Visual Annotation of Clinically Important Anatomical Landmarks for Vitreoretinal Surgery

Team Member: Vincent Ng
Mentors: Rogerio Richa, Marcin Balicki, Russell Taylor

May 19, 2011

Abstract

We describe a process to provide registration of preoperative images to live intraoperative microscopic stereo feed, overlaying landmarks on the video feeds to give surgeons more visual information during surgeory. This method does not run close to real time(150-300ms), but offers better accuracy than other registration methods. This process can complement faster methods like template tracking when those methods lose registration.

1 Introduction

This paper delves into the topic of using Computer Vision algorithms to aid surgeons in their surgical procedure. The OpenCV library and Computer Integrated Surgical Systems and Technology (CISST) framework is used so that other programs can easily integrate this process. Previous work has been done on this topic using older algorithms[2], and this paper attempts to use newer algorithms and processes to achieve the goals.

1.1 Background

During vitreo-retinal surgery, surgeons look through a microscope while manipulating tools to perform tasks on the retina. They usually sit in a fixed position to look through the optical lenses on the microscope.

1.2 Problem

The surgeon already has limited mobility of the tools inside the retina. The surgeon also has to mentally track where he wants to explore from the preoperative images, and not re-trace his previous steps. This process in this paper attempts to reduce the amount of info the surgeon has to remember, by trying to detect such info in the live feed(based on info from the pre-op scans).

Visual information can be useful in guiding the surgeon during an operation where he experiences limited visual detail. Instead of fixing his posture on the microscope, he can look at a stereoscopic display. Landmarks can be added by the computer onto the image. The surgeon can focus more on the actual procedure. The preoperative data from the OCT image can be useful in guiding the surgeon during the procedure, by integrating it to the live video feed from the microscope and displaying in a high resolution format.

Therefore, this paper describes a process on how registering two images and overlaying landmarks onto the video screen can be done in near-realtime speeds.

2 Technical Approach

The major part of the problem involves registration. The registration of images is feature-based: the detector(SURF) must be able to output the same features in different images.

In the original pre-op image, we choose a small region to track. We run the small region through SURF to get feature points. Then, we obtain an image from the live video feed, and run SURF over the entire image to get a larger set of feature points.

We want to find how the original set of features fit on the new image. To do so, the two sets of features will be matched using Normalized Cross Correlation (NCC), and a transformation between the pre-op image and the live feed will be calculated from these matched pairs of features. Using the transformation, we can overlay the outline edge of the original small region on the live video. Validation is to be done by visual inspection of the overlayed outline to ensure the accuracy and repeatability of the feature detection and registration.

2.1 Feature Detection

The feature detection used in this paper is the Speeded-Up Robust Feature(SURF) detector[1]. This detector is influenced by a previous-generation detector called Scale-invariant feature transform(SIFT). SURF claims to perform better and faster than its predecessor, and be invarient to various image transformations such as rotations and scaling. In the results section of this paper, we discuss how well SURF performs to our expectations.

The only parameter in SURF that can be modified directly is the hessian threshold to accept or reject features. For vitreo-retinal surgery, lighting of the retina is not uniform and bright. Therefore, hessian threshold must be low to get enough features detected.

SURF generates a 64-element matrix describing the neighborhood around the feature point. This matrix is used for feature matching. SURF also offers an option to use a 128-element matrix instead.

