Statistical Atlas of the Human Knee

600.446 Computer Integrated Surgery II

Project Report

Team members: Ceylan Tanes, Murat Bilgel

Mentors: Dr. Russell Taylor, Xin (Ben) Kang

Abstract

We describe a semi-automated statistical atlas building pipeline that uses CT images, and present our results for the atlas of the femur and tibia that were built using this pipeline. The method described is a modified and more automated version of the pipeline developed by Dr. Gouthami Chintalapani [2]. The modified pipeline reduces the user interaction, and thus makes the process more accessible and less prone to errors. The femur and tibia atlases obtained using this procedure have good accuracies that enable them to be used for post-operative evaluations.

1. Introduction

A statistical atlas is a model of an organ that captures the inherent anatomical variability in a given training population. The anatomical properties used in an atlas building method are organ shape and appearance. These properties can be extracted from 3D medical imaging modalities such as CT or MRI. The organ shape is defined as the boundary of the organ, and the appearance is represented by the organ's texture or its intensity values in the medical image. Statistical models that represent only the shape variations are called statistical shape models (SSM), while those that represent both shape and intensity are called appearance models [1]. In this project, since we are only concerned about the shape of the femur and tibia, we will be building a SSM.

There are four steps to statistical atlas construction:

1. Model Representation / Parameterization: The training images are parameterized according to a representation of anatomical properties. This step includes preprocessing and segmentation.
2. Model Correspondence / Alignment (Registration): A template image is selected from the training images and the rest of the images are registered onto it. Different methods used for this step are iterative closest point algorithm, Procrustes algorithm, thin plate spline and mesh-based registration. In our project we have used a deformable 3D-3D registration method called Mjolnir that was developed by Professor Prince et al. at Johns Hopkins University [4].
3. Statistical Analysis: This step is used to create a statistical model which represents the

anatomical variability in the training population. The methods include principal component analysis (PCA), independent component analysis (ICA), kernel PCA and principal geodesic analysis (PGA).

4. Bootstrapping: This is an optional step where the initial atlas is fed back into steps 2 and 3 to eliminate the dependency to the initial template image that was selected manually. By bootstrapping, the resulting atlas converges to a better representation of the organ.

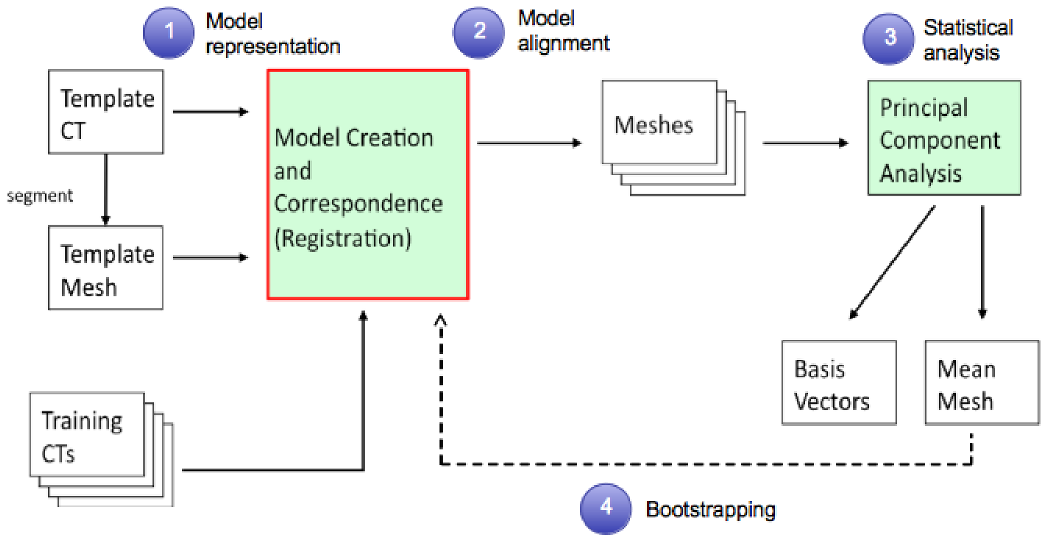


Figure 1. Basic atlas building process [1]

Motivation and Significance

Statistical atlases have a wide range of applications, including monitoring disease progression, accounting for anatomical variation in large populations and surgical planning. A comprehensive statistical atlas of the knee would help in such areas and also aid in post surgical evaluation of knee surgeries such as anterior cruciate ligament (ACL) surgeries.

The main goal of this project is to improve and automate the statistical atlas building pipeline developed by Dr. Gouthami Chintalapani at the Johns Hopkins University [2]. There are several limitations to building statistical atlases. Building an atlas has several steps and requires the user to interact with different programming platforms. The pipeline also depends largely on manual work from the user in terms of dealing with processing the raw data and managing the functions necessary for the process. Since the pipeline is not automated, it is harder for the user to keep track of the following steps in the pipeline. Automating the pipeline makes the atlas building process more accessible to those with little or no programming experience. Also by minimizing the human factor, the atlas building process becomes more standardized, less prone to human error and the time required is significantly shortened.

2. Technical Approach

The main purpose of the project is to automate the atlas building pipeline so that it requires minimum user interaction and also to build statistical atlases of femur and tibia. Therefore we have written user friendly scrips and functions to guide the user through the process so that they don't have to worry about function configurations and parameters.

The atlas building procedure can be broken down into preprocessing the images, creating the template mesh, model correspondence and alignment, statistical analysis and surface mesh creation. An overview of our improved pipeline is presented in Figure 2.

