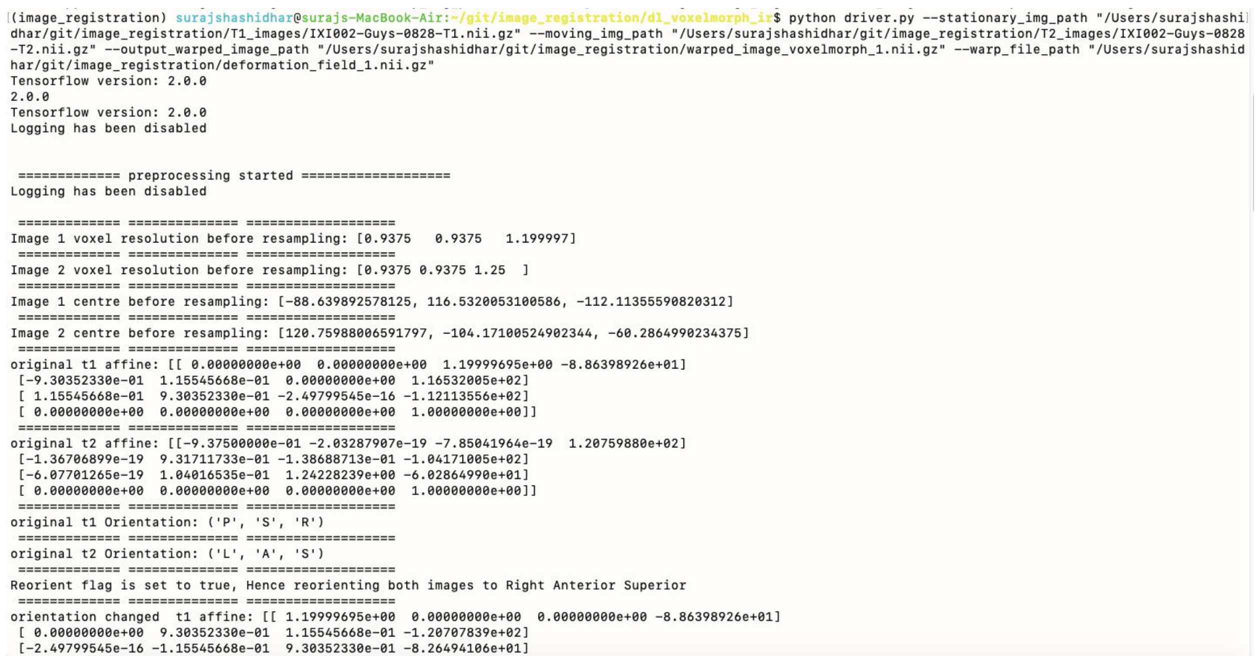
About Voxelmorph

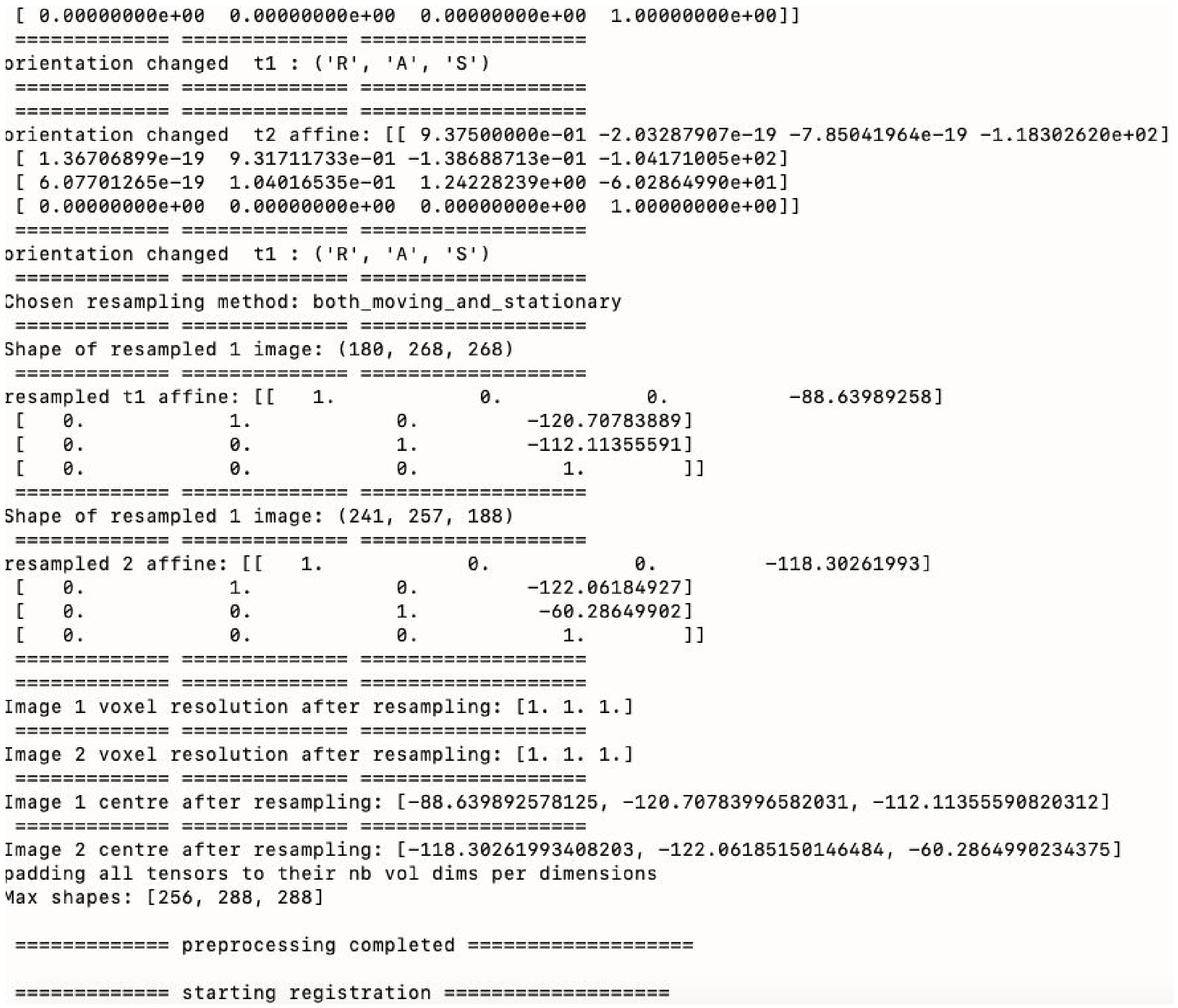
Voxelmorph is a self supervised registration technique and not a framework/library. There can be supervision and also probabilistic approach too added to voxelmorph as different flavours to enhance quality.
Voxelmorph tries to learn the deformation field through neural network rather than directly like airlabs/ANTS/FSL, this leads to amortized inference and much better generalization and representation of deformations.
But the learned parameters can provide good deformations if it has seen same/similar data before. Lets say Voxelmorph is only trained on brain data and if suddenly we give it data of chest, there is no guarantee that it could generalize as it had not seen this kind of data before.
