Segmentation of Femoral Head from CT after Femoral Neck Fracture

Jan Horáček[1], Lukáš Maršálek[1,2], Martin Horák[3], Philipp Slusallek[2,4], Josef Pelikán[1]

[1]Faculty of Mathematics and Physics, Charles University, Prague, Czech Republic
[2]Saarland University, Saarbrücken, Germany
[3]Radiology Department of University Hospital Bulovka, Prague, Czech Republic
[4]DFKI, Saarbrücken, Germany

Abstract

We present a semi-automatic method for femoral head segmentation from CT dataset based on finding an optimal path through a polar transformation of axial slices. The cost function is a combination of corticallis properties (the directional behavior of 3D gradients and their magnitude in 2D slices, where they form typical "channels"). The volume is computed using filling and morphological operations. Since no explicit information about the shape of the femoral head is used, the method is also suitable for many other types of fractures or damaged femoral heads.
The implementation was experimentally validated on the Radiology Department of University Hospital Bulovka in Prague and allows radiologists to intuitively and accurately estimate the femoral head density in approximately 1 to 3 minutes.

1 Introduction

A precise segmentation of a femoral head is needed for a trabecular bone density estimation. Such estimations (in [1]) can be used both in the sense of post-injury analysis and treatment and also as a means of prevention of femoral neck accidents.

We present a novel approach using several existing methods to semi-automatically segment the trabecular bone of the femoral head even in the case of broken femoral neck or dislocated femur.

Here is a short overwiew of existing approaches:

Atlas based segmentation merges image segmentation with image registration [2, 3]. However, it is not good for segmenting dislocated hip joints or femoral heads with fractured necks.

A region growing approach in [4] is a very good approach for segmenting whole skeletal structures, but separating individual bones (for ex. pelvis and femoral head) may be problematic.

Segmenting using dynamic programming has been already tried in [5], although in a slightly different approach than our algorithm. Also the paper discusses only one slice segmentation and not the whole volume.

Deformable models [6, 7] may be used for this type of segmentation, especially after we have some sort of corticallis enhancing preprocessing. Deformable models were an option we considered, should the shortest path search algorithm not work.

We give a short overview of our method in Section 2, present preprocessing in Section 3, segmentation in Section 4 and postprocessing in Section 5. Sections 6 and 7 present our results and conclude this paper.

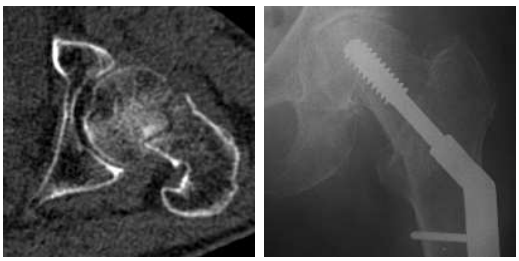

Figure 1: Example of a femoral neck fracture on CT and a postoperative CR of internal osteosynthesis.

2 Method overview

Our algorithm strives for a simple solution, both from the implementation and theoretical point of view. Moreover, we require our method to find the corticallis of the femoral head knowing only that it has a shape of a deformed sphere and that it is locally more dense than its close surroundings. First, we preprocess the dataset to remove noise and enhance the corticallis with our custom cost function. Next, we segment this enhanced dataset using a shortest path search in polar coordinates and obtain a volume of the femoral head. Finally, we postprocess this volume using morphological operations and perform statistical bone quality analysis on it.

3 Preprocessing

Good preprocessing makes the segmentation phase much easier. To obtain good resolution for segmenting the femoral head, our input CT scans are obtained with very small collimation (0.6mm – 0.8mm). However, such collimation introduces high levels of *noise*. For its removal it is usually enough to apply a Gaussian lowpass filter with a kernel of size 5^3. This removes most of the noise and at the same time does not modify the visibility of corticallis very much.

