Bob Manasco
manasco2
CS 498: Internet of Things
Capstone Project Report

"Lines of Reflection" – the Actor's Automated Scene Partner

For my final capstone project, I decided to build a smart mirror specifically for someone practicing a speech or actors who are learning lines for a play. In my free time, I participate in the local community theatre, and one of the challenges faced by actors is memorizing lines on their own. It is far easier to learn lines (especially to get the exact wording of each line) with a scene partner – someone who can follow along in the script and read all other characters' lines and prompt the actor if necessary when it is his or her turn to speak again. But this is a very time-consuming, repetitive, rather boring job, and it is not always possible to find another person to take on the task. It would be extremely beneficial for actors if they had a device that could serve this purpose. It would be even better if it was something that could allow them to both see their facial expressions and gestures (for instance, by looking in a mirror) and would also follow along with spoken lines (including prompting when necessary). This could allow the actor the benefit of experiencing the exact words from the script, without trying to follow the script themselves and without accidentally seeing their own lines.

Smart mirrors (or "magic mirrors") are relatively commonplace on the internet, but the key difference that makes this an innovative project is that I have enhanced the mirror with cloud-based voice recognition, so that it does not merely display information, but also can understand and respond to vocal input. When actors are trying to memorize a part, they can speak their lines in front of the mirror. This mirror can follow along, display the actual line if the actor isn't accurate, and then show the next lines from other characters to cue the actor for his or her next line.

Not only does this gadget actually solve a real-world problem, it also leads very naturally to future enhancements. I had originally planned to also enable text-to-speech, to allow the mirror to actually read cue lines aloud to the actor, and I actually did all of the coding for it (you can see that on my Github code, link below), but the Adafruit speaker bonnet that I bought seems to be crashing my Raspberry Pi – after installing it, I was only able to reboot into command-line and the X-server stopped working – so perhaps this will make a nice feature to add in the future. Another useful task would be leading an actor through vocal warmups, tongue twisters, and stretching exercises. It could give feedback on pacing or timing. This could be a tool for singers to warm-up on scales and arpeggios, or perhaps could be programmed with background music for singers to practice their songs. The further I worked on this project, the more excited I got for its future growth potential. I am looking forward to tailoring this for community use. In fact, the script I am using as I test is the libretto for the musical Mamma Mia, for which I am the music director this summer. I plan, as early as next week, to take this mirror to the theatre and allow cast members who are not actively rehearsing to use it to practice their lines.

