Software for an Intra-Operative "Kinect"

Group 13: Shohini Ghosh and Elli Tian
Mentors: Dr. Austin Reiter, Dr. Russell Taylor

Table of Contents

Background

Video-guided minimally invasive surgery can reduce adverse effects for patients through the use of an endoscope that is inserted through a small incision instead of open surgery. However, endoscopy is limited in that the surgeon only sees a geometrically distorted 2D image of the surgical site. It would be desirable for the surgeon to have more accurate information on the surface topology of the surgical site from a 3D imaging system as this can guide diagnosis and surgery. For example, a collapsed airway that can be difficult to see in a 2D image could be clearly visualized using a 3D reconstruction of the space. Stereo reconstruction has been used to do reconstruction in endoscopy, but these systems may not perform as well on human tissue with few features. A structured light approach to 3D reconstruction can address this issue, as the projected light can create feature points on tissue where there were previously no distinguishable features.

By using a small camera and laser fiber that projects a pseudo-random pattern of green dots, we hope to develop software that will allow for precise and accurate 3D reconstruction of a variety of objects. Our goal is to make this software portable and flexible, so that it can eventually be adapted for real-time use during surgery with the laser fiber inserted down the working channel of a flexible endoscope.

Technical Approach

Mechanical Setup

The camera used was the Chameleon 1.3 MP Color USB 2.0 manufactured by Point Grey Research (model CMLN-13S2C-CS) with a Fujinon varifocal lens (model YV2.8x2.8SA-2) attached. The laser fiber, manufactured by 3Dintegrated, emits a 520nm light at 50mW with approximately 4 dots per square centimeter. A 3D printed frame was used to hold the camera and laser fiber so that their optical axes were parallel, as shown in Figure 1 below. The camera was 1.468 inches in diameter and the laser fiber was 0.1 inches. The frame was printed in two symmetric halves and secured with screws, nuts, and washers.

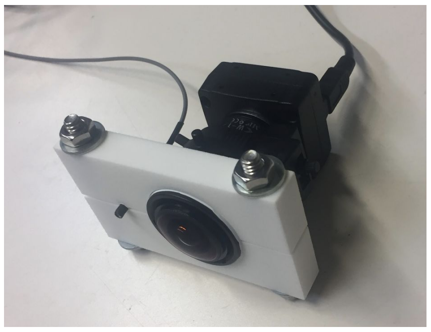


Figure 1. Mechanical setup of the camera and laser, using a 3D printed frame.

The lens was adjusted so that the focal length was maximized and a wide-angle focus was achieved. These conditions were maintained as much as possible for all subsequent images in order to prevent the need for repeated camera calibration before each image acquisition session. The aperture of the lens was sometimes adjusted manually based on the environmental conditions in order to capture the laser pattern dots as clearly as possible.

Camera Calibration

In order to calibrate the camera, 33 images of a checkerboard with known dimensions were taken. Then, a camera calibration toolbox (http://www.vision.caltech.edu/bouguetj/calib_doc/) was used to determine intrinsic camera parameters, including skew, focal length, principal point, and distortion coefficients. These parameters were used to un-distort all images collected. This calibration also indicated that distances measured from the front of the camera holder to an object were 11mm shorter than the true distances of objects from the camera origin. Thus, 11mm was added to all distances measured for training images collected for camera-laser calibration (discussed next).

Camera-Laser Calibration

Once the camera alone was calibrated, a set of training images were collected of the pattern being projected onto a surface normal to the camera's optical axis. Two data sets of training images were collected. The first set of training images was collected for planes ranging from 12-19cm from the front of the camera holder in 1mm increments (actually 131mm to 201mm); three sample calibration images are shown in Figure 2 below.

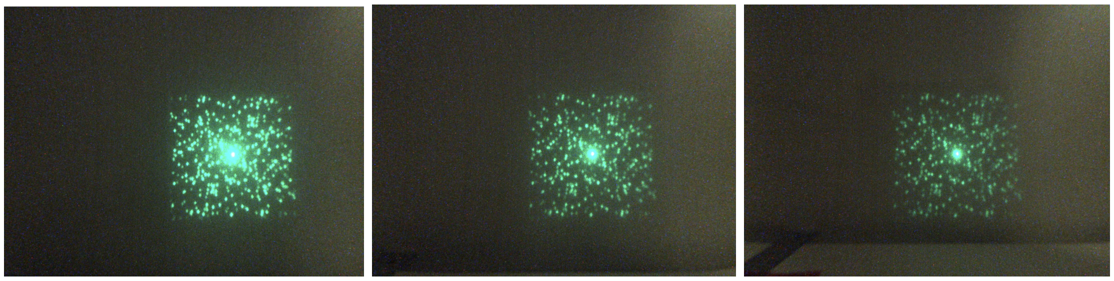


Figure 2. Calibration images taken with the laser pattern projected onto a plane at 12cm, 15cm, and 19cm from the front of the camera holder (from left to right).

The second set of training images was collected for planes ranging from 5-15cm from the front of the camera holder in 5mm increments (actually 61-161mm). For the first data set, the aperture was held constant for all images so images at shorter distances had larger dots. For the second data set the aperture was manually adjusted to get mid-sized dots in images. However, this adjustment between images was not consistent - the aperture was only adjusted when judged necessary while moving the camera from shorter to longer distances.

