# A 3D MODEL OF HUMAN EJACULATORY DUCTS

Chathuri Lakshani Gunasekera

(138049A)

Degree of Master of Science

Department of Electronic and
Telecommunication Engineering.

University of Moratuwa.
Sri Lanka.

May 2017

# A 3D MODEL OF HUMAN EJACULATORY DUCTS

Chathuri Lakshani Gunasekera

(138049A)

Thesis submitted in partial fulfilment of the requirement for the degree of Master of Science by Research

Department of Electronic and Telecommunication Engineering.

University of Moratuwa.
Sri Lanka.

May 2017

## DECLARATION OF THE CANDIDATE AND THE SUPERVISOR

“I declare that this is my own work and this thesis/dissertation does not incorporate without acknowledgment any material previously submitted for a Degree or Diploma in any other University or Institute of higher learning and to the best of my knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text.”

Also, I hereby grant to University of Moratuwa the non-exclusive right to reproduce and distribute my thesis/dissertation, in whole or in part in print, electronic or other medium. I retain the right to use this content in whole or part in future works (such as articles and books)”.

Signature: Date:

The above candidate has carried out research for the Master’s thesis/Dissertation under my supervision.

Signature of the Supervisor: Date:

# DEDICATION

To my husband, my parents and my brother…

# ACKNOWLEDGMENTS

I would like to express my deepest appreciation to all those who provided me the possibility to complete this dissertation.

Foremost, I would like to express my deepest thanks to my supervisor, Dr. Nuwan Dayananda for his patience, encouragement and comments which contributed immensely to this dissertation. It is a pleasure to have known him and I consider myself lucky for having been his student. I would also like to thank all the faculty members in the department of Electronic and Telecommunication Engineering, University of Moratuwa, who helped me in various aspect.

Furthermore, I would also like to thank Dr. Ajith Malalasekera for coming up with this project idea and for his valuable comments. In addition, I would like to thank Dr. Y. Mathangasinghe and Dr. D.N. Weerakoon for their valuable support for identifying boundaries of ducts, urethra and prostate during the segmentation process.

I would like to thank Sudaraka Mallawarachchi, who is currently a doctoral researcher at Monash University, for his greatest support and effective feedback to make the outcome of the research a success.

Finally, my warmest thanks would go to my parents, brother and Dr. Sevvandi Jayakody who motivated to do my PG studies and without their support this wouldn’t have been a realistic task. I am so much grateful to my husband for his valuable support during the time of my research.

# TABLE OF CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

# LIST OF ABBREVIATIONS

| **Abbreviation** | **Description** |
|---|---|
| BPH | Benign Prostatic Hyperplasia |
| TURP | Transurethral Resection of Prostate |
| ASM/AAM framework | Atlas segmentation or Active Shape/Appearance model |
| RBF | Radial basis function |
| MPGA | Multi Population Genetic Algorithm |
| GA | Genetic algorithm |
| TRE | Target registration error |
| TPS | Thin-plate Spline |

# ABSTRACT

Benign Prostatic Hyperplasia (BPH) is a common non-malignant ailment effecting in ejaculatory duct of aging men. BPH induces bothersome lower urinary tract symptoms. The standard treatment for BPH is Transurethral Resection of the Prostate (TURP), which mitigate urinary symptoms and enhance urinary flow. Smooth sphincter of the bladder neck accumulates and resides seminal fluid as it reaches the prostatic urethra before it ejects during ejaculation. Retrograde ejaculation occurs due to removal of this smooth sphincter of the bladder neck during TURP. Hence, about 53-77% patients develop retrograde ejaculation after the procedure. The research has shown that preserving the portion of supramontanal prostatic tissue during TURP leads to preserve antegrade ejaculation in about 80% of patients. The accuracy of this surgical procedure could be enhanced by the aid of 3D modelling. A literature survey on the existing procedures for model construction indicated that further improvements could be achieved through reconstructing a 3D model. A 3D model will enhance the understanding of the anatomical relationship of the ejaculatory ducts and prostatic urethra in cross sections of the prostate gland and to determine a safe zone with the prostate to remove without damaging the ejaculatory ducts.

We used photographic images of prostates obtained from male cadavers above the age of fifty years. The prostate samples fixed on to a wax block and uniform 2 mm thick slices were removed sequentially while taking photographs with a digital camera. Major steps in constructing a 3D model from the acquired images include: image registration to align series of slices, segmentation of the prostate, urethra and ducts and 3D modelling of the segmented structures. A simple landmark based image registration technique was employed by manually selecting points along the four edges of the wax block and automatically detecting the vertices of the block using intersections. Then rotation, translation and scaling were estimated on individual slices to align all the slices. The prostate was then segmented manually using an existing software tool program. The ejaculatory ducts and the urethra were segmented using a simple active contour based segmentation tool. Finally, a 3D mesh model was developed using boundary points of each of the segmented structure. The following three surgically important measurements calculated using to the model: the angles of the centre of the left duct, to the centre of urethra and to the centre of right duct, perpendicular distance from the centre of urethra to the line joining the two centres of ducts, and width of the prostate. Results showed a large angle both proximally and distally, 3D relationships of ejaculatory ducts and urethra depended on the maximum width of the prostate. During TURP, safe distances to resect the prostate without damaging the ducts are calculated based on the maximum width of the prostate. Depth can be safely resected without damaging the ejaculatory ducts. In the future, it is quite essential to test these results on clinical grounds.

**Keywords**: Benign Prostatic Hyperplasia (BPH), Transurethral Resection of the Prostate (TURP), 3D modelling.

# 1. INTRODUCTION

## 1.1 The Background

Benign Prostatic Hyperplasia (BPH) is the most common urological disease in aging men [1]. It usually evolves after the age of 40 years. Natural history of BPH can be divided into two phases: pathological and clinical. The first phase takes place at the periurethral level with nodular hyperplasia while the second phase exists when enlarged prostate gland compresses the urethra consequently leads to an increment of resistance of urine flow [1].

Symptoms of BPH are resistance of urinary flow, urgency, frequency, nocturnal (wake up to urinate more than once at night), and weak urine stream with incomplete emptying [2]. TURP is the commonest surgical procedures carried out for treating BPH. TURP has accounted for 39% of surgical interventions for BPH carried out in the USA in 2008 [3]. In developing countries like Sri Lanka, these rates are much higher. Common complication of TURP is the loss of antegrade ejaculation [4, 5]. In 1994 and 1997 a randomized controlled trial had shown the possibility of preserving a portion of supramontanal prostatic tissue during TURP [6].

BPH is mostly treated by transurethral resection of the prostate (TURP) and other minimally invasive surgical methods. The use of conventional surgical procedure diminished by 39% in the USA contrary to a dramatic increase in surgical therapy in minimal invasive technologies [7]. Approximately 53-77% of patients undergo retrograde ejaculation which is a complication of TURP procedure [8]. The cause for the loss of antegrade ejaculation is the destruction of the internal sphincter of the bladder neck. During TURP, preserving the portion of supramontanal prostatic tissue lead to preserve the antegrade ejaculation in about 80% of patients [8]. Analysis of dynamic TRUS assessments of ejaculation has shed a new light which focus to a coordinated contraction of the prostate and seminal pathways eventually leading to ejaculation.

The minimally invasive technology for surgeries depends heavily on 3D models. At present, several models have been developed and are in use [9], [10], [11], [12], [13]. They have assisted partially in determining the anatomical relationships of the ejaculatory ducts to the prostate, prostatic urethra and verumonotanum by reconstructing a 3D model for ejaculatory ducts. However, existing 3D models are not adequate in rendering additional information such as anatomical relationship of the ejaculatory ducts to the prostate, the prostatic urethra and verumontanum. Such information is important for preserving the antegrade ejaculation in the male patients who undergo prostatic surgery.

Accordingly, a desk based study was conducted to evaluate the existing models with the objectives of assessing the anatomical relationship of the ejaculatory ducts and prostatic urethra in cross sections of the prostate gland. It was also used to evaluate the morphological variations within the prostate gland and this resulted in developing a new approach to construct a 3D model of the ejaculatory duct.

## 1.2 Research Motivation and Approach.

Damages to ejaculatory ducts can be prevented if morphological structure of the ejaculatory duct is known. Therefore, construction of a 3D model of ejaculatory duct is necessary to study the morphological structure of the duct. A 3D model of these structures can be built using image processing techniques such as image registration, segmentation and 3D modelling of histological images of slices of the prostate.

### 1.2.1 Objective

The objective of this study is to construct a 3D model of human ejaculatory duct and its morphological variation within the prostate gland.

## 1.3 Image Processing

Digital image represents as a 2D function f (x, y) which (x, y) are spatial coordinates. Digital image refers to the spatial coordinates (x, y) and amplitude of ‘f’ are discrete values. Digital images are formed of picture elements which are called as pixels. Image processing refers to manipulation and analysis of information contained in images. The benefit of image processing is to extract useful information from the raw data. Fundamental steps in image processing are as follows: image acquisition, image enhancement, image restoration, image compression, colour image processing, segmentation, morphological processing, representation and description, recognition, and knowledge based [14].

Image processing allows the user to visualize and segment interested regions from any cross-sectional (2D) or volumetric (3D) data [15].

Image registration is an essential step in determining the spatial relationship between 2 images under variety of conditions e.g.: employing different sensors, at different time occurrences, from diverse viewpoints or combination of latter situations [16]. Registration of two images determine the motion that allow *moving image* into the best possible alignment with the *reference image* [16]. There are four critical components in the development of registration algorithm: feature space, search space, search strategy and similarity metric [3]. The same image features within the image should be matched. Search space can also be called as range of transformations. Search strategy finds the optimum transformation within search space and similarity metric measures the optimality of a transformation for the chosen feature data set [18]. Transformation is necessary to identify each 3D point within image which corresponds the location of the other image. Rigid body transformation changes the position and orientation of the moving image without changing shape or size [19].

Generic model needs automated pre-processing to establish the surface edge points. 2D digital images are processed initially to extract the edge points [3], [20]. For all projects of serial reconstruction, reference points for proper alignments of image slices are crucial [21], [3].

Image segmentation needs certain amount of user assistance in most cases. Hence, defining objects iteratively and their visualization during the process are essential to guide user’s actions for the sub-sequent iteration. During the segmentation, slice visualization is significant for precise object definition which help immensely for 3D visualization [10]. During model construction, smooth contours are necessary for easy manipulation [21]. Accurate model contour initialization is necessary for active contour model method [3].

There are four basic steps for 3D modelling operation [22].

Pre-processing: Defines the object system. Suppresses unwanted distortions or enhances some image features important for further processing.

Visualization: Helps viewing and comprehending the structure and dynamics of the object system. Transforms digital data into images representing information about the data.

Manipulation: Adjusts individual objects or connections among the objects in the object system. Transforming or altering an image using various methods/techniques to achieve desired results.
Analysis: Quantifies the morphological and functional information about the system. Converts image into fundamental components to extract statistical data.

## 1.4 Image registration techniques

According to P. Markelji et al. 2012 [4], regardless of dimensional correspondence 3D/2D registration methods can be classified as extrinsic, intrinsic or calibration-based. Intrinsic methods have sub categories such as feature-, intensity- or gradient-based, while extrinsic methods depend on artificial objects like stereotactic frames, or a small number or markers attached to frames, dental casts, or implanted into bone soft tissue or skin affixed. In order to evaluate the registration process quantitatively, researches altered the optimizers, similarity metrics and interpolators. After the experimental results, deformable image registration in 3D medical images, can competently reconstruct a 3D brain from volumetric data [23].

## 1.5 Image segmentation techniques

Segmentation is one of the major and most challenging steps in image analysis [5]. The one of segmentation solutions are based on thresholding, region growing [24] or feature extraction methods [25]. More advanced algorithms are Atlas segmentation or Active Shape/Appearance Model (ASM/AAM) framework [25] and level set of techniques [25]. Khlafia et al. [25] employed level set of techniques with shape priors While Huan et al. [25], proposed Chan-Vase model employing in shaping the model.

## 1.6 Three-dimensional (3D) modelling techniques

The advantage of 3D model in the medical field is to assist doctors in clinical diagnosis for teaching purposes for surgical training and for remote operation [26]. Some of the reconstruction techniques are thresholding operations, minimum production tree segmentation and morphology and FFD (Free-Form Deformation) method.

## 1.7 Models of Prostates and its internal structure

### 1.7.1 1st model

Cool D. et al, proposed an approach to construct a patient specific 3D prostate model. This was done by employing sparse collection of 2D TRUS (transrectal ultrasound) biopsy images. Semi-automated segmentation technique was used to segment the prostate. 2D prostate boundaries are acquired from both orthogonal orientations. Then by employing radial basis function (RBF), 3D prostate surface is fitted.

Surface approximation formula:

$$s(x) = p_1(x) + \sum_{i=1}^{n} \lambda_i \Phi(\|x - x_i\|),\ x \in \Re^3,\ \lambda_i \in \Re \text{ ......................... Eq. 1}$$

RBF uses a function s: $\Re^3 \rightarrow \Re$ approximates the input function f: $\Re^3 \rightarrow \Re$, while {f(x $_i$ ):i=1,2,…..,n} which portrays set of input prostate boundary points. p $_i$ is 1st order polynomial. $\|.\|$ is an Euclidean norm. $\phi(r) = r$ is a biharmonic splines. x $_i$ is radial center.

$\lambda_i$ is determined by requiring s satisfy interpolation condition.

$$s(x_i) = f(x_i), \quad i=1, 2, \ldots, n \quad \text{.................................. Eq. 2}$$

Also, side condition:

$$\sum_{j=1}^{n} \lambda_j q(x_i) = 0 \quad \text{for all } q \in \pi_1^3 \quad \text{.................................. Eq. 3}$$

By minimizing, spline smoothing of s(x) is been achieved:

$$p\|s\|^2 + \frac{1}{n}\sum_{i=1}^{n}(s(x_i) - f(x_i))^2 \quad \text{.................................. Eq. 4}$$

Once both RBF and $s(x)$ are estimated, iso-surfacing technique is employed to develop the patient's 3D prostate model. Advantages of this method are large data-free gaps when interpolating, and RBF is utmost effective and accurate. As the basic function, Biharmonic splines were employed.