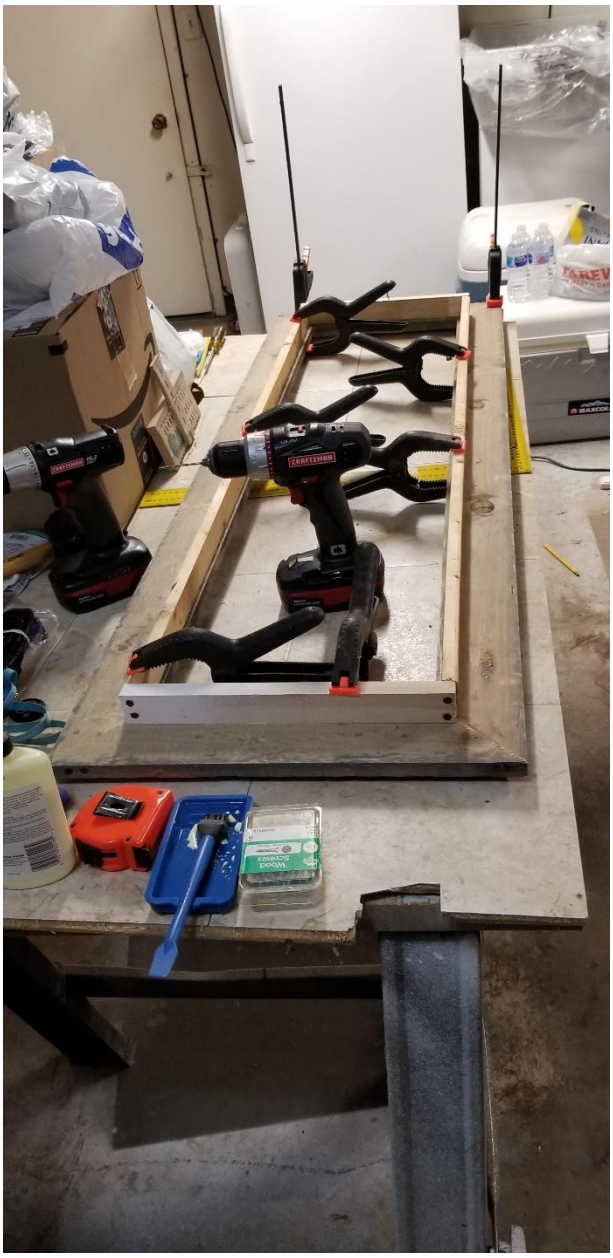
Step One: Planning and Procurement

The first step of this project was planning how I was going to build this device, what materials I was going to need, and then acquiring them. I ordered another Raspberry Pi from Amazon, I picked up some silver ghost wood trim and miscellaneous hardware from a big box home improvement store, and ordered a cheap, rebuilt flatscreen monitor with no stand on eBay.

The most difficult item to come by was the one-way mirror glass, but I was finally able to track down a supplier at a local glass company, who was able to place an order for some MirroView glass with a company called Pilkington.

Step Two: Construction

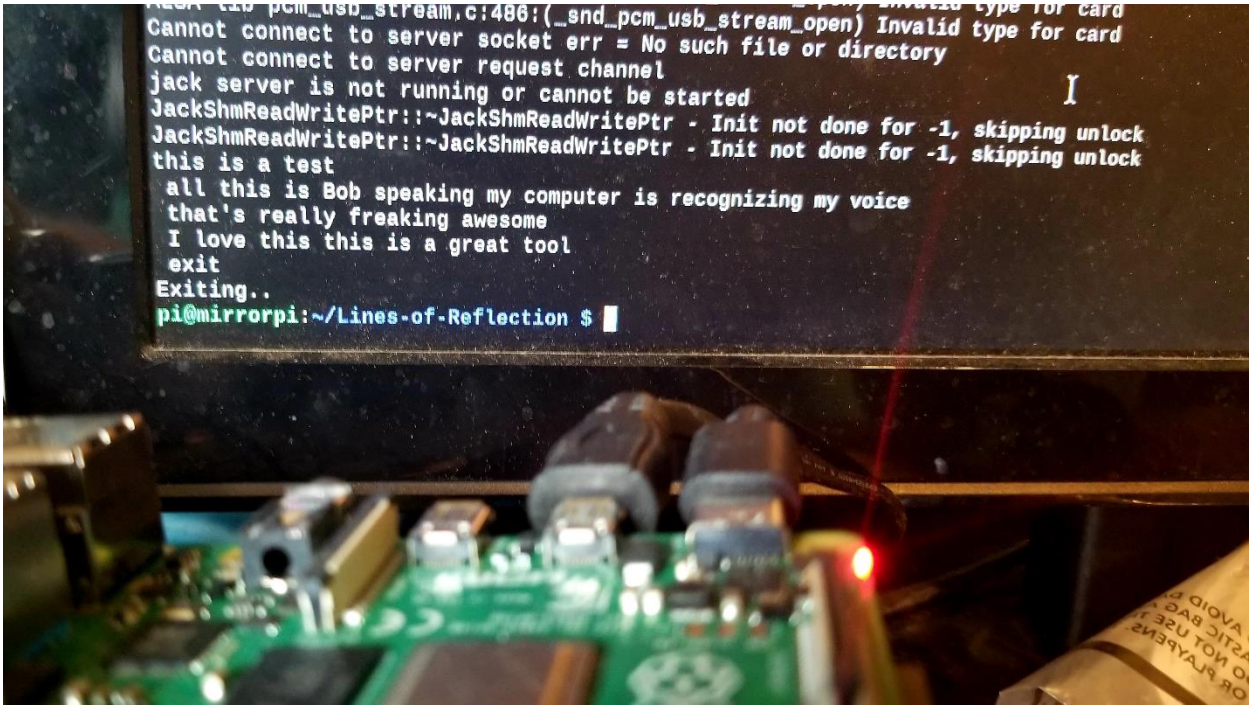
Once I had the raw materials, next came the physical build. This was not terribly complicated, I built a rough box the same dimensions as the mirror glass, then made a flat frame out of the silver ghost wood. The trickiest part of this step was making sure the wooden box was just the right size, so that I could slide in the mirror glass, but it would still hold it snug.




