Exploration of Virtual Reality Control for the Stanford Pupper Quadruped Robot

Ankush Dhawan
Stanford University
450 Serra Mall, Stanford, CA 94305
ankushd@stanford.edu

Raghav Samavedam
Stanford University
450 Serra Mall, Stanford, CA 94305
raghavsa@stanford.edu

Abstract

In this project and report, a virtual reality (VR) control system was developed to control the movement of the Stanford Pupper Robot. By interfacing with a series of hardware components, including a VR headset, the Pupper robot, a laptop, and an Inertial Measurement Unit (IMU), the orientation of the user's head was mapped to a series of commands for the Pupper robot's movement. This work extends control of the Pupper from joystick control to a VR setting where a user can remotely control the robot using just their head movements while achieving a first person point of view. This system has potential applications in area such as natural disaster response, general surveillance, and as robotics education. The code for this project is available on GitHub with a ReadMe.md file that details setup: `https://github.com/ankushDhawan5812/VR_Pupper_Control.git`.

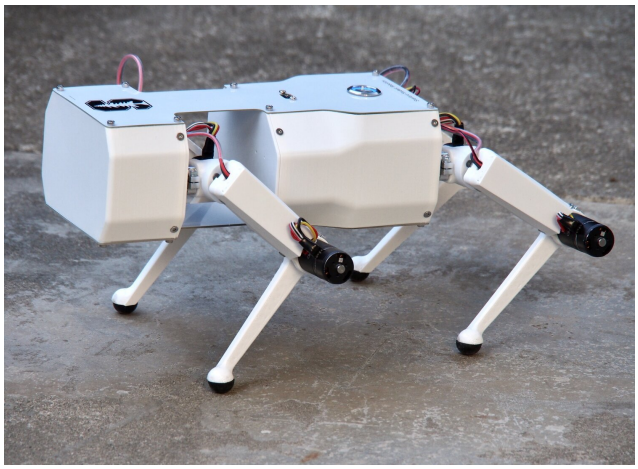

Figure 1. Stanford Pupper Robot [6]

1. Introduction

1.1. Context

The Pupper robot is a robot designed by the students in the Stanford Robotics Club [7]. The Pupper robot is a quadruped robot that mimics the gait mechanics and movement of a dog. Current research into quadruped robots involves using the robot to study gait mechanics, such as researching dynamic walking control, autonomous control, and simulation to real environment dynamics exploration among many other topics [14]. The need for VR robotic simulations is great for testing and training models to autonomously control robots through trained algorithms [11]. In addition, remote VR control of the Pupper has potential applications to natural disaster response, surveillance, and education.

1.2. Scope

In this project, an VR control system environment will be designed for the Stanford Pupper. This study will work to extend previous joystick control to VR control, where VR headset orientation is mapped to the Pupper orientation. This then routes to move the Pupper robot while giving the user a first person point of view of the robot through the VR headset. Hence, the applications and use cases of quadruped robots will not be addressed in this project and the VR control system will be developed and addressed.

In the Fall of 2021, the authors of this paper took the Stanford class, CS199: Pupper Independent Study. In this class, the authors built the Pupper robot, and learned the principles of how the robot works. While this project builds off of work done in this class and uses this robot, the build for the robot will not be discussed in this project since it is out of scope.

1.3. Significance

Quadruped walking robots are of special interest because of their unique applications over other types of robots such as flying drones or wheeled robots. One can envision using a quadruped robot especially in the case where obstacles are present [13]. Compared to wheeled robots, wheeled robots have a difficult time moving or rolling over obstacles because of small wheel sizes. While drones are able to fly over many obstacles, they suffer from the issue that they have battery/charge issues, where the weight of carrying a larger battery greatly hampers the flying ability of the drone. When obstacles are present, walking quadruped robots are able to step over obstacles while still being able to carry the load of larger batteries without a large hindrance in performance [10]. For this reason, quadruped robots can be potentially very useful in applications where rough surfaces and obstacles are present, such as building surveillance/planning, home robots, and disaster first response among a variety of other applications [3].

