BET2 - MRI-Based Estimation of Brain, Skull and Scalp Surfaces
FMRIB Technical Report TR06MP1

Mickaël Péchaud, Mark Jenkinson, Stephen Smith
Oxford University Centre for Functional MRI of the Brain (FMRIB)

6th February 2006

Contents

1 Introduction

Finding brain, skull and scalp surfaces using MRI data has important applications, especially in EEG/MEG forward or inverse modelling, where this allows one to use more accurate models than (the commonly-used) concentric spheres, thus increasing modelling precision.

BET2 (Brain Extraction Tool v2) is a fast and entirely automated tool for extracting mesh surfaces of brain, inner and outer skull and scalp, from MR images. It ideally requires both a T1- and a T2-weighted anatomical MRI, each of $<$2mm resolution in each direction, although it can still attempt to find the surfaces given only a single T1-weighted image. The resulting surface representations are convenient for BEM-based algorithms in MEG/EEG solutions, and are easy to transform into volumetric representation for FEM-based methods.

BET2 is based on the original BET (Brain Extraction Tool) [Smith, 2002], which finds the brain boundary given a single MR image, but not the skull and scalp surfaces. (This original BET tool can attempt to find a set of external skull surface voxels, but does not fit a surface to this, and the resulting crude "skull" image contains a relatively large number of false negatives and false positives.) BET2 is available as part of FSL (FMRIB Software Library, www.fmrib.ox.ac.uk/fsl).

2 Method details

BET2 comprises two main programs. *bet* takes a T1-weighted image as input, and performs brain extraction, i.e., outputs a mesh representation of the outer brain surface, including the cerebellum and part of the brainstem. This program implements in C++ a virtually identical brain-surface-finding algorithm to the original C BET program. *betsurf* then uses the T1-weighted volume and a T2-weighted volume from the same subject (henceforth these will be referred to simply as "T1" and "T2"). The T2 image must already have been registered to the T1. *betsurf* also requires the *bet*-produced brain surface mesh, and the affine transform matrix from the T1-weighted volume into MNI152 standard space (provided by the registration tool FLIRT [Jenkinson and Smith, 2001, Jenkinson et al., 2002]). *betsurf* outputs three meshes in the *.off* (Geomview) format and also images containing corresponding outline or mask volumes. *betsurf* is based on both local analysis (which determines the points that are likely to be part of the brain/skull, skull/scalp and scalp/background interfaces), and also a deformable model (which fits meshes to those points).

2.1 Intensity clamping

The relative intensity of brain and scalp on T1 can vary a lot from one scanner to another. In order to increase the robustness of *betsurf* to these variations, we constrain the maximum scalp intensity relative to the brain intensities. To avoid the problem of over-bright scalp, we compute a maximum intensity threshold; we take the "robust range" of intensities within the extracted brain volume (this being the 2% and 98% points in the cumulative histogram of brain voxel intensities) and increase the upper value by 10% of this range. We then clamp all voxels (in the whole-head image) to this threshold.

2.2 Surface point detection

A first analysis is computed along normals to the initial brain surface mesh, in order to detect points belonging to the three interfaces. All these analyses are computed locally, which gives robustness against low frequency changes in image intensity across the image.

Generally, in an MR image, the skull appears darker than brain and other tissues. Its intensity is comparable to that of background. In T1-weighted images, CSF is dark, sometimes as low as air/skull (see Figure 1). In T2, CSF is bright, and muscles are often dark (making muscle hard to distinguish from skull).

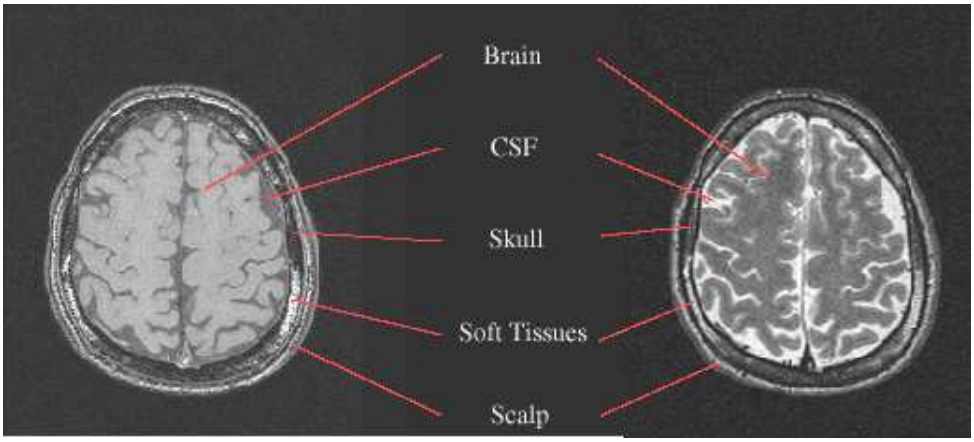

Figure 1: An axial slice from typical T1- and T2-weighted images

2.2.1 Main profile analysis algorithm

First, for each vertex on the initial brain surface mesh, we extract the intensity profiles of the T1 and T2 images along a line perpendicular to the mesh surface. The profile runs from 3mm inside to 60mm outside the surface, and the sampling step is 0.5mm. The profile is created by sampling the original images using trilinear interpolation. With adult human brains the mean mesh vertex spacing (and hence the mean distance between neighbouring perpendicular profiles) is approximately 5mm.

