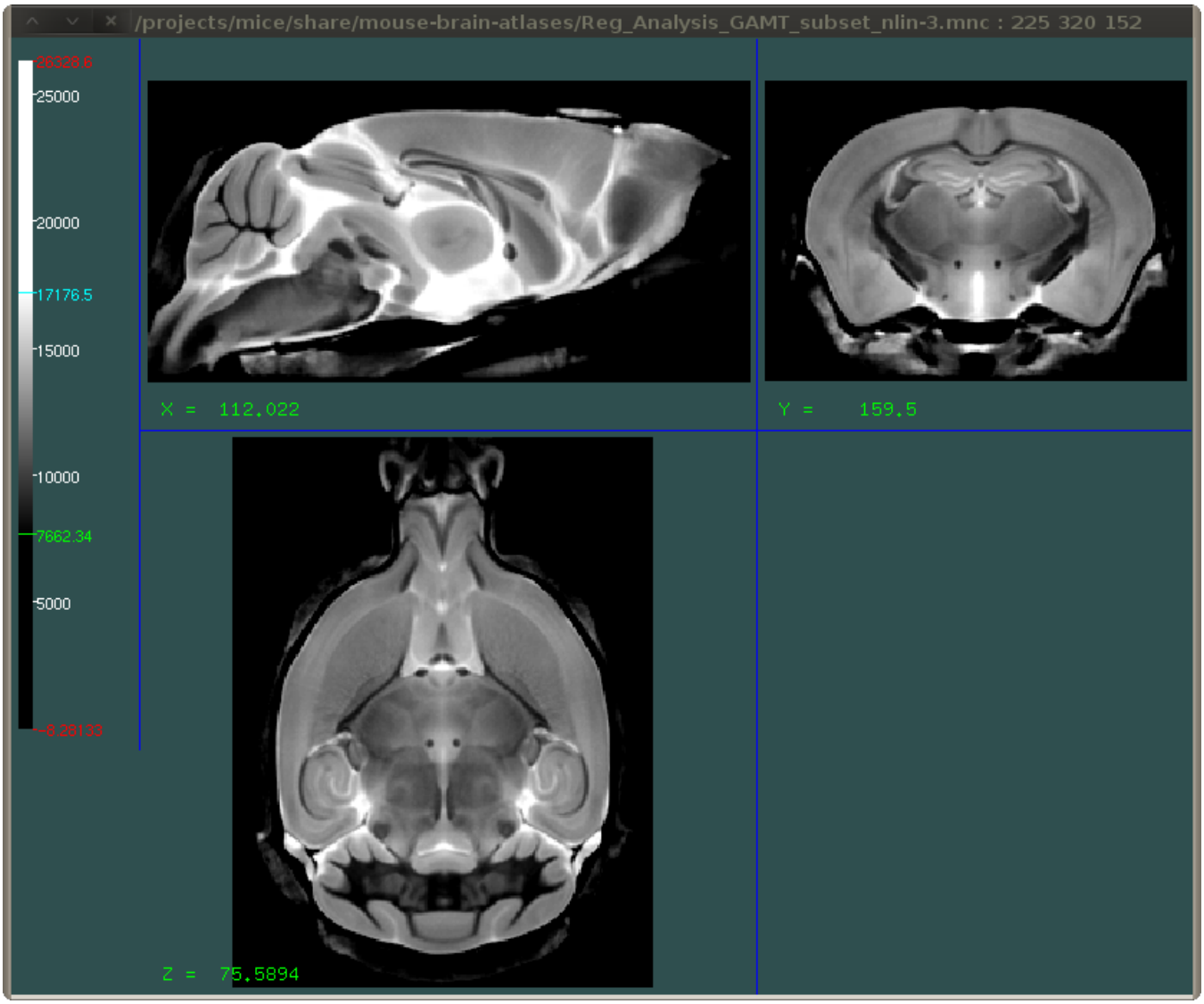
Creating a symmetric target for registration

Purpose

When you run a registration using MICe-build-model often you will use an initial model. The files in your initial model will determine the sizes and dimensions of your output files, as well as the viewing orientation. The models we normally use will show the different slices of the brain orthogonal to the cardinal directions. However, depending on what your initial model looks like, the resulting images of your registration might not be in that same viewing orientation. Using an initial model that has symmetric targets in it, will ensure that your results will also be orthogonal to the cardinal directions.

