BoneSplit - A 3D Texture Painting Tool for Interactive Bone Separation in CT Images

Johan Nysjö, Filip Malmberg, Ida-Maria Sintorn, and Ingela Nyström
Centre for Image Analysis, Dept. of Information Technology,
Uppsala University, Sweden
{johan.nysjo,filip.malmberg,ida.sintorn,ingela.nystrom}@it.uu.se

ABSTRACT

We present an efficient interactive tool for separating collectively segmented bones and bone fragments in 3D computed tomography (CT) images. The tool, which is primarily intended for virtual cranio-maxillofacial (CMF) surgery planning, combines direct volume rendering with an interactive 3D texture painting interface to enable quick identification and marking of individual bone structures. The user can paint markers (seeds) directly on the rendered bone surfaces as well as on individual CT slices. Separation of the marked bones is then achieved through the random walks segmentation algorithm, which is applied on a graph constructed from the collective bone segmentation. The segmentation runs on the GPU and can achieve close to real-time update rates for volumes as large as 512^3. Segmentation editing can be performed both in the random walks segmentation stage and in a separate post-processing stage using a local 3D editing tool. In a preliminary evaluation of the tool, we demonstrate that segmentation results comparable with manual segmentations can be obtained within a few minutes.

Keywords

Bone Segmentation, CT, Volume Rendering, 3D Painting, Random Walks, Segmentation Editing

1 INTRODUCTION

Cranio-maxillofacial (CMF) surgery to restore the facial skeleton after serious trauma or disease can be both complex and time-consuming. There is, however, evidence that careful virtual surgery planning can improve the outcome and facilitate the restoration [27]. In addition, virtual surgery planning can lead to reduced time in the operating room and thereby reduced costs.

Recently, a system for planning the restoration of skeletal anatomy in facial trauma patients (Figure 1) has been developed within our research group [25]. As input, the system requires segmented 3D computed tomography (CT) data from the fractured regions, in which individual bone fragments are labeled. Although a collective bone segmentation can be obtained relatively straightforward by, for instance, thresholding the CT image at a Hounsfield unit (HU) value corresponding to bone tissue, separation of individual bone structures is typically a more difficult and time-consuming task. Due to bone tissue density variations and image imprecisions such as noise and partial volume effects, adjacent bones and bone fragments in a CT image are typically connected to each other after thresholding, and cannot be separated by simple connected component analysis or morphological operations. In the current procedure, the bones are separated manually, slice by slice, using the brush tool in the ITK-SNAP software [30]. This process takes several hours to complete and is the major bottleneck in the virtual surgery planning procedure.

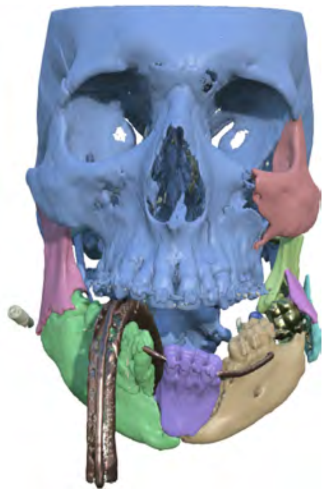
Figure 1: Example of a patient who has suffered complex fractures on the lower jaw and the cheekbone. The individual bone fragments in the CT image have been segmented with our interactive 3D texture painting tool to enable virtual planning of reconstructive surgery.

1.1 Contribution

Here, we present an efficient interactive tool for separating collectively segmented bones and bone fragments in CT volumes. Direct volume rendering combined with an interactive 3D texture painting interface enable the user to quickly identify and mark individual bone structures in the collective segmentation. The user can paint markers (seeds) directly on the rendered bone surfaces as well as on individual CT slices. Separation of marked bones is then achieved through the random walks segmentation algorithm [12]. A local 3D editing tool can be used to refine the result. In a preliminary evaluation of the bone separation tool, we demonstrate that segmentation results comparable with manual segmentations can be obtained within a few minutes.

1.2 Related Work

Model-based segmentation techniques have been used for automatic segmentation of individual intact bones such as the femur and tibia, but are not suitable for segmentation of arbitrarily shaped bone fragments. Automatic bone segmentation methods without shape priors have been proposed [10][18][2] but are not general enough for fracture segmentation.

Manual segmentation can produce accurate results and is often used in surgery planning studies. However, it is generally too tedious and time-consuming for routine clinical usage, and suffers from low repeatability. Another problem with manual segmentation is that the user only operates at a single slice at the time and thus may not perceive the full 3D structure. This tends to produce irregular object boundaries.

Semi-automatic or interactive segmentation methods combine imprecise user input with exact algorithms to achieve accurate and repeatable segmentation results. This type of methods can be a viable option if automatic segmentation fails and a limited amount of user-interaction time can be tolerated to ensure accurate results. An example of a general-purpose interactive segmentation tool is [6]. Liu et al. [22] used a graph cut-based [4] technique to separate collectively segmented bones in the foot, achieving an average segmentation time of 18 minutes compared with 1.5–3 hours for manual segmentation. Fornaro et al. [9] and Fürnstahl et al. [11] combined graph cuts with a bone sheetness measure [7] to segment fractured pelvic and humerus bones, respectively. Mendoza et al. [23] adapted the method in [22] for segmentation of cranial regions in craniosynostosis patients. The TurtleMap 3D livewire algorithm [16] produces a volumetric segmentation from a sparse set of user-defined 2D livewire contours, and have been applied for segmentation of individual bones in the wrist. It is, however, not suitable for segmentation of thin bone structures such as those in the facial skeleton. Segmentation of individual wrist bones has also been investigated in [15][24].

In all the semi-automatic methods listed above, the user interacts with the segmentation via 2D slices. A problem with using slice-based interaction for bone segmentation is that it can be difficult to identify, mark, and inspect individual bone structures and contact surfaces, particularly in complex fracture cases.