Developing a virtual simulation environment for the Pupper robot is important because it can allow for remote control of the robot in a first-person setting. Currently, Pupper simulation is in a 2D computer screen format [7]. Adding a first person view using virtual reality can be an interesting and useful addition to the simulation of this quadruped robot. Especially when the Pupper robot is to be applied to the real-world, this VR environment would be useful since a user can see the world around the Pupper remotely as if it were there itself. Virtual environments to control/simulate robots have been effectively used to simulate underwater robots in previous research [5].

In addition, the Stanford Pupper is currently being used for educational purposes [7], and having an virtual reality environment that is integrated with the Pupper quadruped robot can give students an entertaining platform to experiment with the robot, while learning the principles behind both virtual reality and robotics. Because of these applications, it is important and useful for one to remotely control the Pupper from a VR setting.

1.4. Goal

The goal of this project is to develop an augmented environment for the Stanford Pupper Quadruped Robot. This first-person view of the robot can give users and resarchers a better understanding of what the Pupper robot "sees", and is potentially very useful for the applications of quadruped robots.

1.5. Related Work

1.5.1 Simulations

Currently, a simulator using the Pybullet physics engine has been developed to simulate the Pupper environment on a 2D computer screen [7]. This simulator is commonly used for testing and training algorithms. Extending this work to a virtual reality setting can help make simulation of the robot more realistic, and be interesting and useful for its application settings. This is very useful to simulate the behavior of the robot in a virtual environment, but lacks a full virtual experience that may be given through a virtual reality environment. There are many For example, if a Pupper robot is being used for building surveillance, a user can remotely control the Pupper robot using a virtual reality headset and see around the Pupper's environment as if the person was there instead.

1.5.2 Karel Pupper API

To control the Pupper, a framework, called karelPupper, was developed in the Fall of 2021, drawing inspiration from the classic Karel programs from the Stanford class CS106A. The goal of this framework is to develop a simple coding API that allows the programmer to control the movement of the Pupper, much like the Karel programs from CS106A. Such available commands include wakeup(), forward(), turnI(), and turn _for_time() among others. Doing so makes simple movements of the Pupper simple to program, so that one without deep knowledge of motor movement and kinematics can program the Pupper to perform given movements. This project interfaces with many of the commands available in the karelPupper API to make this VR control system. The code and setup instructions for the karelPupper API are detailed in this GitHub repo: `https://github.com/stanfordroboticsclub/karel-pupper-api.git`.

1.5.3 Applications of Quadruped Robots

Many companies/organizations are currently researching into quadruped robots for a variety of applications as mentioned earlier. While no current quadruped robot is being used in the field on a large scale or being sold on a large scale, building an VR environment for the Pupper extends its capabilities to one day being massively used, whether that be as a robotics education tool, as a surveillance robot, or for natural disaster response. Hence, further work is needed in the application area, but as mentioned in the scope section, will not be tackled in this project due to scope.

1.6. Outline

The remainder of this paper will address the exploration into an AR environment for the Stanford Pupper. The method of exploration will be explained and justified, and then further testing and design choices will be elaborated on. This report will end with a discussion about conclusions and future extensions of this work.

2. Hardware

There are three main pieces of hardware necessary for this project.

2.1. Stanford Pupper Robot

The Pupper robot is the first piece of hardware necessary. The Pupper robot has four legs, two hind and two rear legs. Each leg has three motors, corresponding to the hip, knee, and foot of the robot, combining for twelve total motors. The Pupper has an onboard Raspberry Pi 4, a Teensy 3.2 microcontroller, and a Pi Camera. The Raspberry Pi acts as the memory system for the robot, and handles the data from the Teensy microcontroller and the Pi Camera. The Teensy Microcontroller is responsible for driving the motor movements. The Pi Camera records the front of the Pupper, and streams this video to a website.

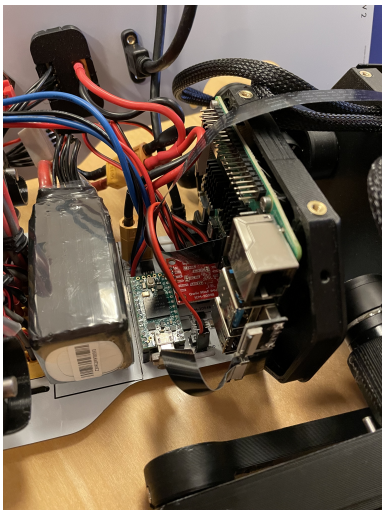

Figure 2. Raspberry Pi and Teensy Microcontroller Location on the Pupper Robot [9]

The build of this robot is detailed on the Stanford Robotics Club website [9].