Dots in the laser pattern for each of these images were found by using MSER and filtering the results by size and roundness. For each data set a single template training image was selected (161mm for first data set, 111mm for second data set) to be used as the template for the dots in laser pattern. This template was used to identify dots across images based on which dot in the template training image they corresponded with. For each non-template training image, dot correspondences between the non-template training image and the template training image were identified. Correspondences were identified by first considering the translation of the center of the laser pattern between the images (the center of the laser pattern was determined to be the centroid of the brightest points in the image). This translation was assumed to be similar for all dots in the pattern, so correspondences were identified by comparing windows around each of the dots for only those dots near the expected location (as determined by the base translation). A correspondence was only used if both dots are found to be the best match for each other. Once these correspondences were established for a given non-template training image, information about the pixel coordinates of the center point of each dot in the non-template training image was stored.

Based on these data points, a graph of y pixel coordinate vs x pixel coordinate for dot centers was created, with a line to represent each dot in the pattern. Each of the points along the line represented the pixel coordinates of the center of the dot in a training image at a certain distance. Thus, this graph essentially shows the possible locations of the center point of each dot in the pattern if the dot is projected onto something at a distance within the range of training image distances. Second degree polynomials were fit to the relationship between x and y pixel coordinates for each dot center. A plot of these polynomials was stored in a 2D lookup table (see Figure 4 right image). Second degree polynomials were also fit to the relationship between x pixel coordinate and real world distance for each dot in the laser pattern.

3D Reconstruction

Dots in testing images were again found by using MSER, then thresholding by size and eccentricity. The lookup table described above from camera-laser calibration was first used to find correspondences for each dot in the testing image. This was done for each dot separately, by finding all lines in the lookup table that are nearby the location of the center of the dot in the testing image. Each nearby line represents a dot identified in the template training image and is a potential match for the current dot in the testing image because there is some training image in which the center of the dot appears near where it is seen in the testing image. If there was exactly one possible match, the dot was used to compute a 3D point. If there were no matches or multiple possible matches the dot was not used. If a match was identified, the real world z coordinate of the 3D point was computed by using the equation relating x pixel coordinate of the center of the dot and distance (since the real world z coordinate is equal to the distance). Then, using the focal length and optical axis center found from camera calibration, the x and y coordinates of the 3D position of the point relative to the camera were computed. The real world x coordinate was calculated as follows:

$$x_{real} = (x_{dot\ center,\ pixel} - x_{principal\ point,\ pixel}) * d / f_x$$

Where d represents distance calculated as explained above, f_x is the focal length along the x axis, $x_{dot\ center,\ pixel}$ is the x pixel coordinate of the center of the dot in the new image and $x_{principal\ point,\ pixel}$ is the x pixel coordinate of the principal point. The real world y coordinate was calculated equivalently.

Finding Dots in Realistic Setting

In order to evaluate how well laser pattern dots could be found in unknown, irregular, and more realistic situations, we simulated the surface imaged during an endoscopy with several pieces of ham. We then projected the laser pattern onto the surface without any other external lighting conditions in order to simulate conditions during an endoscopy when the endoscope light is turned off; this would be necessary because a bright backlight would wash out the laser pattern and make consistent dot detection nearly impossible. SIFT and MSER were used to find prominent features in the images. The time it took for each feature detector to run was determined, and the results were refined by keeping only features that were above a certain threshold for green color or within a range of radii. The results were then evaluated by counting the number of detected features that overlapped closely with actual laser pattern dots (correct matches), the number of features that did not correspond to dots (false positives), and number of dots that were not detected (false negatives).

Results/Discussion

3D Reconstruction

Dot correspondences between training images were used to determine how the (x,y) pixel coordinates of the center of each dot vary with distance. Figure 3 depicts an example of dot correspondences identified between training images, with lines drawn between matched dots. It is clear that the majority of dots, but not all dots in the laser pattern, were matched. The assumption that all dots experience similar translations as distance varies proved to be correct as there were no incorrect matches identified on manual inspection of many image pairs. One issue with this step is that all training images have dots identified based on dots in the template training image, so any dots not found by MSER in the template image are not used for computing depths. Thus if a dot was detected in a testing image that was not detected in the template image, this dot in the testing image may be incorrectly matched to a different dot in the template training image, which would produce incorrect matches and incorrect 3D points (contributes to outliers discussed later).

