AUTONOMOUSLY RETRACTABLE ENDOSCOPE HOLDER FOR OTOLOGIC AND ASSOCIATED PROCEDURES

by

Can Kocabalkanli

A thesis submitted to Johns Hopkins University in conformity with the requirements for the degree of Master of Science and Engineering

Baltimore, Maryland

May 2020

Abstract

For years, endoscopy has been an essential method for providing surgeons improved visualization in difficult areas to access. With recent developments in endoscopic technology, narrower endoscopes have opened new possibilities in assisting surgeons in different procedures. Operations in narrow spaces such as the ear canal are particularly of interest, as smaller endoscopes can significantly reduce the invasiveness of the surgical procedure and improve the field of vision of the surgeon. However, using the endoscope safely inside such delicate parts of the body requires it to be safely held, and safely retracted if the patient moves their head.

To make the most out of endoscopic assistance while preventing any collision inside the ear canal during surgery, an *automatically retracting endoscope holder* has been proposed. Such a system can make the surgery easier and more efficient by allowing the surgeon to operate bimanually and improve patient safety by autonomously retracting if the patient's head moves towards the endoscope in a way that can harm delicate structures down the ear canal. I have developed and implemented such a retraction mechanism with two alternative danger assessment pipelines in this thesis and evaluated the system to ensure it demonstrates the speed, responsiveness, and robustness that the surgical scenario demands. This prototype therefore explores and confirms the feasibility and potential of an automatically retracting endoscope holder, and acts as the first step towards a commercial design that can be widely implemented in operating rooms to make otologic and related procedures easier and safer.

Readers:

Russell H. Taylor, PhD

Francis Creighton, MD

Acknowledgements

I thank my advisors Russell Taylor and Pete Creighton for guiding me through this project.

I would also like to thank Anton Deguet for all his extensive help, and Seena Vafee and Anna Goodridge for their work on the preliminary and next generation hardware implementations, respectively. I thank the Louis B. Thalheimer Fund for Translational Research for funding this research endeavor.

I would finally like to thank my parents and my brother for their never-ending love and support. Thank you for inspiring me to work on helpful things that have the potential to change people's lives for the better.

Contents

List of Figures

1 Introduction

1.1 Thesis Statement

During transcanal ear surgery, a system to hold the endoscope can make the surgery easier and safer by allowing a surgeon to operate bimanually and ensuring safety by retracting the endoscope if the patient has unintentional head motion to avoid trauma to the tympanic membrane and ossicles.

1.2 Clinical Background on Ear Canal Endoscopy [1]

While performing surgery in the ear canal, an operating microscope has been traditionally used to gain visual access to the middle ear, mastoid, and inner ear. This relies on line of sight and given the small aperture of the ear canal often requires the drilling of the mastoid bone to improve visualization and access. With the development of smaller and higher definition, ear surgeons can now access these areas far less invasively using an endoscope without the need for drilling the mastoid. Endoscopes also allow for a wider field of view than a microscope, and angles endoscopes allow the taking of previously unattainable images of pathology. Considering these advantages, endoscopic ear surgery is a rapidly developing field in otology. However, a major disadvantage of this approach is the need to hold the endoscope. This forces the surgeon to operate one handed, while the other hand holds the scope. This makes many technical maneuvers difficult to complete and can increase the time and risk of surgery. An alternative is to use an assistant just to hold the endoscope, or more commonly, to use a stationary endoscope holder.

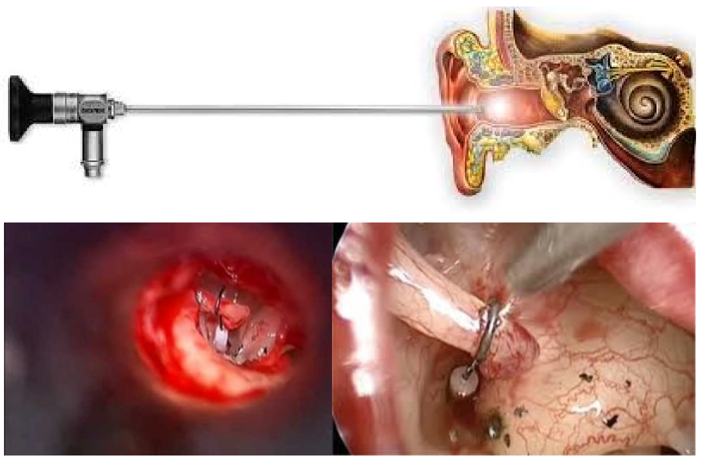


Figure 1: Use of endoscope during ear canal surgery for better imaging (top), and image samples taken with operating microscope (left) endoscope (right) [1]

1.3 Problem Statement & Proposed Solution

The endoscope holder or a robotic arm can be helpful for holding the endoscope tool allowing the surgeon to use both their hands while operating. However, if this holder is stationary the movement of the patient's head towards the endoscope can cause collision inside the ear canal and become dangerous as the endoscope is placed very close (about 2 cm away) to critical structures. Even under anesthesia, many patients have been observed to move their heads in ways that can be harmful since patients cannot be paralyzed for ear procedures [1]. Hence, by using a system that automatically retracts the endoscope when necessary, the surgeon can be allowed to use both their hands during operation; making surgery easier for them and much safer for the patient [2]. Although the focus of this project are ear canal surgeries, such a system can be generalized into other endoscopic or laparoscopic procedures, for example during sinus surgery and colonoscopy.

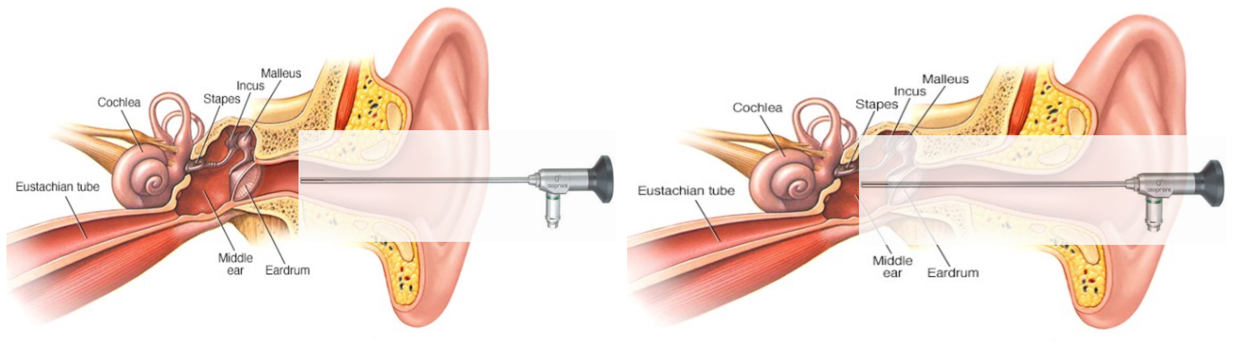


Figure 2: Visualization of the surgical problem [1]

1.4 Objective of Proposed Solution & Research

To develop and test a reliable system integrated with a commercial surgical endoscope that:

1) Detects if the patient's head is moving.

2) Assesses whether the movement is dangerous, filtering unharmful movement such as that caused by the surgeon operating.

3) Swiftly retracts the endoscope when detected movement is deemed dangerous for the patient.

It is also desired but not necessary that the system should not require additional equipment, simply using the images captured by the endoscope to assess movement and determine whether it is dangerous.

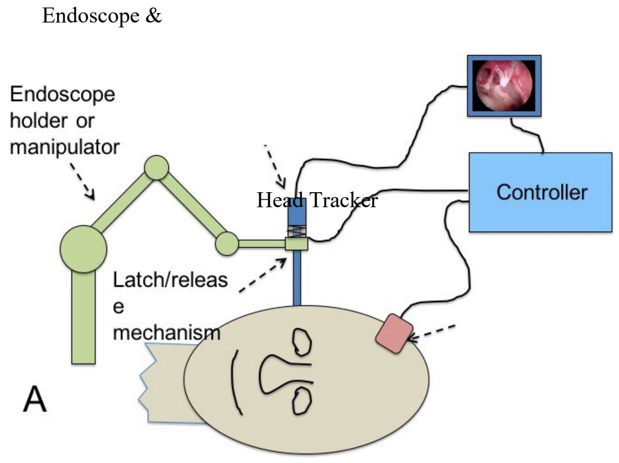


Figure 3: Schematic of Surgical Scenario with Auto-Retraction System Representation [2]

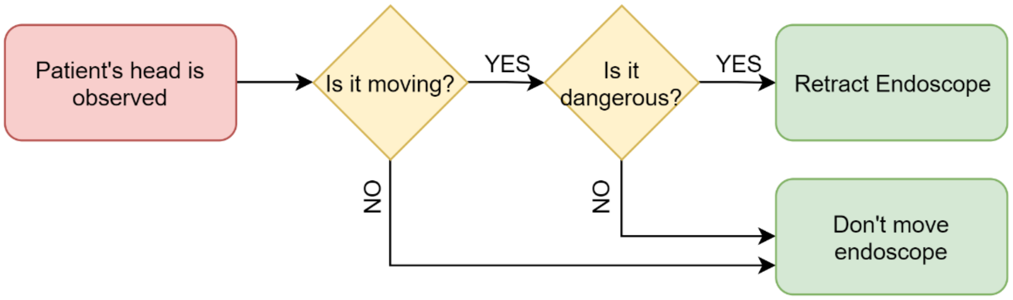


Figure 4: Flowchart representing the high-level basic function of the proposed system

This research implements two alternative sensing methods to address Objectives (1) and (2) as explored under Methods and implements a mechanism to resolve Objective (3) as detailed in Implementation.

2 Methods

There are two alternative methods used in this thesis to detect and assess the movement of the patient's head: One involves the use of an external tracking system in the form of the Northern Digital Instruments (NDI) Electromagnetic Tracking (EM) System to track the position of the endoscope relative to the patient's head, and the other relies on the real-time processing and analysis of the endoscope video to infer the relative movement of the ear canal with respect to the endoscope. The first one is beneficial in its ease of implementation and the second is highly preferred for its elegance and minimal disturbance to the surgical workflow.

2.1 Assessment of Danger through the Electromagnetic Tracking System

The ultimate goal of this project is to accurately detect head motion with no additional equipment to the endoscope itself through the means of computer vision using video taken by the endoscope. However, a system using the external electromagnetic trackers has proven to be a robust and quick method to safely retract the endoscope and therefore has been developed first as a backup option in case the results from the vision system were not sufficient to satisfy the thesis objectives in the given timeframe.

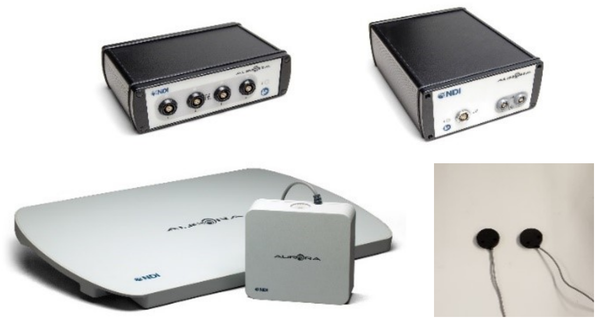


Figure 5: Aurora tracking system components [3]

The NDI system seen in Figure 5 can reliably provide the 3D Cartesian position and orientations of its trackers relative to the system's base coordinates granted that they are above the system's field plate. Empirically, the NDI system has been observed to be capable of providing such data every 0.02 s, or at a rate of 50 Hz with precision of 0.1 mm.[1] The developed system uses the position information from the trackers (or fiducials) to calculate the trackers' relative positions. In the operating room, one fiducial is meant to be rigidly attached to the patient's head (for example onto their forehead) while the other is attached rigidly to the endoscope through an endoscope holder part. This results in a kinematic configuration that can be seen in Figure 6.

[1] Trackers don't always provide data at 50 Hz, at times this rate is observed to be 25 Hz. The precision mentioned here is the last significant figure recorded and displayed by the NDI GUI.

