Using Computer Vision to Track Anatomical Structures During Cochlear Implant Surgery

Nicholas Bach
Washington University in St. Louis

Follow this and additional works at: https://openscholarship.wustl.edu/eng_etds

Part of the Bioimaging and Biomedical Optics Commons

Recommended Citation
https://openscholarship.wustl.edu/eng_etds/581

This Thesis is brought to you for free and open access by the McKelvey School of Engineering at Washington University Open Scholarship. It has been accepted for inclusion in Engineering and Applied Science Theses & Dissertations by an authorized administrator of Washington University Open Scholarship. For more information, please contact digital@wumail.wustl.edu.
Using Computer Vision to Track Anatomical Structures During Cochlear Implant Surgery

by

Nicholas Bach

A thesis presented to the McKelvey School of Engineering of Washington University in partial fulfillment of the requirements for the degree of

Master of Science

August 2021
Saint Louis, Missouri
Acknowledgments

I would like to thank my parents for allowing me to pursue my goals through this master’s degree. I would also like to thank my family for supporting me in all my endeavors.

I would like to thank Dr. Silva for the inspiration and guidance that fueled this master’s thesis.

A special thanks goes to the many graduate students and distinguished faculty within my department who have reviewed this thesis and helped support the related research.

Nicholas Bach

Washington University in Saint Louis
August 2021
Dedicated to my amazing parents.
# Contents

Acknowledgments ................................................................. ii

List of Tables ................................................................. v

List of Figures ................................................................. vi

Abstract ................................................................. vii

1 Introduction ................................................................. 1

2 Methods ................................................................. 2
   2.1 Video Content and Creation ........................................... 2
   2.2 Algorithms and Hyperparameters .................................... 5
      2.2.1 Discriminant Classifier Trackers ............................. 6
      2.2.2 Optical Flow Trackers ......................................... 7
      2.2.3 Lucas-Kanade Flow Algorithms ............................... 7
      2.2.4 Hyperparameters .............................................. 8

3 Results ................................................................. 9
   3.1 Error Comparison ..................................................... 9
   3.2 Speed Comparison ................................................... 14

4 Discussion ................................................................. 15

5 Conclusion ................................................................. 17

References ................................................................. 18

Vita ................................................................. 21
List of Tables

3.1 Algorithm Speeds ................................................. 14
List of Figures

2.1 Parts of a cochlea implant surgery. a) incision, b) mastoidectomy, c) implant of electrode and receiver. Reproduced from Vincenti et al.[24]. ................. 3
2.2 Image showing final results of a cochlear implant surgery. Reproduced from https://www.nidcd.nih.gov/health/cochlear-implants.[1]. ......................... 3
2.3 Histogram of x-axis flow values estimated visually. .............................. 4
2.4 Histogram of y-axis flow values estimated visually. .............................. 4
2.5 Process of creating a synthetic video. a) frame without medical equipment, b) surgical equipment and hands from real video without background, c) single frame of synthetic video. ............................................. 5

3.1 The error values of each algorithm when calculating the error of video 1. . 10
3.2 The error values of each algorithm when calculating the error of video 2. . 10
3.3 The error values of each algorithm when calculating the error of video 3. . 11
3.4 The error values of each algorithm when calculating the error of video 4. . 12
3.5 The error values of each algorithm when calculating the error of video 5. . 13
3.6 The error values of each algorithm when calculating the error of video 6. . 13
ABSTRACT OF THE THESIS

Using Computer Vision to Track Anatomical Structures During Cochlear Implant Surgery

by

Nicholas Bach

Master of Science in Computer Science

Washington University in St. Louis, August 2021

Research Advisor: Doctor Jonathan Silva

There is a steep learning curve for surgeons performing cochlear implant surgeries. We aimed to use computer vision to track anatomical features with the goal of helping surgeons perform cochlear implant surgery without damaging the cochlea. We compared nine algorithms in total, seven object tracking algorithms and two optical flow algorithms utilizing the Lucas-Kanade method, on manually created cochlear implant surgery videos to determine the accuracy associated with each. Compared with eight other algorithms, we observed that an iterative pyramidal implementation of the Lucas-Kanade (IPLK) method, implemented through OpenCV, performed the best. The IPLK method had the lowest error rate on five out of the six videos, with zero error on four. In conclusion, the IPLK method is the most accurate at tracking the locations of the anatomical structures in a video of a cochlear implant surgery. Computer vision may be a novel and valuable tool to improve surgical results.
Chapter 1

