# Recipe to register embryos

For the registration of embryo data, we use slightly different settings for MICe-build-model compared to the registration of brain data. So far we have not been able to determine a workable set of parameters for mincANTS, and so we use minctracc. To allow for larger deformations (embryo data tends to be more dissimilar) a non-linear stage with a larger blurring kernel is added at the start of the non-linear stage, and the stiffness settings for the elastic grid for minctracc are set more loosely.

For input data at a resolution of 15micron, the following is a sample incantation of the MICe-build-model.pl script :



Assumptions:

- the embryo data is either OPT or CT (because we use the -no-nuc option, meaning that we do not do non-uniformity correction caused by the MR)
- the data is already roughly aligned, i.e., they overlap in space and face the same direction. If this is not the case,the -lsq6-identity flag should be changed

```
> MICe-build-model.pl
-pipeline-name some_embryo_pipeline
-init-model /some/initial/model/prefix
-no-inormalize
-no-resample-atlas
-no-registration-accuracy
-no-nuc
-lsq6-initial-step 0.1
-lsq6-identity
-no-lsq6-large-rotations
-lsq6-kernels 1.0_blur,0.8_blur,0.5_blur
-lsq12
-lsq12-initial-step 0.1
-lsq12-kernels 2.0_blur,1.0_blur
-lsq12-max-pairs 0
-nlin
-nlin-stats
-nlin-registration-method minctracc
-nlin-protocol embryo_smasher.pl
-stiffness 0.5
-similarity 0.5
-weight 0.3
[ input embryo MINC files ]
```

where the file embryo_smasher.pl contains:

```
> cat embryo_smasher.pl
 (
 { memory => 4,
 generation => 1,
 kernel => 0.6,
 iterations => 40,
 simplex => 3,
 ncpus => 1,
 use_gradient => 0,
 optimization => "-use_simplex",
 step => 0.3,
 },
 { memory => 4,
 generation => 2,
 kernel => 0.4,
 iterations => 40,
 simplex => 3,
 ncpus => 1,
 use_gradient => 0,
 optimization => "-use_simplex",
 step => 0.3,
 },
 { memory => 4,
```

```
 generation => 3,
 kernel => 0.2,
 iterations => 20,
 simplex => 3,
 ncpus => 1,
 use_gradient => 0,
 optimization => "-use_simplex",
 step => 0.2,
 },
 { memory => 4,
 generation => 4,
 kernel => 0.15,
 iterations => 20,
 simplex => 3,
 ncpus => 1,
 use_gradient => 0,
 optimization => "-use_simplex",
 step => 0.15,
 },
 { memory => 4,
 generation => 5,
 kernel => 0.09,
 iterations => 10,
 simplex => 3,
 ncpus => 1,
 use_gradient => 0,
 optimization => "-use_simplex",
 step => 0.09,
 },
{ memory => 4,
 generation => 6,
 kernel => 0.05,
 iterations => 10,
 simplex => 3,
 ncpus => 1,
 use_gradient => 0,
 optimization => "-use_simplex",
 step => 0.05,
 },
{ memory => 4,
 generation => 7,
 kernel => 0.03,
 iterations => 10,
 simplex => 3,
 ncpus => 1,
 use_gradient => 0,
 optimization => "-use_simplex",
 step => 0.03,
 },


 );
```