The contribution of grayscale cameras to the accuracy of surgical augmented reality goggles

Lesley Franschman
Supervisor(s): Pierre Ambrosini , Ricardo Guerra Marroquim
EEMCS, Delft University of Technology, The Netherlands

A Dissertation Submitted to EEMCS faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering

Abstract

The way that surgeons currently use surgical navigation technology impacts their hand-eye coordination and their ability to view images and data critically. To tackle this issue Augmented Reality goggles, built from the HoloLens 2, have been developed specifically for the purpose of aiding surgeons during surgery and it is called: The HoloNav. Optical reflective spheres are commonly used to track surgical instruments. This research aims to find out if grayscale cameras could contribute to locating these spheres more accurately with the HoloNav. This is done by finding the spheres on the infrared images, reducing the search space on the grayscale images and finding the spheres on the reduced search spaces of the grayscale images. By triangulating those retrieved coordinates and comparing them with the optically tracked coordinates a conclusion can be drawn about the accuracy. The accuracy measured in this research is between 0.66 mm and 10.64 mm (mostly between 1.8 mm and 6.4 mm) depending on the frame. This is quite accurate and similar to the results from related work. With better image quality or different input conditions the accuracy could be improved even more.

1 Introduction

Finding the right surgical site is imperative for surgeons when performing surgery. This exact location of where to use the surgical tools can be found with surgical navigation technology [1]. Surgical navigation usually makes use of optical or electromagnetic sensors which outputs data and images to a screen that the surgeon can use as a map for guidance during the surgery. Surgical navigation allows the surgeon to see instrument positions relative to the preoperative imaging made by, for example, a CT scan.

The downside of this technology is that the surgeon has to switch focus between the screen and the surgical site continuously throughout the surgery. This affects the hand-eye coordination of the surgeon. It also makes it more difficult for the surgeon to view the sensor output imagery critically. Overcoming these challenges can be done with the use of augmented reality [2]. Basic functionalities and components for such an augmented reality surgical navigation system can be build using the HoloLens 2 [3] and it is called: The HoloNav.

The Accuracy in terms of image-to-patient registration and optical tracking needs to be optimal in order for the HoloNav to be used reliably during surgeries. That is why the main question this research aims to answer is: Should the HoloNav make use of stereo grayscale cameras combined with the HoloLens 2 infrared sensor to make the 3D optical tracking more accurate? By trying to answer this question it will be clear to see how much the stereo grayscale cameras contribute to improving the accuracy compared to the optical tracking methods used in related researches while possibly bringing the HoloNav closer to its optimal accuracy.

In order to answer this main research question in a step by step manner, this question is divided into the following three sub-questions:

1. How can the 2D location of the optical reflective spheres be found on the grayscale images with the use of an infrared image?
2. How can those locations on the grayscale images be used to find the desired 3D location of the optical spheres?
3. How can the accuracy be measured and compared?

This paper will firstly present related work and an introduction to the HoloNav sensors and their different coordinate systems. Subsequently, an extensive explanation is given of how to find the 3D coordinates of the optical reflective spheres with the use of grayscale and infrared images. This will be presented first by introducing the concepts of Blob Detection and line projection and by explaining how this has been used in the first stage of the 3D coordinate extraction. Afterwards, the second stage is explained. In this stage, the search space on the grayscale images is reduced to a line. Once those lines are found, spheres need to be located on the grayscale images using image processing techniques, this is the third stage. Triangulation is the fourth and final stage of the 3D coordinate extraction. After the section on triangulation, an evaluation is performed based on accuracy comparisons with tracking methods from related work and the results can be found after this evaluation. Lastly, a theoretical discussion on the results is held right before concluding this paper, discussing possible future work and analysing the reproducibility.

2 Related Work

Some research on the HoloLens 2 and its accuracy has already been conducted. The Erasmus Medical Centre, for example, has done research on the use of augmented reality (AR) during neurosurgery which proved that the use of AR in the operating room is clinically feasible [4]. However, they also concluded that the accuracy of this surgical navigation tool needs to be improved.

The work by Gaxner et al.[5] shows what accuracy can be achieved with just the grayscale cameras of the HoloLens 2. Their method of finding the 3D sphere locations is based on stereo vision techniques and a single-constraint-at-a-time extended Kalman filter (SCAAT-EFK). Both are executed in separate but complementary pipelines. An accuracy of 1.7 ± 0.81 mm was reported when both pipelines were utilised. When only the stereo vision techniques were used, accuracy of 8.07 ± 0.91 mm was reported.