Introduction

Cochlear implants are used to treat “severely to profoundly deaf” patients who do not get enough benefit from hearing aids.\[8\] The surgical procedure has a steep learning curve, however, and can lead to significant complications if performed incorrectly.\[9, 21\] The possibility of complications is compounded by the fact that the surgeon cannot see any anatomical structures of the cochlea, aside from the round window, during surgery.\[6, 16, 18, 23\] We therefore hypothesized that using a computer vision approach to help visualize the location of the cochlea would be beneficial in decreasing surgical complications.
Chapter 2

Methods

2.1 Video Content and Creation

A cochlear implant surgery consists of three parts, an incision behind the ear, a mastoidectomy up to the round window, the opening joining the middle ear and the cochlea, and the insertion of the electrode and receiver, as shown in Figure 2.1. The electrode is inserted through the round window, and is connected to a receiver which is placed behind the ear below the skin. A transmitter, speech processor, and microphone are placed above the skin; when sound waves come into contact with the microphone they are converted into electric current by the speech processor and travel to the transmitter. The transmitter sends this electric current to the receiver, which sends that current down the electrode and into the cochlea. The electrode stimulates part of the cochlea, sending impulses to the brain that the brain interprets as sound. The final result is shown in Figure 2.2.

We used the insertion step of the cochlear implant surgery for tracking. Once the surgeon has completed the mastoidectomy, they have a view of the round window, which we wanted to track over the course of the video in order to track the three dimensional (3D) location of the cochlea. We therefore proposed to assess whether object tracking algorithms would be able to correctly track the round window in videos obtained during real surgical procedures.

When using the real surgical videos, we didn’t know the exact movement of the aural structures in the videos. This meant we didn’t know the optical flow we were looking for from our algorithms, and so we didn’t have a gold standard to compare them to. We tried to identify a single point’s movement across the entire video visually, but doing this with adequate
accuracy proved to be impossible. Therefore, we decided to create synthetic videos, because we could control the movement and thus would know what flow values were correct. We took a frame from each video without any medical equipment, sometimes combining two frames to remove the equipment, and then moved that image around to create our videos. We used our hand tracking estimates to approximate a distribution that our created videos should follow. Figures 2.3 and 2.4 shows that the movement of the videos was roughly equivalent to a normal distribution with mean zero and standard deviation 1.5 in both the x and y directions.
Figure 2.3: Histogram of x-axis flow values estimated visually.

Figure 2.4: Histogram of y-axis flow values estimated visually.
We made the synthetic videos more realistic by adding the medical equipment or other obstruction from each actual video back into our synthetic videos using Adobe Photoshop. This process is depicted in Figure 2.5. We made six videos of various levels of difficulty for the trackers, increasing in difficulty as the video number increases. We consider videos 1-3 to be the most realistic and fair videos, as they depict the medical equipment and surgeon’s hands covering a small amount of the screen for most of the video. Video 4 had the added difficulty of containing a large window, about a fourth of the screen, from another video photoshopped in, giving the tracker a larger and more consistent set of pixels to track with different flow values than the true target. Video 5 is very similar to video 4, however the part of the video that we wanted to track was out of focus while the window from the other video was in focus. This sometimes occurred in the real videos, as they would automatically focus on the medical equipment in the foreground and the background would be out of focus. Finally, video 6 was the most difficult because we chose a portion of a video where the surgeon’s hands blocked either the entire frame or almost the entire frame for a large period of time. We chose to include this video because it occurred in a real-life surgery and we wanted to test our algorithms against it, but realistically no optical flow algorithm would be able to realistically track something that is completely occluded by another object.

2.2 Algorithms and Hyperparameters

We used seven of OpenCV’s tracking algorithms as well as two other optical flow algorithms in order to track the location of the round window. The trackers could be split into two categories: those that use discriminant classifiers and those that use optical flow.
2.2.1 Discriminant Classifier Trackers

In computer vision a single discriminant classifier will pick a particular feature of an image and use that feature to choose between different classifications for the image. All of the discriminant classifiers used by the trackers try to classify regions of the image as either the object to be tracked or the background. The boosting tracker is a simple implementation of Adaboost, combining many of these weak discriminant classifiers into one strong one.[10] The Multiple Instance Learning (MIL) tracker is an improvement on the boosting algorithm, as it uses positive and negatively classified bags with multiple examples in each bag instead of singular positive and negative examples. A bag will contain multiple neighboring examples, and a positive bag has at least one positively classified example. Using bags instead of single examples allows the classifier to adapt better to partial occlusions and results in higher accuracy.[3] The Minimum Output Sum of Squared Error (MOSSE) tracker uses a special adaptive correlation filter called a MOSSE filter, which is effective at producing stable correlation filters. This filter makes the tracker robust to changes in the tracking object’s lighting, scale, and rotation, as well as allowing the tracker to be incredibly fast.[4] The Kernelized Correlation Filter (KCF) tracker was developed to allow for more negative examples to be used. Discriminant classifiers require positive and negative examples, but there are many more negative examples than there are positive examples because the object is expected to take up only a small part of the background. This results in a larger number of negative examples being available to the algorithms than they can use while still maintaining real-time speeds. The KCF tracker utilizes the properties of a circulant matrix in the Fourier domain to add many more negative examples at no extra runtime.[11, 12] The Channel and Spatial Reliability Tracker (CSRT) introduces a spatial reliability map to the discriminative process, allowing for the tracker to focus on areas best suited for tracking. This allows for easier discrimination between the object and its background and allows the tracker to track irregularly shaped regions (the KCF tracker only tracks rectangular regions due to using a circulant matrix).[2]
2.2.2 Optical Flow Trackers

The optical flow trackers are more conventional trackers as they aim to calculate the flow directly instead of indirectly through classifiers. The median flow tracker calculates the Lucas-Kanade optical flow of all points and then returns the median value in the x and y directions.[14] The Tracking, Learning, and Detection (TLD) tracker is a more advanced version of this; it has the same tracking step as the median flow tracker but it has additional steps to learn what the image looks like and detect it in future images. This should make the algorithm more robust to partial occlusion.[15]

2.2.3 Lucas-Kanade Flow Algorithms

The proposed Lucas-Kanade flow algorithms are different from the trackers, since they use the optical flow of pixels within and without the round window. The proposed global Lucas-Kanade (GLK) algorithm calculates the optical flow of every pixel in the image using Lucas-Kanade optical flow, and then returns the mode of those values in the x and y direction.[17] The proposed iterative pyramidal implementation of the Lucas-Kanade (IPLK) method, implemented through OpenCV, uses Shi-Tomasi corner detection algorithm to find the best set of points to track.[22] Shi-Tomasi corner detection finds the points of interest in an image which can be a corner, i.e. the intersection of two edges, but can also be an isolated point of maximal or minimal intensity, line endings, or points along curves with local maximal curvature. By calculating the eigenvalues of the covariance of the partial derivative of the image intensity values with respect to x and y axes, the Shi-Tomasi corner detection quantifies how interesting a point is. The most interesting points are used for tracking, because, by definition, these points have a large gradient of change in multiple directions, thus they are the easiest to track. These points are then tracked using a pyramidal implementation of the Lucas-Kanade algorithm developed by Jean-Yves Bouguet.[5] This algorithm calculates Lucas-Kanade flow using the original image as well as scaled down versions of the image. So, for instance, a pyramidal algorithm with image size 1280x1024 pixels and 2 levels would use the original 1280x1024 image, the scaled down 640x512 image for level 1, and the further scaled down 320x256 image for level 2. Using additional scaled down versions of the image
increases the accuracy of the flow calculation compared to an algorithm that only uses the original size.

