Assisting breast biopsy by visualizing the inside structures of a patient on top of the patient in real time using the HoloLens

L.R.W. (Leon) Klute

BSc Report

Committee:
Dr.ir. F. van der Heijden
V. Groenhuis, MSc
Dr. F.J. Siepel
Dr.ir. L.J. Spreeuwers

July 2017

029RAM2017
Robotics and Mechatronics
EE-Math-CS
University of Twente
P.O. Box 217
7500 AE Enschede
The Netherlands

Summary

This project is part of the MURAB project. The MURAB project is aimed at improving cancer diagnostic operations. To perform breast biopsy, Magnetic Resonance Imaging (MRI) scans and ultrasound (US) data are used to locate the position of the lesions which need to be targeted. A biopsy gun/needle will be inserted which needs to be guided to these targeted areas. The needle will take a sample of the region. Currently a radiologist inserts the needle assisted by an ultrasound probe. The output of the ultrasound probe is shown on a monitor next to the person who is being scanned.

The HoloLens developed by Microsoft provides a way to create virtual objects in the real world, these objects can be tied to positions in the real world. The HoloLens can also be used to create a 3D model of the region of interest in the patient which can then be explored for diagnosis purposes. This can also be used to show the inside of the patient to guide the radiologist toward the lesion.

In this report is described how one can develop an application for the HoloLens to visualize the MRI scans over a patient in real time and show the location of tools such as a needle and its proximity to the target. To do this, the location of the breast, lesions and needle is gathered and translated to the same coordinate system. The gathering of the positional data is done by external tracking devices and sent to the HoloLens. The HoloLens translates the information to give useful information to the physician. Such that it improves the ability of the radiologist to take a sample. It was found that the accuracy of the current system is not enough to perform biopsies, but with several possible improvements it could likely be made sufficiently accurate.

Contents

1 Introduction

To determine whether someone has breast cancer, imaging methods such as mammography, ultrasound (US) and Magnetic Resonance Imaging (MRI) are used to see if there is any indication of anomalies in the tissue. If suspicious regions are found, samples must be taken for further investigation. The current most common way of collecting a sample is using a biopsy gun in a stereotactic biopsy. This needle has a hollow point which can collect some material when placed against the lesion. After it is placed against this lesions, the hollow point will shoot forward after which the outside follows this point to encapsulate the collected material. this can be seen in Figure 1.1.

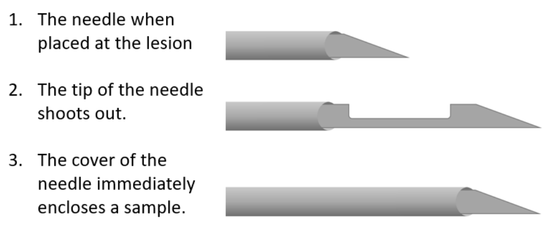

1. The needle when placed at the lesion
2. The tip of the needle shoots out.
3. The cover of the needle immediately encloses a sample.

Figure 1.1: The biopsy gun/needle system, the sample is collected in two quick movements, 1 and 2.

Stereotactic biopsy uses a computer and imaging in at least two planes to localize the region and guide the physician, while he or she places the needle at the suspicious region. This is usually done with live imaging by US. However, of the various imaging methods MRI is the most accurate(3), but can not be used during the biopsy. Because no metal objects, such as the needle, can be present during MRI. In this report an application that uses MRI to assist in probing is explored. Using an Augmented Reality (AR) Head mounted Display (HMD), the HoloLens, the MRI can be used to show the location(s) of the lesion(s) while taking the breast biopsy. Furthermore this HMD can be used to show the location of the needle. Both these were previously performed with US, but this was limited as it can only show one plane at a time, requiring acquired skill by the radiologist to find the lesion with the needle.

1.1 Aim

The aim of this project is to explore whether it is possible to use an Augmented Reality device, such as the HoloLens, to assist in biopsies and whether this improves the ease of taking the biopsies.

1.2 Stakeholders

The stakeholders of this projected system are:

- **Hospitals**, they want cheaper diagnostic methods. Using less material would be an example of decreasing the cost of the diagnosis process.
- **Radiologists**, they want an easier way to find the lesions.
- **Patients**, they want to be punctured by a biopsy gun/needle the least amount of times. Also, the process of taking a biopsy should not last a large amount of time.

1.3 Background

1.3.1 MURAB

This report is part of the MRI Ultrasound Robot Assisted Biopsy project, which aims to improve the precision and effectiveness of cancer diagnostic operations.

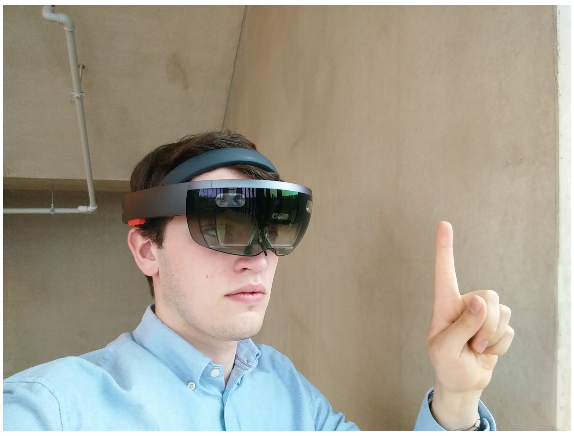


Figure 1.2: The HoloLens being worn by a user, who is holding up his hand so it is noticed by the device. The device can be used wireless and the glasses are see-through. The hand is recognized by the 4 depth of field cameras present in the upper front side of the HoloLens.

1.3.2 HoloLens

The HoloLens is Microsoft implementation of creating a virtual overlay on the real world, what they call mixed reality. The device is worn on the head and does not need an external computer to generate the graphics, because that can all be done on the device itself. The device has several ways it communicates with the real world. The device can be seen in Figure 1.2. With these sensors it overcomes problems that previous augmented reality glasses suffered from, and withheld common use(5). The HoloLens is portable, but still has enough power to render stereo images. It can adjust the focal length, because it can map its surroundings.
The glasses on the device are optical see-through, meaning they are transparent, but when something is displayed on them this is seen in front of the real world. Both eyes have a display in their respective glasses which can show different images, creating a 3 dimensional experience.

- **Voice command;** the HoloLens has support to recognize spoken words, which can be tight to function it can perform.
- **Gaze;** the HoloLens is aware what the user is looking at and can react to this.
- **Gestures;** the HoloLens can recognize hands with the depth of field cameras (11). It can recognize such as ’tap’, ’double tap’, ’hold’ and can combine this with the location of the hands to create complex instructions.
- **Other sensors;** the device also interprets the world with depth of field cameras, and tracks its own movement with accelerometers.

2 Design

In this chapter the desired operation and the parts needed to achieve this operation are discussed.

2.1 Intended mode of operation

The current course of action during the diagnostics consists of the following steps: First, mammography is used to see whether there are suspicious anomalies inside the breast. If anything is found, the radiologist will use US to take a closer look at the anomaly. If this shows it it could malignant, a biopsy gun will be used to take a sample. The needle is guided by US. The sample will need further testing at the hospital to give conclusive results.
The way this new system would work, would be: An MRI scan will be taken from someone at risk of breast cancer. If suspicious anomalies are found, the scan must be processed and loaded into the HoloLens. Then the radiologist uses the HoloLens and a biopsy gun to collect a sample of the anomalies. The taken sample will again be investigated at the hospital to give conclusive results.

2.2 Visualization

One important aspect of the whole system is to show the radiologist, the collected information in a useful way. The information to show exists of multiple objects, which are discussed further on in this chapter. The HoloLens is well suited to show the the 3 dimensional information in an easy to interpret way. The scan needs to be preprocessed to be most useful during the biopsy.

2.2.1 Patient

The information on the patient will come from MRI scans, which give a 3 dimensional scan of structures inside the patient. This can be loaded into the HoloLens as a volume rendering in which all data is shown in transparent points. This way the radiologists can see all structures, but they might not be very clearly visible. Furthermore, volume rendering is computationally heavier, while the HoloLens has limited computational power. Otherwise, the scan can be preprocessed to only show surfaces of important structures, the important structures in this case include the lesions and possibly the skin. Segmentation needs to be performed to get these surfaces from the MRI scan. This will clearly show the location and depth of the lesions. The skin surface can be made transparent or not shown at all. This enables him to reach it with the needle as it can be seen from the outside.

2.2.2 Needle

The needle will also be shown in the virtual world. This is used to see the location of the needle inside the patient, greatly increasing the physician's perception of its distance from the target. Thus a virtual version of the needle will be created. Some form of tracking needs to be used to get the location of the of the needle.

2.3 Tracking of real world objects

There are multiple possible ways of locating the objects in real space. Several methods will be explored. Because the tumors can be found from sizes of about 2 mm in diameter, the accuracy and precision should be at least below 1 mm(1). Which would ensure, when the the HoloLens signals the user a lesions is hit, it is actually hit.

2.3.1 Camera(s) on the HoloLens

The HoloLens has 5 cameras on it, of these 4 are depth of field cameras and are used by the device itself to map the surrounding. These 4 can not be accessed directly by developers and could thus not be used to locate objects. That leaves one camera, which has a resolution of 720 by 1280 pixels(Microsoft). As this camera has only one point of view and does not have a very high resolution it will likely not be enough to accurately find the location of markers on the needle.

2.3.2 Using other device to locate the objects

Other devices such as the NDI Aurora can be used too very accurately and quickly find the location of magnetic markers. These devices will need to communicate with the HoloLens. It is also possible to use a stereo camera setup, which is specifically made for locating needle position, breast position, rotation and deformation. Such a setup which can also see the deformation of the breast would be ideal.
The locating setups need to use an origin which is known to the HoloLens, so the HoloLens can get the information and place the needle at the right location. Both can define a common origin dependent on a specific marker that will be tracked by both, the disadvantage will be that the camera on the HoloLens must be used to find this marker, introducing the inaccuracy of the camera again. However, this marker can be stationary and thus the tracking can be more accurate by also using the HoloLens build in movement and environment tracking sensors.

Communication

The HoloLens has Bluetooth and WiFi capabilities. Bluetooth is often not available on all devices. To use the HoloLens' WiFi capability the measurement device will require standard networking, which most computers have. Assuming the measurement setup can be connected to a computer.
Then a network between the two needs to be set up over which they can communicate. A local network could be set up using a wireless router for safety, or the internet could be used for ease of use while in development.

Origin agreement

The location of the origin for the measurement setup is known by the user, or can be set by him. This location needs to be known by the HoloLens as well. An easy implementation is using an existing image recognition library which could recognize the measurement setup. An open source library such as ARtoolkit could be used with a wrapper(10). Or a commercial object recognition library can be used which has more support. Vuforia is a tool that is supported by Microsoft itself for use in creating Mixed Reality applications in Unity, and has access to the tracking sensors in the HoloLens for extended tracking.
Furthermore, the directions of the axes in the HoloLens and within the measurement setup might well be different. This needs to be accounted for when using the data to place objects. This can be done by applying a transformation to the data that is received.

3 Design Evaluation

How were the designed parts implemented and do they perform as expected? This will be discussed in this Chapter.