2.2.2 Outer skin surface

The outer scalp surface is found as the most outer point that is over 20% of the range from min to max in the T1 profile (see Figure 2 point 1). We check that this point is correct using the T2, by also finding the most outer point that is over 25% of the range from min to max in the T2 profile. If these two points are not close enough (<5mm), this probably signifies artefacts in one of the volumes. In this case we re-find the same points after changing the 60mm outer search limit to the minimum of the two original found "scalp" points, plus 15mm. This is intended to prevent sensitivity to artefacts on the border of the volumes. If the resulting points are then close enough (<10mm), we assert that the point found on T1 is acceptable. If not, we do not use this profile further.

2.2.3 Inner skull surface

We then compute the inner skull surface:

- Find the first point of the T1 profile that is under 25% between the minimum and the maximum intensities of the T1 profile (which we will refer to as "25% threshold" below); see Figure 2 point 3. This point should be in cerebro-spinal fluid or possibly in the skull if there is little CSF in this area.
- Compute the first point, that is outside of the previous point, in the T2 profile, and that is under 30% threshold (see Figure 2 point 2). This avoids placing the "inner skull point" in CSF lying between brain and skull.

We then set a new region of interest (ROI) in the profile using points 1 and 2: this will allow us to ignore bright CSF or dark background voxels for the thresholds that we will now compute.

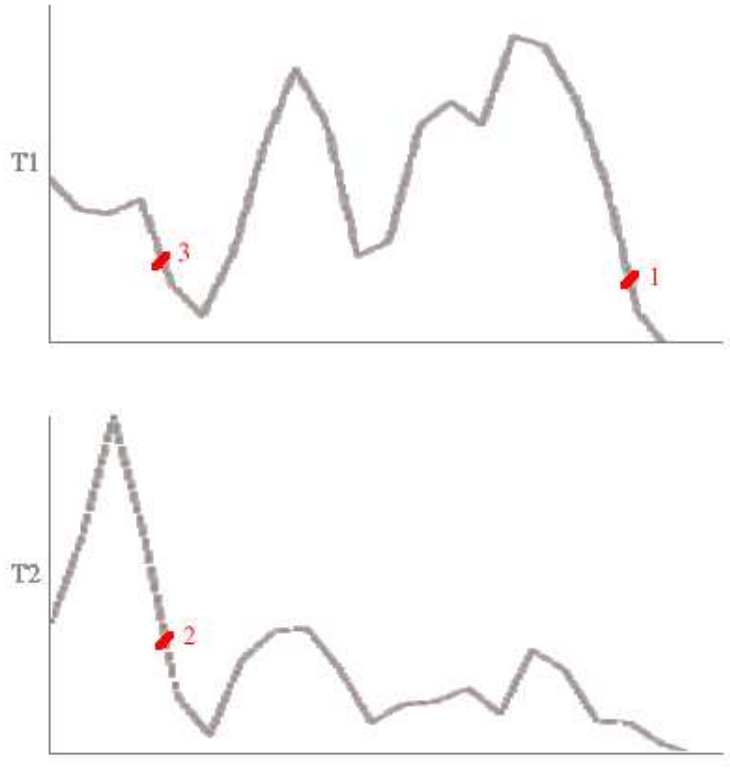

Figure 2: Inner skull and outer scalp algorithm on T1 and T2 - typical profiles

2.2.4 Outer skull surface

This is the most difficult point to find reliably, since the topology of skull and scalp can be quite different in different parts of the head, and between different subjects. The most obvious problem is the presence of soft tissue (marrow) within the skull, which is brighter than the rest of the skull on both T1 and T2, and prevents us from applying simple thresholding, or a traditional level set method. The second problem is that scalp intensity in the T1 is far from being uniform, and often has a complex structure. Just deflating the skin/background surface to find outer skull surface may find points in muscle rather than in skull.

The scalp is often simpler on T2 than on T1: it is just a single thin stripe, most of the time easily distinguishable from marrow. But the drawback is that the interface between skull and scalp is impossible to see on T2, since muscles are in the same range of intensities as skull.

We use the T2 to identify the general shape of the profile (are there soft tissues or not, and where are they?), and then the T1 to locate precisely the interface we are looking for.

- *determine if there is soft tissue in the profile, and set outer_skull_search_point*: start from the inner skull point in the T2, and, moving outwards, find pairs of points where the first is under the 30% threshold and the second is over the 40% threshold. If we reach the 40% threshold more than once, it means that soft tissue is likely to be present. We will then start our outer skull point search after the second point where the 30% threshold is reached (see Figure 3 point2), unless the first point over 40% is more than 10mm away from the inner skull point (which is likely to mean that we are in an area where scalp is complex on T2). By default, *outer_skull_search_point* is set to inskull point plus 0.5mm. We now set the inner limit of the ROI to this point.

