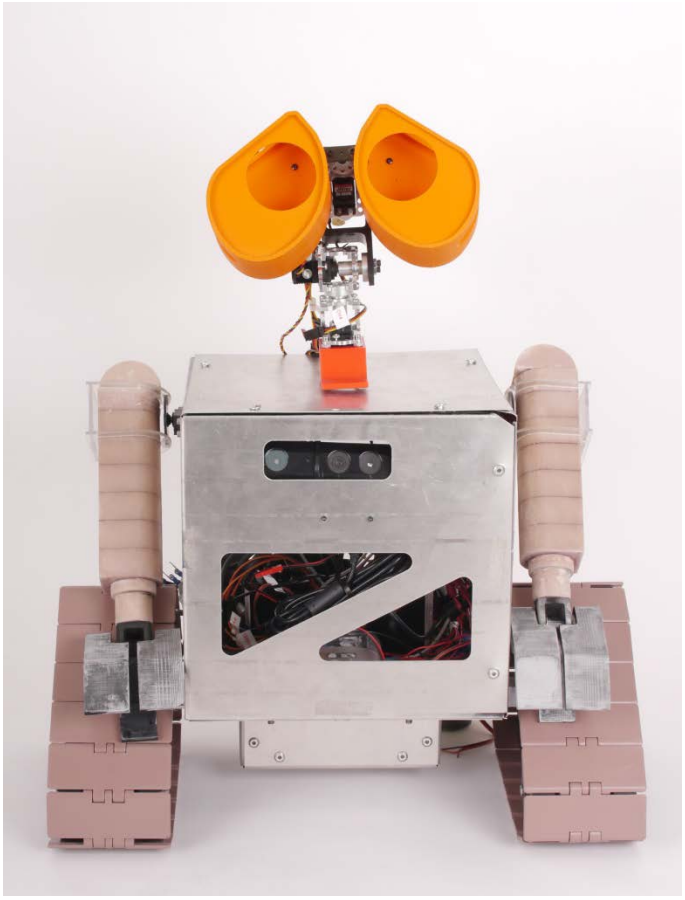
COMPUTATIONAL ROBOTICS FINAL PROJECT: INTERACTIVE GESTURE CONTROL OF A ROBOT

INTEGRATING MOBILE ROBOTS AND COMPUTER VISION TO BUILD A ROBOT CAPABLE OF INTERACTIVE GESTURE CONTROL

ADELA WEE AND SOPHIE LI, FALL 2014



EXECUTIVE SUMMARY

As the price of integrated sensor and computing packages drops, the availability of robotic technologies is quickly increasing. The challenges of developing robotic technologies today is rapidly diversifying to developing behaviors and sensors to make robots more compatible with humans. Our expressive mobile robot based off the robot from the Disney-Pixar movie of the same name, *WALL-E*, was designed with the goal of demonstrating more natural forms of communication through the form of gesture control. This makes communication more intuitive without putting the burden on the human to learn specific commands through traditional elements like a joystick, keyboard, or a mouse.

The Microsoft *Kinect* was initially developed as a game controller for the *Xbox360.* It's unique in that it has two cameras onboard and provides a color (RGB) and a depth image (produced by infrared lasers and structured light techniques), and uses the two to construct estimations of human torsos and is available for less than $100. The *Kinect* has a slew of powerful algorithms behind it; many of them use the depth image and estimate where specific body parts and then extrapolates joint positions based off of that to read the skeletal images. This allowed us to use the black box provided to give us the desired outputs, namely that of hand tracking.

Ultimately we were able to develop a fully integrated mobile robot system that greets you and performs a simple following behavior. By tracking your hand position, WALL-E is able to follow you around a hallway, as it tries to center its field of view around where your hand is located in the image stream. By doing so, we developed a foundation for future projects to take advantage of the recognition capabilities in the *Kinect,* and future interactive platforms that would use ROS, C++, Python, and Arduinos/hardware serial devices.

BACKGROUND

In order for robots to become more integrated into our daily lives, we must be able to have more natural interactions with them. More natural forms of communication involve both explicit and implicit communication. Explicit communication is limited to specific environments, and implicit communication can add clairty, contextual input, and efficiency. If robots were able to partner with humans as effectively as dogs do with their handlers, the social and technical roadblocks to having human-robot teaming and collaboration would be hugely reduced.

We all have a predisposition towards nonverbal cues, as we can easily intuit others emotional states through discussions and personal interactions. Gestures also replace speech when we cannot or do not want to verbalize our thoughts and help regulate the back-and-forth flow of speech in a conversation. Robots will need to be able to interact with a lay audience which will not necessarily be used to interacting with a robot through a computer and a screen, but are generally more familiar with cues such as pointing in a specific direction or waving.

Early on it was determined that an interaction sequence would best be accomplished by building an autonomous robot based upon the Disney-Pixar movie by the same name, *WALL-E*. This specific robot was picked because it communicates primarily through nonverbal interaction with humans and other robots; therefore it has the necessary expressive capabilities to communicate emotional cues to a human teammate. Additionally, it is a non-threatening, friendly platform with which the younger generation can identify, which makes it well-suited for interacting with test subjects. As WALL-E already has a set of expressions and implicit expectations in its interactions, it lends itself well to the predictive aspect necessary to successfully interact with humans.

THE MICROSOFT *KINECT*

It is becoming increasingly important to integrate various technologies and knowledge across multiple fields in order to develop better technical solutions. Advances in consumer electronics have resulted in smaller hardware components that are becoming more affordable and portable every year, and much research has benefitted from the $100 RGB-D camera system developed for the Microsoft *XBox 360*, the *Kinect*. The software utilized in detecting human-like figures has become even more advanced with better machine learning algorithms and filters, resulting in facial feature tracking, gesture tracking, and 3D mapping of indoor spaces. It is now possible to quantitatively measure human body language, as other researchers have done to construct things like interpersonal trust in social interactions.

Thanks to the large team of computer scientists and vision experts that Microsoft assembled to develop the software necessary to use the RGB-D data to do useful computer vision, as well as a large team of hackers/developers from the manufacturer of the original *Kinect*, the device has a lot of easily-accessible libraries. Many of these libraries were developed with machine learning principles where they took data of body movements from a large sampling of individuals and trained their algorithms (using deep randomized decision forest classifiers). These libraries allow for the use of directly accessing specific outputs like skeletal joint positions of a person or detection and tracking of multiple people in a room. The way that the *Kinect* detects objects is through structured light; the principle of looking at a known laser beam diffraction pattern and observing the offsets between the dots, particularly when placed at different distances. This is exemplified in figures 1 and 2.

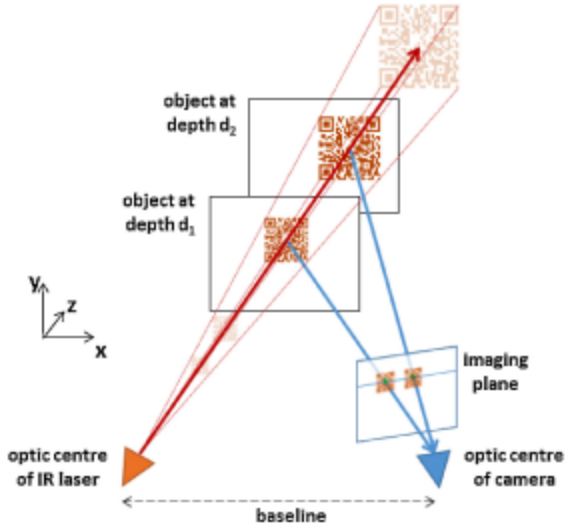


Figure 1.

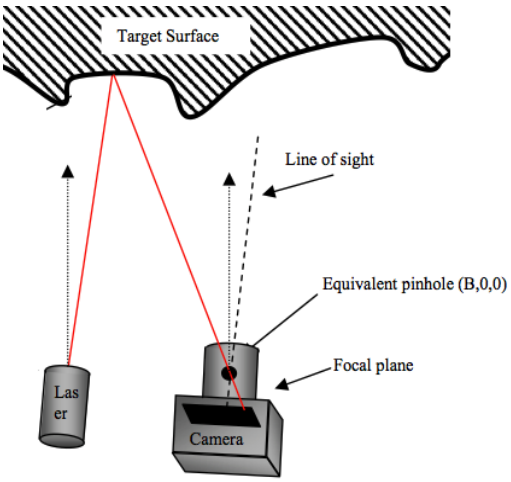


Figure 2.

