TV Remote Begone Project Proposal

Brent George (bdgeorge), Samuel Gonzalez (samgonza), and Tyson Lin (tysonlin)

Problem Statement & Idea:

TV Remotes are not the most natural interface, and also come with a lot of problems. From an interface perspective, there are a wide variety of controls, especially on older remotes. Part of the appeal of streaming sticks is the simplification from 100+ buttons to sometimes less than 10 buttons to navigate and control the television. Additionally, controlling a TV traditionally has required a remote in your hand. If you have lost the remote (a common occurrence), or your hands are preoccupied or dirty, you cannot control your TV.

Gestures have had their time in the spotlight as an interface over the past 20 years. They have fallen out of favor, but mainly because of the domains where they were applied. If you have a complex interface, gestures are difficult to use because there are only so many natural gestures you can use - swiping left or right, rotating, sticking up fingers, etc. We don't want to require users to learn a subset of sign language in order to send hundreds of commands.

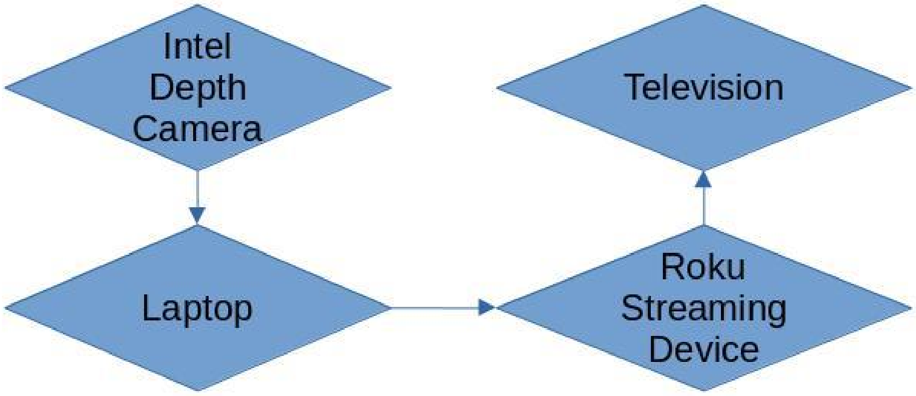
TVs only have a limited number of commands to start with, as referenced above. Combined with voice search, a list of commands such as "up", "down", "left", "right", "volumeUp", "volumeDown", "volumeMute", "power", "fast forward", and "play\pause" would capture nearly all of the functionality required for a streaming service. Hence, even limited gesture actions may be enough to effectively control a TV.

This is our project - we want to control our TV using gestures. So for example, swiping your hand can relate to directional commands, moving your hand in a circle may control volume, and other ideas for commands may arise. We want commands to be both intuitive and accurate, so that users rarely feel the urge to reach for a remote.

Our approach to the project is going to focus on the Human Computer Interface (HCI) aspect of gestures - intense user testing and feedback to find a truly appropriate set of gestures for a TV. We do not see any product offerings on the market offering this, so this can be a unique project.

System Block Diagram

Here's what we plan to use for sensors and actuators:



Sensing

Detecting User Gestures

We need to detect the user's gestures.

List all the sensors that would allow you to sense the user performance:

There are two main sensing options that we researched:

1. **XBox Kinect**: Constantly views the user. Use computer vision and machine learning to accurately detect if the user is making a gesture, and what gesture it is.
    a. Price: ~$30
    b. Resolution: 640x480 at 30 fps, or 1280x960 at 15 fps
    c. Range: 1.2-3.5m
    d. Horizontal FOV: 57 degrees
    e. Vertical FOV: 43 degrees
    f. 16-bit depth sensor
        i. 640x480 at 30 fps
        ii. Horizontal FOV: 58 degrees
        iii. Vertical FOV: 45 degrees
        iv. At the farthest distance, this equates to 3mm per pixel at a depth of 2m, which is perfectly fine
    g. Notes: the depth sensor uses IR light, which means that sunlight will mess this up

2. **Intel RealSense Depth Camera D415**: Much more expensive camera (~$300) than the Kinect, but much better. Again, use computer vision and machine learning to accurately detect if the user is making a gesture, and what gesture it is.
    a. Price: $300
    b. Range: 0.3-3m
    c. RGB Sensor
        i. Resolution: 1920x1080 at 30 fps
        ii. Horizontal FOV: 69 degrees
        iii. Vertical FOV: 42 degrees
    d. Depth Sensor
        i. Resolution: 1280x720 at 90 fps
        ii. Horizontal FOV: 87 degrees
        iii. Vertical FOV: 58 degrees
    e. Notes: website says this one works outdoors, likely unaffected by sunlight

Which sensor from the list do you plan to use?