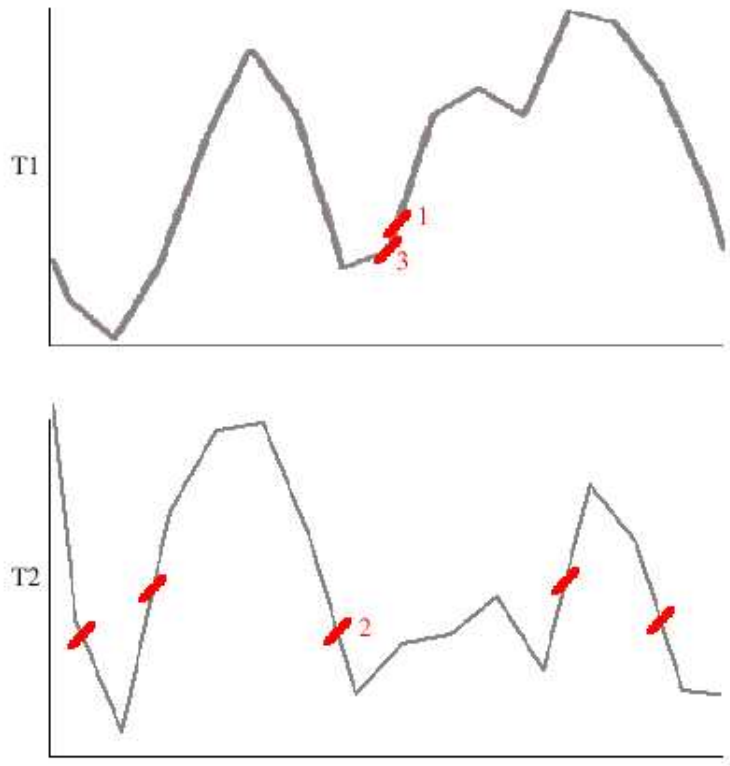

Figure 3: Outer skull algorithm on T1 and T2 - typical profiles

- *compute a possible outer skull point moving inwards from the outer skin* (this computation is performed so that if soft tissues are present and hardly distinguishable from scalp, the point found will not be in the middle of the skull): find the last point over 70% threshold in T1 moving inwards from the outer skin point (the inner limit of this search being restricted by the previous step), and then, moving inwards from here, the last point under 30% threshold (see Figure 3 point1). Most of the time, this point is close to the true outer skull point, but the complexity of scalp structure can lead to it being wrong; the point is often found within muscles, 2 or 3 mm too far away from the brain. Soft tissues can also lead to problems here. So we now check if this point is good, starting from inner skull point.
- *find the outer skull point from inside*: we start from the T2-derived point 2. The algorithm moves out along the T1 profile, detects the furthest point under 15% threshold, as long as it is before the first point over 25% threshold. This point is the outer skull point(Figure 3 point3). (We need to use the 25% threshold condition to try to avoid detecting muscles in the scalp as part of the skull (Figure 4)).
- we *compare* the two points found (1 and 3), and if 1 is less than 2mm outside 3, and less than 3mm inside it, we assert that 3 is the optimal one. The 3mm tolerance is for the case when the first point is found in muscle, and the second one is good. If the two points are not close enough, the profile is rejected. If only one of the two methods were used, too many incorrect points are obtained by the algorithm.

This algorithm detects correctly the outer skull points in much of the head, but fails on the lower areas, where tissues and skull topology are too complex. Thus we have an alternative algorithm for those areas.

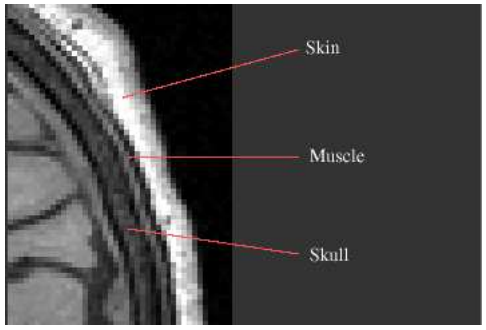

Figure 4: Muscle on a T1

2.2.5 Profile analysis algorithm for the "inferior-anterior head"

To determine the area in which this alternative algorithm is needed, we use a-priori spatial knowledge: this area has been manually determined in standard space. It consists of a space delimited by being below a plane (Figure 5), minus a solid region for the frontal sinuses where the general algorithm works correctly (Figure 6). We use FLIRT to derive a standard space transform matrix and transform this mask into the space of our data.