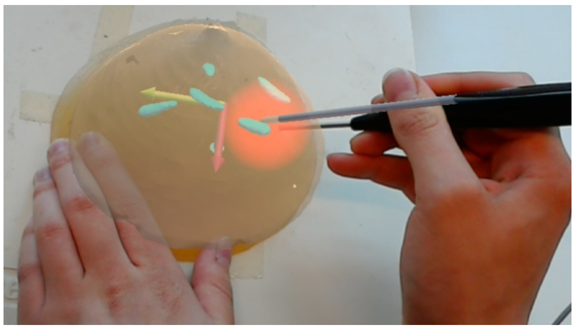


Figure 3.1: When a lesion is hit, a light appears to notify the user. The virtual needle is touching the the virtual lesion, which should mean the real needle is touching the real lesion. Image taken using Mixed Reality capture.

3.1 Steps for operating the developed system

An MRI scan is taken of the breast. The scan must then be segmented and loaded into the HoloLens. Multiple scans can be in the application at the same time, the user can switch between them using voice commands.
During the biopsy, the tracking system must be activated. The HoloLens needs to connect to the computer which gathers the tracking data, this is done from the HoloLens itself by giving it the IP address of the tracking computer, this is done after the program has started.
From this point the position of the needle and the breast are updated. However, they are not yet in the same coordinate system as the measurement setup. Thus the marker must be placed at the origin of this setup, then it must be 'tapped' by the user, which will let the HoloLens know this is the origin of the measurment setup. If the needle is not at the right location, the user will have to put the needle at the origin and tap the 'set offset' virtual button, this will place the virtual needle at the origin as well. After this is done, everything is set up for sample collection. The physician decides the best location the take a sample and can puncture. When the HoloLens assumes a lesion is hit it will light up this location Figure 3.1. Then the sample is taken and needs to be investigated further.

3.2 Visualization

From the various ways to build applications for the HoloLens, the best way was using the game engine unity. Microsoft, the creator of the HoloLens, worked together with Unity3D to build support, the HoloToolkit (Microsoft) especially makes the implementation of input specific to the HoloLens easier to use. These inputs include gaze, where the user is looking, the hand tap, if the user selects using the hand command and voice input. Furthermore, in Unity3D Holograms can easily be added by placing them in a 3 dimensional virtual environment. To these Holograms code can be added which will describe their behaviour. This creates a very user friendly environment to create a the virtual environment for the HoloLens.
Once a project is finished it can be exported to a visual studio solution. Visual studio is an IDE for creating Microsoft applications. The solution can be built into an application which will be run on the HoloLens. Unlike older AR systems the which biggest issue was blending the real world and the virtual world(2), the HoloLens does not suffer from this. It uses its knowledge of

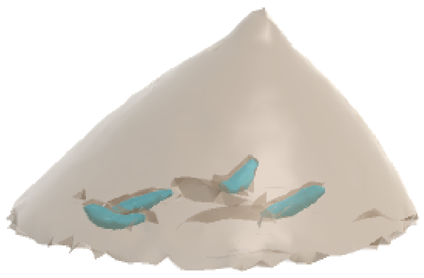


Figure 3.2: Surface renders of the Scan with the outside surface transparent and the lesions having contrasting colors. In the application this virtual breast will be in the same location as the real world breast. The user can see the real world breast through the transparent parts, and can meanwhile see the lesions inside the breast.

the distance to virtual and real world objects to create the objects at the right focal points, and at the right location on the screen for each eye.

3.2.1 Patient

The scans were first segmented, which resulted in surfaces of the exterior of the breast and of the exterior of the lesions. In Unity the skin was made transparent, while the lesions were given an contrasting color as can be seen in Figure 3.2.

3.2.2 Needle

For the stereo camera setup, two markers are placed on the biopsy gun/needle, then the stereoscopic setup is used to find the locations of the markers. These two locations are then sent to the HoloLens which will use them to calculate the position and direction of the needle. A transformation matrix for the location of the breast is determined by the setup and is also sent to the HoloLens. A 3d model of the biopsy gun/needle was created where the distance from the markers to the tip is known. For use of the electromagnetic sensor, only the needle position and rotation are measured and sent to the HoloLens. In both cases the HoloLens transforms the received data to its coordinate frame.

3.3 Locating of real world objects

Multiple devices were used to locate real world objects: The NDI Aurora(NDI) and a stereo camera setup. Both of these setups were accessed through MATLAB (6) which acted as a server for the HoloLens to access the locations and rotations read by the setups.

3.3.1 Vuforia

Vuforia\$ref\{vuforia\}$ is a tool which can be used to recognize images in the real world. In this case it was used to recognize a marker. This marker would be placed at the origin of the measurement setups, so the location of the needle and the breast can be read and placed relative to its real world origin.

3.3.2 NDI Aurora

The Aurora device is a tracking device that uses an electromagnetic field with which sensor can track their location with sub millimeter and sub degree accuracy (NDI). It measures the orientation and location up to 40 times per second. This information is shared through Plusserver software.

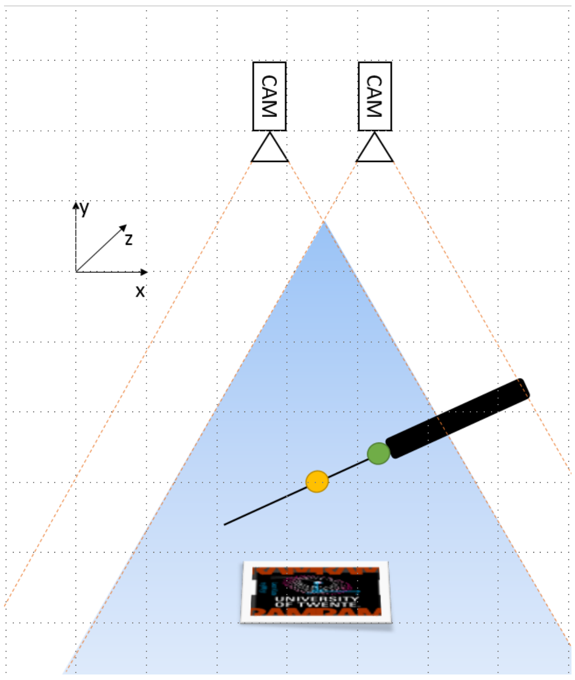


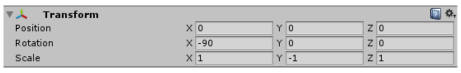
Figure 3.3: Schematic diagram of the stereo camera setup. The light blue region is the region in which the yellow and green marker can be seen by both cameras and thus the 3 dimensional position can be determined. On the bottom a marker is placed at the origin of this system. The marker will be recognized by the HoloLens so it knows where this is, afterwards the marker can be removed.