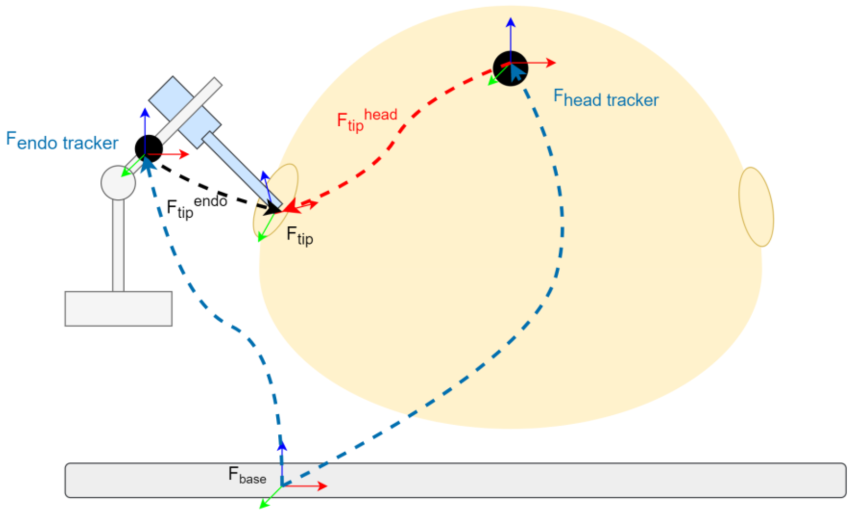


Figure 6: Kinematic Configuration of Retracting System

The blue lines in Figure 6 represent the poses of the head and endoscope trackers with respect to the stationary field plate, which can be represented by SE(3) homogeneous transformation matrices F_{endo} and F_{head} respectively. If the transformation between the endoscope tracker and the endoscope tip, F_{tip}^{endo} ,can be calculated, we can write the relative position of the endoscope tip with respect to the head tracker as:

$$F_{tip}^{\text{head}} = F_{\text{head}}^{-1} F_{endo} F_{tip}^{endo} \tag{1}$$

A standard pivot and axis calibration method can be followed to calculate the transformation F_{tip}^{endo} once the tracker and endoscope are rigidly attached together. The pivot calibration procedure is visualized in Figure 7, and consists of placing the tip of the endoscope in a small dimple in order to fix the tip's position with respect to a reference frame, and rotating the endoscope into different poses without displacing the tip from the dimple. The reference frame is rigidly attached to the body containing the pivot dimple.

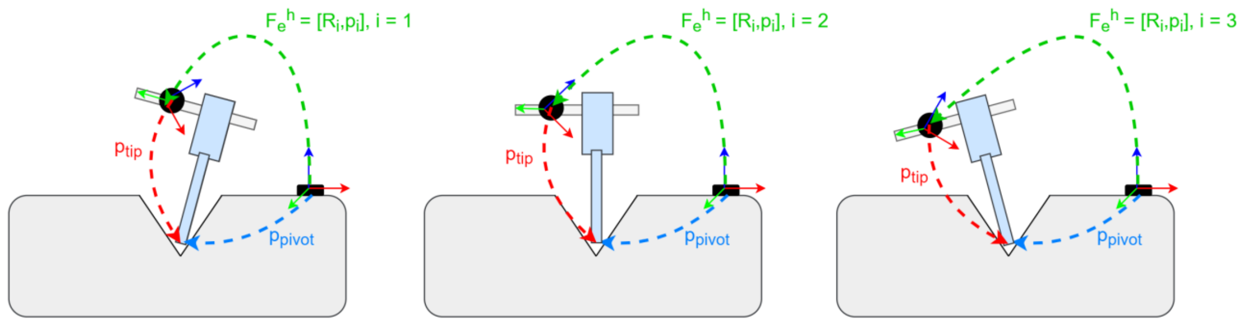

$F_e^h = [R_i, p_i]$, i = 1
p_{tip}
p_{pivot}
$F_e^h = [R_i, p_i]$, i = 2
p_{tip}
p_{pivot}
$F_e^h = [R_i, p_i]$, i = 3
p_{tip}
p_{pivot}

Figure 7: Pivot calibration visualization: closed-loop kinematics for different endoscope poses

The closed-loop kinematics for each endoscope pose can be written as the following for the position of the pivot dimple and endoscope tip:

$$R_i p_{tip} + p_i = p_{pivot} \tag{2}$$

Where R_i is the rotation between the reference tracker and endoscope tracker, p_{tip} is the position of the endoscope tip with respect to the endoscope tracker, and p_{pivot} is the position of the pivot dimple with respect to the reference tracker. We can rearrange Equation (2) as:

$$R_i p_{tip} = p_{pivot} - p_i \tag{3}$$

$$R_i p_{tip} - I_{3x3} p_{pivot} = -p_i \tag{4}$$

Where I_{3x3} is the identity matrix signifying no rotation. If one calculates and records the transformation between the reference and endoscope trackers for at least two poses, we have a system of equations that we can express as the following:

$$\begin{bmatrix} \vdots & \vdots \\ R_i & -I \\ \vdots & \vdots \end{bmatrix} \begin{bmatrix} p_{tip} \\ p_{pivot} \end{bmatrix} \cong \begin{bmatrix} \vdots \\ -p_i \\ \vdots \end{bmatrix} \tag{5}$$

From which solving for p_{tip} and p_{pivot} becomes a least squares problem. One way of acquiring the solution is to perform a Singular Value Decomposition on the matrix defined as:

$$A = \begin{bmatrix} \vdots & \vdots \\ R_i & -I \\ \vdots & \vdots \end{bmatrix} = U \begin{bmatrix} S \\ 0_{(m-n)\times n} \end{bmatrix} V^T \ , \qquad A \in \mathbb{R}^{n\times 6}, U \in \mathbb{R}^{n\times n}, S \in \mathbb{R}^{6\times 6}, V \in \mathbb{R}^{6\times 6} \tag{6}$$

Let $x = \begin{bmatrix} p_{tip} \\ p_{pivot} \end{bmatrix}, p = \begin{bmatrix} \vdots \\ -p_i \\ \vdots \end{bmatrix}$. Then, we can express the problem as $Ax = USV^Tx = p$ (7) and solve:

$$x = {V^T}^{-1} S^{-1} U^{-1} p \tag{8}$$

Since V is an orthogonal matrix,

$$p_{tip}^{endo} = x \ = VS^{-1}U^{-1}p \tag{9}$$

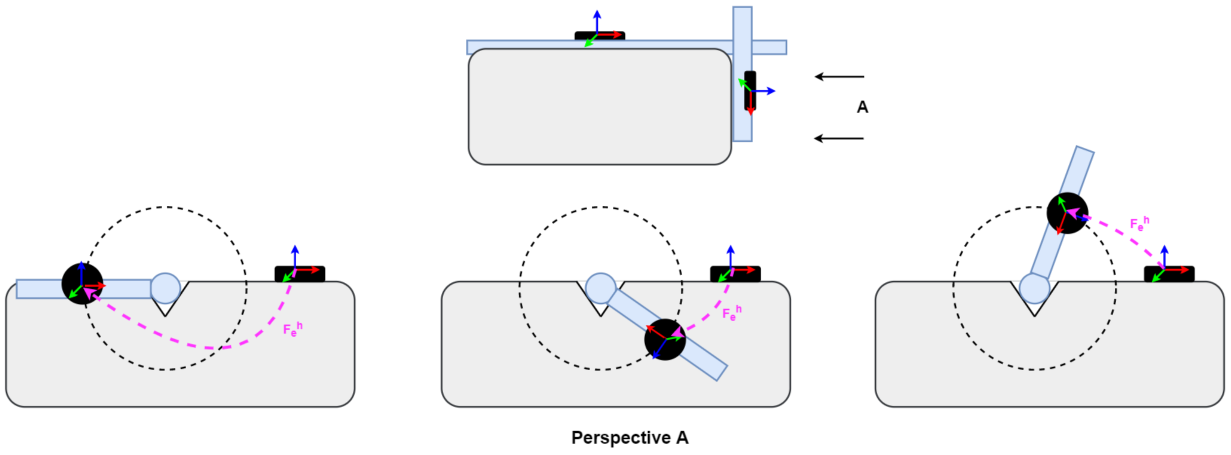


Figure 8: Axis calibration setup side view (top) and data collection procedure (bottom)

This process allows us to know the position of the tool tip with respect to the tool tracker but does not provide information on the direction the endoscope is pointing toward. The directionality of the endoscope can prove to be a useful piece of information in classifying the patient's head motion as dangerous or not, as acceleration in in the direction of the endoscope axis would be the most dangerous to critical ear structures. Hence, we would like to know the direction of this axis, and can calculate it by performing axis calibration

A standard axis calibration procedure consists of placing the endoscope with the tracker in a groove and rotating it so that the tracker moves in a circle as visualized in Figure 8. If the end position is secured, the tracker will move only in a single plane which is easier to operate with. The position data from the tracker can be processed so that the center of the circle is the origin, and then a plane can be fitted to this data. The vector that starts from the origin and is orthogonal to the plane will then be the axis direction that the endoscope is pointing towards, $\boldsymbol{d_{axis}}$. The code and functions used to process the data and fit the plane can be seen in Appendix B, and the circle that the tracker moved in and the axis calculated by this code can be seen in Figure 9.

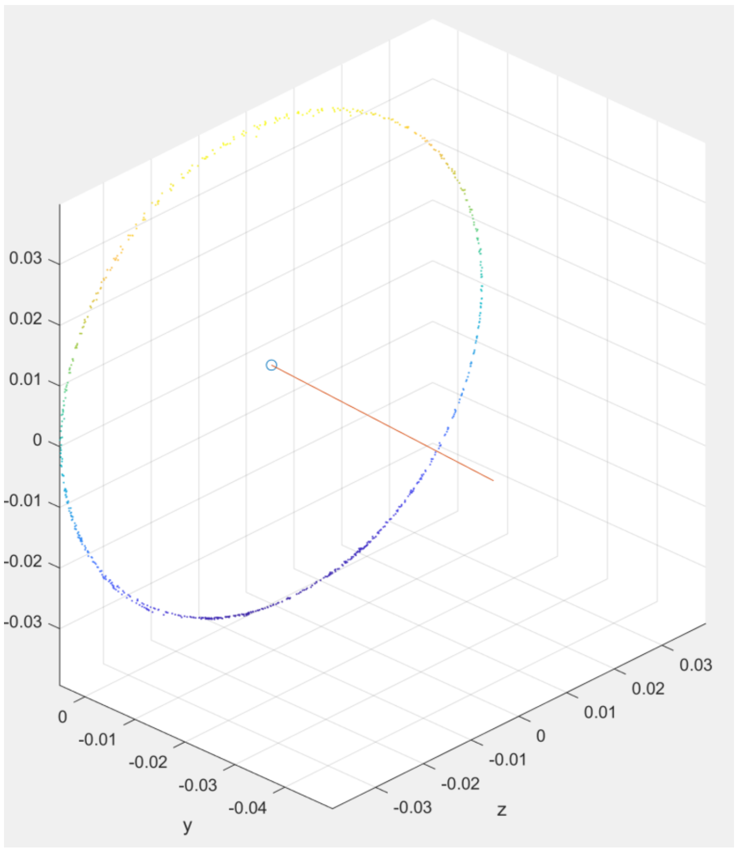


Figure 9: Tracker positions when endoscope is spun around the axis, and the calculated endoscope axis direction

Once both pivot and axis calibration is done, both the position and the direction vector of the endoscope tip is known with respect to the head tracker and tracker base coordinates:

$$p_{tip}^{\mathrm{head}} = F_{\mathrm{head}}^{-1} F_{endo} p_{tip}^{endo} \tag{10}$$

$$d_{axis}^{base} = F_{endo} d_{axis} \tag{11}$$

This allows us to examine the velocity and acceleration of the head tracker in the direction of the endoscope axis in isolation as the projection of the velocity of acceleration on the axis direction vector:

$$v_{axis} = (v_{head} \cdot d_{axis}) d_{axis} \tag{12}$$

In addition to p_{tip}^{head}, the relative movement between critical patient anatomy and the endoscope tip can also be observed. At the time the surgeon sets up the endoscope in the desired position, the target anatomy with respect to the endoscope tracker is defined as:

$$p_{targ,0}^{endo} = p_{tip}^{endo} + \delta d_{axis} \tag{13}$$

Since the endoscope tip position relative to the head tracker is known, the location of the target anatomy with respect to the head tracker can be expressed as:

$$p_{targ}^{head} = F_{\mathrm{head,i}}^{-1} F_{endo} p_{targ}^{endo} = F_{\mathrm{head,i}}^{-1} F_{endo} \left(p_{tip,i}^{endo} + \delta d_{axis} \right) \tag{14}$$