### 1.7.2 2nd model

Cosio F. A. et al. proposed a novel method for automatic segmentation of the prostate boundary in ultrasound images [12]. This depends on the automatic initialization of an active shape model [21]. First, Bayes' classifier employed to perform pixel classification on the grey level ultrasound images. Next, by using Multi Population Genetic Algorithm (MPGA) initialized the Active Shape Model (ASM) of the prostate. This generates a rough approximation of an appropriate initial shape and pose of the ASM. In the following stage, MGPA is employed on the grey level of the prostate image and eventually refine the initialization of the ASM. During the last stage, ASM of the prostate is used to segment the final boundary of the gland.

Theoretically, Bolt et al. (1987) [15] following theorem is used:

$$S = \bar{s} + \sum_{k=1}^{10} b_k p_k \; ; \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{ Eq. 5}$$

$\bar{s}$ is the mean prostate shape, $p_k$ is a principal component vector, $b_k$ is a weight of $p_k$ (-$\sqrt[3]{\lambda_k} \leq b_k \leq \sqrt[3]{\lambda_k}$ ) and $\lambda_k$ is the Eigen value associate to each $p_k$.

New prostate shape is generated from the weighted sum of the 10 principal component vectors, $p_k$ and mean shape, $\bar{s}$. Then statistical grey level models can be produced by taking the pixel profiles perpendicular to the shape model. Hence, correct prostate boundary is located by sampling at each point of the point distribution model. In order to reduce the global intensity changes, normal derivative profiles were employed. A new position is always estimated. A Mahalanobis distance to the corresponding mean derivative profile is calculated at each position of the search. Hence, optimum position of the boundary point resides when the Mahalanobis distance is minimum. Even a Gaussian image pyramid (with four levels) was generated for each training images.

This method is fast, robust and suitable for different organs on different imaging modalities. Weaknesses of this protocol includes: requirement of more than 2 point on the prostate from the user, errors generated by convergence of the genetic algorithm (GA) to local minima and existence of absolute minimum values of the objective function not corresponding to the prostate boundary images.

### 1.7.3 3rd model

Histology, paraffin block face and magnetic resonance images of six prostates were captured. On each image, from 7 to 15 homologous landmarks were labelled. Then, the researchers manually identified landmarks and quantified the misalignment of histology sections from the front faces of tissue slices. A particular reconstruction model is then developed. The reconstruction approach depends on the least-squares best-fit transformation of selected homologous intrinsic landmarks under various reconstruction models. Target registration error (TRE) has been used to validate the reconstruction model. Front face assumption cannot be directly applied because Thin-plate Spline (TPS) transformation is an interpolating spline and from the front face assumption, fiducials may i.e. at non-zero depth. Front face assumption can be determined by projecting the target fiducials and to illustrate a TPS transformation. For specific transformations like: rigid, similarity and affine; limited least-square fittings of transformed source fiducials to target fiducials is mathematically equivalent to unlimiting least squares fitting of transformed source fiducials to the projected target fiducials. Therefore, front face assumption for the reconstruction is created [28]

Strengths of this parameter-free method are local optima and their accuracy depend on fiducials. Weakness of this methodology is that if 3D image registration comes after the reconstruction then that would conflict in isolating, which was not address in this study.

### 1.7.4 4th model

The need of image segmentation of Ultrasound images was due to low signal to noise ratio and significant presence of artefacts of US images. According to Cosio [27], new method was discovered for automatic segmentation of the boundary of the prostate in transurethral ultrasound images. Final goal here was to measure the prostate of a patient intraoperatively during a computer assisted TURP. Therefore, they accurately constructed a 3D prostate model based on automatic initialization of an active shape model. Active shape model was initialized in order to perform automatic segmentation of prostate boundary based on automatic initialization of an active shape model [27].

# 2. METHODS

## 2.1 Histological Image Slice Acquisition

Self-donated six male cadavers of age above fifty years, were selected from the Sri Lankan community. First, the selected cadavers placed accordingly and incisions were made. Then, prostate was harvested along with the seminal vesicle and trigone region of the bladder. Next, preserved the dissected prostate, pair of seminal vesicles and a small segment of vas deferens. After that, tissues were processed accordingly. Once the processed prostate was fixed to the mould in the wax block, cross sections were made manually with a sharp knife and employed a single uniform force which produced a slice thickness of 2mm for whole throughout the slices. Following that, cut cross sections were photographed from a fixed point with a high resolution digital camera. These images were acquired at the Department of Anatomy University of Colombo.

The Original image, shown in Figure 1, has a dimension of 4288 pixels x 3216 pixels with a bit depth of 24 and colour representation RGB. The registration procedure rotated, translated and scaled individual slices to align all the slices. Hence inaccuracies of point selection, z plane camera distortion, and wax block deformation were corrected in registering images.

## 2.2 Histological slice registration

On the assumption, that two consecutive slices should be closely matched, we used a slice by slice registration to align all slices of each prostate. We fitted lines along all four edges of the wax block using manually selected points in a pre-defined order. Vertices were defined automatically suing the intersections of these lines. Using the four corners as feature points, the moving image slice was rotated, translated and scaled to align with the preceding slices.

Hence, registration deformations were analysed in two perspectives, e.g.: wax block height/width and wax block area to confirm whether the registered image had any distortions.

### 2.2.1 Scaling issue

We have assumed that the wax block is a rigid object. This means that for a given subset of images, the height to width ratio for all boxes must be the same (ignoring negligible differences). It was not the case with some data sets due to the inaccuracies of point selection, z plane camera distortion and wax block deformation (e.g.: inaccuracies in cross sections, errors in slicing sections).

Distortion was minimized by performing an intensity based registration. However, for some data sets, after the intensity based registration, images were scaled out of proportion and hence intensity based rigid registration was not employed. Instead, we aligned urethra in all slices before making the 3D model. This enabled us to estimate and visualize important measurements of orientation of structures within the prostate effectively.

## 2.3 Calibration of images

Following assumptions have been made during this study

1. Wax block is a square, rigid and
2. Thickness is uniform.

A known distance is measured along the length and width of the wax block (a ruler has been kept both sides of the block while acquiring images). Then the calibration factor can be calculated as:

$$a \text{ (cm)} / N \text{ (pixels)}$$

a = length/width of the wax block

N= Number of pixels

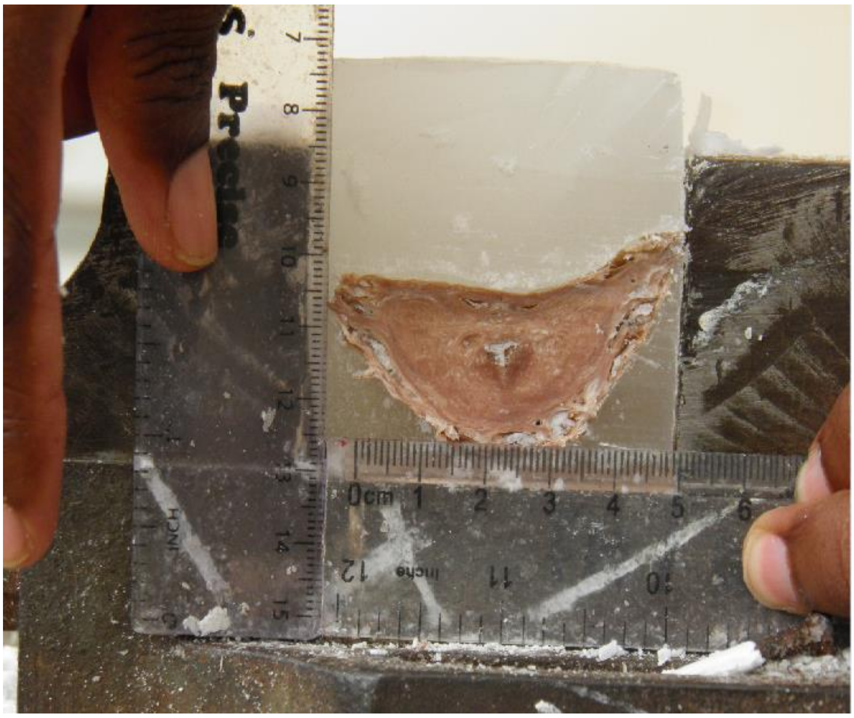


Figure 1: Original image (Original in Colour)

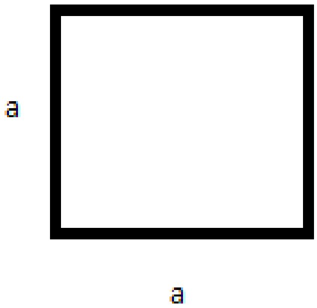
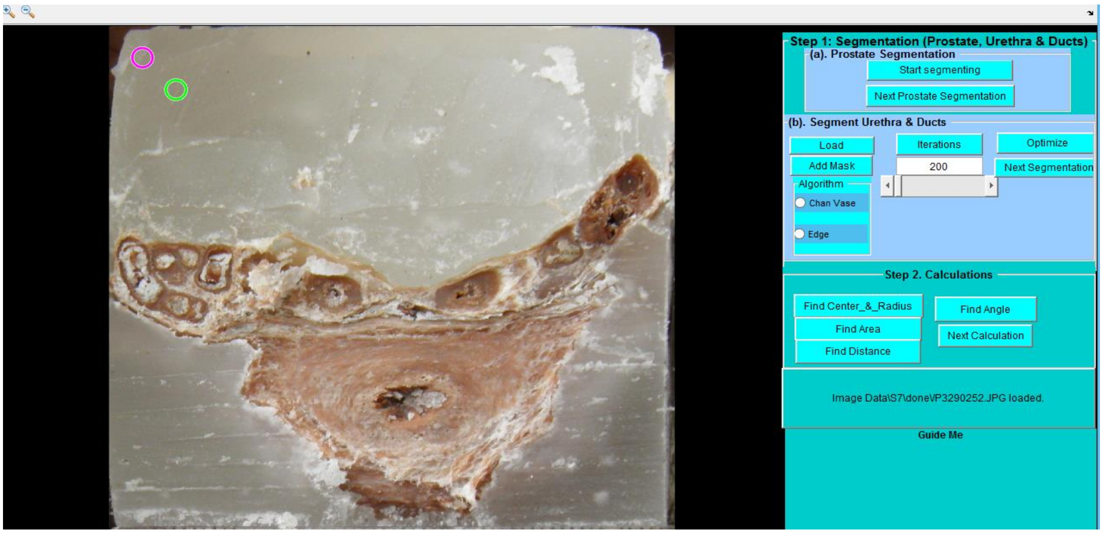


Figure 2: Calibration suqare for the wax block

## 2.4 Segmentation of the prostate structures and 3D modelling

Prostate is manually segmented along the boundary using the active contour method. By employing an adjustable radius, we manually segmented the Urethra and two ducts with the assistance of medical professionals. When applying snake following parameters were taken into consideration: image, mask, max iterations, algorithm (Chan - vese & edge) and smoothness as shown in Figure 3. Finally, a mesh model was developed using boundary points for Prostate, Urethra and Ejaculatory ducts as shown in Figures 8 & 9. We made the Urethras align on a line in our prostate 3D model construction. All anatomical measurements were calculated relative to the aligned urethra in our 3D prostate model.



Iterations: 200

Image Data\S7\done\P3290252.JPG loaded.

Figure 3: Segmentation program GUI

## 2.5 Measurements

A mesh model was constructed using boundary points for the prostate, ejaculatory ducts and the urethra. The following measurements were calculated using the model to evaluate the relative orientation of important sub structures with the prostate.

1. The angle between the two ejaculatory ducts as measured from the centre of the prostatic urethra. Above mentioned measurement represents the relation between ejaculatory ducts and urethra as shown in Figure 4. This calculation was done by using the Cosine rule. In the triangle, all three sides' distances were calculated and a particular angle (angle between the two ejaculatory ducts) need to be determined hence Cosine rule was employed.
2. The perpendicular distance between the urethra and the line joining ejaculatory ducts against the distance from the verumonatanum, as shown in Figure 5. This

is a critical measurement to identify the maximum linear distance surgeons could go through during the laparoscopic procedure.

3. The maximum prostate width for each set of patients as shown in Figure 6. This measurement is a direct measurement of the size of the prostate in different patients.

$$\Theta=\cos^{-1}(R_1{}^2+R_2{}^2-d^2)/(2R_1R_2) \qquad \text{Eq. 6}$$

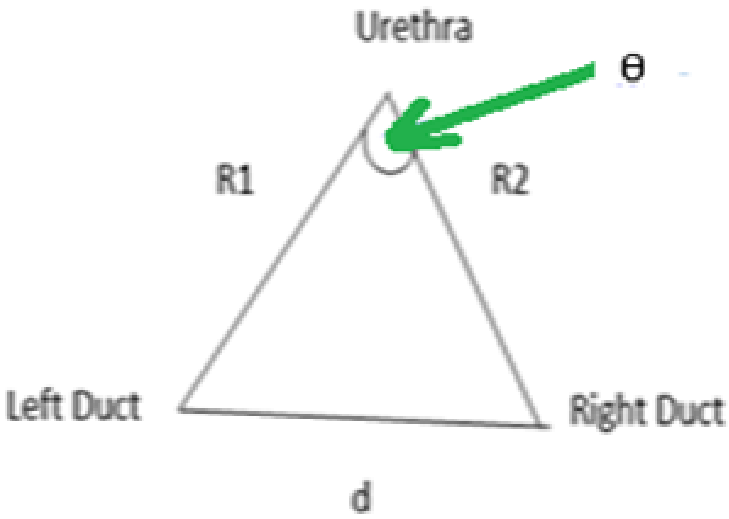


Figure 4: Angle between the two ejaculatory ducts as measured from the centre of the prostatic urethra

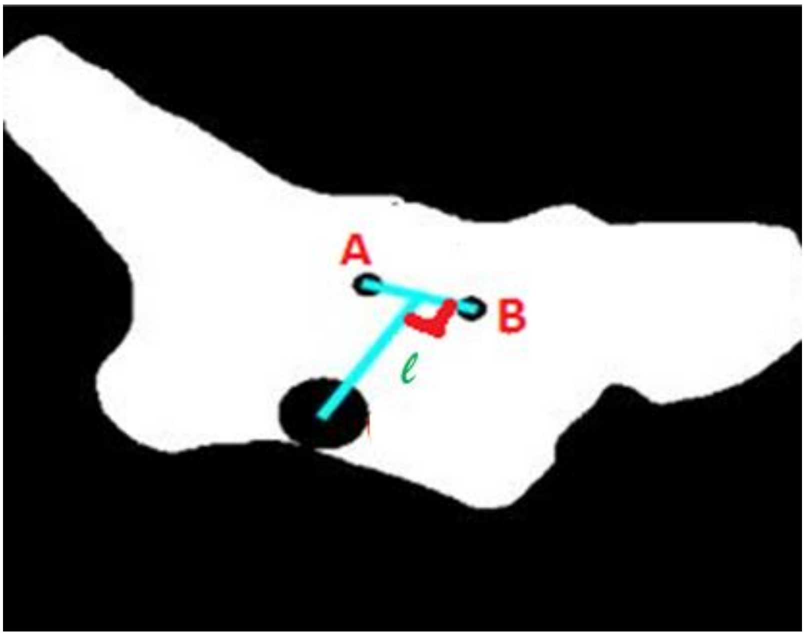


Figure 5: The perpendicular distance between the urethra and a line joining ejaculatory ducts against the distance from the verumontanum.