Besides research on the HoloLens grayscale cameras, the HoloLens long and short-throw infrared illuminators and the depth camera have also been experimented with. This has been done by Kunz et al.[6]. They reported accuracy as low as 0.76 mm by using these sensors in combination with an extra infrared light source mounted on the HoloLens 2. This result is achieved by binarizing the short-throw reflectivity frame before applying Blob Detection to find the spheres. Once those are found, the depth frames are used to extract the corresponding 3D locations of the spheres. The movement of the spheres was extremely controlled. A robotic Stewart platform, called the Hexapod, was used for this.

3 The Cameras and Coordinate Systems

As can be seen in Figure 1, the HoloLens 2 is equipped with the following cameras: multiple grayscale cameras, long and short-throw infrared illuminators to make infrared images, an infrared depth camera and a colour video camera.

Figure 1: The HoloNav and its cameras.

The HoloLens 2 API can translate The HoloLens 2 camera and corresponding image coordinates into the coordinates of another camera or pixels of their images. Figure 2 contains the complete coordinate system mapping.

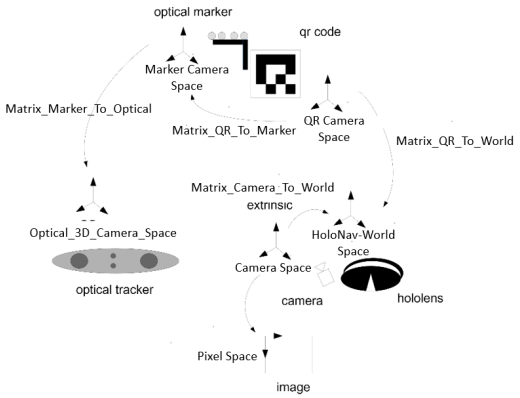

Figure 2: The coordinate mapping between images, their corresponding camera space and HoloNav-world space and the more elaborate mapping from markers to optical and HoloNav-world space through the use of the QR-codes in the images.

One of the sub-systems from this coordinate mapping used the most in this research are the mappings between pixel, camera and HoloNav-world space to translate infrared coordinates to grayscale coordinates via infrared camera space. Another important sub-mapping is the mappings between markers, optical and HoloNav-world space to compare the extracted 3D sphere coordinates with the optically tracked coordinates in HoloNav-world space.

4 Finding the 3D coordinates of the spheres

This section will provide a detailed overview of how the 3D coordinates of optical reflective objects can be found in the HoloNav-world space with stereo grayscale and infrared images and the OpenCV library. The four stages of this process are visualised in Figure 3. OpenCV is the open-source library that contains numerous computer vision and image processing algorithms that proved quite useful during this research [7].

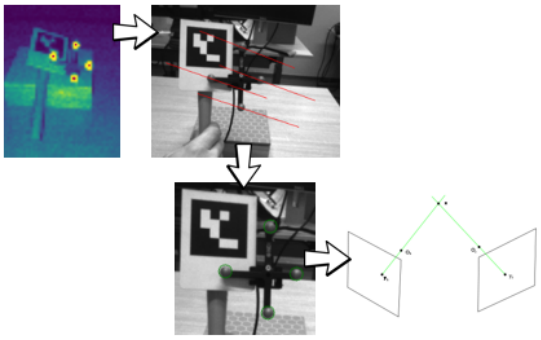

Figure 3: Stage 1: Blob Detection, Stage 2: Reducing search space, Stage 3: Find the spheres on the grayscale images, Stage 4: Triangulation.

4.1 Blob Detection

The very first step in retrieving the 3D sphere coordinates is finding the spheres on an infrared image. Figure 4 contains an example of the kind of infrared images used during this research. It can be seen clearly here that the spheres are the brightest objects. This is the main reason why optical reflective spheres are commonly utilised for surgical navigation.

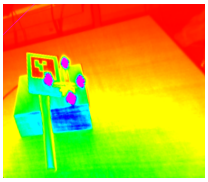

Figure 4: Infrared image of optical reflective spheres.

Extracting the image coordinates of the spheres can be done with Blob Detection. Blob Detection is a computer vision method that can detect regions in an image that differs in properties, such as colour or brightness, compared to surrounding regions.