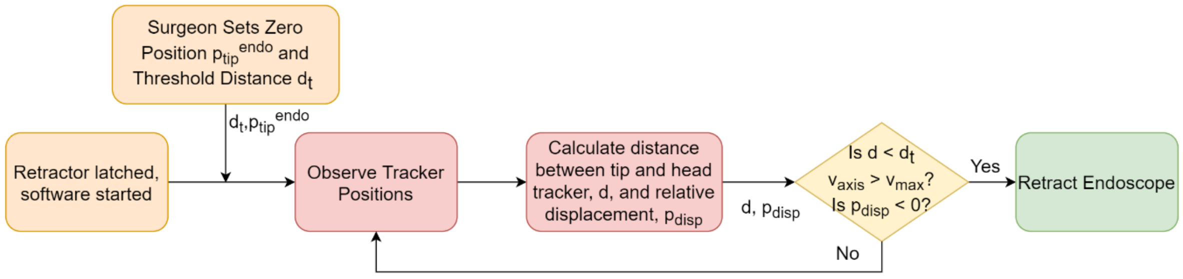
Although the parameter δ is not known, since $p_{targ,i}^{head}$ is a constant transformation, the relative displacement between the target and tip can be calculated with respect to the initial target position at setup as:

$$p_{targ}^{head} = F_{\mathrm{head,0}}^{-1} F_{endo} p_{targ,0}^{endo} = F_{\mathrm{head,i}}^{-1} F_{endo} p_{targ,i}^{endo} \tag{15}$$

$$p_{targ,i}^{endo} = F_{head,i} F_{\mathrm{head,0}}^{-1} p_{targ,0}^{endo} \tag{16}$$

$$\boldsymbol{p}_{\boldsymbol{disp}} = p^{endo}_{targ,0} - p^{endo}_{targ,i} = (I - F_{head,i}F^{-1}_{\text{head},0})p^{endo}_{targ,0} \tag{17}$$

The relative displacement and velocity of the endoscope tip, as well as the magnitudes of the velocity and acceleration in the direction of the axis can be used as metrics to decide whether the endoscope should be retracted or not. For example, once the distance is observed to fall short of a threshold set by the surgeon and the movement is observed to be in the direction of the endoscope, the endoscope is retracted. The surgical workflow for this system can be seen in Figure 10.



Surgeon Sets Zero Position p_{tip}^{endo} and Threshold Distance d_t
d_t,p_{tip}^{endo}
Retractor latched, software started
Observe Tracker Positions
Calculate distance between tip and head tracker, d, and relative displacement, p_{disp}
d, p_{disp}
Is d < d_t
v_{axis} > v_{max}?
Is p_{disp} < 0?
Yes
No
Retract Endoscope

Figure 10: Surgical Workflow with EM System

In this system, the surgeon input (orange) consists of latching the retractor, starting software, and pressing a key on their computer to set distance when endoscope is in desired position. Tracker positions are communicated, and the relative distances, velocities, and accelerations are calculated in real-time by the software, which retracts the endoscope tip once the trigger conditions in the form of relative distance, velocity, or acceleration are met.

2.2 Assessment of Movement and Danger through Endoscope Video

2.2.1 Challenges Unique to a Vision-Based System

The use of endoscope video as the input for determining head (thus ear) motion is highly preferred, as this approach requires no additional hardware or external systems – just the endoscope hardware that is already being used, and a computer with the necessary software environment suffice to implement this approach.

A challenge our system faces is the fact that endoscopes used in ear canal surgery and other narrow-workspace surgeries are most often mono-endoscopes. The lack of a stereo camera means that the depth of objects within the image cannot be directly determined as commonly done by most vision systems. However, there still are several ways to use mono endoscope videos to infer the movement of the head towards the endoscope. One such way the author briefly experimented with was the detection and tracking of landmarks found in the ear canal and their apparent sizes in the frame. However, this approach was quickly abandoned for a few reasons, most important of which being the lack of one consistent landmark throughout the surgical procedure during which the endoscope is expected to be readjusted as the procedure moves further down the ear canal, and also as landmarks drastically change appearance with the introduction of blood and get even harder to track when surgical operating tools introduced into the frame obstruct them. The need for multiple different landmarks throughout the surgery would also require the system to be trained on a set of landmark images to automatically detect them without any human input. I therefore decided to steer away from the detection or tracking of specific objects and proposes a simpler and much more universally applicable way to make use of the endoscope video input.

2.2.2 Using Optical Flow

Optical flow involves the estimation of the apparent movement of each individual pixel between consecutive frames [4] and can be described as the set of trajectories pixels appear to move in over some number of frames. Since the endoscope is rigidly held by the holder and does not move without a retraction command, this apparent movement of pixels can be used to infer the relative movement of the patient’s ear canal with respect to the stationary endoscope camera, without resorting to any object tracking. Without the need to detect and define a set landmarks unique to the ear canal, a robust optical flow based motion estimator becomes an attractive option as it could be used universally not only throughout the entirety of the ear canal surgery, but could also be easily generalized to other uses such as sinus or gastrointestinal procedures.

Using optical flow comes with the assumption that pixel intensities do not change drastically between frames, and that neighboring pixels move similarly. The assumption of constant intensity can be expressed by [5]:

$$I(x,y,t) \approx I(x+\delta x, y+\delta y, t+\ \delta t) \tag{18}$$

Where δx and δy represents the displacement of the pixel (or group of pixels) over time δt. The Taylor Series expansion of this expression yields [5]:

$$I(x,y,t) = I(x,y,t) + \frac{\delta I}{\delta x} \cdot \frac{dx}{dt} + \frac{\delta I}{\delta y} \cdot \frac{dy}{dt} + \delta t \frac{\delta I}{\delta t} + O^2 \tag{19}$$

Where O^2 is the second and higher order partial derivatives, assumed to be negligible. With this assumption, subtracting intensity term $I(x,y,t)$ from both sides and dividing by δt gives us the *optical flow equation*:

$$\frac{\delta I}{\delta x} \cdot \frac{dx}{dt} + \frac{\delta I}{\delta y} \cdot \frac{dy}{dt} + \frac{\delta I}{\delta t} = 0 \tag{20}$$

Or, defining the spatial intensity gradient $\nabla I = (\frac{\delta I}{\delta x}, \frac{\delta I}{\delta y})$ and the optical flow vector as $\boldsymbol{v} = (\frac{dx}{dt}, \frac{dy}{dt})$:

$$\nabla I \cdot \boldsymbol{v} + \frac{\delta I}{\delta t} = 0 \tag{21}$$

This expression, although both ∇I and $\frac{\delta I}{\delta t}$ are known, results in a system of two equations and two unknowns: $\frac{dx}{dt}, \frac{dy}{dt}$. Several methods have therefore been developed to solve for $\boldsymbol{v}$, and one such method is the Lucas-Kanade flow method [6].

Lucas-Kanade Optical Flow

Assuming that the displacement of a pixel between two frames is small and that neighboring pixels will have about the same displacement, the Lucas-Kanade method looks at the motion of a block consisting of the neighboring pixels centered around the pixel one wants to track. Thus, the n points in this block are assumed to have the same motion hence same unknown optical flow $\boldsymbol{v}$ with known ∇I and $\frac{\delta I}{\delta t}$. This results in a system of n equations with two unknowns:

$$\begin{gathered}\frac{\delta I(p_1)}{\delta x}\cdot\frac{dx}{dt}+\frac{\delta I(p_1)}{\delta y}\cdot\frac{dy}{dt}=-\frac{\delta I(p_1)}{\delta t}\\ \frac{\delta I(p_2)}{\delta x}\cdot\frac{dx}{dt}+\frac{\delta I(p_2)}{\delta y}\cdot\frac{dy}{dt}=-\frac{\delta I(p_2)}{\delta t}\\ \vdots\\ \frac{\delta I(p_n)}{\delta x}\cdot\frac{dx}{dt}+\frac{\delta I(p_n)}{\delta y}\cdot\frac{dy}{dt}=-\frac{\delta I(p_n)}{\delta t}\end{gathered} \tag{22}$$

Where $p_1, p_2, \dots, p_n$ are the pixels in the neighborhood, and $\frac{\delta I(p_i)}{\delta x}, \frac{\delta I(p_i)}{\delta y}, \frac{\delta I(p_i)}{\delta t}$ are the intensity partial derivatives from Equation (14). A least squares solution can be fitted to this system to solve for $\boldsymbol{v}$:

$$A=\begin{bmatrix}\delta_x I(p_1) & \delta_y I(p_1)\\ \delta_x I(p_2) & \delta_y I(p_2)\\ \vdots & \vdots\\ \delta_x I(p_n) & \delta_y I(p_n)\end{bmatrix}, \qquad \boldsymbol{v}=\begin{bmatrix}\frac{dx}{dt} & \frac{dy}{dt}\end{bmatrix}^T, \qquad \boldsymbol{b}=\begin{bmatrix}-\delta_t I(p_1)\\ -\delta_t I(p_2)\\ \vdots\\ -\delta_t I(p_n)\end{bmatrix} \tag{23}$$

$$A\boldsymbol{v}=b\ \rightarrow\ A^TA\boldsymbol{v}=A^T\boldsymbol{b} \tag{24}$$

$$\boldsymbol{v}=(A^TA)^{-1}A^T\boldsymbol{b} \tag{25}$$

Choosing Points to Track

An important factor affecting the success of tracking are the choice of points to track. Shi and Tomasi have shown that corners generally make for good points to track [7]. Consider the neighborhood of image discussed before being shifted by $\boldsymbol{v} = \left(\frac{dx}{dt}, \frac{dy}{dt}\right)$. The change in intensity of the neighborhood is then given by [8]:

$$E\left(\frac{dx}{dt}, \frac{dy}{dt}\right) = \sum_{x,y} w(x,y) \left[I\left(x + \frac{dx}{dt}, y + \frac{dy}{dt}\right) - I(x,y)\right]^2 \tag{26}$$

Where $w(x,y)$ is a window function assigning weights to the pixels in the neighborhood. Corners, since they are at the boundaries of different groups of pixels with similar intensity, maximize this function, particularly the second term. Like in Equations (12)-(13), the Taylor series expansion of $I\left(x + \frac{dx}{dt}, y + \frac{dy}{dt}\right)$ results in the approximation:

$$E\left(\frac{dx}{dt}, \frac{dy}{dt}\right) \approx \sum_{x,y} w(x,y) \left[\frac{\delta I}{\delta x} \cdot \frac{dx}{dt} - \frac{\delta I}{\delta y} \cdot \frac{dy}{dt}\right]^2 \tag{27}$$

Which can eb rewritten in a matrix form using A and A^T from Equations (17)-(18):

$$B = w(x,y) A^T A \tag{28}$$

$$E\left(\frac{dx}{dt}, \frac{dy}{dt}\right) \approx \boldsymbol{v}^T B \boldsymbol{v} \tag{29}$$

The Shi-Tomasi algorithm looks at the eigenvalues λ_1, λ_2 of matrix **B** and uses a scoring function of $R = \min(\lambda_1, \lambda_2)$ to determine what pixels in the image are corners. We calculate the Lucas-Kanade optical flow for the corners with high R scores and use it to calculate a *focus of expansion* for the image that is used to assess motion towards the endoscope tip. The chosen points and their visualized optical flow trajectories for a sample ear-canal endoscopy video [9] can be seen in Figure 12.

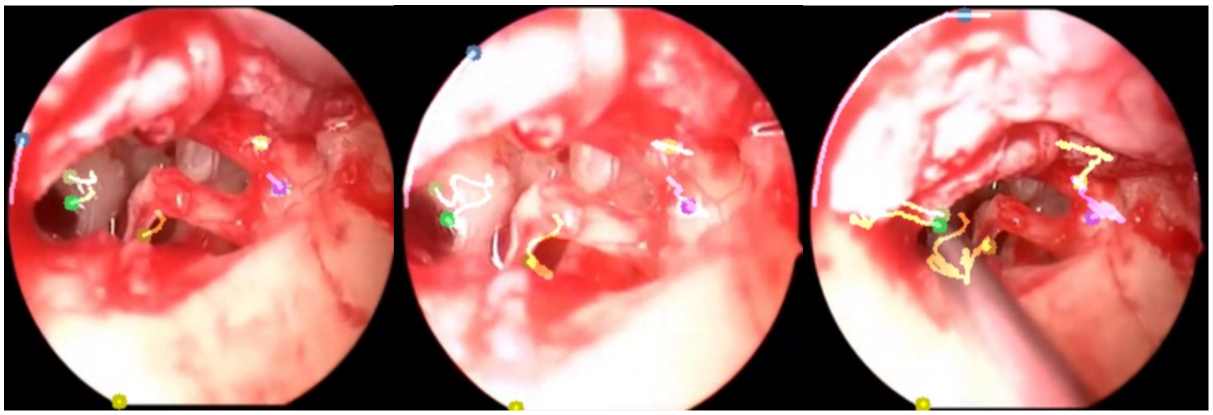


Figure 11: Optical flow of tracked points in the ear canal over time[2]

2.2.3 Finding a Focus of Expansion

The *focus of expansion* (or contraction) is defined as the point on the image plane from which the optical flow trajectories diverge from (or converge to). This divergence, visualized in Figure 12 A, is caused by the motion of either the camera or the image frame moving towards the other and getting closer as seen in Figure 13 C. Therefore, if there is any relative motion in the Z-direction, the direction perpendicular to the image plane, there is assumed to be a focus of expansion. This is useful for our system as we are primarily interested in motion that is in the direction of the endoscope axis, the Z-direction, since this has the greatest potential to damage the patient's ear. If the motion is purely translational in the directions parallel to the image plane (or rotational in these axes) as seen in Figure 13 A,B,D,E, a focus of expansion would not be defined, making the FOE a useful indicator to isolate motion primarily in the Z-direction. Thus for the cases where there is movement in the Z-direction, we can calculate the focus of expansion as the intersection of the optical flow vectors of the pixels that belong to the image background as seen in Figure 12 B, which would be the ear canal in our case.