Standard viewing orientation

At MICe a file that is in standard viewing orientation will look as follows when you open it up in Display:



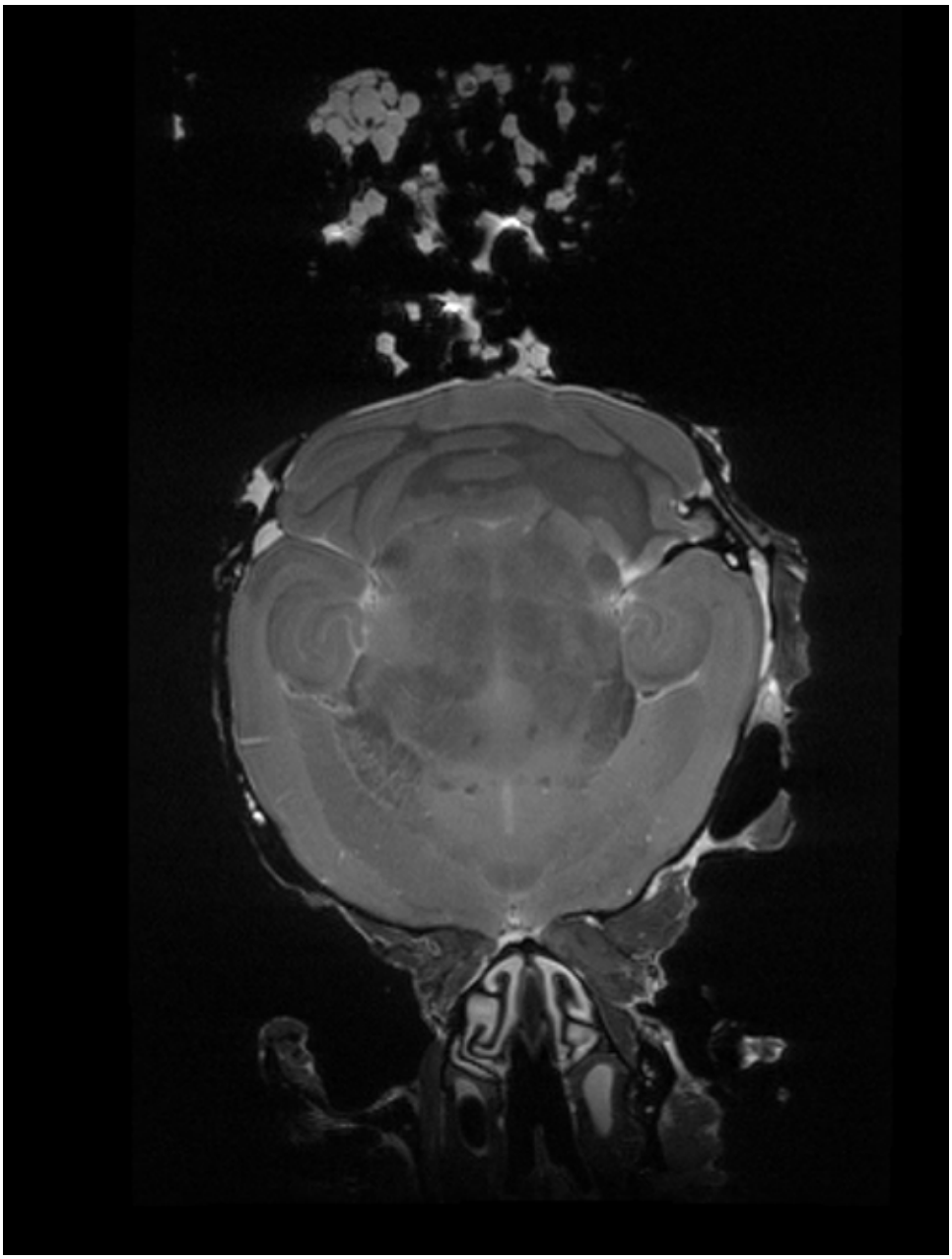
Symmetric image

To create a symmetric image you will have to create a copy of the image that is flipped along one of the axes and then overlay the two images. The following example will show you how to do this.

> ⚠ This example is done using a single image. Normally for an initial model you would use an average of a whole number of images. For more information about what kind of files you should use for your initial model, take a look here

1. Your input image:

```
> mincinfo mouse_brain.mnc
file: mouse_brain.mnc
image: unsigned short 0 to 65535
image dimensions: zspace xspace yspace
 dimension name length step start
 -------------- ------ ---- -----
 zspace 290 0.056 40.908
 xspace 292 -0.055556 -14.4167
 yspace 450 -0.055556 12.4722
```



