control system that enables the user to effectively maneuver the car in intricate surroundings, while providing a comprehensive view of the environment, even when the vehicle is beyond the range of visual perception.

1.1.2 Solution

The use of 3D reconstruction technology is becoming increasingly prevalent across various industries, including the automotive sector. Our proposed solution involves the utilization of 3D reconstruction technology[3] to provide users with a better understanding of their surroundings while controlling a car. To achieve this, we plan to equip the robot car with an RGBD camera, which will capture the surrounding data and transmit it to a remote server (personal computer). The 3D reconstruction module will then run on the remote server in real-time[2], generating a 3D model of the car's surroundings.

With our product, users will be able to control the robot car remotely via a joystick while also viewing the car and its surroundings from a third-person perspective[1]. The perspective can be easily shifted to allow for a better understanding of the car's surroundings.

1.2 Visual Aid

The figure on the left shows a car being navigated around a location. The figure on the right shows a user inspecting the environment by viewing reconstructed images of the surroundings and maneuvering the car using a joystick.

Subfigure 1: Environment Scenario

Subfigure 2: User Interface with Joystick Controller

Figure 1: Pictorial Representation of our Solution

1.3 High-Level Requirement List

1. The robot car can be remotely controlled using a joystick. The control system should allow real-time control with a delay of no more than 200 ms.
2. Four-wheel independent driving is necessary to achieve omnidirectional movement for the car, so it can move in complicated environment.
3. The WIFI bandwidth between the Raspberry Pi and the server should reach 10 MB/s to guarantee the enough image information be transmitted.
4. An appropriate down sampling method should be proposed to transfer the 640*480 image to 480*360 while containing as much informations. An efficient and reliable 3D reconstruction algorithm should also be proposed to produce a reconstructed 3D model based on the 640*480 image or even lower.
5. To ensure real-time performance, the 3D reconstruction algorithm must be efficient, with a calculation time of less than 0.2 seconds for each frame and a frame rate of 5 FPS.

2 Design

2.1 Block Diagram

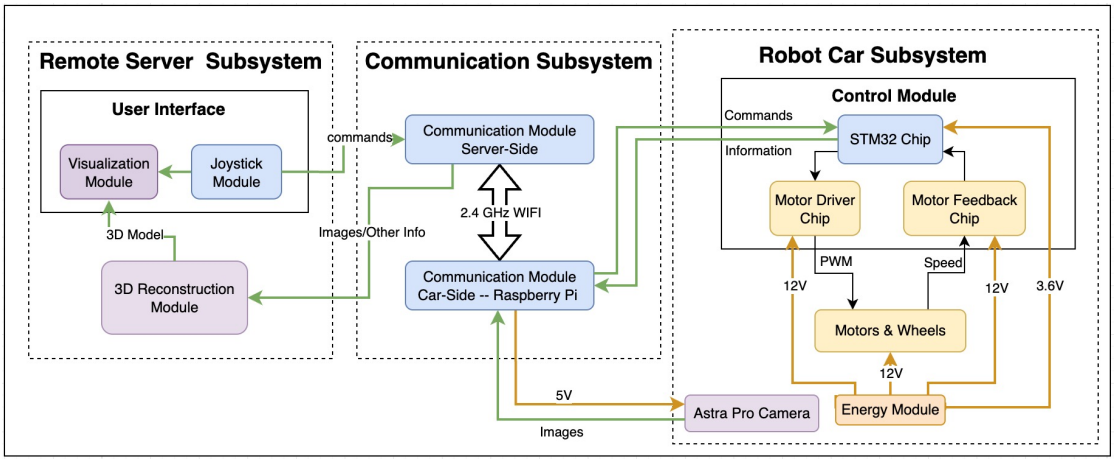

Figure 2: Block Diagram of our Solution

We basically divide our solution into three subsystems that are framed with dotted lines,

1. a **remote server subsystem** which will handle the visualization and the 3D-reconstruction tasks for users to observe the surroundings,
2. a **communication subsystem** which works as a communication bridge between the remote server and robot car for data and commands transfer, and
3. a **robot car subsystem** which holds a camera to gather information and can be controlled by a joystick.