Figure 1. How the Kinect determines the distance from itself to an object in the field of view. The Kinect looks at the reflection pattern from the laser on the Kinect and uses triangulation techniques and diffraction to determine where an object is. (Image from J. Han's paper, "Enhanced Computer Vision with Microsoft Kinect Sensor: A Review.")

Figure 2. Another perspective of how the depth camera calculates its images based on the laser reflection off the surface as seen by the camera. (Image from *http://media.tumblr.com/tumblr_mba6oaKIyb1r4nbic.png)*

INTEGRATION WITH THE *KINECT* AND NITE

We decided to take advantage of the somewhat open-source toolkit developed for the Microsoft *Kinect*. (It's somewhat open-source because although you can download and manipulate the code, most of the documentation was pulled off the internet once Apple bought the company, OpenNI, that was behind its development.) A lot of the computational analysis

from the Kinect has been abstracted and contained inside the OpenNI and NiTE APIs. While it would certainly be feasible to write our own code using contour detection in OpenCV and simulate binocular vision using two regular webcams, piggybacking off of what OpenNI and NiTE has already accomplished allows us to focus on the higher level responses to human interaction.

The main code that interacts with the Kinect is the "detectfinger" node, which tracks a user's hand after completion of the activation gesture, which is a dramatic push using one hand towards the camera. After the user has done the gesture, the detectfinger node will constantly return the X,Y,Z position of the hand that it is tracking. We then use this stream of information to choose WALL-E's correct response depending on the location of the tracked hand. The detectfinger node also outputs basic gesture information, such as detecting when a session has begun/ended, or when a person has waved. We pass this gesture information on to our python nodes as well, encoded as a custom message type with Boolean variables. A visual representation of this aspect of the code (also known as the "forebrain" in the Olin robot brain architecture) is shown below.

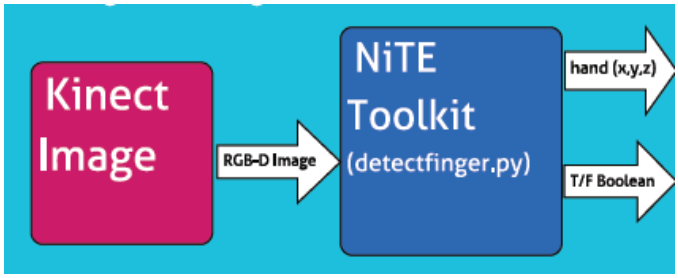


Figure 3. This is a high-level overview of our software flow and shows what we're passing in from the time we take an image with the *Kinect* to after the image is processed by the NiTE toolkit and outputs the values we want (hand position and gesture Boolean information).

INTEGRATION WITH ROBOT

We take a 3 step approach to translating human movement into something WALL-E can interpret. At the highest level, we use OpenNI and NiTE to find the user's hand and track its movements. As explained in the previous section, the "detectfinger" script allows very easy translation of human movement into data points that we can analyze. This "detectfinger" node publishes the XYZ positions of the hand that two different python nodes listen to. The "move.py" is in charge of sending motor commands to the WALL-E's drive-train, while "emotion.py" is in charge moving the servos in the eyes and arms. By analyzing the information from detectfinger, move.py and emotion.py send the desired motor commands and servo positions to the Arduino microcontroller.

WALL-E has an ultrasonic sensor mounted between his eyes which gives us distance information. While we could technically extract distance from the RGB-D images from the Kinect, the ultrasonic sensor is directly connected into the Arduinos that control the servos on WALL-E, which allows for a more reliable transmission of distance data and stopping when WALL-E has gotten too close to the user, particularly when a human is closer than the minimum required distance for the Kinect (which is approximately 4 feet). This distance data is also transmitted as a ROS node of its own, which emotion.py listens to and generates the correct responses based on WALL'E's distance from the user (i.e. say whoa when you're too close).

WALL-E has two Arduino strapped to his back. The first controls the drive train, while the second controls the servos in his eyes and arms. These Arduinos are connected to the ROS network on our computers through rosserial which constantly listens for new messages being posted to their node. The Arduinos have separate state machines, and switch states based on what the incoming message demands. Passing state machine values allow us to communicate quickly across rosserial, which is ideal for a hindbrain layer that runs independently of the whole system. Our full software diagram can be seen in figure 4.

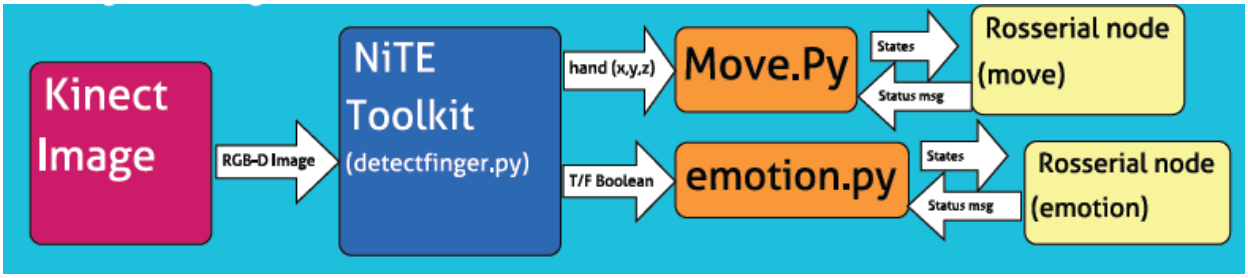


Figure 4. This is a high-level overview of our software flow. In terms of robotic software development, the robot only has two brains: a hindbrain and a forebrain. The forebrain takes in the images and figures out what to do with the given coordinates and Boolean values from the NiTE toolkit, and then commands specific states to the hindbrain which consists of Rosserial nodes running on the arduinos to control the hardware. Because there is no intermediate filtering or behavior arbiter, there is currently no midbrain on this robot.

CONCLUSION

We have demonstrated that it is feasible to interface with the Microsoft *Kinect* through OpenNI and integrate it with an expressive mobile robot to create convincing interaction sequences. WALL-E, as a mobile platform built completely in ROS, successfully demonstrates integration with high-level vision algorithms and multiple computing devices. This system uses the fairly black boxed Kinect API and is able to transmit information through ROS. Via the gesture recognition abilities of OpenNI and NiTE, WALL-E is capable of responding to easily recognizable gestures such as “pushing motions”, or clicks, or simple waves. We’ve written CMakeFiles that can serve as a foundation for future projects that also want to take advantage of the the RGB-D images that the Kinect can produce. Overall we’ve created a fairly believable, interactive character that is mobile and accurately responds to simple waves and can follow us down a hallway.

FUTURE WORK

In the span of 4 weeks, we weren’t able to fully exploit the the capabilities of OpenNI and NiTE. Given the NiTE documentation, we could see that were several modules we left completely untouched during this project. Abilities like scene analysis for the number people, or for push/pull gesture recognition opens up entirely new avenues for robotic responses. Combinations of these modules could be used in future iterations to greatly increase the range of gestures WALL-E could respond to and help the robot become better at understanding more natural forms of communication. Additionally, all processing is currently done on a laptop that WALL-E must carry around. A future version could take advantage of the size and power of modern embedded computers. Although this would require additional research into interfacing with an operating system that isn’t as easy to use as a full desktop version of Ubuntu, the use of an embedded computer would greatly increase WALL-E’s mobility.

ACKNOWLEDGEMENTS

Many thanks to Paul Ruvolo, for his magical C++ and ROS skills, Victoria Coleman and Victoria Preston for their debugging help, Heather Boortz for ROS support, and Drew Bennett for giving us lab space and equipment. Thanks for helping make this vision a reality!

SOURCES

J.J. Lee, et al, “Computationally modeling interpersonal trust.” *Front. Psychol.* 4, 892 (Dec 2013)

DOI: http://dx.doi.org/10.3389/fpsyg.2013.00893

J. Shotton, et al. “Real-Time Human Pose Recognition in Parts from Single Depth Images.” Communications of the ACM, Vol 56, Issue 1 (Jan 2013). Pp.116-124.

http://research.microsoft.com/pubs/145347/BodyPartRecognition.pdf

W. Zeng. “ Microsoft Kinect Sensor and Its Effect.” IEEE MultiMedia. Vol 19, Issue 2. (Feb 2012)

DOI: 10.1109/MMUL.2012.24

http://research.microsoft.com/pubs/179157/Microsoft%20Kinect%20Sensor%20and%20Its%20Effect%20-%20IEEE%20MM%202012.pdf

J. Han, et al. “Enhanced Computer Vision with Microsoft Kinect Sensor: A Review.” IEEE Transactions on Cybernetics. Vol 43, No. 5. (Oct 2013)

http://research.microsoft.com/apps/pubs/default.aspx?id=194894