# Gesture recognition using a Microsoft Kinect – An image processing and computer vision project.

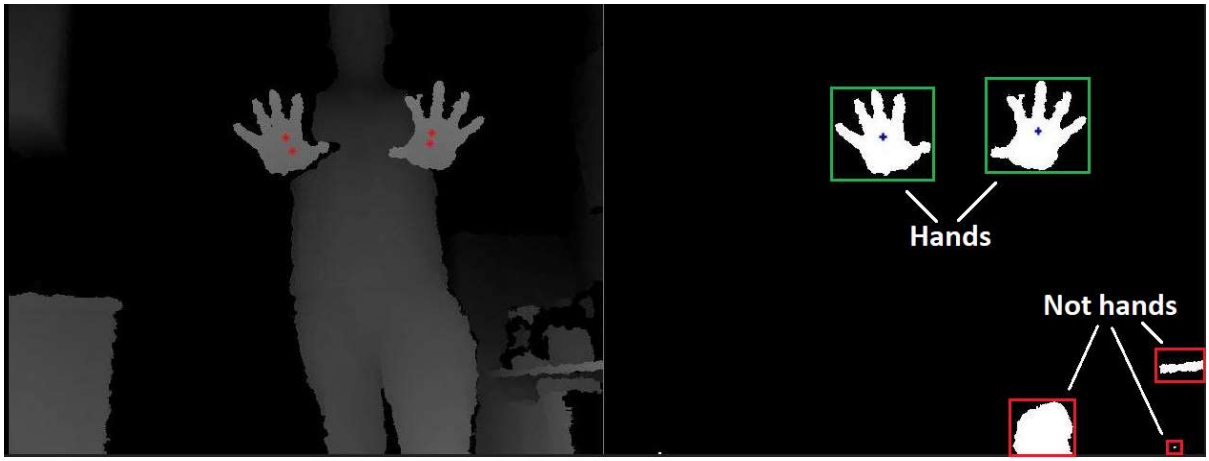
## About this project

This project was done as part of the *Image Processing and Computer Vision* course on 8th semester of Medialogy. The assignment was to use special cameras or sensors ford recognition of different objects. I chose to use the Microsoft Kinect's depth sensor for hand gesture recognition. The project is a reimplementation of a previously conducted experiment by Wang et al (https://ieeexplore.ieee.org/document/7338941)

A video of the result can be seen here: https://youtu.be/6VrQuMu3kyw

## Hand recognition

To recognize hands, I used the Kinect's function to find the "skeleton" of the one in front of the camera. The infrared sensor was set up so that the closer the user is to the camera, the brighter the colour. The program removes all grey values below a certain threshold. When the user stands far enough back (approximately 4 meters) and stretch their arms only the hands are detected. However, if there are other objects close to the sensor, this might cause noise as can be seen in the image below.



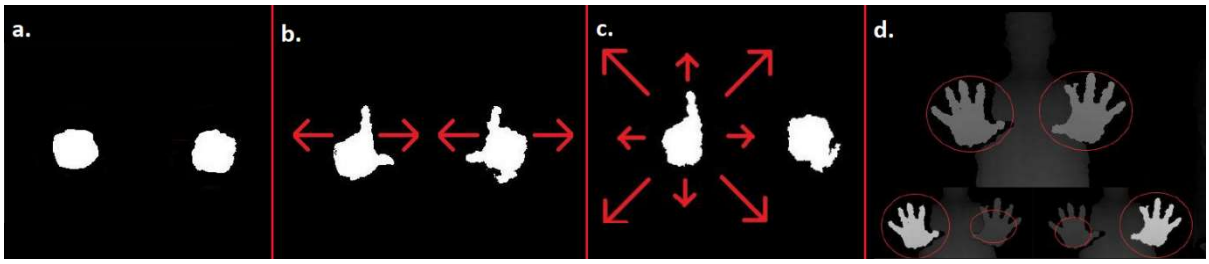

To avoid detecting noise as hands, the skeleton from the Kinect was used to find the objects closest to the hands on the skeleton (The red dots on the left image and blue on the right).

## Gestures

The program would detect four types of gestures, which can be seen below. The actions are:

a. No action
b. Move
c. Scale
d. Rotate

Having both hands closed will work as a neutral state, where nothing would happen.
Having both hands make am “L” shape would allow the user to scale an object, by moving the hands closer or farther away from each other.
Having one hand closed and the index finger pointing upwards on the other hands would allow the user to move the object. Moving the hand with the index finger stretched in either a horizontal, vertical or diagonal direction would move the object in that direction.
Having all fingers stretched would allow the user to rotate the object, by moving one hand closer to the camera and the other one farther away.




## Code snippets

All code was written in MATLAB as it provides optimized libraries for Kinect and image processing.

### Finding the hands between all the detected blobs

Determining the distance from each blob to the hand coordinate of the skeleton, is divided into two steps. First the Euclidean distance is calculated from the center of each blob, to the hand coordinate and saved as a separate entry in an array as shown below.

```
for i = 1:numberOfBlobs
    %The x,y coordinates of the current blob
    xCoord = blobMeasurements(i).Centroid(1);
    yCoord = blobMeasurements(i).Centroid(2);

    %Distance = sqrt( (x2-x1)^2 + (y2-y1)^2 );
    blobMeasurements(i).Distance = sqrt( (objectPoint(1)-xCoord)^2 + (objectPoint(2)-yCoord)^2 );
end
```

Then the closest point is found by looping through the array, comparing each value with the currently lowest one with the current value of the array. If the value is lower, it is stored as the new lowest value. After looping through the array, the blob for the hand is found. This is done for each of the hands separately. I wanted to keep both findings in one function, but this could have been improved by calling the same function twice, parsing the two hands’ coordinates separately.

```
for i = 1:numberOfBlobs
    if ~isempty(closestBlob)
        if blobMeasurements(i).Distance < closestBlob.Distance
            closestBlob = blobMeasurements(i);
        end
    else
        closestBlob = blobMeasurements(i);
    end
end

leftHand = closestBlob;
```