2.2 Feature Matching

In order to determine if two feature points on different images are the same, we can compare their 64-element descriptors. This paper uses Normalized Cross Correlation(NCC) to determine if the 64-element descriptons are similar. We only accept matches above a threshold of 0.95.

$$NCC = \frac{\sum (preopdescriptor_i - preopdescriptor_{mean}) * (intraopdescriptor_i - intraopdescriptor_{mean})}{standarddeviation_{preopdescriptor} * standarddeviation_{intraopdescriptor}} \tag{1}$$

2.3 Calculating Transformation

Having a list of matched feature points, the problem left is to generate a transformation between the two matches. We initially used OpenCV's cvFindHomography function which finds the perspective transformation between two planes via RANSAC. Later, we encoded our own RANSAC function

that calculates only 3 degrees of freedom(rotation of the image plane, and translation in x/y coordinates). The transformation is calculated by solving the least squares equation below.

$$\begin{bmatrix} preop_x & -preop_y & 1 & 0 \\ preop_y & preop_x & 0 & 0 \end{bmatrix} \times \begin{bmatrix} \alpha sin\theta \\ \alpha cos\theta \\ translation_x \\ translationy \end{bmatrix} = \begin{bmatrix} intraop_x \\ intraop_y \end{bmatrix}$$

2.4 OpenCV, CISST framework

OpenCV provides an abundance of tools that were used. It allows fine-grained manipulation of image data, built-in functions for doing operations(cvFindHomography). Initially, we used those functions to prove the viability of the process. We then rewrote some functions to increase the speed or accuracy of the process.

CISST is an existing framework in the laboratory. The microscope and existing projects use this framework, which provides a detailed process on pipelining any tools we need(video recorder, exposure/image resizer methods). This paper attempts to create a CISST filter that does this process, while being extremely portable and can be slipped into other projects easily.

2.5 Integrating with other CISST filters

This process runs quick, but not exactly real time. There exists other programs(Template Tracker) that performs fast, but not as accurately. Therefore, it would be ideal for the Template Tracker to switch to this process when it fails, and switch back when registration is good.

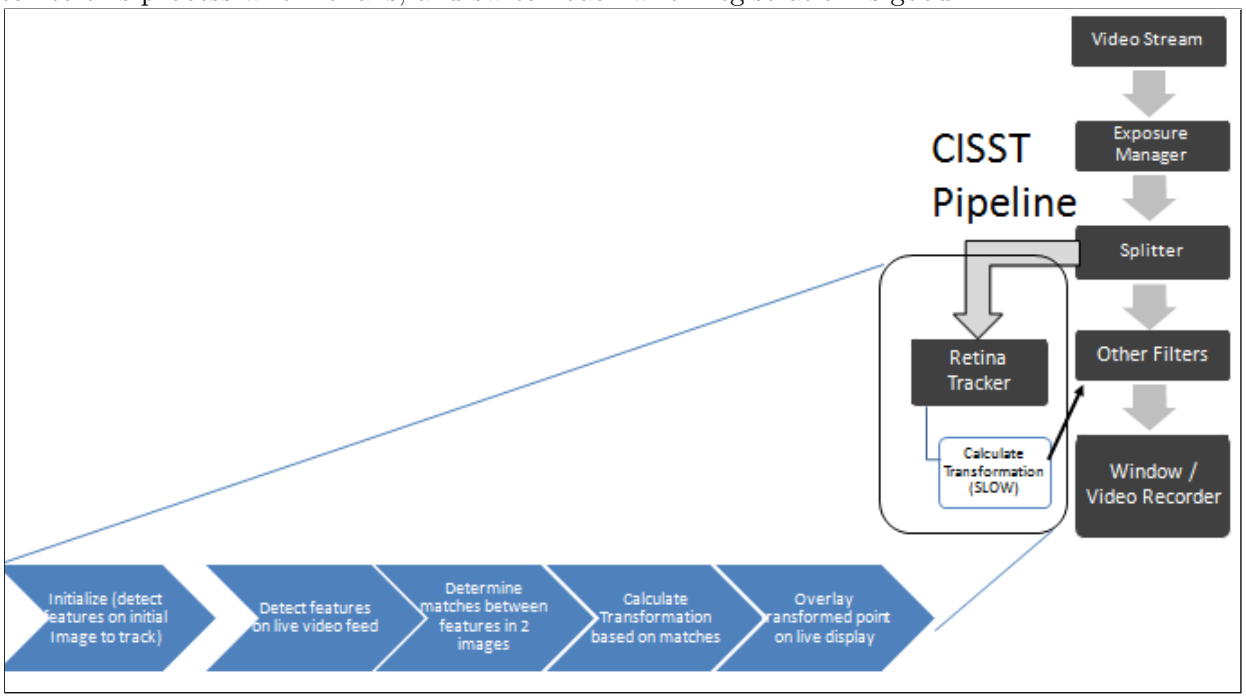

Figure 1. Sample CISST Pipeline

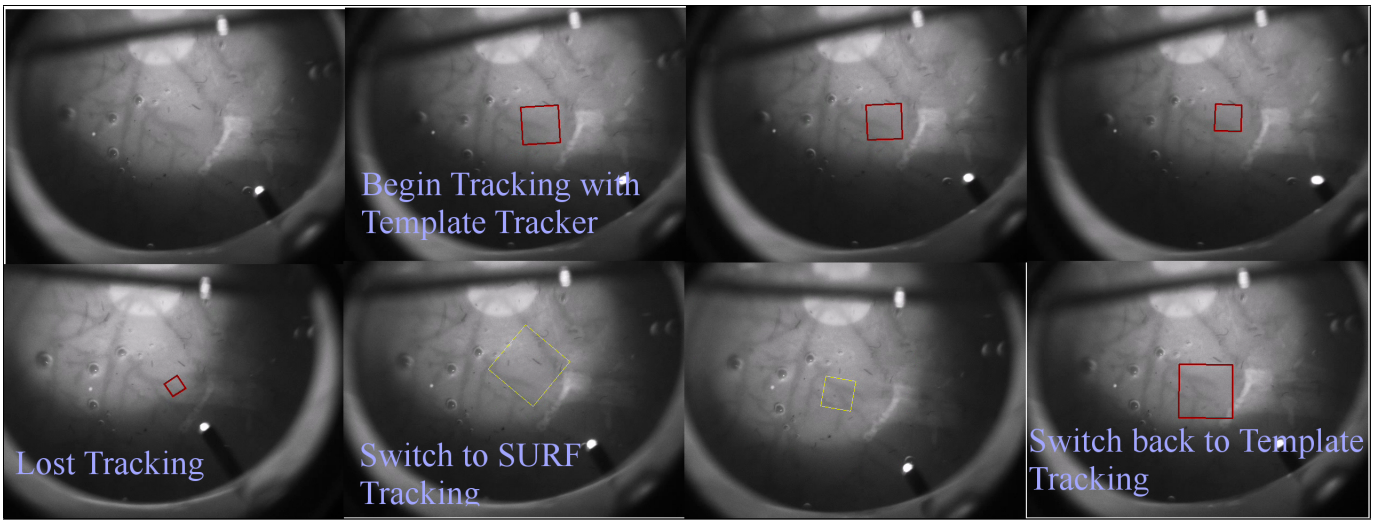

Figure 2. Mockup of how the this process can interact with other CISST Filters

3 Results and Discussion

3.1 Non-surgical data

We used a simple flat-image of a retina as the test data. The input image was set at 640x480, hessian threshold at 10, and tried to track a 100x100 pixel region(40 SURF points) of the image. This process was able to generate a transformation every 100-300ms, transformation valid almost in every single calculation. The program was able to maintain tracking of the region if it is present in the video feed.

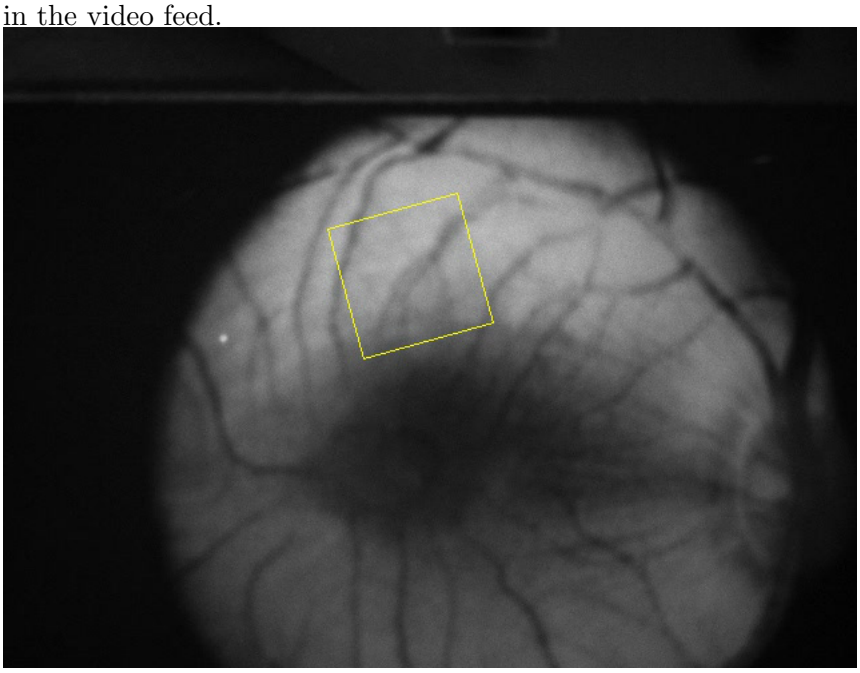
Figure 3. Results of running algorithm on flat image of a retina