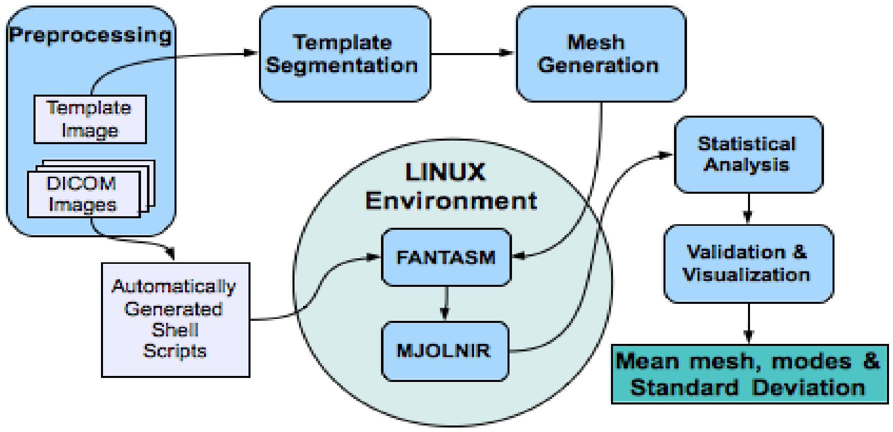


Figure 2. Overview of the improved pipeline

For a detailed explanation on how to run the pipeline using the code provided, please refer to Appendix A.

Preprocessing

To build the atlas, we start with the DICOM images of left/right femur and tibia of cadavers. However the images have to be processed to be accepted as inputs to the next step in the pipeline which is model correspondence and alignment.

Model correspondence and alignment phase requires:

- The images to be saved as 8 bit unsigned integers (uint8). DICOM images are saved as int16. If we use the built in uint8() function, the highest intensity value in each image is mapped to the same highest value possible. This causes inconsistency across different images. To convert them from int16 to uint8 it is suggested to add 1000 to all the values, set the negative values to 0 in order to account for the noise. Finally we divide the values by 16 to convert them from uint16 to uint8 [1]. This type of conversion makes sure that the conversion is consistent among all the images and removes some noise in the data.

- The voxels to be isotropic. This is one of the assumptions in the alignment step. We are using an trilinear interpolation calculate the intensity values in each image to make the voxels isotropic.
- All the patient images to have the same volume size. We keep track of the maximum x,y and z dimensions in all the images being processed. We then iterate through the data and pad them with 0 intensity values to make them the same size.
- It is also recommended that the x and y values are not to be greater than 250 pixels and the z axis to be as small as possible. The images that we have worked with have a lot of empty empty space around the bone. To make the image size smaller, we wrote an algorithm to automatically detect where the anatomical region is and crop around that region. That method relies on the assumption that there is a good enough contrast between the background and the anatomical structure. Due to the structure of femur and tibia, there are about 600-700 slices on the axial plane. To deal with a smaller volume size, we squeezed the z axis by half. It is important to note that during the statistical analysis phase, the data have to be resized again to it's original dimensions.

The preprocessing step deals with these issues one by one and the resulting images are ready to be used as inputs for the model correspondence and alignment phase. The diagram of the procedure is as follows:

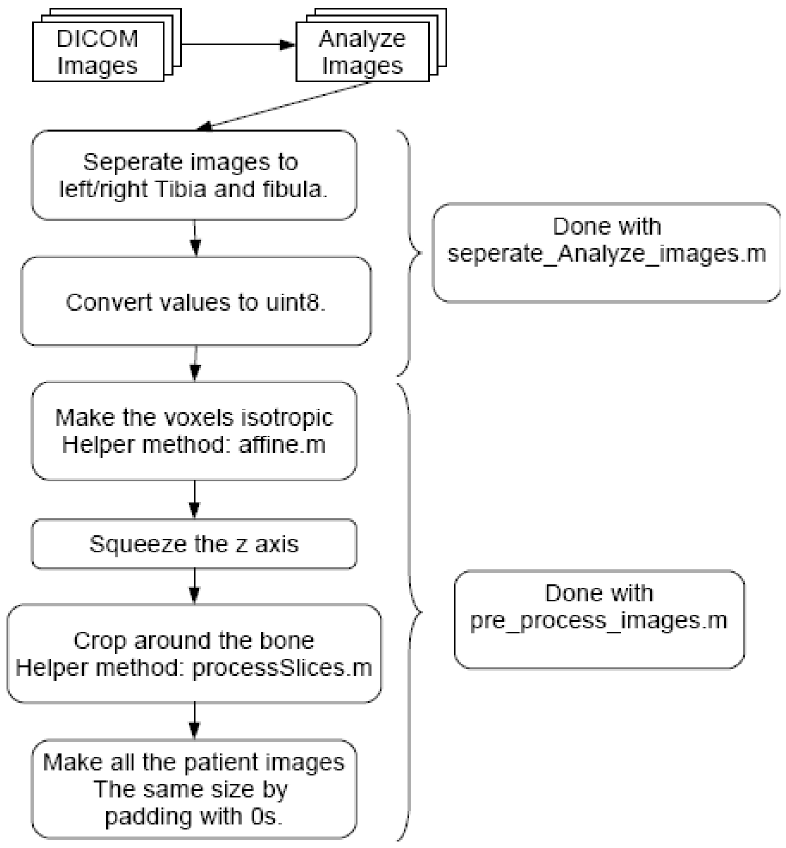

DICOM Images
Analyze Images
Seperate images to left/right Tibia and fibula.
Convert values to uint8.
Done with seperate_Analyze_images.m
Make the voxels isotropic Helper method: affine.m
Squeeze the z axis
Crop around the bone Helper method: processSlices.m
Make all the patient images The same size by padding with 0s.
Done with pre_process_images.m

Figure 3: Flow chart diagram of the preprocessing phase.

Template Segmentation

The atlas building pipeline requires one image to be selected manually as the template. The image should be a good representation of the anatomical structure and shouldn't look like an outlier. This template data is segmented to create its volumetric mesh. The selected template image is segmented semi-automatically using the free software ITK-SNAP and exported as an STL surface mesh [6]. The surface mesh is then converted to a volumetric mesh by using the MATLAB file iso2mesh [5]. This mesh will then be used in the model alignment step to register the rest of the images onto the template mesh.