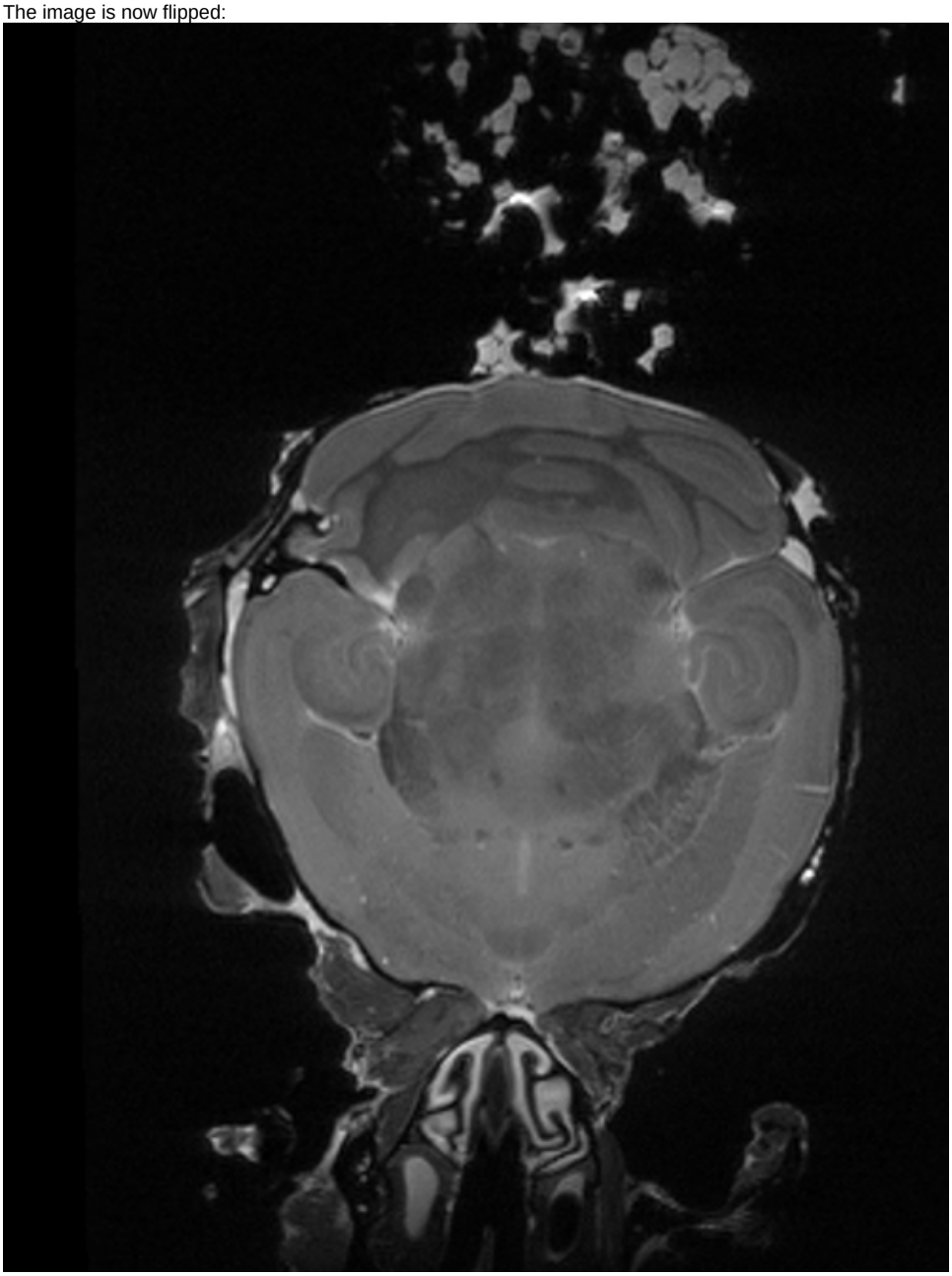
2. We want to flip this image along the x-axis (the axis from left to right). In order to do that we will simply change the sign of the step size in that direction:

```
> mincheader mouse_brain.mnc | grep xspace
 xspace = 292 ;
 short image(zspace, xspace, yspace) ;
 image:dimorder = "zspace,xspace,yspace" ;
 int xspace ;
 xspace:length = 292 ;
 xspace:varid = "MINC standard variable" ;
 xspace:vartype = "dimension____" ;
 xspace:version = "MINC Version 1.0" ;
 xspace:comments = "X increases from patient left to right" ;
 xspace:spacing = "regular__" ;
 xspace:alignment = "centre" ;
 xspace:step = -0.055556 ;
 xspace:start = -14.416658 ;
 xspace:direction_cosines = 1., 0., 0. ;
 xspace:units = "mm" ;
 xspace:spacetype = "native____" ;
 xspace = 0 ;
```

Using mincheader we see that the full step size of the xspace is -0.055556

```
> cp mouse_brain.mnc mouse_brain_flipped_in_x.mnc
{change the sign of the step size in x}
> minc_modify_header -dinsert xspace:step=0.055556 mouse_brain_flipped_in_x.mnc
```

The image is now flipped:



3. So the image is flipped. However, the original image started along the x-axis in -14.416658 and was 292 voxels long, so ended in -30.639008 (start point + stepsize * length = -14.416658 +(-0.05556 * 292)) . The flipped image as it is now, starts at the same point, but because it has a positive step size in x, it will end in 1.805692. This means that the images do not overlap in space anymore. We can account for this by changing the start point of our new image to -30.639008.

```
> minc_modify_header -dinsert xspace:start=-30.639008 mouse_brain_flipped_in_x.mnc
```

4. Now we are ready to overlay the two images. Well... one last thing. mincaverage does not like it when two files have different start coordinates. But we know these files are in the same location in space, so we can use a mincresample command to fix that issue:

```
> mincresample -like mouse_brain.mnc -use_input mouse_brain_flipped_in_x.mnc
mouse_brain_flipped_in_x_like_original.mnc
> mincaverage mouse_brain.mnc mouse_brain_flipped_in_x_like_original.mnc mouse_brain_symmetric.mnc
```

And the final image now is symmetric: