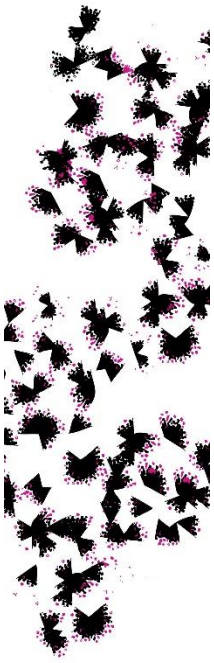

Segmenting knee bone structures from 3dimensional MRI data.

A.R. (Arthur) Vogelzang

BSc Report

Committee:
Prof.dr.ir. C. Slump
Assoc.prof.dr. S. Manohar
O. Schenk, MSc

August 2015

021RAM2015
Robotics and Mechatronics
B-BMT
University of Twente
P.O. Box 217
7500 AE Enschede
The Netherlands

Summary

The knee bone is a very important joint which supports the body when upright. In older people however wear and tear of the joint cause pain, assessing the damage is very difficult. For assessing damage MRI is a valuable tool, however interpreting data has to be done manually and is very time consuming. The Medical Imaging group from the university has developed an algorithm to interpret, segment, the image data from an MRI scan. The current model however is not yet accurate enough, the current assignment is to improve the accuracy of the bone model.
The algorithm is built using an Active Shape Model (ASM). The ASM requires datasets for input, these datasets all need to have points at approximately the same location on the surface, landmarks. For this the sets are pre-processed before building the model. Next to the pre-processed datasets it has several variables important for processing these datasets into a mean shape with the most common variations. This is done using principal component analysis. The resulting model is then used to segment new data sets. This part has its own variables needed to correctly apply the model to new data.
With this algorithm tests are run. The first test is a generalised cross validation using a homogenous set selected from the available training sets from MICCAI. All variables are kept constant while running this test. The second test is to optimise the variables in order to increase the accuracy. This is done with a less homogenous dataset and the variables are varied one by one to test their effect on the algorithm.
The generalised cross validation gives results with an average AvgD of 3,5 times higher than the difference between trained experts. Next to this the results have a standard deviation 2 times higher than the difference between trained experts. The results from optimising the variables shows that improvements can be made but that the algorithm is far from being accurate enough.
The algorithm works as it is now, it however has room for improvement. In order to increase the accuracy recommendations are made, including rebuilding large parts of the algorithm.

Table of Contents

Introduction

The knee joint is a complex joint in the human body and is subjected to stress when in upright positions. It consists of 3 bones and is fortified by ligaments. The joint consists of three parts, there is a joint between the patella and femur (femoropatellar joint) and there is a double joint between the femur and tibia (tibiofemoral joint). In this research the focus will be on the tibiofemoral joint. It is made up from two joints, a medial and a lateral joint (figure 1 (Kramer)) (Marieb, Hoehn, 2010). The tibial plateau is covered by the meniscus to give extra stability to the joint.

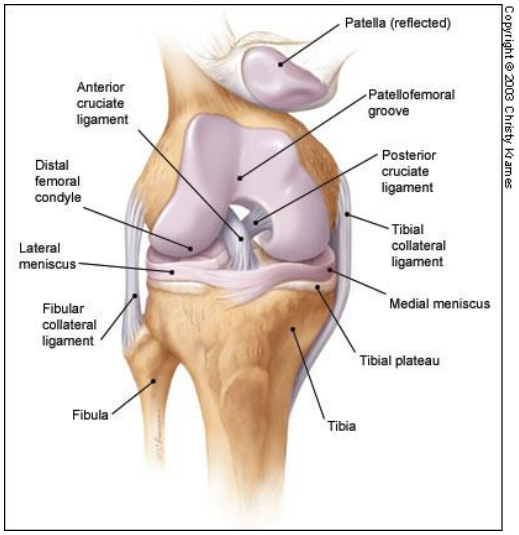

Figure 1, an image containing the main components of the knee (Kramer). In this project the aim is to find the edges of the tibial and femoral bone and cartilage.

On the surface of the tibial- and femoral- bone there is a layer of hyaline cartilage. Hyaline cartilage is made up mostly of water, collagen fibers and glycosaminoglycan's (GAG's). This extracellular matrix is produced mostly by chondroblasts. Hyaline cartilage has properties between those of dense connective tissue and bone. It is flexible, durable and it provides rigidity (Marieb, Hoehn, 2010).

Cartilage is an avascular structure which means that it only gets nutrients from diffusion through the extracellular matrix. This means that cartilage heals slower than other types of tissue. Aging cells from cartilage also lose the ability to divide (Marieb, Hoehn, 2010). For elders this results in thinner cartilage, mostly in the femoral cartilage plate (Hudelmaier, et al., 2001). As people get older the tensile strength also diminishes because of several biomechanical factors (Temple, et al., 2007). This may lead to pain development, dysfunction and possibly osteoarthritis (Temple, et al., 2007).

Assessing the degradation of the cartilage is difficult. Conventional radiographs have proven to be unreliable (Rogers, et al., 1990). Magnetic Resonance Imaging (MRI), however, is better at detecting morphologic damage to the cartilage (Li, et al., 2007).

The interpreting of this data is done manually. This process is very time consuming. Identifying the different tissues and their respective edges is called a segmentation. Completely segmenting an image can take an expert several hours (Roemer, Eckstein, & Guermazi, 2009). This has to be done by experienced physicians which makes it expensive. Because of this, algorithms were designed for the data segmentation. These algorithms still need input and correction from trained physicians to accurately segment the image data. (Stammberger, et al., 1999).

The next step in development after semi-automatic segmentation is building fully automated segmentations that work without additional input and corrections from physicians. An algorithm for this purpose was designed and developed at the Medical Imaging group of the University of Twente. The resulting algorithm was sent in to the MICCAI grand challenge (www.ski10.org). The results from the evaluation showed that the algorithm is not accurate enough in comparison with ground truth segmentations.

In order to improve the model it is important to know how the images are evaluated at MICCAI and what is important in the segmentation. Evaluating the bone models is done with the use of 2 different metrics. The average symmetric surface distance (AvgD) and the root-mean-square symmetric surface distance (RMSD) were used.

MICCAI scores the algorithms based on these two measures for the bone segmentations. The scoring is based on an independent second rater. The second rater scores 0.45 mm for the AvgD and 0.77 mm for the RMSD. This is done in comparison to reference data from the first rater. For the test sets from MICCAI this data is hidden. The scoring of the algorithm is based on these scores. For a perfect segmentation the AvgD and RMSD needs to be 0 compared to the hidden reference. If the AvgD and RMSD are equal to those of the second rater the segmentations scores 75 points. With an error twice as high as the second rater the segmentation scores 50 points and so on. Scores do not go below 0 points. This is done in order to not influence the total score too much with one set.

The research question for the bachelor assignment is, how can the algorithm for the knee bone segmentation be improved? This will be answered in several steps. First, how does the current algorithm work? Second, how accurate are the results of the current algorithm for the femur bone? Finally, can the accuracy of the current algorithm for the knee bone be improved?

Materials

For the materials the most important part is the algorithm previously made. This algorithm is explained in the paper written by the creators (Kroon, et al.). In the next section the workings of the algorithm will be explained. In this section the ISO-surface extraction of the bone is explained first, after this the training of an Active Shape Model is explained.
This is followed by a section about the variables that are used in the algorithm. The last part of the materials is about the metrics that are used in the algorithm and that are used in the testing.

The algorithm

The algorithm consists of two parts. The first part is building a model, the second part is using the model to segment the data from a new dataset. This is made visible in the image from the original paper, see figure 2 (Kroon, et al.).

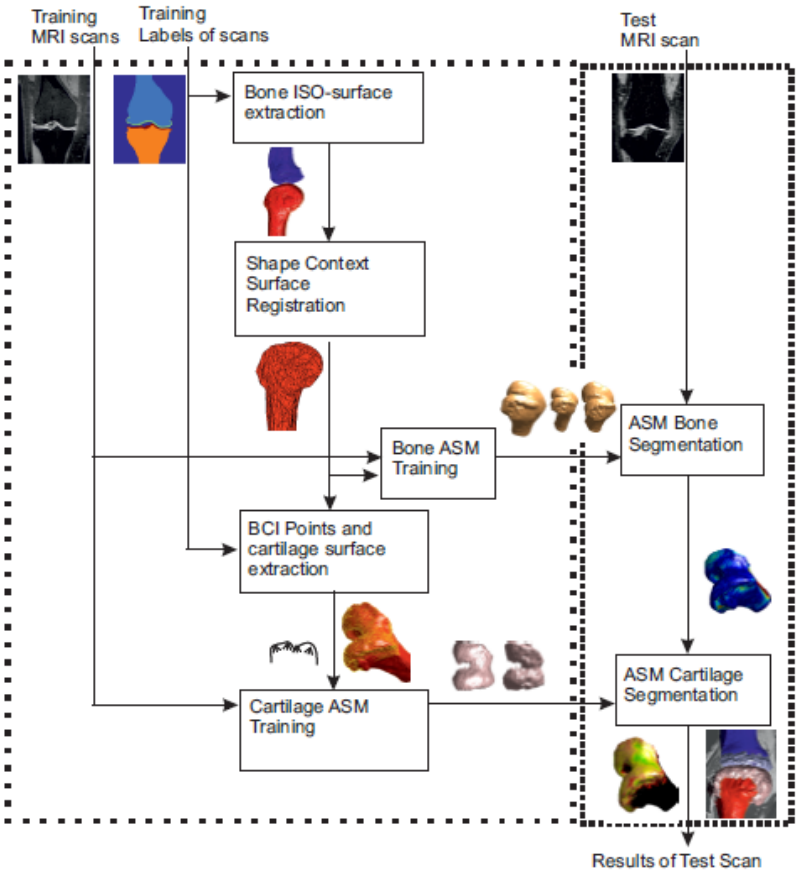

Figure 2, a schematic showing the workings of the algorithm (Kroon, et al.). From above left the label data is taken together with the original scans to build a model from the bone surface. The models built here are then used to apply to new data sets to find the surfaces in that data set.

ISO-surface extraction and shape context registration

The model is built using a selected number of datasets. The training datasets that are used were supplied by MICCAI. All training sets have the raw image data and label data included. The label data consists of voxels denoted by values from 0 to 4. 0 Stands for not relevant, 1 and 2 are femoral bone and cartilage respectively, 3 and 4 are tibial bone and cartilage. For the training of the bone active shape model the labels 1 and 3 are used. The first thing that needs to be done in order to train the model is to find the surface of the bone. This is done by ISO surface extraction. First the label data is smoothed using a Gaussian filter, after smoothing the data the ISO surface is taken for the relevant label needed for model training. The ISO surface is taken using the marching cubes algorithm (William E. Lorensen, 1987).

The surface is defined by vertices and faces, the faces are triangular surfaces defined by vertex numbers. The vertices from the surface are used to build the model. For the training of an active shape model their needs to be a set of landmarks corresponding across all datasets. These are points which are located at places that can be found in each training set. These are mostly defined on sharp edges and on corners. Every dataset however has different vertices after extracting the ISO surface. In order to get corresponding vertices in every data set the surface from one set, the master set, is non-rigidly registered to all the other sets. This gives the vertices from the master, placed as accurately as possible in the other training sets. The vertices are matched between datasets using a 3D Shape Context algorithm, made by Kroon (Kroon D. , 2012). This is an extension of the original Shape Context (Belongie, et al., 2002)(reference to shape context). Each point gets a description based on a coarse histogram which is uniformly distributed in a log - spherical space. In order to get the histogram space uniformly distributed the polar angle θ is multiplied by its cosine. This results in the following histogram coordinate system $(O_r, O_\theta, O_\varphi)$:

$$O_r = \frac{n_r}{\log(r_{max}) - \log(r_{min})} \log(\frac{r}{r_{min}})$$
$$O_\theta = n_\theta \frac{z}{r} \quad (1)$$
$$O_\varphi = \frac{n_\varphi}{2\pi} \arctan(\frac{y}{x})$$

n, is the number of histogram bins set per coordinate. r_{max} and r_{min} are the maximum and minimum distance respectively considered when determining the histogram. The histograms are called the feature vectors of the point from which it was taken. Matching is done using the feature vectors. The matching cost *C(p,q)* of points *p* and *q* with the feature vectors h_i and h_j, each having K bins, is given by the following formula:

$$\mathrm{C(p,q)} = \frac{1}{2}\sum_{k=1}^{K} \frac{(h_i(k) - h_j(k))^2}{h_i(k) + h_j(k)} \quad (2)$$

With this formula the cost for matching every point from one to another set is calculated. From the cost matrix the lowest value is sought to find a match for every point in *p*. Several points in *p* can be registered to one point in *q.* The set with the lowest number of vectors is chosen as the master. The points *p* from this master set are than matched to points *q* in a reference set. After the matching is done the master set is warped towards the other set using b-spline grids. This is done iteratively while scaling up from coarse to a fine grid in order to get robust and accurate registrations. After the transformations the master has the same shape as the reference set with the vectors from the master set. With this vectors are created which are the same across all training data sets. These vectors are used as the landmarks for the ASM.
In order to get the master as close as possible to the reference an iterative closest point registration is done after using the b-splines. A modified version is used which registers point to plane except that it only matches when the surface normals are approximately the same. The matching and registration is represented in figure 2 by the block with ‘shape context surface registration’.

Training an Active Shape Model

After matching and registering the next step is to build an Active Shape Model (ASM) (Cootes T. et al. 1995), also visible in figure 2. An ASM first has to be trained before it can be used to segment new, unknown data sets. The ASM is built up using a set number of training sets which all have the same landmarks. Getting corresponding points in every set was done before, now moving on to the next block from image 2, the bone ASM training. All intensity data is normalised to a grey scale image between 0 and 1. This is done by first sorting all intensity values in ascending order. The minimum value is set as 0,1* the length of the sorted string containing all intensities. The maximum is 0,9* the same length. The data image grey scale is then normalised by subtracting the minimum and dividing by the maximum.

For all vector sets translation is removed by first calculating the mean of the x-, y- and z- components. The mean is than subtracted from the x, y and z centring the data on 0. Next to the translation being

normalised the angle is also normalised. This is done by first calculating the tangent between the x and y for all points of both the rotating and the mean set. The mean set in this case being the first from all the training sets, all other sets have their rotation normalised to this set. The difference is than calculated between the means of the tangents and the difference is added to the angles of the rotating set. With the new angles new vectors are calculated giving the same set with a different rotation. These steps are repeated with the tangent between the y and z. All the transformations applied to the data set are also saved and later used to determine a mean translation and mean rotation transformation for the shape model

After the normalisation a principal component analysis (PCA) is done to determine the mean shape and the possible variations on this shape. The PCA is done using singular value decomposition (SVD). All input training data sets are used for this PCA. The vectors x,y,z are all put in one column vector for a single set. A matrix is made from all these column vectors, after this the PCA is run on this matrix. The resulting eigenvalues and eigenvectors give the mean shape and the variations for the ASM. The rotation and translation data is also stored with the sets and a mean translation is also stored. SVD first uses householder transformations to reduce the matrix to a bidiagonal form, after this givens rotations are used to reduce it to diagonal form. This gives the eigenvalues, the eigenvectors are determined by multiplying all left-multiplication transform matrices. The PCA is than completed by taking the square of the eigenvalues while the eigenvectors are equal to the left multiplication matrix from the SVD.

Next to the shape data there is also a need for appearance data. For the appearance data intensity profiles are taken along the normals of the faces. This is done for all training sets and the collected profiles are stored with the shape data. Next to the grey profile the first derivative of the profile is also stored.

With this data stored it can be applied to new datasets. When applying to new datasets first the raw image data is loaded. The first step is to normalise this to the same range as the other data. So the unknown is normalised to a grey scale image with values between 0 and 1. After the normalisation the mean shape from the model data is placed in the image, the shape is placed first around zero and after this it is rotated and translated by the mean transform from the model data. After the transform the registration process is started. This is an iterative process, first it searches for the best starting location. This is done by giving 7 more small random translations and checking which has the smallest regularization error. The regularization error is the surmised cost of the matching of the grey profiles between the new dataset and the profiles from the model data.

After choosing the best starting location the iterative registration is started. This is done from a coarse to a fine scale. For every image scale a number of iterations can be chosen. In this iterative process the grey profiles from the model data are used to find the best matching profile in the new data set. After scaling up and refining the searching is stopped after a set number of iterations. The resulting vector data is then translated back in voxel data and given as the resulting segmentation together with the resulting vector data and corresponding faces.

The variables used in the algorithm

Within the algorithm multiple variables are determined before running parts of the algorithm. In order to optimise the variables knowledge about the workings is essential.

The first part of the algorithm, the registering and matching of datasets has a number of different variables that can be changed. The first part, the iterative shape context algorithm, has five variables. These are, the number of log distance bins, the number of angle bins for the matching histogram, the degree of rotation normalization, the point to point matching method and the maximum matching distance.

Increasing the number of bins for the histogram makes for a finer characterisation of the points and a better matching between the points. The number of bins can be increased by increasing the number of log distance bins and by increasing the number of angle bins. Increasing the number of bins will also increase computation time.

The rotation normalization can be either none, minimal align rotation or heavyside flip in the algorithm. Minimal align rotation calculates the angles between the master and the registration set and then rotates the registration set towards the master set. The heavyside flip checks if there more points higher or lower than 0 for x,y and z. If there are more points lower than 0 it flips the axes multiplying all x,y or z with -1.

The matching method can either be ‘Munkres one to one’ (Munkres, 1957) or ‘minimum one to multiple’ matching. The set default is ‘minimum one to multiple matching’.