The diagram of the procedure is as follows:

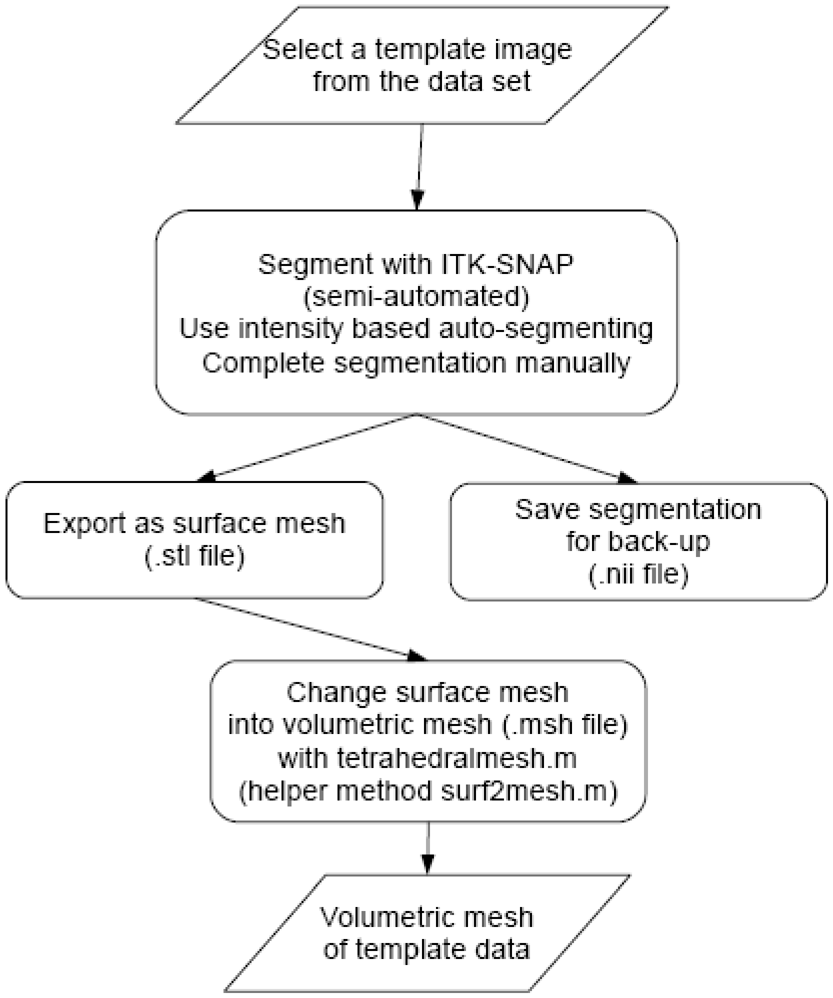

Select a template image from the data set
Segment with ITK-SNAP (semi-automated) Use intensity based auto-segmenting Complete segmentation manually
Export as surface mesh (.stl file)
Save segmentation for back-up (.nii file)
Change surface mesh into volumetric mesh (.msh file) with tetrahedralmesh.m (helper method surf2mesh.m)
Volumetric mesh of template data

Figure 4. Tetrahedral mesh generation from a template image

Model Correspondence and Alignment

In order to analyze the variance of the data, they first need to be registered onto the template mesh in order to find the correspondences between the images. The images are first segmented with FANTASM [7]. FANTASM is a fuzzy segmentation algorithm that assigns a probabilistic value of belonging one of the three classes to each voxel in the image [7]. A sample result from FANTASM is as follows:

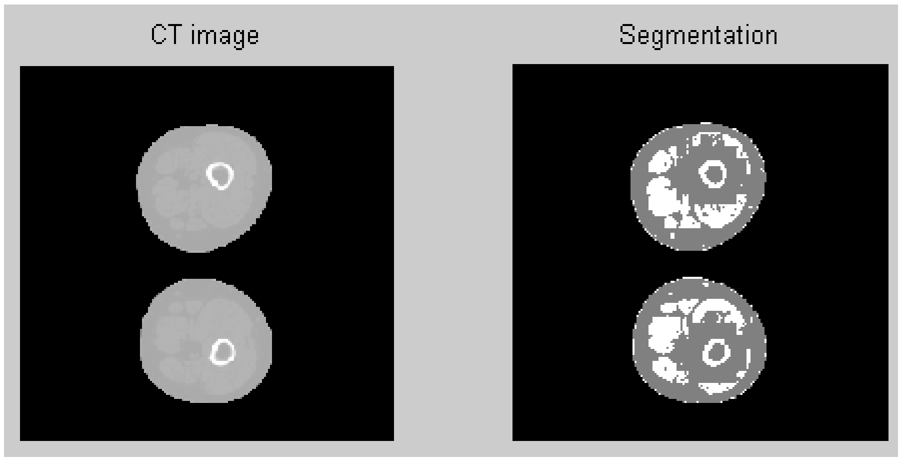


Figure 5: CT slice of a femur on the left and FANTASM segmentation result on the right. You can see the three memberships denoted as black, gray and white on the segmentation result.

After all the data is segmented, they can be registered onto the template mesh to create a warped volume. The method in the pipeline is a 3D-3D deformable registration algorithm called MJOLNIR developed by Professor Prince et al. at Johns Hopkins University [4]. In this algorithm, a deformation field is defined for the vertices of the template mesh. With the information from fuzzy segmentation, MJOLNIR finds the vector field that maps the template mesh to the subject image. By this registration process, a mesh can be created that represents the subject image and the homologous features. A sample result from MJOLNIR is as follows:

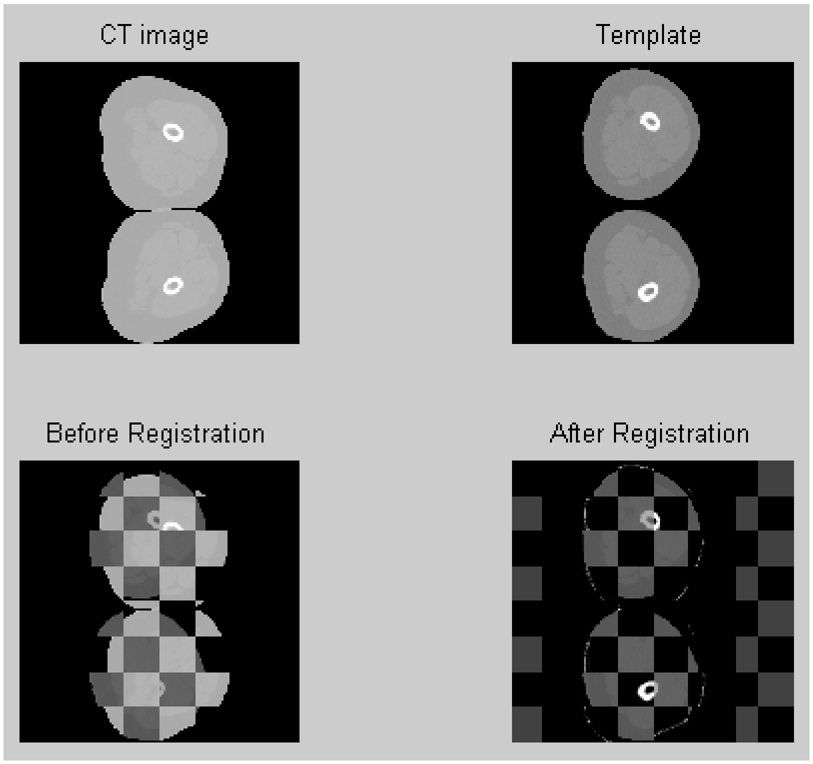


Figure 6: Two CT images are being registered with MJOLNIR. Before registration the images are not aligned but after registration we can observe a good alignment.

FANTASM and MJOLNIR run on the Linux servers of Johns Hopkins University. In our pipeline, we have a function that asks the user the data s/he would like to process and generates a shell script that can be executed on the Linux server to run FANTASM and MJOLNIR with the necessary data.

Statistical Analysis

The first step of the analysis involves rigidly aligning the deformed meshes that are generated by MJOLNIR. For this, we used the rigid alignment method presented in Cootes, et al [3]. After all the deformed subject meshes are aligned with the template mesh, the point-wise mean shape is calculated. Modes of variation are found using principal component analysis (PCA) on the deviations of the examples from the mean.

Surface Mesh Creation

A possible use of the statistical knee atlas is the post-operative evaluation of an ACL surgery. In order to do this, the x-ray images taken after the surgery have to be registered onto the mean mesh of the atlas with a 2D-3D registration algorithm. The input to this algorithm is the mean surface mesh of the knee atlas, the surface modes and the 2D images. In our code, we added a function to convert the mean volumetric mesh resulting from PCA to a surface mesh. This leaves room for improvement in our project.

Validation

In order to validate our results, we estimated the mode weights λ^{est} for each subject mesh used in the atlas construction process with the formula

$$\lambda^{est} = D^T \left(S^{true} - \bar{M} \right) \quad ,$$

where $D = [D_1 \;\; D_2 \;\; \cdots \;\; D_N]$ contains the *N* variational modes, S^{true} is the original aligned subject image, and $\bar{M}$ is the mean image for all the subjects [2]. The estimated subject image was then generated using the orthonormal modes λ^{est} by

$$S^{est} = \bar{M} + \sum_{k=1}^{n} \lambda_k^{est} D_k \quad ,$$

where *n* is the number of modes used in the atlas while estimating the shape [2]. We used the maximum vertex-vertex Euclidean distance between S^{true} and S^{est} to quantify the accuracy of the atlas for different values of *n*.

Evaluation of the Similarity of the Left and Right Femur Atlases

The central axis of the femur head was obtained by fitting a 3-D line to its surface vertices. The mirror image of the right femur was then obtained by flipping the mean right femur mesh about a plane perpendicular to the central axis of the femur head. The right femur mirror image was then rigidly aligned with the left femur image. Similarity of the two atlases was quantified using the maximum vertex-vertex Euclidean distance between the two surfaces.

3. Results and Discussion

Using the atlas building procedure described in this report, atlases of the right femur, left femur, and right tibia were obtained using cadaveric CT scans. Including the template image, the right and left femur atlases were build using 8 data sets, and the right tibia atlas was built using 5. 7 and 5 variational modes were obtained from PCA for the femur and tibia atlases, respectively. Figures 7 and 8 show the first three modes of the right femur and tibia aligned with their mean meshes.