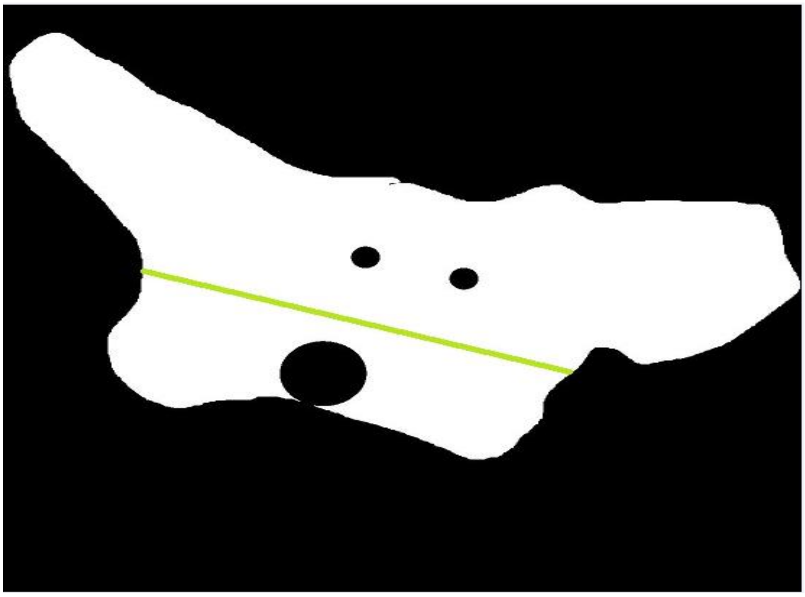


Figure 6: Prostate width (maximum)

# 3. RESULTS

## 3.1 Results of histological slice registration

We assumed that the wax block is rigid. Therefore, while maintaining the aspect ratio of the block, we performed  two possible scaling approaches. Such as: scaling by the height ratio (minimum block height/block height of the current block), and the width ratio (minimum block width/block width of the current block). Both approaches, as expected, show the same variation as the original blocks. Due to this reason, the scaled images are of different sizes as shown in Figure 10. Figure 11 depicts the area for each slice. In both figures, the circles show an acceptable outcome in terms of registration (matching height: width and block areas). In order to achieve this, scaling needs to be done separately using both height and width parameters producing a distorted image. Figure 9 elaborates the distortion as seen in image 40.

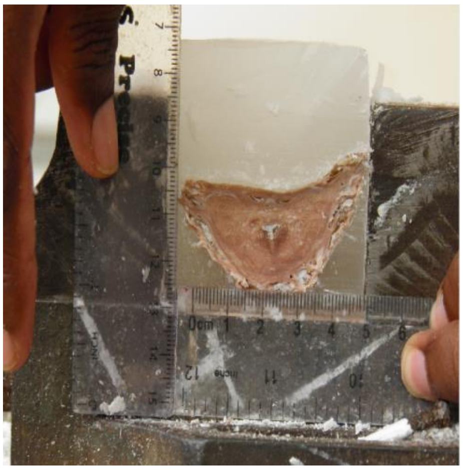


Figure 7: Raw image

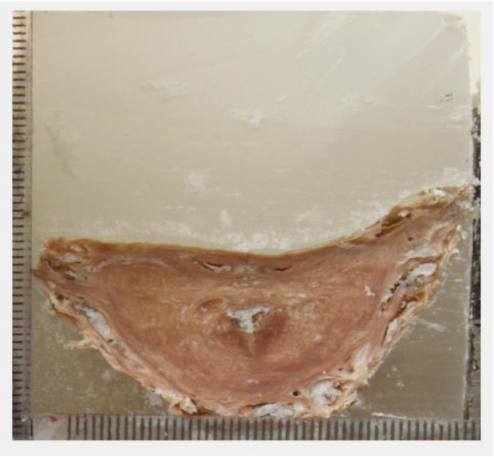


Figure 8: Cropped and registered images

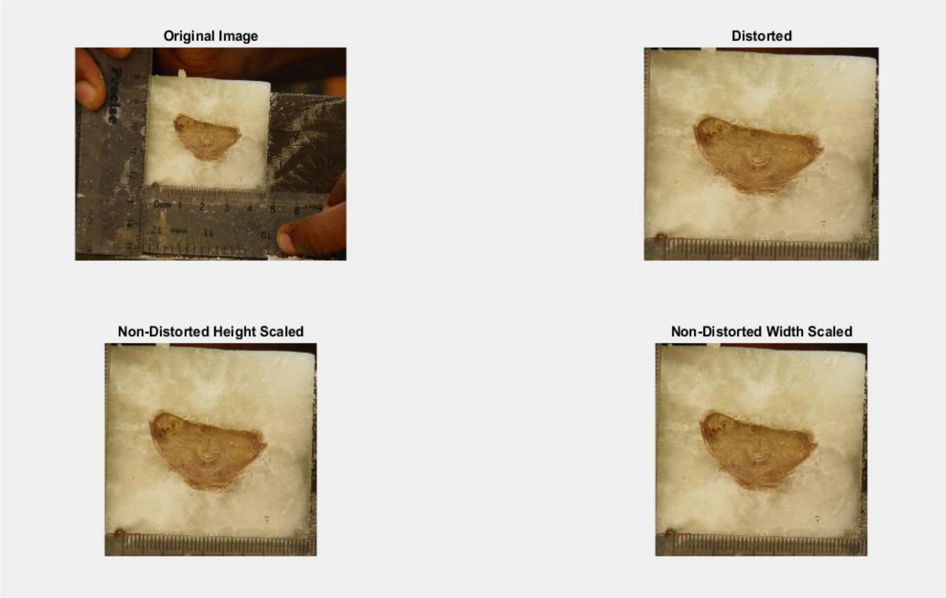


Figure 9: Original, distorted, non-distorted width scaled and non-distorted height scaled images

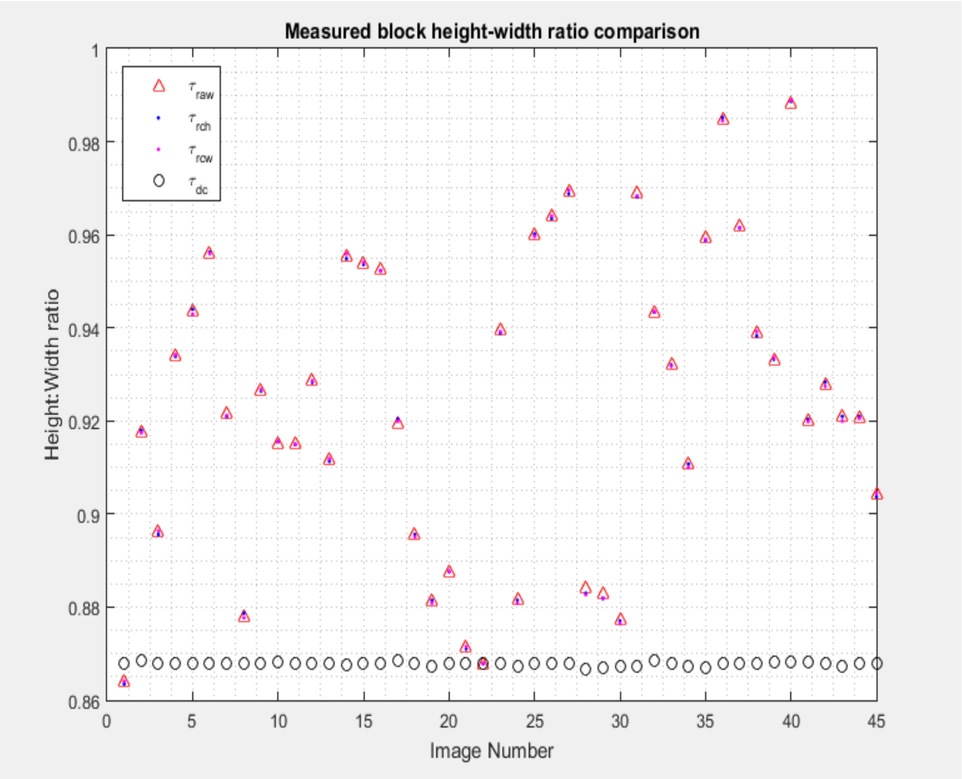


Figure 10: Measured block height:width ratio comparison

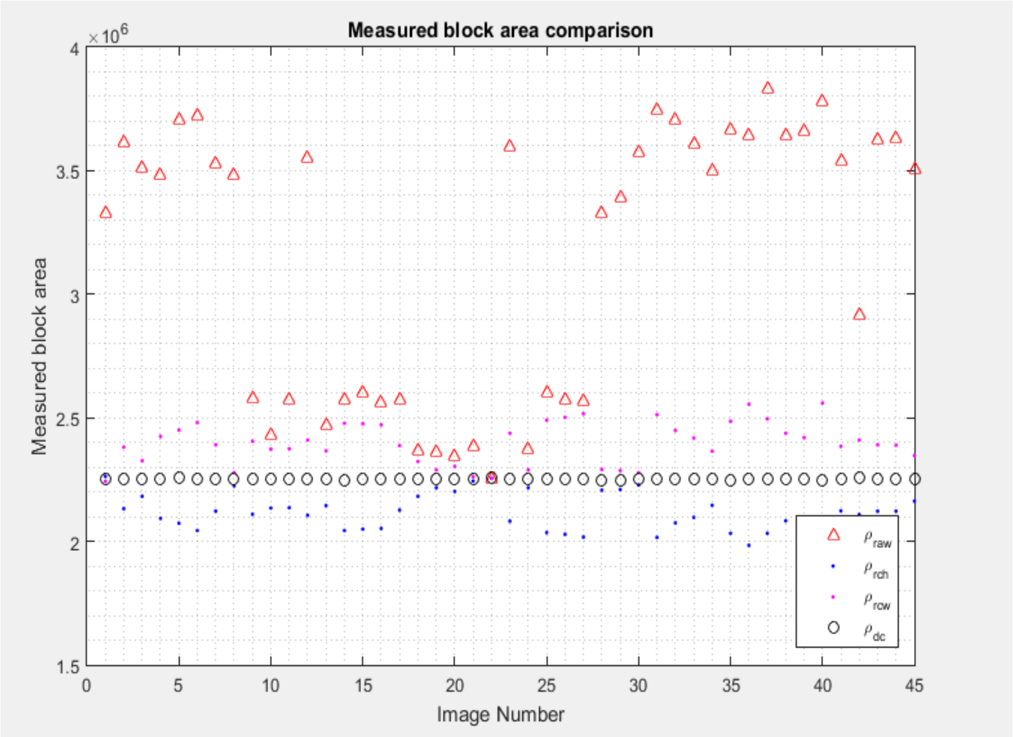


Figure 11: Measured block area comparison

## 3.2 Calibration measurement

Calibrated the images from transformed image coordinate to word coordinate using the ruler on the image is shown in Figure 12. A 5 cm long line was selected on the ruler to calculate spacing of each pixel.

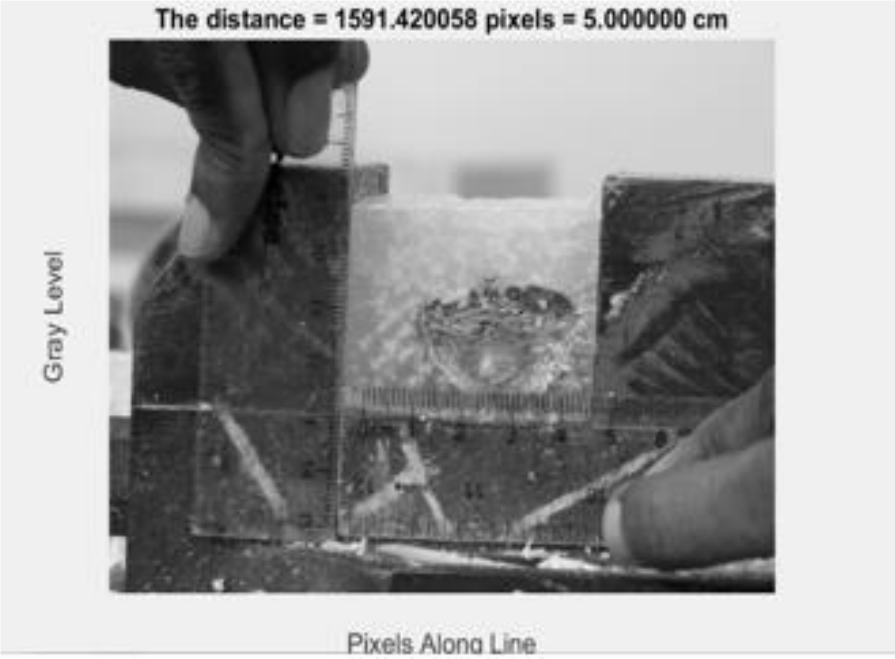


Figure 12: Calibration of image

### 3.3 Segmentation of prostate

As shown above, prostate had been manually segmented along the boundary as shown in Figure 13. Then the urethra and two ejaculatory ducts were manually segmented using a circular cross-sectional template with varying the radius as shown in Figure 14.