For the matching there is one last variable to set, this is the maximum matching distance. All points are first normalised by dividing by the mean of the length of all vectors. Hereby averaging vector length to one. After this the cost matrix is built with formula (2). The maximum matching distance is then used to set a maximum over which matches are made. First the difference between the points of the 2 sets is determined and then this is compared to the maximum matching distance. If it is higher than the set distance the cost value is set to ‘inf’. Hereby making it very unlikely to become a match. This concludes the variables used to match and register the data sets.

The next step in the algorithm is the training of the model. In this part of the process there are several variables which influence the final model. During this part of the algorithm some of the variables for applying the model are also defined. The length of the contour profile, the number of image resolution scales and how many of the PCA eigenvectors and eigenvalues are stored are all variables that are set during the training of a model. For the application of the model the variables that are defined are the length of the search profile for the contour profile, the normal contour for the search limit and the number of iterations used per image scale.

The most significant variable however is the number of datasets that are used to build the model. This can be determined in this part of the model as well. More data for training can increase robustness of the model because it includes more variance. The added variables however make it harder to apply to new datasets and will take more time to compute. Models using more data can become more accurate but need more time.

For the contour profiles two variables are set. One is used to set the length of the contour line. A smaller contour line when taken correctly is better for sharp edges. Longer profiles are however needed in order to completely store profiles from smoother edges. The profile should be chosen just long enough to store the smoother edges keeping the irrelevant data for sharp edges to a minimum.

A second variable is set for how far away from the current edge the data will be searched for a better match. This is the length of the search profile. This value determines the range within which the best match is sought for the contour profile. A higher value gives more search space and may give a better fit. It can however also lead to the algorithm finding a location other than the bone giving a better fit.

After the best location is found within the searching boundary this is compared with the normal contour. The normal contour is a set distance around the mean shape and the variations of the eigenvectors. The new vectors first have the mean of the shape data subtracted. After this they are left-multiplied with the transpose of the eigenvectors. This is than compared to the set variable multiplied with the square root of the eigenvalues. If it exceeds this value either in the positive or negative direction it is set back to the maximum value given by the variable multiplied with the square root of the eigenvectors.
This limit is invariant of the iterations keeping the model from deviating to far from the mean shape. Making this value to high will result in too much freedom with probability of the model collapsing, to little variation and it will not be possible to completely deform to the shape of a new set. The optimum should be somewhere between the mentioned extremes.

A second variable which constrains the possibility for model variation is determined as well. This variable is the percentage of the eigenvalues and eigenvectors which is kept. This is done by sorting the eigenvectors and eigenvalues in descending order of the eigenvalues. Then the variable determines what percentage is kept, keeping the vectors with the highest eigenvalues and deleting the other percentage. Keeping the ones with the highest eigenvalues gives the eigenvectors which correspond to the largest significance and change in the model. Because of this only a little bit of

accuracy is lost by deleting the smallest eigenvectors. Removing to many eigenvectors however will result in significant loss of possible variations and with that reducing accuracy in new segmentations.

This process is iterated in order to find the best approximation to the model data. This is done in 2 ways. The image can be scaled from coarse to fine in order to first approximate the location than scaling up and finding the exact location. For every resolution scale a number of iterations can be set as well. Starting from a lower resolution the image data is rescaled to a higher resolution adding more grey values. In the finer scale the search is continued in order to find a more accurate fit for the contour. The number of scales can be set, more scales means finer scales and are expected to give more accurate results because of the finer search.

Within each scale the number of iterations is set. With every iteration the algorithm further approximates the best fit for the contour. Setting the number of iterations gives more chance to find the exact edge of the bone. Next to this more iterations also take more time to run. The setting should be high enough to find the best location but not too high as to find a better fit somewhere else in the data while also keeping the computation time as low as possible.

The metrics used for the assessment of accuracy

Within the algorithm all the data that is gathered needs to be evaluated. Within the script this is mostly done using the Dice coefficient. The Dice coefficient varies between 0 and 1, giving 1 for complete overlap and 0 for no overlap. It is determined as 2 times the number of similarities between two sets divided by the total number of entries from both sets. In formula this can be seen as follows:

$$Dice\ coefficient = \frac{2\ |A \cap B|}{A + B}$$

With A and B being the datasets, in this case the label data from the segmentation and the label data from the original or from the previous step during the searching algorithm. Next to the dice coefficient 2 other metrics are calculated for the evaluation.

These metrics are the average symmetric surface distance (AvgD) and the Root mean square surface distance (RMSD). Both values use the shortest distance from one surface to another based of the label data. For the AvgD the average is taken from all distances based on the edge points of the first set. For the RMSD the same is done using the Root mean square instead of the average.

Method

The generalised cross validation

Knowing the principle workings of the algorithm it is important to test the performance. The testing was started with a generalised cross validation. This will give an indication as to how accurate the current algorithm is. However more important is that it shows how consistent the algorithm is in getting the wanted accuracy.

The generalised cross validation was run as follows. From the total 100 available training sets 22 were selected. These were selected on eye sight. All sets show a dark bone, the cartilage is relatively bright and the images are well defined, giving relatively sharp edges. This subset from all sets were selected to give a more homogenous training set. The more homogenous set should be easier to segment. A slice of one of the data sets is visible in figure 3. This slice is from data set 001. The other sets that are used for the generalised cross validation can be found in attachment 1.

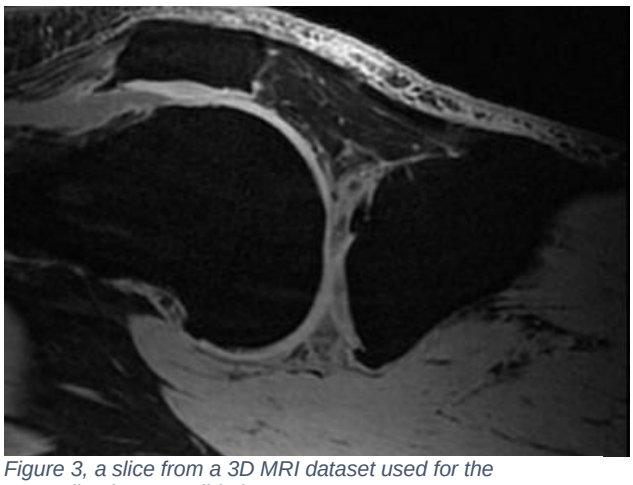

Figure 3, a slice from a 3D MRI dataset used for the generalised cross validation.

From the 22 sets 21 will be used to build a model. The one that was not used for building the model is then used as the test set and compared to its original label data. This is done with all the sets, building a model with the other sets and then testing the remaining set and comparing it to the experts label data. This can be done because all 22 sets are chosen from the training data from MICCAI which has label data available.

The other variables are all kept constant as to not influence the results. For this all values were kept at the default values gotten from the script as it was run for the segmentations sent in to MICCAI. These were as seen in table 1.

Table 1, the table contains the variables and their set values used in the generalised cross validation, the variables are not varied in this test.

The variables and their values for the generalised cross validation	
Variable	*Set value*
Length of contour profile	8
Search length for profile	10
Number of resolution scales	3
Number of search iterations	10 for every step
Length of normal contour	3
Percentage of eigenvectors	98 %
Number of training sets used	21

The data registered by the algorithm is then compared to the corresponding original label data. This was done using the same metrics as from the MICCAI grand challenge, the AvgD and the RMSD. The Dice coefficient was also added because this metric is used within the algorithm.

The optimising of the variables

After the generalised cross validation more tests are done in order to find the optimal settings for the variables. The goal is to find the setting for the variable which results in the lowest AvgD. Because of the available time the variables will only be varied one by one. Even though varying several variables together might be useful as well. These tests have two reasons. It is done to find the optimal values for the variables. The second reason is to test and see if all the variables respond in the way that is expected.

For the testing of the model the same output metrics are determined and used as with the generalised cross validation. This is because these metrics make for easy comparison with other steps of the model, between the different variables used in the script and because the AvgD and RMSD have a physical significance.

For the testing of the variables the following variables were included:

- Length of intensity profile
- Search length for intensity profile
- Number of image resolution scales
- Length of normal contour
- Number of search iterations
- Percentage of the eigenvalue/vectors kept from total.
- The number of training sets that are used to build the model.

These variables were included because they are used in the building and applying of the model. These variables all have a significant function within the model. Because of this these variables should also make visible if the algorithm works as intended. This is accompanied by the fact that within the algorithm these variables are all set at the beginning of the script showing that these were earlier used for the optimisation as well.

These variables are all varied between set numbers. The default values are the same as when the model was used to collect the data for the MICCAI grand challenge. A window is chosen around the default value in order to see if the default is at the optimum or if the default value should be changed. All below mentioned variables are varied piece by piece to study their effect on the model.

Table 2, in the table are the variables used in the optimisation of the model, for every variable the default value is shown and the range within the value is varied.