[2] Unprocessed video from [9]

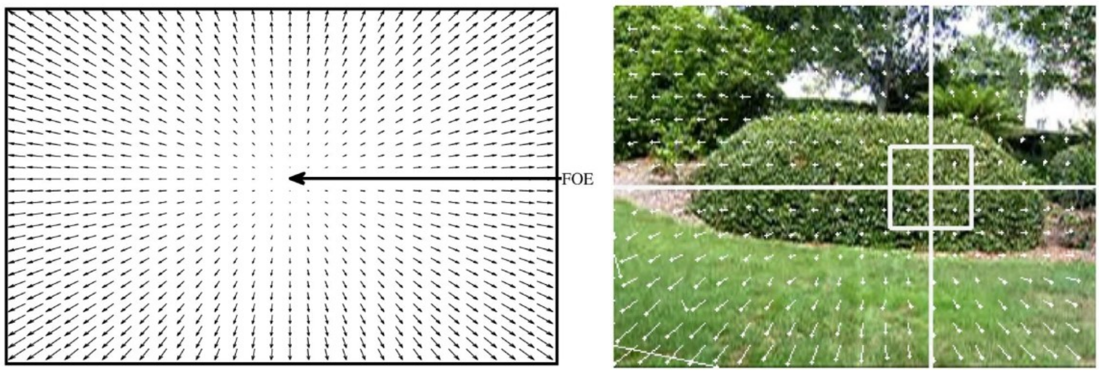


Figure 12 A: The focus of expansion (FOE) of flow vectors in purely Z-direction motion (left), and 12 B: as seen for a real-life case of a robot approaching a bush (right) [10]

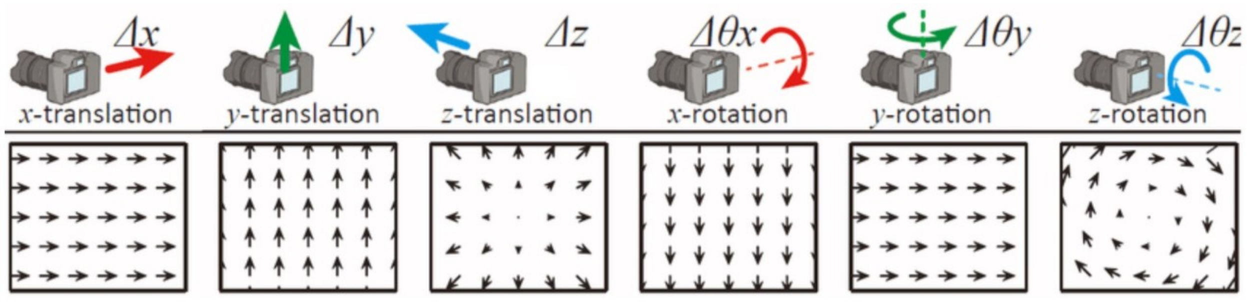


Figure 13 A,B,C,D,E,F: Optical flow created by relative motion between camera and objects [11]

The intersection of two lines directed alongside the optical flow vectors can be calculated geometrically. Let ***a*** and ***b*** denote two optical flow vectors between the tracked pixels in Frame 0 and Frame 1, $\mathbf{a_0}$, $\mathbf{a_1}$ and $\mathbf{b_0}$, $\mathbf{b_1}$ denoting the pixels' locations, respectively as seen in Figure 14.

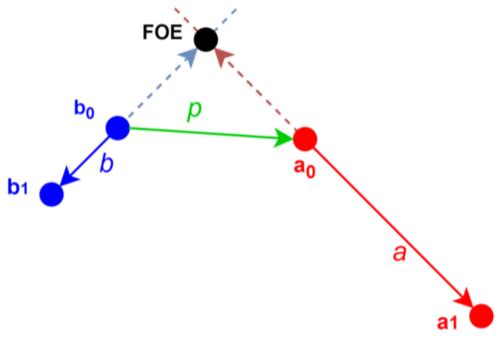


Figure 14: Geometry of (Optical Flow) Vectors and their Intersection

If a and b are not parallel or collinear, their intersection can be represented as:

$$\boldsymbol{FOE} = \mathbf{a_0} + \boldsymbol{a}t_1 = \mathbf{b_0} + \boldsymbol{b}t_2 \tag{30}$$

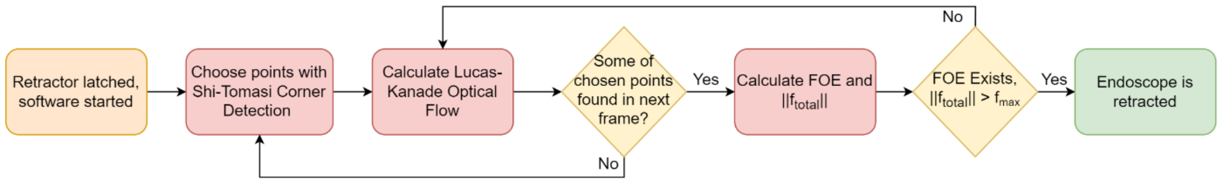
Where t_1 and t_2 are scalar coefficients. Meanwhile, let $\boldsymbol{p}$ denote the vector from b_0 to a_0, $\boldsymbol{p} = \mathbf{a_0} - \mathbf{b_0}$. As seen in Figure 14, $\boldsymbol{p}, \boldsymbol{a}t_1, \boldsymbol{b}t_2$ form a triangle and therefore $\boldsymbol{p} = \boldsymbol{b}t_2 - \boldsymbol{a}t_1$. This provides us with a system of equations using the x and y-components of vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ and allows us to solve for the coefficients t_1 and t_2 and calculate the focus of expansion using Equation (25):

$$A\boldsymbol{t} = \begin{bmatrix} a_x & b_x \\ a_y & b_y \end{bmatrix} \cdot \begin{bmatrix} \mathrm{t}_1 \\ \mathrm{t}_2 \end{bmatrix} = \boldsymbol{p} \tag{31}$$

$$\boldsymbol{t} = A^{-1}\boldsymbol{p} \tag{32}$$

While estimating the focus of expansion in practical cases which are not purely translational in the Z-axis, multiple intersections will be present for the optical flow vectors. We can take the centroid of these intersections points to calculate the most likely focus of expansion and can further make use of a weighted centroid to disregard outliers and noise [12].

The presence of a FOE calculated this way is the main trigger condition upon which the retractor is activated. More precisely, if a FOE exists, and the total magnitude of optical flow vectors, f_{total}, exceeds a threshold f_{max}, the retractor is activated. This approach assumes small lateral movement in the frame plane, as large movement in these directions would mean that overall flow magnitude cannot be accurately corresponded to movement perpendicular to the frame. In my implementation, this threshold was set and tuned empirically after running the pipeline on several endoscopy videos. However, to make this a reliable and robust trigger fit to be used in surgical scenarios, the optical flow magnitude and real-world velocity must be corresponded. One way of calculating this correspondence is to move either the endoscope or the observed object in a known velocity at a known distance (ideally about 1.5-2.0 cm to mimic the surgical scenario), and fit a polynomial relationship between observed optical flow magnitude and real world velocity. This study has been outside the scope of my work so far, as I initially focused on EM system development and lost access to the endoscopy equipment and a realistic test setup due to reasons discussed under Sections 2.2.4 and 4.2.3.Hence, the current state of the vision pipeline to detect motion can be summarized as the following individual steps and contribute to the surgical workflow as seen in Figure 15.


Retractor latched, software started
Choose points with Shi-Tomasi Corner Detection
Calculate Lucas-Kanade Optical Flow
Some of chosen points found in next frame?
Yes
No
Calculate FOE and $||f_{total}||$
FOE Exists, $||f_{total}|| > f_{max}$
Yes
No
Endoscope is retracted

1. Select points to track with Shi-Tomasi Corner Detector
2. Calculate Lucas-Kanade optical flow
3. *Check if points are still "good" to track. If lost in next frame, back to (1)
4. Calculate focus of expansion/contraction
5. Find magnitude of flow vectors
6. *If FOE exists and flow magnitude is above threshold, retract

Figure 15: Surgical Workflow for Computer Vision-Based Retraction Pipeline

2.2.4 Tool Rejection and Improved Robustness

One of the greatest challenges to the optical flow and FOE based approach is the movement of tools within the frame. Although this was not observed to be the case in preliminary tests, a tool moving towards the endoscope has great potential to trigger the retraction system. A simple improvement meant to address and prevent this issue was to filter the points on the tool while choosing the points to track. We can paint the tool a unique color and exclude pixels in a color range from our possible candidates so that points on the tool are not chosen to be tracked if the tool is present in the frame. Painting the tool would also reduce the probability of the system mistaking a point on the tool for a pixel it was tracking due to the different intensity and appearance of tool pixels once they are painted. Good choices of color would be green and cyan contrasting the primarily red, pink, and white ear canal features. Matte paint would also be preferred to reduce the reflection of light, making it harder to mistake pixels on the tool with white pixels elsewhere in the frame. A basic filter function eliminating green pixels from our point selection has been implemented, but it has not been tested due to the COVID-19 lockdown preventing laboratory access and experimentation.

Similarly, considering the total movement of the head over time in the form of an integral controller could help protect the patient against slow but consistent movement that the FOE and flow magnitude-based trigger might not prevent. This idea was briefly experimented with, yielding many false positives, and was left out of the current implementation when the pandemic lockdown prevented further experimentation to improve its robustness.

3 Implementation

This section details the auto-retracting endoscope holder prototype in terms of the hardware that performs the retraction and the software implementation of the tracking and vision methods. Files containing the code and designs here can be reached

3.1 Hardware Implementation

The endoscope-holder, trigger latch, and electronic hardware described here are shared by the implementations of both methods. In addition, the electromagnetic implementation features the NDI Aurora V2 (2011) [3] electromagnetic tracking system, with fiducials mounted on the endoscope holder (as seen in Figure 16) and the patient. In both cases a *Karl Storz Image 1 H3 22220150-3 HD* camera, *222010 20 Image 1 Hub* and *20133120 Xenon 300* light source (Karl Storz Endoskope, Tuttlingen, Germany) *[13]* was used as the equipment used in the operating room by the surgeons, and for the vision system the endoscope is also serially connected to the computer running the vision software in the form of S-Video.