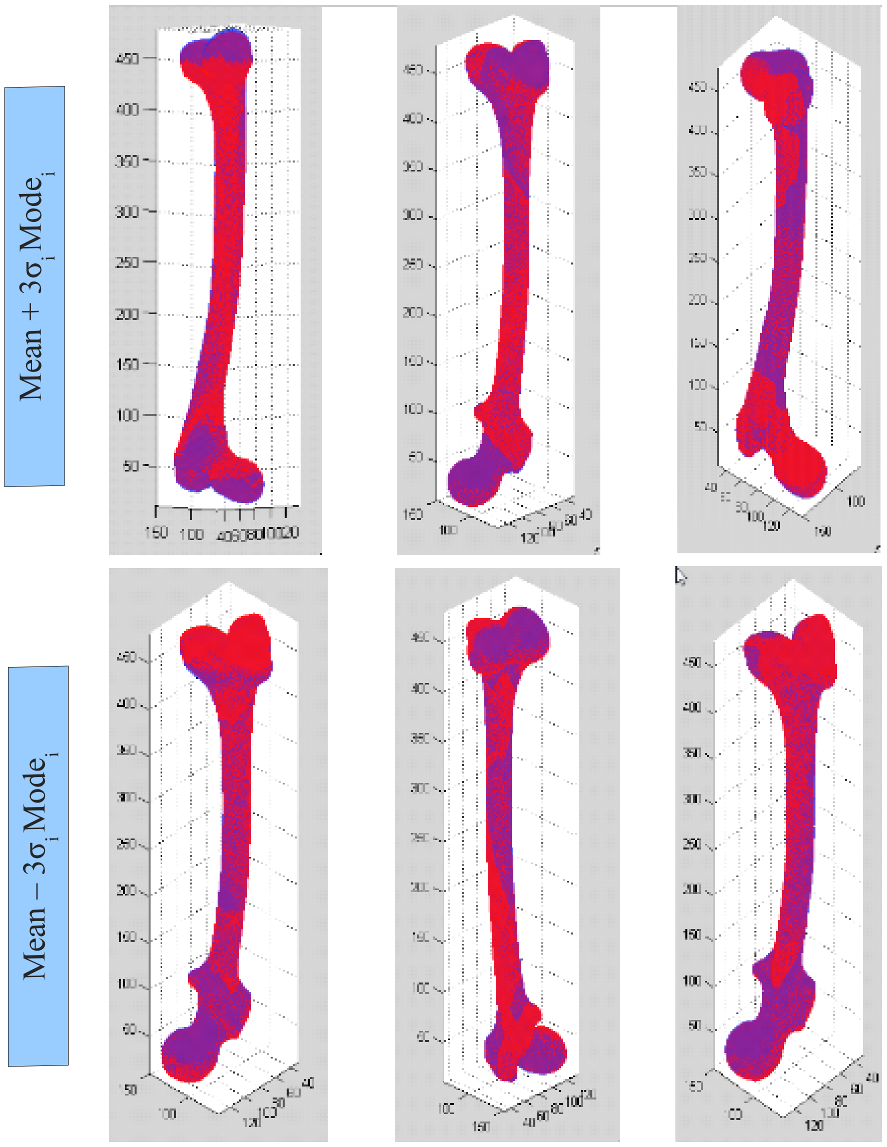


Figure 7. Superimpositions of the mean right femur mesh (blue) and its first 3 mode images (red). Each column corresponds to a different mode. The first row is obtained by adding $3\sigma_i$ mode$_i$ to the mean shape, and the second column is obtained by subtracting this value from the mean shape.

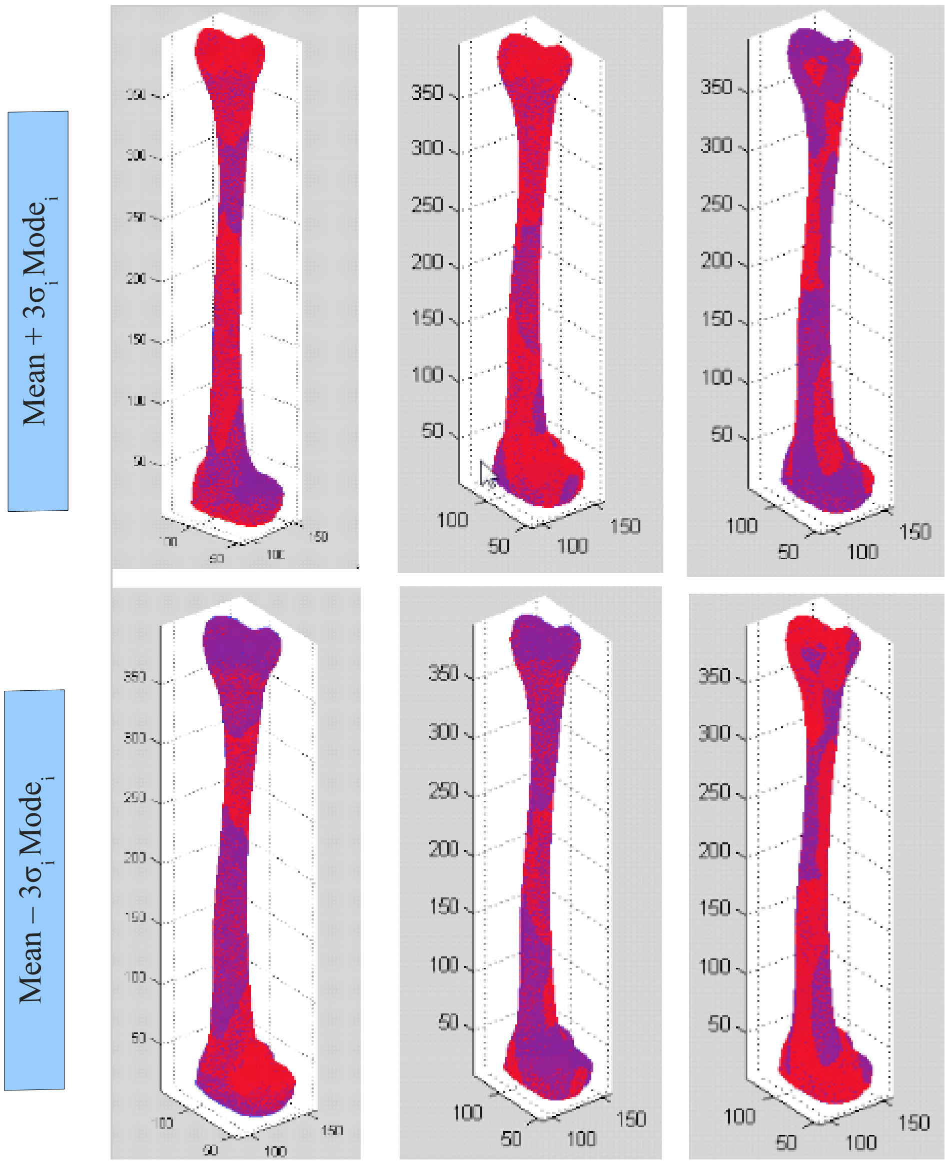


Figure 8. Superimpositions of the mean right tibia mesh (blue) and its first 3 mode images (red). Each column corresponds to a different mode. The first row is obtained by adding $3\sigma_i$ mode$_i$ to the mean shape, and the second column is obtained by subtracting this value from the mean shape.