Each subsystem contains several modules that work together to achieve the specific feastures.

2.2 Car Subsystem

The car subsystem is a movable robot car platform, including a car platform, wheels, motors, a STM32 based control module, and an energy module with a Li-ion battery pack. The main functionality of the car subsystem is that it can accept movement commands to perform actions such as moving and turning, and provide an expandable platform for various devices placement such as Raspberry Pi and RGBD camera.

2.2.1 Control Module

For control module, We use an STM32 based robot control board to control the motors. STM32 is a widely used low-energy microcontroller that can communicate with other devices through SPI, UART, and PWM, etc. The control board will receive raw control signals from Raspberry Pi through SPI or CAN bus.

Based on the car's current movement status, control signals are parsed using an incremental PID control algorithm to generate the correct PWM signal for the four motors. Four AM2857 DC motor driving chips are used to receive the PWM signals and drive the motor accordingly. The motion and position information of the car will also be collected and sent to the server if needed.

Requirement	**Verification**
1. Control Module can receive commands from Raspberry Pi and control the motors accordingly. 2. Can compute the PWM for each motor based on the commands to enable moving omnidirectionally within several preset speed ranges. 3. Car can turn at a speed of about 25 degrees/second according to the command. 4. Car can move omnidirectionally and turn simultaneously based on the commands.	1. Perform a moving test with custom movement commands sent from Raspberry Pi, i.e., left and right panning, forward and backwards, rotation, and a combination of these movements. 2. Evaluate the performance of these movements, i.e., how well they meet the user's expectations. 3. Write down a reference table for the data interface of control module. 4. Record and evaluate the performance of the average time delay between a given input and the output.

Table 1: R&V Table for Control Module

2.2.2 Car Platform

An expandable car platform is the base of our car subsystem, with wheels, motors, control module, communication module (car side) and RGBD camera to be assembled and fixed on it. Because we have many devices to assemble on the car platform, the structure of the robot car platform must be strong enough to carry a variety of equipment, including four 150g motors, one 200 g STM32 control board, one 50g Raspberry Pi, 500 g RGBD camera and one 170 g Li-ion battery pack, the overall 1520 g weight loading must be guaranteed. When working under the

vibration of the motors and bumps during movement, the structural integrity and stability of the equipment placement must be guaranteed. Besides, the robot car platform should be facilitated for assembling and disassembling devices such as motors, Raspberry Pi, and RGBD camera.

Requirement	Verification
1. The net weight of the car platform is limited to less than 2 kg. 2. The structural integrity of a 3-4 kg loading weight should be able to guarantee. 3. The stability of the equipment placement under the vibration must be guaranteed. No equipment loose or drop during long working periods, i.e., 1 hour.	1. Weighing the robot car platform, its net weight should not exceed 2 kg 2. Load testing of the robot car platform with 3 4 kg of weight. 3. Perform a movement test on bumpy roads, and do acceleration, deceleration, and turning operations. 4. Record the car condition among the tests as the results.

Table 2: R&V Table for Car Platform

2.2.3 Wheels and Motors

Input: PWM signal from motor driving chips
Output: Motor rotation speed and direction

In order to provide a robot car that can move freely in complex environments, the McNamee wheel was selected to support omnidirectional movement. The McNamee wheel is a classical mechanic for omnidirectional movement and consumes less energy than the traditional differential wheel model. However, the adoption of the McNamee wheel requires the control module to have individual control over each wheel and motor. The motor needs to provide sufficient power for the movement of the robot car system and guarantee a reasonable running speed under a limited weight load. We choose to use the MD520Z30_12V motor to satisfy our requirement.