3.1 Corticallis enhancement functions

First we compute the *directional corticallis enhancement* functions c_X, c_Y, c_Z, which utilizes our model of

how the corticallis *should* look in a 1D cut. Detected corticallis in given direction is represented by locally higher values.

A *complete corticallis enhancement* function $c(x, y, z)$ is computed by taking the maximum of $c_X(x, y, z), c_Y(x, y, z), c_Z(x, y, z)$ for each point (x, y, z). The corticallis is represented by locally higher values.

A function $d(x, y, z)$ is a clamped and lowpass filtered linear transformation of $c(x, y, z)$. The *final cost* function $e(x, y, z)$ suitable for graph search algorithms is a clamped linear combination of function $d(x, y, z)$ and the magnitude of the 3D gradient vector.

Here we will give an example for computation in the X axis and how to compute an enhancing function $c_X(x, y, z)$ in this direction. Let's denote the original Gauss-filtered data $f(x, y, z)$ and its gradient $\nabla f(x, y, z)$. For each point we look in both directions along the axis and look at the two closest neighboring gradients $g_{L1}, g_{L2}, g_{R1}, g_{R2}$ and choose the one with larger magnitude.

$$\begin{aligned} g_{L1} &= \nabla f(x-1, y, z) \\ g_{L2} &= \nabla f(x-2, y, z) \\ g_L &= \begin{cases} g_{L1}, & |g_{L1}| > |g_{L2}| \\ g_{L2}, & |g_{L1}| \leq |g_{L2}| \end{cases} \end{aligned}$$

Computation of g_{R1}, g_{R2} and g_R is similar, only with indices $+1$ and $+2$.

Next, we compute an angle between each of those two gradient vectors and a vector pointing from the position where the gradient was taken to the point (x,y,z). We get angles α_L, α_R. A graphical representation of these angles can be seen in Figure 2.

$$cos(\alpha_L) = \left\langle g_L, \begin{bmatrix} x \\ y \\ z \end{bmatrix} - pos(g_L) \right\rangle$$

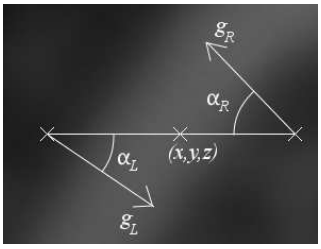

Figure 2: A graphical representation of angles α_L and α_R. The brighter path in the middle of this image represents magnified corticallis.

If any of these two angles α_L, α_R is greater than or equal to 90° ($cos(\alpha_i) \leq 0$) then the point (x, y, z) is probably not the local maximum in the X direction and thus we do not compute further and finalize the *directional corticallis enhancing* function as $c_X(x, y, z) = 0$.

If the two angles are *less* than 90°, we compute the final value of $c_X(x, y, z)$ by multiplying the magnitudes of gradients(g_L and g_R) with a negative cosine of their angles (the more opposite they are, the bigger the function value is).

$$\begin{aligned} \beta_g &= acos\left(\frac{\langle g_L, g_R \rangle}{|g_L| \cdot |g_R|}\right) \\ c_X(x, y, z) &= -cos(\beta_g) \cdot |g_L| \cdot |g_R| \end{aligned}$$

For c_Y and c_Z the only difference is that we take the neighbor gradients not along the X axis, but along the Y and Z axis, respectively. The final *corticallis enhancing* function is computed as a maximum of these three functions.

$$c(x, y, z) = max(c_X(x, y, z), c_Y(x, y, z), c_Z(x, y, z))$$

Searching 2 neighbor voxels is good for data with voxel dimensions around 0.5-0.8mm. One voxel is not enough, 3 voxels would already reach to the pelvic bone acetabulum in some places.

In the end our *corticallis enhancement* method proved to be quite good in localizing corticallis, while suppressing most of the other structures and trabecular bone. Of course, noise may bring local peaks of the cost function, but in the overall view these peaks are mostly solitary (and after the Gaussian filter also quite small) and insignificant in the segmentation step where some global localization of connected paths is applied.