OpenCV simpleBlobDetector() does the thresholding, grouping, merging of nearby blobs, the centre calculations and radius calculations. The resulting blobs can be filtered by colour, size and shape. This can be done by setting the parameters and passing them as input to simpleBlobDetector().

This method has been tested on multiple infrared images and it was not always able to find all spheres. According to fellow research team member Omar Hussein, this is because of the lighting conditions as OpenCV's Blob Detection is light sensitive. After adding an offset using Numpy, this light sensitivity gets countered and simpleBlobDetector() works much better.

Another OpenCV blob extraction algorithm that seemed to yield good results without offsetting was connectedCompo-

nentsWithStats() [8]. This method solves the problem of finding parts of the image that are connected physically, irrespective of colour. This omits the need to set parameters yourself. By sorting the output of this function, the blobs belonging to the spheres can be retrieved as they are the largest blobs beyond a certain threshold in the images like the one in Figure 4. Besides returning the blob centroids, this method is also able to return the bounding boxes and the pixel areas of the blobs.

For this research simpleBlobDetector() was used for compatibility reasons as this is what fellow research team member Omar Hussein is using after extensive research [9].

4.2 Reducing the search space

Now that the pixel coordinates of the reflective spheres on the infrared image can be found, a line can be projected on the grayscale images which serve as a reduced search space to look for the spheres. To perform this projection there are several things needed:

1. The 2D pixel coordinates of the spheres in the infrared image need to be found.
2. The depth-line through the centre of that sphere needs to be found.
3. For each point on the depth-line, find the corresponding pixel on the left and right grayscale images.

In the previous section it is explained how the sphere coordinates can be found on the infrared image . When the sphere coordinates on the infrared images are found, they have a z-value of 1 (in an (x, y, z) coordinate). This coordinate can be brought to different depths by simply multiplying the coordinate by some value (t) and adding that result to the origin (0, 0, 0) as it follows the formula: 3D-point = Origin + t * (infrared coordinate - Origin). This formula is visualised in Figure 5.

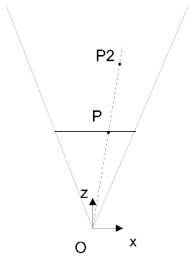

Figure 5: The depth-line through a sphere calculated by finding multiple points on this line using the formula: p2 = O + t * OP for multiple values of 't'.

All points for different values of 't' lie in a line in 3D camera space. During this research, the values 0-2000 were used as 't'. When this 3D line is projected on the grayscale images, the result is a line in 2 dimensional space that passes through the centre of the corresponding sphere. Finding this line would drastically decrease the search space for the sphere as it is now known that a specific sphere is located somewhere on this line. However, it is still unknown where exactly this sphere is located on this line. The next section will provide an explanation on how the exact sphere location is retrieved. The 3D line and the 2D projected lines are depicted in Figure 6 as blue lines.

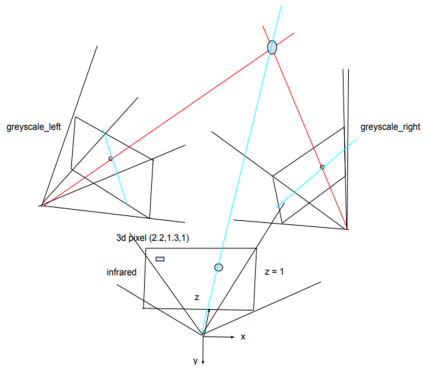

Figure 6: An infrared image and stereo grayscale images oriented in camera space and HoloNav-world space. The blue lines correspond to the 3D depth-line and the reduced search spaces. The red lines correspond to the triangulation.

In order to project the 3D line onto the grayscale images, there are two kinds of matrices needed: the intrinsic and extrinsic matrices of the infrared and grayscale cameras. The intrinsic matrices are used for bringing pixel coordinates to camera space coordinates and back. The extrinsic matrices are used for translating camera space coordinates to HoloNav-world space and back. Projecting the infrared coordinates can now be done by taking 2D infrared pixel coordinates to 3D HoloNav-world space via 3D infrared camera space and then bringing those resulting HoloNav-world space coordinate to 2D grayscale image space (left and right) via 3D grayscale camera space.

The API of the HoloLens provides the properties and position of the HoloLens 2 cameras. Based on this API, the matrices can be retrieved. A look-up table is used instead of an intrinsic matrix as this was already provided by the supervisors at the start of this research project. This look-up table maps a coordinate in an image to a 3D point in camera space. It is in the same data set as where the images are stored. A representation of the matrix mapping can be seen in Figure 7. This figure is a subsystem of the coordinate mapping depicted in Figure 2 in Section 3.