3.3.3 Stereo camera setup

Two cameras are recording in parallel, with their respective locations known. Spherical markers will have colors which are not present in the view of the cameras, so both cameras can see the spheres. With their respective positions known, and the angles to the spheres from each camera known, the locations can be calculated. With two spheres placed on a needle, its location and orientation can be determined and shown in the virtual world. The setup can be seen in Figure 3.3

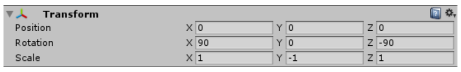
Communication

In MATLAB a function is created which starts a server and sends the gathered data to any client who connects. The HoloLens application has an IP address input. Which needs to be filled in when the IP address of the server with the location and rotation data is known. The communication is set up using a MATLAB function, which will be called every time there is data to send. The first time the function is called it will wait for the HoloLens to connect to it, afterwards it can immediately send the data. The MATLAB function to be used on the measurement setup PC can be found in Appendix A. There you can find one MATLAB function, which acts as a server The script takes in the a matrix with data points and transforms it into an array which can be decoded by the HoloLens. The functions to receive and decode the data are also found here. It can be seen that no Transformation is performed here as it is done within Unity, by placing the the objects in a different coordinate system which originates from the HoloLens coordinate system with a transformation as seen in Figure 3.4. Before the quaternion received



```
Transform
Position    X 0      Y 0     Z 0
Rotation    X -90    Y 0     Z 0
Scale       X 1      Y -1    Z 1
```

(a) The transformation applied to the coordinate system in which the position and rotation of the stereo camera setup sends rotation and position data. The y axis is inverted, which means we switch from a left handed coordinate system to a right handed. Then the coordinate system is rotated -90 degrees around the x-axis.



```
Transform
Position    X 0      Y 0     Z 0
Rotation    X 90     Y 0     Z -90
Scale       X 1      Y -1    Z 1
```

(b) The transform applied to the coordinate system in which the Aurora sensor is placed. Again, the left handed system is turned into a right handed system by inverting the y axis. The system is rotated around the x-axis with 90° and around the z-axis with −90°.

Figure 3.4: The transformations for the different coordinate systems from which data is received.

from the NDI Aurora could be used however. The scalar needs to be moved to the last position. The quaternion in Unity has the scalar in the last position while the Aurora system in the first.

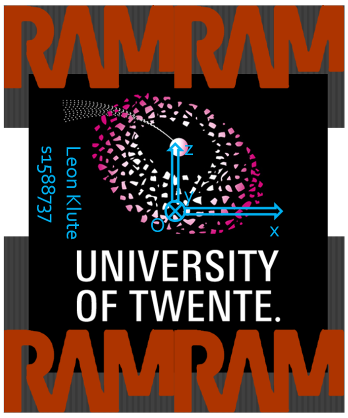


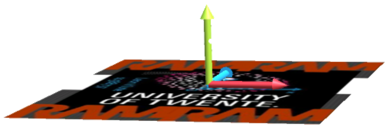
Figure 3.5: Marker which can easily be recognized by Vuforia and which shows the created origin and direction of the three axes. Vuforia recognizes by contrast points, so the letters and the specks on the black background create a lot of recognition points.

Origin agreement

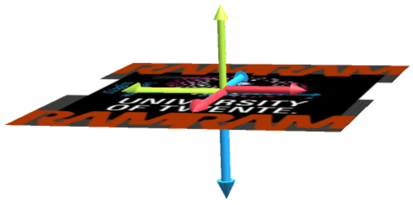
The location of the origin for the measurement setup is known by the user, or can be set by him. Of the two possible recognition libraries discussed in Chapter 2, Vuforia was implemented, as it proved to be more stable by using the internal movement sensor systems of the HoloLens. At this location the Marker in Figure 3.5 is placed. This shows how the HoloLens sees the x, y and z axes. The marker is an image with a lot of contrasting points, which is best for Vuforia to be recognized. To transform the axes in the measurement setup to the axes in the HoloLens virtual world the objects which are placed on these coordinates are placed within a Unity Transform, which is edited in such a way, it is the same as the measurement setup, as seen from the common origin. This can be seen in Figure 3.6a and 3.6b. The origin of the stereo camera setup is defined as the center of the breast. This can set to the origin of the HoloLens using the Marker which it can recognize.

3.3.4 Resulting accuracy

To measure the resulting accuracy, the difference between the virtual and actual needle position was measured. This was done using the NDI aurora, as the accuracy of the stereo camera setup is not yet documented. Also, due to time mismanagement testing using the stereo camera setup was not possible. The distance was measured along multiple axes as can be seen in Figure 3.7. The application was started up 5 times and calibrated 5 times to measure the meas-

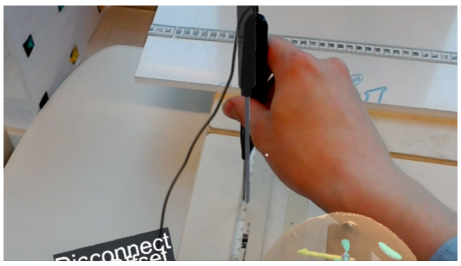


(a) The axes from the virtual world.