Intel RealSense Depth Camera D415, if Professor Sample has one. (Update: he has the D435 model and lent it to us for the duration of this project - this has similar specs to the D415 and the assertions in this proposal still hold)

Why is this sensor a better choice than the other options from above?

Both are cameras with RGB + depth. We plan to use both attributes, but depth will probably give us more information. RGB can help us determine what part of an image is an arm. But if our back drop is a similar color to our skin, RGB will not be that effective. This is also why we need a depth sensor. Our arms will never be at the same depth as our couch (if they were, my arm would have to be in the couch). So our arms will stand out pretty easily using a depth sensor.

The RealSense Depth Camera is a much better camera than the Kinect. The range of the cameras are about the same. However, the resolution and frame rate of the D415 is far better than the Kinect. Especially with the depth sensor- the D415 has more than three times as many pixels at thrice the frame rate. This is important because we want our gesture classifier to have as much data as possible.

We also timed how long it takes one of us to wave our hand from right to left (sample gesture), and it averaged around 0.4 seconds. Since the gestures are so fast, having a fast frame rate is going to be really important.

Actuation

Controlling the TV

In order for the user to control what is on the TV screen with gestures we need a way to send controls associated with gestures to the TV.

List all the actuation methods that would allow you to adapt the physical tool:

1. **Roku + Roku External Control API**:
   A Roku will be used to output video to a dumb TV. We will then send remote control commands to the Roku from a host computer classifying gestures from the 3d-depth camera. Roku commands will be sent using HTTP requests and Roku's External control API. From our preliminary testing we have verified that we can send commands to the Roku from a laptop.

2. **IR transmitter for an existing smart TV:**
   Using a IR transmitter in the form of a modified TV remote, IR LED connected to a microcontroller, or IR remote control smartphone app, we would spoof remote control signals sent to a smart TV. Next, we would develop a program that controls the IR transmitter and run it from a laptop classifying gestures from the 3D-depth camera.

3. **Mock Smart TV interface:**
   We would write a mock smart TV user interface that runs on and is controlled by a laptop classifying gestures from the 3d-depth camera. We would then stream a video signal from the laptop to a smart TV, Fire Stick, Chromecast, or Roku so we can evaluate how well the gesture system performs with users in a TV environment.

Which actuation method from the list do you plan to use?

Method 1: Roku + Roku External Control API

Why is this actuation method a better choice than the other options from above?

We chose method one because it demonstrated the least amount of drawbacks and engineering hurdles when compared to the other methods. If implementing method two our team would be limited to working with smart TVs and tasked with developing the electronics and software to control an IR transmitter. The time needed to build the IR transmitter would detract from our team's focus on developing and implementing a robust depth gesture system. Likewise, although developing a mock smart TV interface in method three would grant us the most flexibility in designing the onscreen UI, it would require a significant engineering effort and take time away from the core of the project.

We also considered using alternative platforms to the Roku such as Google Chromecast dongles and Amazon Fire Sticks. However, from our research these other devices have more restricted APIs and don't allow for easy spoofing of remote control inputs. Because of this we determined that the Roku would work best for our project.