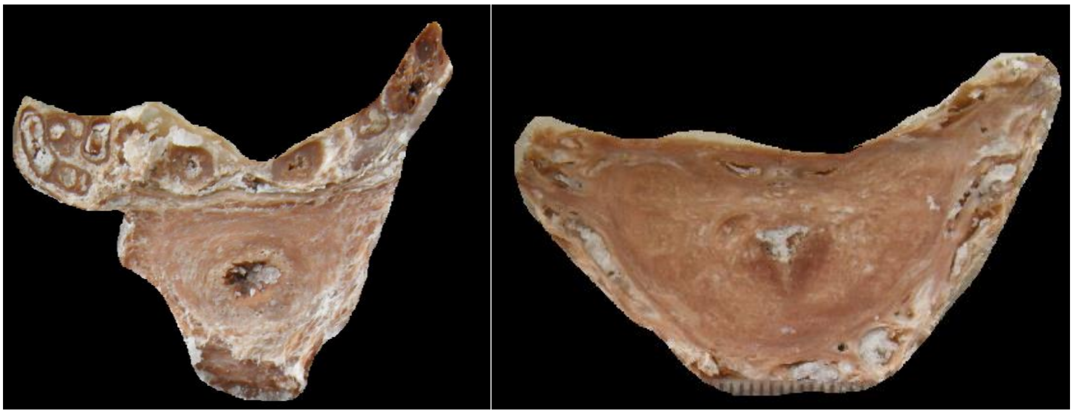


Figure 13: Segmented prostates, Before_ Ducts and Urethra segmentation

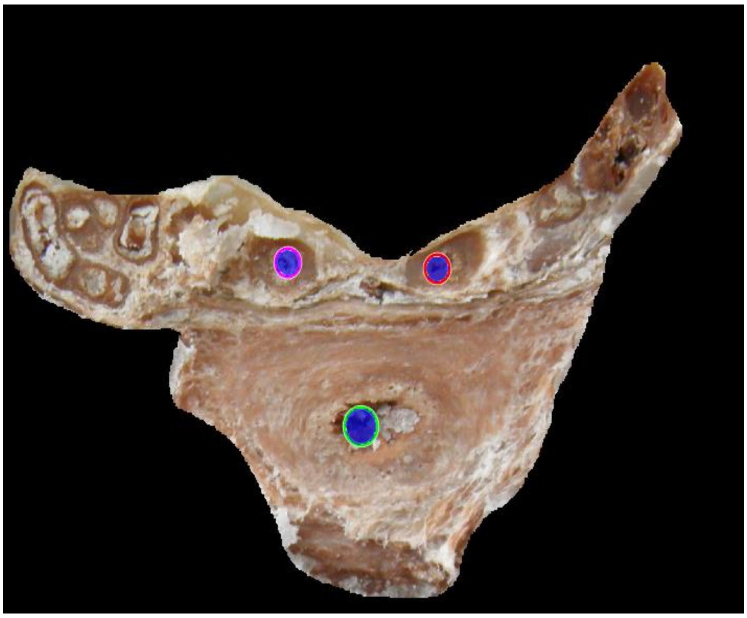


Figure 14: After_Ducts & Urethra segmentation

### 3.4 Measurements

Table 1 and Figure 15 depicts the angle between two ejaculatory ducts as measured from the urethra for the prostate no. 7. Table 2 and Figure 16 illustrates the distance between the urethra and line joining two ejaculatory ducts vs slice number for the same prostate.

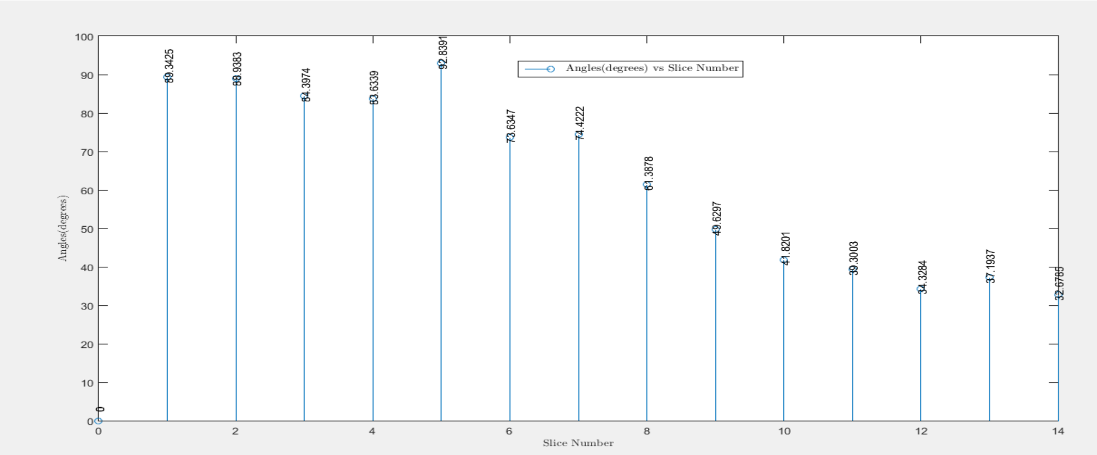


Figure 15: Angle between two ejaculatory ducts as measured from the urethra vs slice numbers for the prostate no.  7.

Table 1: Slice number vs. angle between two ejaculatory ducts as measured from the urethra for the prstate no.  7.

| **Slice Number** | **Angles(Degrees)** |
|---|---|
| 1 | 89.3425 |
| 2 | 88.9383 |
| 3 | 84.3974 |
| 4 | 83.6339 |
| 5 | 92.8391 |
| 6 | 73.6347 |
| 7 | 74.4222 |
| 8 | 61.3878 |
| 9 | 49.6297 |
| 10 | 41.8201 |
| 11 | 39.3003 |
| 12 | 34.3284 |
| 13 | 37.1937 |
| 14 | 32.6785 |

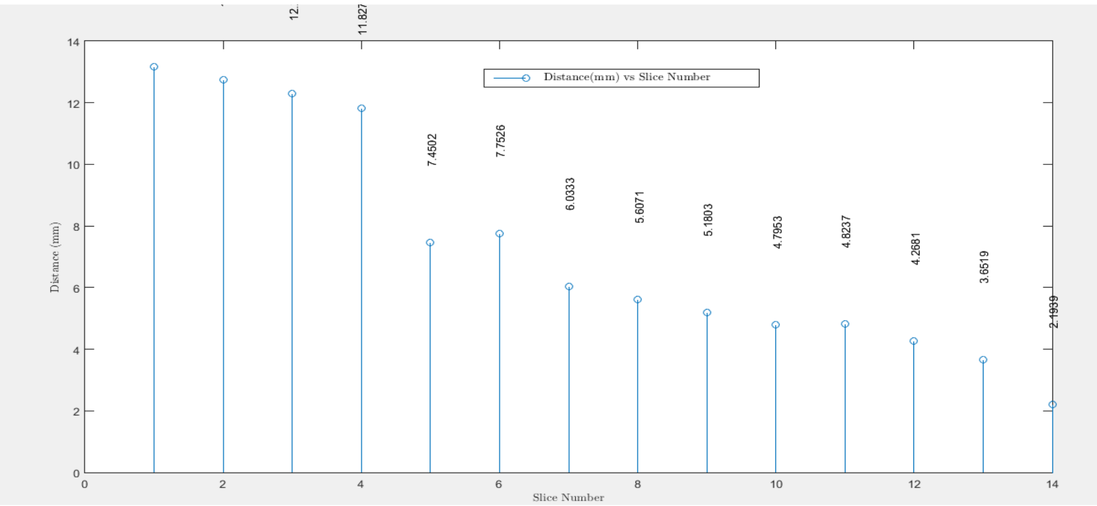


Figure 16: Distance between the urethra and line joining two ejaculatory ducts vs. slice number for the prstate no. 7.

Table 2: Slice number vs distance from the urethra to the line joining two ejaculatoy ducts for the prstate no. 7.

| **Slice Number** | **Distance(mm)** |
|---|---|
| 1 | 13.1695 |
| 2 | 12.7387 |
| 3 | 12.2949 |
| 4 | 11.8278 |
| 5 | 7.4502 |
| 6 | 7.7526 |
| 7 | 6.0333 |
| 8 | 5.6071 |
| 9 | 5.1803 |
| 10 | 4.7953 |
| 11 | 4.8237 |
| 12 | 4.2681 |
| 13 | 3.6519 |
| 14 | 2.1939 |

Table 3: Slice number vs prostate widths for the prostate no.  7.

| Slice<br>Number | widths(pixels) |
|---|---|
| 1 | 954 |
| 2 | 1263 |
| 3 | 1287 |
| 4 | 1311 |
| 5 | 1470 |
| 6 | **1524** |
| 7 | 1464 |
| 8 | 1350 |
| 9 | 1320 |
| 10 | 1278 |
| 11 | 1203 |
| 12 | 1167 |
| 13 | 1107 |
| 14 | 1104 |

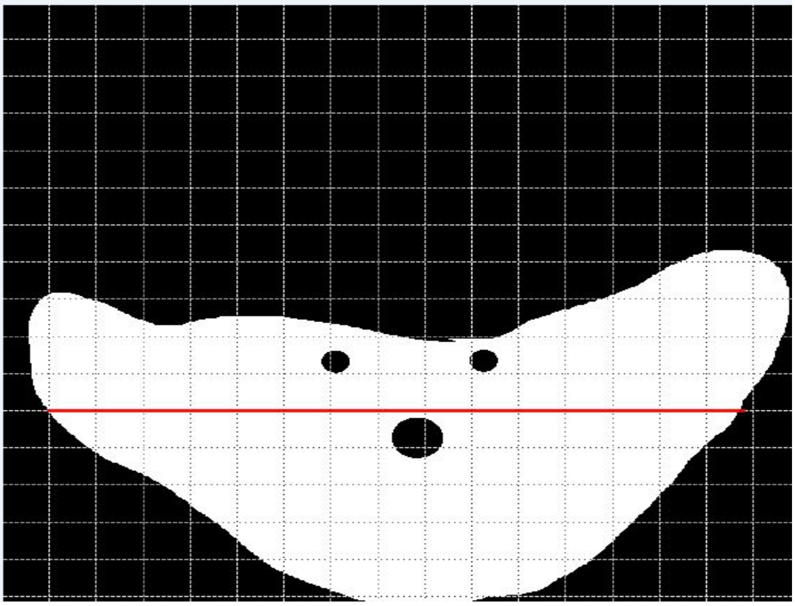


Figure 17: Maximum prostate width

Calibration factor= true height (mm)\number of pixels
= [10(mm)\342.41406 (pixels)]
= 0.0292

Max Prostate width for the prostate no. 7.
=1524(pixels)*0.0292(mm\pixels)
= 44.50 mm

Table 3 and Figure 17 exhibits the prostate width vs slice number for for the prostate no. 7.

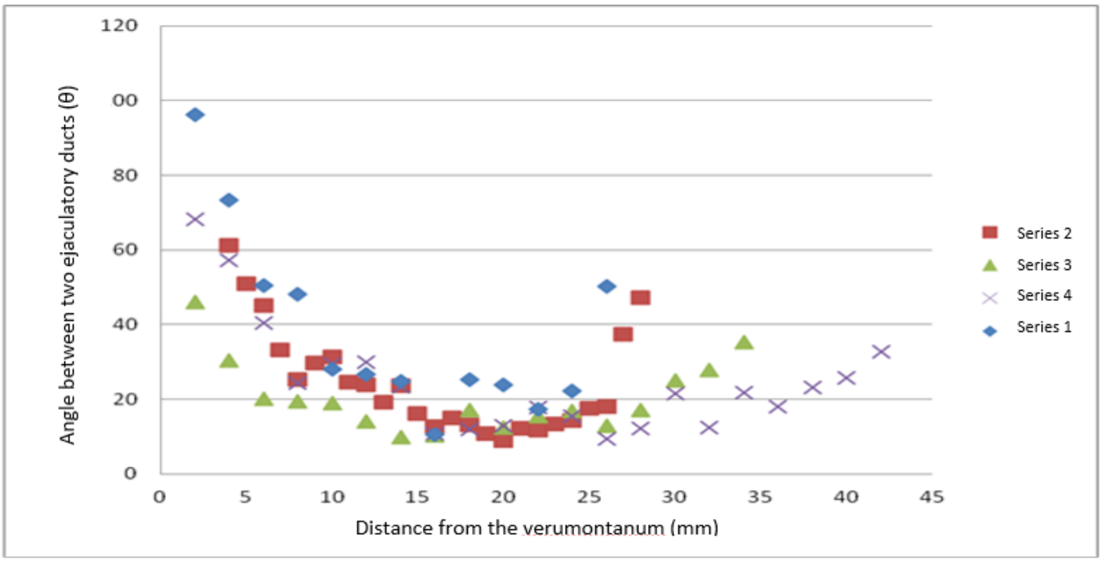


Figure 18: The angle (θ) between the two ejaculatory ducts as measured from the centre of the prostatic urethra against the distance from the verumontanum proximally (height, “a”) in four series of prostates.

Figure 18 depicts the angle (θ) between the two ejaculatory ducts as measured from the centre of the prostatic urethra against the distance from the verumontanum proximally (height, “a”) in four prostates. Prostates no. 1, 2 and 4 have similar patterns, which have a steep slope at the beginning and then increasing concave up to some extent. On the other hand, for series 3 the pattern differs slightly which the starting point start at low value of y and level off to some extent and finally surged to a short distance. Hence, series 1 has the lowest and series 4 has highest angular variation between two ejaculatory ducts with respect to the centre of prostatic urethra.

Figure 19 illustrates the perpendicular distance (depth, “d”) between the urethra and a line joining ejaculatory ducts against the distance from the verumontanum proximally (height, “a”) in six prostates. Series 1, 2, 5, 6, and 7 increases gradually while series 1 and 6, scatters after the surged. Then series 2, 5, and 7 have a vertex following the gradual rise and start to decline to a short distance. Therefore, series 5 has the maximum while series 2 has the minimum value for the depth between the urethra and a line joining the ejaculatory ducts.

Figure 20 is an extraction of the first 20 mm of the perpendicular distance (depth) between the urethra and a line joining ejaculatory ducts against the distance from the verumontanum upwards (proximally) in the six prostates in Figure 19. All series are linear graphs where it rose sharply. Out of all, series 5, 6, and 7 have quite similar slopes while series 1 has the highest slope on the other hand series 2 has lowest slope. Therefore, series 1 has the maximum and series 2 has the minimum perpendicular distance from the urethra to the line joining two ejaculatory ducts. The final outcome of the 3D model is shown as boundary points in Figure 21 and as surfaces in Figure 22 & 23.

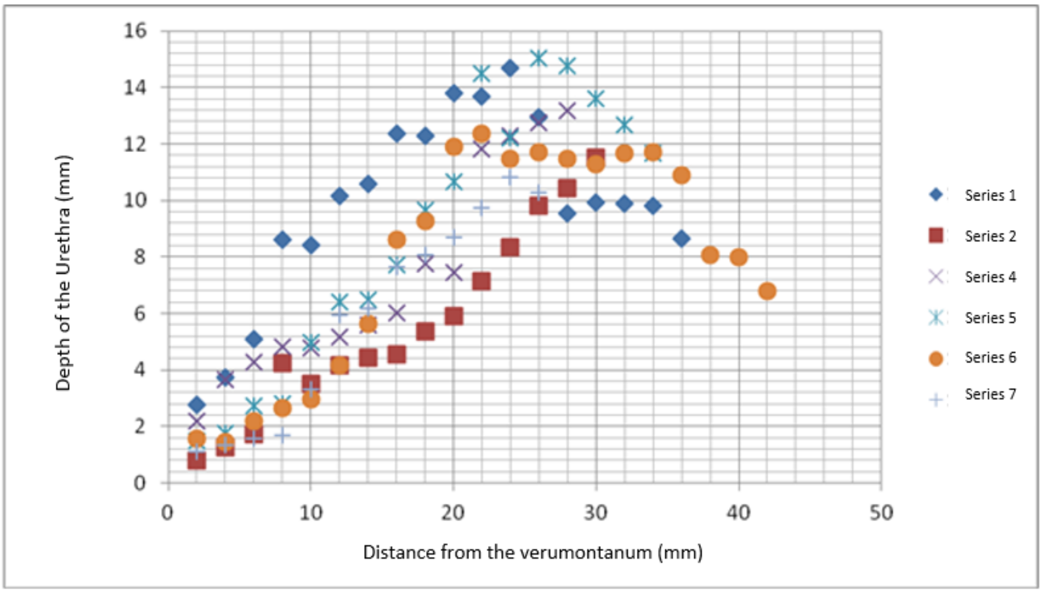


Figure 19: The perpendicular distance (depth, “d”) between the urethra and a line joining ejaculatory ducts against the distance from the verumontanum proximally (height, “a”) in six series of prostates.

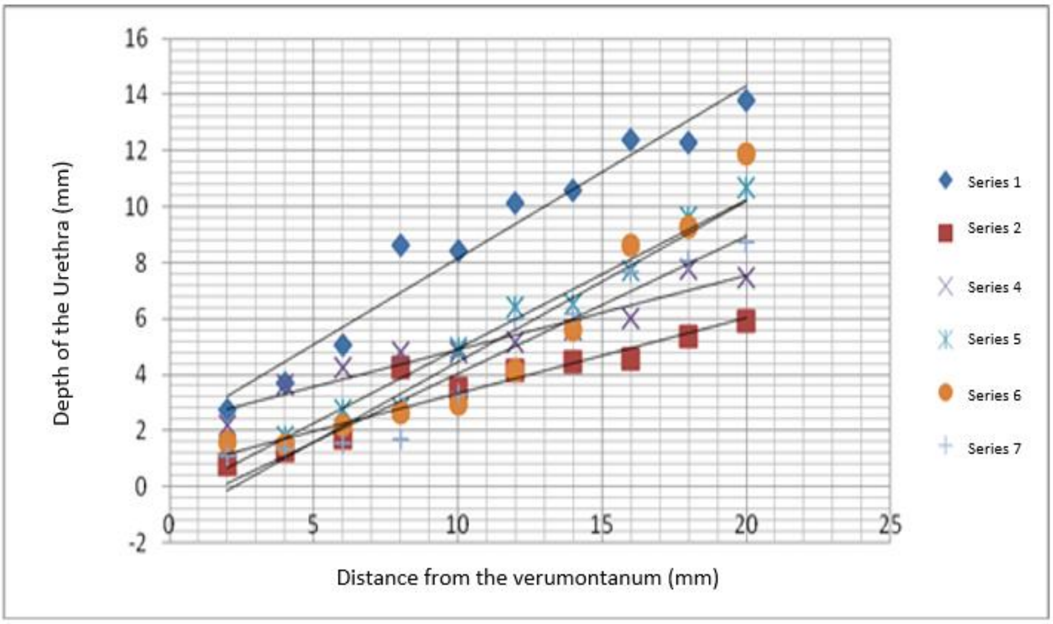


Figure 20: Analysis of the first 20 mm of the perpendicular distance (depth) between the urethra and a line joining ejaculatory ducts against the distance from the verumontanum upwards (proximally) in six series of prostates.

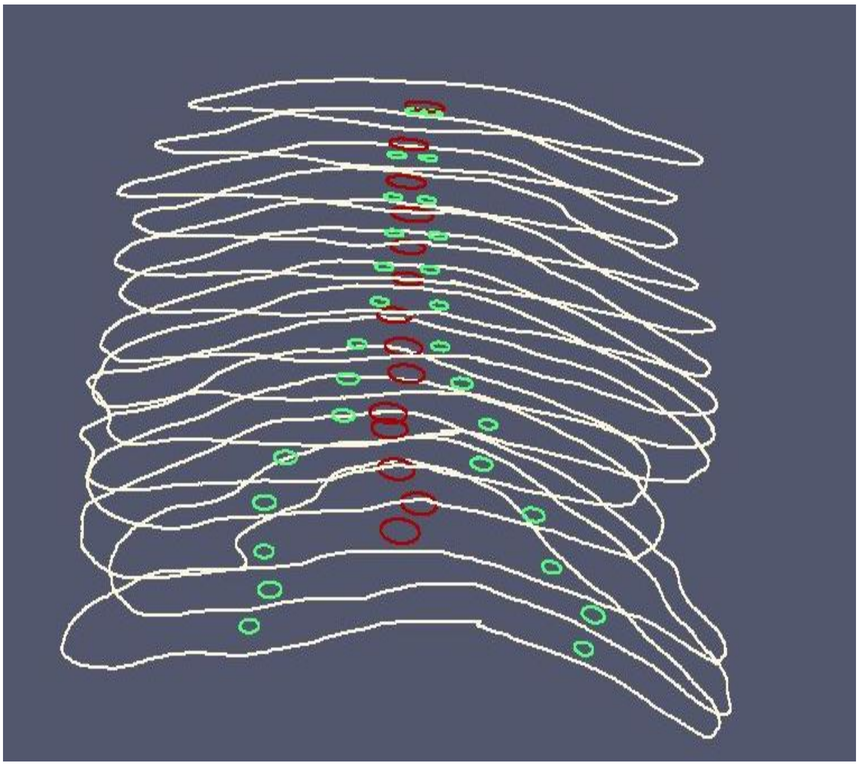


Figure 21: Model of the boundaries of urethra and 2 ducts

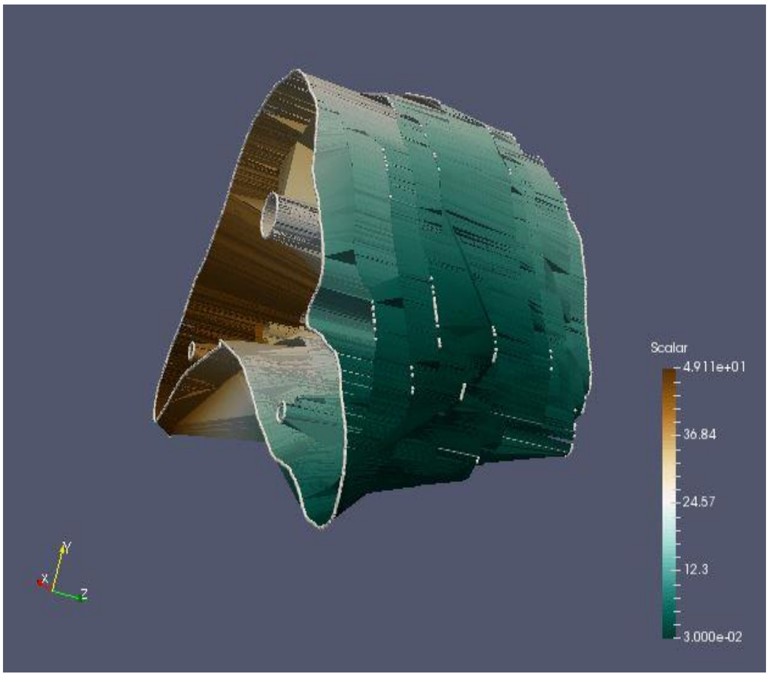


Figure 22: Modeled the prostate, Ducts and Urethra – view 1.

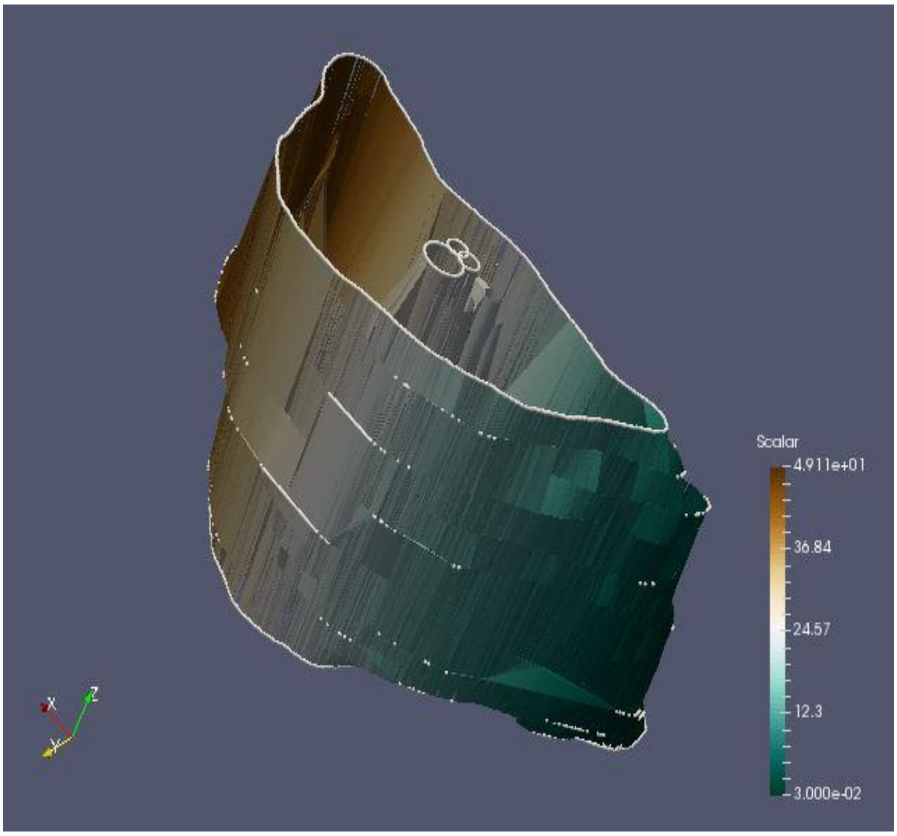


Figure 23: Modeled the prostate , Ducts and Urethra – view 2.

# 4. CONCLUSIONS AND FUTURE WORK

The aim of this dissertation was to build a 3D model of the prostate to understand the morphological variation of the human ejaculatory duct. Various 3D reconstruction models for the prostate has been reported in research literature. Initially we obtained 8 sets of cadavers to construct a 3D model. But set no 6 and 8 were discarded due to low image quality. Hence, only the remaining data sets were used in this research.

These models were used to study both shape and size of the prostates. In contrast to existing 3D models of the prostate, our mesh model was used to estimate the angle between the two ejaculatory ducts as measured from the centre of the prostatic urethra, perpendicular distance between the urethra and a line joining ejaculatory ducts against the distance from the verumontanum, and the maximum prostate width for each set of patients. Through this model, we were able to study the anatomical relationship of the ejaculatory ducts and prostatic urethra in cross sections of the prostate gland for the first time.

The 3D relationship of ejaculatory ducts and prostatic urethra, and safe distances to resect the prostate without harming the ejaculatory ducts during the surgery relative to the maximum width of the prostate can be considered extremely useful clinical findings. After further testing of these values clinically, surgeons could calculate the amount of periverumontanal prostatic tissue which will eventually preserve the ejaculatory ducts for forbidding the retrograde ejaculation after TURP procedure.

This study has few limitations, which can be improved in the future. Firstly, study was done based on a small sample, a large population required for more meaningful interpretation. Secondly, the subtle deterioration of the anatomical relationship may have been intervened with the measurements as we overlapped the prostatic urethra. Image acquisition itself had scaling, blurring, lighting issues etc. but image distortion was minimized through image registration. Finally, evidence either on clinical or histopathological of BPH among study subjects were unavailable to determine whether observed relationships can be reproduced.

In the future, larger studies with carefully designed clinical trials are essential for the recommendation for the safe distance during TURP of the 3D mesh model develop for this research can be utilized during such clinical trials.

# REFERENCES

[1] O. Allkanjari, and A.Vitalone, “What do we know about phytotherapy of benign prostatic hyperplasia?,” Life Sci., vol. 126, pp. 42–56, 20157

[2] S.A.Kaplan and K.T.Mcvary. Male lower urinary tract system and BPH,2nd ed, Wiley Blackwell, 2014

[3] Yu X, Elliott SP, Wilt TJ, McBean AM. Practice patterns in benign prostatic hyperplasia surgical therapy: the dramatic increase in minimally invasive technologies. J Urol. 2008;180(1):241-5; discussion

[4] Bolt JW, Evans C, Marshall VR. Sexual dysfunction after prostatectomy. Br J Urol. 1987;59(4):319-22.

[5] Chung A, Woo HH. Preservation of sexual function when relieving benign prostatic obstruction surgically: can a trade-off be considered? Curr Opin Urol. 2016;26(1):42-8.

[6] Vecchis DE. Preservation of anterograde ejaculation after transurethral resection of both the prostate and bladder neck. Brit J Urol. 1998;81(6):830-3.

[7] Yu X. , Elliott S.P. ,Wilt T. J., McBean A.M.(2009). Practice patterns in benign prostatic hyperplasia surgical therapy: the dramatic increase in minimally invasive technologies. J Urol, 180: 241-245.

[8] Madersbacher S, Marberger M. (1999). Is transurethral resection of the prostate still justified? Br. J Urol.