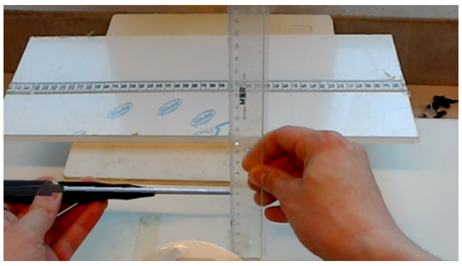


(b) The axes from the measurements of the NDI Aurora overlayed after the Transformation seen in Figure 3.4b.

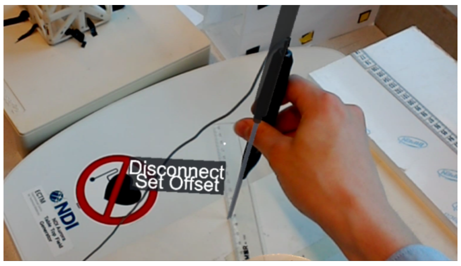
Figure 3.6: Lime: y-axis, red: x-axis and turquoise: z-axis.



(a) measurement along the y axis, visible are the sensor(real world) and the virtual needle(virtual world. Image taken using Mixed Reality capture.



(b) measurement along the z axis. Image taken using Mixed Reality capture.



(c) measurement along the x axis. Image taken using Mixed Reality capture.

Figure 3.7: Lime: y-axis, red: x-axis and turquoise: z-axis.

Table 3.1: 5 times the system was calibrated using the by placing the marker at the origin, connecting to the measurement setup. And resetting the needle position. Afterwards the error from the real needle position was measured using a ruler.

	measure 1	measure 2	measure 3	measure 4	measure 4	average
x	3 mm	1 mm	6 mm	4 mm	2 mm	3.2 mm
y	6 mm	8 mm	15 mm	15 mm	12 mm	11.2 mm
z	9 mm	8 mm	12 mm	10 mm	14 mm	10.6 mm
average	6 mm	5.7mm	11 mm	9.7 mm	6 mm	8.3 mm

urement error. The results can be seen in Table 3.1. Average deviation from the actual point in one axis was 8.3 mm, the accuracy is not sufficient to hit lesions as small as 2 mm.

3.3.5 Resulting ease of use

Setting up the connections and the right coordinate frame still take quite some time, this only needs to be done once until the device is turned of. When the system is eventually all set up, the user can see the locations of the lesions very easily. The best place to insert the needle can be chosen intuitively.

4 Conclusion

Even though not all goals were met, the prototype shows it is possible to create an application that could assist during a biopsy. The system did make it easier to comprehend the positions and size of the lesions compared to using an ultrasound probe. As the presented information was 3 dimensional, it is very intuitive to comprehend. Another advantage was the position of the information, the user is looking where the information is while performing the biopsy, thus does not need to avert his or her attention to a monitor and back. Even though the device itself was not accurate enough to track the objects in its surrounding such as the needle, it was able to communicate with other devices which were much more accurate.
The produced prototype shown is not as accurate as desired, but further exploration in tracking should be performed. Because this prototype shows it is possible develop applications that could assist in performing biopsies, if the accuracy of the device is increased.
Concerning the various stakeholders; the cost of the diagnosis will not be decreased by removing the Ultrasound probing, because MRI is now needed for all patients. Previously, a combination of mammography and ultrasound were used which is less costly even when combined. Meaning hospitals will not benefit from the current system. Radiologist however, will have an easier time finding the lesions. Resulting in a shorter time for the the Biopsy gun being inside the patient. This will give a better experience for the patients as well.

5 Discussion

5.1 Tracking system

A more powerful and faster tracking system could be used. Of the two devices used, the NDI Aurora and the stereo camera setup, only the stereo camera setup could currently also track the breast position rotation and deformation. Which will be crucial information to show for the physician to always be able to reach the lesion in one time. Furthermore, the camera of the HoloLens was still used to do some tracking which was not as accurate as desired. Another method could be to add a better 3d tracking device to the HoloLens. This could be done as by M. Garon et al. (4).

5.2 Origin finding

The origin tracking could be improved, by adding a needle holder attached to the marker. This would ensure the HoloLens could know the real world position and rotation and virtual world position and rotation. The holder would ensure the offset is the same and can be adjusted for more accurately. If this system would not suffice, another option would be to use a tracking system which can also track the position of the HoloLens. To do this a tracking system with a much larger working space must be used, while still achieving the same accuracy. This removes the inaccuracy of using the HoloLens camera for tracking. A third option would be to use a calibration which could take longer with a bigger marker where the user could be used to define the location of multiple positions on this marker as in (2).

5.3 Breast deformation

In the current system, breast deformation is not yet implemented in the HoloLens, while the breast deforms a lot when a needle is inserted, or the patient is in a slightly different position. In Unity the vertices of the scan can be accessed to change the shape according to the measured deformation.

5.4 Different imaging technique

Instead of taking an MRI beforehand and using this to guide the physician during the biopsy, US might be used. US is the current used method for find the lesions. With the tracking system in place the US probe can itself be tracked. Then the gathered US images could be streamed to the HoloLens instead of a computer screen. The HoloLens could, using the location of the probe, show the stream in the location it is being taken. This would greatly increase the ease of use of US assisted biopsy. The radiologist would have a way easier time interpreting the US image as it shows the insides at the exact location. Using US has the benefit of lower cost compared to MRI.

5.5 Loading new scans

No method of loading scans while the program is active is implemented. This would be a useful functionality for a future iteration of the application. In unity assets can be created which can be loaded from a server. To load a new scan into the HoloLens, it would need to run an 'asset-bundle manager', which can retrieve assets, such as the scan from a server. The scan needs to be turned in to a Unity assetbundle asset, meaning that when a new scan is received it needs to be segmented, loaded into unity, turned into an assetbundle asset and put on server where it can be reached by the HoloLens.