Texture painting tools [17][29] enable efficient and intuitive painting of graphical models (3D meshes) via standard 2D mouse interaction. Mouse strokes in screen space are mapped to brush strokes in 3D object space. Mesh segmentation methods [20] utilize similar sketch-based interfaces for semi-automatic labeling of individual parts in 3D meshes. Bürger et al. [5] developed a direct volume editing tool that can be used for manual labeling of bone surfaces in CT images. Our proposed 3D texture painting interface extends this concept to semi-automatic segmentation.

2 METHODS

Our bone separation tool combines and modifies several image analysis and visualization methods, which are described in the following sections. In brief, the main steps are (1) collective bone segmentation, (2) marking of individual bone structures, (3) random walks bone separation, and (4) segmentation editing.

2.1 Collective Bone Segmentation

A collective bone segmentation is obtained by thresholding the grayscale image at the intensity value t_{bone} (see Figure 2). The threshold is preset to 300 HU in the system, but can be adjusted interactively, if needed, to compensate for variations in bone density or image quality. The preset value was determined empirically and corresponds to the lower HU limit for trabecular (spongy) bone. Noisy images can be smoothed with a $3 \times 3 \times 3$ Gaussian filter ($\sigma = 0.6$) prior to thresholding. The Gaussian filter takes voxel anisotropy into account and can be applied multiple times to increase the amount of smoothing, although usually a single pass is sufficient. Both the thresholding filter and the Gaussian filter utilize multi-threading to enable rapid feedback.

2.2 Deferred Isosurface Shading

We use GPU-accelerated ray-casting [19] to render the bones as shaded isosurfaces. The isovalue is set to t_{bone}, so that the visual representation of the bones matches the thresholding segmentation. Similar to [14] and [13], we use a deferred isosurface shading pipeline. A 32^3 min-max block volume is used for empty-space skipping and rendering of the ray-start positions (Figure 3a). We render the first-hit positions (Figure 3b) and surface normals (Figure 3c) to a G-buffer via multiple render

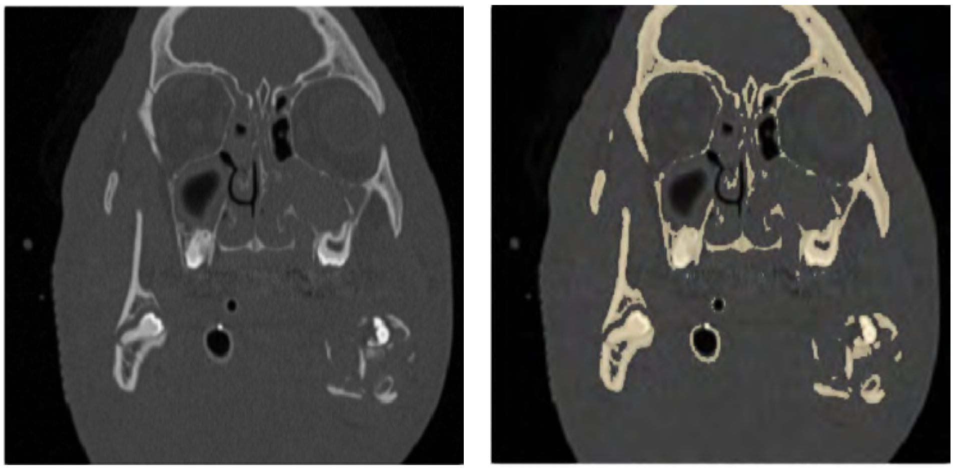

Figure 2: Left: Coronal slice of a grayscale CT volume of the facial skeleton. Right: Collective bone segmentation obtained by thresholding the CT volume at a Hounsfield unit (HU) value corresponding to trabecular bone.

targets (MRT), and calculate shadows and local illumination in additional passes. Segmentation labels are stored in a separate 3D texture and fetched with nearest-neighbor sampling in the local illumination pass.

Local illumination (Figure 3e) is calculated using a normalized version of the Blinn-Phong shading model [1]. To make it easier for the user to perceive depth and spatial relationships between bones and bone fragments, we combine the local illumination with shadow mapping to render cast shadows (Figure 3f). The shadow map (Figure 3d) is derived from an additional first-hit texture rendered from a single directional light source's point of view. The shadows are filtered with percentage closer filtering (PCF) [26] and Poisson disc sampling to simulate soft shadows. It is possible to disable the shadows temporarily during the segmentation if they obscure details of interest.

Ambient lighting is provided from pre-filtered irradiance and radiance cube maps [1]. Unlike the traditional single-color ambient lighting commonly used in medical visualization tools, the color and intensity variations in the image-based ambient lighting allow the user to see the shape and curvature of bone structures that are in shadow. The image-based ambient lighting also enables realistic rendering of metallic surfaces, e.g., metallic implants that have been separated out from the bones as part of the planning procedure. To enhance fracture locations, we modulate the ambient lighting with a local ambient occlusion [21] factor, which is computed on-the-fly using Monte-Carlo integration.

2.3 3D Texture Painting Interface

As stated in Section 1.2, a problem with 2D slice-based interaction is that it may be difficult to identify, mark, and inspect individual bone structures. Even radiologists, who are highly skilled at deriving anatomical 3D structures from stacks of 2D images, may find it difficult to locate and mark individual bone fragments in complex fracture cases. To overcome this issue, we implemented a 3D texture painting interface that enables the user to draw seeds directly on the bone surfaces.

Our 3D brush (Figure 4a) is implemented as a spherical billboard and uses the first-hit texture (Figure 3b) for picking and seed projection. The brush proxy follows the bone surface and can only apply seeds on surface regions that are visible and within the brush radius (in camera space). To prevent the brush from leaking through small gaps in the surface of interest, we compute a local ambient occlusion term from a depth map derived from the first-hit texture, and discard brush strokes in areas where the ambient occlusion value at the brush center exceeds a certain threshold. The radius of the ambient occlusion sampling kernel corresponds to the radius of the brush.

Additional tools include a label picker, an eraser, a floodfill tool, and a local editing tool (Section 2.6). A 3D slice viewer enables the user to mark occluded bones or place additional seeds inside the bones. The latter can be useful when the boundaries between the bones are weak or when the image is corrupted by streak artifacts from metal implants. We also provide interactive clipping tools that can be used to expose bones and contact surfaces. Both the 3D slice viewer and the clipping tools are useful during visual inspection and editing of the segmentation result.