As can be observed from Figure 7, the first mode of the femur captures length variations (along the z-axis), while the second mode captures girth variations. Even though the femur atlas was built using 8 subjects, the meaning of the modes can be extracted from the results.

We estimated all subjects included in the atlas building process using the validation method described in the previous section, and obtained the mean of the maximum vertex-vertex error values across all the subjects. The results are presented in Figure 9. Using more modes of variation when estimating shapes results in more accurate approximations. The vertex-vertex errors obtained using the 8-subject atlas are within a reasonable range, meaning that atlases with an accuracy required to perform post-surgery evaluations can be obtained using the method described in this report.

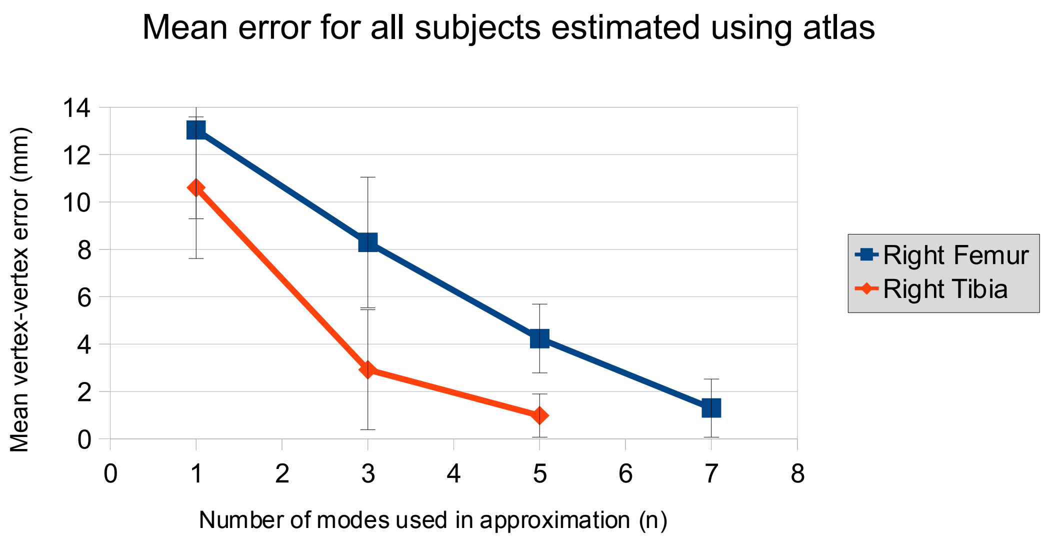


Figure 9. Mean vertex-vertex error across all subjects as a function of the number of modes used in estimating the ground truth image.

The aligned left femur and the mirror image of the right femur meshes are shown in Figure 10. The maximum vertex-vertex distance was 8 mm, indicating that separate atlases for the left and right femur might not be necessary due to the similarity between these two anatomical structures.

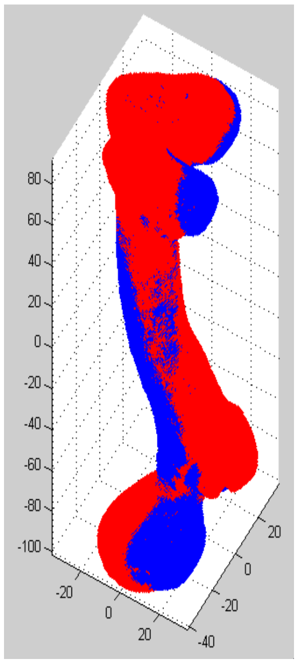


Figure 10. Superimposed meshes of the left femur (red) and the mirror image of the right femur (blue).

5. Conclusion

The integrated pipeline allows for a faster creation of statistical atlases of anatomical structures compared to the original method developed by Dr. Chintalapani. The automation of the preprocessing step has significantly reduced the time required to ensure that the images have the correct dimensions and properties, and made the atlas building process less prone to errors by requiring less user interaction. The improved pipeline enables the users to include more data sets in the atlas easily; hence giving the user control over the accuracy of the resulting atlas.

Working on this project allowed us to understand the atlas building pipeline and gave us an opportunity to research and try different segmentation and registration methods.

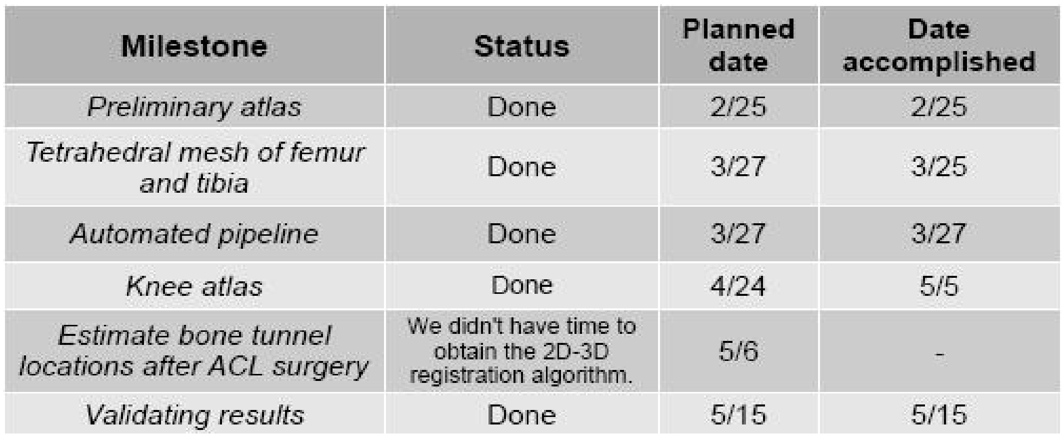
Future Work

The main improvement to our project would be to implement the 2D-3D algorithm to register the x-ray images of ACL surgery patients to our knee atlas in order to estimate the bone tunnel locations. This would allow surgeons to evaluate the success of the surgery post-operatively.
To improve out statistical atlas, we can train more CT images both for the femur and tibia atlases. This would create a better mean mesh and introduce more modes to account for the variability in the data. In order to converge to a more accurate mesh, another improvement would be to implement the bootstrapping phase of the atlas and feed the preliminary atlas back into the model correspondence and alignment phase. This eliminates the dependency of the atlas to the initial template image chosen.