The mapping is learned from a spatial transformer

Run process: Command line call looks like below with run screenshots

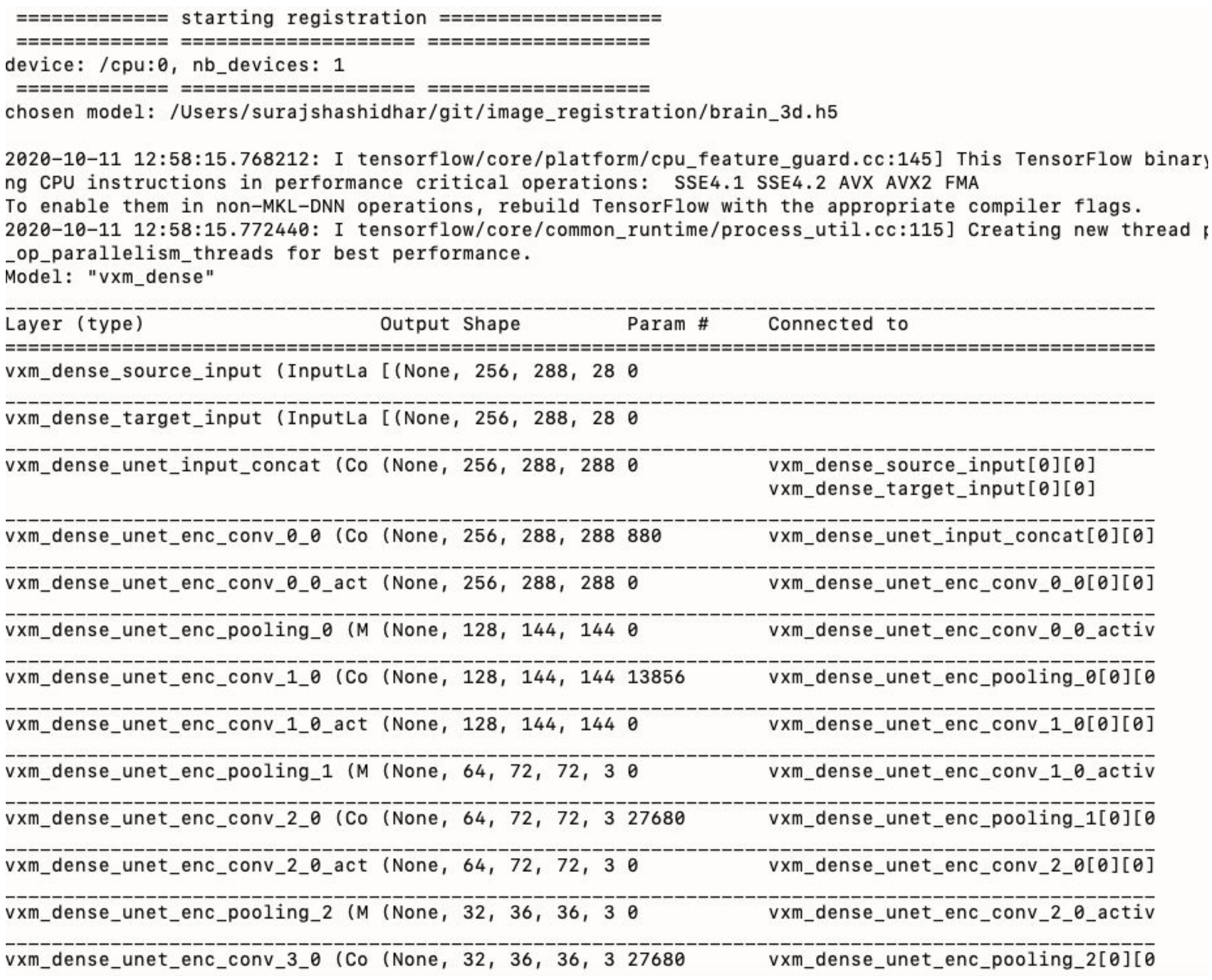
```
python driver.py --stationary_img_path
"/Users/surajshashidhar/git/image_registration/T1_images/IXI020-Guys-0700-T1.nii.gz"
--moving_img_path
"/Users/surajshashidhar/git/image_registration/T1_images/IXI021-Guys-0703-T1.nii.gz"
--output_warped_image_path
"/Users/surajshashidhar/git/image_registration/voxelmorph_warped_image.nii.gz"
--reorient_flag "False" --loss_fnc "MSE"
```



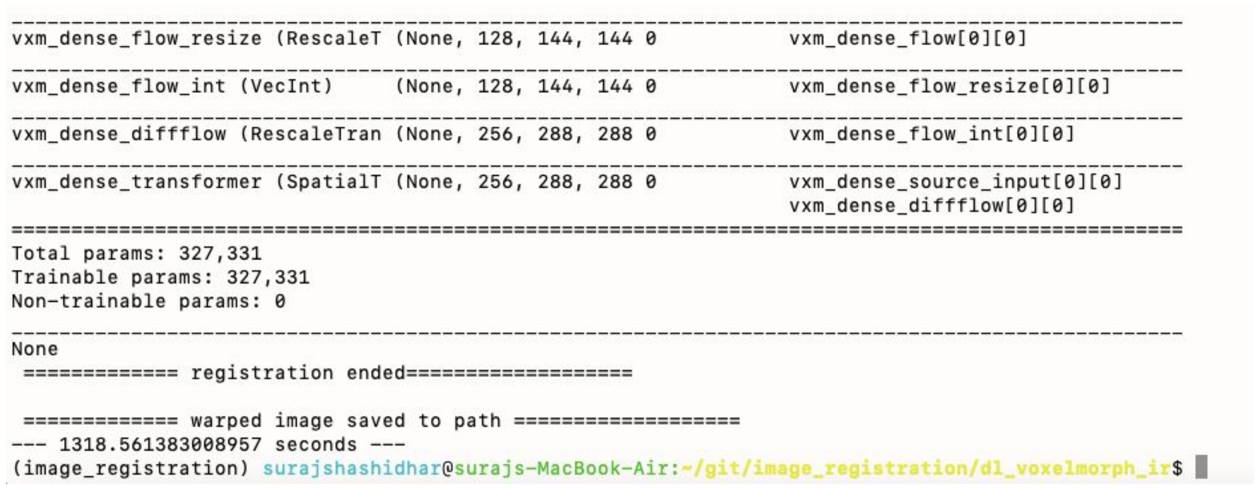
```
(image_registration) surajshashidhar@surajs-MacBook-Air:~/git/image_registration/dl_voxelmorph_ir$ python driver.py --stationary_img_path "/Users/surajshashi
dhar/git/image_registration/T1_images/IXI002-Guys-0828-T1.nii.gz" --moving_img_path "/Users/surajshashidhar/git/image_registration/T2_images/IXI002-Guys-0828
-T2.nii.gz" --output_warped_image_path "/Users/surajshashidhar/git/image_registration/warped_image_voxelmorph_1.nii.gz" --warp_file_path "/Users/surajshashid
har/git/image_registration/deformation_field_1.nii.gz"
Tensorflow version: 2.0.0
2.0.0
Tensorflow version: 2.0.0
Logging has been disabled


 ============= preprocessing started ===================
Logging has been disabled

 ============= ============== ===================
Image 1 voxel resolution before resampling: [0.9375   0.9375   1.199997]
 ============= ============== ===================
Image 2 voxel resolution before resampling: [0.9375 0.9375 1.25  ]
 ============= ============== ===================
Image 1 centre before resampling: [-88.639892578125, 116.5320053100586, -112.11355590820312]
 ============= ============== ===================
Image 2 centre before resampling: [120.75988006591797, -104.17100524902344, -60.2864990234375]
 ============= ============== ===================
original t1 affine: [[ 0.00000000e+00  0.00000000e+00  1.19999695e+00 -8.86398926e+01]
 [-9.30352330e-01  1.15545668e-01  0.00000000e+00  1.16532005e+02]
 [ 1.15545668e-01  9.30352330e-01 -2.49799545e-16 -1.12113556e+02]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  1.00000000e+00]]
 ============= ============== ===================
original t2 affine: [[-9.37500000e-01 -2.03287907e-19 -7.85041964e-19  1.20759880e+02]
 [-1.36706899e-19  9.31711733e-01 -1.38688713e-01 -1.04171005e+02]
 [-6.07701265e-19  1.04016535e-01  1.24228239e+00 -6.02864990e+01]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  1.00000000e+00]]
 ============= ============== ===================
original t1 Orientation: ('P', 'S', 'R')
 ============= ============== ===================
original t2 Orientation: ('L', 'A', 'S')
 ============= ============== ===================
Reorient flag is set to true, Hence reorienting both images to Right Anterior Superior
 ============= ============== ===================
orientation changed  t1 affine: [[ 1.19999695e+00  0.00000000e+00  0.00000000e+00 -8.86398926e+01]
 [ 0.00000000e+00  9.30352330e-01  1.15545668e-01 -1.20707839e+02]
 [-2.49799545e-16 -1.15545668e-01  9.30352330e-01 -8.26494106e+01]
```



```
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  1.00000000e+00]]
 ============= ============== ===================
orientation changed  t1 : ('R', 'A', 'S')
 ============= ============== ===================
 ============= ============== ===================
orientation changed  t2 affine: [[ 9.37500000e-01 -2.03287907e-19 -7.85041964e-19 -1.18302620e+02]
 [ 1.36706899e-19  9.31711733e-01 -1.38688713e-01 -1.04171005e+02]
 [ 6.07701265e-19  1.04016535e-01  1.24228239e+00 -6.02864990e+01]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  1.00000000e+00]]
 ============= ============== ===================
orientation changed  t1 : ('R', 'A', 'S')
 ============= ============== ===================
Chosen resampling method: both_moving_and_stationary
 ============= ============== ===================
Shape of resampled 1 image: (180, 268, 268)
 ============= ============== ===================
resampled t1 affine: [[   1.            0.            0.          -88.63989258]
 [   0.            1.            0.         -120.70783889]
 [   0.            0.            1.         -112.11355591]
 [   0.            0.            0.            1.        ]]
 ============= ============== ===================
Shape of resampled 1 image: (241, 257, 188)
 ============= ============== ===================
resampled 2 affine: [[   1.            0.            0.         -118.30261993]
 [   0.            1.            0.         -122.06184927]
 [   0.            0.            1.          -60.28649902]
 [   0.            0.            0.            1.        ]]
 ============= ============== ===================
 ============= ============== ===================
Image 1 voxel resolution after resampling: [1. 1. 1.]
 ============= ============== ===================
Image 2 voxel resolution after resampling: [1. 1. 1.]
 ============= ============== ===================
Image 1 centre after resampling: [-88.639892578125, -120.70783996582031, -112.11355590820312]
 ============= ============== ===================
Image 2 centre after resampling: [-118.30261993408203, -122.06185150146484, -60.2864990234375]
padding all tensors to their nb vol dims per dimensions
Max shapes: [256, 288, 288]

 ============= preprocessing completed ===================

 ============= starting registration ===================
```

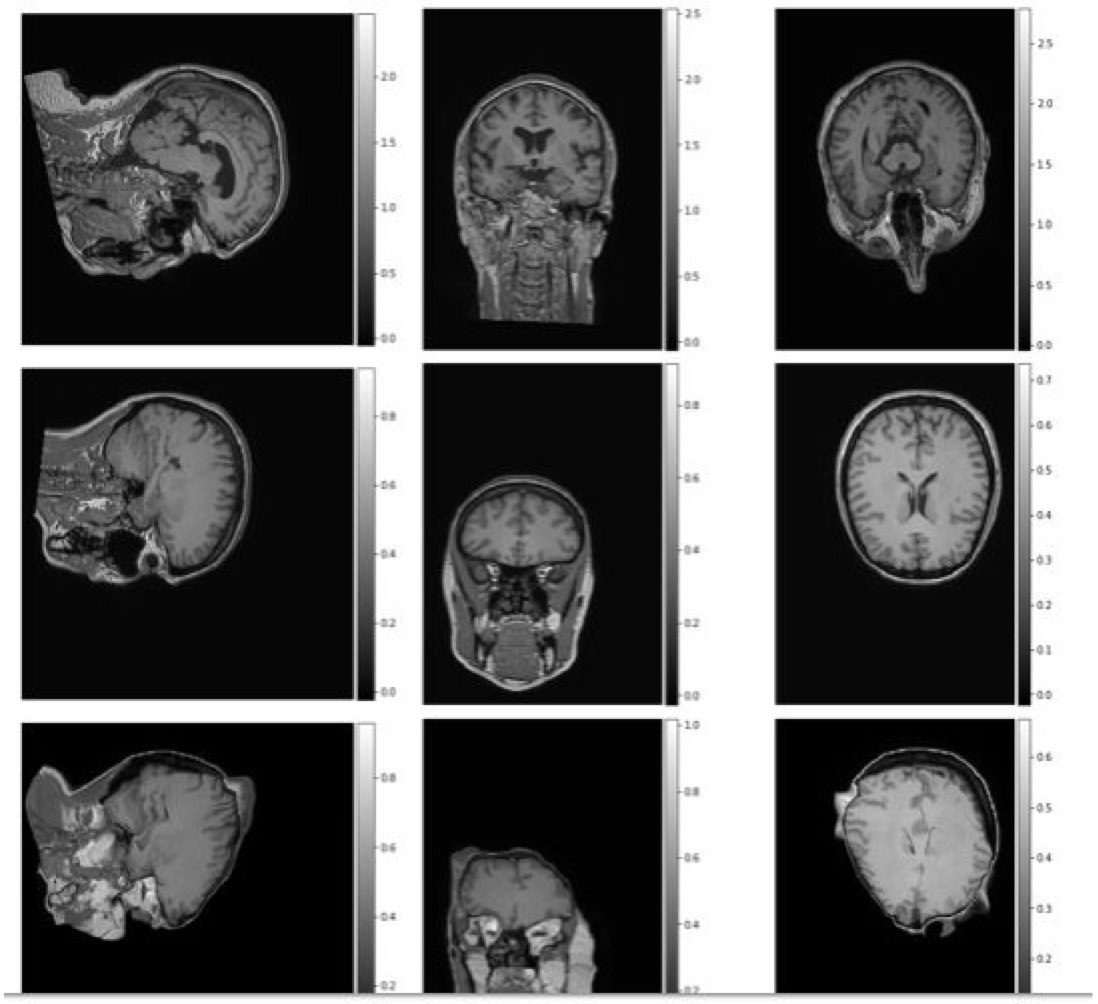


```
 ============= starting registration ===================
 ============= ==================== ===================
device: /cpu:0, nb_devices: 1
 ============= ==================== ===================
chosen model: /Users/surajshashidhar/git/image_registration/brain_3d.h5

2020-10-11 12:58:15.768212: I tensorflow/core/platform/cpu_feature_guard.cc:145] This TensorFlow binar
ng CPU instructions in performance critical operations:  SSE4.1 SSE4.2 AVX AVX2 FMA
To enable them in non-MKL-DNN operations, rebuild TensorFlow with the appropriate compiler flags.
2020-10-11 12:58:15.772440: I tensorflow/core/common_runtime/process_util.cc:115] Creating new thread 
_op_parallelism_threads for best performance.
Model: "vxm_dense"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
vxm_dense_source_input (InputLa [(None, 256, 288, 28 0
__________________________________________________________________________________________________
vxm_dense_target_input (InputLa [(None, 256, 288, 28 0
__________________________________________________________________________________________________
vxm_dense_unet_input_concat (Co (None, 256, 288, 288 0           vxm_dense_source_input[0][0]
                                                                 vxm_dense_target_input[0][0]
__________________________________________________________________________________________________
vxm_dense_unet_enc_conv_0_0 (Co (None, 256, 288, 288 880         vxm_dense_unet_input_concat[0][0]
__________________________________________________________________________________________________
vxm_dense_unet_enc_conv_0_0_act (None, 256, 288, 288 0           vxm_dense_unet_enc_conv_0_0[0][0]
__________________________________________________________________________________________________
vxm_dense_unet_enc_pooling_0 (M (None, 128, 144, 144 0           vxm_dense_unet_enc_conv_0_0_activ
__________________________________________________________________________________________________
vxm_dense_unet_enc_conv_1_0 (Co (None, 128, 144, 144 13856       vxm_dense_unet_enc_pooling_0[0][0
__________________________________________________________________________________________________
vxm_dense_unet_enc_conv_1_0_act (None, 128, 144, 144 0           vxm_dense_unet_enc_conv_1_0[0][0]
__________________________________________________________________________________________________
vxm_dense_unet_enc_pooling_1 (M (None, 64, 72, 72, 3 0           vxm_dense_unet_enc_conv_1_0_activ
__________________________________________________________________________________________________
vxm_dense_unet_enc_conv_2_0 (Co (None, 64, 72, 72, 3 27680       vxm_dense_unet_enc_pooling_1[0][0
__________________________________________________________________________________________________
vxm_dense_unet_enc_conv_2_0_act (None, 64, 72, 72, 3 0           vxm_dense_unet_enc_conv_2_0[0][0]
__________________________________________________________________________________________________
vxm_dense_unet_enc_pooling_2 (M (None, 32, 36, 36, 3 0           vxm_dense_unet_enc_conv_2_0_activ
__________________________________________________________________________________________________
vxm_dense_unet_enc_conv_3_0 (Co (None, 32, 36, 36, 3 27680       vxm_dense_unet_enc_pooling_2[0][0
```



```
__________________________________________________________________________________________________
vxm_dense_flow_resize (RescaleT (None, 128, 144, 144 0           vxm_dense_flow[0][0]
__________________________________________________________________________________________________
vxm_dense_flow_int (VecInt)     (None, 128, 144, 144 0           vxm_dense_flow_resize[0][0]
__________________________________________________________________________________________________
vxm_dense_diffflow (RescaleTran (None, 256, 288, 288 0           vxm_dense_flow_int[0][0]
__________________________________________________________________________________________________
vxm_dense_transformer (SpatialT (None, 256, 288, 288 0           vxm_dense_source_input[0][0]
                                                                 vxm_dense_diffflow[0][0]
==================================================================================================
Total params: 327,331
Trainable params: 327,331
Non-trainable params: 0
__________________________________________________________________________________________________
None
 ============= registration ended===================

 ============= warped image saved to path ===================
--- 1318.561383008957 seconds ---
(image_registration) surajshashidhar@surajs-MacBook-Air:~/git/image_registration/dl_voxelmorph_ir$ 
```

20 minutes for registration in the local CPU. On colab using 6GB CPU registration under 10 min



About Code

Code has been packaged into 4 pieces so that preprocessing pipeline is somewhat configurable. We have used pretrained model of Unet+spatial transformer provided by Voxelmorph team.

Image Registration

Image registration activities are put into img_registration.py

Image registration consists of registration type, optimizers and loss functions and just does image registration. Currently we have both 3-D affine registration and deformable registration. As expected deformable registration works a bit better than rigid. Loss functions are configurable hence we can use them for both intra and inter modal registration.