[9] Gamage P., Xie S. Q., Delmas P., Xu W. L., (2011). Diagnostic radiograph based 3D bone reconstruction framework: Application to the femur. Comput. Med. Imaging Graph, 35, 427–437.

[10] P. E. Fadero and M. Shah, “Three dimensional (3D) modelling and surgical planning in trauma and orthopaedics.,” Surgeon, vol. 12, no. 6, pp. 6–11, 2014.

[11] D. Cool, D. Downey, J. Izawa, J. Chin, and A. Fenster, “3D prostate model formation from non-parallel 2D ultrasound biopsy images,” Med. Image Anal., vol. 10, pp. 875–887, 2006.

[12] F. a. Cosío, “Automatic initialization of an active shape model of the prostate,” Med. Image Anal., vol. 12, pp. 469–483, 2008.

[13]A. S. Korsager, U. L. Stephansen, J. Carl, and L. R. Østergaard, “The use of an active appearance model for automated

[14] V. Singh, “Introduction” in Digital Image Processing with MatLab and LabView, Bangalore, India, 2013,pp. 2

[15] M. Kwacz, J. Wysocki, and P. Krakowian, “Reconstruction of the 3D Geometry of the Ossicular Chain Based on Micro-CT Imaging,” Biocybern. Biomed. Eng., vol. 32, no. 1, pp. 27–40, 2012.

[16] E. Bermejo, O. Cordón, S. Damas, and J. Santamaría, “A comparative study on the application of advanced bacterial foraging models to image registration,” Inf. Sci. (Ny)., vol. 295, pp. 160–181, 2015.

[17] L. Shen, S. Kim, and A. J. Saykin, “Fourier method for large-scale surface modeling and registration,” Comput. Graph., vol. 33, no. 3, pp. 299–311, 2009.

[18] P. Markelj, D. Tomazevic, B.Likar and F. Pernus. “A review of 3D/2D registration ethods for image-guided interventions,” Journal of Medical Image analysis. 16, 2012.

[19] H. Allioui, M. Sadgal and A Elfazzki., “A cooperative approach for 3D image segmentation”, IEEE, 2016

[20] O. Friman, M. Hindennach, C. Kühnel, and H. O. Peitgen, “Multiple hypothesis template tracking of small 3D vessel structures,” Med. Image Anal., vol. 14, no. 2, pp. 160–171, 2010.

[21] J. O. Pentecost, J. Icardo, and K. L. Thornburg, “3D computer modeling of human cardiogenesis.,” Comput. Med. Imaging Graph., vol. 23, pp. 45–49, 1999.

[22] R. Audigier, R. Lotufo, and A. Falcão, “3D visualization to assist iterative object definition from medical images,” Comput. Med. Imaging Graph., vol. 30, pp. 217–230, 2006.

[23] H.F.Garc´ıa† et al., “3D Brain Atlas Reconstruction Using Deformable Medical Image Registration: Application to Deep Brain Stimulation Surgery”. Harvard publications,

[24] C. Kim, J Yoon and Y.J Lee. “Medical image segmentation by more sensitive adaptive thresholding”, 2016

[25] A. Skalski, T. Dreweniak, and J. Jakubowski. “Kidney Tumor segmentation and detection on computed tomography data”. 2016

[26] L. Zhu, Y Li, Y Yu, B Zhang and L Wang. “3D construction for soft tissue of the human body”, 2016.

[27] Cosio F.A.,”Automatic initialization of an active shape model of prostate, Medical Imgae Analysis”, 2008

[28] A. Fenster, A. Ward, C. Crukley, E. Gibson, G. Bauman, J. Gómez, J. Chin, M. Moussa, M. Gaed, and S. Pautler, “3D prostate histology image reconstruction: Quantifying the impact of tissue deformation and histology section location,” J. Pathol. Inform., vol. 4, p. 31, 2013.

# APPENDIX – A: MATHLAB CODES

## A.1 Registration-1st step

```
function varargout = register_new(varargin)
% REGISTER_NEW MATLAB code for register_new.fig
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @register_new_OpeningFcn, ...
                   'gui_OutputFcn',  @register_new_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before register_new is made visible.
function    register_new_OpeningFcn(hObject,    eventdata,    handles,
varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to register_new (see VARARGIN)

% Choose default command line output for register_new
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes register_new wait for user response (see UIRESUME)
% uiwait(handles.figure1);
varargout{1} = handles.output;
% paths           % Running the path setter

global I h w index imdirectory Files currentset impath

index=1;
imdirectory='S1_data/';
Files=dir([imdirectory '/*jpg' ]);
currentset=Files(index).name;
impath=[imdirectory '/' Files(index).name];
I=imread(impath);
```

```
[h, w, ~] = size(I);
axes(handles.axes1)
imshow(I)


set(handles.infobox,'String',[currentset ' Loaded....'])


% --- Outputs from this function are returned to the command line.
function varargout = register_new_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

global I h w
% paths
hold on
imshow(I)

[h, w, ~] = size(I);
axes(handles.axes1)





% --- Executes on button press in top.
function top_Callback(hObject, eventdata, handles)
% hObject    handle to top (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
axes(handles.axes1)
hold on


for i=1:6
    [x(i),y(i)]=ginput(1);
    plot(x(i),y(i),'r*')
end


global mt ct w
[M]=polyfit(x,y,1);
mt= M(1)
ct= M(2)
xc= 1:w;
yc= mt*xc;
plot(xc, yc+ct, 'b');



% --- Executes on button press in left.
function left_Callback(hObject, eventdata, handles)
% hObject    handle to left (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

axes(handles.axes1)
hold on


for i=1:6
    [x(i),y(i)] = ginput(1)
    plot(x(i),y(i),'r*')
end


global ml cl w
[M] = polyfit(x,y,1);
ml= M(1)
cl= M(2)
xc= 1:w;
yc= ml*xc;
plot(xc, yc+cl, 'b');


% --- Executes on button press in right.
function right_Callback(hObject, eventdata, handles)
% hObject    handle to right (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
axes(handles.axes1)
hold on

for i=1:6
    [x(i),y(i)]=ginput(1);
    plot(x(i),y(i),'r*')
end


global mr cr w
[M]=polyfit(x,y,1);
mr= M(1)
cr= M(2)
xc= 1:w;
yc= mr*xc;
plot(xc, yc+cr, 'b');

% --- Executes on button press in bottom.
function bottom_Callback(hObject, eventdata, handles)
% hObject    handle to bottom (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

axes(handles.axes1)
hold on

for i=1:6
    [x(i),y(i)]=ginput(1);
    plot(x(i),y(i),'r*')
end
```

```
global mb cb w I J
[M]=polyfit(x,y,1);
mb= M(1)
cb= M(2)
xc= 1:w;
yc= mb*xc;
plot(xc, yc+cb, 'b');

% J=imrotate(I,rad2deg(atan(poly(mb))));
% axes(handles.axes1)
% imshow(J)




% --- Executes on button press in findvert.
function findvert_Callback(hObject, eventdata, handles)
% hObject    handle to findvert (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

global ml mb mr mt cl cb cr ct currentset  Files index corpath ltx lty
rtx rty rbx rby lbx lby
ltx = (cl-ct)/(mt-ml);
lty = ml*ltx+cl;
rtx = (cr-ct)/(mt-mr);
rty = mr*rtx+cr;
rbx = (cr-cb)/(mb-mr);
rby = mr*rbx+cr;
lbx = (cl-cb)/(mb-ml);
lby = ml*lbx+cl;



axes(handles.axes1)
hold on
plot(ltx,lty,'bd','MarkerEdgeColor','k',...
                'MarkerFaceColor',[.49 1 .63],...
                'MarkerSize',10);

plot(rtx,rty,'cd','MarkerEdgeColor','k',...
                'MarkerFaceColor',[.49 1 .63],...
                'MarkerSize',10);

plot(rbx,rby,'yd','MarkerEdgeColor','k',...
                'MarkerFaceColor',[.49 1 .63],...
                'MarkerSize',10);

plot(lbx,lby,'rd','MarkerEdgeColor','k',...
                'MarkerFaceColor',[.49 1 .63],...
                'MarkerSize',10);

rot_angle = rad2deg(atan(mb));
corpath = 'RESULTS\Variables\NewResearch_1mm_S1\set2\';
save([corpath                  '\'                  Files(index).name
'.mat'],'ltx','lty','lbx','lby','rtx','rty','rbx','rby','rot_angle');
```

```
text(ltx-120,lty-100,'P1','BackgroundColor',[.7                    .9
.7],'HorizontalAlignment','right');
text(lbx-100,lby+100,'P2','BackgroundColor',[.7                    .9
.7],'HorizontalAlignment','right');
text(rbx+100,rby+100,'P3','BackgroundColor',[.7                    .9
.7],'HorizontalAlignment','left');
text(rtx+40,rty-40,'P4','BackgroundColor',[.7                      .9
.7],'HorizontalAlignment','left');





% --- Executes on button press in correctrot.
function correctrot_Callback(hObject, eventdata, handles)
% hObject    handle to correctrot (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

global J I mb

J=imrotate(I,rad2deg(atan(mb)));
axes(handles.axes1)
imshow(J)



% --- Executes on button press in save_next.
function save_next_Callback(hObject, eventdata, handles)
% hObject    handle to save_next (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)



global index currentset Files impath imdirectory I mb J rotpath corpath


index=index+1;

currentset=Files(index).name;
impath=[imdirectory '/' Files(index).name];
I=imread(impath);
axes(handles.axes1);
imshow(I);

% rotpath='Data\rotimages\'
% imwrite(J,[rotpath '\' Files(index).name]);



set(handles.infobox, 'String',[currentset ' Loaded.....'])

set(handles.infobox, 'String',[currentset ' Saved.....'])
```

## A.2 Registration-2nd step

```
%% Initialize
clear all
clc
%% Set Paths
current_sub_folder = 'S1_1mm\';      % Give path
impath = ['raw\' current_sub_folder];
varpath = ['Variables\' current_sub_folder];
savpath = ['Cropped_Output\' current_sub_folder];
addpath('Functions');
% distort = 0;                    % Do you want the output distorted?
1, else 0
% scale_factor = 1;              % Scale using widths? 1, else 0
ImDir = dir(impath);

%% General functions for all images
% if distort == 1
%     savpath = [savpath 'distorted\'];
% else
%     if scale_factor == 1
%         savpath = [savpath 'original\width\'];
%     else
%         savpath = [savpath 'original\height\'];
%     end
% end

% if unavailable, find and save min_blk_height and blk_height params
at
% variables path

if   ~exist([varpath   'min_blk_height.mat'],'file')||~exist([varpath
'blk_heights.mat'],'file')||~exist([varpath
'blk_widths.mat'],'file')||~exist([varpath
'min_blk_width.mat'],'file')
    [blk_heights,blk_widths,min_blk_height,min_blk_width]           =
findMinBlockHeight(varpath);
else
    load([varpath 'min_blk_height.mat'])
    load([varpath 'blk_heights.mat'])
    load([varpath 'blk_widths.mat'])
    load([varpath 'min_blk_width.mat'])
end



%% Image Explorer
for i = 3:length(ImDir)           % Correct code
    % for i = 3:3                         % Comment after finding all
coordinates
    current_image = imread([impath ImDir(i).name]);

    %% Rotation of Image and tracking intersect points...
    [im_height, im_width, ~] = size(current_image);
    cx = round(im_width/2);
    cy = round(im_height/2);

    load([varpath ImDir(i).name '.mat']);
    rotated_image = imrotate(current_image,rot_angle,'crop');       %
Rotate the image
    inv_rot_angle = -rot_angle;                                     %
Invert the angle to facilitate coord rot.
    old_coord    =    [lbx-cx,lby-cy,ltx-cx,lty-cy,rbx-cx,rby-cy,rtx-
cx,rty-cy];  % Old_Coord @origin
    rot_mat    =    [cosd(inv_rot_angle),    sind(inv_rot_angle);    -
sind(inv_rot_angle), cosd(inv_rot_angle)];

    for j = 1:2:length(old_coord)-1
        new_coord(j:j+1)  =  old_coord(j:j+1)*rot_mat;              %
Coord(x,y)*rotation_matrix
    end
    new_coord(1:2:end) = new_coord(1:2:end)+cx;            % Bias the
coordinates back to center
    new_coord(2:2:end) = new_coord(2:2:end)+cy;

    %       Visualize  all  outputs  -  Warning!  Comment  before  running
bulk!!!
    %     figure
    %     subplot 121
    %     imshow(current_image)
    %     hold on
    %     plot([lbx ltx rbx rtx],[lby lty rby rty],'g*');
    %     subplot 122
    %     imshow(rotated_image)
    %     hold on
    %     plot([lbx ltx rbx rtx],[lby lty rby rty],'g*');
    %     plot(new_coord(1:2:end),new_coord(2:2:end),'r*');
    %     title('Old-coord - green, New-coord - red');

    %% Rescaling all images and tracking intercept points...

%     if distort==1              % Scale by both height and width
%           height_scale_factor = min_blk_height/blk_heights(i-2); %
Down scaling only! (height)
%          width_scale_factor = min_blk_width/blk_widths(i-2);      %
Down scaling only! (width)
%                                                 rescaled_image     =
imresize(rotated_image,[height_scale_factor*im_height
width_scale_factor*im_width]);
%         rescaled_coord = new_coord;
%                                         rescaled_coord(2:2:end)     =
height_scale_factor*rescaled_coord(2:2:end);
%                                         rescaled_coord(1:2:end)     =
width_scale_factor*rescaled_coord(1:2:end);
%
%     else                         % Preserve original aspect ratio
%         if scale_factor == 1     % Scale using widths
%            width_scale_factor = min_blk_width/blk_widths(i-2);     %
Down scaling only! (width)
%                                                  rescaled_image    =
imresize(rotated_image,width_scale_factor);
%             rescaled_coord = width_scale_factor*new_coord;
%         else                     % Scale using heights
%              height_scale_factor = min_blk_height/blk_heights(i-2);
% Down scaling only! (height)
%                                                  rescaled_image    =
imresize(rotated_image,height_scale_factor);
%             rescaled_coord = height_scale_factor*new_coord;
%         end
%     end


%     figure;
%     imshow(rescaled_image)

 height_scale_factor = min_blk_height/blk_heights(i-2); % Down scaling
only! (height)
    width_scale_factor  =  min_blk_width/blk_widths(i-2);       %  Down
scaling only! (width)


    % These two lines are for a non-distorted image!
        rescaled_image = imresize(rotated_image,height_scale_factor);
        rescaled_coord = height_scale_factor*new_coord;

    % These four lines are to resize with distortion to get higher
accuracy!

    rescaled_image                                                    =
imresize(rotated_image,[height_scale_factor*im_height
width_scale_factor*im_width]);
    rescaled_coord = new_coord;
    rescaled_coord(2:2:end)                                           =
height_scale_factor*rescaled_coord(2:2:end);
    rescaled_coord(1:2:end)                                           =
width_scale_factor*rescaled_coord(1:2:end);

    %         Visualize all outputs - Warning! Comment before running
bulk!!!
    %         figure
    %         subplot 121
    %         imshow(rotated_image)
    %         hold on
    %         plot([lbx ltx rbx rtx],[lby lty rby rty],'g*');
    %         plot(new_coord(1:2:end),new_coord(2:2:end),'r*');
    %         title('Old-coord - green, New-coord - red');
    %         subplot 122
    %         imshow(rescaled_image)
    %         hold on
    %         plot(new_coord(1:2:end),new_coord(2:2:end),'g*')
    %
plot(rescaled_coord(1:2:end),rescaled_coord(2:2:end),'r*');
    %         title('Old-coord - green, New-coord - red');

    %% Cropping the image to the size of the block
    %    old_coord  =  [lbx-cx,lby-cy,ltx-cx,lty-cy,rbx-cx,rby-cy,rtx-
cx,rty-cy];  % Old_Coord @origin
%         cropped_image  =  imcrop(rescaled_image,[rescaled_coord(3)
rescaled_coord(4)            abs(rescaled_coord(7)-rescaled_coord(3))
abs(rescaled_coord(4)-rescaled_coord(2))]);
%     imwrite(cropped_image,[savpath ImDir(i).name]);



   cropped_image      =      imcrop(rescaled_image,[rescaled_coord(3)
rescaled_coord(4)            abs(rescaled_coord(7)-rescaled_coord(3))
abs(rescaled_coord(4)-rescaled_coord(2))]);
    [crop_height,crop_width,~] = size(cropped_image);
    %       Visualize  all  outputs  -  Warning!  Comment  before  running
bulk!!!
    %     [crop_height,crop_width,~] = size(cropped_image);
    %
    %     figure
    %     imshow(cropped_image)
    %      title([num2str(crop_height) '\times' num2str(crop_width) '
= ' num2str(crop_height*crop_width)]);

     figure
    imshow(cropped_image)
    title([num2str(crop_height)  '\times'  num2str(crop_width)  '  =  '
num2str(crop_height*crop_width)]);
    J=cropped_image;
   imwrite(J,[savpath '\' ImDir(i).name ]);
end
```