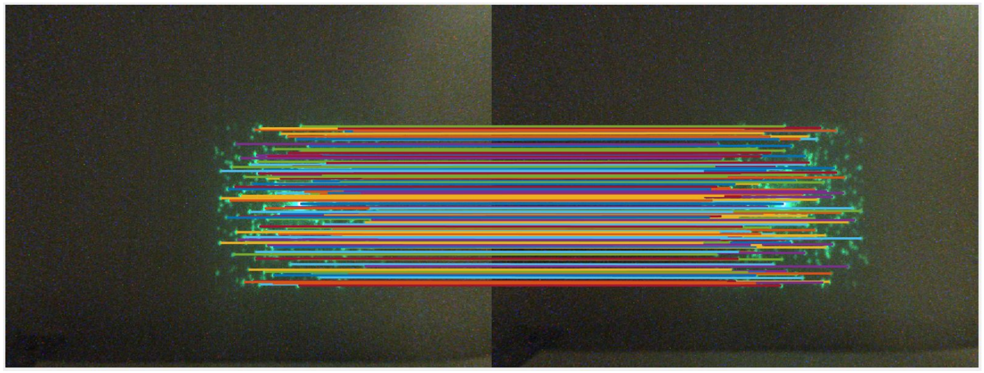
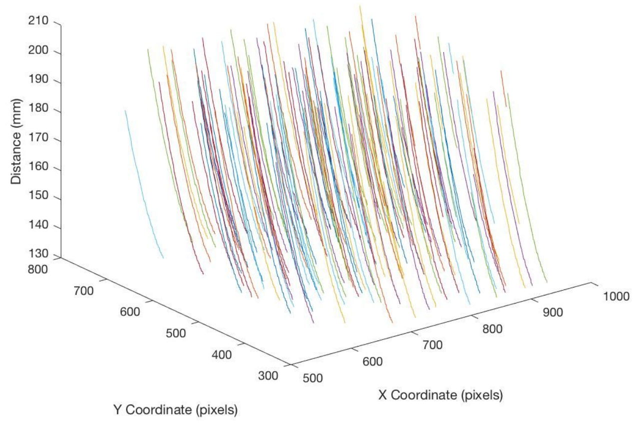


Figure 3. Dot correspondences between 151mm and 161mm images from data set 1.

These dot correspondences were then used to find how (x,y) pixel coordinates of the center point of each dot in the laser pattern varied with distance. This data gave us the 3D plot of this relationship depicted on the left in Figure 4 where each line in the plot represents a single dot in the laser pattern across images. The 2D lookup table with lines representing the possible (x,y) pixel coordinates of the center of each dot in the laser pattern is depicted on the right in Figure 4.



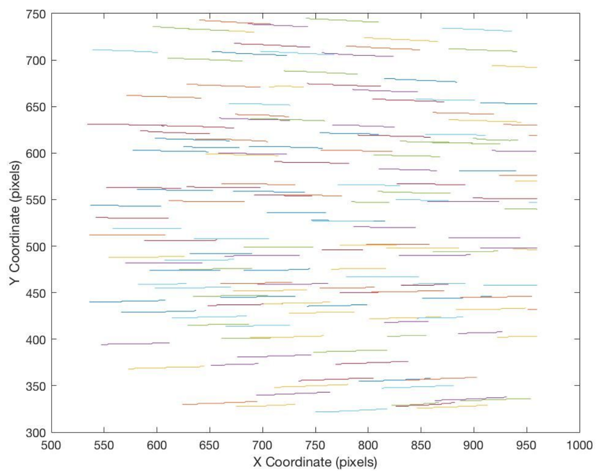
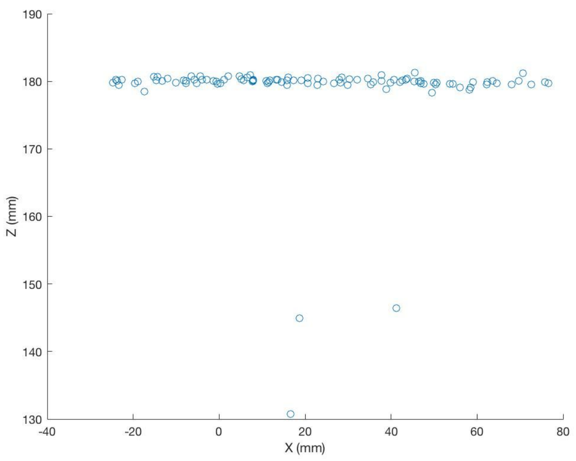


Figure 4. Left - Plot of x pixel coordinate, y pixel coordinate, distance for training data in first data set. Right - Lookup table used to identify potential matches based on data set 1. Each line represents possible coordinates for a given dot in the laser pattern if the dot is projected onto something at a distance within the range of distances for training images.

Using this lookup table and equations for each dot relating x pixel coordinate of dot center and distance as described earlier, we computed 3D point clouds based on data from various images. First, we sought to evaluate our system in a very simple way by doing a 3D reconstruction based on one of the training images. It can be seen in Figure 5 that the accuracy was generally good but that there were a few outliers. For example, the image at 180mm has 104 total 3D points generated, of which 101 were within 2mm of the true distance and 3 were farther from the true distance (97% of plotted 3D points were good matches). Figure 5 also depicts the same data with a surface fit to the points that is close to being a plane at z=180mm. These outliers may be surprising since this one of the training images, but they come from the fact that a polynomial was fit to the (x,y) pixel coordinates of each dot across images. So, if the dot in the training image at 180mm was relatively far from the polynomial fit to the (x,y) pixel coordinates across images for that dot, then it may be incorrectly matched to a different dot's polynomial fit instead if that other dot can (at some different distance) appear in a similar location in the image.



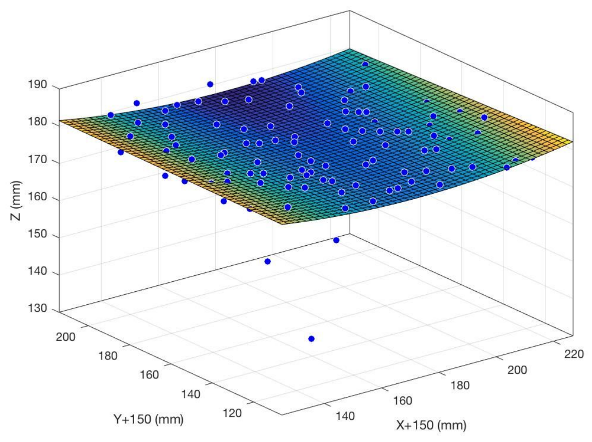


Figure 5. X and Z coordinates of 3D point cloud for training image at 180mm from data set 1 and surface fit to this point cloud in 3D coordinates.