Step Three: Enabling Voice Recognition

The vast majority of the time for this project was spent on getting voice recognition to work properly. In planning, I determined that I needed to utilize cloud-based voice recognition APIs for the best possible vocabularies. Off-line voice transcription is fine for a limited number of words, for instance, if you were enabling a few command words. But since I wanted my device to be able to understand most of the English language including even proper names, cloud-based services were going to provide the most robust solution. I also thought I wanted to use them in an asynchronous manner, where I would get words as they were spoken, rather than waiting for an entire sound block to be completed before submitting for transcription. I thought this would allow me to keep better track of a person's place in their script, for instance, if they missed a word, or substituted another similar word, the device could look a few words ahead to see if it was close enough to proceed. I spent many days of effort trying to

make AWS Transcribe work through WebSockets before finally giving up and moving over to Google Cloud Speech-to-Text APIs, which worked much, much better. After setting up a new Google Developer account, I was able to get real-time voice recognition working on the Raspberry Pi within a few days.

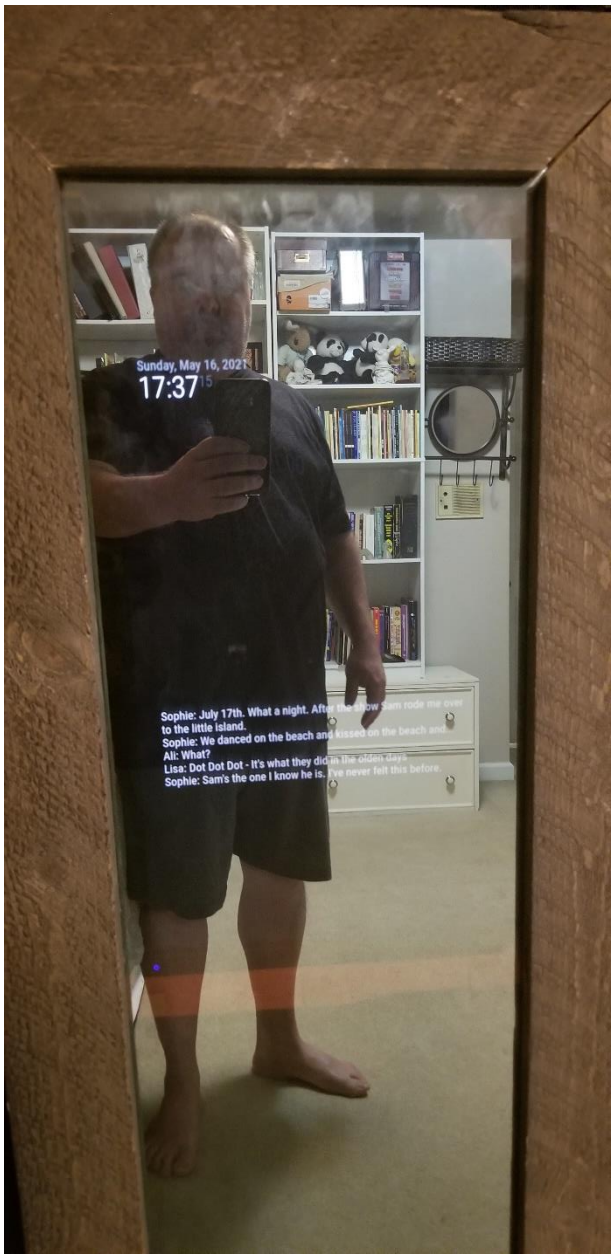


Step Four: Bringing It All Together

The final step of the process was to take the transcribed words, compare them to the provided script, and properly display the results in the mirror. For the display piece, I elected to go with MagicMirror², an open-source modular smart mirror platform. This tool made it easy to display text in a large font on the screen, in an aesthetically-pleasing manner, by using modules previously released by other MagicMirror² developers. Unfortunately, when I got to this point, I realized that I could not actually use the real-time voice recognition Python program I had just written. MagicMirror² is programmed primarily in Javascript and NodeJS, but when it interfaces with Python, it expects to call one Python script and get one element of data to display. Since I was planning to have one long-running Python script that periodically provided multiple updates into a passed parameter, this was not really compatible. So, I ended up outputting everything from the Python script to a text file (disabling buffering), then picking up the last few lines of that text file every second in MagicMirror². Instead of keeping track of which line and which word was being spoken, I instead checked the entire transcribed line with the entire original line from the script, using the Levenshtein ratio, comparing the distance between the two as if they were vectors in the Vector Space Model. For this, I used the Python package FuzzyWuzzy, which was great for this type of comparison. Even after pre-processing the data to remove capitalization and punctuation, I noticed in actual testing that some lines, even if spoken correctly, were given a score as low as 60% correct due to minor mistakes made by the voice-to-text software, so I set that as the threshold for success.

I also coded command phrases of “Line, Please” and “Computer, Exit” to allow users to ask the computer to either feed them their full line (if they need help remembering) or to exit from that portion of the program, if desired.

