Robot-assisted steady ultrasound imaging and visual servoing enabled by deep learning

CIS II

Group 7
Tian Xie

Mentors: Dr. Emad Boctor, Dr. Mahya Shahbazi

May 2019

Contents

1 Introduction

A tumor biopsy is a procedure to remove sample cells from the lesion for the diagnose of cancer. During biopsy, ultrasound imaging is always used to guide the pass of the needle. The sonographer will hold and press the probe against the patient to gain a view of the lesion. Then the doctor can insert a needle and take samples. It is a common practice to have a pathologist in room during the biopsy, who will immediately examine the sample to see if it is diagnosable. If the sample acquired is not informative, then the procedure needs to be repeated. A steady ultrasound image should be provided throughout multiple acquisitions, which makes a biopsy a very tedious and cumbersome task for sonographers because they have to hold and press the probe by keeping a static pose. Musculoskeletal injuries become a severe problem for sonographers because they have to do the same work for many patients one day and every day.

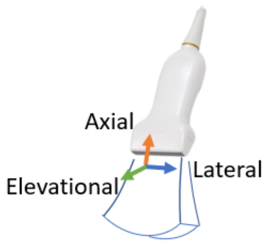
Therefore, the development of a robot-assisted system to complete this tedious task for sonographers will be of great significance. In this project, a pipeline is developed to first track in-plane motion with an image region tracking algorithm (Hager and Belhumeur, 1998)[1]. Then deep learning method is used to estimate out-of-plane motion patch-wise. With the estimation of SE(3) transformation between two neighboring images, visual servoing can further be integrated into the control loop to control a UR5 robot arm to which the ultrasound transducer is attached.

In this report, a brief literature review is provided in Section 2. Then the technical approaches are described in details in Section 3. Description of experiments and results are included in Section 4. Management including execution and future plans are discussed in Section 5.

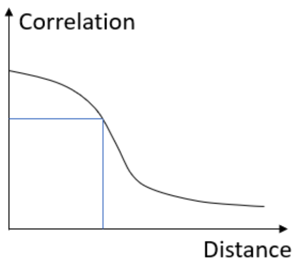
2 Background

A large portion of this project focuses on finding out the SE(3) transformation between two ultrasound images. The coordinate of an ultrasound image is defined in Fig.1(a). The in-plane motion includes two translations (axial and lateral) and one rotation. And the out-of-plane motion includes two rotations and one translation (elevational). In-plane motion can be estimated via registration or image region tracking algorithms. However, it is hard to estimate the out-of-plane motion. Speckle decorrelation is a mostly used method to estimate the elevational translation[2]. Speckles are the granular appearance in ultrasound images which are generated by the interference of scatterers in a resolution cell. Because of the poor resolution along the elevational direction, there is correlation between corresponding patches in two neighboring ultrasound images. A decorrelation curve can be obtained using a calibration phantom with fully developed speckles (see Fig.1(b)). The Gaussian shaped curve can then be used to estimate elevational translation. However, this method works on the assumption that all speckles are fully developed speckles. Some previous studies [3],[4] used the first order statistical properties of RF signals under k-distribution to detect regions with fully-developed speckles. However, these methods all require some heavy computation of RF signals. They can hardly be used directly for visual servoing in real time.

The rapid development of Convolutional Neural Network (CNN) can provide a powerful solution to estimate the elevational translation. CNN is widely used in fields like image recognition, segmentation, etc. In this project, a regression problem, predicting the elevational translation, needs to be solved. One study [5] proposed using deep learning to estimate elevational translation. But their result is not convincing and they do not make use of speckles even though the paper talked a lot about speckle correlation.



(a) Ultrasound coordinate



(b) Speckle decorrelation curve

Figure 1: Ultrasound coordinate and speckle decorrelation

3 Technical approaches

3.1 In-plane motion

The original plan was to estimate 6 degrees of freedom at the same time by end-to-end deep learning. A standard two-input-channels CNN, the Siamese Network[6] and multitask training were explored for end-to-end learning. However, none of the models are able to estimate in-plane motion without severe over-fitting. Therefore, conventional computer vision method was used instead to estimate in-plane motion.