We then sought to evaluate our system on objects with simple shapes. However, we found that our mechanical setup had likely shifted slightly since our training images were collected, as new images did not give good dot correspondences in the lookup table. Thus, we collected the second data set and on the same day collected images of simple objects on which to test our system (to minimize movement of mechanical setup between acquisition of training and testing images). However, there were significantly more outliers in this data than there were in the first data set, potentially because of the range of distances used (shorter distances produced lower quality images) and because a larger range of distances was used (so different dots are more likely to appear in the same place in the image at different distances). This difference between the data sets can be seen in Figure 6 below. For this second data set, only 78 out of 112 of the 3D points (70%) were good matches for the training image at 161mm. By manual inspection, we determined that most of these outliers came from dots identified in the new image that were not present in the template training image that were incorrectly matched to dots identified in the template training image that appeared in a similar (x,y) pixel location at a very different distance.

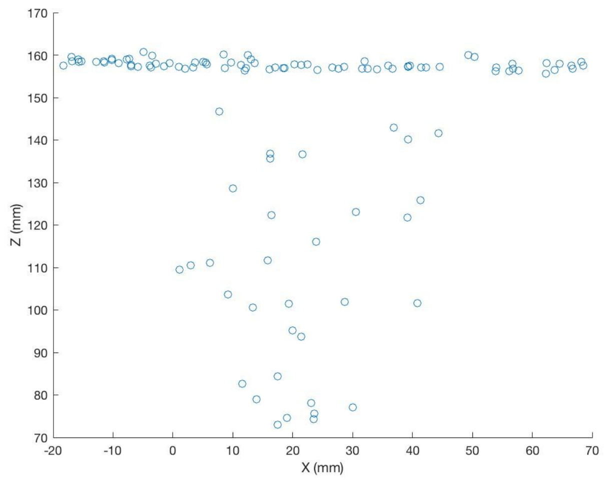
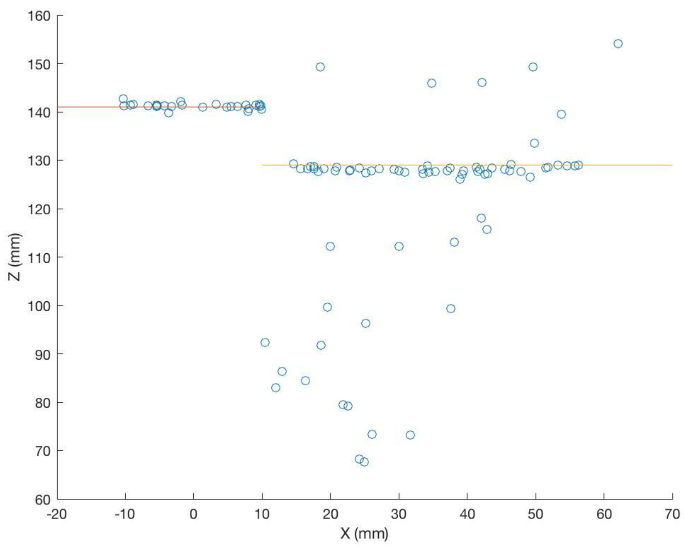


Figure 6. 3D point cloud for training image at 161mm from data set 2.

We attempted to address the large number of incorrect matches (especially in the second training data set) by finding multiple potential matches for each dot in a new testing image then comparing windows around the dot in the testing image with the window around each of the potential match dots in a training image (training image for comparison was selected based on what the estimated distance would be if that dot was the correct match), and using the best match. However, we found that this did not improve our results, and in some cases produced even more outliers. Even setting a minimum threshold for the similarity between these windows for the best match found only served to remove more correct matches than incorrect matches. Thus, we decided to carry on using our original dot matching approach.

We first tested our setup on a simple step of height 1.2cm using a thin rectangular prism, looking at the step straight on so the edge of the step was not visible from the camera. This setup was essentially 2 flat planes, with one plane at a distance of 141mm and another at a distance of 129mm. For this object, our results were generally good but there were still many incorrectly matched points. Out of the 98 points plotted, 72 were good matches within 5mm of the actual distance (73%). Our results for the step and a picture of the setup are depicted in Figure 7.