A Appendix 1

A.1 MATLAB server code

```
function SendStringOverTcp(mode, input_matrix)
% This receives two arguments: (string mode, matrix matrix_to_send)
% The first time this funciton is called it starts a server and waits for
    a
% connection. After that it will send matrix_to_send as a coded string to
% the client which connected. When mode is 'close', it will close the
% connection. Otherwise, the mode is send as an identifier to the client.
% The matrix will be stringified from the left column going down, then the
% next column. See index of matrix for matlab.

%% create the connection the first time
persistent t;
if(isempty(t) && strcmp( mode,'close') ~= 1 )
    fprintf('Waiting for a connection...\n');
    t = tcpip('0.0.0.0', 4012, 'NetworkRole', 'server');
    fopen(t);
    fprintf('Connected to ');
    fprintf(t.RemoteHost);
    fprintf('\n');
end
%% finish if asked
if(~isempty(t) && strcmp( mode,'close'))
    fprintf('Closing connection...\n');
    fclose(t);
    delete(t);
    fprintf('Closed connection\n');
    return;
elseif(strcmp( mode,'close'))
    fprintf('No connection open.\n');
    return;
end
%% process input
sizes = size(input_matrix);
new_string = '';
for n = 1:(sizes(1)*sizes(2))
    new_string = strcat(new_string, sprintf('%5f', input_matrix(n)), ', ')
        ;
end
str2send = [mode,',',new_string,'\n'];
%% send data
fprintf(t, str2send);

%fprintf('Sent data\n');
return;
end
```

A.2 TCPIP client

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.IO;
using UnityEngine;
using UnityEngine.Events;

#if !UNITY_EDITOR
using System.Threading.Tasks;
using Windows.Networking;
using Windows.Networking.Sockets;
#endif

/// <summary>
/// This is a client script that will connect to a server which will give
    it information on the locations of objects.
/// The output is handled by UwpTcpIpClientHandles.cs.
///
/// This script does not connect from the unity editor, as it only runs in
     Universal Windows Platform. A script that connects in the editor can
    be found in TcpIpClientAsync.cs.
/// </summary>
public class UwpTcpIpClient : MonoBehaviour {
#if !UNITY_EDITOR
    private StreamSocket socket;
    private HostName remote_host;
    private Task<bool> reading = null;
#endif
    /// <summary>
    /// Wether the script is currently connected to a server.
    /// </summary>
    public bool Connected { get; private set; }
    /// <summary>
    /// IpAddress or URL
    /// </summary>
    public string remote_host_name;
    public string remote_port = "4012";
    /// <summary>
    /// Invoked when a message is received
    /// </summary>
    public UnityEvent message_received;
    /// <summary>
    /// The String received from the server.
    /// </summary>
    public string response = "No data received";
    private static bool ready_to_read;
    public TextMesh textmesh;

    void Start (){
        Connected = false;
        try
        {
            if (String.IsNullOrEmpty(remote_host_name))
            {
                remote_host_name = "localhost"; // default for test
                    application
                Debug.LogError("No remote_host_name to connect to,
                    defaulting to:" + remote_host_name);
            }
            else
            {
                //Debug.LogError("remote_host_name is:" + remote_host_name
                    );
            }
            if (message_received == null)// create the unityevent if
                necessary
            {
                message_received = new UnityEvent();
            }

        }catch(Exception e)
        {
            Debug.LogError("Start of UwpTcpIpClient failed with error: " +
                 e);
        }
    }

        void Update ()
    {
#if !UNITY_EDITOR
        try
        {
            if (Connected)// If we have a connection and should read, we
                have to start the process or keep it running
            {


                if (reading == null)
                {
                    ReadSocket();
                }
                else if (ready_to_read)
                {
                    ReadSocket();
                    ready_to_read = false;
                }

            }

        }catch(Exception e)
        {
            Debug.LogError("Update of UwpTcpIpClient failed with error: "
                + e);
            textmesh.text = "Update of UwpTcpIpClient failed with error: "
                 + e;
        }
#endif
    }
    private void OnDisable()
    {
#if !UNITY_EDITOR
        socket.Dispose();
#endif
    }

#if !UNITY_EDITOR
    /// <summary>
    /// Connect to the server specified by the remote_host_name object on
        port remote_port
    /// </summary>
    public async void Connect()
    {
        try
        {
            if (!Connected)
            {
                socket = new StreamSocket();
                remote_host = new HostName(remote_host_name);
                //textmesh.text = "starting the connect await";
                await socket.ConnectAsync(remote_host, remote_port);
                //textmesh.text = "await finished, connected";

                Connected = true;
            }
        }
        catch(Exception e)
        {
            Connected = false;
            Debug.LogError("failed to connect! Error message:" + e);
        }
    }
    /// <summary>
    /// Listen to the socket and invoke signal if a text line was received,
    /// </summary>
    public async void ReadSocket()
    {
        try
        {
            Stream stream_in = socket.InputStream.AsStreamForRead();
            StreamReader reader = new StreamReader(stream_in);
            response = await reader.ReadLineAsync();
            //Debug.LogError("response is:'" + response + "'");
            if (!String.IsNullOrWhiteSpace(response))
            {
                message_received.Invoke();
            }
            ready_to_read = true;
        }
        catch(Exception e)
        {
            Debug.LogError("Failed to read data with error: " + e);
            textmesh.text = "Failed to read data with error: " + e;
        }
    }