2.4 Random Walks Bone Separation

Given the collective binary bone segmentation, the next step is to separate the individual bones and bone fragments. We considered two graph-based segmentation algorithms, graph cuts [4] and random walks [12], for this task. In the end, we selected the random walks

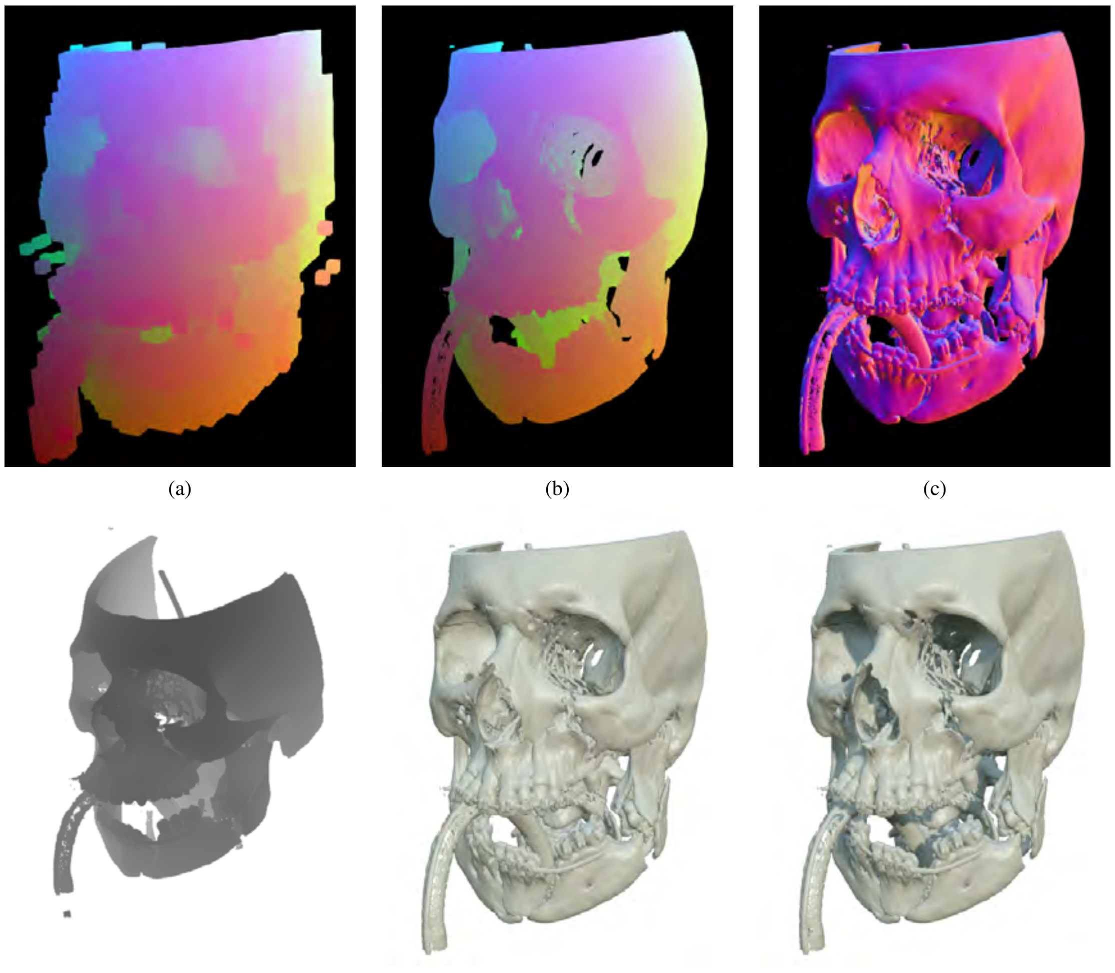
(a) (b) (c) (d) (e) (f)

Figure 3: Deferred isosurface shading pipeline: (a) ray-start positions; (b) first-hit texture; (c) surface normals; (d) shadow map derived from an additional first-hit texture rendered from a directional light source's point of view; (e) local illumination; (f) local illumination with shadows.

algorithm since it is robust to noise and weak boundaries, extends easily to multi-label (K-way) segmentation, and does not suffer from the small-cut problem of graph cuts. The main drawback and limitation of random walks is its high computational and memory cost (which, to be fair, is also a problem for graph cuts). For interactive multi-label segmentation of volume images, this has traditionally limited the maximum volume size to around 256^3, which is smaller than the CT volumes normally encountered in CMF planning. Our random walks implementation overcomes this limitation by only operating on bone voxels.

We construct a weighted graph $G = (V, E)$ from the collective bone segmentation and use the random walks algorithm to separate individual bones marked by the user. Figure 4 illustrates the segmentation process. For every bone voxel, the random walks algorithm calculates the probability that a random walker starting at the voxel will reach a particular seed label. A crisp segmentation is obtained by, for each bone voxel, selecting the label with the highest probability value. The vertices $v \in V$ in the graph represent the bone voxels and the edges $e \in E$ represent the connections between adjacent bone voxels in a 6-connected neighborhood. The number of neighbors can vary from zero to six. Each edge e_{ij} between two neighbor vertices v_i and v_j is assigned a gradient magnitude-based weight w_{ij} [12] defined as

$$w_{ij} = \exp(-\beta(g_i - g_j)^2) + \varepsilon, \tag{1}$$

where g_i and g_j are the intensities of v_i and v_j in the underlying grayscale image, and β is a parameter that determines the influence of the gradient magnitude. We add a small positive constant ε (set to 0.01 in our implementation) to ensure that v_i and v_j are connected, i.e., $w_{ij} > 0$. Increasing the value of β makes the random walkers less prone to traverse edges with high gradient magnitude. Empirically, we have found $\beta = 3000$ to work well for bone separation; however, the exact choice of β is not critical and we have used values in the range 2000-4000 with similar results.

