## **Louis gives vision to the visually impaired by bridging the gap between what is seen and felt: Braille on the go.**

## **Inspiration**

A member of our core team is very close with his cousin who is severely disabled. Thus, we approached YHack with a social conscious Hack that would assist those who don't have the same opportunities to attend Hackathons as we do. Although our peers who are visually impaired use current methods and technology including echolocation, seeing-eye dogs, and a white cane for assistance, **existing aides fall short from the potential presented by today's technology**. We decided to design and construct a revolutionary product that allows those who are blind to have a greater sense of their surroundings rather than what lies five feet ahead.

Looking to our community, we reached out and spoke with a prominent economics professor from Brown University, professor Roberto Serrano. He explained that, "The cane isn't perfect. For example, if an obstacle is not on the floor, but is up above, you are likely to bump into it. I would think that some electronic device that alerts me to its presence would help."

Thus, Louis was born, a proprietary, **mobile braille reader** that not only alerts but also locates and describes one's surroundings from a small, integrated camera.

## **What Louis is capable of**

Louis uses a raspberry-pi camera to take images that are then uploaded and processed by the Microsoft Azure (vision) API, Google Cloud (vision) API, and Facebook Graph API to provide short-text summaries of the image. This text is converted to a Braille matrix which is transformed into a series of stepper motor signals. Using two stepper motors, we translate the image into a series of Braille characters that can be read simply by the sliding of a finger.

## **How Louis came to be**

The hardware was designed using SolidWorks run on Microsoft Remote Desktop. Over a series of 36 hours we ventured to Maker Spaces to prototype our designs before returning to Yale to integrate them and refine our design.

## **Challenges we ran into**

In order to make an economically feasible system rather than creating actuators for every braille button, we devised a system using a series of eight dot-combinations that could comply with an unlimited amount of brail characters. We designed our own braille discs that are turned into a recognizable Braille pattern. We ran into a huge roadblock of how to turn one Braille piece at a time while keeping the rest constant. We overcame this obstacle and devised and designed a unique, three-part inner turning mechanism that allowed us to translate the whole platform horizontally and rotate a single piece at a time.

At first, we attempted to transform a visual input to an audio headset or speaker, but we realized we were making a product rather than something that actually makes a difference in people's lives. When someone loses one of their senses, the others become incredibly more precise. Many people in the world who are visually impaired count on the sounds we hear everyday to guide them; therefore, it's imperative that we looked towards touch: a sense that is used far less for reference and long-range navigation.

## **What we learned**

**In 36 hours we were able to program and generate a platform that takes the images we see and others cannot and converts it into a physical language on a 3D printed, and completely self-designed system.**

In addition, we explored the numerous applications of Microsoft Azure and the bourgeoning field of image processing.

## **What's next for Louis**

We are going to Kinect!

Unfortunately, we were unable to gain access to a Microsoft Kinect; nevertheless, we look forward to returning to Brown University with Louis and integrating the features of Kinect to a Braille output. We hope to grant our peers and colleagues with visual impairment unparalleled access to their surroundings using touch and the physical language of braille.

## **Built With**

* microsoft-remote-dekstop
* microsoft-azura
* putty
* [python](https://devpost.com/software/built-with/python)
* solidworks
* 3d-printing
* [raspberry-pi](https://devpost.com/software/built-with/raspberry-pi)
* step-motors
* legos
* gears