Drone Dome: Immersive Environments for Mobile Robots

BSRI 2019 Henry Chang, Jackson Spargur, Shuang Cai, Keith O'Hara

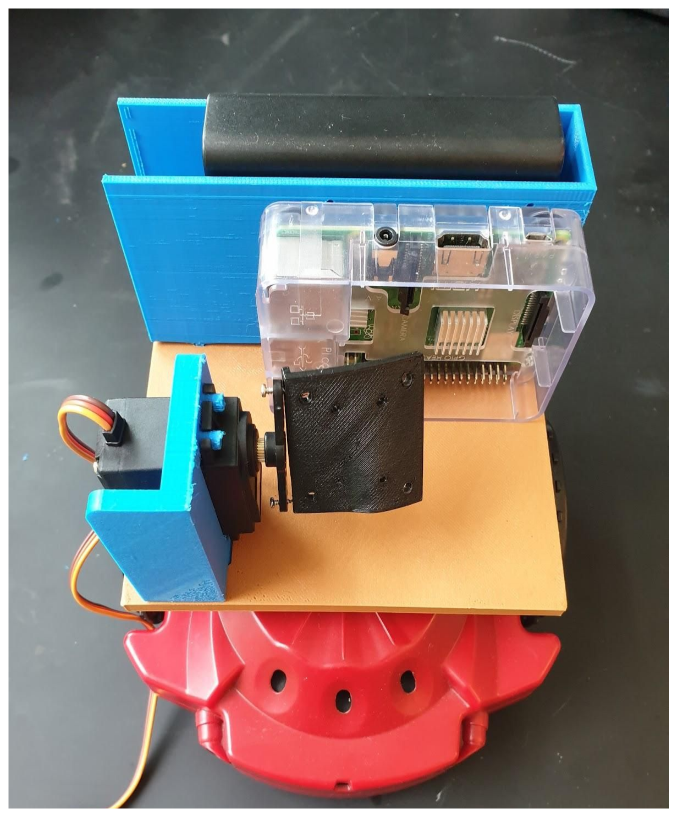
A hardware interface for combining Raspberry Pi, Battery, Camera and a Scribbler

Importance: Creates a remote sensing robotics interface so students can focus on robot configuration in class.

Headless Raspberry Pi

Goal: Control Raspberry Pi/robot without additional hardware setup.

Action: Used Linux and Raspberry Pi default commands to obtain the IP of a Raspberry Pi, save it into an SD card on bootup, and use a Linux terminal to remote control the Raspberry Pi with the obtained IP via SSH.


(The 3D Printed mount)

Hardware Mount

Goal: Create a platform for additional hardware (e.g. raspberry pi, portable battery, pi camera, stepper motor) installation onto Scribbler.

Action: Build on top of open-source CAD files, used fusion 360 to CAD design mount modules, and 3D printed them to create the platform.