## A.3 Registration Analysis

```
clear all
clc

%% Load data for analysis
current_sub_folder = 'S1\new\';      % Give path
varpath = ['Variables\' current_sub_folder];
% crop_path = ['Cropped_Output\' current_sub_folder];
crop_path=['Cropped_Output\' current_sub_folder];
impath = ['raw\' current_sub_folder];
if exist([varpath 'Analysis_output.mat'],'file')
    load([varpath 'Analysis_output.mat'])
else

    load([varpath 'blk_heights.mat'])
    load([varpath 'blk_widths.mat'])

    %% Raw Variation in ratio and area
    Tau_raw = blk_heights./blk_widths;
    Rho_raw = blk_heights.*blk_widths;

    %% Variation in crop with no distortion - height param
    Crop_no_distort_path = [crop_path 'original\height\'];
    CropNoDistDir = dir(Crop_no_distort_path);
    crop_orig_height   =   zeros(1,length(CropNoDistDir)-2);         %
Definition
    crop_orig_width = crop_orig_height;
    for i = 3:length(CropNoDistDir)
        [crop_orig_height(i-2),crop_orig_width(i-2),~]              =
size(imread([Crop_no_distort_path CropNoDistDir(i).name]));
    end
    Tau_rch = crop_orig_height./crop_orig_width;
    Rho_rch = crop_orig_height.*crop_orig_width;

    %% Variation in crop with no distortion (Tau_rc) - width param
    Crop_no_distort_path = [crop_path 'original\width\'];
    CropNoDistDir = dir(Crop_no_distort_path);
    crop_orig_height  =  zeros(1,length(CropNoDistDir)-2);          %
Definition
    crop_orig_width = crop_orig_height;
    for i = 3:length(CropNoDistDir)
        [crop_orig_height(i-2),crop_orig_width(i-2),~]              =
size(imread([Crop_no_distort_path CropNoDistDir(i).name]));
    end
    Tau_rcw = crop_orig_height./crop_orig_width;
    Rho_rcw = crop_orig_height.*crop_orig_width;

    %% Variation in after crop - with distorted
    Crop_distort_path = [crop_path 'distorted\'];
    CropDistDir = dir(Crop_no_distort_path);
    crop_dist_height  =  zeros(1,length(CropNoDistDir)-2);          %
Definition
    crop_dist_width = crop_orig_height;
    for i = 3:length(CropNoDistDir)
        [crop_dist_height(i-2),crop_dist_width(i-2),~]              =
size(imread([Crop_distort_path CropDistDir(i).name]));
    end
    Tau_dc = crop_dist_height./crop_dist_width;
    Rho_dc = crop_dist_height.*crop_dist_width;

end

%% View ratio outputs
figure
set(gcf, 'Position', get(0,'Screensize')); % Maximize figure.
plot(Tau_raw,'r^')
hold on
plot(Tau_rch,'b.')
plot(Tau_rcw,'m.')

plot(Tau_dc,'ko')
grid minor
title('Measured block height-width ratio comparison');
xlabel('Image Number - subfolder-S3');
ylabel('Variants of \tau = Height:Width');
legend('\tau_{raw}','\tau_{rch}','\tau_{rcw}','\tau_{dc}','Location',
'northwest')



%% View area outputs
figure
set(gcf, 'Position', get(0,'Screensize')); % Maximize figure.
plot(Rho_raw,'r^')
hold on
plot(Rho_rch,'b.')
plot(Rho_rcw,'m.')
plot(Rho_dc,'ko')
grid minor
title('Measured block area comparison');
xlabel('Image Number - subfolder-S3');
ylabel('Variants of \rho = Height\times Width');
legend('\rho_{raw}','\rho_{rch}','\rho_{rcw}','\rho_{dc}','Location',
'southeast')
```

```
save([varpath                                'Analysis_output.mat'],
'Tau_raw','Tau_rch','Tau_rcw','Tau_dc','Rho_raw','Rho_rch','Rho_rcw',
'Rho_dc');

%% Showing the difference in the form of images
% Finding the worst affected image
[~,max_diff_idx] = max(abs(Tau_raw-Tau_dc));
Im_dir = dir(im_path);
figure
set(gcf, 'Position', get(0,'Screensize')); % Maximize figure.
subplot 221
imshow([im_path Im_dir(max_diff_idx-3).name]);  % Show the raw image
title('Original Image');
subplot 222
imshow([crop_path 'distorted\' Im_dir(max_diff_idx-3).name]);  % Show
the distorted crop
title('Distorted');
subplot 223
imshow([crop_path 'original\height\' Im_dir(max_diff_idx-3).name]);  %
Show the height scaled crop
title('Non-Distorted Height Scaled');
subplot 224
imshow([crop_path 'original\width\' Im_dir(max_diff_idx-3).name]);  %
Show the widht scaled crop
title('Non-Distorted Width Scaled');
```

## A.4 Segmentation

```
function varargout = MAIN(varargin)
% MAIN MATLAB code for MAIN.fig
%      MAIN, by itself, creates a new MAIN or raises the existing
%      singleton*.
%
%      H = MAIN returns the handle to a new MAIN or the handle to
%      the existing singleton*.
%
%        MAIN('CALLBACK',hObject,eventData,handles,...) calls the
local
%        function named CALLBACK in MAIN.M with the given input
arguments.
%
%       MAIN('Property','Value',...) creates a new MAIN or raises
the
%      existing singleton*.  Starting from the left, property value
pairs are
%      applied to the GUI before MAIN_OpeningFcn gets called.  An
%       unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to MAIN_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows
only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES
% Edit the above text to modify the response to help MAIN

% Last Modified by GUIDE v2.5 11-Mar-2016 10:56:13

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
    'gui_Singleton',  gui_Singleton, ...
    'gui_OpeningFcn', @MAIN_OpeningFcn, ...
    'gui_OutputFcn',  @MAIN_OutputFcn, ...
    'gui_LayoutFcn',  [] , ...
    'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT



% --- Executes just before MAIN is made visible.
function MAIN_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to MAIN (see VARARGIN)

% Choose default command line output for MAIN
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

addpath('Functions')
% UIWAIT makes MAIN wait for user response (see UIRESUME)
% uiwait(handles.figure1);
%
global impath I
global currentSet  Files Imdirectory index
index = 1;
Imdirectory='Data\S3\';
Files = dir([Imdirectory '/*.jpg']);
currentSet = Files(index).name;
impath = [Imdirectory '/' Files(index).name];
I = imread(impath);



% set(handles.infoBox,'String',['Image ' impath ' loaded. Select
Left and Right masks...'])

%%

% global fname
% % fname = 'P3290177';
% I = imread(['Data\S3\' fname '.jpg']);

% axes(handles.axes1);
% imshow(I);

% I = rgb2gray(imread('P3290224.jpg'));
% imshow(I)
%  str = 'Click to select initial contour location. Double-click to
confirm and proceed.';
% title(str,'Color','b','FontSize',12);
% disp(sprintf('\nNote: Click close to object boundaries for more
accurate result.'))
% mask = roipoly;
%
% figure, imshow(mask)
% title('Initial MASK');
% maxIterations = 200;
% bw = activecontour(I, mask, maxIterations, 'Chan-Vese');
%
% % Display segmented image
% figure, imshow(bw)
% title('Segmented Image');
%
global I impath
axes(handles.axes1);
imshow(I);

global radiusR
global radiusL
radiusR = 30;
radiusL = 30;
global xR
global yR
global xL
global yL ix iy

[ix, iy, ~]=size(I);

xL = 100;
yL = 100;
xR = 200;
yR = 200;



global hcirc
if radiusL > 0
    hcirc = viscircles([xL yL],radiusL,'EdgeColor','m');
end
global hcircR
if radiusR > 0
    hcircR = viscircles([xR yR],radiusR,'EdgeColor','r');
end




% --- Outputs from this function are returned to the command line.
function varargout = MAIN_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;



% --- Executes on mouse press over figure background, over a
disabled or
% --- inactive control, or over an axes background.
function figure1_WindowButtonDownFcn(hObject, eventdata, handles)
% hObject    handle to figure1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


global xL xR yL yR radiusR radiusL
global hcirc
global hcircR
global flag

if strcmp(get(handles.figure1,'selectionType') , 'normal')
    flag = 1;                       % Left click = 1
    axes(handles.axes1);
    [xL,yL] = ginput(1);

    if radiusL > 0
        delete(hcirc)
    end
    if radiusL > 0
        hcirc = viscircles([xL yL],radiusL,'EdgeColor','m');
    end
end
if strcmp( get(handles.figure1,'selectionType') , 'alt')
    flag = 0;               % Right click = 0
    axes(handles.axes1);
    [xR,yR] = ginput(1);
    if radiusR > 0
        delete(hcircR)
    end
    if radiusR > 0
        hcircR = viscircles([xR yR],radiusR,'EdgeColor','r');
    end
end



% --- Executes on scroll wheel click while the figure is in focus.
function figure1_WindowScrollWheelFcn(hObject, eventdata, handles)
% hObject    handle to figure1 (see GCBO)
% eventdata  structure with the following fields (see FIGURE)
%   VerticalScrollCount: signed integer indicating direction and
number of clicks
%   VerticalScrollAmount: number of lines scrolled for each click
% handles    structure with handles and user data (see GUIDATA)
set(gcf, 'WindowScrollWheelFcn', @figScroll);

function figScroll(src,evnt)
direction = evnt.VerticalScrollCount;
```

```
global radiusR radiusL
global hcirc
global xL xR
global yL yR
global hcircR
global flag

if flag
    if radiusL - 5*direction < 0
        radiusL = 5;
    else
        radiusL = radiusL - 5*direction;
    end
    if radiusL > 0
        delete(hcirc)
    end

    if radiusL > 0
        hcirc = viscircles([xL yL],radiusL,'EdgeColor','m');
      end

else
    if radiusR - 5*direction < 0
        radiusR = 5;
    else
        radiusR = radiusR - 5*direction;
    end
    if radiusR > 0
        delete(hcircR)
    end

    if radiusR > 0
        hcircR = viscircles([xR yR],radiusR,'EdgeColor','r');
    end
end


% --- Executes on button press in addmask.
function addmask_Callback(hObject, eventdata, handles)
% hObject    handle to addmask (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global xL yL xR yR ix iy
global I

global radiusL radiusR currentSet meshL meshR

meshL = mesh_grid(xL, yL, ix, iy, radiusL);
meshR = mesh_grid(xR, yR, ix, iy, radiusR);
% axes(handles.axes1);
% I = rgb2gray(imread('P3290262.jpg'));
% imshow(I)
% BW = roipoly;
%
%
% maxIterations = 200;
% seg = activecontour(I, mask, maxIterations, 'Chan-Vese');
```

```
%
% % Display segmented image
% figure, imshow(seg)
% title('Segmented Image');



save(['Masks\' currentSet(1:end-4) '.mat'],'meshL','meshR')
set(handles.infoBox,'String',['Mask for ' currentSet ' saved...'])




% --- Executes on button press in optimize.
function optimize_Callback(hObject, eventdata, handles)
% hObject    handle to optimize (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
addpath('Functions\Optimizer');
global I meshL meshR hOvm
%                            optiMaskL                            =
deploy_snake(rgb2gray(I),meshL,maxIterations,algorithm,smoothness)
;
%                            optiMaskR                            =
deploy_snake(rgb2gray(I),meshR,maxIterations,algorithm,smoothness)
;
size(I)
size(meshL)
algorithm = get(handles.chanvase,'Value');
iterations = str2double(get(handles.iterations,'String'));
smoothness = 2*get(handles.smooth,'Value');


optiMaskL                                                         =
deploy_snake(rgb2gray(I),meshL,iterations,algorithm,smoothness);
optiMaskR                                                         =
deploy_snake(rgb2gray(I),meshR,iterations,algorithm,smoothness);
axes(handles.axes1)
if exist('hOvm')
    delete(hOvm)
end
hOvm = alphamask(optiMaskL|optiMaskR);


  h = msgbox({'Operation' 'Completed'});




% --- Executes on button press in next.
function next_Callback(hObject, eventdata, handles)
% hObject    handle to next (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global Files Imdirectory currentSet index meshL meshR I
index = index+1;
currentSet = Files(index).name;
impath = [Imdirectory '/' Files(index).name];
I = imread(impath);
axes(handles.axes1);
imshow(I);
```

```
global radiusR
global radiusL
radiusR = 30;
radiusL = 30;
global xR
global yR
global xL
global yL ix iy

[ix, iy, ~]=size(I);

xL = 100;
yL = 100;
xR = 200;
yR = 200;


global hcirc
if radiusL > 0
    hcirc = viscircles([xL yL],radiusL,'EdgeColor','m');
end
global hcircR
if radiusR > 0
    hcircR = viscircles([xR yR],radiusR,'EdgeColor','r');
end




% save(['OptiMasks\' currentSet(1:end-4) '.mat'],'meshL','meshR')
% save(['OptiMasks\' currentSet(1:end-4) '.raw'],'meshL','meshR')
save(['OptiMasks\' currentSet(1:end-4) '.mat'],'meshL','meshR')


set(handles.infoBox,'String',['Mask for ' currentSet ' saved...'])


% --- Executes during object creation, after setting all properties.
function iterations_CreateFcn(hObject, eventdata, handles)
% hObject    handle to iterations (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called



function iterations_Callback(hObject, eventdata, handles)
% hObject    handle to iterations (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of iterations as
text
%        str2double(get(hObject,'String')) returns contents of
iterations as a double



% --- Executes on slider movement.
```

```
function smooth_Callback(hObject, eventdata, handles)
% hObject    handle to smooth (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine
range of slider


% --- Executes during object creation, after setting all properties.
function smooth_CreateFcn(hObject, eventdata, handles)
% hObject    handle to smooth (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: slider controls usually have a light gray background.
if                         isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end
```