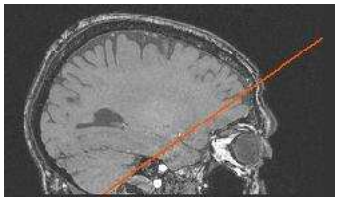

Figure 5: Plane delimiting the "inferior-anterior head"

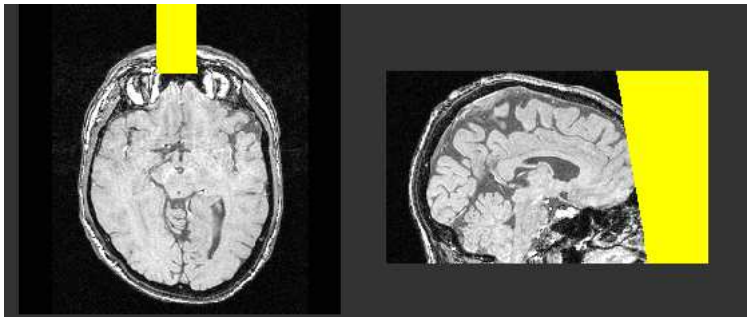

Figure 6: Frontal area excluded from the "inferior-anterior head" special case algorithm

The topology of skull is generally too complex in the "deleted" area to be represented by the simple (spherical topology) meshes that we are using; we simply model it as being thin here. The outer skin and inner skull points are detected as explained above. The outer skull is detected as the first point over a 35% threshold after the first point under a 25% threshold on the T1 profile. If this first point is too far (greater than 7mm) outside the inner skull we revert to the simplified assumption that the outer skull point is 0.5mm outside the inner skull point. Because these inferior-anterior areas are on the opposite side of the brain to where EEG/MEG sensors are placed, this approach is expected to be sufficient for generating surfaces/volumes for use in EEG/MEG analysis.

2.3 Mesh fitting

We now have detected image points corresponding to the three expected surfaces. The next step is to fit meshes to those surfaces - and to force those meshes to not intersect with themselves and each other.

2.3.1 Blurring

The detected surface points are drawn into three different 3D images, one for each surface. The images are then blurred by convolving with a Gaussian with $\sigma = 3mm$, in order to create an attraction area near the surface points.

2.3.2 Mesh expansion

First, the inner skull surface mesh is computed. The mesh is initialised using the outer brain surface, provided as input to *betsurf*. We use exactly the same force model as *bet* to expand the mesh: at each step of computation, 3 forces are applied to the points:

- The first component is tangential to the local surface and aims at maintaining a uniform distance between the points of the mesh.
- The second is normal to the local surface and controls surface smoothness.
- The third is proportional to the local gradient of the blurred volume in the normal direction, and causes the mesh points to be attracted to maximum value points of the volume.

In order to speed up the algorithm, the third term is put to its maximum value if the intensity of the corresponding voxel is too low, i.e., if we are far from the detected points (we will refer to this as mesh deformation speed-up). Since this is likely to force an over-expansion of the mesh if no points were detected in an area, we find the maximum intensity along the normal of each point. If it is too low, we let this point be handled by the smoothness term only. This computation is fairly slow, so we do it only once every 100 mesh update iterations.

We apply 800 mesh update iterations to find the inner skull mesh.

We then do the same for the outer skull mesh, starting from inner skull mesh, except that we have to avoid intersection with previously computed meshes. To do this, we draw a binary mask of the previous mesh, and if a point appears to be near the previous mesh (e.g. if the corresponding value on the mask using trilinear interpolation is greater than zero), we replace the second and the third term by a force that pushes outwards.

This can fail in 2 cases:

- The first is shown in Figure 7: the yellow voxels represent the inner skull mesh. The red voxels are points that were incorrectly detected as outer skull points. If the mesh is attracted by those points, it will self-intersect.

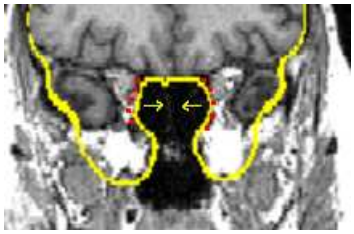

Figure 7:

The simple solution is to remove all points (before blurring) of the outer skull edge image which are inside the inner skull mask.

- The second problem is a drawback of discretization that can occur when drawing the inner skull mesh leads to a change of the topology of the inside of mesh (Figure 8). The mesh will expand and follow the indicated arrow,

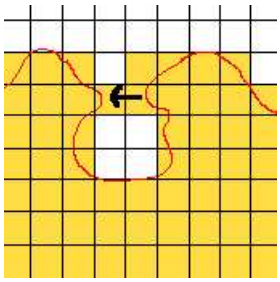

Figure 8: Change of topology in 2D due to discretization

causing a self intersection. This pattern was not found in the volumes we tested.

The scalp surface is computed starting with a sphere computed from the previous mesh : the centre is the mean of all the points of the outer skull mesh, and the radius is the mean of distances between the centre and the initial mesh points multiplied by 0.75. (The mesh deformation speed-up term leads to self intersection of the mesh if we start from the outer skull mesh). We apply 1500 mesh deformation iterations.

The same algorithm as above is used to avoid intersecting the outer skull mesh.

This part of the algorithm does not explicitly force non-intersection of the various meshes. Self-intersection is theoretically possible, but involves many tests for all the points at each step of computation, and has not been seen to occur to date.

2.4 T1-only analysis

For the (less optimal) cases where only a T1-weighted image is available, we have developed a variant of the above. The technique is very similar to that described above, except for outer skull point detection, where we cannot use a T2 to determine whether there are soft tissues or not. It gives decent meshes in many cases, but will fail if soft tissues within the skull are too bright.