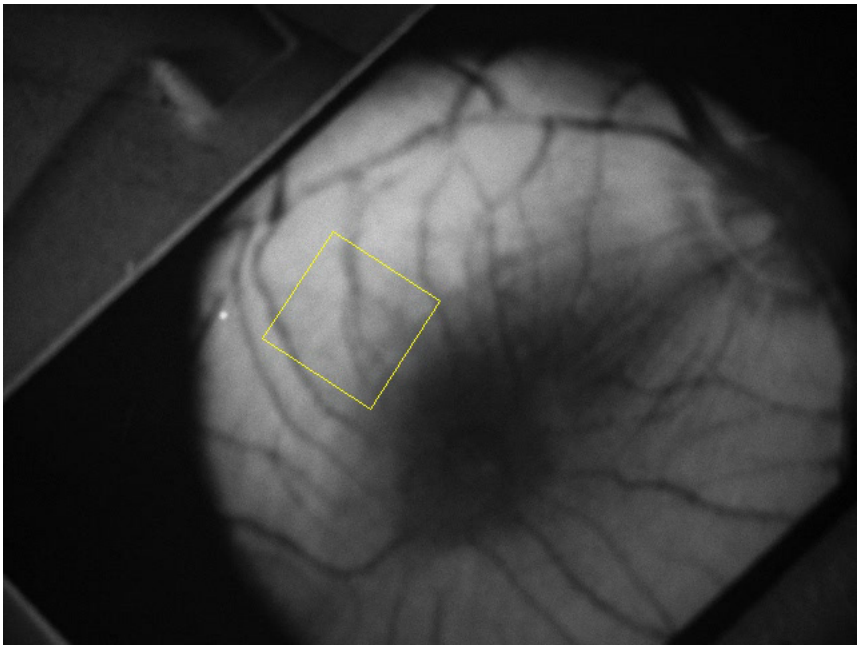

Figure 4. Results of running algorithm on flat image of a retina

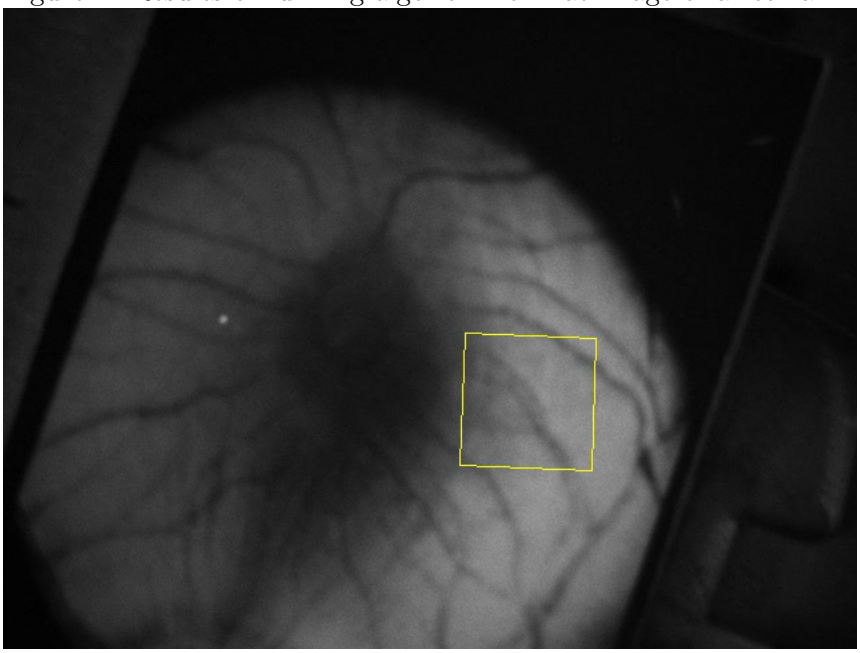

Figure 5. Results of running algorithm on flat image of a retina

3.2 Surgical data

The results were similar to non-surgical data. However, the accuracy of the transformation dropped. Instead of getting a transformation valid in almost every calculation, the accuracy rate was between one correct transformation in every 4-10 calculations.

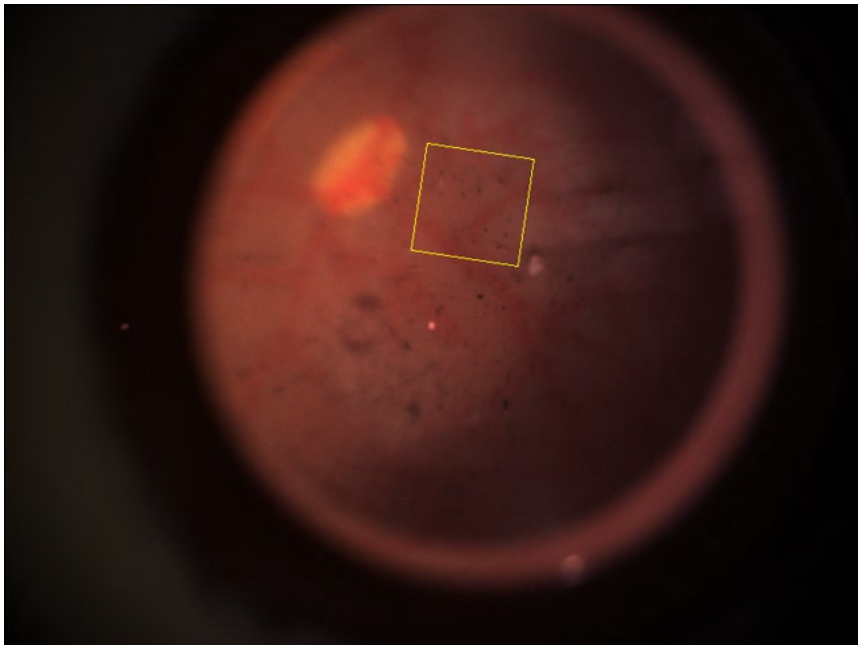

Figure 6. "Window to track" on retina phantom.