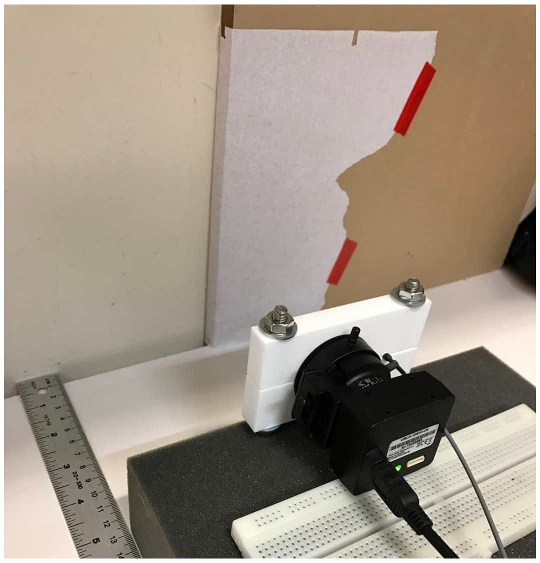
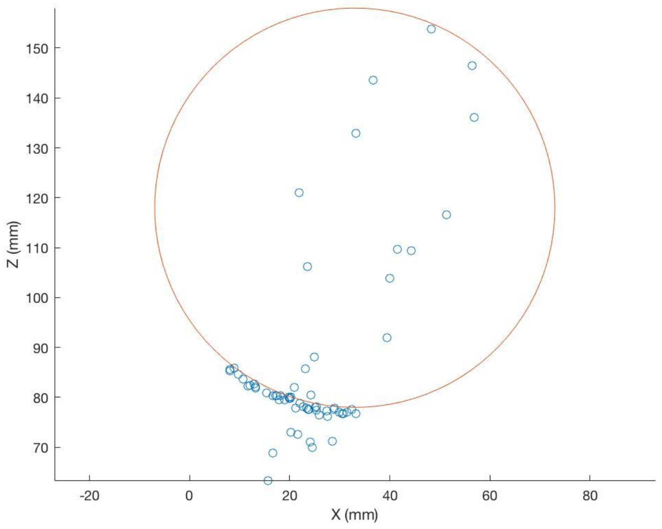


Figure 7. Real world x and z coordinates of step at distances 141mm and 129mm and image of setup.

We also evaluated our system on a cylinder with a radius of approximately 80mm placed in front of a wall at a distance of 160mm from the camera. Figure 8 depicts the data for this cylindrical object and an image of the setup. Again, there are clear outliers but 44 of the 65 3D points (68%) are within 5mm of the estimated cylinder position.



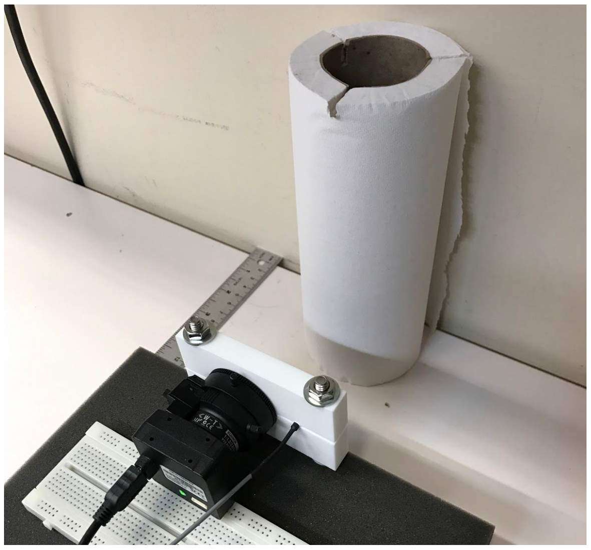


Figure 8. Real world x and y coordinates of cylinder at 80-160mm away and image of setup.

Overall, for 3D reconstruction we found that although many of the generated 3D points were correct, our system's performance was limited by dot detection and dot identification. Dot detection was an issue if a dot was found in a testing image but not in the template training image or not in a training image at the same distance as that dot appears in the testing image. When this happen, an incorrect match to a different dot may be generated, which can give a distance estimate that is completely incorrect. This is the cause of most of the outliers seen in the results above. Better dot identification could have solved this problem as these incorrect matches would be avoided by correctly identifying dots in testing images. However, dot identification is difficult when dots are not detected consistently. One way to resolve this issue may be to improve dot detection, which may in turn improve identification of correspondences if every dot can be detected in every image. For this reason, we also evaluated different methods besides MSER for detecting dots in images (discussed below). However, the best solution to this issue of identifying dot correspondences may simply be to have a sufficiently dense dot pattern to allow for window matching rather than matching of specific dots. This is similar to what the Kinect does and allows for a more dense 3D reconstruction because any window containing some laser pattern can be used to compute a depth for that window. However, this would require a significant change in hardware, so for the scope of this project we focused on software improvements to address this issue.

Finding Dots in Realistic Setting

The most effective of the feature detectors used to find the laser pattern dots was the scale-invariant feature transform (SIFT), which not only computed the locations of the features but their scale and orientation as well. Figure 9 shows an example testing setting, where ham was used to simulate the surfaces that may be imaged during endoscopy.