2.4.1 Profile analysis algorithm

The outer skin point detection is the same as for the main algorithm.

The inner skull point detection is a simple thresholding: we detect the first point under a 25% threshold. This threshold is suitable for most of the volumes that were tested, but since the CSF is dark on the T1 and is sometimes hard to distinguish from skull (a problem in large atrophy cases), we allow the user to adjust this threshold.

The outer skull points are detected by starting from the inner skull point, and finding the furthest point under a 15% threshold before finding a point over a 60% threshold. We need such a high threshold since soft tissues can be very bright - but this will cause the algorithm to sometimes move too far and fall into muscle, often up to a 2mm error. As before, we check this point by also starting from outside the scalp.

2.4.2 Mesh fitting

We use exactly the same algorithm as in the first method.

3 Example results

In this section we will show some of the first results of *betsurf*.

3.1 Comparison with CT

A good way to validate the results is to compare skull masks obtained by *betsurf* with a computed tomography (CT) image, where skull is very bright and easily distinguished from soft tissue. Unfortunately, such data is not commonly available for whole-head coverage at high resolution; we tested one subject with CT, T1 and T2 volumes with relatively thick slices (.4x.4x3mm resolution for the CT and .8x.8x3mm for the MRI). The T1 is shown in Figure 9.

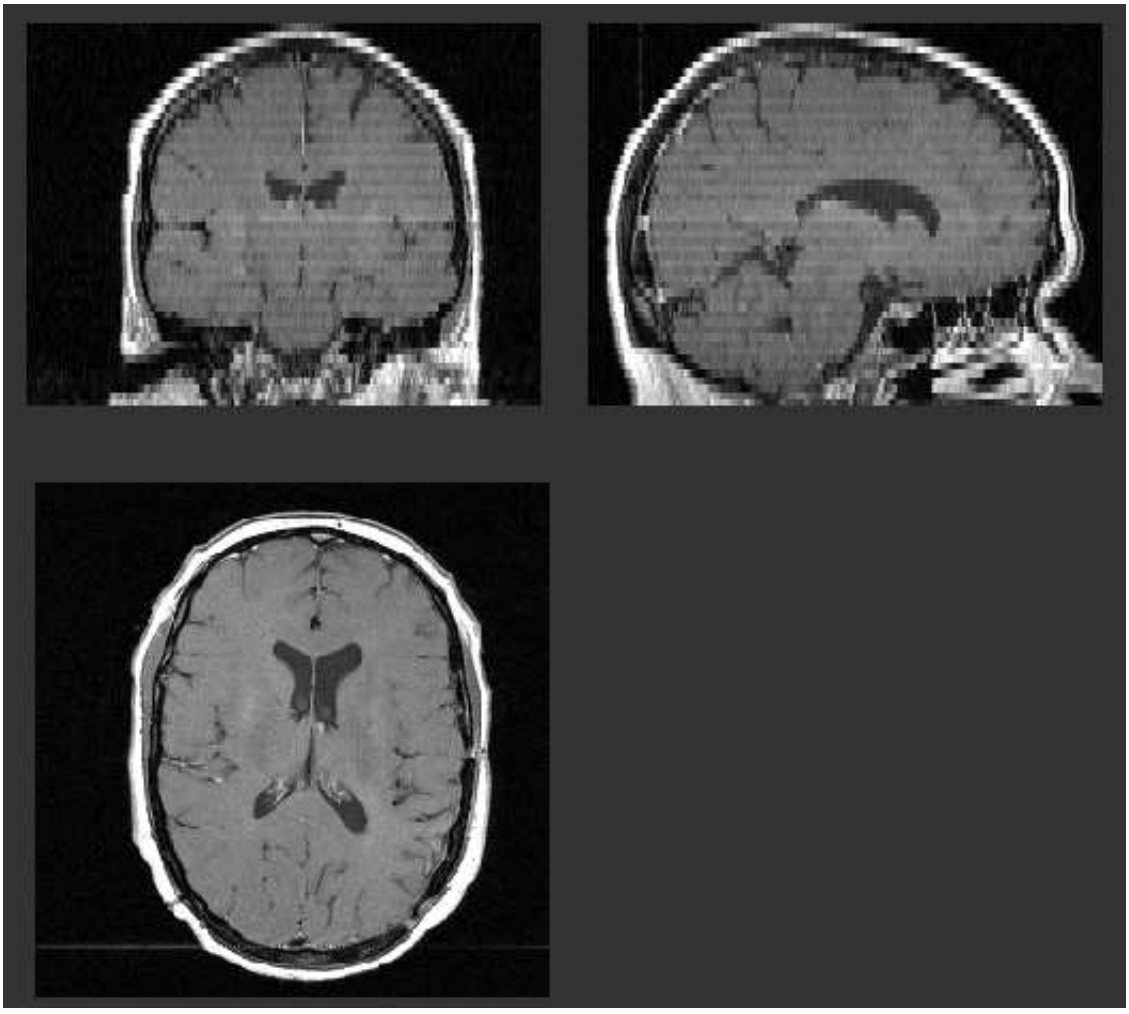

Figure 9: T1 weighted volume

We applied simple thresholding on the CT image to determine a skull mask, and then compared it to the skull mask

obtained with *betsurf* (after registering the CT volume to the T1 using FLIRT). We then measured mask overlap, ignoring voxels below the anterior-inferior plane described above. The results are:

- 90% of the *betsurf* mask voxels were inside the CT mask.
- 82% of the CT mask voxels were inside the *betsurf* mask .
- The Dice coefficient between the two masks (a standard overlap measure, $2\sharp(S1 \cap S2)/(\sharp S1 + \sharp S2)$) was 0.85.