Variables, their set default and the range within which they are varied

Variable	Default	Range
Length of contour profile	8	1 - 20
Search length for profile	10	1 - 20
Number of resolution scales	3	1 - 5
Number of search iterations	10 per image scale	5 - 50 per image scale
Length of normal contour	3	1 - 5
Percentage of eigenvectors	98 %	90% - 99%
Number of training sets used	All variations were done with all model sizes	5, 10, 20, 30 and 50

For every change in a variable a new model is built to investigate how it influenced the model. If the changing variable is used only for the application of the model and does not influence the building of the model the built model from the first of that series of tests is reused for all iterations of that variable. This is done to save time, when moving to the next series of tests with a different variable new models are built again. For each set of parameters the first 5 training sets were used to apply the model on. At the end the resulting metrics were taken as an average from these 5 sets.

Results

As said in the method the models were run and the results were gathered. These results will be shown and interpreted in the results. This will be done separately for the generalised cross validation and the optimising of the parameters. After these tests an additional part is added to visualise the iterative process of the application of the model.

The generalised cross validation

The results from the generalised cross validation are displayed in table 3, on the next page. In the table the first column contains the numbers from the training set that was excluded for that model. The resulting numbers are from the comparison of the original label data and the registration from the algorithm for the given set. In the data there is variation, 3 sets however seem to deviate. These are set numbers 37, 52 and 69. The results from these 3 sets have errors 2 times higher than the average from all the sets. Examining these sets reveals that the bones are lighter than those of the other sets. This results in smoother edges with different profiles. A slice from image 52 can be found in figure 4.

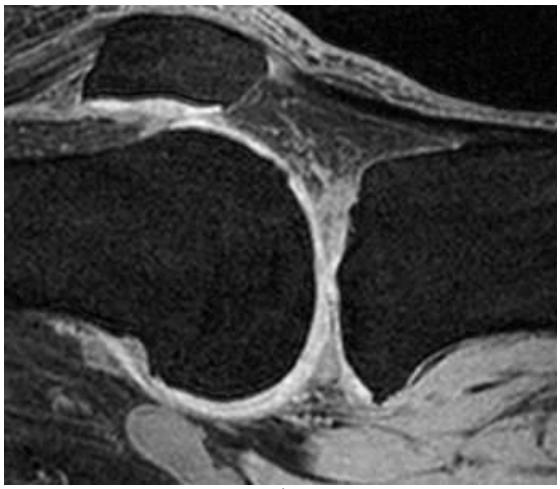

Figure 4, a slice from the 52nd training data set. In this picture the bone is greyer than the bone from the first set.

Because of the lighter bone the model cannot find a good matching profile and keeps searching. The normalisation misses on the background being very close to zero using the grey of the bone could give better, more robust results. When these three sets are excluded the standard deviation goes down. The Dice coefficient is close to or higher than 0,9, the AvgD and the RSVD give high values. The average of the AvgD is 3,5 times higher than that of the expert. For the RMSD it is 2,7 times higher than the trained expert. This is in comparison to the standards from MICCAI, the 3 outliers were excluded from the average. Next to the higher average the standard deviation is also quite high. The standard deviation is 2 times higher than the than the difference between trained experts.

Table 3, the resulting numbers from the generalized cross validation. The table shows the results for that set being used as the test set and the others being the training data.

The resulting metrics from the generalised cross validation.				
Set Number	Dice coefficient	Average symmetric surface distance (AvgD,[mm])	Root mean square symmetric surface distance (RMSD,[mm])	
1	0,898	1,899	2,349	
7	0,895	1,857	2,368	
13	0,938	1,222	1,498	
21	0,923	1,543	2,086	
22	0,894	2,179	2,898	
23	0,944	1,274	1,662	
26	0,838	2,593	3,255	
37	0,659	5,694	6,722	
38	0,920	1,719	2,212	
46	0,927	1,593	2,190	
48	0,938	1,131	1,339	
51	0,937	1,314	1,669	
52	0,753	4,316	5,347	
56	0,942	1,438	2,092	
58	0,904	2,193	2,897	
62	0,940	1,154	1,516	
66	0,940	1,677	2,473	
69	0,725	4,729	5,910	
81	0,914	1,619	2,047	
82	0,935	1,380	1,672	
85	0,960	1,424	2,483	
92	0,952	1,064	1,347	
	0,894	2,046	2,638	Average
	0,080	1,245	1,472	STD
	0,052	0,806	1,005	STD without outliers (37, 52, 69)

The next step was optimising the variables from the model to see if the accuracy could be improved. By applying this method much data was generated. All this data was thereafter compiled in several tables showing optimum values for the variable in combination with the defaults of the other variables.

The optimising of the variables

A number of variables were tested. Some showed possible relations between input and the resulting accuracy, with other variables there seemed no apparent relation. One of these variables is the search profile. The search profile length is not further examined. The tests average AvgD was 0,5 mm higher than the average of all tests and although the values fluctuate they show now relation. This is probably related to the landmark profile not being accurate enough. This should be optimised first before optimising the landmark profile.
Next to the search profile length the number of iterations was also excluded. This was done because of an error found in the script for applying, see attachment 3. While the iterations were set up front, before starting the process, this proved to have no effect in the algorithm. As visible in attachment 3 the variable `nsearch` is set back to `[1,1,1]`, because of this only 1 iteration is done every image scale. The resulting values therefore do not show anything. This is visible in the standard deviation. Within the model using the same number of sets the maximum for the standard deviation is 0,053 mm and an average of 1,454 mm.
The normal contour also showed vary little variation. The standard deviation is very low with the highest being 0,031 mm and an average AvgD of 2,428 mm. Having this little effect indicates a normal contour too large to effect the searching.
Other variables however showed some relations. The first relation was visible in the number of training sets that were used to build the model. When more training sets were used to build the model accuracy went down. The accuracy for the models using 5 datasets for training gave results for the AvgD which averaged 1,65 mm. This is about 3,5 times higher than the difference between experts but within these tests it is the lowest average. For the models using 10 sets for training the average was 1,94 mm. The average values increase further for models which use more datasets for training. The average AvgD for all tests per model size can be found in table 4. The values for the length of the search profile were excluded when making these averages. This was done because the values on average were 0.5 mm higher when compared to the average of all test.

Table 4, the average values for the AvgD taken over all data from the optimising of the variables. The values for the search profile length were excluded.

The number of sets used to build the model versus the average AvgD and RMSD over all the tests.					
# of sets used to build model	5	10	20	30	50
Average over tests for AvgD [mm]	1,658	1,956	2,407	2,529	2,709
Average over tests for RMSD [mm]	2,081	2,443	3,073	3,219	3,450

The second relation that was found is between the landmark intensity profile and the resulting AvgD. The optimum value caught within the range shifts toward longer intensity profile indicating that longer search profiles work better with more model sets in training. The values for the 5 set model are very close to each other not showing a relation. This can be because not all search profiles have the edge at the exact same location. This is because of possible registration errors and because of differences between experts putting the label data a little bit off the edge of the bone. This leads to the resulting point being placed just off the edge as well. The longer profiles include the edge making the search for the location close to the edge instead of another place. The relation is made visible in table 5. The same relation was visible with the RMSD.

Table 5, Average symmetric surface distance [mm], the length of the profile set on the rows versus the number of training sets used in the model set on the columns. The colouring gradient was done per column.

The resulting AvgD [mm] from the tests with the length of the intensity profile.					
Average symmertric surface distance (AvgD, [mm])	5 sets model	10 sets model	20 sets model	30 sets model	50 sets model
length of intensity profile	Colouring per column				
1	1,501	1,706	1,974	2,169	2,222
2	1,224	1,700	2,053	2,066	2,125
3	1,278	1,805	2,176	2,217	2,425
4	1,216	1,694	2,363	2,351	2,444
5	1,294	1,790	2,597	2,605	2,673
6	1,412	2,005	2,753	2,700	2,853
7	1,401	2,057	2,714	2,867	3,014
8	1,368	2,104	2,897	3,078	3,206
9	1,300	2,078	3,020	3,247	3,717
10	1,219	1,769	2,906	3,374	3,788
11	1,225	1,799	2,785	3,206	3,908
12	1,346	1,807	2,686	3,266	3,815
13	1,307	1,870	2,736	3,108	3,566
14	1,324	1,710	2,573	2,867	3,369
15	1,386	1,697	2,285	2,633	3,124
16	1,373	1,585	2,119	2,315	2,666
17	1,420	1,549	1,986	2,054	2,379
18	1,445	1,494	1,860	1,898	2,209
19	1,435	1,498	1,671	1,830	2,065
20	1,446	1,604	1,586	1,722	1,966

The next parameter that was tested in order to find an optimum was the number of image scales used when applying the model. This works well for 2 scales but after more than 2 scales the accuracy goes down rapidly. This is made visible in table 6.

Table 6, Average symmetric surface distance [mm], the number of image scales set versus the number of training sets used for training the model. Colouring was done per column and goes from green to red, for higher values.

The resulting AvgD [mm] from the tests with the number of image scales					
Average symmertric surface distance (AvgD, [mm])	5 sets model	10 sets model	20 sets model	30 sets model	50 sets model
Number of image scales	Colouring per column				
1	1,305	2,145	2,415	2,808	2,903
2	1,201	1,338	1,923	2,343	2,594
3	1,334	1,938	2,980	3,077	3,395
4	1,675	2,734	3,943	5,528	5,689
5	6,804	11,861	16,788	17,811	21,174

This can be explained by the landmark intensity profile being defined by a number of pixels instead of a distance. This means that if the scale becomes smaller the landmark intensity profile becomes smaller as well. If the model is not in the right vicinity it will never find the right location on a higher scale. Adding on to this are 2 errors found in the algorithm. The first being that the scales are implemented incorrectly. Scaling goes from a fine to a coarse scale, because of this it starts with the smallest steps and after that starts taking bigger steps. The number of iterations being 1 per scale

also affects the results. Because of these errors this test also does not give a good representation of the variable.

The lowest value from this table is 1,201 mm. This is the lowest AvgD found from the entire test. The default value is set to 3 image scales and 2 consistently gives better results so for now setting the number of scales to 2 should be considered. Repairing the errors in the script will be good for the accuracy using multiple scales.

Another important parameter in searching for the correct location is the boundary of the search area. This is limited by the percentage of the PCA that is kept and also by the size of the normal contour. The normal contour was mentioned earlier not giving any significant results. The results from the tests with the percentage of the PCA are displayed in table 7.

Table 7, the percentage of eigenvalues and eigenvectors kept from the total training dataset, set versus the number of sets used for training the model. Colouring was done per column .The models with bigger sets were excluded because the values overall were very high and didn't give a good indication.

The resulting AvgD [mm] from the tests with percentage of the eigenvalues that were kept.			
Average symmertric surface distance (AvgD, [mm])	5 sets model	10 sets model	20 sets model
percentage of eigenvalues used	Colouring per column		
0,90	1,324	1,728	2,568
0,91	1,320	1,787	2,854
0,92	1,303	1,891	2,609
0,93	1,291	1,891	2,723
0,94	1,420	1,908	2,581
0,95	1,339	1,761	2,634
0,96	1,345	1,880	2,767
0,97	1,289	2,064	2,743
0,98	1,411	2,038	2,911
0,99	1,309	1,883	2,862

The second way that the search area is limited is by the percentage of the eigenvectors and values from the PCA. The values were varied between 90 and 99 percent. The highest standard deviation is 0,086 mm with the model using 20 training sets. The average AmgD is 3,110 mm. Variations show no relation except for staying stable while the percentage kept from the PCA is lowered. A lower percentage gives for an easier model to implement because it has les input variables. Further reducing the percentage is a good next test to see how low the percentage can go while retaining the same accuracy.

With this al relations notable within the data have been named. All values have a high AvgD and the standard deviations overall are pretty low. The low standard deviation means it does score steady values. If the segmentation scores can be decreased these values should be steady as well. With the algorithm that was tested now, the lowest score was 1,201. The value from the second rater was 0,45 mm for femur registrations. This means that the best result from the tests receives a score of 33 points and all other scores are lower. From all tests 78,8% scores 0 points. This is in accordance with the rating of the MICCAI grand challenge.

Visualising the process

Because of the greatly varying results from the optimisation of the parameters further testing is necessary. For additional research the iterative process run when applying the model to new sets was made visible. Before running segmentations the error with the number of iterations was repaired by removing the code line that set nsearch to 1 per image scale, see attachment 3. Visualising the iterative process was done on datasets from which segmentations from experts were available as well. By selecting 3 different segmentations with very different values for the AvgD the workings of the algorithm should become better visible.
The 3 segmentations are chosen with a varying number of scales and number of training sets used for training the model. All other variables are set at their default values. The number of scales used are 2, 3 and 5. These are respectively combined with 5, 20 and 50 training data sets. These were chosen based on the results from the optimising of the variables. One was chosen with a low AvgD, a good result, the second and third were chosen with a little higher AvgD and a very high AvgD. With more image scales the AvgD is higher. This is based of values from before repairing the above mentioned error. A better indication was not available.

Figure 5 shows the results from the segmentation with 2 data scales and 5 training sets. The shown segmentation at the last iteration has an AvgD of 0,519 mm, this is 0,7 mm more accurate compared to the lowest AvgD value from the optimising of the variables. After the first initialisation it is already visible that the model shape gotten from the label data is aligned really well. This is because it is one of the five building sets for the model used for segmentation. Overall the changes are small. The AvgD at the initialisation was 0,880 mm, the changes however do decrease the AvgD. Small changes at the shaft and head can be seen, both become a little bit larger.

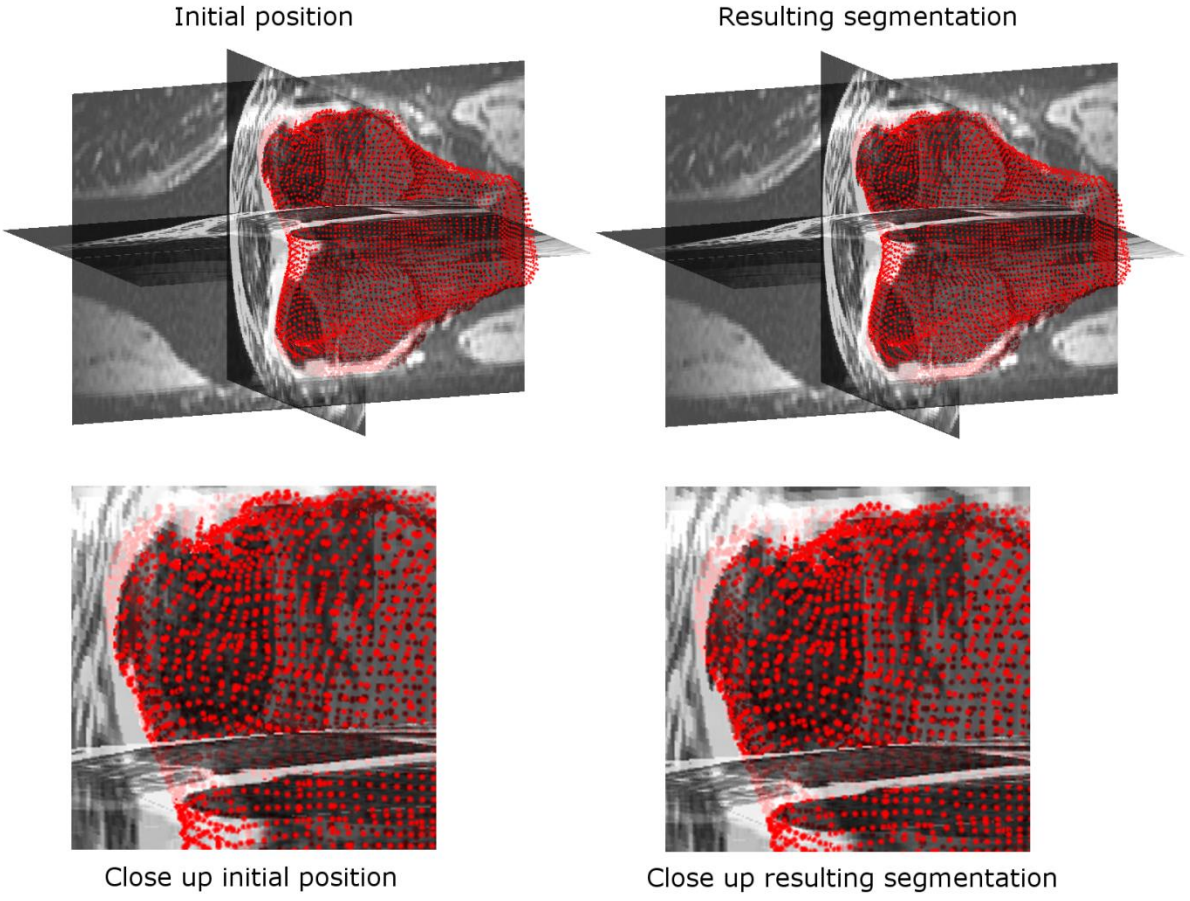

Figure 5, this image shows the initial and final position of the segmentation using 5 sets, with a total of 20 iterations over 2 image scales. A close up is included of an interesting region. The resulting AvgD is 0,519 mm.

More interesting however is looking at an iterative process from an image which has a higher resulting AvgD. In figure 6 images are visible from a second iteration process. This process used 3 image scales and the model was based on 20 data sets. When watching the segmentation step by step small differences are visible. The bone shaft searches inward creating an error. Next to this the segmentation widens around the head. This is visible in the lower edge of the image as well as in the top part of the image. Both changes increase the accuracy of the segmentation. The cartilage surface of the bone remains mostly the same but fills out the little edge of grey visible at the initialisation. The AvgD is 2,497 mm at the initialisation and is 1,674 after the segmentation. This shows that in principle the model does work. The image however displays one of the errors as well. The first 10 iterations give almost no change, the 10 after have more effect and the last 10 show the biggest steps. This relates very well to the image scales being reversely implemented in the scripts. Repairing this error in the script will improve the accuracy.

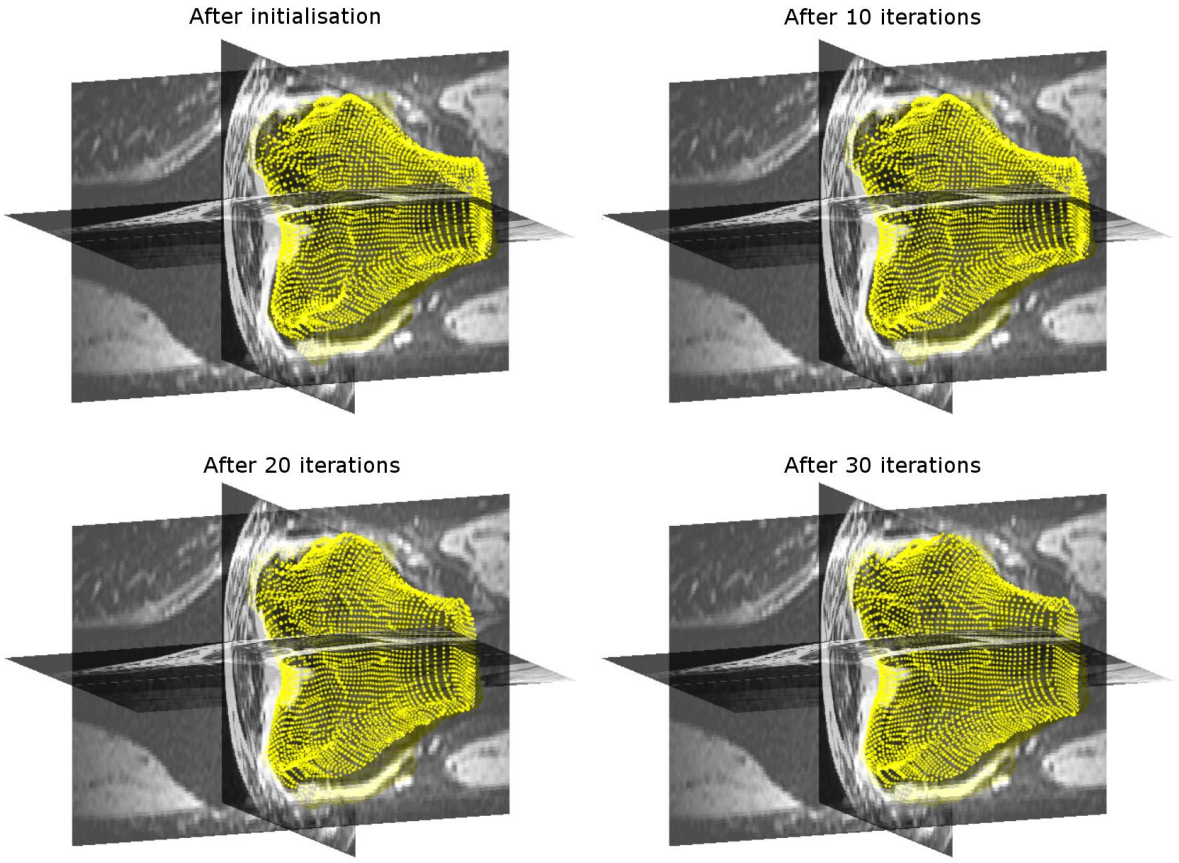

Figure 6, in the image the segmentation is made visible from a model using 20 sets, the model does 30 iterations on 3 different image scales. The resulting AvgD is 1,674 mm

In figure 7 a segmentation is made visible with a resulting AvgD of 23,129 mm, the highest from all the segmentations done for the optimising of the variables. This segmentation was done with 5 data scales and the model was based on 50 training sets. At the initialising step it is visible that something is not right with the initialisation. The initial step is much larger than the femur in the image data. This can be because of the model or it can be an error in initialising. This however is strange seeing that it does not happen with the other data sets used for the visualising.
Because of the bad initialisation all points get really bad matches on the shape profile which are small because of the small scale, starting with the finest and going to rougher scales because of the error. Because of this no point can find the right location and every point is set inward. When the scale becomes larger it unfolds a little but because of the bad location it cannot match the shape profiles. This results in the AvgD of 23,129 mm while it initialised at 7,392 mm.

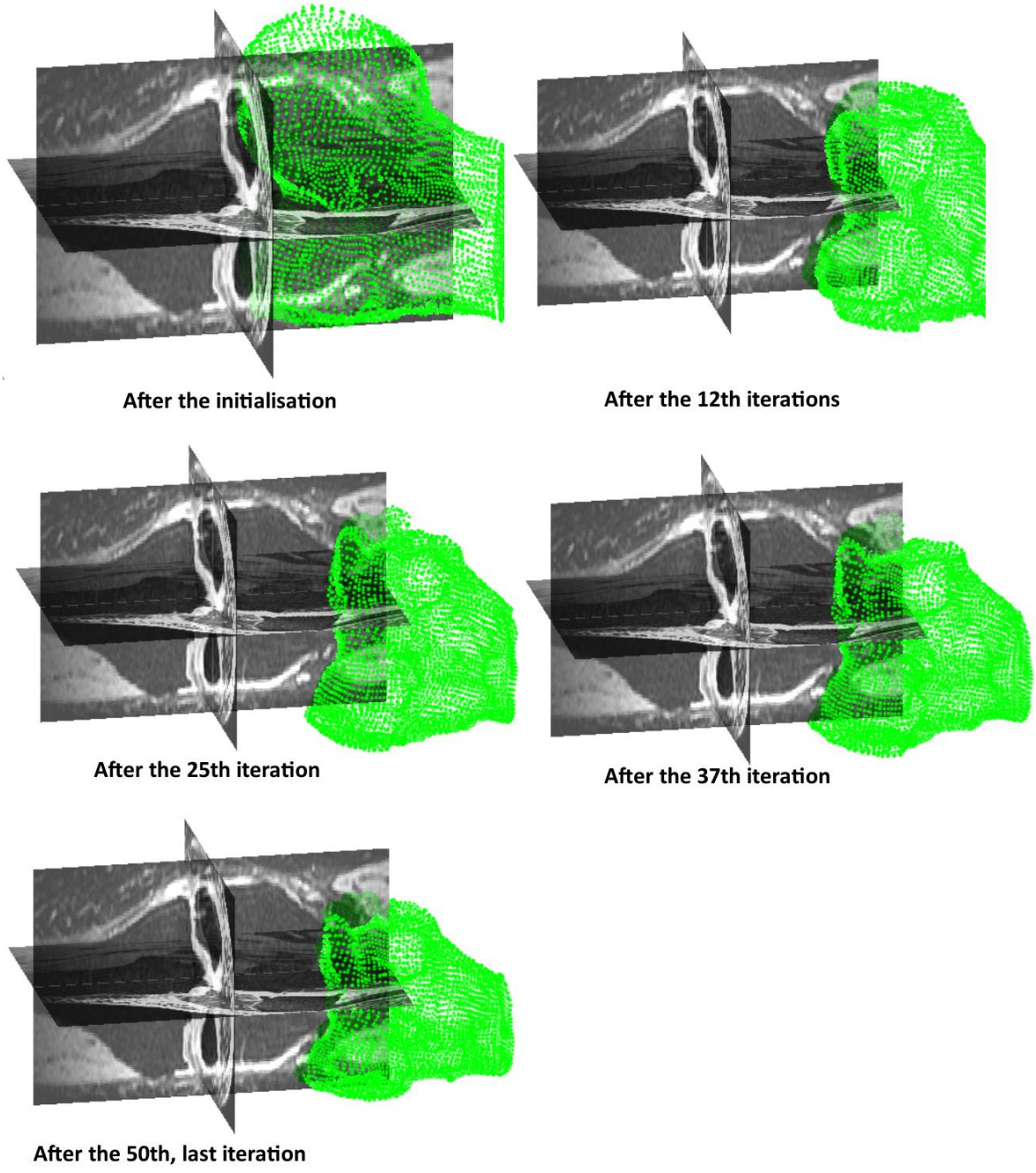

Figure 7, showing several steps from the iterative segmentation of the femur cartilage. This registration was done with 5 image scales and the model was based on 50 training sets. The AvgD of the final segmentation is 23.129 mm.

Discussion and recommendations

The first part will discuss the results found above. This also includes recommendations made on the basis of these results. After this other recommendations will be made based on work done with the model and experience with the algorithm.

The searching of the algorithm still leaves room for improvement. The profiles do not always find the right locations. Improving could be done in two ways, or maybe both combined. The normalisation now uses the background for the lowest intensity values, maybe the bones intensity can be used instead. This should give a better contrast between the bone and surrounding tissue. The second improvement is named in the original paper (Cootes T. , Taylor, Cooper, & Graham, 1995). Using the derivative of the grey profiles results in getting better results for them. The same can probably be said for our algorithm.

The algorithm is a complex process that takes multiple scripts to run. Every step is built on the previous step. Because of this accuracy in the earlier steps is crucial. The resulting sets after surface extraction, matching and registering however had a resulting AvgD of 5,7 mm when compared to the original labels. This error effects all following steps. Improving this will improve overall accuracy and helps take the contour profiles at the correct location.