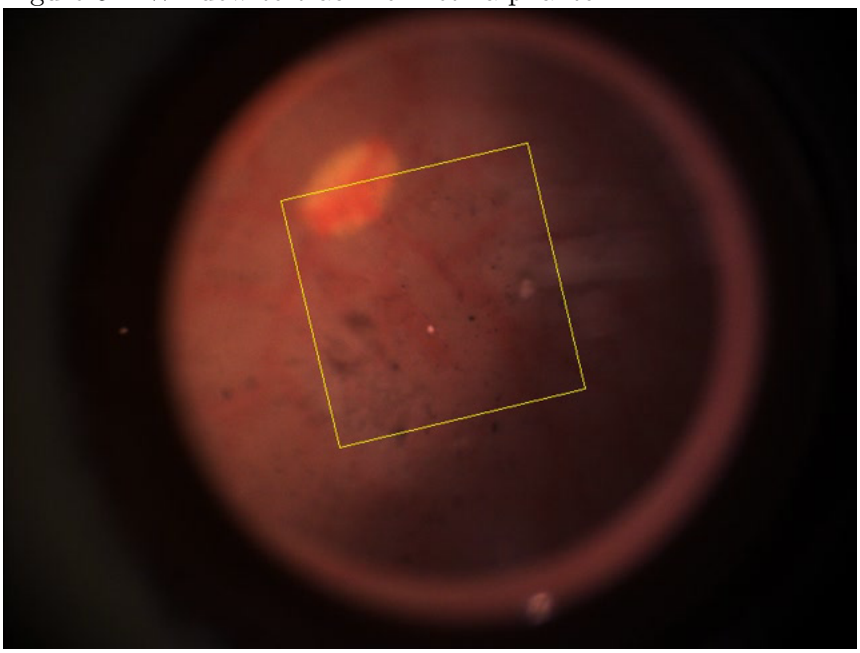

Figure 7. Bad match on retina phantom.

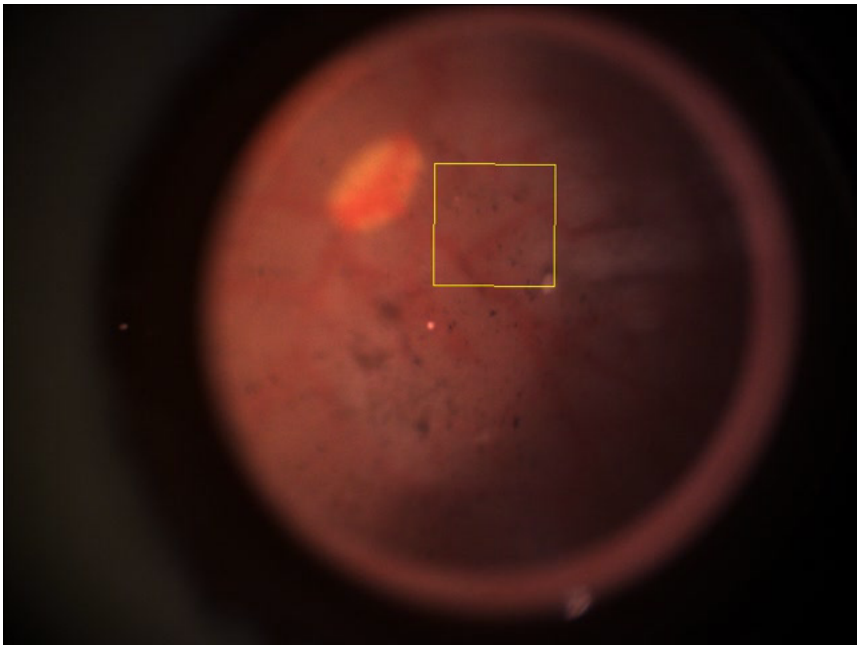

Figure 8. Relatively Good match on retina phantom.