The boundaries of the skull computed with *betsurf*, overlaid onto the registered CT are shown in Figure 10.

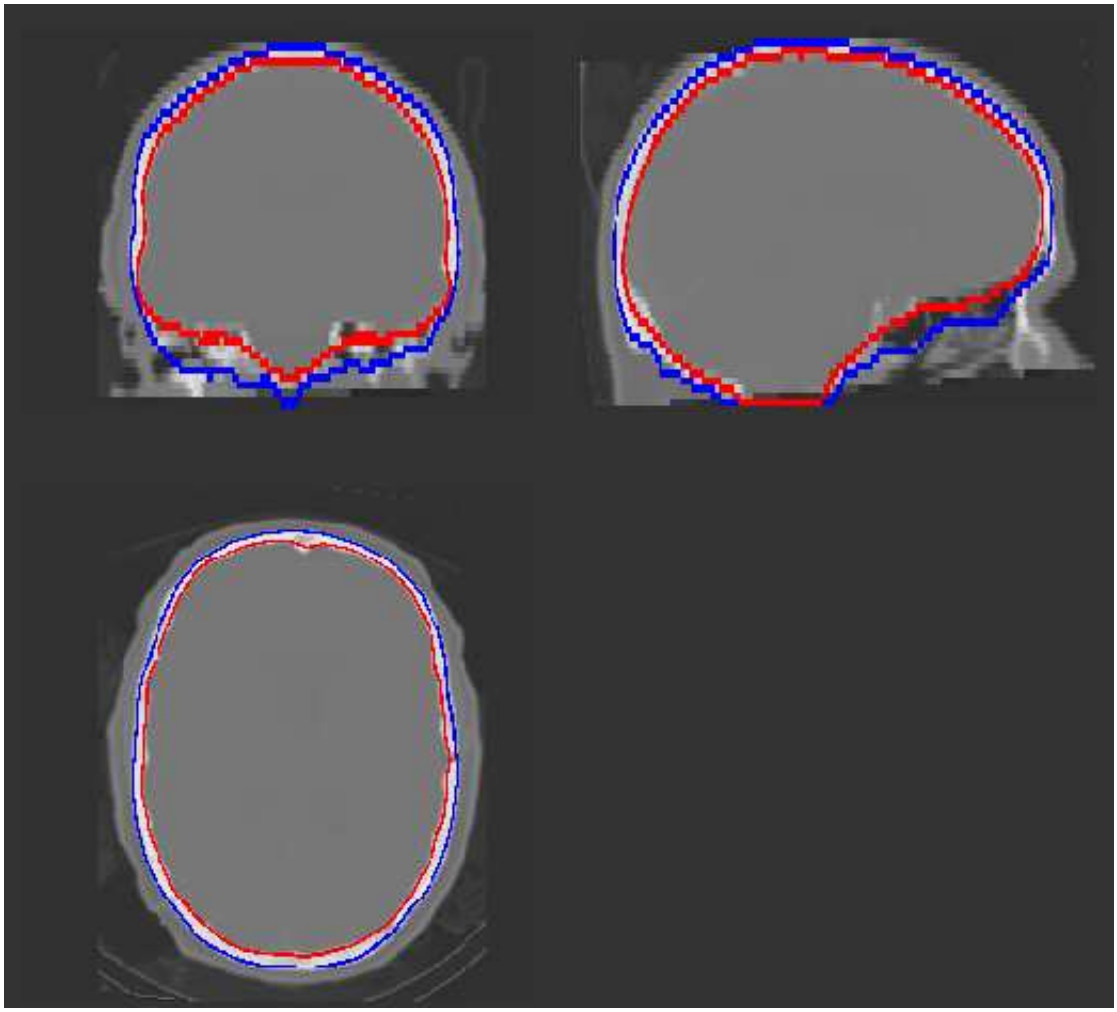

Figure 10: meshes from *betsurf* outlined on CT volume

3.2 Head image with soft tissue

The most difficult cases are when there are a lot of soft tissues in the T1 volume - an example is shown in Figure 11.

In Figure 12, the meshes found by *betsurf* are shown overlaid onto the T1 volume, showing robustness and accuracy in the presence of the soft tissue areas.

