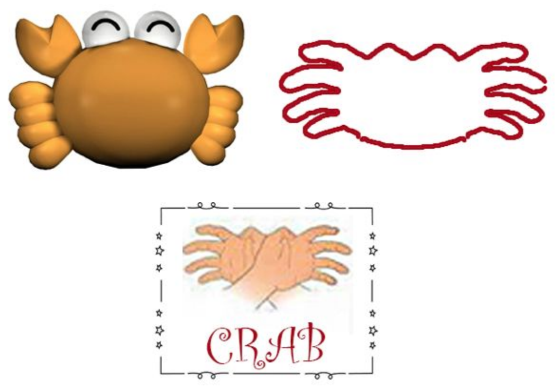
# Project Proposal

Twist Your Fingers -- A Hand Shadow Game

Yixin Wang, Chenyue Meng

## Problem Statement

We plan to create a hand-shadow game with a ViewMaster HMD and an Intel RealSense RGB-D camera. To clearly illustrate the game setting, it will be easier to explain with an example. For example, the user is asked to play the hand shadow of a crab. The scene of this game is shown in Figure 1. It consists of three parts. (1) 3D model of a crab (the upper left part of the figure). (2) The expected contour of hand shadow (the upper right of the figure). (3) A reference photo of the hand to give some hint to the player (lower part of the figure).



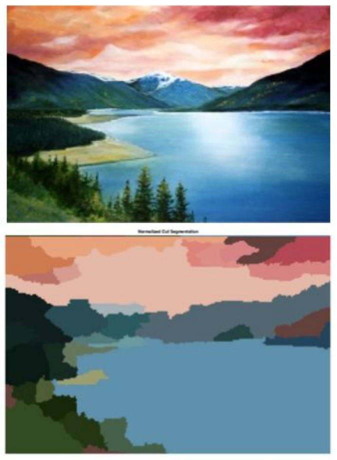


Figure 1. Game scene Figure 2. Normalized Graph Cut Segmentation

When user poses his/her hand in front of the RealSense camera, the following things happen. (1) RGB and depth data are used to distinguish user's hand from background. We use normalized graph-cut algorithm on RGB data to do image segmentation, which is elaborated in the next part. Then we use depth data to take out the segment that is the smallest in depth. This is expected to be user's hand. This is a real-time algorithm and should run very fast. (This is the hard part. If this approach does not work, see Backup Plan section).

(2) We take user's hand segment and display it onto the HMD screen. So the user can actually see his/her hand moving. The user should try to move his/her hand into the expected contour (the upper right part of Figure 1), and make his/her hand fit the shape of the expected contour. With this step, we ensure that user's hand is of fixed size on image plane, so that no further detection or scaling is necessary in step (3).

(3) We take user’s hand segment and convert it to a binary image. Then we compare this binary image with the expected shadow image (also converted into a binary image). We compare these images by looking at how much region these two images does not overlap. This can be easily done by subtracting them and count the area of non-zero pixels.

## Related Work

**Image segmentation for Hand Gesture Recognition**
Segmenting the hand out of complex background is a hard problem. Some people use Fully Convolutional Neural Network to do image segmentation [1]. It can achieve very good results but is computationally expensive (requires large training dataset, lots of training, and GPU support), and cannot be done in real time with our current resources. Most real time algorithms require user to place his/her hand in front of a white wall, and uses RGB value to determine hand contour directly. This can be done fast and easily. However, forcing the user to sit in front of a white wall to play this game may sacrifice user satisfaction. Some real time algorithms require user to fix camera position and take a photo of the background, so that background can be subtracted and hand contour will be revealed. However, this method cannot be used in HMD scenario, because user’s head is going to move.

Another problem with most current approaches of hand segmentation is that they are used for gesture recognition, and thus do not need to be super accurate because down-stream classifier for hand gesture will take care of this offset. For our game, however, we need segmentation to be accurate, since we want user’s to twist their fingers and make the exact shadow we want.

**Graph Cut**
Graph cut is a fast algorithm for image segmentation. The idea is to break links that have low affinity, so that the image is cut in such a way that similar pixels in the graph remain in the same sub-graph. The problem with Graph Cut is that it tends to cut graph into very small components. To solve this problem, Normalized Graph Cut [2] is introduced. An example of normalized graph cut that we implemented with Matlab is shown in Figure 2.

## Creativity

(1) Use Normalized Graph Cut and depth information to capture accurate hand contour. It is real time, uses much less data and computational resources than other algorithms, and is accurate.
(2) We haven’t seen hand-shadow games with VR yet. So this might be the first one. ;)

## Timeline

(1) Get RealSense camera working, and get used to its interface. (May 29)

(2) Implement Normalized Graph Cut and depth thresholding algorithms. Test how well it can segment out user’s hand contour. (Jun 1)
(3) Render the entire scene, including 3D objects, shadow contour and hint image. Display user’s hand image on the same plane as shadow contour. (Jun 3)
(4) Implement the comparison algorithm, which compares user’s hand contour with ground truth shadow. (Jun 5)
(5) Collect more data and make 6 to 7 scenes. (a) Obtain 3D mesh and texture from free websites. (b) Make more images of hand shadow. This data can be obtained by turning off all lights except one point light, make this exact gesture to cast a shadow on a white wall, and capture a photo of the shadow. Then we use Photoshop to preserve the edge of the shadow. (c) Collect reference photos for hints. This data can be obtained by downloading from web or taking a real-world photo. (Jun 7)

## Backup Plan

We are not sure what the final outcome will be with Normalized Graph Cut and depth threshold method. If it does not work, we might have to use the following backup plan. First, we use depth threshold to sample several pixels of user’s hand. Then we get several binary image based on sampled color value. We sum these binary images together and use a median blur filter to get a smooth and noise-free contour of the hand.

### Reference:

[1] Long, Jonathan, Evan Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2015.
[2] Shi, Jianbo, and Jitendra Malik. "Normalized cuts and image segmentation." IEEE Transactions on pattern analysis and machine intelligence 22.8 (2000): 888-905.