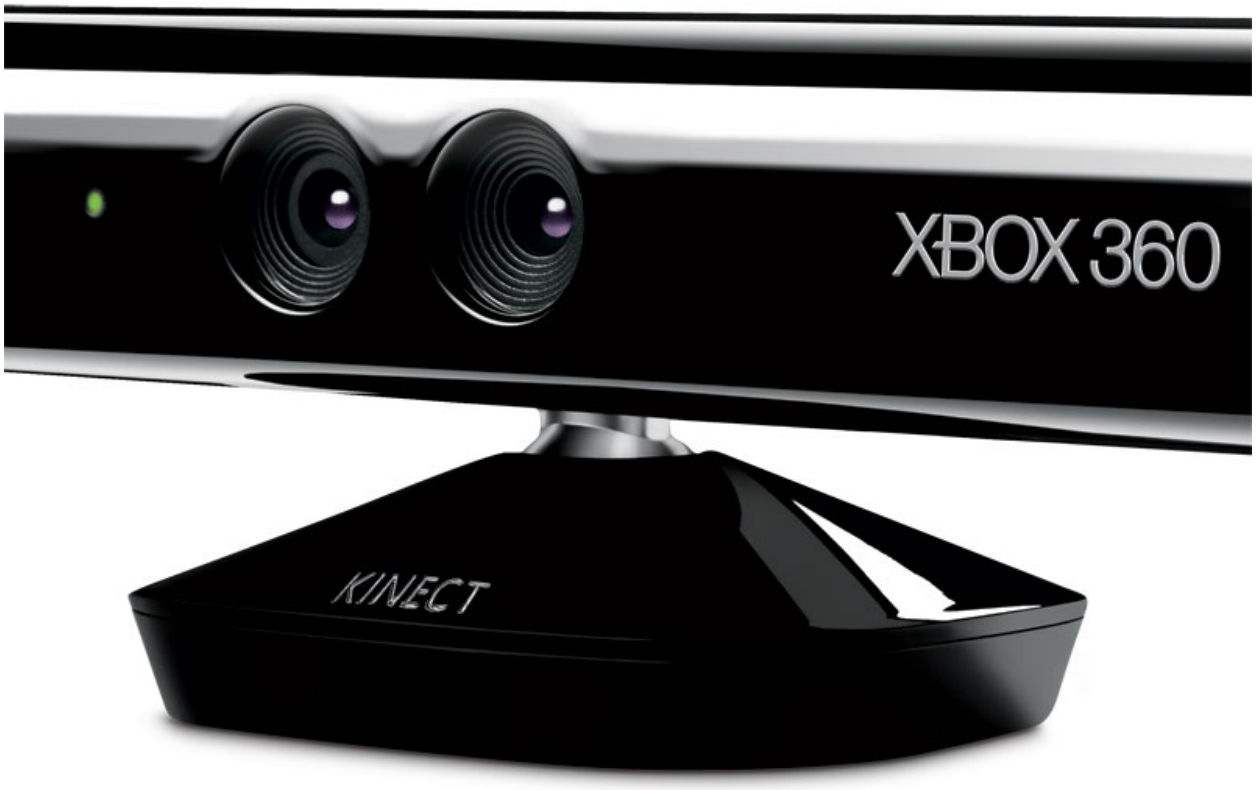
Kinect Gesture Recognition

Ben Fishman,Roman Zubenko, Yan Olshevskyy, Billy Kranich,



Motivation:

A subset of the field of computer vision is camera-based gesture recognition. The advancements in computer vision has motivated many new HCII (Human Computer Intelligent Interaction) applications. As more and more of these applications are developed, they are becoming more dependant upon physical gestures for basic interaction with a user. Applications that make use of gesture recognition are transitioning from novel pieces of software for developers to play with, to a necessary technology that allows for basic improvement in daily life. Some of the practical use cases for gesture recognizing software include minimizing hardware, improving access to computer applications for disabled or impaired users, and advancing medical procedures. Although there has been much research in the field of gesture recognition, the area is still in its infancy.

Problem Statement:

Our goal was to implement a basic hand gesture recognizer using the Microsoft Kinect.

Measure of Success:

We define success in these three categories:

1. Implement three gestures:
    - Zoom
    - Rotate
    - Swipe

2. Accuracy:
    - The hand gestures should manipulate an image in both direction and magnitude proportional to the current state of an image.
    - The transformation of the image due to the hand gesture should make intuitive sense to the user.

3. Visualization:
    - Apply these transformations to 2D and 3D images and visualize them.

Data:

The data used in our project are hand coordinates extracted from the Kinect.

Prior Knowledge and Assumptions:

1. Moving objects are assumed to be foreground and not background.

2. Hand gestures are not moving faster than the hand-tracker can detect.

3. The environment is well lit.

4. We can calculate depth for roll rotations using the Kinect's depth sensor.

5. User is a sufficient distance from the Kinect so that the depth sensor can register the user.

Technical Approach:

Initializing Hand Detection

The first step in building the hand gesture recognizer was simply to detect the hands. We used OpenNI's gesture recognition to recognize the hands using OpenNI's notion of a "click". A click is a specific hand gesture that OpenNI listens for to start tracking hands. The click we chose to you was extending both arms with open palms facing the camera.

Figure1 shows Billy demonstrating a click. The green dots show that OpenNI has begun to track the left and right hand.

After initializing the hand tracker, we were then able to extract the hand coordinates for the left and right hand to be able to implement zoom and rotation transformations. We assigned a queue of size two to each hand in order to store the hand's coordinates. The queue was of size two to keep track of the hand coordinates in the current frame and in the previous frame.

Zoom implementation:

For each new frame, we compare the ratio of the euclidean distance between the hands in the current frame to the euclidean distance between the hands in the previous frames. We then apply this affine transformation to the given image using the OpenCV function getRotationMatrix2D() and warpAffine().

Rotate implementation:

By initializing a vector from the left hand to the right hand we can calculate the angle of rotation as the hands move. In each new frame we compute the angle between previous frames vector and the current frames vector to calculate the law of cosines. $c^2 = a^2 + b^2 - 2abCos\gamma$. This affine transformation is then applied to the image using the OpenCV function getRotationMatrix2D() followed by warpAffine().

To accomplish a 3D visualization we used NodeWebKit and WebGL to render 3D graphics. We created a unix socket to pass the C++ program data to render the the 3D visualization on another computer using Javascript.

Results

We feel that we accomplished the majority of our goals we defined in our measure of success. The two affine transformation of zoom and swipe were successfully implemented in 2D and 3D, and were accurate. Our goal of visualizing both the 2D and 3D image transformations was also successful and was done in a way that is intuitive to the user. The object moves as it would if the user was actually holding the object in his or her hands. We did not implement swipe for we did not see a significant application of the gesture in the context of our program.

Discussion:

The first major challenge we dealt with was how sensitive the Kinect was at picking up minute movements of our hands; a slight shift in hands due to small shakes were captured by the Kinect and resulted in undesirable results. Initially our image would be quivering as a result of the these tiny hand movements. To deal with this issue, we decided to set a lower bound threshold (0.009), that if any delta falls below, the delta is disregarded. In addition to setting threshold we also had to scale down the data we received to allow for smoother transformations of the images. When computing the angle of rotation, we were at first not able to distinguish between positive and negative rotations. We solved this by distinguishing each hand and basing the direction rotation based on the position of the hands in relation to one another. Based on the left hands position relative to the right hand, either above or below, we we were able to apply the proper rotation. One issue we were not able to solve was when the hands cross over one another horizontally. Occlusion of the hands is not supported by the OpenNI hand tracker and usually results in the destruction of one or both of the hand trackers.

Future Possibilities:

There's still things that could be implemented to make this program more useable and practical. This includes the addition of the swipe gesture for a translation of the 2D or 3D object being manipulated. This would be necessary if we would like to place more than one object on a canvas to be manipulated. You could translate the different objects

to different parts of the canvas to make the other easier to work with. This would not be very practical for use with a large amount of objects and would necessitate the need for another gesture listener, maybe a grab gesture, so that the program knows which object you want to manipulate. The grab gesture could also provide a great improvement in usability, for the user would be able to explicitly tell the program when she wishes to be manipulating the object.

Conclusion:

Overall, we feel the project went successfully and we learned many new things about gesture based technology and the theory behind gesture recognition.

Code:

Our complete code can be found at:
https://github.com/wkranich/KinectGestureRecognition