3.2 Final cost function

Other information that we would like to add is the gradient magnitude, which creates *channels*, or paths with very small values on the ridge of corticallis and larger values around. Also very high peaks are found on the borders of very dense bones, such as the pelvis.

The drawback of gradient magnitude is that it also enhances the trabecular bone and is often confused by noise. So we would like to utilize the information from both our corticallis enhancement method and the gradient magnitude to increase robustness.

The corticallis enhancement method is able to pinpoint the corticallis, while gradient magnitude makes accidental confusion of the pelvis for the femoral head less probable. The gradient magnitude does not spoil the cost function on places where the actual femoral corticallis is, but adds other *penalties* around, so that we can stick to it during segmentation.

The actual cost function adjustment is as follows. First we clamp the values to remove peaks. Next, the function is filtered with a Gaussian filter, kernel size 5^3. The raw function $c(x, y, z)$ has mostly a form of curves and surfaces about 2-4 voxels vide (see Figure 3 (d) and (e)) and smoothing helps the optimal path search algorithm to find its axis. Let's denote this result a modified cost function $d(x, y, z)$.

The final function $e(x, y, z)$ is computed by subtracting $d(x, y, z)$ from the gradient magnitude. We have to make sure that it is non-negative everywhere (for the sake of a correct graph-search algorithm, more in Section 4.2). The clamping prevents the occurence of few solitary peaks that would attract the graph-search algorithm. A step-by-step example of cost function construction can be seen in Figure 3.

4 Segmentation

[8] presents a method for segmenting in polar coordinates, which inspired our approach. Such approach is convenient for spherical or cylindrical objects.

We need to find several input parameters: the upper and lower extremes where to begin and end segmentation and the center of polar transformation in each slice. For our case we also need a starting point

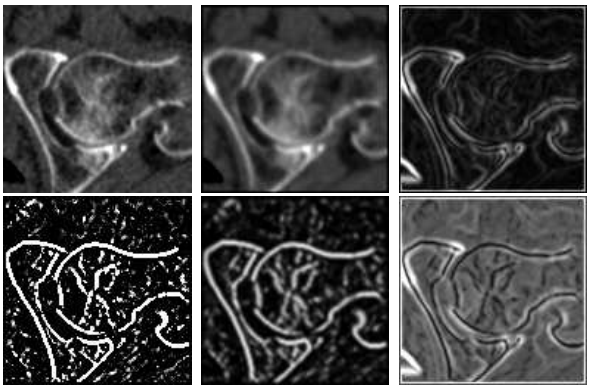

Figure 3: Cost function construction (from top left to bottom right): (a) Original data. (b) Gauss-filtered original data. (c) Gradient size approximation. (d) Corticallis enhancing function $c(x, y, z)$. (e) Function $d(x, y, z)$. (f) Final cost function $e(x, y, z)$.

for segmentation and correction points that add additional shape information to the algorithm.

The advantage of using polar coordinates for spherical and cylindrical objects is that even in places where there is no relevant information for segmentation, the algorithm has a tendency to go along the shortest path (in polar coordinates along a line, in spatial coordinates along a circle) until it hits another defined path.

Our chosen 3D segmentation is based on a slice-by-slice basis. Each of these slice segmentations creates a contour that locates the femoral corticallis. The steps performed during the process are as follows. First perform segmentation on several user-defined slices that provide seeds for the rest of the slices. Then use these seeds and perform segmentation on the rest of the data and finally floodfill each slice, so that the whole femoral head is filled.

4.1 Slice selection

Only one *control slice* is insufficient for real world CT scans. The most of the friction happens in the proximal posterior (or *upper rear*) part of the femoral head, the articular cartilage is thinned and the density of corticallis is reduced, so that it is barely visible in the CT scan. The pelvis is much denser than the corticallis and confuses the algorithm.

The most unclear slice is the *sagittal* cut. Here we can add most of the required information in one or a few slices. So increased effort in segmenting this slice gives us an advantage in segmenting the other slices, because it generates seeds in the most difficult part of the femoral head. More control slices should help to overcome a larger unclear area.