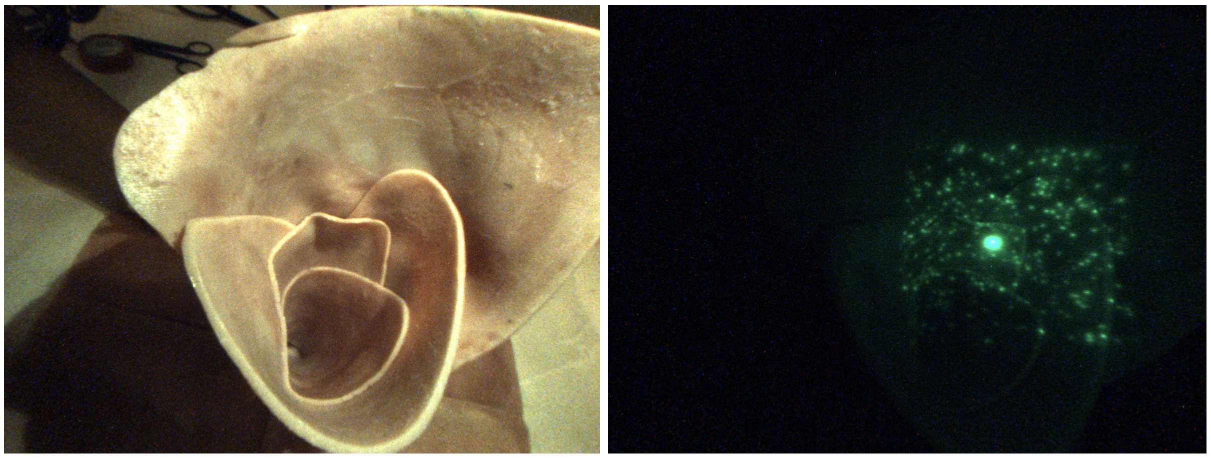


Figure 9. A sample testing setting using ham to simulate realistic esophageal surfaces (left) and the laser pattern projected onto this surface (right).

Figure 10 shows the SIFT features detected in this setting using a variety of green thresholds (g) and maximum feature size thresholds (s, in pixels). A total of 152 features are initially detected by SIFT prior to application of these thresholds and the algorithm takes approximately 1 second to run in all cases. The green thresholds chosen for this setup reflect the dimmer lighting conditions, while the small feature size threshold reflects the expected size of the laser pattern dots on the surface.

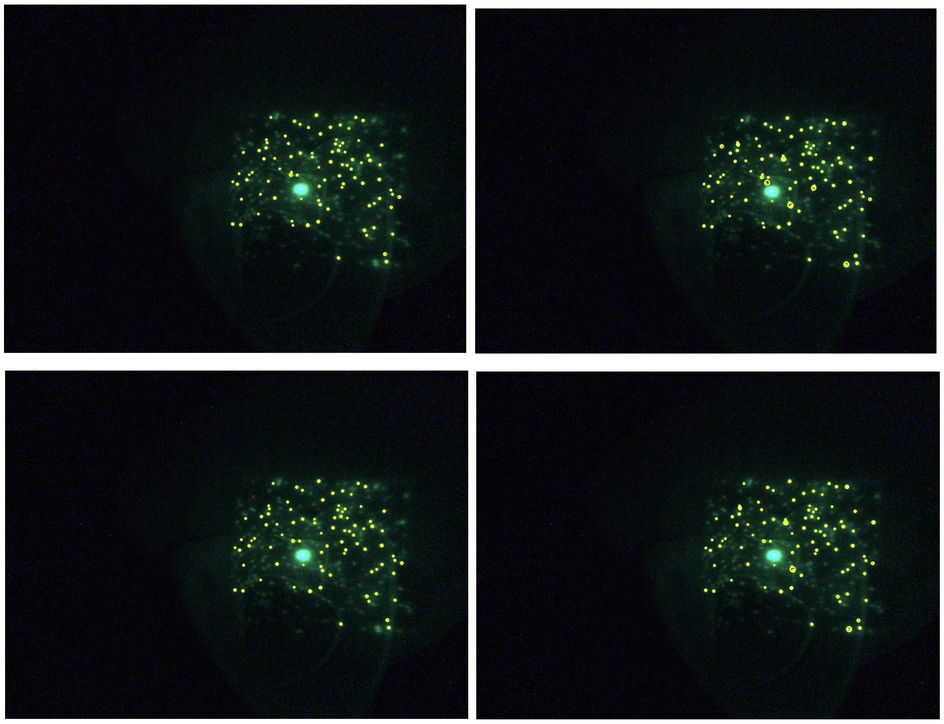


Figure 10. SIFT features detected using g=0, s=4 (top left), g=0, s=7 (top right), g=100, s=4 (bottom left), and g=100, s=7 (bottom right), where g is measured in bits and s is measured in pixels.

Maximally stable extremal regions (MSER), which identified blobs in the image, also provided relatively accurate results in some situations, but was less effective in others. Figure 11 shows MSER features that were detected in the same testing setting as above using different thresholds to remove features that were too close to each other (and thus potentially on the same laser dot). The features were also filtered by their roundness to only keep those that were relatively round, since it could be reasonably assumed that the laser dots would be approximately circular when projected onto the testing surfaces. The algorithm takes approximately 0.7-0.8 seconds to run, with larger distance thresholds corresponding to shorter running times.

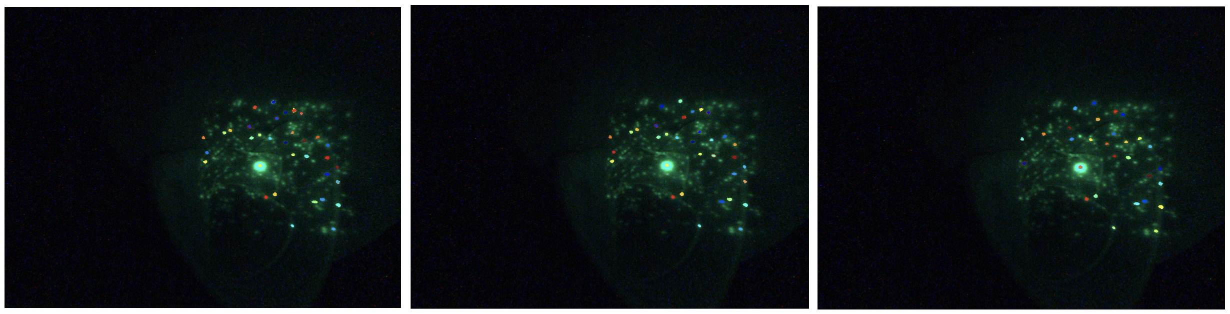


Figure 11. MSER features detected using m=10 (left), m=20 (center), and m=30 (right), where m is measured in pixels. 44, 43, and 32 features were found with each threshold respectively.