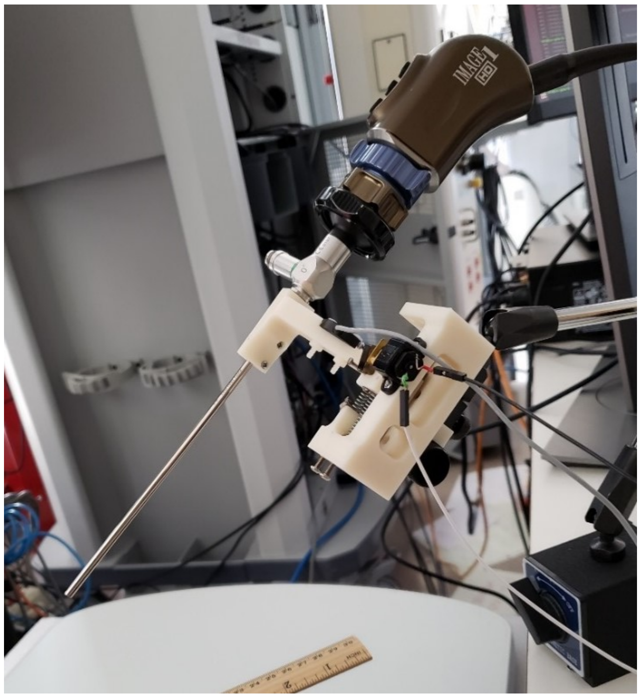


Figure 16: Current Implementation of auto-retracting endoscope holder with endoscope and camera mounted

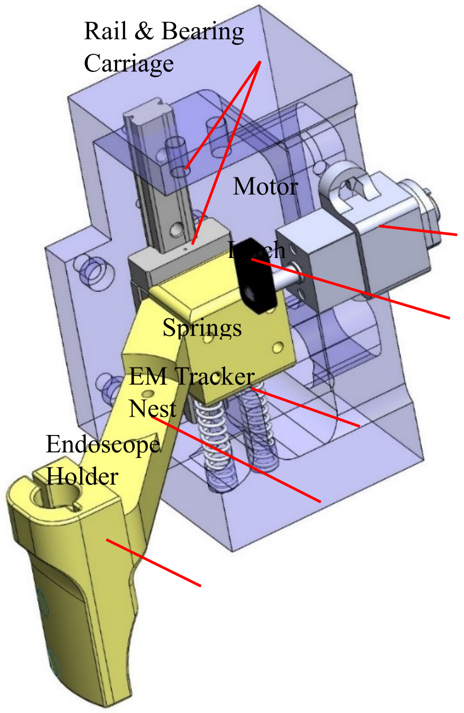


Figure 17: CAD model of retracting mechanism components

3.1.1 Retracting Mechanism

The latest retracting mechanism features the endoscope holder put on a bearing rail which grants it one degree of freedom. To reach a very high linear travel speed during retraction, two springs with spring constants k = 1235 N/m [14] have been put on this rail to push against the endoscope holder to actuate the mechanism. The springs are manually compressed by 9.5 mm until the holder stop, and are kept compressed by the torque output of a DC motor and gearbox, which when retracting rotates out of the springs' way, allowing them to push the endoscope (and attached tracker) away with the stored potential energy. This design means that the system will automatically retract when there is a power outage regarding the motor, providing a safer failure mode. With two springs of spring constants of k = 1235 N/m and a compression x of 9.5 mm, the springs push against the endoscope holder with a force of F = 2kx = 23.465 N while latched.

3.1.2 System Components

The EM system uses the NDI Aurora electromagnetic tracking system with two trackers, which are connected to the NDI Sensor Interface Unit (SIU). The SIU can serially communicate with the user computer via USB. Meanwhile, the Storz endoscope system is connected to the user computer via S-Video using a Hauppauge Live2 frame grabber (Hauppauge Digital, Hauppauge, NY). An overview of how different system hardware and software components come together for both the EM and vision-based system can be seen under Sections 3.2 and 3.3.

An H-Bridge circuit with a TI-l293d and an Arduino UNO microcontroller has been used in this prototype to actuate the motor to latch and release the endoscope holder as seen in Figure 18. The circuit powers the motor while the endoscope holder is latched on and reverses the voltage direction of the motor when desired to make the latch rotate the other way and get out of the springs' way. This way, springs can push the endoscope back as quickly as possible without having to push against the inertia of the motor. Arduino UNO was chosen to be the controller for this circuit since it can be serially communicated with easily in a MATLAB or Python script and allowed for easy and quick changes to the circuit. As further explained under *3.2 Software Implementation*, the Arduino is directly controlled in the EM MATLAB script using the MATLAB Arduino Support Package, and in the vision script using the a serial communication protocol.

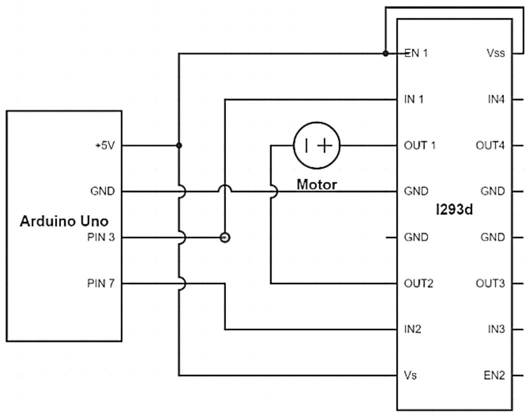


Figure 18: Diagram of circuit to control retracting mechanism

3.2 Software Implementation for Electromagnetic Tracking

For this iteration of the auto-retracting endoscope, a pair or NDI Aurora electromagnetic (EM) position trackers are used. Although this introduces some unwanted novelty to the surgical procedure in the form of having to set up and operate with the electromagnetic tracking system components, it is simpler to implement and is a robust and reliable method for triggering the retraction. An external GUI interface is used to start communicating with the trackers, and the EM tracking pipeline is launched once communication is established, tracking the head and endoscope position throughout the surgery and retracting when necessary. A block diagram showing how the different system components work together for the EM implementation is presented in Figure 19.

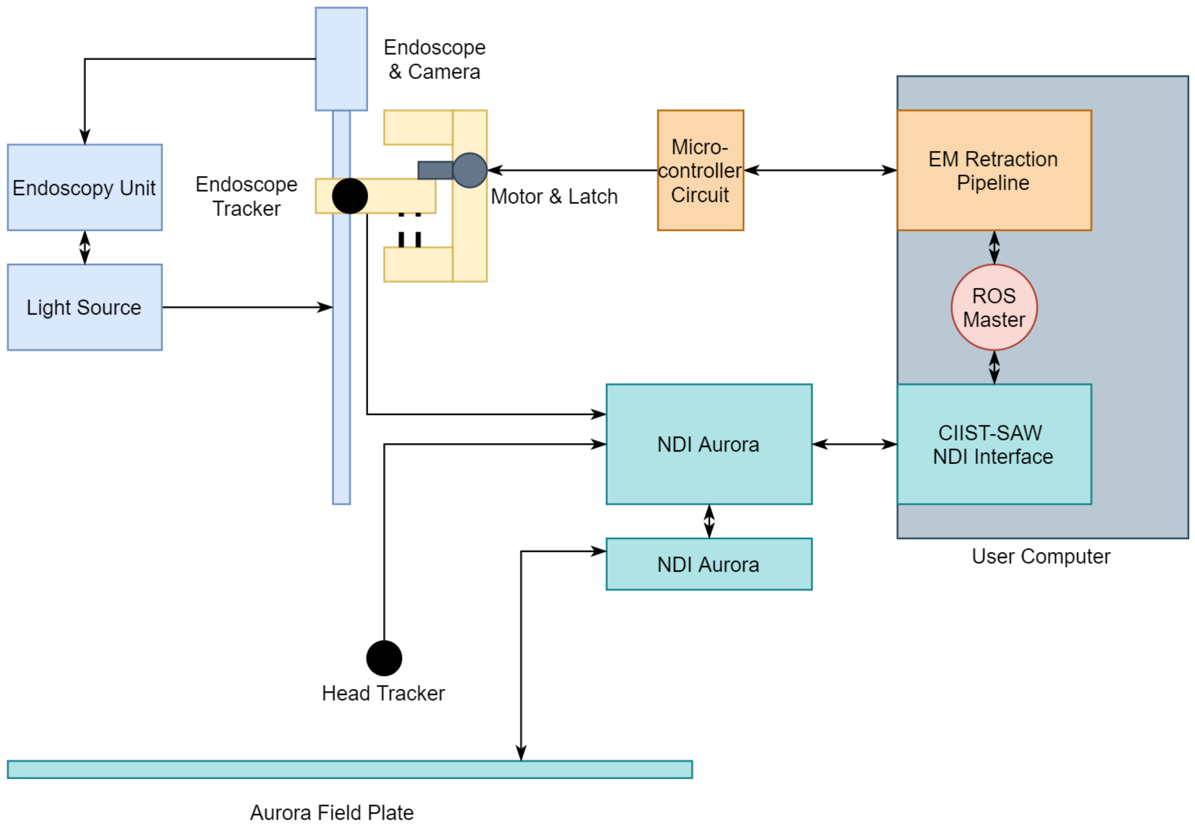


Figure 19: Block diagram of EM Retraction system components[3]

To communicate with the tracking device, the sawNDITracker package [15] is used alongside the Computer-Integrated Surgical Systems and Technology Surgical Assistant Workstation *(cisst-saw)* libraries [16] developed in the Johns Hopkins Laboratory for Computational Sensing and Robotics.

The sawNDITracker package can be ran with the Robot Operating System (ROS, Open Robotics Foundation Inc.) to create a ROS Node to communicate with the EM trackers, publishing data from the sensors into different ROS Topics including the Cartesian position and orientations of the different trackers with respect to the tracker system base. Once such a Node is created, one can subscribe to the relevant ROS Topics to extract and use relevant information in their program. In our case, the MATLAB ROS Toolbox can be used to read and work with real time position data from the

[3] Arrows represent flow of information

EM trackers both for actual tracking and for evaluation experimental data to evaluate the performance of both systems as described under Section 4.1. The pseudocode for the implementation of this method is presented below in Figure 20, and the specific helper functions in this implementation are provided in Appendix A. The pipeline requires the MATLAB ROS Toolbox and Arduino Hardware Support Packages (Mathworks, Natick), as well as the sawNDITracker and CISST-SAW libraries to be installed on the device prior to running the system.

EM Pipeline Pseudocode
Required MATLAB Toolboxes: ROS Toolbox, Arduino Hardware Support Package

```
% Connect MATLAB to ROS Master and subscribe to the two EM tracker topics
rosinit;
sub1 = rossubscriber('/ndi_tracker_ID_1/position_cartesian_current');
sub2 = rossubscriber('/ndi_tracker_ID_2/position_cartesian_current');
% Create Arduino object to connect to Arduino board. Turn on the latch
a = arduino;
writeDigitalPin(a, 'D3', 1);
writeDigitalPin(a, 'D7', 0);
% Initiate and pre-allocate variables
% Start main loop
n = 1;
while time < time_end
    if n == 1
        disp('Press a Key When Distance Set')
        pause;
    end
```

Figure 20: Pseudocode for EM Tracking and Retraction

```
% Extract positions of trackers from rosnode

msg1 = receive(sub1);

msg2 = receive(sub2);

pos1 = msg1.Pose.Position;

pos2 = msg2.Pose.Position;

% Calculate position of the endoscope tip based on previous
% calibration and distance between tip and head tracker

pos_tip, axis = tip_Kinematics(pos1);

diff_inst = pos_tip – pos2;

dist = norm(diff_inst);


% If it is the first measurement (when the surgeon sets desired
% distance), set threshold as desired distance plus some tolerance

if n == 1

    pos1_0 = pos1; pos2_0 = pos2; pos_tip_0 = pos_tip;

diff_0 = diff_inst; threshold = dist – tolerance;

% If it isn't the first measurement, calculate velocities, including
% head velocity in the direction of the endoscopeaxis

elseif n > 1

    v1 = (pos1(n)-pos1(n-1))./(t(n)-t(n-1));

    v2 = (pos2(n)-pos2(n-1))./(t(n)-t(n-1));

    v2_axis = dot(v2, axis)*axis;

end

% After the first two measurements, calculate acceleration, including
% head acceleration in the direction of the endoscope axis

if n > 2

    a1 = (v1(n)-v1(n-1))./(t(n)-t(n-1));

    a2 = (v2(n)-v2(n-1))./(t(n)-t(n-1));

    a2_axis = dot(a2, axis)*axis;

end
```

Figure 20 (continued): Pseudocode for EM Tracking and Retraction

```
    % If distance between tip and head tracker are below threshold or if
    % the head velocity or acceleration is dangerously high, retract by
    % changing motor direction to unlatch mechanism.

    if dist <= threshold || (a2_axis >= max_a || v2_axis >= max_v)

        writeDigitalPin(a, 'D3', 0);

        writeDigitalPin(a, 'D7', 1);

        % Record useful information like time, velocities, and distance at
        % the time of trigger
        t_end = t;

        v_end = [v1 v2];

        diff_end = norm(diff_inst(1:3,1));

    end

end
```

Figure 20 (continued): Pseudocode for EM Tracking and Retraction

3.3 Software Implementation for Endoscope Vision

The endoscope vision pipeline processes real-time mono-endoscope video, and decides on whether the retractor should be triggered to signal the Arduino microprocessor to execute retraction as described in Section *2.2 Assessment of Movement and Danger through Endoscope Video.* The pipeline makes use of several external Python libraries and packages, most notably *OpenCV 2* [17] for video processing, *Rospy* for communicating via ROS, *Pyfirmata* [18] to communicate with Arduino via serial, and *GScam* [19] and *CVBridge* [20] to process video from ROS into more convenient images for OpenCV. It also uses a ROS launch file based on those from the *Johns Hopkins da Vinci Research Kit* [21]. Specific helper functions and classes used in this implementation can be found under Appendix A. The system components for vision-based retraction are presented in Figure 21. The pseudocode is presented in Figure 22.