In our current implementation, we are creating separate meshes for the left and right knee. Our preliminary results comparing the mean mesh of the left and right femur show that a single atlas can be created to represent both left and right sides. A better algorithm to take the mirror images of the warped volumes can be developed and the resulting images can be used to improve the atlas. This would double the data set and create a better representation of the knee.

Management Summary

The following table represents our initial plan and what was accomplished:



Milestone	Status	Planned date	Date accomplished
Preliminary atlas	Done	2/25	2/25
Tetrahedral mesh of femur and tibia	Done	3/27	3/25
Automated pipeline	Done	3/27	3/27
Knee atlas	Done	4/24	5/5
Estimate bone tunnel locations after ACL surgery	We didn't have time to obtain the 2D-3D registration algorithm.	5/6	-
Validating results	Done	5/15	5/15

Figure 11. Milestones and progress summary

Unresolved dependencies:

For our project we have worked with cadaver CT images only. They had very high resolution and were very straightforward to segment. Our initial plan was to improve the pipeline by using the cadaver images and build the actual atlas by using patient data. However we decided only to use the cadaver images due to time constraints of running MJOLNIR.

Who did what:

Coding:
- ✔ preprocessing: Ceylan &Murat
- ✔ segmentation: Ceylan &Murat
- ✔ registration: Murat
- ✔ statistical analysis: Murat
- ✔ automating the pipeline: Ceylan & Murat

Final report:
- ✔ Abstract, Results, Discussion: Murat
- ✔ Motivation, Technical approach, Conclusion, Appendix: Ceylan

6. References

1. G. Chintalapani. Statistical Atlases of Bone Anatomy and Their Applications. Diss. Johns Hopkins University, 2010.
2. G. Chintalapani, L.M. Ellingsen, O. Sadowsky, J.L. Prince, and R.H. Taylor. Statistical Atlases of Bone Anatomy: Construction, Iterative Improvement and Validation. MICCAI, 2007. Part I, LNCS 4791, pp. 499-506. N. Ayache, S. Ourselin, A. Maeder (editors).
3. T.F. Cootes, C.J. Taylor, D.H. Cooper, and J. Graham. Active Shape Models – Their Training and Application. Computer Vision and Image Understanding, 1995. Vol 61, No 1, pp. 38-59.
4. L.M. Ellingsen, G. Chintalapani, R.H. Taylor, and J.L. Prince. Robust deformable image registration using prior shape information for atlas to patient registration. Computerized Medical Imaging and Graphics 34 (2010) 79–90.
5. ISO2MESH: A free 3D surface and volumetric mesh generator for matlab/octave. Last modified 27 February 2011. Last accessed 17 May 2011. <http://iso2mesh.sourceforge.net/cgi-bin/index.cgi>
6. ITK-SNAP. Last modified 17 February 2011. Last accessed 17 May 2011. <http://www.itksnap.org/pmwiki/pmwiki.php>
7. D.L. Pham, and J.L. Prince. FANTASM: Fuzzy and Noise Tolerant Adaptive Segmentation Method. IEEE Trans. Medical Imaging, 1999.

APPENDIX A

Preprocessing the images:

INPUT: DICOM images

OUTPUT: preprocessed images saved in Analyze 7.5 format

The detailed explanations of the methods the user is required to run are as follows:

separate_Analyze_Images.m: (MATLAB script)

This script is specific to the anatomical area of interest that the user wants to create the atlas of. In our case, since the CT images include both the right and the left femur/tibia, this method cuts them in half and saves them as left and right anatomical structure. The method also converts the image values to uint8.

INPUT: The script asks the user for directory of the files s/he wants to process.

OUTPUT: The script saves the results in a directory created as 'Analyze_separated_results'

pre_process_images.m: (MATLAB script)

This script asks the user the directory of the images s/he wants to process. The method makes the images isotropic (with the helper method affine.m), resizes the z axis by half, crops the images around the bone (with the helper method processSlices.m) and makes all the patient images the same size by keeping track of the maximum dimensions of the patient images and padding the smaller ones with 0.

In this method we are also considering the fact that not all patient data is pre-processed at the same time. MJOLNIR requires that the template data and all the images have the same origin and have the the same volume size. The user may pre-process and register some data using a template and decide to improve the atlas by adding more data. In this case during the padding step, the method asks the user the name of the template data used previously for MJOLNIR. If there is any input file name, the program makes sure that all the data is processed accordingly so that there won't be any problems running MJOLNIR with additional data.

The method uses analyze75read and analyze75write to read and write analyze files. These functions take the transpose of the images along the x=y axis. This reorientation creates problems while running MJOLNIR. Therefore at the end of the pre-processing, the images are saved with the default file writer of MATLAB. The results can be imported to MATLAB with readRawCTUChar.m function.

Helper methods: (MATLAB scripts)

analyze75read.m:

This method reads the images that are in Analyze 7.5 format.

INPUT: name of the image file.

OUTPUT: the slices returned in a 3D matrix

analyze75info.m:

This method imports the header file associated with a file name

INPUT: name of the image file.

OUTPUT: information about the image file returned in a struct.

analyze75write.m:

This method saves an image and an associated header file in Analyze 7.5 format.

INPUT: the image in a 3D matrix, the file name and other optional parameters specified in the method

readRawCTUChar.m:

This method reads the raw binary format the images are stored as at the end of pre-processing step. [1]

INPUT: volume size which can be found in the vol_size plain text file generated by the pre_process_images method and the file name.