### 2.2.4 Hyperparameters

For each of the algorithms we had to decide the values of their hyperparameters. Each of the seven trackers needed a window to begin tracking. We decided to initialize the window to be the area surrounding the round window as closely as possible. The same window was used for all trackers on each video in order to keep them consistent. The GLK algorithm required a window size as its hyperparameter (as the window size increases, accuracy of this algorithm tends to increase up to a point and then plateau). We found 21 pixels to be a good window size to reach the top of the plateau. Finally, the IPLK method had hyperparameters associated with its corner detection as well as its optical flow. For the Shi-Tomasi corner detection algorithm we used at most 100 corners with a minimum quality of 0.3, a minimum distance between each other of 7 pixels, and a block size of 7 pixels. For the pyramidal Lucas-Kanade flow we used a window size of 21 pixels, a maximum of 3 levels, and termination criteria of a maximum of 10 iterations or a minimum accuracy of 0.03. The hyperparameters of the IPLK method were tuned slightly for performance, but many different values resulted in the same performance in all videos. If we were to use many more videos, these hyperparameters could be further tuned for optimal performance.
Chapter 3

Results

3.1 Error Comparison

Running the algorithms on videos 1, 2, and 3 all resulted in similar error values, found in Figures 3.1, 3.2, and 3.3, respectively. The IPLK implementation had the lowest error at 0 in both the x and y directions, meaning it returned the correct flow for every frame. Of the trackers, MOSSE performed the best followed closely behind by Median Flow, which had the second and third lowest errors respectively. The GLK algorithm performed slightly better than the control algorithm, and the TLD tracker performed the worst of all algorithms, even worse than the control algorithm in videos 1 and 2.

The results from video 4 were very similar, shown in Figure 3.4, with the IPLK implementation having zero error and the GLK algorithm as well as the TLD tracker performing particularly poorly. Unlike the previous three videos, though, the MIL tracker performed particular poorly on this video and had the lowest accuracy out of all the other algorithms.

In video 5’s results, shown in Figure 3.5, all the trackers performed equal to or worse than the control. The TLD tracker performed the worst, with over double the control’s error in the y direction and over ten times the control’s error in the x direction. The GLK performed moderately well comparatively, and the IPLK implementation performed very well, with zero error in the x direction and about 0.0006 pixel/frame error in the y direction.

Figure 3.6 show the results from video 6, which ended up being a big outlier when it came to the IPLK implementation. The IPLK implementation had higher error than all algorithms
Figure 3.1: The error values of each algorithm when calculating the error of video 1.

Figure 3.2: The error values of each algorithm when calculating the error of video 2.
other than the TLD tracker and had higher error than the control. The MOSSE tracker had the lowest error for this video, and once again the TLD tracker had the highest.

Of all the algorithms, the TLD tracker performed consistently the worst. This was unexpected, because the TLD tracker was advertised as being very sophisticated and as the best tracker to use in most situations. What makes it different from the other trackers is that it detects what it is tracking and searches the frame for whatever looks the most similar. This also made the tracker worse for our tests, because when the round window would get partially or fully obscured the TLD tracker would find another set of pixels that looked similar to it, and since the objects in the frame were mostly the same color as the round window, it would find those pixels in a random location in the frame. This led the tracker to jumping around erratically at points, and having values of flow larger than expected. Video 4 was the first video that had the round window at least partially obscured for most of the video. While the boosting, median flow, and MOSSE were still able to track the round window well, the other trackers, especially the MIL tracker, had a great deal more trouble tracking the structure than it did in the previous three videos. Since the two Lucas-Kanade algorithms didn’t need vision of the round window to track where it moved, they were unaffected. Video
Figure 3.4: The error values of each algorithm when calculating the error of video 4.

5 had the round window obscured by a greater amount than video 4, and the frame that we wanted to track was out of focus and therefore more difficult to track. This made all the trackers, including the ones that performed well on video 4, to have an increase in error. Even the IPLK implementation was affected and returned the wrong value for a single frame of this video. Video 6 was the most difficult video to track. All of the patient’s ear structures were obscured for a portion of this video when the surgeon’s hands completely covered the screen. In this video, we see that the property of the IPLK implementation that has been benefiting it in all previous videos, the fact that it tracks more than just the round window and uses more areas for its flow calculation hurts it in this circumstance. When the surgeon’s hands cover most of the screen some of the trackers see that they cannot track the round window anymore and return flow values of 0, but the IPLK implementation continues tracking whatever takes up most of the frame. Since the surgeon’s hands are taking up most of the frame, the IPLK implementation begins tracking them and returns large erroneous flow values, as the hands are moving quickly in relation to the background. Since the expected flow values are small, the trackers returning 0 have their error increased by a little while the IPLK implementation has its error increased a large amount by the large values it returns.
Figure 3.5: The error values of each algorithm when calculating the error of video 5.

Figure 3.6: The error values of each algorithm when calculating the error of video 6.
3.2 Speed Comparison

In considering the speed of each algorithm, we expect that the algorithm needs to reach a certain threshold to allow for the other processes to be implemented and maintain real time rendering. This threshold will not be known until the rendering step is implemented, but we know that the algorithm should be at least 30 frames per second. As shown in Table 3.1, we expect the IPLK implementation and the boosting, KCF, median flow, MOSSE, and CSRT trackers to be fast enough. The GLK algorithm as well as the MIL and TLD trackers do not reach this minimal threshold.

Table 3.1: Algorithm Speeds

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Average Speed (FPS)</th>
</tr>
</thead>
<tbody>
<tr>
<td>IPLK</td>
<td>877.1</td>
</tr>
<tr>
<td>GLK</td>
<td>1.4</td>
</tr>
<tr>
<td>Boosting</td>
<td>54.1</td>
</tr>
<tr>
<td>MIL</td>
<td>26.5</td>
</tr>
<tr>
<td>KCF</td>
<td>248.2</td>
</tr>
<tr>
<td>TLD</td>
<td>25.3</td>
</tr>
<tr>
<td>Median Flow</td>
<td>447.4</td>
</tr>
<tr>
<td>MOSSE</td>
<td>3478.7</td>
</tr>
<tr>
<td>CSRT</td>
<td>53.9</td>
</tr>
</tbody>
</table>
Chapter 4

Discussion

Cochlear implant surgeries are technically difficult and have a steep learning curve for the surgeons who perform them.[9] Here we described a method for tracking the round window in real time using computer vision algorithms. We found that, among the nine tested algorithms, the IPLK method performed the best, having almost no error on five of the six videos, and that it was sufficiently fast to track the round window in real time. With the accuracy and speed shown by the IPLK implementation, we believe a real-time 3D rendering of the cochlea would be feasible and precise.

Cochlear implant surgeries can cause trauma to the cochlea leading to greater loss of hearing and injury to the facial nerve.[21] The insertion angle and insertion site of the cochlear implant have been shown to affect the efficacy and safety of the implant.[6, 16, 18, 23] Studies have been performed to explore the efficacy of using additional visual tools to reduce the risk for damage from suboptimal insertion. Microscopes have slowly evolved in ear surgeries with great improvement to safety, going from monocular microscope to binocular microscope[19], with studies showing further improvements with the addition of an endoscope[20] or 3D exoscope[7]. As for computer aided visualization technology, techniques such as optical coherence tomography[13], stereo vision tracking[25], and the da Vinci Surgical System[16] were all shown to have potential in their respective studies. We believe that using a computer vision algorithm to track the anatomical structures and then applying the tracking in a 3D rendering of the cochlea has the potential to result in the greatest technical improvement and greatest ease of use compared with all the strategies previously discussed. Having access to the location, rotation, and shape of the cochlea in real time may be a huge benefit to the surgeon.
The limitations of this study should be acknowledged. The manually created videos looked visually similar to real videos to the human eye, but it was impossible to implement exact reflections, shadows, and other noise into the synthetic videos. In future studies it would be important to create an easily trackable marker drawn or placed on the auricular structure during a real surgery so that they could be used to calculate the true movement of a real surgical video. The algorithms could then be run on the videos, with the pixels making up the markers excluded, to get a more realistic view of their performance.
Chapter 5

Conclusion

In conclusion, cochlear implant surgery is an important treatment for patients with severe or profound deafness, but has a steep learning curve and can lead to significant complications if performed incorrectly. We tested nine computer vision approaches and assessed the accuracy and speed of localization of the round window. We found that the IPLK method performed the best compared to all other algorithms tested. With the accuracy and speed shown by the IPLK method, we believe a real-time 3D rendering of the cochlea would be feasible and precise. We conclude that computer vision represents a novel and valuable tool to improve surgical results.
References


[8] Center for Devices and Radiological Health. What is a cochlear implant?


Vita

Nicholas Bach

Degrees

B.S. Computer Science, May 2020

August 2021

21