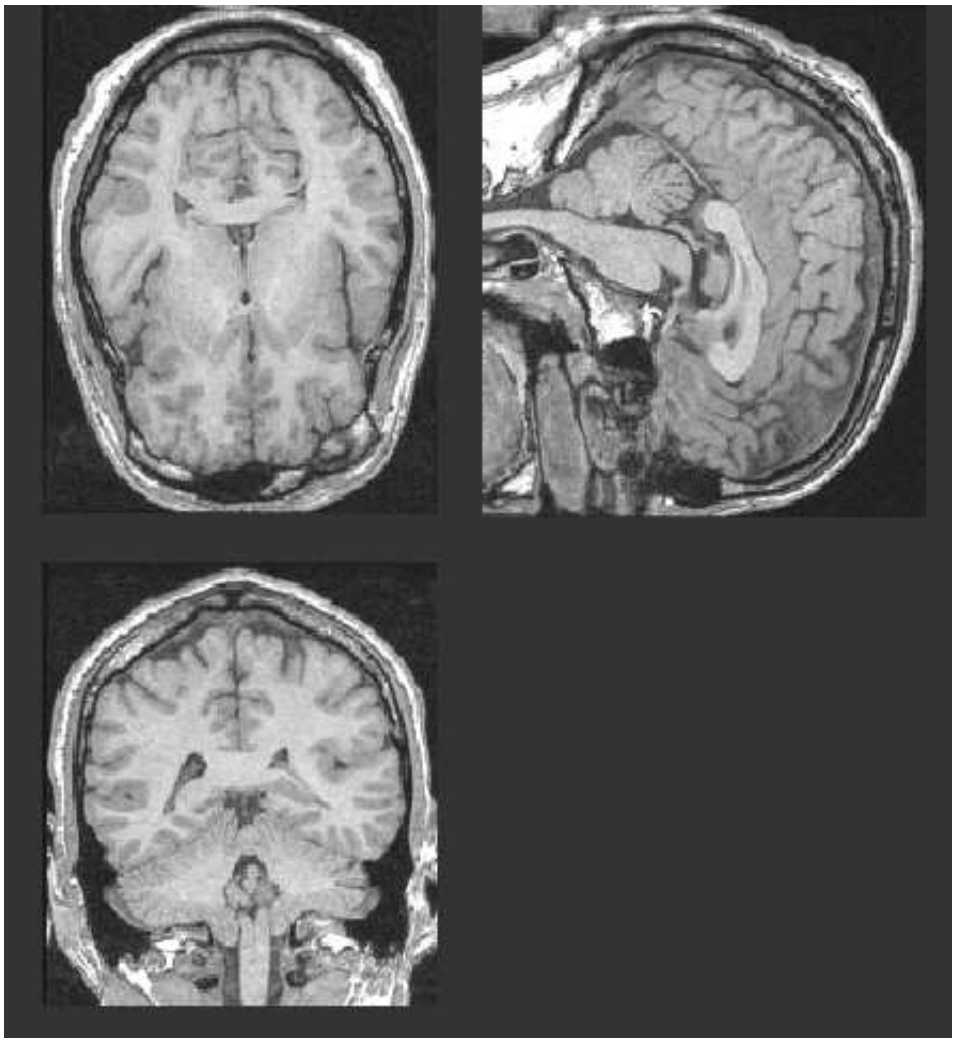

Figure 11: T1-weighted volume

A Appendix - Efficient Self-Intersection Algorithm

Testing mesh self-intersection in a computationally efficient way is not a simple problem. A naive approach would be to browse all pairs of triangles and test for intersection. Since our meshes usually have more than 5000 triangles, this is considered impractical. There are methods from algorithmic geometry in 2D to test exact self-intersection of a set of segments in $\bigcirc(Nln(N))$ ("sweep line" algorithms like the Bentley-Ottmann algorithm [Bentley and Ottmann, 1979, Cormen and Leiserson, 1992]), but to our knowledge there is not an equivalent for 3D. We adapted one such 2D method, leading to an algorithm in $\bigcirc(N\sqrt{N})$ time.

A.1 Bentley-Ottmann algorithm in 2D

This algorithm checks the self intersection of a set of segments in $\bigcirc(Nln(N))$. It first sorts all the points according to their location in X ($\bigcirc(Nln(N))$). Next it browses all the points in horizontal order (hence the name of "sweep line" algorithm - it can be visualized as a vertical line "sweeping" all the points), putting the current segments in a structure like a bicoloured-tree, ordered using the Y location of the segments:

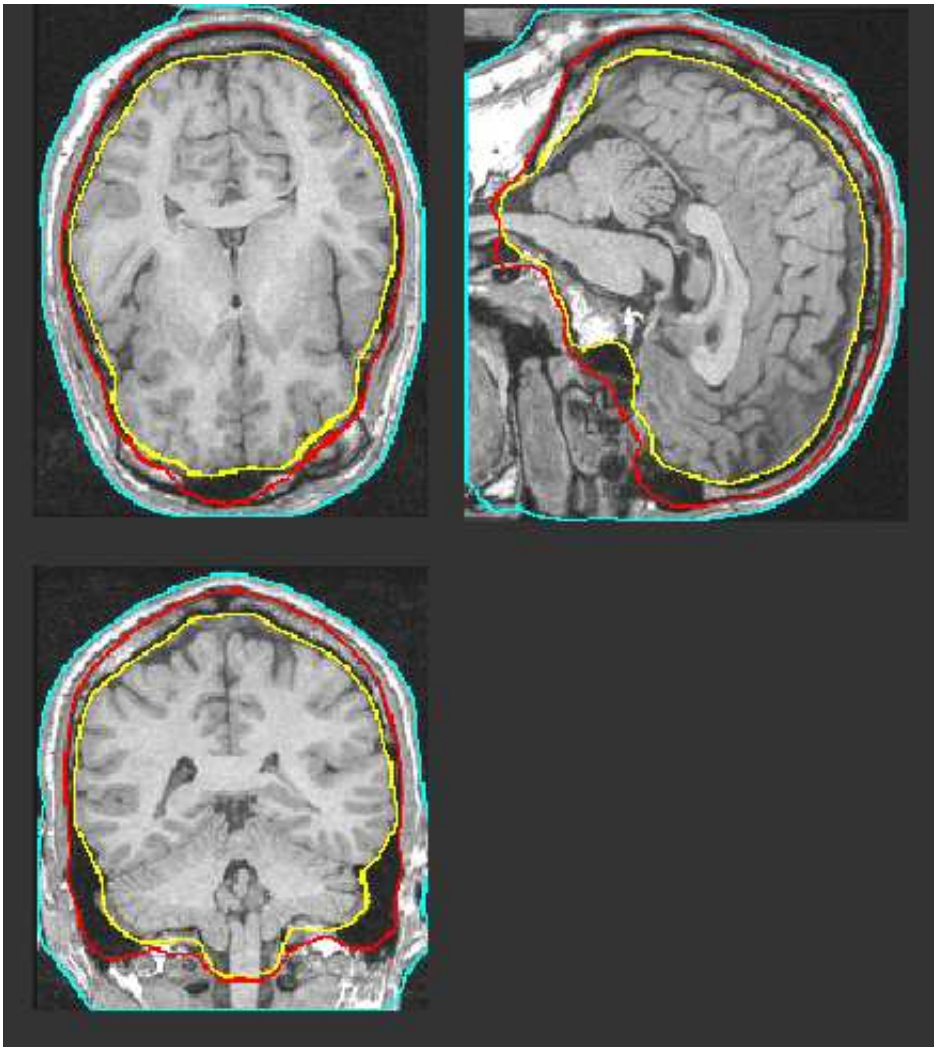

Figure 12: T1 with outlined surfaces

- If the point is the first of the segment, then it puts the segment in the right position in the tree ($\bigcirc ln(N)$ - in fact $\bigcirc ln(N')$, where N' is the number of segments in the tree) and checks it is not intersecting with its neighbours.
- If the point is the last of the segment, it removes the segment from the tree.

A.2 Adaptation to 3D

The natural way to transpose this algorithm into 3D is to use a "sweep plane" instead of a sweeping line. But we then have to check intersections between triangles belonging to the same plane, and there is no straightforward way to order the triangles, and thus no way to put them in a structure with a $(ln(N))$ access. However, since the number of triangles on a given plane is $\bigcirc(\sqrt{N})$, this leads to a $\bigcirc(N\sqrt{N})$ algorithm:

- We sort all the mesh points according to X and declare a list of triangles.
- We browse all those points in order. For each point, we browse the adjacent triangles:

- If the triangle was reached for the first time, we test its self-intersection with the triangles present in the list (We have a simple algorithm to test intersection between triangles that checks if the pair of vertices of a triangle are on different sides of the plane made by the other triangle, and then projects one of the vertices onto this plane and checks if it is inside or outside the triangle).
- If the triangle was reached for the second time, we do nothing.
- If the triangle was reached for the third time, we remove the triangle from the list.

A.3 Results

For a mesh with 5120 triangles, this leads to calculated triangle intersections for approximately 7×10^5 pairs of triangles instead of 13×10^6 with a naive algorithm. Along with other improvements (such as pre-computing the centres of the triangles, the maximum distance from a vertex to the centre and using this information to reject pairs of triangles), the algorithm is 30 times faster than the naive one (350ms vs 10s for a 5120 triangle mesh).

References

[Bentley and Ottmann, 1979] Bentley, J. and Ottmann, T. (1979). Algorithms for reporting and counting geometric intersections. *IEEE Trans. Comput.*, C-28:643–647.

[Cormen and Leiserson, 1992] Cormen, T. and Leiserson, C. an Rivest, R. (1992). *Introduction to Algorithms*. MIT Press.

[Jenkinson et al., 2002] Jenkinson, M., Bannister, P., Brady, J., and Smith, S. (2002). Improved optimisation for the robust and accurate linear registration and motion correction of brain images. *NeuroImage*, 17(2):825–841.

[Jenkinson and Smith, 2001] Jenkinson, M. and Smith, S. (2001). A global optimisation method for robust affine registration of brain images. *Medical Image Analysis*, 5(2):143–156.

[Smith, 2002] Smith, S. (2002). Fast robust automated brain extraction. *Human Brain Mapping*, 17(3):143–155.