After the sagittal cuts are segmented, one slice in the *frontal* plane brings spatial coherency to the final axial segmentation. This frontal plane can be corrected by the user as well.

Now the final segmentation in *axial* planes starts. It finds the extremes from the sagittal control cuts and performs segmentation in each slice in between. This part usually runs completely automatically. In case any of the slices are too unclear, the axial slices can be corrected as well. But here we usually do not encounter any problems, only a very small number of slices goes wrong (they are usually near the extremal positions along the up/down direction), larger errors should be corrected by adding another control sagittal slice.

4.2 Graph search algorithm

Our *input* in each slice is a set of one or more control points and the CT slice data. *Output* should be a set of voxels that define the border of the femoral head, or more precisely, the *corticallis*. As mentioned before, we use a polar transformation in each slice to simplify the data walkthrough (see Figure 4).

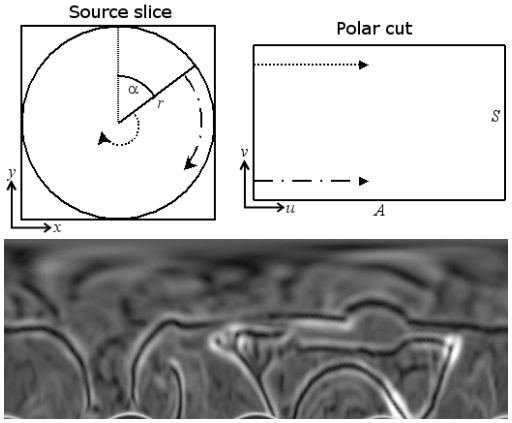

Figure 4: Polar transformation of the cost function $e(x, y, z)$ from Figure 3.

The path along the corticallis consists of low values and is almost continuous, but may be broken in some places and some segments may be missing completely. Additionally, we have a set of control points that we want to *hit* in a specific order (this order is given by their respective positions along the u coordinate).

We have used *Dijkstra's shortest path graph search* algorithm [9] for this problem. We define the weighted directed graph for this algorithm as follows: vertices V are the pixels in the polar cut image. Edges E lead from each pixel to its right neighbor, upper right neighbor and lower right neighbor if they exist. For vertices at the horizontal end, the right neighbors are defined as vertices created from pixels at the beginning of horizontal coordinates (with coordinate $u = 0$). The *weight* of each edge is defined as the original pixel value of the vertex the edge is pointing to.

Let's consider k control points $C_0, \ldots, C_{k-1}$ sorted by their u coordinate. We want to find a path between each two consecutive control points in this image (the *last* control point has the *first* control point as its consecutive neighbor). So we define the graph as mentioned above and run the shortest path search algorithm for each pair $C_i, C_{(i+1) mod\ k}$ where $i \in [0, k-1]$.

The set of voxels obtained by backward transformation from polar to spatial coordinates define the femoral head corticallis.

4.3 User-defined cutting plane

Three user-defined control points in the CT dataset define the boundary between femoral head and femoral neck. This plane defines the end of femoral head and

prevents from including unwanted data from the fracture. It is used as a location for creating an artificial corticallis for leading the segmentation algorithm through the femoral neck. This is done by creating a volume where voxels close to the plane are 0.0 and everywhere else 1.0. Next step is Gaussian filtering and multiplication with the original data.

5 Postprocessing

When the corticallis is correctly segmented as a borderline in each axial slice, the whole volume is obtained by running a 2D floodfill algorithm in each axial slice with a seed in the middle of the original polar computation. This way we get the whole volume *including* the corticallis. After this we need the radiologist to estimate the thickness of the corticallis and for each cortical voxel perform an *erosion*. Usually 2-3 voxels are enough.