Requirement	Verification
1. Robot car can move freely in complex environments and achieve omnidirectional movement. 2. The motor needs to provide sufficient power to enable the robot car to run at a normal speed of 0.5-1m/s when forward and backward under a reasonable weight load 1520 g as discussed.	1. Perform a moving test with predefined movement to cover an omnidirectional movement set, i.e., left and right panning, forward and backward, rotation, and a combination of these movements. 2. Perform a forward and backward movement test under a 1520 g weight load, the speed is within 0.5-1m/s. 3. Perform a movement test on bumpy roads, and do acceleration, deceleration, and turning operations. 4. Take a video for the omnidirectional movement test, and record the speed and rotation performance in a table.

Table 3: R&V Table for Wheels and Motors

2.2.4 Energy Module

The devices on the robot car platform require a removable power supply. Meanwhile, these devices have different voltage and current requirements, e.g., the Raspberry Pi requires 5V DC and consumes 6.25W(max), and the DC motor driving chips require 11-16V(12V recommended) DC and motors consume 4W(max), etc. So, the energy module needs to provide a different DC voltage supply and work with at least 50W. Considering the power consumption calculated above, the Li-ion battery pack is selected with a battery capacity 2200mAh, a maximum continuous current of 6A, and a nominal voltage of 12V.

Requirement	Verification
1. The energy module supplies 12.0 $\pm$ 1.0 V for DC motor driving chip. 2. The energy module supplies 5V $\pm$ 0.5V for Raspberry Pi. 3. The energy module supplies 5V $\pm$ 0.5V for STM32 Chip. 4. The energy module work for at least 20 minutes in full power operation mode.	1. Connect chips to the Energy Module, all chips should work properly. Measure the voltage output of the energy module using a multimeter or oscilloscope and ensure that the voltage supply satisfy the requirements. 2. Run the car subsystem in full power mode, and measure the time operates using a timer. Ensure that the Energy Module work for at least 20 minutes without any issues such as overheating or voltage fluctuations. 3. Record the measured values in a table and compare them with the expected value.

Table 4: R&V Table for Energy Module

2.2.5 Astra Pro Camera

In our design, the camera will be placed on a base of the car. The images captured by the camera will be sent to a remote server via Raspberry Pi and a communication module for further processing. The power is supplied by the Raspberry Pi with a USB line and it sends RGBD signal to the Raspberry Pi through a USB line.

We control the car to rotate if needed and the rotating speed should be less than 3 rads/s, which can be controlled by the control subsystem. The camera could provide live information of color and depth with precision in millimeters. Detailed requirement and test procedure of this part is mentioned in the remote server section.

2.3 Communication Subsystem

We need a communication subsystem to realize remote control and sensor signal transmission. We will use Raspberry Pi as the core of the communication subsystem on the car side to provide communication with the remote server.

1. RGBD raw data from the camera will be received by Raspberry Pi through USB, and then Raspberry Pi will preprocess the images to get the depth information. Then, both the RGB image and depth image will be sent to the remote server via Wi-Fi;
2. The car motion commands sent from the remote server will be delivered to Raspberry Pi, and Raspberry Pi will parse the data received and send the parsed command to the Control Module through UART.

To provide real-time data for reconstruction, Raspberry Pi must have the capacity to handle raw data from the RGBD camera in a real-time. It should send the RGB images and depth images to the remote server at 5-10 FPS or above. The image resolution is expected to be above 320x240, and optimally 640x480. Also, it should receive the control signal at a frequency of 20-50 Hz or above, so that the system can work smoothly and won't crash due to high latency.

Requirement	**Verification**
1. RGB and depth images with a resolution of at least 240x320, optimally 480x640, will be send from the Raspberry Pi to the remote server at a speed of 5-10 FPS. 2. Control commands can be delivered to Raspberry Pi from the remote server 3. Transmission information is end-to-end encrypted.	1. Perform throughput test to make sure we have at least 5 MB/s bandwidth. 2. Perform image transmission test to verify that the transmission rate is satisfied. 3. Perform command transmission test to verify that commands can be sent correctly. 4. Check whether the sent and received data are encrypted. Record the average time delay between send and receive. Record the average error.

Table 5: R&V Table for Communication Subsystem

2.4 Remote Server

The remote server can perform 3D reconstruction and provide an interface for the user to gain knowledge about the surroundings and remotely control the car with a joystick. We have subdivided the remote server into the following parts.

2.4.1 Joystick Module

Input: Joystick raw signals
Output: Commands for Control Module

A **Joystick Module** is designed to phrase the user's joystick operation to the commands for car movement. The Joystick is connected wireless or wired to the remote server host. The user's actions, such as pushing the joystick and pressing the trigger, are monitored at 100Hz and then interpreted into the Control Module's specified command format in a Python script.

Requirement	Verification
1. Joystick Module can monitor and read the joystick signal correctly. 2. Joystick Module can phrase the joystick signal to the control command for Control Module.	1. Connect the joystick to the remote server and run the script, the correct joystick signal value should be shown. 2. The user's control action should be phrased correctly and generated in a predefined format.

Table 6: R&V Table for Joystick Module

2.4.2 3D Reconstruction Module

Input: RGBD camera images
Output: Reconstructed 3D model

A **3D reconstruction module** processes RGBD camera images with OpenCV and performs 3D reconstruction using algorithms such as SLAM (simultaneous localization and mapping) or structure from motion. As the user gradually moves the car around the environment, the 3D reconstruction module should be working simultaneously to keep building up the surrounding environment's model. We would use a Ubuntu system on the server to carry the ROS together with the RTAB-Map (Real-Time Appearance-Based Mapping) algorithm.

The whole process would be real-time which means the calculation time for each frame should be short (a small delay) and the reconstructed model should be smooth and continuous, namely even if the photos have gap when we try to combine the continuous ones, the model should be able to fix the gap and make a consistent output. Detailed requirements and verification would be illustrated in the chart.

Requirement	Verification
1. Reconstruct the frame with 10fps and the output will be not less than 320*240 resolution. 2. The whole reconstruction should be done within 10s after scanning and the processing delay should be less than 0.5s.	1. The R-tab-map should be able to receive the RGBD signal from the camera and launch the reconstruction process on the remote server properly. 2. The final demo will be a live presentation with the output on the user interface. And the resolution details should show on the screen.

Table 7: R&V Table for 3D Reconstruction Module

2.4.3 Visualization Module

A **Visualization Module** is a tool that displays the output model of the reconstruction algorithm for the user to check. The RTAB-Map software will naturally have image output of the result of the algorithm, but we want to build a visualization based on that which enables the user to adjust the viewpoint easily. It might be hard to realize this if it's still during the construction phase, so we may want this functionality to be active only after a whole module is constructed.

Requirement	Verification
1. The User should be able to see clear icons for mode switching. 2. The Interface should provide no less than 10 fps and 320*240 resolution.	1. Running on Ubuntu system, 2. Use the Obbrec Viewer for the user to switch the image type and provide depth information, 3. The final demo will be a live presentation where the user can use watch the simultaneous reconstruction model and the images on the remote server. The setting parameters should show on the screen.

Table 8: R&V Table for Visualization Module

2.5 Points Summary

Based on the requirements of each subsystem and module along with the importance and difficulty considerations for the project, we give the following points summary