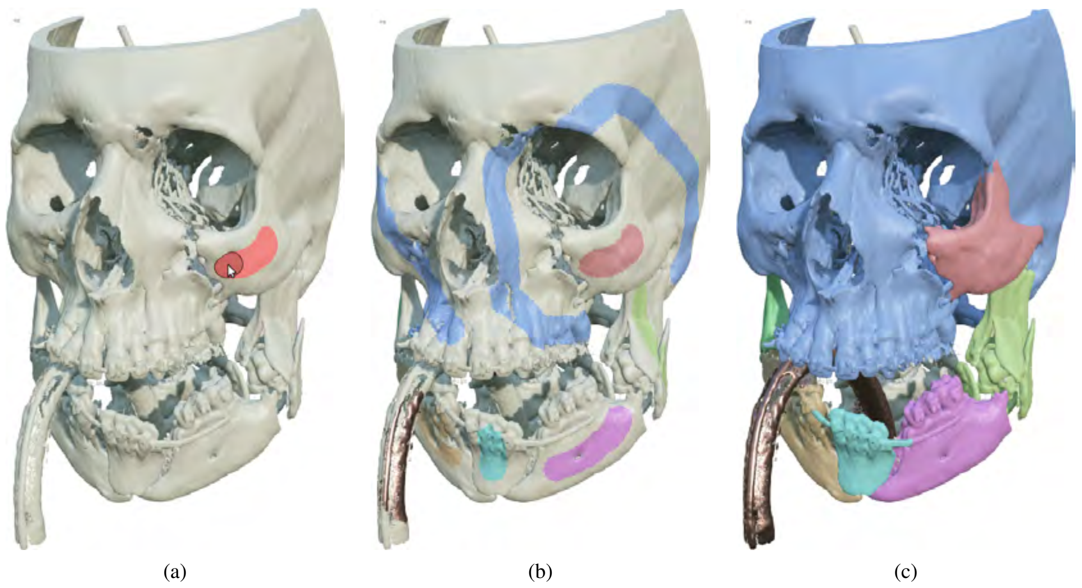

Figure 4: 3D texture painting interface for interactive random walks segmentation: (a) 3D brush used for painting seeds directly on the bone surfaces; (b) marked bones; (c) bone separation obtained with random walks.

As proposed in [12], we represent the weighted graph and the seed nodes as a sparse linear system and use an iterative solver (see Section 2.5) to approximate the solution for each label. By constructing the graph from the bone voxels in the collective segmentation, rather than from the full image, we simplify the random walks segmentation task from separation of multiple tissue types to bone separation. Moreover, we reduce the memory and computational cost substantially (by $\sim 90\%$ in our test cases). The head CT volumes encountered in CMF planning typically contain between 3 and 8 million bone voxels, which is a small fraction, $\sim 10\%$, of the total number of voxels. Combined with fast iterative solvers, this enables rapid update of the segmentation for volumes as large as 512^3.

A problem with constructing graphs from collective bone segmentations is that the sparse matrix A in the linear system becomes singular if some of the bone voxels are isolated (which, due to noise, is often the case.) This prevents the iterative solver from converging to a stable solution. The problem does not occur for graphs constructed from full images, where every voxel has at least one neighbor. To remove the singularity, we simply add a small constant weight $\kappa = 0.001$ to the diagonal elements in A. The value of κ is set smaller than ε to not interfere with the gradient weighting.

2.5 Iterative Solvers

We compute the random walks probability values iteratively using the Jacobi preconditioned conjugate gradient (CG) [28] method. The CG solver consists of dense vector operations and a sparse matrix-vector multiplication (SpMV), where SpMV is the most expensive operation. Although the Jacobi preconditioner improves the convergence rate, we found a single-threaded CPU implementation to be too slow for our problem sizes. Hence, to enable an interactive workflow, we followed the suggestion in [12] and implemented multi-threaded and GPU-accelerated versions of the solver. The multi-threaded solver was implemented in OpenMP and uses the compressed sparse row (CSR) matrix format for SpMV. The GPU-accelerated solver was implemented in OpenCL and supports two sparse matrix formats: CSR and ELLPACK [3]. ELLPACK has a slightly higher memory footprint than CSR, but enables coalesced memory access when executing the SpMV kernel on GPUs, which usually leads to better performance [3]. Our OpenCL SpMV kernels are based on the CUDA implementations in [3]. A benchmark of the implemented solvers is presented in Section 3.

2.6 Segmentation Editing

The user can edit the initial random walks segmentation by painting additional seeds on the bone surfaces or individual CT slices and running the iterative solver again. To enable rapid update of the result, the previous solution is used as starting guess [12]. Visual inspection is supported by volume clipping (Figure 5). The editing process can be repeated until an acceptable segmentation result has been obtained.

Further refinement of the segmentation can be achieved with a dedicated 3D editing tool (Figure 6), which updates a local region of the segmentation in real-time and allows a selected label to grow and compete with other labels. The tool is represented as a spherical brush and affects only voxels within the brush radius r. A voxel p_i marked with the active label will transfer its label to an adjacent voxel p_j in a 26-neighborhood if the editing

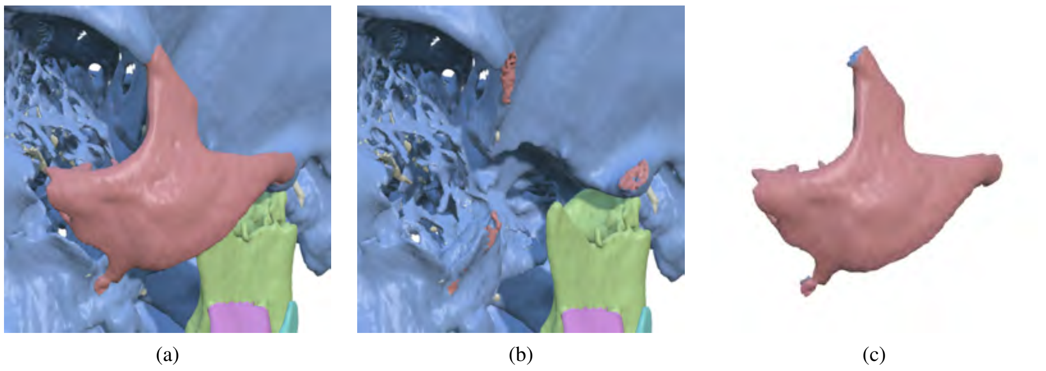
Figure 5: To support visual inspection and editing of bone fragments and contact surfaces, a segmented region (a) can be hidden (b) or exposed (c) via volume clipping . The clipping is performed by temporarily setting the grayscale value of the segmented region to $t_{bone}-1$ and updating the grayscale 3D texture.