## A.5 Preparation for modelling

```
function align_urethra
fname='S7';                          % call images from subset:S7
A=dir(['prostate_mask\' fname]);      % list the folder contains:
'prostate mask'
mkdir('binMatrices');             % make a new folder: 'binMatrices'
n=length(A);                            % run all the images inside
variable:'A'
count=1;                             % 1st count is taken as 1

%convert to 1 layer
for i=3:n
    load(['prostate_mask\' fname '\' A(i).name]);       % load the:
'prostate mask'
    maskedRgbImage=(maskedRgbImage(:,:,1)>0);            % convert
to 1D binary
    save(['binMatrices\' A(i).name],'maskedRgbImage');   % save it
inside: 'binMatrices'
end


% Combine 3 layers
mkdir('CombinedMatrices')                                    % make
a new folder: 'combinedMatrices'
for i=3:n
    load(['binMatrices\' A(i).name]);                        % load
the: 'binMatrices'
    load(['ducts_n_urethra\' fname '\' A(i).name]);          % load
the: 'ducts_n_urethra'
    FinalMesh=meshL + meshM + meshR;                        % Final
mesh  comprise  of  meshL(~Left  Duct),meshM(~Urethra)&  meshR(~Right
Duct)
```

```
    maskedRgbImage=maskedRgbImage-FinalMesh;                    %
Determine the: maskedRgbImage(~Prostate mask)
    save(['CombinedMatrices\' A(i).name],'maskedRgbImage'); % save
maskedRgbImage in the: 'combinedMatrices'
end

% Overlap Urethra
mkdir('overlappedMatrices')                                    % make
a new folder:'overlappedMatrices'

% Take out the reference
load(['ducts_n_urethra\' fname '\' A(3).name]);             % load:
'ducts_n_urethra'
ure_center=bwmorph(meshM,'shrink','inf');                    % shrink
objects to points for meshM (~Urethra)
[urefy,urefx]=find(ure_center);                                % find
the   center(x,y   coordinates)   of   the   1st   image   of   the
Urethra(REFERENCE)



for i=3:n
    load(['combinedMatrices\'  A(i).name]);                    % load
the: 'combinedMatrices'
   load(['ducts_n_urethra\' fname '\' A(i).name]);             % load
: 'ducts_n_urethra'
   ure_center=bwmorph(meshM,'shrink','inf');                 % shrink
objects to points for meshM (~Urethra)for the rest of the  images
[uy,ux]=find(ure_center);                                      % find
the center(x,y) of the urethra for rest of the images within the
same set(set:S7)
dy=urefy-uy;                                                   % find
translated y distance (dy)
dx=urefx-ux;                                                   % find
translated x distance (dx)
FinalMesh=meshL + meshM + meshR;                              % final
mesh  consist  of  meshL(~Left  Duct),meshM(~Urethra)&  meshR(~Right
Duct)
    maskedRgbImage=maskedRgbImage-FinalMesh;                   % find
the maskedRgbImage (~prostate mask)

maskedRgbImage=imtranslate(maskedRgbImage,[dy,dx],'fillvalues',255
,'outputview','full'); % translated all the objects by the value of
[dy,dx]
save(['overlappedMatrices\' A(i).name],'maskedRgbImage');   % save
it as : 'overlappedMatrices'
end



% % testing purposes
% imshow(maskedRgbImage)
% hold on
% plot(urefx,urefy,'rx');
% pause(5)
%
%
% imshow(maskedRgbImage)
%
% figure;
% plot(ux,uy,'gx');
% hold off
```

```
%
%    stackedMat(:,:,i-2)=maskedRgbImage(urefy-682:urefy+350,urefx-
443:urefx+550);
% end

% find the largest dim
A=dir('combinedMatrices');
a=0;
b=0;
n=length(A)-2;                                % nnumber of matrices
in the stack
for i=3:n+2
    load(['combinedMatrices\'  (A(i).name)]);        %  load  the:
'combinedMatrices'
    [p,q]=size(maskedRgbImage);                 % find the rows and
columns  (size)  of the maskedRgbImage
    if p>a                                       % Find the largest
height
       a=p;
    end
    if q>b                                       % find the largest
width
        b=q;
    end
end

mkdir('PaddedMatrices')                        % make a new folder:
'PaddedMatrices'
for i=3:n+2
    load(['combinedMatrices\'   (A(i).name)]);           %   load:
'combinedMatrices'
    [p,q]=size(maskedRgbImage);                  % find the size of
the maskedRgbImage
    maskedRgbImage=padarray(maskedRgbImage,[round((a-
p)/2)],[round((b-q)/2)]);  %  perform  padding  to  the  largest
dimension
    save(['PaddedMatrices\'          A(i).name],'maskedRgbImage');
% save maskedRgbImage as: 'PaddedMatrices'
end



% Perform Cropping
mkdir('croppedMatrices')                                     % make
a new folder: 'croppedMatrices'

for i = 3:n+2
    load(['paddedMatrices\' (A(i).name)]);                 % load:
'paddedMatrices'
    maskedRgbImage = imcrop(maskedRgbImage,[0  0  b  a]);      % Not
happy with 0 0? (crop the image). now a and b contains the largest
dimensions
    save(['croppedMatrices\' A(i).name],'maskedRgbImage');  % save
the maskedRgbImage as: 'croppedMatrices'
end


% % Perform Stacking
stackedMat = zeros(a,b,n);                      % create a matrix
of all zeros
```

```
for i = 3:n+2
    load(['croppedMatrices\'   (A(i).name)]);             %   load:
'croppedMatrices'
    stackedMat(:,:,i-2) = maskedRgbImage;      % stack the matrices
end

stackedMat(stackedMat == 0) = -100;
stackedMat(stackedMat == 1) = 100;

save('stackedMatrix.mat','stackedMat');        % save the matrices
as: 'stackedMatrix.mat'
end




% % Employed overlapped urethras
% function boundaryurethra
% fname='S7';
% A=dir(['prostate_mask\' fname]);
% n=length(A);
% points=[];
%
% for i=3:n
%      load(['overlappedMatrices\'  A(i).name]);
%     load(['ducts_n_urethra\' fname '\' A(i).name]);
%     [u_shell]=makeshell(meshM);
%     u_p=sorter(u_shell,i);
%     points=[points;u_p];
%    end
%
% assignin('base','points',points);
% X=points(:,1);
% Y=points(:,2);
% Z=points(:,3);
% save(['overlappedMatrices\' A(i).name],'maskedRgbImage');
% end
%
%
% % Obtain Urethra points
%
% function obtainurethrapoints
%
% fname='S7';
% A=dir(['prostate_mask\' fname]);
% n=length(A);
% points=[];
%
% for i=3:n
%     load(['ducts_n_urethra\' fname '\' A(i).name]);
% %     load(['prostate_mask\' fname '\' A(i).name]);
%      load(['PaddedMatrices\' (A(i).name)]);
%     [u_shell]=makeshell(meshM);
%     u_p=sorter(u_shell,i);
%     points=[points;u_p];
% end
%
% assignin('base','points',points);
% X=points(:,1);
% Y=points(:,2);
% Z=points(:,3);
% vtkwrite('UrethraOverlapOnly.vtk','polydata','lines',X, Y, Z)
% end
%
%
%
%
% % % % Make the shell
% function varargout=makeshell(varargin)
% for i=1:nargin
%     maskedRgbImage=maskedRgbImage(:,:,1)>0;
%     bw_im=maskedRgbImage;
%     bw_im=varargin{i};
%     bw_im=bwmorph(bw_im,'remove');
%     varargout{i}=bw_im;
% end
% end
%
%
% % Sort the points
% function points=sorter(bw_im,i)
%    bw_im=bwmorph(meshM,'remove');
%     [idy,idx]=find(bw_im);
%     refx=idx(1);
%     refy=idy(1);
%     sortedx=refx;
%     sortedy=refy;
%     idx(1)=[];
%     idy(1)=[];
%
%     while~isempty(idx)
%         Pointidx=findEucDist(refx,refy,idx,idy);
%         refx=idx(Pointidx);
%         refy=idy(Pointidx);



%         sortedx=[sortedx refx];
%         sortedy=[sortedy refy];
%         idx(Pointidx)=[];
%         idy(Pointidx)=[];
%     end
%
%     sortedx = 0.0292*sortedx;
%     sortedy = 0.0292*sortedy;
%     sortedz = 2*(i-2)*ones(1,length(sortedx));
%     points = [points; sortedx' sortedy' sortedz'];
% end
%
```

1. Stacking

```
% fname='S7'
A = dir(['prostate_mask\S7\']);
load(['prostate_mask\' (A(3).name)]);         % temp loaded
mkdir('binMatrices')
for i = 3:length(A)
    load(['prostate_mask\' (A(i).name)]);
    maskedRgbImage = (maskedRgbImage(:,:,1)>0);
    save(['binMatrices\' A(i).name],'maskedRgbImage');
end

% Combine the three layers now...
```

```
mkdir('combinedMatrices')
for i = 3:length(A)
    load(['binMatrices\' (A(i).name)]);
    load(['ducts_n_urethra\' (A(i).name)]);
%    maskedRgbImage = bwmorph(maskedRgbImage,'remove');
    FinalMesh = meshL + meshM + meshR;
%    maskedRgbImage = maskedRgbImage + FinalMesh;
        maskedRgbImage = maskedRgbImage -FinalMesh;

    save(['combinedMatrices\' A(i).name],'maskedRgbImage');
end

% overlap urethras

mkdir('overlappedMatrices')

% Take out the reference
load(['ducts_n_urethra\' (A(3).name)]);
ure_center = bwmorph(meshM,'shrink','inf');
[urefy, urefx] = find(ure_center);

for i = 4:length(A)
    load(['combinedMatrices\' (A(i).name)]);
    load(['ducts_n_urethra\' (A(i).name)]);
    ure_center = bwmorph(meshM,'shrink','inf');
    [uy, ux] = find(ure_center);
    FinalMesh = meshL + meshM + meshR;
    maskedRgbImage = maskedRgbImage -FinalMesh;
    maskedRgbImage  =  imtranslate(maskedRgbImage,[urefx-ux,urefy-
uy],'FillValues',255);

    save(['overlappedMatrices\' A(i).name],'maskedRgbImage');
end


% find the largest dim

A = dir('combinedMatrices');
a = 0;
b = 0;
n = length(A)-2;    % number of matrices in the stack

for i = 3:n+2
    load(['combinedMatrices\' (A(i).name)]);
    [p,q] = size(maskedRgbImage);
    if p > a
        a = p;
    end
    if q > b
        b = q;
    end
% now a and b contains the smallest dimensions
end

% Perform Padding
mkdir('PaddedMatrices')

for i = 3:n+2
```

```
    load(['combinedMatrices\' (A(i).name)]);
    [p,q] = size(maskedRgbImage);
    maskedRgbImage    =    padarray(maskedRgbImage,[round((a-p)/2),
round((b-q)/2)]);     % Not happy with 0 0?
    save(['paddedMatrices\' A(i).name],'maskedRgbImage');
% now a and b contains the largest dimensions
end

% Find the smallest dimensions
A = dir('overlappedMatrices');
a = 10000;
b = 10000;
n = length(A)-2;    % number of matrices in the stack

for i = 3:n+2
    load(['overlappedMatrices\' (A(i).name)]);
    [p,q] = size(maskedRgbImage);
    if p < a
        a = p;
    end
    if q < b
        b = q;
    end
% now a and b contains the smallest dimensions
end

% Perform Cropping
mkdir('croppedMatrices')

for i = 3:n+2
    load(['paddedMatrices\' (A(i).name)]);
    maskedRgbImage = imcrop(maskedRgbImage,[0  0  b  a]);      %  Not
happy with 0 0?
    save(['croppedMatrices\' A(i).name],'maskedRgbImage');
% now a and b contains the largest dimensions
end

% % Perform Stacking
stackedMat = zeros(a,b,n);
for i = 3:n+2
    load(['croppedMatrices\' (A(i).name)]);
    stackedMat(:,:,i-2) = maskedRgbImage;
end

stackedMat(stackedMat == 0) = -100;
stackedMat(stackedMat == 1) = 100;

save('stackedMatrix.mat','stackedMat');


%% write to vtk file
WriteToVTK(stackedMat,'stacked.vtk')
```