Order Sheet

Item	Price	Website	Delivery Time	Description
Roku Express 4K+	37.59	https://www.amazon.com/Roku-Express-Streaming-Wireless-Controls/dp/B0916TKFF2/ref=sr_1_1?crid=27NKTY019J8OJ&keywords=Roku%2BExpress%2B4K%2B&qid=1677792824&s=electronics&sprefix=roku%2Bexpress%2B4k%2B%2Celectronics%2C104&sr=1-1&th=1	2-4 days	Smart TV Module with power/volume controls
Intel Realsense Depth Camera (D415)	$0 ($125) - Borrow from Lab	https://www.ebay.com/itm/125752938529?epid=5021446129&hash=item1d47759021%3Ag%3AtyQAAOSwrk1j0yBp&amdata=enc%3AAQAHAAAA0A4v%2F%2BCEzP5Fi7U5CNgxz8D2pgGEqwAvNZ90bVyv7hqafEd8XUUcI9ukajkAVBWxrsG24eXqXXouN0W%2B0M6LAiuDxuY8ijORdcRyYu2z6zU6SbkffEAVNiGJZDIwVn6Ti5NvJfxM2taOKJjvI9s%2FTPtdj%2F7bevWhiFfIbNaoCER2I%2BmzCiiIrUY1V%2FvM5X6WpUJAr3zBdP0JBQwK5VceeKgzSsGXjWaksd4MSMhkGyHnwRVHOhmLoYv50EaDMPLaACAiB9YVzkz367ra8Q81gcw%3D%7Ctkp%3ABFBMwNe5xNRh&LH_BIN=1	5-7 days	Depth Sensor

Milestone Plan

Milestone	**Requirements**	**What we will implement during the week (max. #4 separate todos per week):**	**What we will show in the video to demonstrate that it actually works:**
3/15 - Milestone #1 (setup)	Setup dev environments, integrate with camera	#1 Order Rokus and setup development environments for programming and debugging	Show IDE setup with OpenCV on our laptops
		#2 Write a script to interface with depth camera and configure for recording via OpenCV	Show stream of depth camera in OpenCV
		#3 Develop tv gestures/commands to use for controlling the TV.	Demonstrate multiple gestures and verbally describe the associated TV command
		#4 Successfully record training data for the project.	Show recorded depth video playing
3/22- Milestone #2 (single-command)	Develop gestures and single command classification machine learning model	#1 Write a script to forward remote control command to the Roku	Code execution showing example output of gesture classification command being properly sent to Roku, i.e. a GESTURE_LEFT classification causing a left button command to be sent to Roku
		#2 Research various types of ML models for classifying visual gestures	Video of written research and/or verbal description of various models and the pros/cons of each
		#3 Use OpenCV and machine learning to classify gestures for one command	Demonstrate video of gesture being input into model with correct classification as output

		#4 Integrate single-command classifier with Roku command script	Show a single command gesture being successfully relayed to Roku
3/29 - System Integration (multiple-commands)	Setup entire device pipeline, from gestures/camera to the Roku	#1 Update roku command script to support multiple commands	Code execution showing remaining gesture classification commands being properly sent to Roku
		#2 Update computer vision model to support multi-command classification	Demonstrate video of multiple gestures being input into model with correct classification as output (not 100% accurate, just demonstrate multiple gestures being recognized)
		#3 Integrate the multi-command classifier with roku command script	Show multiple command gestures being differentiated and most of them successfully relayed to Roku
4/5 - Evaluation Round 1	User Testing Round 1	#1 Create a user multi-command evaluation procedure	Show a written document with user evaluation procedures and questions
		#2 Conduct first round of user testing with 3-5 users	Spreadsheet of response data from user testing, along with perhaps a video of a user testing the device (if they consent)
		#3 Implement improvements to system based on user feedback	Video of improved system describing the changes and why we made them
4/12 -	User Testing	#1 Revise/augment the original user	Video describing the

Evaluation Round 2	Round 2	evaluation procedure	difference between original and revised user evaluation
		#2 Conduct an second round of user testing with 3-5 *new* users	Spreadsheet of response data from user testing, along with perhaps a video of a user testing the device (if they consent)
		#3 Implement improvements to system based on user feedback	Video of improved system describing the changes and why we made them
4/17 - Expo	Final Deliverables	#1 4-page extended abstract text	Submission of paper
		#2 References, video figure, and still images for abstract	Submitted paper includes references, video figure, and still images
		#3 Expo presentation demo video	Submission of demo video