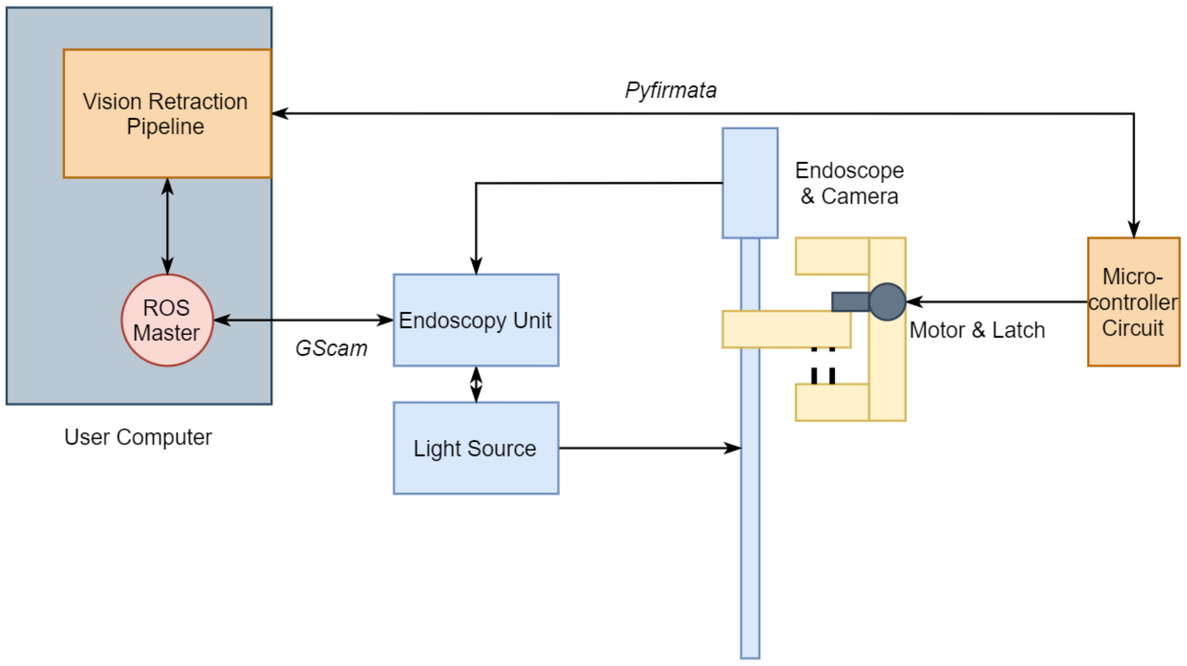


Figure 21: Block diagram of vision-based retraction system components

Vision Pipeline Pseudocode

```
import numpy as np
import cv2
import math
import roslib
import rospy
import pyfirmata
from sensor_msgs.msg import Image
from cv_bridge import CvBridge, CvBridgeError
from time import sleep
# Establish communication with microcontroller and activate latch
# Give the surgeon 10 seconds to put the latch in place
board = latchOn()
sleep(10)
```

Figure 22: Pseudocode for Vision Tracking and Retraction

```
# Initiate image converter object and rosnode to receive video via ros[22]
ic = image_Converter()
rospy.init_node('camera', anonymous=True)

# Assign first frame of video received to imageconverter object
frame0 = np.asarray( ic.cv_image)

#Define parameters for Shi-Tomasi corners, Lucas-Kanade optical flow
st_params, lk_params

# Find points in the first frame of video to track
old_grey, points0 = find_Corners(frame0, st_params)

# Start loop to track detected points
while(run):
    # Assign the next frame in the video to image converter,
    # convert to gray image
    frame = np.asarray( ic.cv_image)
    frame_gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    # Calculate optical flow between points in this frame and previous
    # frame
    points1, error = cv2.calcOpticalFlowPyrLK(old_gray, frame_gray,
    points0, **lk_params)

    # Make sure at least some of the points0 from the previous frame
    # are still found in the current frame, pick new points if none of
    # them are, and calculate optical flow based on these new points
    if points1 is None:
        old_gray, p0 = findCorners(old_frame, feature_params)
        frame =  np.asarray( ic.cv_image)
        frame_gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        points1, st, err = cv2.calcOpticalFlowPyrLK(old_gray, frame_gray,
        points0, None, **lk_params)
```

Figure 22 (continued): Pseudocode for Vision Tracking and Retraction

```
    # Calculate Optical Flow Vectors and their Magnitudes
    V = points1-points0
    avgMag, magnitudesV = flowMagnitude(V)

    # Calculate Focus of Expansion/Contraction
    foe = findIntersection(V, points0, maxCorners)

    # Draw point trajectories on video

    # Check trigger condition and retract if FOE exists and optical
    # flow magnitude is greater than set threshold, then don't run
    # another loop
    # The threshold in this implementation has been empirically determined
    # as the value that best separated sudden, large movements from slow,
    # steady ones in the video samples.
    if ((np.linalg.norm(foe) > 1) and (avgMag > magThresh )):
        latchOff(board)
        run = 0

    # Update previous frame and its points to be the current frame and
    # it's points
    old_gray = frame_gray.copy()
    p0 = good_new.reshape(-1,1,2)

# Close program interface windows, release captured vide
cv2.destroyAllWindows()
cap.release()
```

Figure 22 (continued): Pseudocode for EM Tracking and Retraction

4 Evaluation & Results

4.1 Data Collection and Evaluation Methods via *rosbag*

Since ROS has been used as the communication tool between our different system components, it's *rosbag* package has been used to collect and save experimental data throughout different evaluation procedures. To process necessary information from any saved rosbag, a complementary MATLAB script was developed and is presented under Appendix B. It extracts the relevant data, from the relevant ROS Topics of Cartesian position, calculates the relative velocity and acceleration between the tracker for every timestamped position reading and plots them. The script can process multiple rosbags to plot the different trials and their averages in a single plot for easier visual and statistical analysis. These plots can be seen in Figure 23 and 24.

4.2 Retractor Velocity Characterization

4.2.1 Retractor Velocity Evaluation Procedure

The NDI tracking software is initiated. The head tracker is kept stationary in place, and a rosbag recording is initiated. Then the retractor is latched in place, and is actuated via human input rather than the actual retraction condition to examine the retracting mechanism in isolation from the movement tracking and detection components. The cartesian position of both trackers is extracted from the rosbag, and a plot of the relative distance between the trackers is obtained as seen in Figure 23. The slopes of this plot between time instances (or distance travelled over the time period) is calculated as the velocity of the endoscope tracker.

4.2.2 Retractor Velocity Results

The distance between the stationary head tracker and moving endoscope tracker throughout the retraction process is visualized in Figure 23 for 9 trials. The average of these nine trajectories (seen as the black line) yields a retraction of 9.5 mm over 20 at an average initial velocity of 48.9 cm/s and acceleration of 238/5 cm/s^2. The distance, velocity, and acceleration plots reveal that the distance changes with a consistent slope and profile for each trial, and the initial velocity and acceleration are never below 40 cm/s and 200m/s^2 respectively.

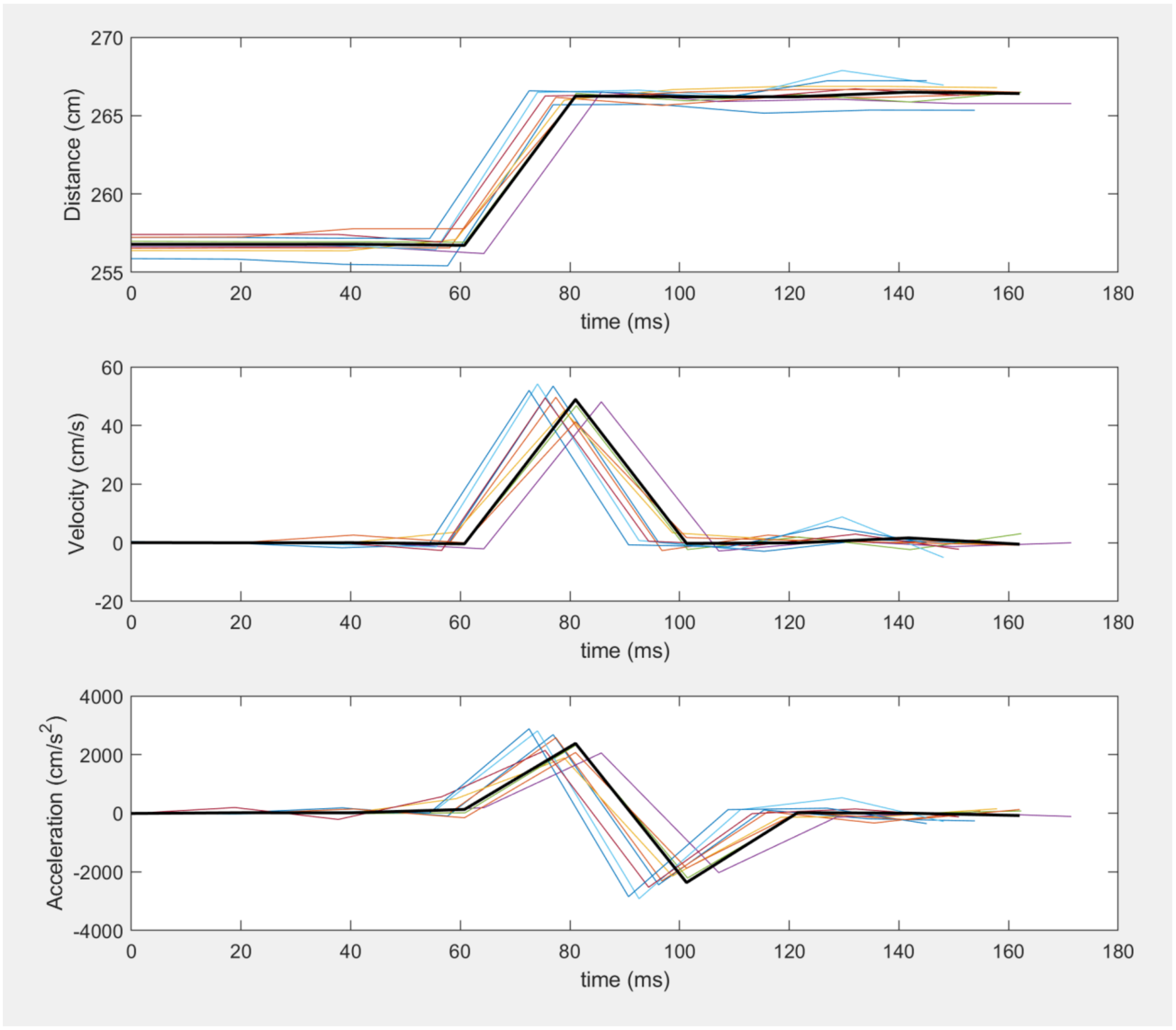


Figure 23: Relative Distance, Velocity, and Acceleration Profiles for the Velocity Characterization Experiment, with 9 trials

4.3 System Latency and Response Time

This test's objective is to simulate the surgical scenario by moving the head tracker towards the endoscope tracker after a threshold distance is set by the user. There is a predetermined buffer (of 2 mm in this set of experiments), and the time difference between the first measurement of distance closer than the threshold and the first measurement where the distance starts to increase again is defined as the latency of the entire system. In other words, this overall system latency refers to the time between the entry of the head tracker in the threshold and the time the endoscope starts retracting. The system latency of the tracking-based system has been evaluated following the procedure below.

4.3.1 Response Time Evaluation Procedure

To characterize the system's overall latency, the tracking code is started, and the user sets a threshold distance. The head tracker is manually moved and brought closer to the endoscope tracker, which triggers the retraction. The positions of the trackers throughout the experiment are recorded via rosbag. The response time is evaluated as the amount of time passed after the head tracker moves past the threshold and when the relative distance starts increasing again in the relative distance plot. The threshold for these experiments was set to be 2mm.