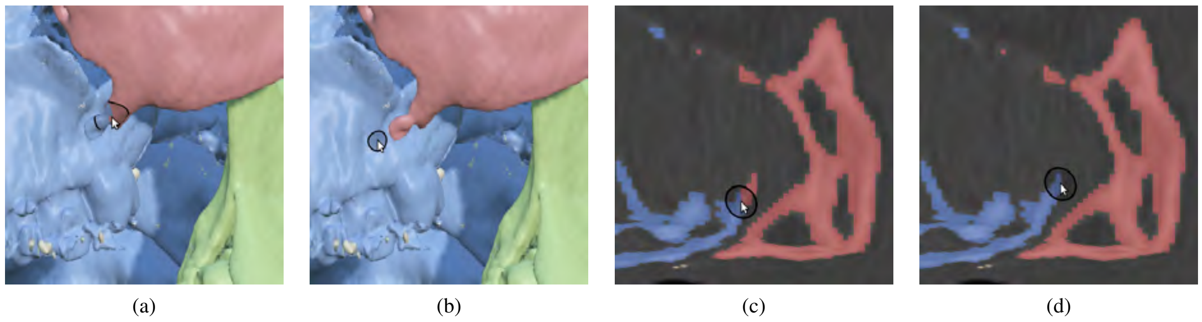
Figure 6: Segmentation editing performed with the local 3D editing tool.

weight function W_{ij} exceeds a given threshold. W_{ij} is defined as a weighted sum of the active label ratio, the gradient, and the Euclidean distance to the brush center.

2.7 Implementation Details

We implemented the segmentation system in Python, using OpenGL and GLSL for the rendering, PySide for the graphical user interface, and Cython and PyOpenCL for the image and graph processing.

3 CASE STUDY

To demonstrate the efficiency of our tool, we asked two non-medical test users to perform interactive segmentations of the facial skeleton in CT scans of three complex CMF cases. The first user, who had prior experience of manual bone segmentation and virtual surgery planning, was a novice on the system and received a 15 minutes training session before the segmentations started, whereas the second user (the main author) was an expert on the system. The CT scans were obtained as anonymized DICOM files. Further details about the datasets are provided in Table 1. Figures 7a–7c show the collective bone segmentations obtained by thresholding. Bone separation was carried out in three stages:

1. Initial random walks segmentation of marked bones.
2. Interactive coarse editing of the segmentation result by running random walks multiple times with additional seed strokes as input.
3. Fine-scale editing with the local 3D editing tool.

We measured the computational time and the interaction time required for each stage and asked the users to save the segmentation result obtained in each stage. Additionally, one of the users segmented case 1 manually in the ITK-SNAP [30] software to generate a reference segmentation for accuracy assessment. The manual segmentation took ∼5 hours to perform and was inspected and validated by a CMF surgeon.

To assess segmentation accuracy and precision, we computed the Dice similarity coefficient

$$DSC = \frac{2|A \cap B|}{|A|+|B|}. \tag{2}$$

DSC measures the spatial overlap between two multi-label segmentations A and B and has the range $[0,1]$, where 0 represents no overlap and 1 represents complete overlap.

The interactive segmentations (Figures 7d–7f) took on average 14 minutes to perform. As shown in Figure 8, most of the time was spent in the local editing stage (stage 3). DSC between the final interactive case 1 segmentations and the manual reference segmentation was 0.97782 (User 1) and 0.97784 (User 2), indicating overall high spatial overlap. The inter-user precision (Table 2) was also high and improved with editing.

Figure 9 shows a benchmark of the implemented CG solvers. The bars show the execution times (in sec-

Case	Region	Description	#Labels	Dimensions	Threshold	#Bone voxels
1	Head	Multiple fractures	15	$512 \times 512 \times 337$	260	4426530
2	Head	Multiple fractures	12	$512 \times 512 \times 301$	300	4769742
3	Head	Tumor	6	$230 \times 512 \times 512$	300	2787469

Table 1: Details about the CT images used in the case study.