Subsystem	Modules	High Level Requirement	Point
Car Subsystem	Car Platform	The car platform is light, with the strong structural integrity of a 3-4 kg loading. No equipment loose or drop during long working periods.	2.5
	Motor and Wheels	The motor and wheels can provide the appropriate movement speed and steering speed, such as in the straight line can reach 0.5-1m / s, in 2s to complete 180 degrees of rotation.	2.5
	Control Module	The Control Module is capable of handling input signals from Joystick Module, and outputting the PWM signals required by the motor driver chip to enable the robot car to move in a straight and turning direction.	4
		Car can turn at a speed of about 25 degrees/second according to the command	3
		Car can move omnidirectionally and turn simultaneously based on the commands	3
	Energy Module	The energy module can supply power to other modules. The voltage and current of the power supply module remain stable.	2.5
Communication Subsystem	Server side	RGBD data should be received at least every 0.2s.	2
		Control command should be sent correctly.	2
	Car side	This module should send RGBD data to the server at least every 0.2s.	2.5
		Control command should be received correctly.	2
	Security	Transmission information should be end-to-end encrypted.	4
Remote Server Subsystem	Joystick Module	The joystick module can correctly monitor and parse user input commands. At the same time the joystick module has to output the required format of input commands according to the control module. The reaction time should be less than 0.5s.	5
	3D Reconstruction Module	This subsystem should successfully reconstruct the surrounding environment captured by the camera within a relatively short period(i.e 1s)	4
		The calculation time for each frame should be no more than 0.2 s.	3
		Reconsturct the frame with 10fps and the output will be not less than 320*240 resolution	3
	Visualization Module	This module should successfully output the live scene of the camera with a processing 3d reconstruction map. As for the scanned part, the shape and the corresponding environment should be clear.	3
		The output should be in relative high resolution ratio so that the user can read the map without trouble.	2

Table 9: Table for Points Summary

2.6 Tolerance Analysis

There are two main topics related to the remote robot car control system. The first topic is the time delay, which can cause deviations in the car's motion, positioning problem, and even collision problem. The second topic is the RGBD image transmission, which is important to enable the reconstruction system to receive real-time RGBD image input.

2.6.1 Time Delay

The time delays in processing and communication of robot car control can cause deviations in robot car's motion, which can cause positioning problem and even collision problem.

The source of the time delay comes from the remote control pipeline. The joystick module processes and analyzes the joystick signal to generate motion commands. The commands are encoded and transmitted by the communication module via Wi-Fi with TCP protocol. The control module calculates the PWM of each motor based on the decoded motion commands. Although this pipeline is complex and it is difficult and tedious to analyze the delay in detail, we can give a maximum allowable delay range because the remote control itself requires low delay and high accuracy. We expect the remote control of the car has a maximum delay $\Delta_{max}t = 100ms$, otherwise the remote control system can crash due to the high latency.

Under the assumption of worst-case latency, and given the designed maximum speed of the car is $0.5 - 1.0$ m/s, the position drift of the car is limited to $10 - 20$ cm.

$$10 \text{ cm} < \Delta_{\max}x = 2 \times v_{\max} \times \Delta_{\max}t < 20 \text{ cm} \tag{1}$$

The ideal latency $\Delta_{ideal}t$ for a good control experience is within 60 ms, in this case, the drift is improved to 6-12 cm. According to the RGBD camera operation guide, the appropriate distance between the depth camera and the objects is in the range of 0.8-8.0 m. Consider two working cases, the first is that the distance between the car and the object is within the working range of the depth camera, and the second is that the distance between the car and the object is less than the minimum working distance.

- In the first case, the depth camera works properly, so the car's surroundings are being modeled in real-time. The proof is as follows: our camera takes RGBD images at 30 fps, and assuming that the car moves at a maximum speed of 1m/s relative to the obstacle, the maximum displacement of the car between two successive images is 3.3cm. 3.3cm is much less than 0.8m and can be neglected. Therefore, we can proof that the surrounding environment can be reconstructed at a minimum working distance of 0.8m. Then, the 3D reconstructed model can in turn assist in positioning and correct the drift.
- In the second case, since the depth camera will likely fail to work properly, we must consider early warnings on the visualization page, such as prompting for obstacle avoidance. Besides, collision prevention like forcing a speed reduction for dangerous situations is also necessary in a real-world application. When the cart is monitored to be less than 1m from the nearest obstacle, the maximum speed should be limited to be less than 20 cm/s. Meanwhile, the RGB image might itself provide better visual information about the surroundings, especially for close obstacles. So, switching the view to RGB images may also be a desirable measure.

Based on the analysis of the positioning error caused by the delay, we can flexibly improve our system design. The first is to reduce the latency. Communication latency is the main source of latency, if the direct use of public network cannot meet the requirements of low latency, we can choose to use LAN hotspot, or use 5G communication to reduce the latency.

To avoid possible collision hazards caused by delay, we can warn on the visualization page when it is close to the obstacle by 1m, and force to reduce the speed of movement when the distance is less than 0.8m. We can even take over the control when the distance is too close, such as less than 20cm, to avoid a possible collision, if necessary.

2.6.2 RGBD Image Transmission

To enable the reconstruction system to receive real-time RGBD image input, it is important that we can control the RGBD image transmission rate to a suitable range, optimally between 10-24 FPS. The transmission rate depends on image size and bandwidth of the WiFi connection.

According to our preliminary measurement, the bandwidth for our school WiFi eduroam between Raspberry Pi and the remote server is roughly 7MB/s. The image output of the camera is maximum 480x640, and each pixel in RGB image takes 3 Bytes, while each pixel in depth image takes 1 Byte. As a result, it takes

$$480 \times 640 \text{ pixels } \times 4 \text{ Bytes } \approx 1.17 \text{ MB} \tag{2}$$

to store one RGBD image. If we adopt JPEG compression with highest quality, the image size can be reduced to half of the original size, which means that now one RGBD image will be 0.6MB. If we transmit it using our school WiFi, the estimated transmission rate is 11.6 images/s.

Notice that the estimation here is based on the assumption that we use the highest quality compression setting and use the school WiFi which we know that it is not very fast. There is a lot of room for us to improve the transmission rate, such as decreasing the compression quality, or making the Raspberry Pi itself to be the wireless access point so that we can have stronger and more stable WiFi signal and faster WiFi speed. Also, we can also downsample the image resolution by a factor of 2 and it will easily boost the transmission rate to 46.4 images/s. There is a trade-off between the tramsmission rate and the image quality, which will in turn affect the reconsutrction quality. We will find a optimal trade-off for our system and use case during our system development.

3 Cost and Schedule

3.1 Cost Analysis

- Labor for each partner: $37(\$/hour) * 10(hours/week) * 7(weeks) = \$5180$
- Parts: detailed components and their verification, part number and cost in table 10.
- Sum of costs into a grand total $5180 * 4 + 12.5 + 54.14 + 75.43 + 61.29 + 150 + 50 = \$21,123.36$

Description	Verification	Part #	Cost
Car with motors and wheels	Yahboom	BC547B	$61.29
12.6V battery pack with charger	Yahboom	18650-3S	$12.50
Onboard microcontroller	Yahboom	STM32F103RCT6	$54.14
Single Chip Microcomputer	WAVE ELECTRONICS	Raspberry Pi 4 Model B 2GB	$75.43
RGBD camera with expansion bracket	ORBBEC	Astra Pro	$150.00
Joysticks	Microsoft	XBox	$50.00

Table 10: Parts List

3.2 Schedule

- **Week 2: (3.20-3.26)**
  1. Together: Design the general structure of the project; Finish the design document
  2. Hao Chen: Install the 3D-Reconstruction SDK; Run the 3D-Reconstruction with camera directly connected to the computer
  3. Han Yang: Assemble the car components; Make the car run
  4. Junyan Li: Realize the communication between Raspberry Pi and computer
  5. Yuhao Ge: Buy necessary parts; Design general schedule
- **Week 3: (3.27-4.2)**
  1. Hao Chen Yuhao Ge: Realize 3D-Reconstruction with still camera; Design API
  2. Han Yang: Realize accurate all-direction control with joystick connected by line
  3. Junyan Li: Realize the communication between Raspberry Pi and STM32
- **Week 4: (4.3-4.9)**
  1. Hao Chen: Realize smooth real-time 3D-Reconstruction when using hands to move the camera around
  2. Han Yang: Control the car by joystick remotely
  3. Junyan Li: Generalize the communication pipeline; Create API for other teammates to use
  4. Yuhao Ge: Build the data preprocess module on Raspberry Pi to downsample the image

- **Week 5: (4.10-4.16)**
    1. Hao Chen: Realize smooth real-time 3D-Reconstruction when using data sent from car
    2. Han Yang: Gather and return car's motion data (position, orientation), and the relative position in the scene
    3. Junyan Li: Analyze the reliability of the communication (error rate, bandwidth)
    4. Yuhao Ge: Design protocols to encode and decode down-sampled data to remotely feed the algorithm with data
- **Week 6: (4.17-4.23)**
    1. Together: Assemble all parts together to have a basic demo version
- **Week 7: (4.24-4.30)**
    1. Together: Solve problems encountered during assembling
- **Week 8: (5.1-5.7)**
    1. Hao Chen: Improve the performance of the algorithm
    2. Junyan Li: Improve privacy and information security
    3. Yuhao Ge Han Yang: Design the perspective shifting functionality
- **Week 9: (5.8-5.14)**
    1. Together: Prepare the mock demo; Prepare the final report draft
- **Week 10: (5.15-5.21)**
    1. Together: Clean up the code; Prepare the final report

4 Ethics and Safety

4.1 Ethics

The ethical concerns for our RGBD reconstruction system include privacy and information security. The use of the RGBD camera for 3D reconstruction raises concerns about the privacy of individuals, as the camera can capture images and other data that can be used to identify individuals. We need to ensure that the people captured will be anonymized properly.

Additionally, information security concerns arise with the storage and transmission of data collected by the camera. If the data transmission and storage is insecure, hackers can get these sensitive data and sell them to evil institutions, bringing us and probably other people huge trouble.

The IEEE and ACM Code of Ethics both emphasize the importance of respecting the privacy of individuals and protecting their personal data. According to the IEEE Code of Ethics, engineers must "protect the privacy of others", and they must only use the information for legitimate purposes. Similarly, the ACM Code of Ethics, Section 1.6 Respect privacy, requires that computing

professionals respect the privacy of others and protect the confidentiality of any data they access.

To avoid ethical breaches, we will implement appropriate security measures to ensure that the data collected by the RGBD camera is stored and transmitted securely, including using an encrypted data transmission protocol, deleting any intermediate and temporary data when applicable, and storing the reconstruction result under an encrypted folder. We will also seek informed consent from individuals whose data is being collected and ensure that their privacy is protected throughout the project.

4.2 Safety

Safety is the highest priority in our project, and we have recognized several potential safety issues, including the risk of hurting other people and damaging things if our car is out of control. If the car is not adequately controlled, it may lead to accidents or collisions. Also, the battery or the power system is also a very important part since it may cause short-circuit if the battery is not functioning properly and even an explosion could happen.

To mitigate safety concerns, we will follow government regulations, industry standards, and campus policies related to the use of remote control cars and unmanned vehicles. We will also ensure that appropriate safety measures are implemented, such as limiting the car's speed and operating it only in controlled environments. The power system will be checked frequently to ensure that it is in a normal state and it can function properly before we actually mount it onto our car. We will also implement fail-safes to prevent any potential accidents and collisions.

5 Reference

References

[1] M Aharchi and M Ait Kbir. A review on 3d reconstruction techniques from 2d images. In *Innovations in Smart Cities Applications Edition 3: The Proceedings of the 4th International Conference on Smart City Applications 4*, pages 510–522. Springer, 2020.

[2] Chen Liu, Jiaye Wu, and Yasutaka Furukawa. Floornet: A unified framework for floorplan reconstruction from 3d scans. In *Proceedings of the European conference on computer vision (ECCV)*, pages 201–217, 2018.

[3] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4460–4470, 2019.