The image region tracking algorithm presented by Hager and Belhumeur (1998) is used to estimate the in-plane rigid body transformation. A brief overview of this algorithm is provided below. The algorithm is based on image constancy assumption that at any time t, there should be a motion parameter $\boldsymbol{\mu}(t)$ such that $I(f(\mathbf{x};\boldsymbol{\mu}(t)),t) = I(\mathbf{x},t_0)$, where t_0 is the initial time. The motion parameter μ can be estimated by minimizing an objective function $O(\mu) = \Sigma(I(f(\mathbf{x};\boldsymbol{\mu}(t)),t) - I(\mathbf{x},t_0))^2$ for all x's in the region of interest (ROI). The motion parameter vector is $[u, v, \theta]$, where u is the lateral translation, v is the axial translation and θ is the in-plane rotation about the elevational axis. The algorithm is used to compute a motion parameter vector that minimizes the difference in brightness of the ROI.

When using UR5, because axial force exerted on the phantom is not controlled in this project, axial translation and in-plane rotation are not considered for visual servoing. Therefore, for simplicity, in-plane rotation is not included in the calculation. The motion model can be simplified as $f(\mathbf{x};\boldsymbol{\mu}) = \mathbf{x} + \mathbf{u} = [x, y] + [u, v]$.

$$f_{\mathbf{x}} = [\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}]$$

$$f_{\boldsymbol{\mu}} = [\frac{\partial f}{\partial u}, \frac{\partial f}{\partial v}]$$

The Jacobian matrix at time t_0 used in the algorithm can be written as, $M_0 = [\mathbf{I}_x(t_0), \mathbf{I}_y(t_0)]$, where $\mathbf{I}_x$ and $\mathbf{I}_y$ are the x and y gradients for each pixel in ROI. Then for each time step, a $\delta\boldsymbol{\mu}$ can be calculated. The vector of motion can be further calculated.

$$\delta\boldsymbol{\mu} = -(M_0^T M_0)^{-1} M_0^T (I(\mu, t+\delta t) - I(0, t_0))$$

$$\boldsymbol{\mu}(t+\delta t) = \boldsymbol{\mu}(t) + \delta\boldsymbol{\mu}$$

Adequate convergence is obtained when $\| \delta\boldsymbol{\mu} \|_2$ is smaller than a certain threshold. When finding a small in-plane translation, I calculate $\boldsymbol{\mu}$ for the windows near ROI and pick the window with the smallest $\boldsymbol{\mu}$ out of all $\boldsymbol{\mu}$'s smaller than the threshold. Finally, the in-plane motion is scaled from the pixel coordinate to the world coordinate.

3.2 Out-of-plane motion

Deep learning is used to estimate the out-of-plane motion. The input of the model is two corresponding patches on two neighboring images.

The structure of the model is shown in Fig.2. The input size is 60x100x2 (HxWxC). The two channels are the two successive images. The ground truth is the elevational translation and a logcosh function is used as the loss function. Logcosh cost function gives a better performance compared with using mean absolute error or mean squared error. Huber loss with $\delta = 0.05$ is also experimented in the project. It is not as good as logcosh. Adam is used as the optimizer. Batch normalization and ReLU are used after each convolutional layer. L2 regularization is used in fully connected (FC) layers. Dropout is applied to FC layers to prevent overfitting. The batch size is 128.

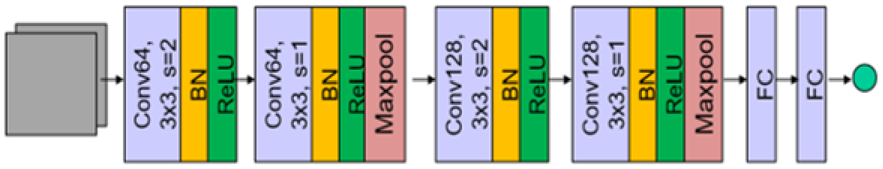


Figure 2: CNN architecture

Different input sizes are also tested. The training results with different input image sizes are very close. An architecture similar to Siamese network is also experimented in the project. It has a comparable performance but it takes a longer computational time. Therefore, it is not used in the project for real-time control.

