Pixy Stuck

Attempts to Find an Augmented Reality Development Solution for Unity3d

Matt Cotter & Sam Henry

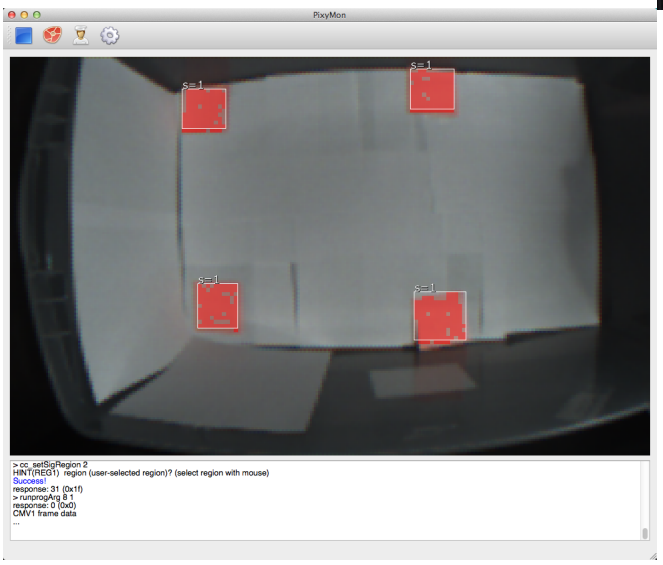
In April 2014, an interdisciplinary group of faculty and students from the North Carolina State University Colleges of Engineering and Design began collaborating with the goal of finding a practical method to develop augmented reality applications. In the first meeting it was decided that the method needed to work with the Unity game engine in an effort to make augmented reality development more accessible. Such a method would would pave the way for future augmented reality development by artists and designers in the Advanced Media Lab at the NCSU College of Design, and elsewhere.

We (the CSC students) came into the project planning to work with the Microsoft Kinect, which has had a non-commercial open-source SDK available since 2011. In the initial meeting the idea of using Kinect was shelved, in favor of Pixy (CMUcam5), which is marketed as a fast, easy-to-use vision sensor. The reasons for this decision were that Kinect's strengths lie in detecting and tracking human bodies through depth perception, while the Pixy's hardware accelerated color and object detection seemed like a perfect fit for the type of interactive games that the group was most interested in exploring.

Experience with the Pixy

Once we received the Pixy we began to explore its capabilities, testing its color and object detection abilities by placing brightly colored sticky notes on a white ceiling approximately four feet away from the Pixy's camera. During this test we began to discover some of the Pixy's limitations, as it correctly identified both full and half sticky notes, but couldn't detect quarters of sticky notes. We also threw a large red ball off of the porch with the camera aimed at the ball's descent path, only to have the ball leave the

Pixy's sensor range about 10 feet down.



In an effort to allay any concerns over the uncontrolled environmental variables on the porch we also tested inside in a clear box, again under ideal circumstances for color detection, using bright sticky notes and a white background with the Pixy's camera placed in a fixed position above the box. While over the box, the Pixy was consistently detecting objects and displaying this information through the PixyMon application,

which is useful for debugging but isn't particularly useful for our project. The libraries that were provided with Pixy weren't working the way they were supposed to.

Connecting Pixy to Unity

There are currently no libraries available to connect a Pixy directly to Unity, so we were forced to try to route the Pixy's data through another platform. Pixy was designed and came with libraries to connect to the Arduino platform, so that was the obvious choice to try next. Unfortunately, when the Pixy was connected to the Arduino via serial port the data transmitted to the computer was incomplete and we could not detect objects through Pixy's provided libraries with consistency and accuracy. We identified buffering issues as the cause of the missing data, but were unable to ameliorate the buffering issues with delays due to our lack of knowledge of the Pixy's architecture.

Still searching for a way to connect the Pixy to Unity, we discovered Uniduino, which "gives you the power to connect the Unity game engine to Arduino" according to its website. When we found this plugin we planned to connect Pixy->Arduino->Uniduino->Unity. Alas, the Uniduino didn't support SPI connections, rendering it useless for this application; however, it did successfully connect Unity to the Arduino through an analog port, which could be useful in future projects.

OpenCV as an Alternative to Pixy

Feeling that we had exhausted our options on the Pixy front, we went back to the drawing board and decided to attempt to integrate OpenCV with Unity. We found some videos of Unity3d projects featuring OpenCV but were unable to replicate this integration. It didn't help that the only guides we could find on the subject were inconsistent, badly written, and/or outdated.

2D Demo Application in Unity

We did create a demo in Unity that we planned to use to demonstrate our (theoretical) augmented reality solution. The demo features multiple objects falling randomly from the top of the screen under the effects of gravity. When the user clicks, a new block is placed on screen, at the point of the mouse, that will disrupt the falling objects' descent, as it would in real life. The plan was for users to be able to place blocks in the real world and immediately see the effects they have on the virtual falling objects, projected onto the surface.

Lessons and Conclusion

Any time a project involves new technologies it runs a risk of failure. New technology doesn't have years of field testing and bug-fixing, or the public documentation that accumulates over time. We didn't have the requisite hardware knowledge to compensate for this, leading to us wasting time on the Uniduino when someone with a different background could have identified the mismatched connectors right away.

Going forward, OpenCV looks like the most viable hope for this project's original goal of enabling augmented reality development, in Unity3d or another environment. Kinect is also an option, as it could potentially differentiate objects by their depth in the right augmented reality situation.