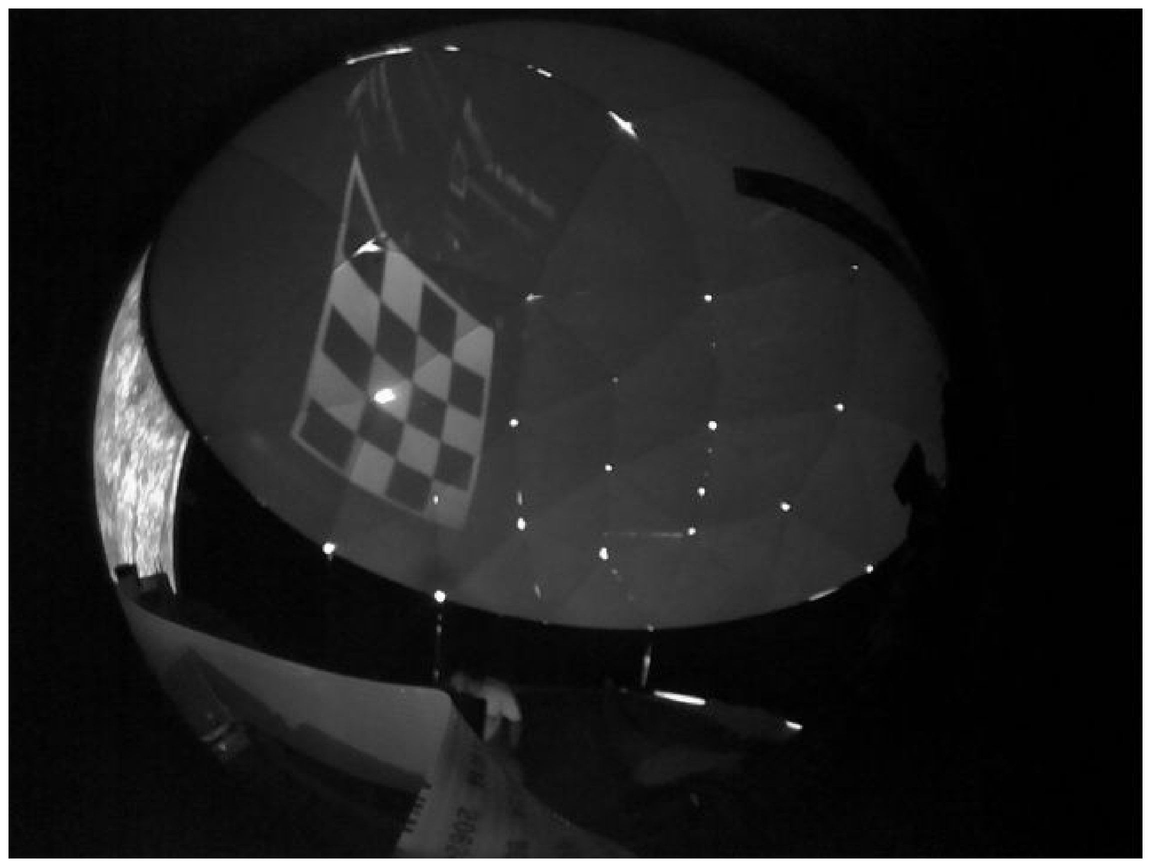
Software: OpenCV and Camera Calibration

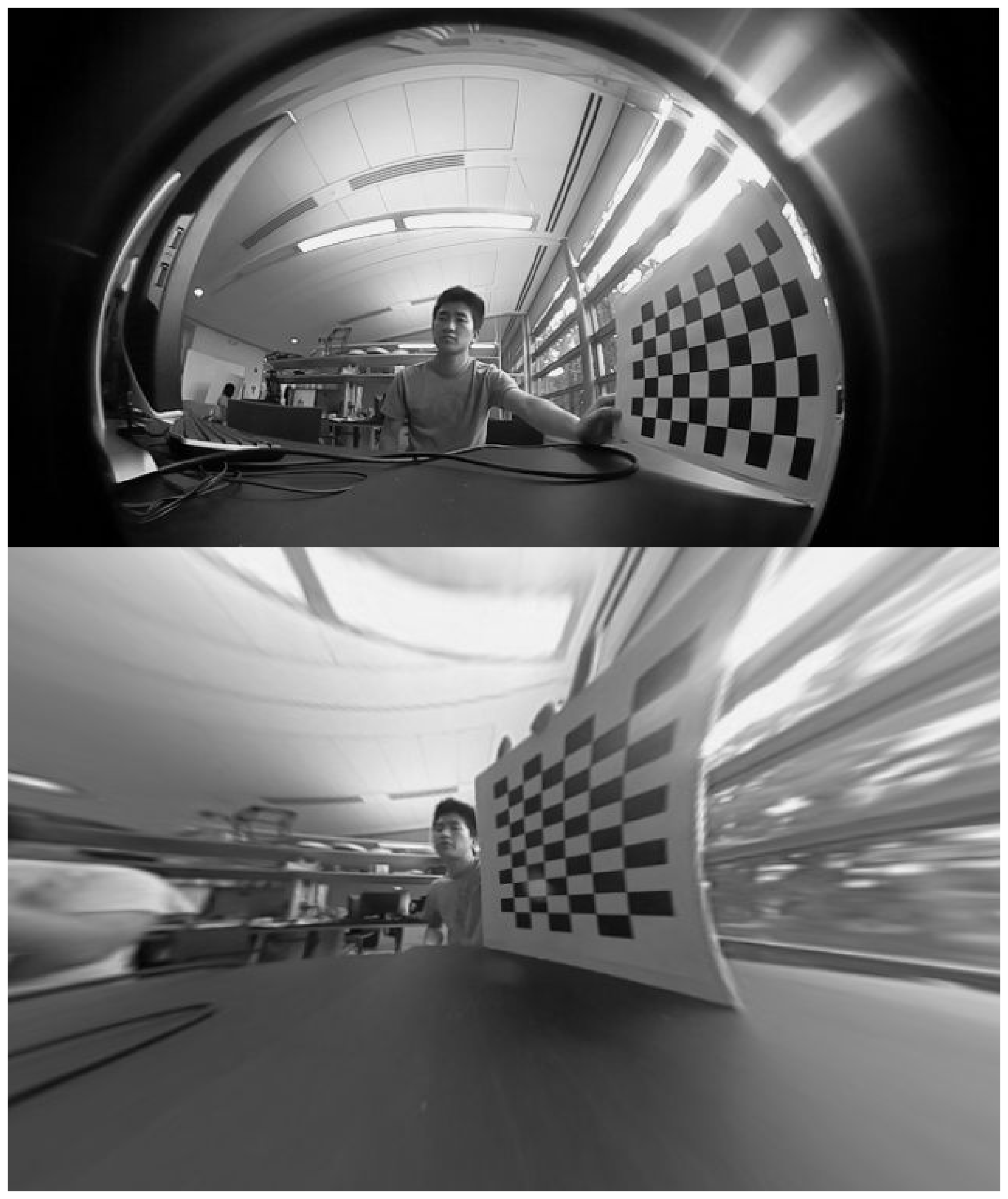
Goal: Utilizing OpenCV's machine learning package to generate a projector model, and based on the projector model, eventually a dome model.