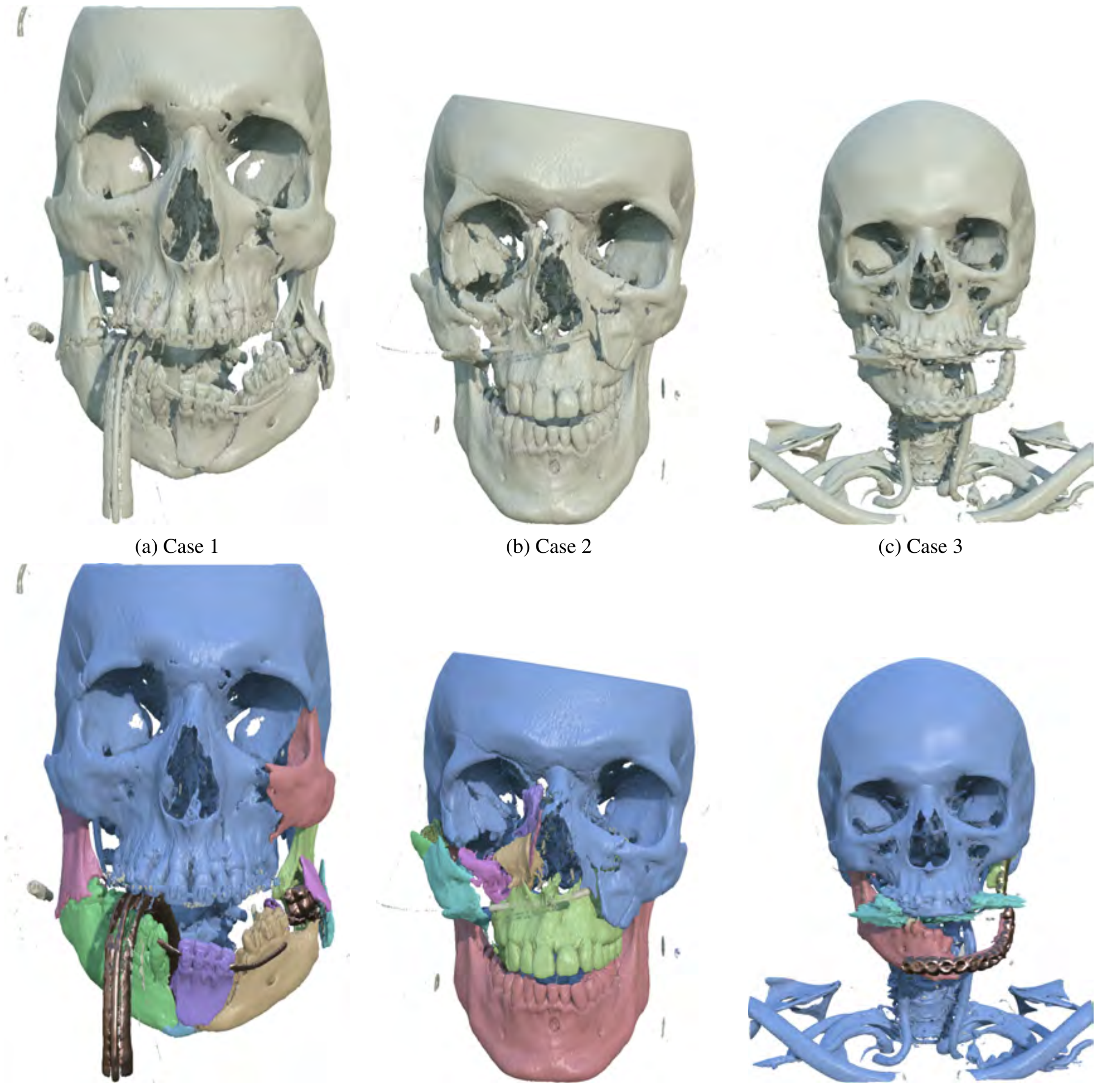

Figure 7: Top row: Collective bone segmentations. Bottom row: Separated bones.

onds) for computing an initial random walks solution on a graph with 4.6M bone voxels and 15 labels. The fastest GPU-based implementation had an average execution time of 0.4 seconds per label, which is a 14× speedup compared with the single-threaded CPU implementation and a 7× speedup compared with the multi-threaded CPU implementation.

4 DISCUSSION

Overall, we found the performance of the bone separation tool to be acceptable for surgery planning. Minor differences between segmentations generated by different users and between interactive and manual segmentations were expected due to the complex boundaries of the bone structures and the interactive editing.

Local editing (stage 3) is the most time-consuming part of the segmentation. The editing tool is of great aid for cleaning up the random walks segmentation and refining contact surfaces between separated bones or bone fragments, but will sometimes grow the active label too far or produce isolated voxels. Further modifications of the weight function could prevent this. Using connected component analysis for removing small isolated components in the segmentation could also be useful.

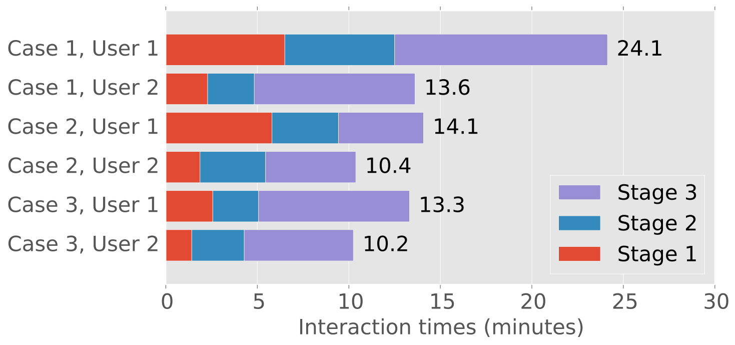

Figure 8: Interaction times (in minutes) for the two users.

Case	DSC		
	Stage 1	Stage 2	Stage 3
1	0.9199	0.9955	0.9971
2	0.9533	0.9968	0.9971
3	0.9832	0.99	0.9915

Table 2: Inter-user precision for the interactive segmentations.

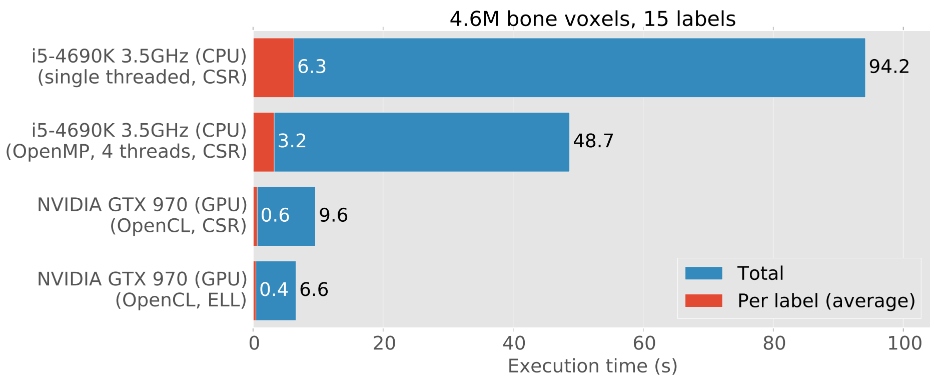

Figure 9: Benchmark of the CPU- and GPU-based Jacobi preconditioned CG solvers. The graph shows the timings (in seconds) for computing the initial random walks solution on a graph with 4.6M bone voxels and 15 labels. The number of iterations per label ranged from 45 to 136 (mean 83). Solver tolerance was set to $3 \cdot 10^{-3}$.

A limitation of our current approach is that the initial thresholding segmentation either tend to exclude thin or low-density bone structures or include noise and soft tissue. However, with minor modifications, the system should be able to display and process collective bone segmentations generated with other segmentation techniques. Postprocessing could potentially fill in holes.