4.3.2 EM Response Time Results

Figure 24 shows the relative distance between the trackers. The results in this graph are normalized around zero centimeters, i.e. they are zeroed around the distance set by the user since each distance set by the user was slightly different. The average of nine trials is seen as the line in black, and reveals a latency time of 120 ms for the tested system. It is important to note that the system evaluated in this experiment simply reacted with retraction when the head tracker was observed to be too close. By simply changing the trigger condition to take into account the current head tracker velocity and acceleration to predict when it is expected to go beyond the threshold, this latency can be improved further with ease.

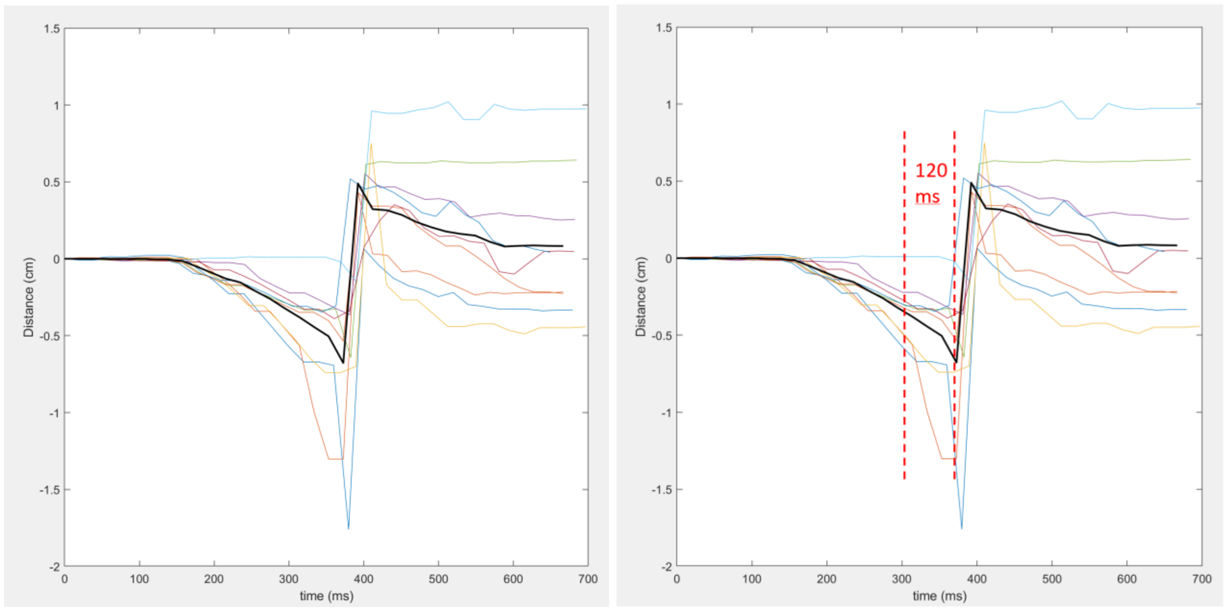


Figure 24: Relative Distance Profiles for Latency Experiment with 9 trials

Since the tracker was moved by hand, the movement speed was not controlled in these experiments, and effects how much overshoot past the threshold is observed before retraction begins. As seen in Figure 24 in the worst case moving more than 1.7 cm past the desired distance for a with instantaneous initial velocity of 90 cm/s and instantaneous initial acceleration of 50 m/s^2. However, a previous study by Creighton et. al has shown that the head motion of a patient under anesthesia was observed to be no more than 0.75 m/s^2 [1]. Hence the worst cases tested and observed in these trials involve acceleration far above the scope of the surgical problem. For the slower cases in this set of trials, with acceleration rates around 20 m/ s^2, the offshoot observed was about 0.3 cm, which is less than the 1.0-1.5 cm buffer zone the surgeon is expected to place between the endoscope tip and any structure in the ear canal. A cadaver experiment where the head tracker is placed on the cadaver's head and the head is moved around at speeds and manners closer to the live surgical scenario can yield the true amount of offshoot that one would expect to see in the operating room. However, I have not been able to perform further experiments using a cadaver or inquiring about slower head velocities due to the loss of laboratory access amidst the COVID-19 pandemic.

4.3.3 Vision System Evaluation and Cadaver Experiments

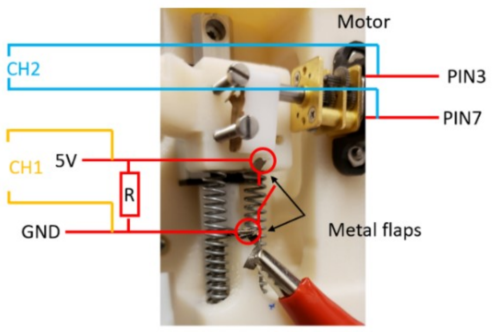
The latency and robustness of the vision system were meant to be evaluated, however due to the loss of access to the necessary equipment due to the COVID-19 pandemic, an extensive evaluation could not be performed in time. The procedure for evaluating latency would be the same as that of the EM system, the trackers would be left on the system to log the relative positions of the endoscope and object representing the patient's head. The system however would be run through the vision-based pipeline and retract upon the visual detection of the object approaching the endoscope.

In addition to evaluating latency, the robustness of both the EM and vision systems was to be evaluated in a set of cadaver experiments. The system would be setup and placed as it was intended to be during surgery, and the cadaver head would be manually moved to replicate the patient's head movement. The success rate of the retraction system would be determined by the rate of successful retractions where there would be no contact or collision inside the ear canal, where failures could be identified from the endoscope video (by for example the frame turning dark due to collision) and the surgeon's external observations. Although both systems require this cadaveric experiment, it is particularly important for the vision system as simulating the surgical scenario from a vision perspective has not been possible in the lab while the relative movement of the EM trackers are expected to simulate the surgical case similarly enough. More difficult cases such as a tool moving within the endoscope video can also be assessed by such a study.

4.4 Hardware Latency

In the early implementations of the system when we evaluated the overall system latency, it was observed to be much higher than expected from the software components. This led to the initiative to separately investigate and identify the mechanical hardware latency as the bottleneck causing much of the delay observed. A separate evaluation procedure has therefore been developed and outlined here to address the mechanical latency of the system, defined as the reversal of the motor current and the actual release of the endoscope holder.

To measure the phase difference between the actuation of the motor and the start of the retracting motion, an evaluation circuit was designed and implemented as seen in Figure 25 A with two channels of an oscilloscope being connected across the motor input pins and across the evaluation circuit respectively. The evaluation circuit featured metal flaps that were attached to the contact points between the stop and endoscope holder, and a parallel resistor. This way, when the endoscope holder was latched, there would be zero voltage difference across channel one and a 5 V difference as soon as the endoscope holder started retracting and lost contact. The phase difference between channel one and channel two of the oscilloscope would therefore be the latency between trigger and the actual retraction. This resulted in a plot presented in Figure 25 B, indicating a hardware latency of about 50 ms. Although the sliding mechanism has been changed in the latest implementation and now uses a faster and more robust linear bearing, the system latency is still observed to be no less than 120 ms as discussed in Section 4.2.2. During the experiments, the primary reason behind this latency was observed to be the speed at which the latch moves out of the endoscope's way.



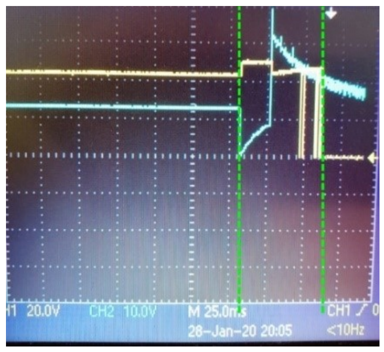


Figure 25 A: Experimental Setup (left) and 25 B: Phase difference observed on the oscilloscope (right)

5 Improvements and Next Steps

The next steps to improve the auto-retracting endoscopy system will involve design and manufacturing improvements regarding the hardware and increasing robustness for the vision system. Further tests with more trials regarding the robustness of both pipelines would also be a significant step towards the commercialization of this product.

5.1 Hardware Improvements

The time it takes for the latch to move completely out of the way of the endoscope holder is identified as the most significant bottleneck of the current system through observation. In response to this, a new retraction mechanism featuring a different latch has been designed by Anna Goodridge as seen in Figure 26. This latch also prevents the endoscope from potentially sliding back down after retraction, which introduces another layer of safety. The design incorporates other improvements such as housing for the electronics and a safe and ergonomic way of manually latching the mechanism at the start of the procedure.

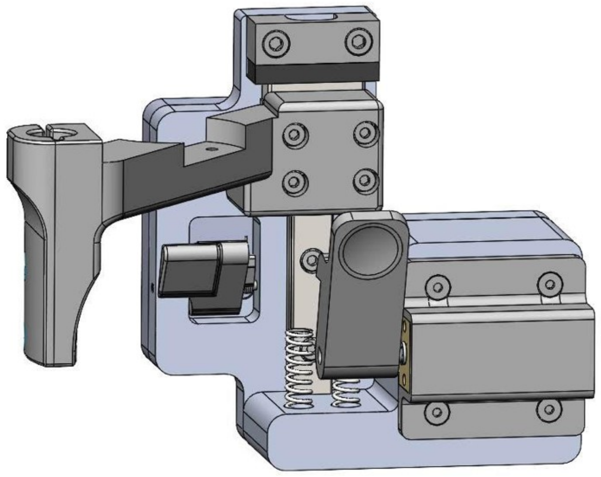


Figure 26: Next Iteration of the Retracting Mechanism [23]

5.2 Testing and Improving the Robustness of the Vision System

5.2.1 Testing for Robustness and Latency

Early tests regarding vision robustness have been conducted, but not extensively. The system must be evaluated over many cases, particularly more difficult situations where there is a tool moving in the video frame. A cadaveric test as outlined in Section 4.3.3 would be an essential next step in the evaluation and further development of both the vision and EM systems.

Besides robustness, the latency of the vision system should be evaluated in a similar manner to the evaluation of the EM system, as discussed in Section 4.3.3. Although hardware is observed to be the bottleneck causing increased latency and early assessments show that the pipeline responds quickly to incoming objects, the pipeline and overall system latencies must be quantified.

5.2.2 Vision System Improvements

After testing the success rate of the vision system, several improvements can help improve it. The tool rejection method described in *Section 2.2.4* can be integrated into the current system, tested, and improved upon. Alternative distance detection methods can also be employed in addition to or in place of the FOE approach. One such method was detecting the blurring of the perceived image due to the object moving in and out of focus [24]. One can quantify blur by computing a Fast Fourier Transform of the image and investigate the distribution of low and high frequencies. In general, the scarcity of high frequencies would indicate a high amount of blur, and quantifying change in frequencies from frame to frame can be used to identify the movement of the patient's head towards the endoscope.

There are also several threshold variables that the current FOE oriented system uses to decide whether it should retract or not, particularly regarding the choice of points to track, and the magnitude of optical flow, which I empirically determined to fit the sample videos and experiments I performed. After thorough tests on a cadaver ear, these thresholds can be optimized to yield the best retracting performance resulting in the smallest number of retractions avoiding false positives and the smallest number of collisions. Determining the threshold for optical flow magnitude is particularly essential to successfully implementing the vision method in a surgically viable system. This requires the vision system to be calibrated in order to relate flow magnitude to actual movement speed. One way to perform calibration on the system might be to move the frame or the endoscope by a known velocity and fit a polynomial to map it to an observed optical flow magnitude.Finally, the trigger condition can be improved by taking the total movement of the head over time into account like an integral controller as discussed in Section 2.2.3. When I experimented with this idea and observed that it led too many false positives, however refining the integral controller further would be a promising idea since it would caution the system against small yet consistent motion towards the endoscope.

6 Conclusions

An autonomously retracting endoscope holder has been implemented to safely hold the endoscope during otologic and related surgical procedures, and two alternative danger assessment methods have been explored in the context of otologic and similar narrow-space surgical procedures. The system monitors the patient's head movement with electromagnetic tracking or endoscope vision, decides whether it is likely to cause collision inside the canal, and swiftly retracts the endoscope if this is the case. To do so, the electromagnetic tracking pipeline uses the kinematic relationship between the electromagnetic fiducials and endoscope tip to assess their relative positions, isolating and quantifying the head movement in the direction the endoscope is pointing at. Meanwhile, the computer vision pipeline calculates the optical flow and focus of expansion of the mono-endoscope video to quantify the relative motion of the head in the same direction. A mechatronic retraction mechanism using springs and a motor-latch system was prototyped and used with the electromagnetic tracking pipeline to test and confirm that the system can robustly and swiftly detect collision and retract with enough speed to satisfy the demands of the surgical scenario. Although cadaveric studies were also planned to more thoroughly test the system with both pipelines, and to better simulate a variety of surgical cases for the vision system to evaluate and improve its performance, these steps were postponed for the moment due to the COVID-19 pandemic. Regardless, the prototyped system has successfully explored and confirmed the feasibility and potential of an automatically retracting endoscope holder. It's implementation of specific danger assessment methods and software and retraction hardware ts as the first step towards a commercial design that can be widely implemented in operating rooms to make otologic and related procedures easier and safer.