#else
    /// <summary>
    /// Dummy connect function for Unity editor
    /// </summary>
    public void Connect()
    { Debug.Log("the wrong connect, you are in Unity_editor");
        textmesh.text = "the wrong connect, you are in Unity_editor";
        Connected = true;
    }
    /// <summary>
    /// Dummy Readsocket function for Unity editor
    /// </summary>
    public void ReadSocket()
    { Debug.Log("the wrong ReadSocket");}
#endif
    /// <summary>
    /// Disconnects from the server
    /// </summary>
    public void Disconnect()
    {
        try
        {
#if !UNITY_EDITOR
            socket.Dispose();
#endif
            Connected = false;
        }catch(Exception e)
        {
            Debug.LogError("socket.Dispose() faild with error: " + e);
        }
    }
}
```

A.3 TCPIP client handler

```
using System;
//using System.Collections;
//using System.Collections.Generic;
using UnityEngine;
/// <summary>
/// This class interprets the data received by UwpTcpIpClient.cs.
/// </summary>
public class UwpTcpIpClientHandler : MonoBehaviour {
    /// <summary>
    /// The UwpTcpIpClient this script is handling.
    /// </summary>
    [Tooltip("The UwpTcpIpClient this script is handling.")]
    public UwpTcpIpClient tcpip_client;
    [Tooltip("This will be moved when a $NEEDLE message is received")]
    public Transform needle_spheres;
    [Tooltip("This will be moved when a $AURORA message is received")]
    public Transform needle_AUR;
    [Tooltip("This will be moved when a $BREAST message is received")]
    public Transform breast;
    [Tooltip("Origin of the measurement setup")]
    public Transform measurement_origin;
    public  CopyHolograms ch;
    private Vector3 needle_pos;// position the needle will be set to.
    private Quaternion needle_rot;// rotation the needle will be set to.
    private Matrix4x4 breast_matrix;// Location and rotation of the breast
         described in homogeneous matrix.
    // Flags for when new data is received.
    private bool new_NEEDLE_spheres= false;
    private bool new_BREAST_TFORM = false;
    private bool new_NEEDLE_aur = false;

    void Start () {

        try
        {
            if(tcpip_client == null)
            {
                gameObject.GetComponent<UwpTcpIpClient>();
            }
            if(breast == null)
            {
                breast = ch.Assembly.transform;
            }
            breast_matrix = Matrix4x4.identity;

            needle_pos = new Vector3();
        }
        catch(Exception e)
        {
            Debug.LogError("Start of UwpTcpIpClientHanler failed with
                error: " + e);
        }
        }
        /// <summary>
    /// Gets called once per frame and sets the objects to there latest
        positions if data is received.
    /// </summary>
        void Update ()
    {
        if (breast == null)
        {
            breast = ch.Assembly.transform;
        }
        if (new_NEEDLE_spheres){
            SetNeedleSpheresTransform(needle_pos, needle_rot);
            new_NEEDLE_spheres = false;
        }
        if (new_NEEDLE_aur)
        {
            SetNeedleAurTransform(needle_pos, needle_rot);
            new_NEEDLE_aur = false;
        }
        if (new_BREAST_TFORM)
        {
            //SetBreastTransform(breast_matrix);
            new_BREAST_TFORM = false;
        }
        //Debug.LogError("Breast_Transform is: (pos  ;  rot)->(" + breast.
            position + "  ;  " + breast.rotation.eulerAngles + ")");
    }

    /// <summary>
    /// Activated by UwpTcpIpClient when it has a new message that should
        be decoded.
    /// </summary>
    public void DecodeMessage()
    {
        try
        {
            string[] values = tcpip_client.response.Split(',');// The
                values are in a string and seperated by a ',' and need to
                be taken apart to be read.
            if (values[0] == "$NEEDLE")// Decode needle message
            {
                Vector3 sphere1 = new Vector3();
                Vector3 sphere2 = new Vector3();
                try
                {
                    // decode the values into floats in Vectors.
                    sphere1 = new Vector3((float)Convert.ToDouble(values
                        [1]) / 1000, (float)Convert.ToDouble(values[2]) /
                        1000, (float)Convert.ToDouble(values[3]) / 1000);
                    sphere2 = new Vector3((float)Convert.ToDouble(values
                        [4]) / 1000, (float)Convert.ToDouble(values[5]) /
                        1000, (float)Convert.ToDouble(values[6]) / 1000);
                    needle_pos = sphere1;
                }
                catch (Exception e)
                {
                    Debug.LogError("Creating locations failed with error "
                         + e);
                }
                Vector3 heading = sphere1 - sphere2;
                needle_rot = Quaternion.LookRotation(heading);
                new_NEEDLE_spheres = true;
            }
            else if (values[0] == "$BREAST_TFORM")// decode Breast message
            {
                for(int i = 0; i < 4; i++)
                {
                    // Convert from text to float and put in the matrix.
                    breast_matrix.SetRow(i, new Vector4((float)Convert.
                        ToDouble(values[1+(i*4)]) , (float)Convert.
                        ToDouble(values[2 + (i * 4)]) , (float)Convert.
                        ToDouble(values[3 + (i * 4)]) , (float)Convert.
                        ToDouble(values[4 + (i * 4)])));