Action:

- Forward-calibration: Took photos of a checkerboard on different tilted planes → Camera model (matrix) (the same algorithm for the projector matrix)
- Tested the camera matrix: apply matrix to the camera-taken images → checkerboards are straight.
- Tested projector matrix: plane-estimation by re-projecting circular patterns on the photos using the camera and projector matrices.

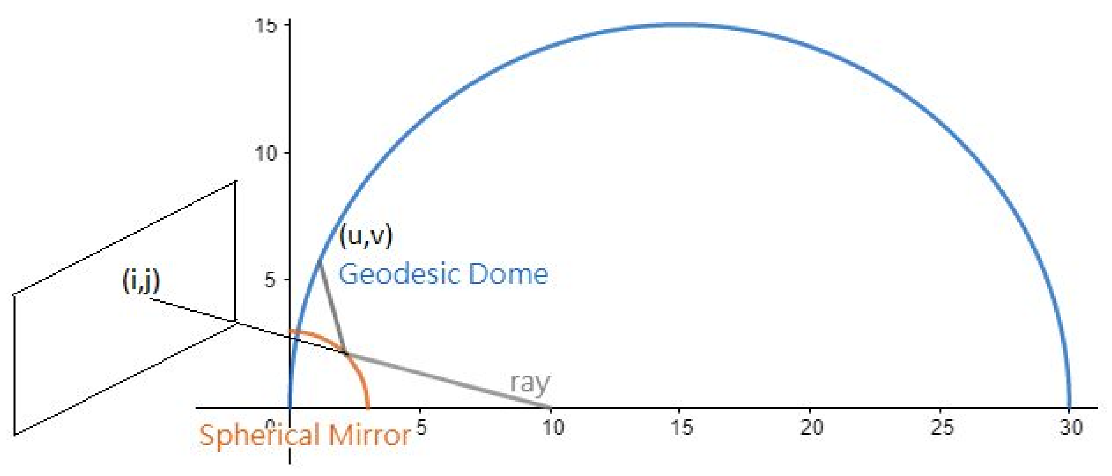
Difficulties: Installing OpenCV on Raspberry Pi. Getting the correct model for the dome.


(Dome with the calibration checkers)


(Undistort fisheye image using the camera matrix)

Dome projection with spherical mirror

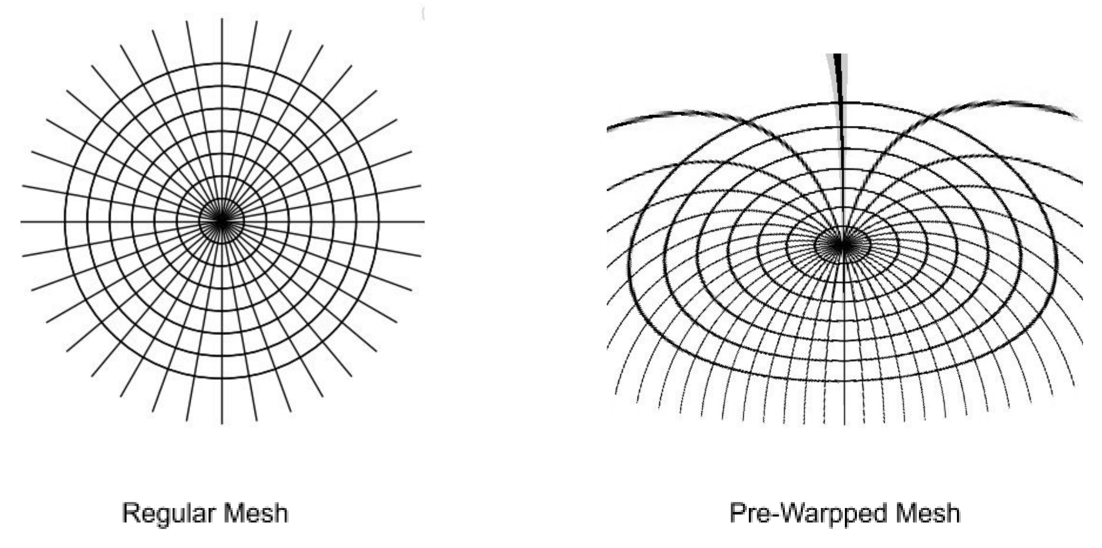
With a spherical mirror and a geodesic dome, create an immersive projection environment for spherical views such as night sky, space and the physical world.