Increasing the accuracy on this part can be done as follows. A combination of several options is likely to have the best result. The first thing that can be changed is the chosen landmarks. In the current algorithm the dataset with the least landmarks on the image surface is used as the master set. Changing this can help greatly. Building a custom vector set gives the possibility to increase overall density. It can also give greater density to important locations like the bone surface and reduced density in less significant locations like the bone shaft. The increased density will give it a larger eigenvalue in the PCA thereby giving it more importance. This custom set can possibly be implemented when taking the ISO surface, this would completely eliminate the need for the matching and registering.

Some parameters are dependent on other parameters, testing these while varying both parameters instead of one at a time will help show relations and optimise the variables. This can be done with the landmark intensity profile and the search length for the landmark profiles.

Extending the search parameters and increasing the step size for testing is also something to look at. Some parameters do not give any or give very little correlation. This can also be because the current tests did not have the optimal range included. Increasing the search area can better help understand the algorithm and show relations that were not visible before. From these tests the best settings can hopefully be found for different variables and models with more and less data sets.

One of the bigger problems with the model is that multiple people have worked on it at different times. Because of this the folders became unclear and running the algorithm became an assignment on itself. In order to counteract this all necessary scripts were copied in to new folders. All scripts are saved in the old folders for later reference but the new folders are enough to run the scripts. Next to this another document was built to accompany the newly organised folders. This was done to make it easier for a new person to find their way with the folders and to know where to find needed scripts and what to run in order for the next parts to run. This document can be found in attachment 1. In attachment 4 the folder structure is made visible as well.
In order to further facilitate the quicker learning of the scripts comments were added in the calling scripts to help understand how they work.

This makes it a lot easier for new people to start work on this algorithm. There is still much to be done on this algorithm. One thing that should be looked in too is the time it takes for the different steps of the algorithm to run. Some steps take several hours of time because of large multidimensional matrices needed in the computing. Clearing large variables when no longer needed helped but better data management is still needed.

One of the parts that takes a long time to compute is the feature histograms made for the registration process. The entire process for registering and matching can take up to 7 hours per data set. Decreasing the number of bins will decrease computation time but will also decrease accuracy, how much is unknown.
Initially my own laptop was not sufficient to run the algorithm, clearing the large variables which were no longer needed helped. This was mostly a problem when training models with 50 data sets. Models with more data can become more robust but these do take more time so choosing the percentage of eigenvectors to keep is very crucial in this context.

When processing the results these were also checked against the expected influence of the variables. Some variables however did not seem to give any difference when being varied and others showed no relation or a different relation than expected. The relation gotten from the intensity profile is working and the image scales are working as well. The variables that did not react as expected should be checked again after repairing the known errors.

Within the algorithm it was also quite hard to track the variables, because of the scripts all calling other scripts. Keeping the names more constant would make it easier to follow, in addition it helps keep the workspace clean and does not cloud computing space.

After the variables are verified and other improvements are made it is important to optimise the variables. The results were different when varying the different variables so optimisation can increase the accuracy of the model. Models based on a smaller or larger training set will have different optimisations.

Conclusion

In conclusion the model has pro's and con's. A lot of work has been put into this algorithm and it is working. The problem is that it gives results which are not yet accurate enough, but with sufficient work and implementation of the abovementioned steps the accuracy can certainly be improved. If it will be enough cannot be said for sure at this stage.

Reworking and adapting this algorithm will however take time and effort. It will have to be done by someone with sufficient knowledge and time to understand every piece of the algorithm. Even then finding the weak spots and fixing them will still take time and testing.

In the same time somebody can probably rebuild large parts of the algorithm, eliminating weak spots and making it easier to implement some of the above mentioned improvements. This algorithm can also be easier to understand if it is written and completed by the same person and the strong parts of the current algorithm can be taken and implemented in the rebuild algorithm. This is preferable above the other option.

References

Belongie, S., Malik, M., & Puzicha, J. (2002, April). Shape Matching and Object Recognition Using Shape Contexts. *IEEE Transactions on pattern analysis and machine intelligence, 24*(24), 509-522.

Cootes, T., Taylor, C., Cooper, D., & Graham, J. (1995, January). Active Shape Models - Their Training and Application. *Computer Vision and Image Understanding, 61*(1), 38-59.

Hudelmaier, M., Glaser, C., Hohe, J., Englmeier, K.-H., Reiser, M., Putz, R., & Eckstein, F. (2001). Age-related changes in the morphology and deformational behavior of knee joint cartilage. *Arthritis and Rheumatism, 44*(11), 2556-2561.

Kramer, C. (n.d.). Anterior view of the osseous, ligamentous and fibrocartilaginous structures of the knee. *American family physician.* American Academy of Family Physicians, Leawood.

Kroon, D. (2012, January 26). *Matlab Central: Active Shape Model (ASM) and Active Appearance Model (AAM)*. Retrieved from Mathworks: http://nl.mathworks.com/matlabcentral/fileexchange/26706-active-shape-model--asm--and-active-appearance-model--aam-?s_tid=srchtitle

Kroon, D., Kowalski, P., Tekieli, W., Reeuwijk, E., Saris, D., Slump, C., . . . Novak, C. (n.d.). MRI based knee cartilage assessment.

Li, X., Benjamin, C., Link, T., Castillo, D.-D., Blumenkrantz, B., Lozano, J., . . . Majumdar, S. (2007). In vivo T1p and T2 mapping of articular cartilage in osteoarthritis of the knee using 3 T MRI. *Osteoarthritis and Cartilage, 15*(7), 789-797.

Marieb, E. N., & Hoehn, K. (2010). Human Anatomy & Physiology. San Francisco: Pearson Benjamin Cummings.

Munkres, J. (1957). Algorithms for the Assignment and Transportation Problems. *Journal of the Society for Industrial and Applied Mathematics, 5*(1), 32-38.

Roemer, F., Eckstein, F., & Guermazi, A. (2009). Magnetic Resonance Imaging-Based Semiquantitative and Quantitative Assessment in Osteoarthritis. *Rheumatic Disease Clinics of North America, 35*(3), 521-555.

Rogers, J., Watt, I., & Dieppe, P. (1990). Comparison of visual and radiographic detection of bony changes at the knee joint. *British medical journal, 300*(6721), 367-368.

Stammberger, T., Eckstein, F., Michaelis, M., Englmeier, K., & Reiser, M. (1999). Interobserver reproducibility of quantitative cartilage measurements: comparison of b-spline snakes and manual segmentation. *Magnetic Resonance Imaging, 17*(7), 1033-1042.

Temple, M., Bae, W., Chen, M., Lotz, M., Amiel, D., RD, C., & Sah, R. (2007). Age- and site-associated biomechanical weakening of human articular cartilage of the femoral condyle. *Osteoarthritis and cartilage, 15*(9), 1042-1052.

William E. Lorensen, H. E. (1987, July). Marching Cubes: A high resolution 3D surface construction algorithm. *Computer Graphics, 21*(4), 163-169.

Attachments

Attachment 1, the sets used for the generalised cross validation

Sets used for the generalised cross validation	
001	051
007	052
013	056
021	058
022	062
023	066
026	069
037	081
038	082
046	085
048	092

Table 1, the numbers from the datasets used for the generalised cross validation.

Attachment 2, manual for the model

This document is made in order to help navigate through the folders needed for the segmentation algorithm. In order to operate the script you will need to have a version of matlab installed on your computer and when working on a Windows computer it is useful to install Microsoft Visual Studio. For some functions there is a script in matlab (.m) and a second script in visual studio (.c). The later one is several times faster so if possible, use the .c files.

Everything you need to operate the script can be found in the folder 'Vogelzang'. Be careful with changing folder names and folder locations. The scripts require additional functions which can be found in the sub folders of the specific parts of the algorithm. In the first layer of folders past 'Vogelzang' there is an 'addfunctionpaths.m'. Matlab uses this function to add the sub folders to its directory, this is needed for the main script to work.

Each step of the process has its own folder. Always start with making the main folder for the process step the current directory. All the parts have their own main script. These scripts are the backbone. First run the 'addfunctionpaths' after this run the mainscript in the folder. This script will call to every other necessary script and generate the needed data for the next step.

If you change map names or locations don't forget to edit the needed functions so the scripts keep running. Folders need to be changed in the 'addfunctionpaths' function and in the main scripts.

Matching

The scripts starts with matching points from the different datasets to the points of the master set. The script for this part can be found in the folder "Matching". My results for this matching algorithm can be found in the folder called "RegisteredAndMatchedLabels". These results are needed for the ASM. I suggest looking at the ones I made to see if these are sufficient because this step takes a lot of time and needs a lot of processing power for it to work. The ones I built can be found in \Vogelzang\Results\RegisteredAndMatchedLabels

If there is need for redoing this process the main script is 'Example3D2'. Open this script, make sure all variables are set correctly and that the parameters for the for-loop isn't too big because each image takes a lot of time. Running the script in several part takes a little more time but is safer.