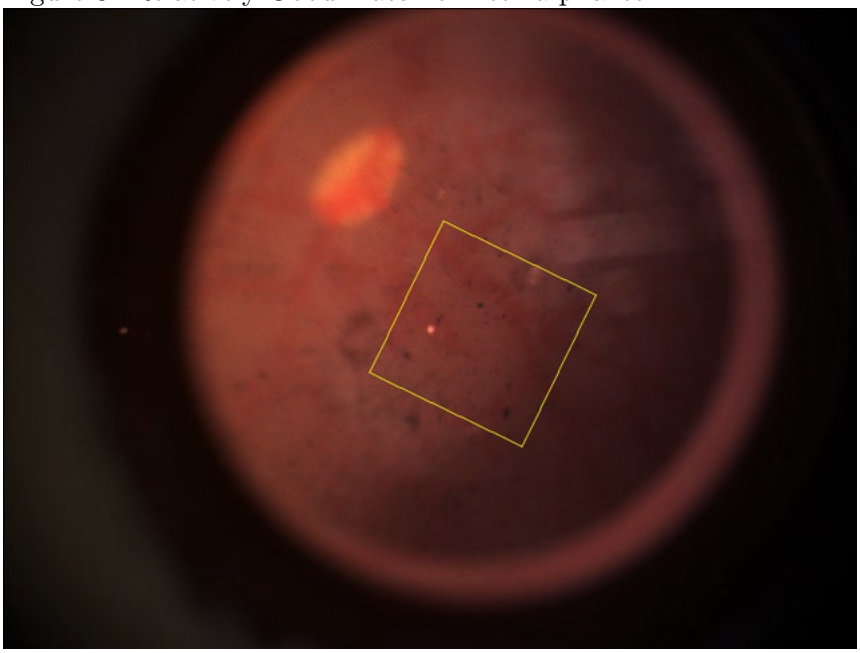

Figure 9. Bad match on retina phantom.

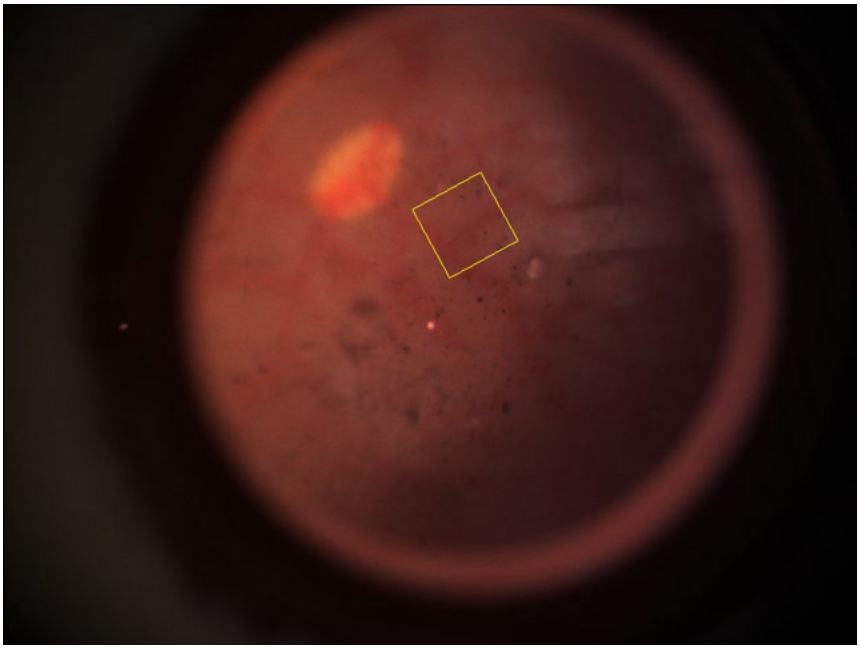

Figure 10. Bad match on retina phantom.

3.3 SURF: Hessian Threshold

SURF's creators suggest setting the threshold between 200 and 500. However, we had to use a threshold of 10 to get enough features for tracking. The original image to track needs at least 30 features for tracking to be viable. We feel that we are going towards the limitation of the SURF algorithm, due to the lack of detail or brightness of our target object, the retina.