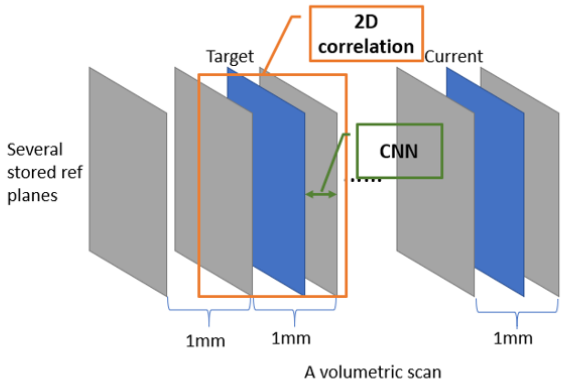


Figure 3: Estimation of large motion with reference planes and small motion with CNN

The model depends on speckle features.The correlation of speckles drops to a very low level beyond 1 mm, so the model can only estimate distance within 1 mm. To increase the range of estimation, 2D correlation coefficient is used to first bound an image between two reference planes. A volume scan can be obtained beforehand, from which a few reference planes can be chosen. The smaller the separation between two images, the larger 2D correlation coefficient the two images have. Thus, it is easy to have a coarse estimation about the location of an image (between two images with the highest 2D correlation). However, it involves heavy computation to calculate 2D correlation coefficient. Hence, the number of reference planes should be kept small.

$$corr2 = \frac{\Sigma\Sigma(A_{mn} - \bar{A})(B_{mn} - \bar{B})}{\sqrt{(\Sigma\Sigma(A_{mn} - \bar{A})^2)(\Sigma\Sigma(B_{mn} - \bar{B})^2)}}$$

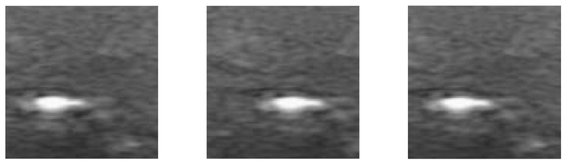
4 Experiments and results

4.1 Test bed setup for data acquisition

A test bed is designed to acquire video captured B-mode ultrasound images and ground truth transformation. The images are obtained using a SonixTouch machine with a L14-5W linear array. The frequency is set to be 10 MHz, and the depth is set to be 4.0 cm. The B-mode images used are the screen shot of the video. For data with pure translation along one direction, the motion is precisely controlled by a linear stage and a dial indicator (precision: 0.001 mm). For data with more than 1 DoF motion, the transformation is read from the UR5 encoder. And calibration using a cross-wire phantom is completed to find out the unknown transformation of the image coordinate with respect to the UR5 end effector. A CIRS elasticity phantom is used for data collection.

4.2 In-plane motion

The algorithm for tracking in-plane motion is tested on 26 images with 1mm step size. For these 25 pairs of images with 1mm lateral translation, the algorithm successfully estimated 18 pairs with an absolute error less than 2 pixel values (i.e. about 0.2 mm). 20 pairs of images with both 0.2mm elevational translation and lateral translation ranging from 0.2mm to 2mm was tested as well. All 20 estimations have absolute errors less than 2 pixel values. The accuracy of this algorithm will increase if there are obvious anatomical features in the image because they have clearer edges. Some results of in-plane tracking are shown in Fig.4. The first figure in each group represents a ROI in the original image. The second figure shows the transformed image in the same bounding box. The third figure shows the region after the lateral translation is compensated.