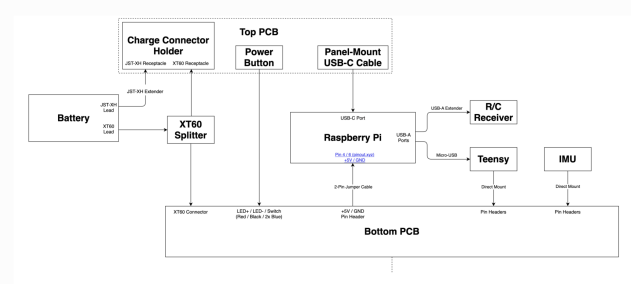

Figure 3. Wiring Diagram for the Pupper Robot [9]

2.2. View-Master VR Headset

The View-Master VR Headset is necessary to display VR images. The specs for this headset are available on the ProductZ website [12], which match those of the Google Cardboard [4]. The View-Master will display a VR-renderable image by connecting to a laptop that connects to a stream from the Pupper [9].

Figure 4. View-Master VR Headset [1]

2.3. VRduino

The VRduino, a device built specially for EE267, was used for its capabilities to send IMU data. The VRduino device is built from a Teensy 3.2 Microcontroller, an InvenSense 9250 IMU, and four photodiodes. The VRduino is then able to mount on the front face of the View-Master VR Headset by using a velcro strap.

Figure 5. VRduino Device [15]

For this project, only the IMU data was necessary, which

gets communicated to the Raspberry Pi on the Pupper through the Teensy microcontroller in the VRduino.

3. Set-Up

The set-up for this project is simple. First, the View-Master VR Headset is connected to a WiFi-enabled Laptop, which is able to display a website to the View-Master's LCD screen. The laptop also connects to the Raspberry Pi onboard the Pupper through secure shell (ssh) to deploy commands and edit code. Next, the VRduino is connected to an open USB port on the Raspberry Pi of the Stanford Pupper. This set-up allows the Raspberry Pi to interpret IMU data from the VRduino and send information to control the Pupper to the Teensy Microcontroller on the Pupper. Configuration for the Stanford Pupper's Raspberry Pi and Teensy Microcontroller can be found on the Pupper Docs website [8]. The Pupper's Pi Camera is then enabled to stream to a website, which is then viewed on the View Master Headset.

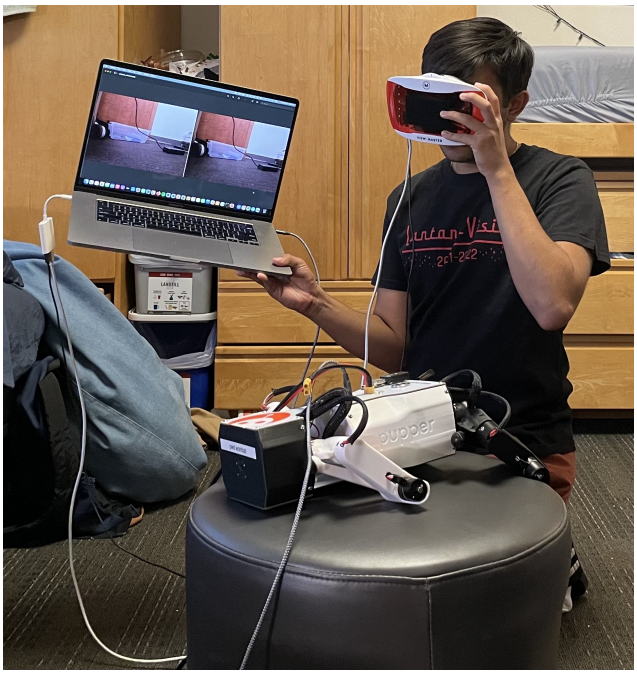

Figure 6. Full Setup to render VR Image [2]

4. Methodology

The methodology for completing this project follows a series of four milestones. The first three milestones, including streaming the Pupper's Pi Camera to a website, sending IMU data from the VRduino to the Pupper's Raspberry Pi, and controlling the Pupper with karelPupper commands, work together for the fourth milestone. The fourth milestone is to interface everything together and using the VR headset orientation to control the Pupper with karelPupper commands.