If I had this project to do over again, I would have simply coded my Python script to record a whole chunk of dialog and submit the entire file for transcription at once. That would have been much less complicated, and probably would have given satisfactory performance. But the solution that I implemented works just fine, and it has no real problems either.



Hardships and Difficulties

As I mentioned before, the biggest issue I faced was trying to enable speech recognition on my Raspberry Pi. I spent an awful lot of time working on AWS Transcribe and WebSockets and investigating HTTP/2 before finally making the switch to Google Speech-to-Text APIs. Even then, once I had that working, there was a fair amount of investigation and re-work to try and make it work with the MagicMirror² platform, before I realied I needed to process it in synchronous rather than asynchronous mode. Finally, I was almost completed with the project when I enabled text-to-speech and attempted to attach the Adafruit Speaker Bonnet, which somehow damaged my SD card, rendering it unable to boot to the desktop or use the X-server. I had to swap out and use the card from my earlier labs (which I fortunately had on hand). All the photos you see in this report were from before the Speaker Bonnet fried my SD card; the video was the only thing I had left to do, but that cost me a fair amount of time, getting everything reformatted, set up, installed, and re-configured again.

Cost and Other Practical Concerns

While I'm very excited about this project and will put this device to good use in my own home and community theatre, I am unsure if it would be a good candidate for wider sales, primarily due to the cost. Between the Raspberry Pi and related hardware, the monitor, the expensive MirroView glass, and the wood frame, it cost on the order of $325 to build, and that doesn't count any charges from Google for running their APIs. I'm sure with economies of scale and other cost-cutting measures, I could reduce that somewhat, but for a convenience tool, I'm not sure there would be a huge market for this. But as I said, I am happy to have it in my own home, even if it's not likely to make me rich.

Conclusion

I am so proud I built my own Internet of Things device from scratch by myself. I never would have believed my final project could be anything this elaborate when I first read the assignment at the beginning of this semester. I learned a lot, not only on programming for machine devices, but also involving cloud computing, both at AWS (even though I wound up not using that particular vendor) and Google. I'm very excited to take this to the theatre next week and allow my cast members to start using it for Mamma Mia right away and see if they have suggestions for improvements. I already know a few things I want to enhance in the future but am eager to have more feedback from others' perspectives. Thanks so much for a terrific course and I look forward to any critiques you and the T.A.s might have, as well! The link to my GitHub and my video are below:

Link to my video: https://mediaspace.illinois.edu/media/t/1_84bzc1ut

Link to my GitHub repo: https://github.com/bo8b/lines-of-reflection