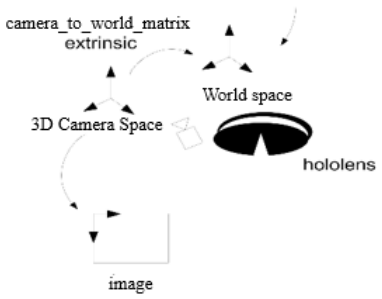

Figure 7: Matrix mapping between image, camera and HoloNav-world space.

4.3 Finding the grayscale coordinates

After the search space on the grayscale images has been reduced to a line per sphere, image processing techniques are needed to find the spheres. Several of those techniques have been utilised and compared during this research to find the best image processing method in terms of accuracy and processing time.

OpenCV Circular Hough Transform is often used when trying to extract circular objects from images. It searches an entire image and uses the gradient information of edges to find circles. The problem with using this method is that the sphere radiuses need to be estimated before execution. This is difficult to do as the scales of the spheres differ per image set. No useful results were able to be obtained using this method. According to Raymond K.K. Yip, Peter K.S. Tam and Dennis N.K. Leung, the circular Hough Transform is limited by slow speed and excessive memory [10]. This was another reason to look for better options.

Another image processing technique that was tested is Edge Detection. The idea was to find the edges of the spheres present in the images and then look for where the reduced search space line intersects with the spheres. Two kinds of edge detection algorithms were tested: Sobel and Canny. The Prewitt algorithm was omitted because, according to research conducted by Mamta Joshi and Ashutosh Vyas [11], Prewitt finds fewer edges and makes edges less visible compared to the Sobel filter.

The Sobel filter works by calculating the gradient of image intensity at each pixel. Based on how abruptly or smoothly the image pixels change from light to dark, pixels that represent edges can be found. By doing this the filter is also able to estimate the orientation of the edge. This method is especially useful when speed is of the essence but not when it has to deal with noisy images [12].

The Canny algorithm is a multi-stage algorithm that produces smoother edges but is less time-efficient. This algorithm consists of five stages [13].

1. Apply Gaussian filter to reduce noise.
2. Mark pixels where gradients of the image have large magnitudes.
3. Preserve the sharpest gradients and discards the rest.
4. Thresholding.
5. Suppress edges that are not connected with strong edges.

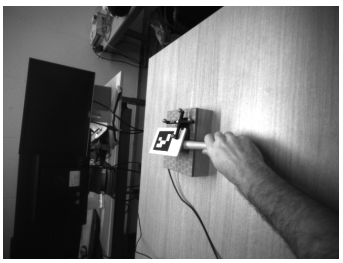

Figure 8: Image before Edge Detection.

The results from applying the Sobel and Canny Edge Detection on the image in Figure 8, can be found in Figure 9. It can be seen that even though Canny Edge Detection yields clearer visuals, the contours of the spheres are not sharp enough with either method.

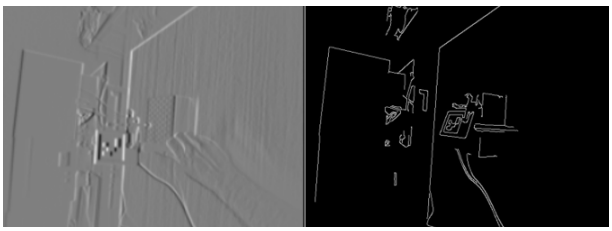

Figure 9: Left: Sobel Edge Detection, Right: Canny Edge Detection.

The next image processing method that has been tested was OpenCV Template Matching. Template Matching is a digital image processing technique that aims to find parts of an image that matches a certain template image. Since most of the spheres in the images look similar, this method seemed like a good candidate as only a few templates would suffice. However, this method has some drawbacks that are difficult to overcome. It is not rotation or scale-invariant, it is light sensitive and background changes can cause inaccurate results.

All of these issues need to be overcome for the available images. The corners of the template show some background pixels because the used templates are images of circular objects. These background differences cause inaccuracies if these templates remain unaltered as the background for each sphere differs. An example of an unprocessed template can be seen in the most left image of Figure 10.

Figure 10: An unprocessed sphere template on the left and three examples of processed templates.