Table 1 below presents the number of correctly identified dots, the number of false positives (algorithm found dot where there was no dot) and the number of false negatives (algorithm did not find dot where there was one) for SIFT on the images above, while Table 2 presents these data for MSER. We can see that one major benefit of MSER was that there were no false positives, though the number of dots detected correctly was sometimes slightly less than the total number of features found if the threshold used for minimum distance between dots was not high enough. Because of the absence of false positives, we used MSER in our 3D reconstruction code to minimize outliers. However, SIFT detected a much larger proportion of the dots. The sum of the number of correct dots and false negative dots were different between the evaluations of SIFT and MSER because MSER detects blobs and frequently groups multiple dots into a single detected blob. However, it is primarily the ratios between these values that is important to compare.

# instances of:	**g = 0 s = 4**	**g = 0 s = 7**	**g = 100 s = 4**	**g = 100 s = 7**
Correct	99	96	81	87
False Positive	1	4	0	3
False Negative	121	124	139	133

Table 1. Performance of SIFT on image of ham with g=0 and s=4, g=0 and s=7, g=100 and s=4, and g=100 and s=7, where g is measured in bits and s is measured in pixels.

# instances of:	**m=10**	**m=20**	**m=30**
Correct	34	34	29
False Positive	0	0	0
False Negative	186	186	191

Table 2. Performance of MSER on image of ham with m=10, m=20, and m=30, where m is measured in pixels.

While the MSER results did not need to be refined with thresholds as variable as those that were used in SIFT, one can see that significantly fewer features were detected using MSER. Since the laser pattern is already relatively sparse, it would be more difficult to generate an accurate 3D reconstruction without detecting as many dots as possible. We also found that adjusting the aperture of the camera for different images affected the clarity and brightness of some of the dots, which would then affect the accuracy of the resulting detected features.

MSER did not detect many of the laser pattern dots in this particular setup, likely because of the dim lighting conditions and because some of the dots appear fainter and less defined on the surface. However, as can be seen in the results above, MSER detected enough features in some situations to allow for relatively accurate 3D reconstruction of simple objects. MSER also tended to perform better with larger dots, so the aperture could be adjusted to increase dot sizes in the image to improve the performance of MSER. In general, SIFT detected a larger number of features in more complex situations and on more realistic surfaces, especially when lighting was dim or laser points were not as clearly visible on the surface that was being imaged. In addition, the larger variety of parameters returned by SIFT could be used for more precise feature matching between images of the same scene. However, the size and greenness thresholds often had to be adjusted to achieve a desired accuracy in different situations, which would be impractical for real-time use when environmental conditions are often unknown and constantly changing. The current running time of the algorithm would also be too slow for a real-time application (although MSER is also too slow for real-time). If these settings could be further refined and optimized, SIFT would be a better choice for feature detection for real-life applications.

Management Summary

Both Elli and Shohini worked on assembly of the mechanical setup for the camera and laser, and worked on collecting images for camera calibration and training images for calibration of the camera-laser system. Elli focused on dot detection using SIFT, and Shohini focused on dot detection using MSER. Shohini was primarily responsible for static 3D reconstruction on the detected point clouds, and Elli looked into dynamic 3D reconstruction using a possible variety of open-source SLAM packages. However, this dynamic 3D reconstruction was not implemented due to the difficulties encountered with obtaining an accurate static reconstruction.

Initially, our minimum goal was to be able to produce static 3D reconstructions of a variety of objects with at least millimeter accuracy, reflecting the small scale of the structures that might be encountered during an endoscopy. We also hoped to implement a simple SLAM or point cloud matching algorithm to allow for real-time 3D reconstruction of simple objects, such as cylinders, inclines, or rectangular prisms as our expected deliverable. If all of this was completed, we hoped to be able to use a similar algorithm for real-time 3D reconstruction of complex objects and more realistic situations as a maximum deliverable. Due to some difficulties encountered with identifying correspondences between laser point dots in the images, however, we were only able to produce static 3D reconstructions that were not quite within the millimeter accuracy that we hoped to achieve. Overall, the generated point clouds could be fitted to a surfaces that were very similar to the objects we were trying to image, but they almost always contained outliers that did not reflect the actual surface structure. Without the desired accuracy in these static 3D reconstructions, we were unable to move on to implementing an algorithm for real-time 3D reconstruction.

While we looked into several open-source SLAM packages and also researched writing our own using an iterative closest point (ICP) process, we also had some difficulties with finding an algorithm that would work with our data. Many SLAM algorithms assume that at least some information is known about the environment, such as prominent landmark locations or depth information obtained from a stereo camera setup, but we were only able to produce static reconstructions as 3D point clouds from our laser pattern images. Therefore, a successful implementation of SLAM with our data would rely even more heavily on accurate point and feature matching between successive images of a scene.