3.4 SURF: 64 vs 128 Element Descriptor

SURF can generate a 128-element descriptor instead of a 64-element descriptor. Theoretically, it allows a better way to distingush between similar feature points. However, as described in the earlier subsection, we deal with images of low levels of details, and using this option does not give us significantly better results. Also, increasing the descriptor size doubles the runtime of the feature matching portion(NCC). Therefore, we elect not to use the 128-element option.

3.5 Transformation

Originally, the cvFindHomography function from OpenCV was used. However, that provides a transformation with 9 degree-of-freedom(DOF) transformation. The resoluting overlay was not stable, and kept fluctuating a lot due to the high degrees of freedom. Therefore, we fell back to obtaining a 3-DOF transformation, which allows the image to be rotated on the image plane and translated in x/y direction. So far, using this transformation with the curvature of the retina yields no problems, probably because the details of the features in the retina are not distinct enough, leading to loose matching of feature points and calculation of the transformation even if the point might be rotated in the other axis.

3.6 Validation

During testing on a retina phantom, the SURF algorithm struggles tremendously on the low light condition and low feature detail. It manages to get a good transformation result every 4-10 calculations.

A good transformation is where the overlayed square is consistent with the expected location(the size, location and rotation of the square is consistent with the actual movement)

4 Conclusion / Significance

The results show that it is possible to provide near real-time tracking and overlay to a live video feed. This process will be useful for a surgeon. The surgeon can load a pre-operative fundus image, choose specific points on the fundus to track, and the program will search for(and highlight) points on the live video feed if there are enough feature matches between the fundus points and the video feed.

However, looking at sample intra-operative data from a real surgery, the amount of detail and lighting is sub optimal, worse than the images presented in this paper. Part of this is because the video recording of the microscope is a secondary task: surgeons look through the lenses and worry more about what they are doing with the tools than the recording quality. Therefore, it is still too early to say if this process will work properly in a live surgical setting.

5 Future Work

1. Speed of the algorithm: Currently, the process takes 150-500ms depending on the parameters(hessian threshold, quality of the preop/intraop images). Work can be done to speed up the process via GPU acceleration.

2. Quality of SURF: Currently, we are working towards the limitation of SURF (hessian threshold to detect features). Perhaps SURF itself can be modified for such low light conditions, or some more image processing be done to increase the detail of features.

6 Management Summary

I was able to achieve the minimal deliverables of a working algorithm/process and figured a way to validate the transformation. This CISST filter was able to mark(or overlay) the result onto a pipeline shared by other filters. However, I was not able to provide a full GUI or further speedup via GPU processing, as I spent more time(3 weeks vs a few days expected) trying to understand the CISST system and debugging. The resulting CISST filter is still not coded in an optimal way: someone who wants to use my filter has to do some pointer manipulation to get to the transformation, instead of using asynchronous ports to pass data from filter to filter.

6.1 Responsibilities

This was a one-person project with help from mentors. I was able to read papers and understand what I wanted to achieve, while getting input and help from my mentors. Rogerio was a lot of help

6.2 Accomplished vs Planned

As mentioned, all of the minimal deliverables(working algorithm, validation) were achieved. Some of the expected deliverables(minimal GUI/overlay, some speedup, image processing) were done.

6.3 What I Learned

Design is important. The theoretical design of the CISST framework was good, but it was difficult to get some parts(asynchronous ports) to work properly. I had to redesign my algorithm a few times(moving the algorithm to its own filter, having a separate mouse handler/init filter), but eventually managed to make my algorithm work with Rogerio's existing Template Tracker.

References

[1] Tinne Tuytelaars Luc Van Gool Herbert Bay, Andreas Ess. Surf: Speeded-up robust features. Computer Vision and Image Understanding (CVIU), 110(3):346–359, 2008.

[2] Z.A. Pezzementi J. Handa R.H. Taylor G.D. Hager. I. Fleming S. Voros, B. Vgvlgyi. Intraoperative visualization of anatomical targets in retinal surgery. WACV, 2008.