                }
                breast_matrix.SetColumn(3, new Vector4(breast_matrix.m03 /
                     1000, breast_matrix.m13 / 1000, breast_matrix.m23 /
                    1000, breast_matrix.m33));// Adjust translation from
                    mm to meter scale.
                new_BREAST_TFORM = true;
            }
            else if(values[0] == "$AURORA")// Decode tabletop NDI Aurora
                message.
            {
                needle_pos = new Vector3((float)Convert.ToDouble(values
                    [1]) / 1000, (float)Convert.ToDouble(values[2]) /
                    1000, (float)Convert.ToDouble(values[3]) / 1000);
                needle_rot = new Quaternion((float)Convert.ToDouble(values
                    [5]), (float)Convert.ToDouble(values[6]), (float)
                    Convert.ToDouble(values[7]), (float)Convert.ToDouble(
                    values[4]));
                new_NEEDLE_aur = true;
            }
        }
        catch(Exception e)
        {
            Debug.LogError("DecodeMessage failed with error: " + e);
        }
        }
        /// <summary>
        /// Translates the location and rotation of the needle transform.
        /// </summary>
        /// <param name="position"> New position the needle should be set to
            .</param>
        /// <param name="rotation"> New rotation the needle should be set to
            .</param>
        void SetNeedleSpheresTransform(Vector3 position, Quaternion rotation)
        {
            try
            {
                if (!needle_spheres.parent.gameObject.activeSelf)
                {
                    needle_spheres.parent.gameObject.SetActive(true);
                }
                needle_spheres.localRotation = rotation;
                needle_spheres.localPosition = position;
            }catch(Exception e)
            {
                Debug.LogError("SetNeedleSpheresTransform(Vector3 position,
                    Quaternion rotation) failed with error: " + e);
            }
        }
        /// <summary>
        /// Set the Transform of the needle in the aurora  according.
        /// </summary>
        /// <param name="position"></param>
        /// <param name="rotation"></param>
        void SetNeedleAurTransform(Vector3 position, Quaternion rotation)
        {
            try
            {
                if (!needle_AUR.parent.gameObject.activeSelf)
                {
                    needle_AUR.parent.gameObject.SetActive(true);
                }
                needle_AUR.localRotation = rotation;
                needle_AUR.localPosition = position;

            }
            catch (Exception e)
            {
                Debug.LogError("SetNeedleAurTransform(Vector3 position,
                    Quaternion rotation) failed with error: " + e);
            }
        }
        /// <summary>
        /// Translates the location and rotation of the breast transform.
        /// </summary>
        /// <param name="matrix"> Transformation matrix containing information
            , for the position and rotation.</param>
        void SetBreastTransform(Matrix4x4 matrix)
        {
            try
            {
                Vector3 start_pos = new Vector3(0, 0, 1);
                Vector3 pos = Matrix4x4Utilities.ExtractTranslationFromMatrix(
                    ref breast_matrix);
                Quaternion qua = Matrix4x4Utilities.ExtractRotationFromMatrix(
                    ref breast_matrix);

                breast.localPosition = new Vector3(pos.x, pos.y, pos.z) +
                    start_pos;// Rotation is needed to correct for coordinate
                    frame differences
                breast.rotation =  qua;
            }
            catch (Exception e)
            {
                Debug.LogError("SetBreastTransform(Matrix4x4 matrix) failed
                    with error: " + e);
            }
        }
}
```

Bibliography

[1] Berg, W. A., L. Gutierrez, M. S. NessAiver, W. B. Carter, M. Bhargavan, R. S. Lewis and O. B. Ioffe (2004), Diagnostic Accuracy of Mammography, Clinical Examination, US, and MR Imaging in Preoperative Assessment of Breast Cancer, **vol. 233**, no.3, pp. 830–849, doi:10.1148/radiol.2333031484, pMID: 15486214.
`https://doi.org/10.1148/radiol.2333031484`

[2] Birkfellner, W., M. Figl, K. Huber, F. Watzinger, F. Wanschitz, J. Hummel, R. Hanel, W. Greimel, P. Homolka, R. Ewers and H. Bergmann (2002), A head-mounted operating binocular for augmented reality visualization in medicine - design and initial evaluation, **vol. 21**, no.8, pp. 991–997, ISSN 0278-0062, doi:10.1109/TMI.2002.803099.

[3] Boetes, C., R. D. Mus, R. Holland, J. O. Barentsz, S. P. Strijk, T. Wobbes, J. H. Hendriks and S. H. Ruys (1995), Breast tumors: comparative accuracy of MR imaging relative to mammography and US for demonstrating extent., **vol. 197**, no.3, pp. 743–747, doi:10.1148/radiology.197.3.7480749, pMID: 7480749.
`https://doi.org/10.1148/radiology.197.3.7480749`

[4] Garon, M., P. O. Boulet, J. P. Doironz, L. Beaulieu and J. F. Lalonde (2016), Real-Time High Resolution 3D Data on the HoloLens, in *2016 IEEE International Symposium on Mixed and Augmented Reality (ISMAR-Adjunct)*, pp. 189–191, doi:10.1109/ISMAR-Adjunct.2016.0073.

[5] van Krevelen, D. (2007), Augmented Reality: Technologies, Applications, and Limitations, doi:10.13140/RG.2.1.1874.7929.
`https://www.researchgate.net/profile/Rick_Van_Krevelen2/publication/292150312_Augmented_Reality_Technologies_Applications_and_Limitations/links/56ab2b4108aed5a01359c113.pdf`

[6] MATLAB (2016), *version 9.0.0.341360 (R2016a)*, The MathWorks Inc., Natick, Massachusetts.

[Microsoft] Microsoft (), HoloLens hardware detail.
`https://developer.microsoft.com/en-us/windows/mixed-reality/hololens_hardware_details`

[Microsoft] Microsoft (), Microsoft/HoloToolkit-Unity.
`https://github.com/Microsoft/HoloToolkit-Unity`

[NDI] NDI (), Medical » Aurora.
`https://www.ndigital.com/medical/products/aurora/`

[10] Qian, L., E. Azimi, P. Kazanzides and N. Navab (2017), Comprehensive Tracker Based Display Calibration for Holographic Optical See-Through Head-Mounted Display.

[11] Sharp, T., C. Keskin, D. Robertson, J. Taylor, J. Shotton, D. Kim, C. Rhemann, I. Leichter, A. Vinnikov, Y. Wei, D. Freedman, E. Krupka, A. Fitzgibbon, S. Izadi and P. Kohli (2015), Accurate, Robust, and Flexible Real-time Hand Tracking, ACM, pp. 3633–3642, ISBN 978-1-4503-3145-6.
`https://www.microsoft.com/en-us/research/publication/accurate-robust-and-flexible-real-time-hand-tracking/`