This background hurdle can be overcome by dropping the template images in paint, deleting the pixels in the corners using the selection tool and saving the new template with an alpha channel as a PNG. The alpha channel is needed to pass a mask to the pattern matching function. It can be extracted from the template by splitting up the image into different channels. In case no alpha channel is available, an opaque substitute is created using the Numpy "ones" function. Examples of processed templates can also be found in Figure 10.

The second obstacle that needed to be handled was the scale differences between the images. This can be tackled by making more templates. The more templates there are, the more accurate the Template Matching results will be. But as the algorithm has to check the Template Matching result for every template in search for the best match everytime the algorithm is run, the processing time will increase drastically when more templates are added. That is why it is preferred to only add the necessary templates. One way to limit the number of templates was to find a solution for the light sensitivity.

There is some shininess present on the spheres as also can

be seen in Figure 10. This shininess causes the need for separate templates for the left and the right images. By rotating both frames and making sure that the highlight on the spheres on the left and right grayscale frames point in the same direction, the total amount of templates needed is halved.

After preparing the templates, the images themselves needed to be processed. This is needed because Template Matching searches an entire image while the aim is to search only the lines on which a sphere is known to be located. A mask is used to remove any result which falls outside lines. The mask is created by initialising a zero image and then drawing the lines on it. The thickness of the mask lines allows for some margin of error in the matching.

After the mask has been prepared, the Template Matching function is called. This function returns a grayscale image where each pixel denotes how much the neighbourhood of that pixel matches with the applied templates. The region with the most correlation is the matched region. The specific pixel that was matched is the centre of that region. A matching score value is calculated for each available template. The template that results in the highest value decides the final matched region.

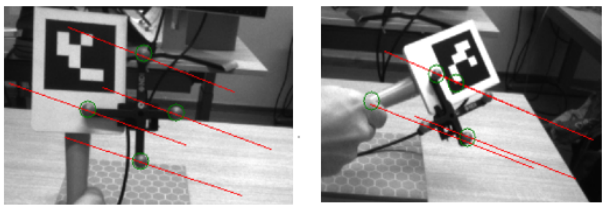

Figure 11: The reduced search spaces are represented by red lines and the Template Matching results are visualised by green circles. The left image is an example where all the spheres are correctly matched. On the right image two of the four spheres are mismatched.

For visualisation and verification purposes, there are circles drawn around the matched pixels. The aim is to have these circles perfectly aligned around the spheres. An example can be seen in the left image of Figure 11. In case it is noticed that for some images the matching is less accurate, a decision can be made to add more templates to increase accuracy. In the right image of Figure 11, there are, for example, two spheres that are not matched correctly. This happens because there are no suitable templates for them made and the existing templates are not compatible. This causes another point on the search line to be a better match.

4.4 Triangulation

Once the coordinates of the centre of the spheres have been found on the left and the right grayscale images, the 3D coordinates of those spheres can be calculated with the use of triangulation.

Before the triangulation can be applied, the centre coordinates of the spheres need to be translated to HoloNav-world coordinates. As explained in Section 4.2 this can be done by finding the intrinsic and extrinsic matrices of the cameras and then by multiplying with the intrinsic one before multiplying by the extrinsic one. This needs to be done for both the left and the right spheres.

The principle of triangulation is that a line can be drawn through the corrsponding spheres of the left and right grayscale images in 3D space, just like has been visualized in Figure 6 by red lines. The point where those lines intersect is the 3D coordinate of the corresponding sphere in HoloNav-world space.

One point by itself does not give enough information to draw a line. For every sphere, a second point on the line needs to be known besides the 3D world coordinate. Luckily, the world space origin points for both left and right grayscale images can easily be calculated by multiplying the origin coordinate [0,0,0,1] with the extrinsic matrix of the corresponding camera. With this information, the lines can be drawn in 3D camera space and the intersection can be found.

5 Experimental setup and results

The method that has been used to retrieve the 3D coordinates of the spheres during this research has been explained thoroughly in the previous section. To find out how accurate this method is it needs to be compared to the ground truth. The ground truth contains coordinates that are located by an optical tracker. The distances between the method's triangulated coordinates and the ground truth are then compared to results from related work like, for example, the work of Kunz[6] and Gsaxner[5].

The method used in this research needs to be applied to a large number of image sets. One image set consists of a left and a right grayscale image and an infrared image with corresponding timestamps. The correspondence between the different images within an image set and the optical tracker information can be kept track of with timestamps. A timestamp contains the date and time an image was created and it gets attached to the image. If, for example, the timestamp of the left grayscale image is taken as reference, the corresponding right grayscale image, infrared image and optical tracking information can be found by searching for the ones that have a timestamp closest to the timestamp of the left grayscale image.

After the correct optical tracking data has been found, the optical markers can be translated to optical tracking space. By making use of the QR code position these optical tracking coordinates can in turn be translated to the world space where the retrieved 3D coordinates and the found optical coordinates can be compared. The difference between those coordinates can be expressed in root mean squared errors (RMSE). The RMSE is calculated by taking the sum of the sphere distances of all well-matched spheres in an image set and dividing it by the amount of well-matched spheres in the set. For each image set, only the spheres with a measured distance smaller than the sphere radius (11.5 mm) are considered to be well matched by the Template Matching.

This has been applied to 600 image sets in this research. Data sets with a timestamp difference bigger than 25 milliseconds are skipped. Another parameter used in this experiment was an OpenCV line-thickness of 10. As mentioned in Section 4.3, this line-thickness serves as a margin of error when

applying the Template Matching mask. The results of the experiment can be found in Figure 12. In this figure, it can be seen that the distances between the optically tracked coordinates and the triangulated coordinates are between 0.66 mm and 10.64 mm.

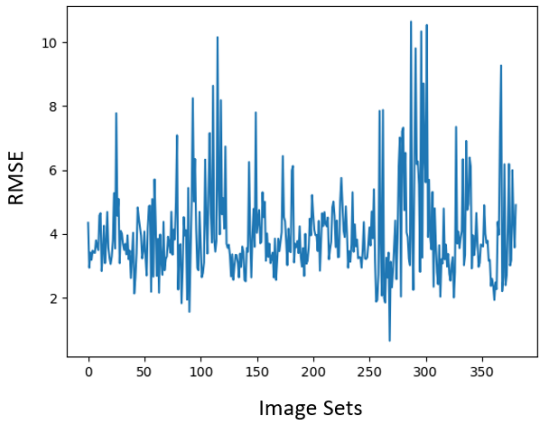

Figure 12: Distances between the triangulated coordinates and the optically tracked coordinates expressed RMSE per image set.

In Figure 12 can also be seen that the data of only 400-420 image sets are reported while 600 image sets were used. This is because those image sets were skipped as the timestamps of the frames within the image sets differed too much from the reference timestamp. The timestamp of the left grayscale image was taken as a reference.

In Figure 13 can be seen that the resulting RMSE values are between 0.66 mm and 10.64 mm and that it is mostly between 1.8 mm and 6.4 mm with a total mean value of 4.07 mm.

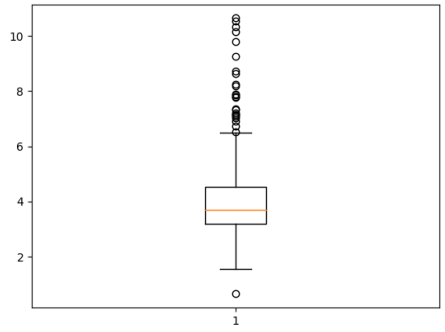

Figure 13: A box plot of the RMSE for every image set. Only spheres with a resulting distance smaller than the sphere radius were taken into consideration. Other spheres would be considered not well-matched by the Template Matching.

The outlier results are caused by not well-matched Template Matching results. Figure 14 contains the left and right grayscale images of the image set with the highest RMSE value (10.64mm). The spheres in the left image are all well-matched while two of the spheres in the right grayscale image are not well-matched. Even though the left sphere of the right frame is not well-matched it is very close to the sphere. When triangulating this sphere and comparing it to the optically tracked coordinate, it results in a distance just a bit smaller than the sphere radius (11.5 mm) so it influences the RMSE of that image set.

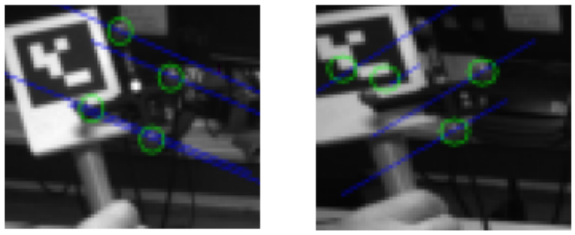

Figure 14: The left and right grayscale images are used as an example of an outlier point created by not well-matched Template Matching results. These frames belong to the image set with the highest RMSE found.