6 Results

Our testing dataset consisted of 30 patients. For a correct segmentation 35% of the patients needed 3 or less control points, 45% needed 4-10 points, 14% needed 11-20 points and for 6% needed more than 20 points. Only 1 patient was not correctly segmented (included between those that needed more than 20 control points). This is a good result considering that most of the patients were mostly of higher age and the femoral head corticallis is either very weak compared to the pelvic corticallis or those two bones almost touched.

We tried how fast a user can learn to do precise segmentation: for a radiologist it was enough to demonstrate one segmentation and let him perform the second one, guiding him by providing information about the algorithm and possible problems. The third segmentation was already correct without a guide.

We have performed a number of tests with one dataset segmentation on a quad core Intel Q6600 computer. The algorithm is parallelized for multiple threads. The times proved to get advantage of multithread processing with a roughly linear acceleration from each additional thread up to about 5 threads. The times for a complete segmentation are from 3.5 to 18 seconds depending on the configuration and number of threads, time for preprocessing is about 1.9 seconds.

7 Conclusion

We have presented a novel approach based on a combination of several existing methods with some advancements in preprocessing. This approach is a working method that is much easier than many existing algorithms, both in the sense of actual program implementation and debugging (much easier than for example various deformable models algorithms) or in the sense of the first-time data preparation (compared to registration and atlas-based segmentations).

Most existing algorithms use only the gradient and its magnitude as a preprocessing step for a segmentation, which is not applicable for CT images of elderly patients with damaged articular cartilage (our target group). Our preprocessing and the possibility to correct the segmentation performs much better with such data. Also no previous estimation of bone positions is necessary. That enables segmentation of dislocated and/or fractured femur. That would be very difficult using some sort of algorithm that relies on an estimated structure (such as atlas-based segmentation).

The time complexity of the algorithm is quite low. Although not real-time, it runs in a reasonable time of several seconds for the *whole* segmentation. One slice segmentation can be considered a real-time algorithm on a mid-level modern PC.

The algorithm performs quite well for an average CT dataset, it takes about 3-10 control points to completely segment a femoral head with broken femoral neck and the segmentation is done in about 1.5 to 3 minutes including loading and visual error checking. Many patients can be segmented with only 1-3 control points in less than a minute.

During testing 97% of patients were correctly segmented. Only one was unsuccessful but the error was insignificant for the final bone density estimation (mean HU error less than 5%).

Our application was used in the Radiology Department of University Hospital Bulovka in Prague for bone quality estimation during a research project that tried to find a relationship between trabecular bone density and suitability of a patient for screw treatment of a femoral neck fracture.

References

[1] F.A. Bonnaire, C. Buitrago-Tellez, H. Schmal, B. Götze, and A.T. Weber. Correlation of bone density and geometric parameters to the mechanical strength of the femoral neck. In *Injury*, volume 33, 2002.

[2] J. Maintz and M. Viergever. A survey of medical image registration. *Medical Image Analysis*, 2(1), 1998.

[3] J. Pettersson, H. Knutsson, and M. Borga. Automatic hip bone segmentation using non-rigid registration. In *Proceedings of the 18th ICPR*, volume 3, 2006.

[4] Y. Kang, K. Engelke, and Kalender W.A. A new accurate and precise 3-d segmentation method for skeletal structures in volumetric ct data. In *IEEE Trans. on Med. Imag.*, volume 22, 2003.

[5] G. Marti, C. Baur, and P.Y. Zambelli. Optimal femoral head contour segmentation in ct images using dynamic programming. In *Technology and Health Care*, volume 12, 2004.

[6] D. Terzopulos and K. Fleischer. Deformable models. In *The Visual Computer*, volume 4, 1988.

[7] V. Krajíček. Měření objemu v 3d datech. Master's thesis, Master Thesis, Faculty of Mathematics and Physics, Charles University, Prague, 2007.

[8] Y. Zhang, B.J. Matuszewski, and L.K. Shark. A novel medical image segmentation method using dynamic programming. In *Proc. of the MediVis*, 2007.

[9] Edsger W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1, 1959.