(a) Pure lateral translation

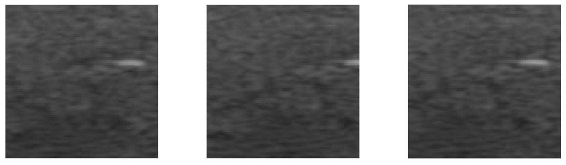

(b) Elevational translation plus lateral translation

Figure 4: In-plane motion estimation

4.3 Out-of-plane motion

A CNN model is trained on patches of size 60x100. The training set is a group of images on the phantom along the short edge. The training set contains about 18,000 pairs of images (Fig.5). The validation set (1283 pairs) and the test set (1152 pairs) are groups of images scanned along the long edge of the phantom. These three sets are from different regions on the phantom. The model checkpoint is based on its performance on the validation set. The mean absolute percentage error of the validation set is around 6.9% while that of the test set is around 7.8%.

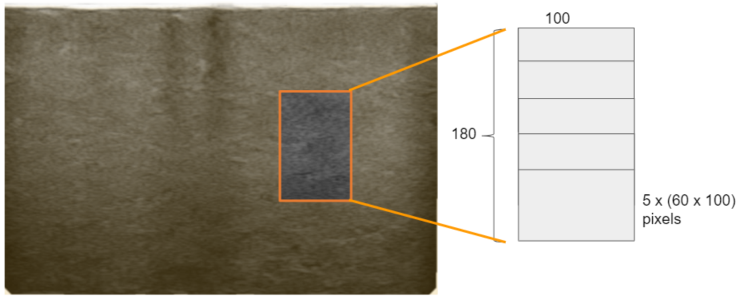


Figure 5: Training set images

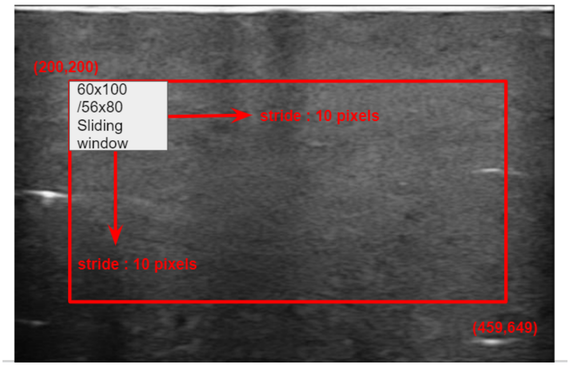


Figure 6: Patch-wise estimations using the CNN

Then the trained CNN model is applied patch-wise. A region with size of 260x450 pixels is considered in the center of the whole image. And a sliding window (60x100, CNN input size) with 10-pixel strides along x and y directions is used to generate 756 patches for one image (see Fig.6). The CNN model is utilized to estimate the elevational translation for each patch given a pair of images. Table 1 below shows the median, mean and variance of the 756 estimations for one group of images with step size equal 1 mm in the data set. The mean absolute percentage error is calculated as $\frac{\|median-true\|}{true} * 100\%$. Fig.7 shows the box plot of this group of images. The blue line in the figure is the ground truth. Table 2 shows the result for a group with 0.05mm step size. The mean absolute percentage error is small for distances between 0.2 mm to 0.7 mm, which corresponds to the characteristics of the Gaussian curve, because correlation spreads out in this range. Similar performance is observed for other groups of images.