The amount of well-matched spheres differ per image set. Figure 15 shows that for most of the image sets, all spheres are found. For the other image sets where not all four spheres are found, at least one sphere is still located. However, for a few image sets, there are no spheres found. Better or more templates for the Template Matching function to use could increase the number of spheres found.

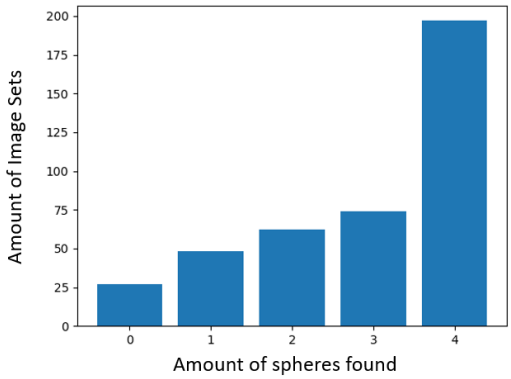

Figure 15: This figure shows how often a certain amount of spheres are found. As there is a maximum of four spheres to be found per image set, every image set can locate 0-4 spheres.

6 Discussion

Based on the results from Section 5 it can be seen that the method used during this research can give very accurate results. It is capable of getting similar and sometimes even more accurate results than the methods reported in the work of Kunz et al. and Gsaxner et al. This can be concluded from the fact that the most accurate result from the method used in this research is a distance of 0.66 mm, while the work from Kunz et al. and Gsaxner et al. report accuracy's of 0.76

mm and 1.7 ± 0.81 mm respectively. The method shows especially a lot of promise since it was able to achieve these results without the special conditions used in related work like extra lighting, a second pipeline (SCAAT-EFK) and controlled movement. These conditions are mentioned more elaborately in Section 2. Making use of these extra conditions could increase accuracy even more.

There are a couple of aspects of the applied method that might have room for improvement. In some of the image sets, one of the grayscale images does not have all spheres visible. This could cause the Blob Detection algorithm to choose another point in that image that is not a sphere as the fourth blob which in turn causes inaccuracy.

Another possible improvement point can be the amount and the quality of the templates used for the Template Matching. As already mentioned in Section 4.3, twenty templates have been made to make the Template Matching algorithm more accurate. However, as the data set is quite large, there are certain image sets for which templates have not specifically been made and have not been manually verified to work with one of the already existing templates. A larger set of templates would also probably remove some of the outlier points mentioned in the previous section. Even though Adding more templates could cause more spheres to be found accurately, it would also make the processing time slower.

It can be seen in Figure 15 that most of the image sets can triangulate at least one sphere accurately. The relative positions between the spheres themselves could be used to find all the sphere positions based on the position of only one or two spheres. Doing this would increase the overall number of spheres found as this would mean that the method does not have to rely as heavily on the Template Matching as it does at the moment anymore.

There are still a lot of paths to be researched to find further improvements.

7 Conclusion

In conclusion, the HoloNav should probably make use of stereo grayscale cameras combined with the HoloLens 2 infrared sensor to make the 3D optical tracking more accurate because this method shows accurate results. The results are roughly similar compared to the results from related work, even though that related work made use of more favourable extra conditions. The resulting accuracy depends on the image sets and their corresponding Template Matching results. The better the Template Matching result, the higher the accuracy.

The spheres on the infrared images were retrieved accurately and those sphere locations were successfully used to reduce the search space on the infrared images. Extracting the spheres from the grayscale images using Template Matching was however less accurate than preferred. As already mentioned in Section 4.3, there were some obstacles to overcome at this stage of the research due to background differences, light sensitivity and scale differences. As discussed in Section 6, more and/or different templates as input for the Template Matching algorithm could cause the algorithm to locate more spheres in total across all the image sets. It could also cause higher accuracy among the spheres that the algorithm was already able to find with the current templates.

Triangulation returned expected results and the results from comparing the triangulated points with the optical tracker coordinates were quite accurate. From this can be concluded that the image processing aspect of this research has the most room for improvement.

8 Future Work

There has been concluded that there is room for improvements in the method used in this research. This means there is reason to conduct follow-up research. Three aspects that seem to be worth focusing on are improving the current sphere locating method by adding and/or changing templates, making use of the geometrical relation between sphere locations and the conditions used in related work like, for example, extra lighting and better motion control.