7 References

[1]: P. Creighton, Personal communication.

[2] Patent R. H. Taylor and F. X. Creighton, "Safety feature for use with robotically manipulated endoscopes and other tools in otolaryngology and neurosurgery", *International Patent Application WO 2020/28747 A1*; filed 2 August 2018; published 6 February 2020

[3]: Northern Digital Instruments, "Aurora". [Online] Available: https://www.ndigital.com/medical/products/aurora/. [Accessed 22-Apr-2020].

[4]: R. Szeliski, "Optical Flow," in *Computer Vision: Algorithms and Applications*, London, Springer, 2011, pp. 360

[5]: S. S. Beauchemin and J. L. Barron, "The computation of optical flow", *ACM Computing Surv*eys, vol. 27, no. 3, 1995, pp. 433–466

[6]: B. Lucas and T. Kanade, " An iterative image registration technique with an application to stereo vision," in *Proceesings of 7th International Conference on Artificial Intelligence,* Vancouver, 1981, pp. 674-679

[7]: J. Shi and C. Tomasi, "Good features to track," *1994 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, Seattle, 1994, pp. 593-600.

[8]: C. Harris and M. Stephens, "A Combined Corner and Edge Detector," *Alvey Vision Conference,* vol. 15, no. 50, 1988, pp. 147-151.

[9]: I.R. Tavárez Rodríguez, "Timpanoplastia endoscópica por Iván Tavárez Rodríguez Otorrinolaringología", YouTube, 11 May 2014, [Online]. Available: https://www.youtube.com/watch?v=rYwGBl9NSek [Acc. 24-Apr-2020].

[10]: C. McCarthy, R.E. Mahoney, N. Barnes, "A Robust Docking Strategy for a Mobile Robot using Flow Field Divergence," *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Beijing, 2006, pp. 5564-5569.

[11]: Nitsche, M. “Appearance-based teach and repeat navigation method for unmanned aerial vehicles”, PhD Thesis, 2016, pp. 40.

[12]: P. Gil-Jiménez and H. Gómez-Moreno, "Estimating the focus of expansion in a video sequence using the trajectories of interest points," *Image and Vision Computing,* vol. 50, no. June 2016, pp. 14-26.

[13]: Karl Storz Endoskope “Cameras, light sources and documentation,” [Online]. Available: https://www.karlstorz.com/us/en/telepresence.htm [Acc. 3-May-2020].

[14]: McMaster-Carr, “Compression Spring 1.25’’ Long, 0.18’’ OD, 0.132’’ ID,” [Online]. Available: https://www.mcmaster.com/9657K326?fbclid=IwAR3Hqp9YFZstRxAPSvbWc5aGB IpfnxoWCur5jcXfMIxULKF4nYuplUvjNkQ [Acc. 3-May-2020].

[15]: A. Deguet, A. Uneri, et. al. “sawNDITracker”, [Online]. Available: https://github.com/jhu-saw/sawNDITracker [Acc. 3-May-2020].

[16]: A. Deguet, Z. Chen, S. Leonard, P. Kazanzides, M. Balicki, P. Chalasani, “cisst-saw,” [Online]. Available: https://github.com/jhu-cisst/cisst-saw [Acc. 3-May-2020].

[17]: Open Source Computer Vision Library, “OpenCV”, [Online]. Available: https://opencv.org/ [Acc. 5-May-2020].

[18]: T. de Bruijn, “Pyfirmata”, [Online]. Available:

https://pypi.org/project/pyFirmata/ [Acc. 3-May-2020].

[19]: J. Bohren, G. T. Jay, C. Crick, “gscam”, [Online]. Available: https://github.com/ros-drivers/gscam [Acc. 3-May-2020].

[20]: P. Mihelich, J. Bowman, “CVBridge”, [Online]. Available: https://github.com/ros-perception/vision_opencv [Acc. 3-May-2020].

[21]: A. Deguet, et al. “dvrk-ros” [Online]. Available: https://github.com/jhu-dvrk/dvrk-ros/blob/devel/dvrk_robot/launch/gscam_hauppauge_live2.launch [Acc. 3-May-2020].

[22]: P. Mihelich, J. Bowman, “Converting between ROS images and OpenCV images (Python)” [Online]. Available: http://wiki.ros.org/cv_bridge/Tutorials/ConvertingBetweenROSImages AndOpenCVImagesPython [Acc. 5-May-2020]

[23]: A. Goodridge, personal communication.

[24]: B. Vagvolgyi, personal communication.

8 Appendices

Appendix A: Helper Functions for EM and Vision Retraction Software Pipelines

Code files including main EM and Vision retraction pipelines and any associated helper functions are available at: https://git.lcsr.jhu.edu/auto-retracting-endoscope-holder/thesis-can.git

EM Pipeline Function: tip_Kinematics
Required MATLAB Toolboxes: ROS Toolbox

```
function tip_Kinematics(T)

% Input homogeneous transformation matrix representing the pose of the %
endoscope tracker, T

% Outputs endoscope axis direction and tip position with respect to
% global frame

% Assign p_tip, endoscope center, and axis as calculated by pivot-axis
calibration

p_tip = [-0.0104; -0.0417; -0.1705];
axis = [-0.1075; -0.9937; 0.0319];


% Separate T into rotation and translation

Rot = trans(1:3,1:3);
pos = trans(1:3,4);


% Calculate tip position with respect to global frame

pos_tip = Rot*p_tip; + pos;
axis_global = Rot*axis + pos;


return pos_tip, axis_global
```

Vision Pipeline Function: latchOn

```
function latchOn():
# Inputs - none
# Outputs: board - board object used to communicate with
#          Arduino via pyfirmata

# Create board object to connect to and identify its serial port
board = pyfirmata.Arduino('/dev/ttyACM0')
```

```
# Activate relevant pints to turn the latch on
board.digital[3].write(1)
board.digital[7].write(0)

return board
```

Vision Pipeline: latchOff

```
function latchOn(board):
# Inputs: board - Arduino board object to communicate with
# Outputs - none

# Activate relevant pints to turn the latch off
board.digital[3].write(0)
board.digital[7].write(1)
```

Vision Pipeline Function: find_Corners

```
function findCorners(frame, params):

# Inputs: frame - the image to find points on and
#         params - parameters for Shi-Tomasi corner detection
# Outputs: frame_gray - grayscale image
#          p0 - points to track


# Convert image to grayscale
frame_gray = cv2.cvtColor(frame_gray, cv2.COLOR_BGR2GRAY)

# Choose good points to track as points
p0 = cv2.goodFeaturesToTrack(old_gray, **params)

return frame_gray
```

Vision Pipeline Function: flow_magnitude

```
function flowMagnitude(V):

# Input: V - an array of optical flow vectors
# Outputs: avgMag - average optical flow magnitude in the frame
#          magnitudes - array of magnitudes corresponding to each vector


for vector in V:
        # Calculate Euclidean magnitude vectorMag and append to magnitudes
        # Filter very small vectors as noise
        if vectorMag > 0.01:
                # Count vectorMag towards the average avgMag

return magnitudes, avgMag
```

Vision Pipeline Function: findIntersection

```
function findIntersection(V, P0):

# Input: V – an array of optical flow vectors
#        P0 – array of points tracked in previous frame (P0 + V gives point
#        locations in current frame)
# Output: foe – calculated location of the focus of expansion as the centroid
#         of vector intersections


# Calculate the intersection of each vector with every other

for v1 in V:
        for v2 in V:
            if (v1 is not v2):

                  # Calculate matrix A as defined in Equation 31 and solve

                  # for parameter t as described in Equation 32

                  A = np.matrix([[a[0][0],b[0][0]],[a[0][1],b[0][1]]])

                  t = np.matmul(np.linalg.inv(A),np.transpose(P0[countB]

                      P0[countA]))

                  # Use parameter t to interpolate intersection point as in

                  # Equation 32

# Calculate foe: the centroid of all intersection points as the focus of

expansion

return foe
```

Appendix B: Scripts for Data Reading and Experimental Evaluation

Data Processing: Pivot Calibration

```
function pivotCal(T)

% Input: T - an array of homogeneous transformation matrices for the relative
%        pose of the endoscope tracker

% Outputs: p_tip - position of endoscope tip wrt. endoscope tracker
%          p_dimple - position of pivot wrt. Reference tracker


% Compile rotations and translations from each frame in the form presented in
% Equation 5 as R_I and p_j

% Perform Singular Value Decomposition on the compiled matrix R_I

[U,S,V] = svd(R_I);

% Solve for x in Equation 8

x = S\(U'*(-p_j));

% Calculate p_tip with Equation 9

ls_solution = V*x;

p_tip = ls_solution(1:3);

p_dimple = ls_solution(4:6);

return p_tip, p_dimple
```

Data Processing: Axis Calibration
Required MATLAB Toolboxes: Computer Vision Toolbox

```
function axiscal(T)

% Input: T - an array of homogeneous transformation matrices for the relative
%        pose of the endoscope tracker

% Outputs: endo_axis - direction vector of the endoscope axis relative to
%          endoscope tracker
%          center - the center point of circle in Figure 9, which is the
%          starting position of the endoscope axis

% Calculate geometric center of the circle in Figure 9 from tracker poses T,
% as the center point of the two points that are farthest apart in the
% circle. This is necessary as simply taking the position mean would not work
% due to a non-uniform point distribution

% Translate every point and store them in G, so that the center is the origin
% Fit Plane to cloud of tracker points using pcfitplane() from the MATLAB
% Computer Vision Toolbox.

[model, inlierIndices,outlierIndices] = pcfitplane(ptCloud, 1);

plane = select(ptCloud, inlierIndices);

% Calculate axis as the vector normal to the plane

axis = model.Normal;

return center, axis
```

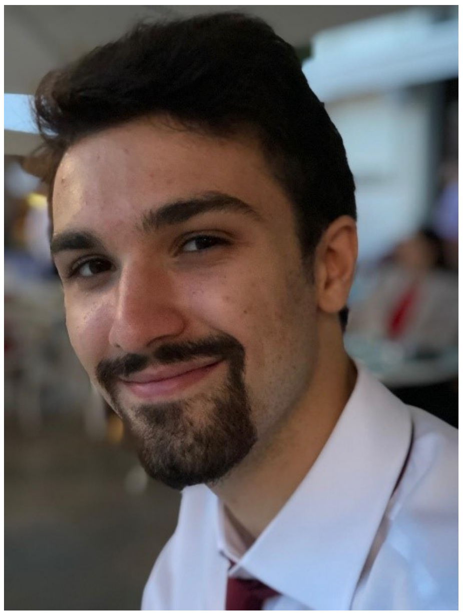
Data Processing: Pivot-Axis Calibration from ROSbag files
Required MATLAB Toolboxes: ROS Toolbox, Computer Vision Toolbox

---

```
function pivot_axis_cal_rosbag()

% Based on data from pivot and axis calibration experiments saved in
% rosbags, calculates the endoscope tip position with respect to the
% endoscope tracker, and the direction of the endoscope axis

% Select either pivot or axis calibration bag saved in the workspace

% Select the topics of interest from bag (pose)

% Read ROS messages from the topics chosen
for n in data_points

      % Calculate endoscope tracker pose with respect to
      % reference tracker, T

If bag_chosen is pivot_bag

      % Calculate endoscope tip position

      [p_tip, p_dimple] = pivotcal(T);

else if bag_chosen is axis_bag

      % Calculate endoscope axis direction and endoscope center position

[endo_axis, center] = axiscal(trans);

return p_tip, endo_axis;
```

---

9 Biography



Can Kocabalkanli was born in Istanbul, Turkey in 1997. He completed his undergraduate work majoring in Mechanical Engineering and minoring in Robotics and Mathematics in Johns Hopkins University, receiving his Bachelor of Science Degree in May 2019. He started working towards his Master of Science and Engineering for Robotics during his senior year, and in Fall 2019 started working with Prof. Russell Taylor and Dr. Francis "Pete" Creighton in the Computer Integrated Interventional Systems Laboratory at Johns Hopkins on the development of an autonomously retractable endoscope holder to assist in otologic and related surgical procedures.