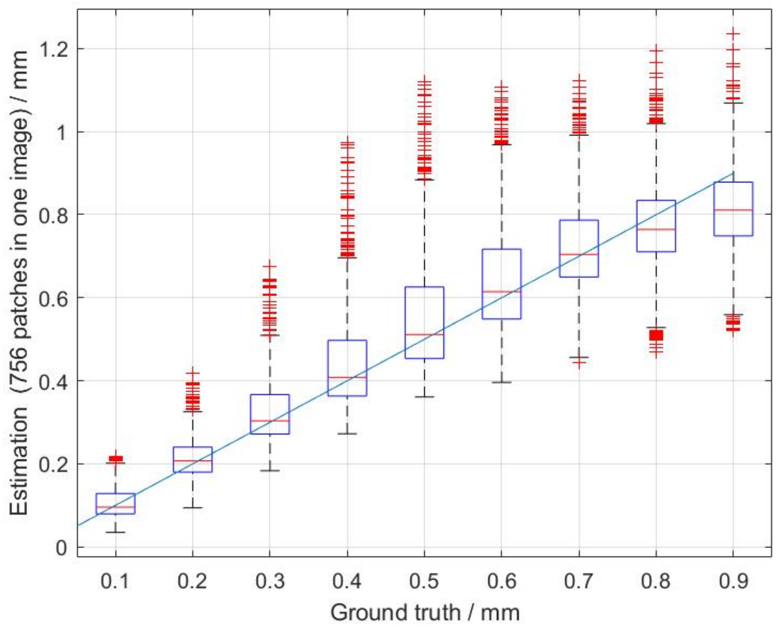
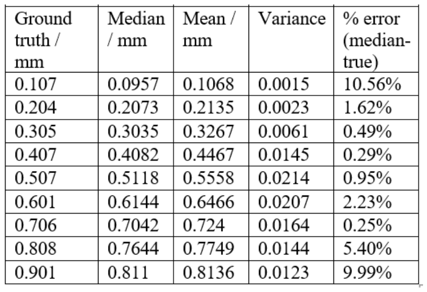
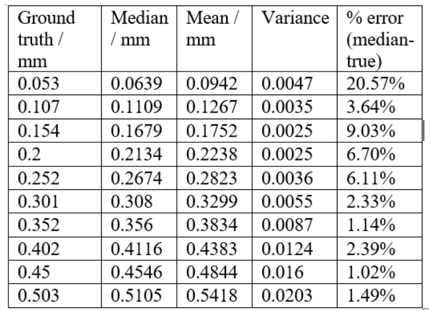


Figure 7: Ground truth vs estimations of 756 patches



Ground truth / mm	Median / mm	Mean / mm	Variance	% error (median-true)
0.107	0.0957	0.1068	0.0015	10.56%
0.204	0.2073	0.2135	0.0023	1.62%
0.305	0.3035	0.3267	0.0061	0.49%
0.407	0.4082	0.4467	0.0145	0.29%
0.507	0.5118	0.5558	0.0214	0.95%
0.601	0.6144	0.6466	0.0207	2.23%
0.706	0.7042	0.724	0.0164	0.25%
0.808	0.7644	0.7749	0.0144	5.40%
0.901	0.811	0.8136	0.0123	9.99%

(a) Table 1
Range: 0.1-0.9 mm,
step size: 0.1mm

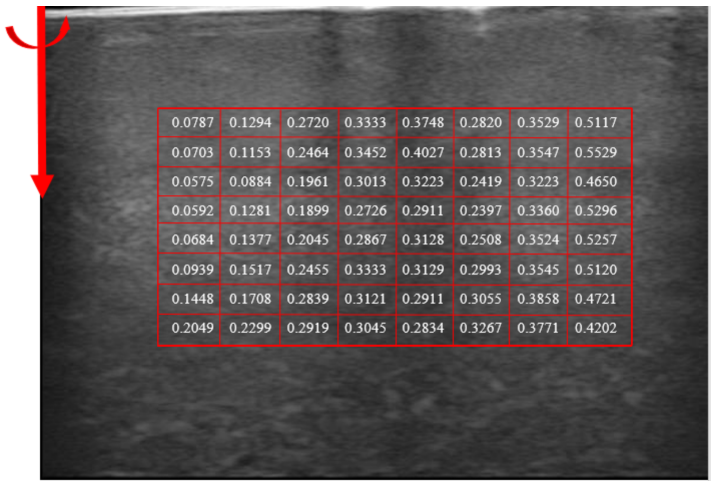


Ground truth / mm	Median / mm	Mean / mm	Variance	% error (median-true)
0.053	0.0639	0.0942	0.0047	20.57%
0.107	0.1109	0.1267	0.0035	3.64%
0.154	0.1679	0.1752	0.0025	9.03%
0.2	0.2134	0.2238	0.0025	6.70%
0.252	0.2674	0.2823	0.0036	6.11%
0.301	0.308	0.3299	0.0055	2.33%
0.352	0.356	0.3834	0.0087	1.14%
0.402	0.4116	0.4383	0.0124	2.39%
0.45	0.4546	0.4844	0.016	1.02%
0.503	0.5105	0.5418	0.0203	1.49%

(b) Table 2
Range: 0.05-0.5 mm,
step size: 0.05mm

Figure 8: Results of two groups of images summarized in tables

For each patch, we know its center is $[x_i, y_i, 0]$ in the image coordinate. In the transformed image, the corresponding patch is $[x_i, y_i, \Delta z]$. Thus, out-of-plane rotations can also be found out by using least median squares to fit all the patches. Given the limited time, this part has not been completed so far. But to prove the feasibility of this idea, I tested a pair of images with rotation along the axial axis (see Fig.9. The origin of the image coordinate is the left upper corner). Instead of least median squares, I performed linear fitting for patches on the same row, and the angle of rotation is the arc tangent value of the slope. The ground truth is 0.5 degree while the coarse estimation is 0.602 degree.



0.0787	0.1294	0.2720	0.3333	0.3748	0.2820	0.3529	0.5117
0.0703	0.1153	0.2464	0.3452	0.4027	0.2813	0.3547	0.5529
0.0575	0.0884	0.1961	0.3013	0.3223	0.2419	0.3223	0.4650
0.0592	0.1281	0.1899	0.2726	0.2911	0.2397	0.3360	0.5296
0.0684	0.1377	0.2045	0.2867	0.3128	0.2508	0.3524	0.5257
0.0939	0.1517	0.2455	0.3333	0.3129	0.2993	0.3545	0.5120
0.1448	0.1708	0.2839	0.3121	0.2911	0.3055	0.3858	0.4721
0.2049	0.2299	0.2919	0.3045	0.2834	0.3267	0.3771	0.4202

Figure 9: Rotation about y axis

Given the characteristics of speckles, the CNN model can only predict elevational translation within 1mm. Therefore, to increase the range of estimation, some other method is needed. First, several reference planes with a separation about 1 mm can be stored beforehand. Then compare the 2D correlation between the current image and the reference planes to bound the current view between two reference planes. Then CNN model can be used for fine estimation. For images in my data set, the 2D correlation coefficient stays above 0.5 when the separation is less than 0.5mm and it drops below 0.2 beyond 1mm. Therefore, this approach is valid to give a rough estimation of large motions. Then the CNN model can be used as fine tuning.

Predicting the distance for 756 patches takes less than 1 second on a standard computer with an NVIDIA GeForce GTX1050Ti GPU. It can be faster in real-time control because it is not necessary to generate so many patches.

4.4 Region detection

In Fig.7, some outliers can be seen from the plot. The regions of these outliers are labeled in the original image. Based on the 756 patches described in the previous subsection, a 10x10 region around the center of each patch is labeled with $\| \hat{z} - z_{true} \|$. As shown in Fig.10(a), the pink regions are the regions with large estimation errors. Surprisingly, these pink regions match some "bad" regions in the ultrasound image. For example, in the middle, that region is dark due to the cracks on the transducer. A mask can be generated by smoothing out the boundary of regions with error great than 18% (assume an error below 18% can be intrinsic error of the CNN estimation). The mask can be further used to detect fully-developed speckles in the image.

(a) Original image labeled with abs error (b) Mask for "bad" regions

Figure 10: Region detection

4.5 1 DoF visual servoing

Visual servoing for lateral translation is implemented. The control is not really real-time at current stage. The ultrasound software stores one image per second with a time stamp. And a MATLAB program reads the images every 5 seconds and sends commands to UR5 via TCP/IP to compensate the motion. Because two programs are running on the same computer, so both the image and the robot pose are time-stamped with the same system clock. A short video demo is available on the project website.

5 Management summary

The project is done by one person. Thus there is no management of work distribution in this project.

The scheduled minimum deliverable is the setup of a test bed and the acquisition of data. It is met in mid April. Training data for elevational translation was collected using a linear stage and dial indicator. Some data with 6 DoF motion was collected with the UR5 after calibration. The expected deliverables include a CNN model to estimate elevational translation and the application of an image region tracking algorithm to find the in-plane rigid transformation. These two tasks have been accomplished, and a pipeline to combine these two methods is developed. In-plane transformation is first calculated and then out-of-plane motion. The maximum deliverables include the estimation of out-of-plane rotation and visual servoing for 1/2 DoF. The approach to calculating out-of-plane rotation is being developed and the evaluation is still in progress. And motion compensation for lateral translation is completed, although with a time lag.

I will follow on this project and extend it to my master thesis. There are several problems that need to be solved in the following steps. First, a pipeline to combine all six degrees of freedom needs to be developed and evaluated. Second, using deep learning to detect fully developed speckles is another direction that worth trying. Cross-validation with baseline methods is needed. Third, communication and synchronization problems need to be solved for real-time visual servoing. Finally, the generalization of this method to real tissues should be done.

From this project, I learned certain knowledge of ultrasound imaging. I studied and used deep learning methods to solve problems. I also got hands-on experience with calibration and synchronization problems that I have met in CIS I homework. The progress of this project is not always successful. For example, in our original proposal, end-to-end deep learning for all 6 degrees of freedom was considered. However, I met with severe overfitting problems for in-plane motion. Every step in this project contains unknowns and uncertainties, but I have fun getting involved in the research.

The data acquired has been uploaded to JH box. Part of the codes have been uploaded to my private repository on GitHub. Other relative materials can be found on the project website.

6 Conclusion

The project uses deep learning to estimate the out-of-plane motion with high accuracy, especially for distances from 0.2 mm to 0.7 mm. The range of estimation can be increased by using reference planes. It is combined with a conventional region tracking algorithm which predicts in-plane rigid body transformation. A demo of simple in-plane motion compensation is also completed in this project. The development of this system brings a promising solution to provide steady ultrasound imaging for cancer biopsy. However, more future studies are needed to improve and generalize the estimation of SE(3) transformation. Also, there are still many problems to solve before realizing real-time 6 DoF visual servoing.

7 References

References

[1] G. D. Hager and P. N. Belhumeur. *Efficient Region Tracking With Parametric Models of Geometry and Illumination.* IEEE Transactions on pattern analysis and machine intelligence, Vol. 20, No. 10, October, 1998.

[2] J-F. Chen, J.B. Fowlkes, P. L. Carson, and J. M. Rubin. *Determination of scan-plane motion using speckle decorrelation: Theoretical considerations and initial test.*. International Journal of Imaging Systems Technology, 8:38–44, 1997.

[3] H. Rivaz, E. Boctor, and G. Fichtinger. *Ultrasound speckle detection using low order moments.* IEEE Int. Ultrasonics Symp., pp. 2092–2095, October 2006.

[4] R. Prager, A. Gee, G. Treece, and L. Berman. *Analysis of speckle in ultrasound images using fractional order statistics and the homodyned k-distribution.* Ultrasonics, vol. 40, pp. 133–137, 2002.

[5] R. Prevost, M. Salehi, J. Sprung, R. Bauer, and W. Wein. *Deep Learning for Sensorless 3D Freehand Ultrasound Imaging.* Medical Image Computing and Computer-Assisted Intervention, MICCAI, 2017.

[6] Chopra, Sumit, Hadsell, Raia, and LeCun, Yann. *Learning a similarity metric discriminatively, with application to face verification.* . In Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on, volume 1, pp. 539–546. IEEE, 2005.