4.1. Milestone 1

The first milestone for this project is to stream a VR Renderable image from the Pupper to a website. This procedure is contained in a python script called NAME SCRIPT. This script essentially takes the streaming output from the Pi Camera, connected to the Raspberry Pi's Camera port using a ribbon cable, and sends that output to a website corresponding to the IP address of the Raspberry Pi using socket servers. The HTML code for the website essentially displays two of the same images side by side. When viewed in the View-Master VR Headset, this gives a first person point of view for the user. The image size is set to fully fill the View-Master's LCD screen. To do this, the laptop and the Raspberry Pi must be connected to the same WiFi network.

Figure 7. Website Image of Pupper Camera Stream [2]

4.2. Milestone 2

The second milestone for this project is to configure the VRduino to send the proper IMU data to the Pupper's Raspberry Pi. In Homeworks 5 and 6 of EE267, the IMU data was interpreted as the world space quaternion orientation of the VRduino. This same framework was used to give the orientation of the VRduino in this project as well. To do so, the code from EE267 Homework 5 was modified to send only the necessary quaternion coordinates to the Pupper's Raspberry Pi via an open USB port. Since the user of the View-Master VR Headset is staying stationary, only the third coordinate was necessary to understand the rotation of the headset. These coordinates were mapped using threshold values, which then allowed for control of the Pupper. The following pseudo-code defines this control scheme.

Algorithm 1 Pupper Control Scheme

```
if third_coordinate > positive_threshold then
    turn_right()
else if third_coordinate < negative_threshold then
    turn_left()
else
    move_forward()
end if
```

Figure 8. VRduino Mounted on View-Master VR Headset [2]

4.3. Milestone 3

The third milestone for this project was to set up the Pupper to respond to karelPupper commands. To do so, firmware for the Teensy Microcontroller on the Pupper had to be re-flashed to properly control the Pupper. This firmware is available on the Pupper Code GitHub repo: `https://github.com/Nate711/DJIPupperTests.git`. Using a combination of commands from this repo, control scripts for the Pupper were developed to test how the movement of the Pupper, using commands such as turnI(), forward(), and wakeup() among others. Videos of these commands are availble on the karelPupper GitHub repo in the README.txt file.

4.4. Milestone 4

The fourth milestone for this project was to integrate all the individual components. This meant being able send VRduino IMU data to the Pupper's Raspberry Pi, and then rendering an image from the Pi Camera onto a website, and viewing this from a remote laptop. After this was completed, the Pupper's movement was able to be controlled using the orientation of the View-Master VR Headset.

5. Results

Through implementing these milestones, the user of the View-Master VR Headset can control the Pupper through the Headset Orientation. Using such a system allows the user to remotely control the Pupper, which is important for applications such as natural disaster response and surveillance. One can imagine using such a control system to surveill an area with the Pupper.

5.1. Analysis

Currently, the Pupper control system acts somewhat like a joystick, where certain orientations of the headset lead to specific controls. This means that the controls are not very smooth between different positions. The Pupper is also fully wired, and only gives a front view of the robot. Solutions to this problem will be analyzed and discussed in the future work section.

6. Discussion

Over the course of this project, many issues, especially regarding hardware complications, were encountered.

6.1. WiFi Complications

First, connecting the Raspberry Pi over Stanford WiFi did not meet the needs for this project, as any firewalls and network constraints prevented SSH capabilities and streaming features. Hence, to overcome this, an unrestricted network was used for testing and development.

6.2. Stanford Pupper Robot Outdated Firmware

The robot used for this project was build in a Stanford Fall Quarter class, CS199. The Pupper robot is under constant development from members of the Stanford Robotics Club, and so many changes were made to the Pupper code base and microcontroller firmware. This was noticed when testing with the robot saw many miscommunications between the robot's motors and the code commands. To overcome this, the fireware on the Teensy microcontroller had to be re-flashed to update the microcontroller to its latest version.

6.3. Reading IMU Data on the Raspberry Pi

To determine which command to call on the robot, the orientation of the VRduino (from its IMU) was sent to the Raspberry Pi. The VRduino has a Teensy microcontroller embedded in it, which typically interfaces with the Arduino IDE and outputs information to the Arduino's Serial Monitor. However, the Raspberry Pi does not have a Serial Monitor as the Arduino IDE does. To work around this, the library, PySerial, was used, which essentially treats the Terminal on the Raspberry Pi as a Serial Monitor. Doing so allowed the Raspberry Pi to interpret data from the Teensy Microcontroller.