Bone Active Shape Model (ASM)

This part can be found in the 'BoneASM' folder. The folder contains the scripts 'ASM_3D_Train' and 'ASM_3D_Apply'. The first one is used for building the models. Don't forget to set al the needed variables and options. All the variables can be found at the top of the script. Most of them are put in comment because I made another script for varying and testing the variables. The script ASM_3D_Apply is for applying the model on other unknown datasets.

Next to the 2 mentioned scripts there is also a script called 'OperationParameterSweep'. This script I made optimising the variables of the model. The script sets the most important variables and varies them one by one while building and testing all the models on 5 datasets. This way the effect of the variables can be made visible. The script builds models based on 5, 10, 20, 30 and 50 datasets and runs 70 cycles with the changing variables. Although time consuming (my laptop took about 5 days of constant crunching on the background), it gives a lot of data ready for analysing. It is also quite easy to reprogram this function to include other variables or to test only a part of the variables. This script only makes calls to the different model scripts, so changing things will not change the model.

Cartilage Active Shape Model

For this part everything can be found in the folder CartilageASM. The scripts to use are called ‘dj_ASM_cartilage_train’ and ‘dj_ASM_cartilage_apply’. Train builds the model and apply runs it for unknown datasets.

Functions

Next to the folders with the different parts of the algorithm there are a few additional folders. One of these is called ‘Functions’. This folder contains a lot of the essential subscripts that are used by more than one part of the model. All the ‘addfunctions’ script add this folder as well.

Data

There is a lot of data included in the folder. They are divided over multiple folders. The folder ‘knee_grand_challenge’ contains all the original datasets and the results from the first time that the data was send in to be evaluated for the miccai grand challenge.

The folder ‘Results’ contains more folders and has almost all the results I generated during my assignment. It has the registered labels from the first step of the process. The results of all my testing from the BoneASM and the statistics calculated with the tests.

Additional reading

All scripts are quite complex and variables are redefined, understanding the principals of the method is very important in understanding the script. I strongly recommend studying this article before starting work on the model:

Cootes, T., Taylor, C., Cooper, D., & Graham, J. (1995, January). Active Shape Models - Their Training and Application. *Computer Vision and Image Understanding, 61*(1), 38-59.

Attachment 3, possible errors found in scripts

Scaling was not correctly implemented, scales were searched from fine to coarse in the search algorithm for the knee data:

This is the part of script that seems to be incorrect. It was found in ASM_ApplyModel3D.m, seen below are rules 30 to 32.

```
for itt_res=options.nscales:-1:1
    % Scaling of the image
    scale=1 / (2^(itt_res-1));
```

`itt_res` goes from a high to a lower number. Because of this the scale goes from very small to 1, this should be the other way around. You start on 1 and work your way down to the smaller scales.

This error has been repaired, after testing was done

Search iterations are set back to 1 after initialisation:

In the script AMS_3D_Apply.m the script sets a variable back to 1 while this was earlier defined to vary for the results. This is line 35 from the above mentioned scripts:

```
options.nsearch=[1 1 1];
```

Rule 35 sets `options.nsearch`. This variable is earlier set to another value to determine the number of iterations made set per image scale.

When determining the histogram features of the points. Every point found out of limits is set in to the limits:

This error has been repaired, after testing was done

This is found in the code of the function getHistrogramFeatures, lines 66 to 71:

```
O=bsxfun(@minus,Pointsl,P);
            R=sqrt(sum(O.^2,2));
            A = (atan2(O(:,1),O(:,2))+pi)/(2*pi);
            R(R<options.r_min)=options.r_min;
            W = ((O(:,3)./R)+1)/2;
            Rlog = log(R);
```

`P` is the master set, `Points1` the set to register to. The 4th lines uses a Boolean to search for every point where the radius value is below the minimum and then replaces it with the minimal value. Points out of range however should be removed for accurate histograms instead of moved to be included.

The PCA is not done correctly for the building of the shape model:

In the code the following lines are found in ASM_MakeShapeModel3D.m, these are lines 21 till 30

```
for i=1:length(TrainingData)
       x(:,i)=[TrainingData(i).CVertices(:,1);TrainingData(i).CVertices(:,2);TrainingData(i).
       CVertices(:,3)];
       tform.offsetV = tform.offsetV*(i-1)/i + TrainingData(i).tform.offsetV/i;
       tform.offsetrxy = tform.offsetrxy*(i-1)/i + TrainingData(i).tform.offsetrxy/i;
       tform.offsetryz = tform.offsetryz*(i-1)/i + TrainingData(i).tform.offsetryz/i;
end



[Evalues, Evectors, x_mean]=PCA(x);
```

In the for-loop all datasets are taken from the `TrainingData` and are put into a single matrix. First every `x,y,z` is put into a single column, first the `x`'s than the `y`'s, ended with the `z` components. This

is done separately for every dataset. All column vectors are collected to build the input matrix for the PCA.

This however differs from the process given in the original paper about Active Shape Models (Cootes T. , Taylor, Cooper, & Graham, 1995). In this paper the vectors are put in a column vector as well. After this the covariance within the set is calculated. This can be done by right-multiplying the column vector with its transpose. This is done for all the sets and the results are averaged surmising the covariance matrices and dividing the resulting matrix by the number of training sets used.

This differs from the application in the script because in the script the covariance is not calculated missing the relation between the points needed to correctly constrain the model for application to a new dataset.

Use of profile instead of derivative

According to the original paper the derivative of the grey profile is used to determine the best fit because this gives a higher accuracy than the grey profile. In the script the grey profile is used instead of the derivative possibly reducing the accuracy.

This is also visible in the next script lines, found in ASM_ApplyModel3D.m on rules 110 through 117:

```
  gi=reshape(gt(ind),k2,ns2);
% Calculate the PCA parameters, and normalize them
% with the variances.
% (Seems to work better with color images than
% the original method)
bc = PCAData(j).Evectors'*(gi-repmat(PCAData(j).Emean,1,ns2));
bc = bc./repmat(sqrt(PCAData(j).Evalues),1,ns2);
f(:,j)=sum(bc.^2,1);
```

`gt` is the grey profile taken from the new image. This while `dgt` is taken using the same function, visible in rule 149 of the same script:

```
[gt,dgt]=ASM_getProfileAndDerivatives3D(Itestsmall,posV*scale,N,n);
```

This function gives the profile along a given normal and the first derivative of the profile.

Attachment 4, folder structure

In figure 1 the main mapping structure is visible. Some folders contain parts of the script, others contain data.

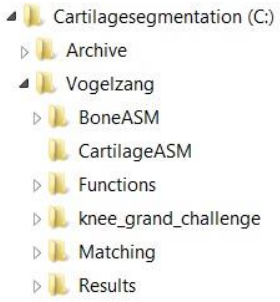

Figure 1, The primary layer of the mapping structure, separating the larger pieces of scripts, the results and the folder with training data.

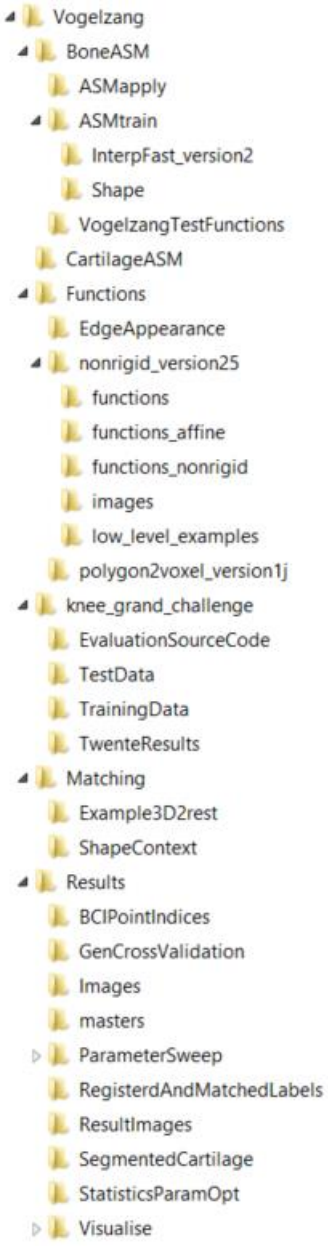

Figure 2, a more expanded view of the mapping structure.

These folders give the primary separation between important parts of the algorithm. Archive contains everything that was included when I first got the algorithm. I rebuild the folder structure in the folder Vogelzang. The folders BoneASM, CartilageASM and Matching contain the scripts that need to be run for the model, the folder Functions contains functions used in multiple pieces of the algorithm. The folder Results contains all the results generated by me, although further use is probably limited it could be usefull as a reference for the next person. The folder knee_grand_challenge contains the training data sets and the test data sets as well as an evaluation source code, the paper from the judges and the results that were sent in for the grand challenge.

A more unfolded structure is visible in figure 2. This shows that all folders still contain more subfolders in order to keep it organised. For running the scripts it is however not necessary to navigate these folders. In Matlab the folders are added to the search path and functions can easily be opened by right clicking the function within matlab and opening it. The shown structure has 6 folders in the first layer and 29 in total, including the primary folders. Within these folders are 3634 files. 291 of these are scriptfiles in .m or in .c, .h, .mexa64 and .mexw64.