In order to continue developing software that can be used for live reconstruction during minimally invasive surgery, a more efficient dot detection algorithm is needed. This would involve a variety of factors, most notably figuring out how to quickly determine appropriate thresholds for size and green amount for different environmental conditions in order to refine the results obtained from any feature detector used. In addition, the range of workable distances would need to be increased in order to account for more complex topographies, which would require an algorithmic or automated method for adjusting the aperture on the camera in order to bring the laser point dots into focus and appear around the same size.

Additional hardware adjustments could also be made to improve the reconstruction process. A huge barrier to achieving the desired accuracy with our particular setup was how sparse the dots produced by the laser fiber were. Combined with inconsistent dot detection from SIFT or MSER, it was difficult to obtain a smooth and reliable static 3D reconstruction. A denser laser pattern, like that produced by the Microsoft Kinect, would improve our reconstruction results because it would allow for matching of windows of regions of the laser pattern rather than matching of individual dots between images. An infrared laser pattern could also be used instead of green light in order to get RGB images of the environment and infrared images of the pattern at the same time, without dealing with lighting that can cause glare or inaccurate dot detection.

After all of this is accomplished, improvements could be made to the space and time complexities of the code, and it could be combined with a SLAM or point cloud matching package for real-time 3D reconstruction. A denser reconstruction (based on a window-matching approach or a significantly denser dot pattern) would allow for better integration with SLAM. In addition, successive images generated by SLAM could be used to further refine the initial depth map, eliminate outliers, resolve discontinuities, and overall generate a denser reconstruction of the environment. Finally, the software could be adapted to work with the laser fiber inserted down the working channel of an endoscope and tested in a more realistic situation.

From this project, we learned that determining a reliable camera-laser calibration with the given setup is more difficult and crucial than expected. We had to assume many things for our experiments that may not be true during a real-life application, including that the focus of the camera would not change and that the imaged scene would be contained within a specific distance range. Even with these assumptions, our results for feature detection were neither as accurate nor as thorough as we would have liked, and as a result we were unable to move ahead with some of our planned deliverables. However, from our research and discussions with our mentors, we believe that it will be possible to produce real-time 3D reconstructions using SLAM once a static reconstruction with the desired accuracy can be obtained. With improvements to efficiency and accuracy of this process, it can hopefully be adapted to work with a variety of camera-laser setups in more realistic situations and applications.

Technical Appendices

Appendix A: Code Repository Link

All code developed for this project, documentation, and images collected can be found in the BitBucket repository at https://bitbucket.org/etian1/3dkinect.

Appendix B: Description of Files

training.m
This script uses training images at known distances from either the new or old data set to find correspondences between dots in each training image with dots in the template training image. The locations of these dots that have been matched to dots in the template training image are stored in the matrix trainingDots. Each row in training dots represents an image/distance, each column represents a specific dot found within the template image, and each entry contains the (x,y) coordinates of the relevant dot if a correspondence was identified.

buildLookupTable.m
This script populates the matrices lookupTable and xDistLines based on the results from the script training.m about the locations of each dot in the laser pattern in each image (stored in trainingDots). The matrix lookupTable has rows representing x pixel coordinate, columns representing y pixel coordinate, and each entry is a list of dot indices for dots that are centered at that (x,y) pixel coordinate at some distance. The matrix xDistLines stores the coefficients of the polynomial fit to the relationship between x pixel coordinate and distance.

makeDepthMap.m
This script uses data from training.m and buildLookupTable.m about the relationship between (x,y) pixel coordinates and distance for each dot in the laser pattern from the training data to produce a 3D point cloud based on a new image. Dots in the new input image are matched to dots in the template training image based on their location within the image (based on which dots could potentially be in that location). This is done using the lookupTable matrix produced by buildLookupTable.m. Distance is calculated based on xDistLines produced by buildLookupTable.m.

getDots.m
This function takes in an RGB image, and T/F parameters. When findCenter is true the function will make the first center point that of the dot at the center of the laser pattern. When eliminateDup is true, no two dots that center points within minInterDotDist of each other will be included in the returned list. When returnPixelLists is true the function will also return a list of the pixel coordinates for each pixel within the dot for each dot found.

getDotsSIFT.m
This function takes in an image file and calculates SIFT keypoints and descriptors using the vl_sift package (http://www.vlfeat.org/overview/sift.html). The results are thresholded with a peak selection threshold PeakThresh, non-edge selection threshold EdgeThresh, a maximum pixel radius maxRad, and green amount greenThresh. The number of keypoints found and the time taken for the code to run are outputted to the command line, along with additional information about the SIFT process.

matchDots.m
This function takes in two images and the coordinates of the center points of the dots in the laser pattern in each of these images. The dots may be in different orders.

This function returns a list of matches between dots in the test and template images. Each row of tmplMatches represents a match with the dot stored in the same row number in centersTmpl. The value stored within a given row in tmplMatches represents the index of the dot in the test image (within centersTest) that was matched to the dot in the template image.

windowComp.m
This script computes and returns the "similarity" in terms of the specified metric (ncc for normalized cross correlation or ssd for sum of squared differences) between a window in image1 centered at center1 with radius windowRad and a window in image2 centered at center2 with radius windowRad. If the window does not fit within either image, a value of 0 will be returned as the windows can't be compared.

correctDistortion.m
This function corrects the lens distortion in a series of images contained in a folder with path imgsPath and calibration parameters contained in the file calibFile. Undistorted images are outputted to the folder with path outPath.