OUTPUT: the slices returned in a 3D matrix

affine.m:

This method uses 3D affine matrix to interpolate the values in the 3D images in order to make them isotropic. [Taken from http://www.mathworks.com/matlabcentral/fileexchange/8797-tools-for-nifti-and-analyze-image/content/affine.m]

INPUT: the image to process, a 4x4 identity matrix, the new element size

OUTPUT: the image that was made isotropic, the transformation matrix.

processSlices:

This method processes the 3D image and finds the smallest bounding cube that encompasses the anatomical structure of interest. The method initially subtracts the first image (assuming that image contains only the bed) from all the rest of the images to get rid of the bed and any other constant artifacts. Then it uses intensity thresholding (in this case the threshold is chosen to be 80) to find the smallest bounding cube around the bone. The function then cuts around that region and returns the resulting cropped image. This method works well if there is a good contrast between the background and the anatomical structure. If the method doesn't work with new images, user can try different intensity thresholds.

INPUT: the 3D image to be processed

OUTPUT: the cropped image

Creating the template mesh:

INPUT: Pre-processed template image selected by the user

OUTPUT: Volumetric mesh of the template image

The detailed explanations of the methods the user is required to run are as follows:

ITK-SNAP: (free software)

This program is used to segment the template image semi-automatically. The detailed tutorial on how to use the software can be found on their website [6]. There are two options to segment the 3D image. One is to use edge information and the other is to use intensity values. In our case the intensity based segmentation should be used because in medical images, the edge information if the organs may not be easily extractable. After the semi-automated segmentation, the segmentation can be completed by manually going through all the slices and labeling the bone that was missed by ITK-SNAP. The resulting volume can then be exported as a surface mesh.

tetrahedralmesh.m: (MATLAB function)

This method asks the user for the name of the surface mesh created by ITK-SNAP and converts it into volumetric mesh and saves the result in a .msh file.

INPUT: 1 if the user wants to visualize the meshes, 0 if not.

Helper methods: (MATLAB scripts)

STL_import:

This method is to read the nodes and vertices of a surface mesh from a file [taken from http://www.softpedia.com/get/Science-CAD/STl-Import.shtml/].

INPUT: The name of the file.

OUTPUT: The nodes, faces and normals to triangles surf2mesh.m

This method converts the surface mesh into volumetric mesh. This function is in the iso2mesh package [5].

INPUT: the vertices, faces of the surface mesh, the two corners of the bounding box.

OUTPUT: the vertices and faces of the tetrahedral mesh generated

Model Correspondence and Alignment:

INPUT: template volumetric mesh, preprocessed subject images

OUTPUT: a warped volume of the template mesh and subject image

The detailed explanations of the methods the user is required to run are as follows:

writeShellScripts.m: (MATLAB function)

This scripts asks the user for the directory of the data s/he would like to process. With the necessary data files extracted from that directory, it generates two shell scripts one for FANTASM and one for MJOLNIR that are ready to be executed on the LINUX server. The configurations necessary to run FANTASM and MJOLNIR are included in the shell script.

INPUT: The script asks the user for directory of the files s/he wants to process.

OUTPUT: Two shell script (.sh) files, one for FANTASM and one for MJOLNIR saved in the same directory as writeShellScripts.m

FANTASM and MJOLNIR: (on the Linux server of Johns Hopkins University)

After all the necessary image files, the template mesh and the necessary scripts are transferred to the Linux server, the scripts can be executed. The results for FANTASM can be found under the folder flirtResults. The files that are being used by MJOLNIR are in the format [subject_name].dat.mem[1-3].vol which define the 3 memberships created by the fuzzy segmentation. MJOLNIR results can be found under the folder MjolnirResults. The files that the user needs for PCA are formatted as [subject_name]_TO_[template_name].vol.deformed_final.msh. These contain the vertices and faces of the mesh created for each subject.

All the MATLAB scripts previously mentioned can be run with the MATLAB on the LINUX server. In this case, the files don't have to be transferred back and forth which makes running the pipeline easier. However there is one drawback to this kind of approach. Since the user will be connecting to the server via an ssh client, the connection can time out. In that case, MATLAB also quits and the scripts that were running can be terminated without producing the results.

Statistical Analysis

INPUT: The warped volumes created by MJOLNIR

OUTPUT: The mean mesh of the atlas, modes, standard deviation

The detailed explanations of the methods the user is required to run are as follows:

atlas_results.m: (MATLAB script)

This function asks the user the names of the files they would like to analyze and the template image used for MJOLNIR. The function evaluates the mean mesh and computes the statistical variation.

INPUT: The warped volumes the user would like to analyze and the template image.

OUTPUT: Mean mesh and variation statistics

Helper methods: (MATLAB script)

FindMeanShapeAndAlignShapes.m:

This method aligns the shape instances by using active shape models and computes the mean shape of the aligned [1].

INPUT: shape instances and the number of iterations for the alignment phase

OUTPUT: The mean mesh calculated

ComputeInstanceStatistics.m:

Calculates the modes and standard deviation of the shape instances [1].

INPUT: shape instances

OUTPUT: modes and standard deviations

Surface Mesh Creation:

INPUT: The mean volumetric mesh and modes created by PCA

OUTPUT: The mean surface mesh and modes

This is the last phase of the atlas building process where the volumetric mean mesh is converted into a surface mesh in case for possible applications such as bone tunnel estimation after ACL surgeries. The code is incorporated to the atlas_results.m and saved as MeanSurfaceFaces. The resulting surface mesh can be visualized by using the list of vertices called MeanShape and MeanSurfaceFaces. The variable SurfaceModes contain the information about the modes that apply to the surface vertices.

Helper methods: (MATLAB functions)

tet2tri.m:

This is a function from the graphical toolbox by Gabriel Peyre [taken from http://www.mathworks.com/matlabcentral/fileexchange/5355-toolbox-graph/content/toolbox_graph/tet2tri.m]. It converts the volumetric mesh into a surface mesh. It only returns the faces of the surface mesh which can be visualized with the vertices of the volumetric mesh.

INPUT: The vertices and faces of the mean volumetric mesh.

OUTPUT: The faces of the surface mesh.