Especially the second aspect should be looked into. Finding at least one sphere has a higher success rate than finding all four spheres. The geometric relation between the sphere locations could be exploited to find all spheres based on only one or two spheres extracted from a grayscale image. As finding only one sphere per grayscale image can be done much more accurately, there is a bigger chance that this would result in more accurate triangulated 3D sphere coordinates when taking the position relative to the other spheres into account. It would also mean that the method does not have to rely on Template Matching as much as it currently does.

9 Reproducibility

Results in this paper have been generated using Python, OpenCV, standard libraries like Pandas, Numpy and Matplotlib, a large set of grayscale and infrared images and a look-up table to translate pixels to 3D camera space coordinates.

All of the libraries including OpenCV are easily accessible as they just need to be installed and imported. The code used to find optical reflective spheres with grayscale and infrared images will be made publicly available on GitHub (https://github.com/lfranschman/HoloNav-Grayscale-triangulation). The data set that contains all HoloNav images and the pixel-camera space look-up table can be found in a public google drive folder (https://drive.google.com/file/d/1-TzLZJoLLTDhpGw8KWVPPedtmXc-pf7T/view?usp=sharing). The only thing that needs to be done after downloading the data is manually linking the data set using the data set location on the user's local host. This path needs to be set in the config.py file.

The code also contains other functionalities like methods to translate pixels to HoloNav-world space and back, methods to manipulate images, the coordinates of the markers that are needed to move from and to optical-tracker space and a method to translate optical tracker coordinates to HoloNav-world space.

By using the available code in combination with the available data set, the results presented in this paper can be reproduced.

Acknowledgements

Thanks to Omar Hussein for his experimental results and insights which were an integral part of the Blob Detection aspect of this paper. Thanks to Cécille Franschman for her feedback on this paper. And finally thanks to Pierre Ambrosini and Ricardo Guerra Marroquim for their feedback and for supervising this research project.

References

[1] U. Mezger, C. Jendrewski, M. Bartels, Navigation in surgery, Langenbecks Arch Surg (2013).

[2] M. Benmahdjoub, T. van Walsum, P. van Twisk, E. Wolvius, Augmented reality in craniomaxillofacial surgery: added value and proposed recommendations through a systematic review of the literature, Int J Oral Maxillofac Surg (2020).

[3] S. Park, S. Bokijonov, Y. Choi, Review of Microsoft Hololens Applications over the Past Five Years, Appl. Sci (2021).

[4] F. Incekara, M. Smits, C. Dirven, A. V. Mathis-Ullrich, Clinical Feasibility of a Wearable Mixed-Reality Device in Neurosurgery, World Neurosurgery (2018).

[5] C. Gsaxner, J. Li, A. Pepe, D. Schmalstieg, J. Egger, Inside-out instrument tracking for surgical navigation in augmented reality, VRST '21, Association for Computing Machinery, New York, NY, USA, 2021. doi:10.1145/3489849.3489863.
URL https://doi.org/10.1145/3489849.3489863

[6] C. Kunz, P. Maurer, F. Kees, P. Henrich, C. Marzi, M. Hlaváč, M. Schneider, F. Mathis-Ullrich, Infrared marker tracking with the HoloLens for neurosurgical interventions, Current Directions in Biomedical Engineering (2020).

[7] About OpenCV.
URL https://opencv.org/about/

[8] A. Rosebrock, OpenCv Connected Component Labeling and Analysis (2021).
URL https://pyimagesearch.com/2021/02/22/opencv-connected-component-labeling-and-analysis/

[9] O. Hussein, Accuracy of the Hololens 2's infrared cameras in the context of surgical navigation (2022).

[10] R. K. Yip, P. K. Tam, D. N. Leung, Modification of hough transform for circles and ellipses detection using a 2-dimensional array, Pattern Recognition 25 (9) (1992) 1007–1022. doi:https://doi.org/10.1016/0031-3203(92)90064-P.
URL https://www.sciencedirect.com/science/article/pii/003132039290064P

[11] M. Joshi, A. Vyas, Comparison of Canny edge detector with Sobel and Prewitt edge detector using different image formats, International Journal of Engineering Research Technology (IJERT).

[12] R. Tian, G. Sun, X. Liu, B. Zheng, Sobel Edge Detection Based on Weighted Nuclear Norm Minimization Image Denoising (2021).

[13] L. Ding, A. Goshtasby, On the canny edge detector, Pattern Recognition 34 (3) (2001) 721–725. doi:https://doi.org/10.1016/S0031-3203(00)00023-6.
URL https://www.sciencedirect.com/science/article/pii/S0031320300000236