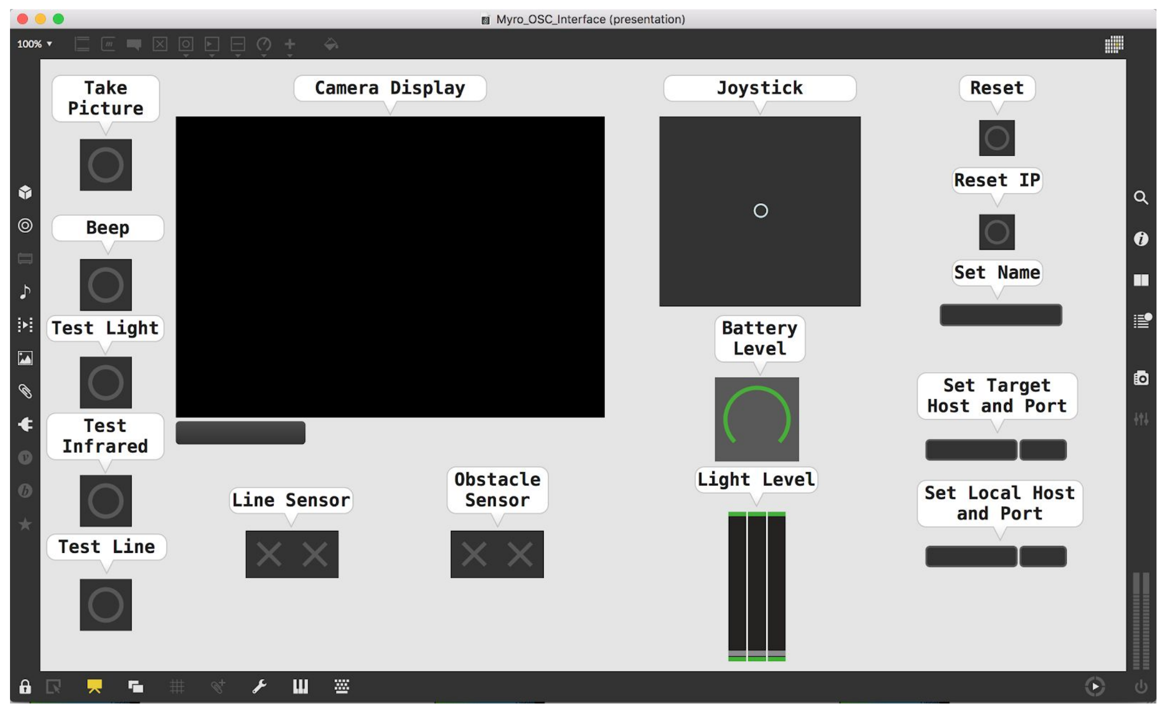
A simplified diagram of the projection evironment

Algorithmic Design (implemented in Python):

- Trace light rays with within the model of the projection environment to obtain $P^{-1}(u,v) = (i,j)$:
  - forward tracing + interpolation.
  - backward tracing (opposite direction of light).
- Obtain the Pre-wrapped image for dome projection $PrewarppedImg[P^{-1}(u,v)] = originalImg[P^{-1}(u,v)]$.



Further Research: incorporate laser pointer control of the robot in the dome to create a reactive environment.



(Scribbler interface)

Software: Scribbler Interface using Max/MSP, Python and OSC

Goal:

Create a simple interface to allow the student to control the Scribbler robot from any computer, including driving, camera operation, and all sensors.

Action:

Used Max/MSP to make the interface:

- joystick for motor control
- camera controls
- battery level indicator
- access to the light, infrared, and line sensors.

Difficulties:

We chose OSC because of its integration into Max/MSP, unlimited operating range (as long as both user and Pi are connected to the internet), and low latency. It presented us with the following difficulties:

- OSC messages are fairly small, so for instance to send an image required a separate OSC message for every 1,000 pixels or so.
- We found that the Pi could get easily saturated with OSC messages if it was being asked to do too many things at once, like sending a picture, updating a sensor display and moving the motors. Under these circumstances, performance would become extremely sluggish.

Working around these limitations was the main challenge of this element of the project.

(Left: Sparguar, O'Hara, Cai, Chang)