5 CONCLUSION AND FUTURE WORK

In this paper, we have presented an efficient 3D texture painting tool for segmenting individual bone structures in 3D CT images. This type of segmentation is crucial for virtual CMF surgery planning [25], and can take several hours to perform with conventional manual segmentation approaches. Our tool can produce an accurate segmentation in a few minutes, thereby removing a major bottleneck in the planning procedure. The resulting segmentation can, as demonstrated in Figure 10, be used as input for virtual assembly [25]. Our tool is not limited to CMF planning, but can also be used for orthopedic applications or fossil data (Figure 11).

Next, we will focus on improving the efficiency of the local editing tool. We will also investigate if the accuracy of the random walks segmentation can be improved by combining the gradient-based weight function with other weight functions based on, for example, bone sheetness measure [7] or local edge density [22]. Finally, we will apply our segmentation tool on a larger set of CT images and perform a more extensive evaluation of the precision, accuracy, and efficiency.

6 ACKNOWLEDGMENTS

Our thanks go to Fredrik Nysjö (Centre for Image Analysis, Uppsala University) for performing the virtual assembly and generating reference segmentations for the case study. The head and wrist CT scans were provided by Uppsala University Hospital. The pelvis and

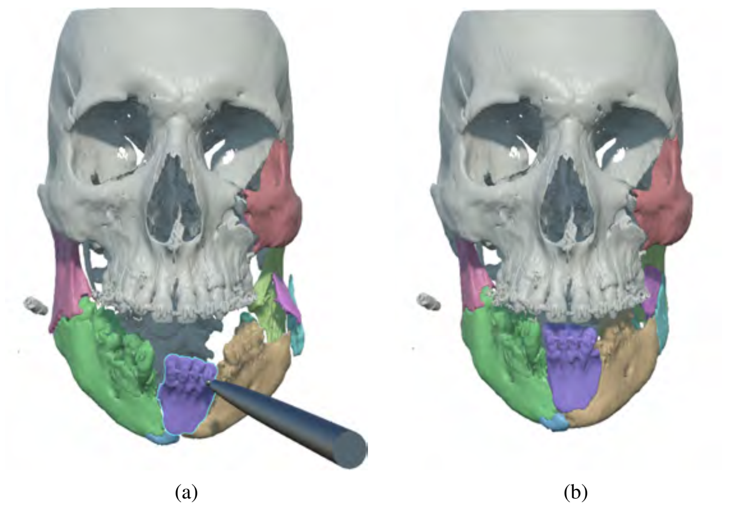
Figure 10: Haptic-assisted virtual assembly of one of the segmented cases, performed with the HASP [25] system.

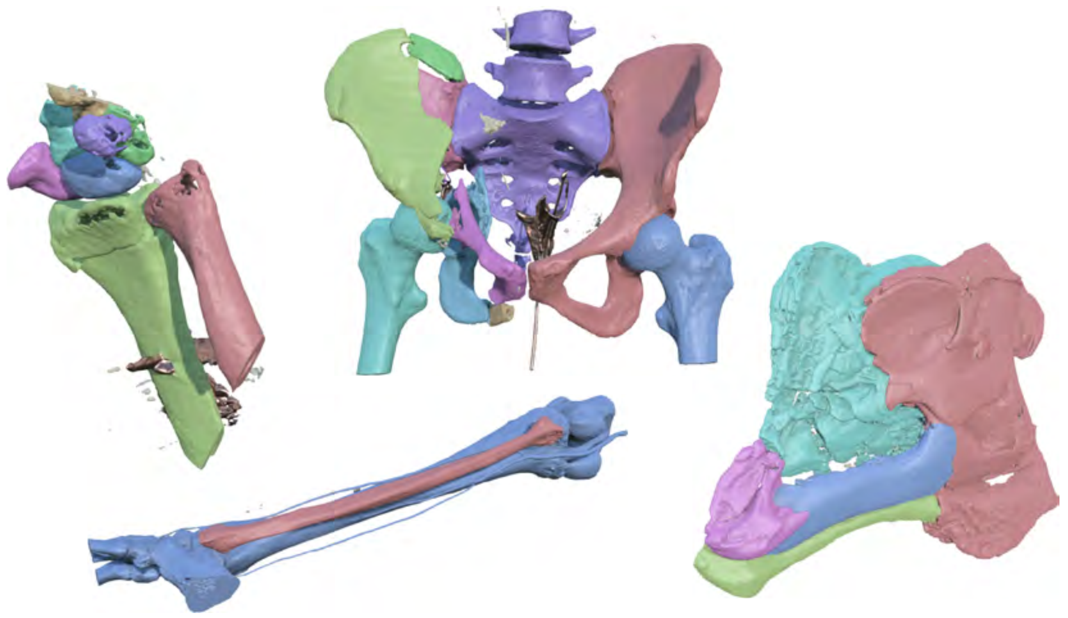
Figure 11: Our tool is not limited to head and neck CT scans; it can be used for rapid segmentation of individual bone structures in other regions such as the wrist, lower limbs, and pelvis. Another potential application (shown in the right image) is segmentation of fossils in μCT scans. Total segmentation time for these four cases was < 1 h.

fibula scans are courtesy of the OsiriX DICOM repository (http://www.osirix-viewer.com/datasets/), and the fossil μCT scan is courtesy of [8].

7 REFERENCES

[1] T. Akenine-Möller, E. Haines, and N. Hoffman. *Real-Time Rendering 3rd Edition*. A. K. Peters, Ltd., Natick, MA, USA, 2008.

[2] T. Alathari, M. Nixon, and M. Bah. Femur Bone Segmentation Using a Pressure Analogy. In *22nd International Conference on Pattern Recognition (ICPR 2014)*, pages 972–977, 2014.

[3] N. Bell and M. Garland. Implementing Sparse Matrix-vector Multiplication on Throughput-oriented Processors. In *Conference on High Performance Computing Networking, Storage and Analysis*, SC '09, pages 1–11. ACM, 2009.