7. Future Work

There are many pieces of future work for this project, which are highlighted in detail with applications below.

7.1. Wireless Connections

The most obvious relevant next step is to make the current system completely wireless, so that the Pupper can operate and send data via Bluetooth. This was not implemented in this project due to logistical reasons regarding scope and time constraints. This can be accomplished by

using the onboard battery rather than the wall power supply to power the Pupper, and use WiFi to SSH to the Pupper's Raspberry Pi instead of wired USB-C connection. In addition, the code can be modified so that the VRduino connects to the laptop rather than the Pupper's Raspberry Pi. The VRduino can then send data to the Pupper by using the PySerial library and SSH. This will allow for true remote control of the robot, and allow the robot to be controlled over long distances not constrained by the length of wires.

7.2. Tuning Control

A second piece of future work is to further tune the Pupper's Control. Currently, the Pupper's control is based on thresholding, where the orientation of the View-Master VR Headset acts somewhat like a joystick controller. However, the Pupper is capable of taking in speed, distance, and angle parameters for many of its karelPupper commands. Hence, the orientation of the View-Master VR Headset can be tuned to edit these parameters to give smoother control of the Pupper. This can be accomplished by using a distribution rather than thresholds, where the outputs of the distribution will be the inputs to karelPupper commands. Doing so will allow for the Pupper to move smoother with the headset orientation.

7.3. Expanding on Camera Control

Another piece of future work would be to add a rotating camera. This can be accomplished by adding another motor on top of the Pupper and an arm. The camera can then spin around on top of the Pupper, giving the user a 360 degree point of view. This will be useful for surveillance tasks, where the user will require many view points to properly surveil an area. The user can control this using a joystick that connects to the laptop, and streams its data from the laptop to the Pupper's Raspberry Pi.

8. Relevant GitHub Links

The following links are important code packages found on GitHub that were utilized for this project.

- Teensy Microcontroller Motors firmware: `https://github.com/Nate711/DJIPupperTests.git`
- karelPupper API Code: `https://github.com/stanfordroboticsclub/karel-pupper-api`
- Full Code for VR Control System: `https://github.com/ankushDhawan5812/VR_Pupper_Control.git`

References

[1] View-master deluxe vr viewer. *Walmart*, 2022.

[2] A. Dhawan and R. Samavedam. Ee267 final video presentation. *Google Slides*, 2022.

[3] M. Fujita and H. Kitano. Development of an autonomous quadruped robot for robot entertainment. *Auton. Robots*, 5:7–18, 03 1998.

[4] Google. Cardboard manufacturer help. 2022.

[5] D. GraEanin, M. Matijasevic, and K. P. Valavanis. Virtual environment testbed for underwater robotics applications. *IEEE 1996 Winter Simulation Conference*, 1966.

[6] N. Kau. Robots - stanford student robotics. *Stanford Robotics Club*, 2022.

[7] N. Kau and S. Bowers. Stanford pupper: A low-cost agile quadruped robot for benchmarking and education. *arXiv preprint arXiv:2110.00736*, 2021.

[8] N. Kau and G. Levine. Lab 6 - pupper assembly. *readthedocs*, 2021.

[9] N. Kau and G. Levine. Stanford pupper. *readthedocs*, 2021.

[10] A. Kumar. Quadrupeds: The new buzz in robotics. 2020.

[11] R. Macredie, S. J. Taylor, X. Yu, and R. Keeble. Virtual reality and simulation: An overview. *IEEE 1996 Winter Simulation Conference*, 1966.

[12] ProductZ. Mattel view-master. 2022.

[13] M. Raibert, K. Blankespoor, G. Nelson, and R. Playter. Bigdog, the rough-terrain quadruped robot, 2008. 17th IFAC World Congress.

[14] J. Tan, T. Zhang, E. Coumans, A. Iscen, Y. Bai, D. Hafner, S. Bohez, and V. Vanhoucke. Sim-to-real: Learning agile locomotion for quadruped robots, 2018.

[15] G. Wetzstein. Stanford ee267: Lecture 9. *Stanford EE267*, 2022.