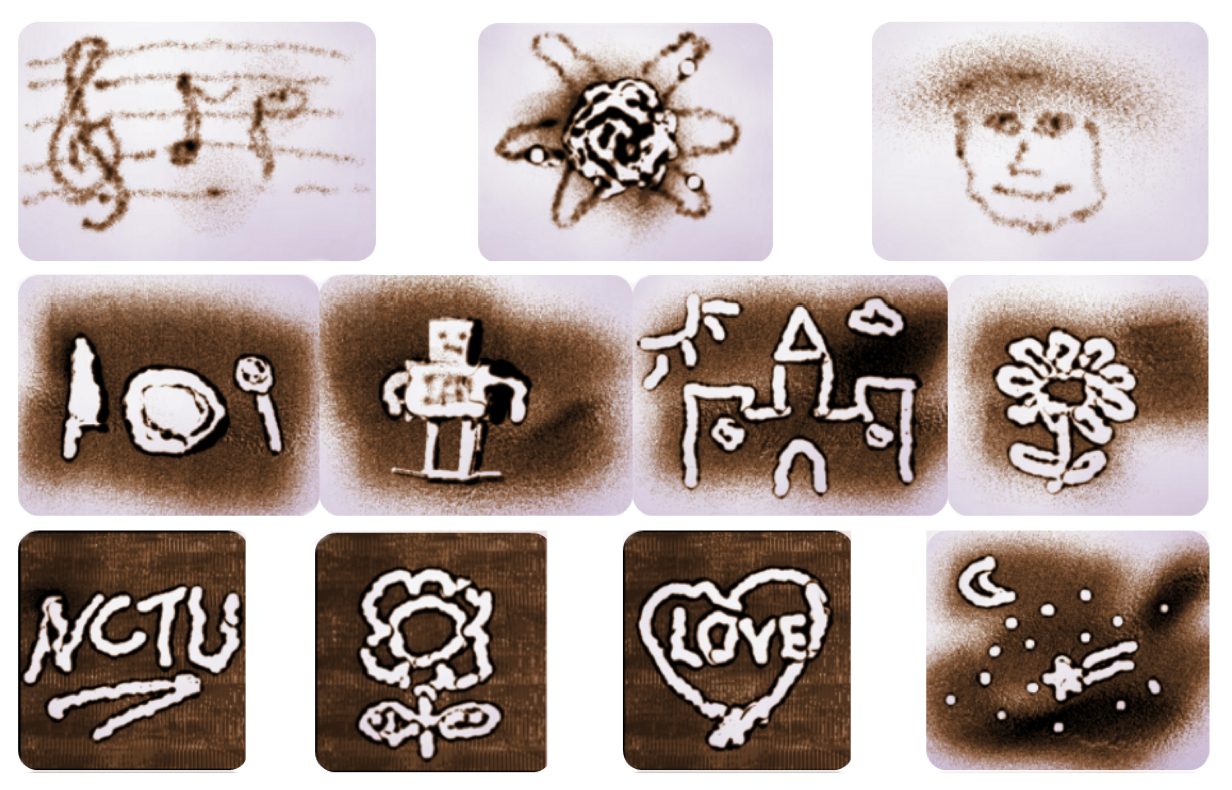
There are many drawing-based iPhone apps available on the App Store, such as KidsDoodle, Sketch, and LINE Brush. The vast majority of these apps involve translating touchscreen movements into brush strokes; apps that enable users to draw using visual gestures such as “ok” signs are much rarer and more difficult to build, especially for the iPhone. Our team aims to develop an iPhone app that allows users to make basic doodles using a small set of gestures.

**Background:**

In late 2014, a group of Taiwanese researchers successfully developed a sand painting user interface using Microsoft Kinect. Users could perform sand-drawing operations by presenting one of four gestures (closed palm, open palm, single-finger, two-finger) to either the command or sand drawing area. The five users who tested the system were able to produce basic drawings within a few minutes; however, most of the testers agreed that the primary challenge presented by the system was their inability to directly touch the sand that they were drawing on (Wong, Chen, & Chen, 2014). This kind of system limitation is expected, as it is much harder to make detailed and precise drawings using gestures, as opposed to pen and paper. However, the system proposed by Wong et al. worked great as a doodling app.

**Figure 1.** Doodles produced by users of the sand drawing interface made by Wong et al. (2014).



Wong et al.’s system was created in three major steps. First, their system detected hands and then computed the hand’s position and gesture. Based on the gesture, the system determined the drawing position and action and sent the information to the sand drawing system. Finally, the drawing system performed the corresponding sand manipulation method.

The task of detecting hands and interpreting hand gestures can in turn be broken down into three major steps, as done by Lee and Lee (2011). First, human skin is detected using background subtracting and thresholding. A scale-invariant angle detection method, based on information about the curvature of the hand, is used to find fingertips. Finally, gestures are recognized by analyzing the position and contours of the detected fingertip (Lee & Lee, 2011).

**Investigation Limits:**

A drawing app as complex as the one made by Wang et al. cannot be implemented in the limited amount of time we have. While Wang et al. was able to detect hands even when the user’s face was visible to the sensor, we will restrict our domain so that the user will be presenting only their drawing hand to the camera. Depending on the sensitivity of the hand detection software to environmental objects, we may impose the additional limit of using a black background. From our experiences with the first assignment, this restriction will most likely be necessary.

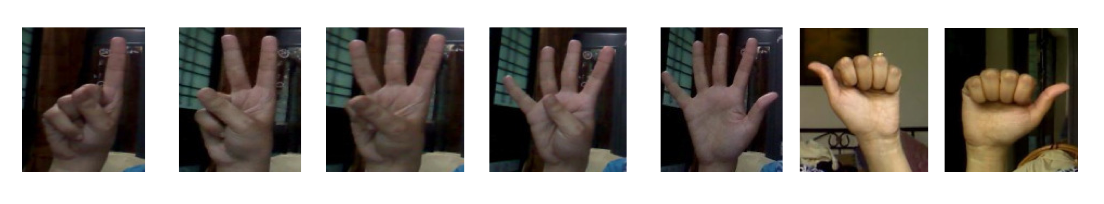
Our iPhone app can be used in two ways: with the camera facing the hand of the artist, so that the person who is drawing can see the effects of their gestures, or facing away from the artist, so that the audience but not the artist can see what is being drawn.

**Anticipated Methods and Results:**

For hand detection, we will be working off one of our assignment 1 projects, which already contains the code for isolating the hand using background subtraction and thresholding (converting it to C++ so that it can be used with iOS). From there, we will refine the code so that it is capable of distinguishing a wider array of gestures, beyond just fists and splays.

In the first assignment, any gesture that wasn’t a fist or a splay was classified as ambiguous. However, now that we require more gestures in our vocabulary, we must also improve our system so that it can identify and distinguish between some of these “intermediate” gestures. Since there’s a lot of room for ambiguity, we need the gestures to be as different from each other as possible. As an example, each of the gestures in the set used by Ghotkar and Kharate (2012) featured either a different number of protruding fingers or a different hand orientation. The gestures we anticipate using are fist, splay (five fingers), pointing up (one finger), peace sign (two fingers), and okay sign (three fingers).

**Figure 2.** Gesture set used by Ghotkar & Kharate (2012).



To get the system to recognize the gestures successfully, the user must have their fingers (if more than one is protruding) widely spaced apart. In this way, our system will be able to discern gestures based on the number of contours between the fingers. Again, this will require a refinement of the algorithm used in assignment 1, which was only capable of accurately discerning between fists and splays. Additionally, the user’s hand needs to be far enough from the camera and from the corners of the image, so that the camera can see their entire hand. If the user breaks this restriction, the system will probably under-count the number of fingers that are protruding from the fist and misinterpret the user’s gesture.

We will probably need an method that analyzes the hand images pixel by pixel and segment the different parts of the hand into clusters based on the locations of the “white” and “black” pixels in the image. First, we define the largest un-interrupted area of white as the “fist.” By this definition, the masses of white pixels that “stick out” from the fist area are considered the fingers. To determine whether the fingers are sticking out, we scan the pixels around the “fist” area for alternating white and black regions. For example, in order to detect the thumb, we need to scan for white (fist)-black (gap)-white (thumb) pixel regions on either side of the fist.

Since we are looking for areas of alternating black and white pixels, the orientation of the hand can be variable – tilting the hand should not affect the system’s ability to recognize gestures. Separating the fist from the forearm, however, is another issue that we need to tackle. In a drawing app, at least one of the gestures must be dynamic, so that the user isn’t always drawing in the same part of the screen. Since the fist is an extremity, we could designate the HOW TO FIND CENTER OF FIST

Once the gestures have been recognized, we will translate them into system actions using the following mapping. We decided to use the fist as the “paintbrush” as tracking and translating the movement of the fist into brush strokes is easier than attempting to translate the position of the tip of the index finger into a brush stroke.

Fist: draw on the screen – the movement of the fist is translated into a stroke

Pointing: change the color of the brush (changes to a random color)

Peace sign: change the style of the brush (ex. from pencil to crayon)

Ok sign: save the image

Splay: erase the image

We will measure program performance by whether it is possible for users to make simple doodles using our system, such as a flower or a house. Since this is a doodling app, not a professional art studio app, we don’t expect to be able to make drawings that contain a great amount of detail. Another measure of performance is whether it works in real-time: in a typical drawing app, the user’s movements on the touchscreen are immediately translated into the appropriate strokes. As a drawing app, we hope to be able to meet this standard as well.

Since coarser gestures such as fist and pointer movements offer far less precision compared to drawing with a pencil on paper, or drawing using Photoshop and a tablet, we anticipate that users will initially have a difficult time producing their desired drawings. For example, in order to draw a smiley face, the user will probably have to draw the entire face in one stroke - the equivalent of completing a drawing without lifting the pencil off the paper. Thus, the quality of drawings produced by our system will be noticeably lower than the quality of drawings made using a typical iPhone drawing app.

Finally, we expect that the gestures will translate to the system action that they are mapped to most of the time. For instance, fists should translate to strokes, rather than resulting in the image getting erased (the expected result of presenting a splay to the screen). To lessen the chances of an unintended image erase, we deliberately left a “gap” in our gestures map – there is no four-fingered gesture in our vocabulary. In the very worst case, only the ok sign should ever be misidentified as a splay. We will consider 95% accuracy as good system performance – 95/100 times, any given gesture should result in the correct system action.

**Evaluation Metric:**

**Management Plan:**

**References:**

Ghotkar, A.S., & Kharate, G.K. (2012). Hand segmentation techniques to hand gesture recognition for natural human computer interaction. *International Journal of Human Computer Interaction, 3(1)*.

Lee, D., & Lee, S-G. (2011). Vision-based finger action recognition by angle detection and contour analysis. *ETRI Journal*, *33(3)*.

Wong, S-K., Chen, K-M., & Chen, T-Y. (2014). Interactive sand art drawing using RGB-D sensor. *International Journal of Software Engineering and Knowledge Engineering*, 17-46.