[4] Y. Y. Boykov and M.-P. Jolly. Interactive graph cuts for optimal boundary and region segmentation of objects in ND images. In *8th IEEE International Conference on Computer Vision (ICCV 2001)*, volume 1, pages 105–112, 2001.

[5] K. Bürger, J. Krüger, and R. Westermann. Direct volume editing. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1388–1395, 2008.

[6] A. Criminisi, T. Sharp, and A. Blake. Geos: Geodesic image segmentation. In *European Conference on Computer Vision (ECCV)*, volume 5302 of *LNCS*, pages 99–112. Springer, 2008.

[7] M. Descoteaux, M. Audette, K. Chinzei, and K. Siddiqi. Bone enhancement filtering: application to sinus bone segmentation and simulation of pituitary surgery. *Computer Aided Surgery*, 11(5):247–255, 2006.

[8] A. Farke and R. M. Alf Museum of Paleontolog. CT scan of left half of skull of Parasaurolophus sp. (Hadrosauridae: Dinosauria). 2013.

[9] J. Fornaro, G. Székely, and M. Harders. Semi-automatic Segmentation of Fractured Pelvic Bones for Surgical Planning. In *Biomedical Simulation*, volume 5958 of *LNCS*, pages 82–89. Springer Berlin Heidelberg, 2010.

[10] P. Fürnstahl, T. Fuchs, A. Schweizer, L. Nagy, G. Székely, and M. Harders. Automatic and robust forearm segmentation using graph cuts. In *5th IEEE International Symposium on Biomedical Imaging (ISBI 2008).*, pages 77–80, May 2008.

[11] P. Fürnstahl, G. Székely, C. Gerber, J. Hodler, J. G. Snedeker, and M. Harders. Computer assisted reconstruction of complex proximal humerus fractures for preoperative planning. *Medical Image Analysis*, 16(3):704–720, 2012.

[12] L. Grady. Random Walks for Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(11):1768–1783, 2006.

[13] M. Hadwiger, P. Ljung, C. R. Salama, and T. Ropinski. Advanced Illumination Techniques for GPU Volume Raycasting. In *ACM SIGGRAPH ASIA 2008 Courses*, SIGGRAPH Asia '08, pages 1–166. ACM, 2008.

[14] M. Hadwiger, C. Sigg, H. Scharsach, K. Bühler, and M. Gross. Real-Time Ray-Casting and Advanced Shading of Discrete Isosurfaces. *Computer Graphics Forum*, 24(3):303–312, 2005.

[15] H. K. Hahn and H.-O. Peitgen. IWT-interactive watershed transform: a hierarchical method for efficient interactive and automated segmentation of multidimensional gray-scale images. In *Medical Imaging 2003*, pages 643–653. SPIE, 2003.

[16] G. Hamarneh, J. Yang, C. McIntosh, and M. Langille. 3D live-wire-based semi-automatic segmentation of medical images. In *Medical Imaging 2005*, pages 1597–1603. SPIE, 2005.

[17] P. Hanrahan and P. Haeberli. Direct WYSIWYG Painting and Texturing on 3D Shapes. *ACM SIGGRAPH Computer Graphics*, 24(4):215–223, 1990.

[18] M. Krcah, G. Székely, and R. Blanc. Fully automatic and fast segmentation of the femur bone from 3D-CT images with no shape prior. In *IEEE International Symposium on Biomedical Imaging*, pages 2087–2090. IEEE, 2011.

[19] J. Krüger and R. Westermann. Acceleration Techniques for GPU-based Volume Rendering. In *14th IEEE Visualization 2003 (VIS'03)*, pages 287–292, 2003.

[20] Y.-K. Lai, S.-M. Hu, R. R. Martin, and P. L. Rosin. Rapid and Effective Segmentation of 3D Models Using Random Walks. *Computer Aided Geometric Design*, 26(6):665–679, 2009.

[21] H. Landis. Production-ready global illumination. *SIGGRAPH Course Notes*, 16(2002):11, 2002.

[22] L. Liu, D. Raber, D. Nopachai, P. Commean, D. Sinacore, F. Prior, R. Pless, and T. Ju. Interactive Separation of Segmented Bones in CT Volumes Using Graph Cut. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI 2008)*, volume 5241 of *LNCS*, pages 296–304. Springer Berlin Heidelberg, 2008.

[23] C. S. Mendoza, N. Safdar, K. Okada, E. Myers, G. F. Rogers, and M. G. Linguraru. Personalized assessment of craniosynostosis via statistical shape modeling. *Medical Image Analysis*, 18(4):635–646, 2014.

[24] A. Neubauer, K. Bühler, R. Wegenkittl, A. Rauchberger, and M. Rieger. Advanced virtual corrective osteotomy. *International Congress Series*, 1281(0):684–689, 2005.

[25] P. Olsson, F. Nysjö, J.-M. Hirsch, and I. B. Carlbom. A haptics-assisted cranio-maxillofacial surgery planning system for restoring skeletal anatomy in complex trauma cases. *International Journal of Computer Assisted Radiology and Surgery*, 8(6):887–894, 2013.

[26] W. T. Reeves, D. H. Salesin, and R. L. Cook. Rendering antialiased shadows with depth maps. *ACM SIGGRAPH Computer Graphics*, 21(4):283–291, 1987.

[27] S. M. Roser and et al. The accuracy of virtual surgical planning in free fibula mandibular reconstruction: comparison of planned and final results. *Journal of Oral and Maxillofacial Surgery*, 68(11):2824–2832, 2010.

[28] J. R. Shewchuk. *An introduction to the conjugate gradient method without the agonizing pain*. Carnegie-Mellon University. Department of Computer Science, 1994.

[29] The Foundry. MARI. `http://www.thefoundry.co.uk/products/mari/`, 2015. Accessed on February 21, 2015.

[30] P. A. Yushkevich, J. Piven, H. Cody Hazlett, R. Gimpel Smith, S. Ho, J. C. Gee, and G. Gerig. User-guided 3D active contour segmentation of anatomical structures: Significantly improved efficiency and reliability. *Neuroimage*, 31(3):1116–1128, 2006.