# Vision Project: Smartboard

**Michael Hinton, Jason Lund**

April 14, 2016

## 1 Introduction

Smartboards are becoming increasingly popular in educational and business institutions. Smartboards are digital whiteboards that enable a new way to interact with software and computers. Most smartboards on the market are expensive and require a large piece of equipment that must be installed in one location. Our aim with this project was to lay the foundation for a less expensive and more portable version.

## 2 Methods

We set up a webcam on a tripod to view a tv or projection screen. We would then calibrate the system to tell the camera where in the view the screen was. Then we could "draw" on the screen with a visual marker and the program would draw that digitally and display it on the screen. Our program uses a straight forward outline to create a simple smartboard program.

- Static calibration with chessboard
- Sort corners: put corners in correct order for perspective transform
- Get Perspective Transform
- Detect marker using HSV and inRange()
- Functionality: draw, erase, change color, change stylus size

## 3 Challenges

We originally thought to use a large green rectangle for calibration but ran into trouble detecting the corners consistently as we switched between viewing media (tv vs projector), between projectors, and even with changing lighting conditions. Changing to the chessboard pattern for our calibration made the system much more robust and portable to different environments.
Finding the correct values for the inRange() function was rather tricky. As a future update, we would consider adding a marker color calibration so we didn't have to manually set values to segment out. When using a projector, it will overlay the color over the marker, which changes the marker's color and makes it undetectable. We had to black out the colors around the marker to prevent this problem. Switching to an infrared tag for drawing might fix this. Also, we couldn't draw green because our marker was green. This is also solved by an iR marker.
We also detect incorrect points occasionally and haven't found a good way of filtering those out. A good filter would help smooth the drawing to make it appear less jumpy.

## 4 Future Development

We are happy with how this project turned out and have some ideas on how this might be developed into a better product in the future. Items that we have considered include the following.

- Raspberry Pi or Android implementation
- Less color dependent marker/tag
- Camera calibration for each implementation to make it more robust
- Dynamic calibration (updates every frame using AR tags in the corners of the projection frame)