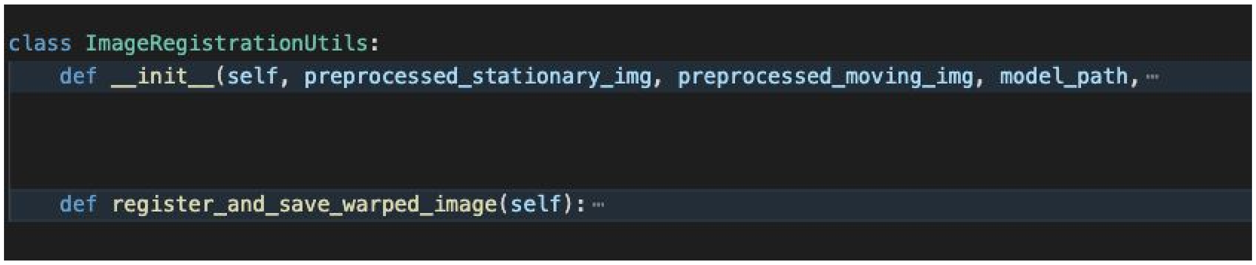
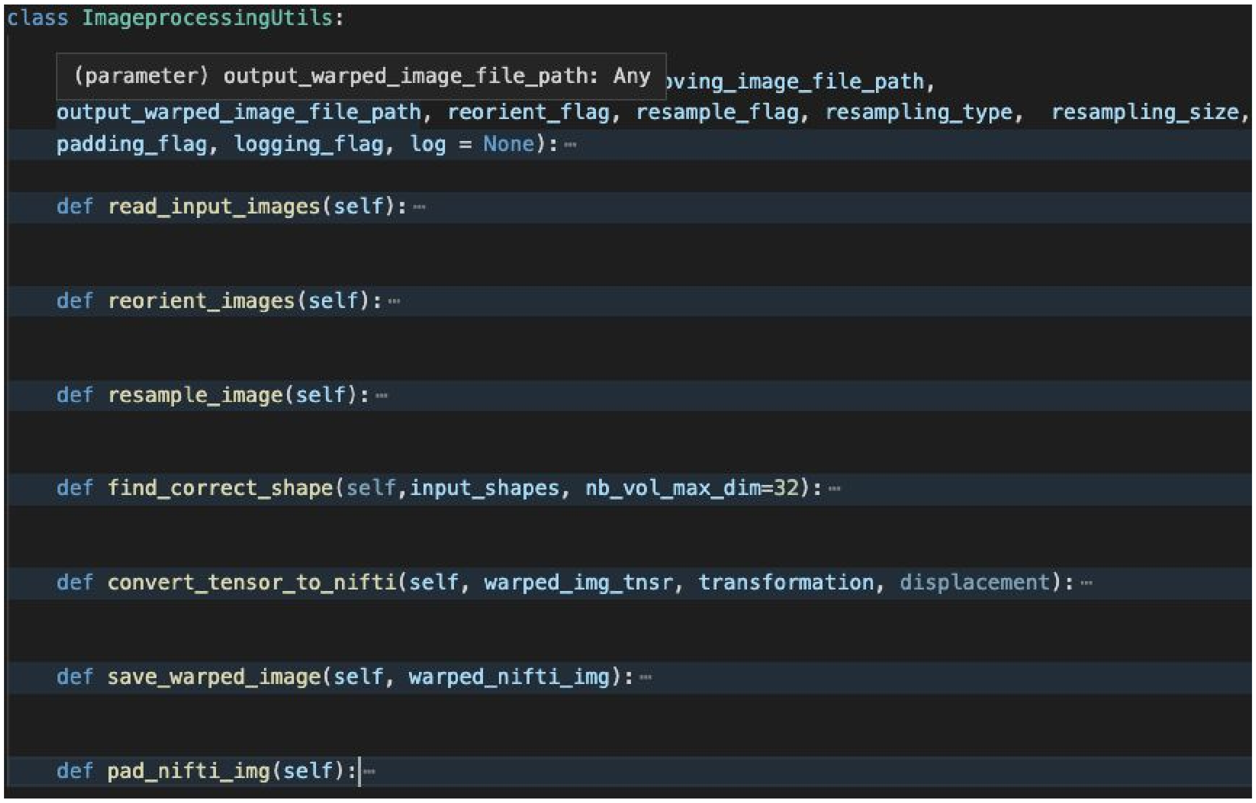

```
class ImageRegistrationUtils:
    def __init__(self, preprocessed_stationary_img, preprocessed_moving_img, model_path,⋯



    def register_and_save_warped_image(self):⋯
```

Image Processing

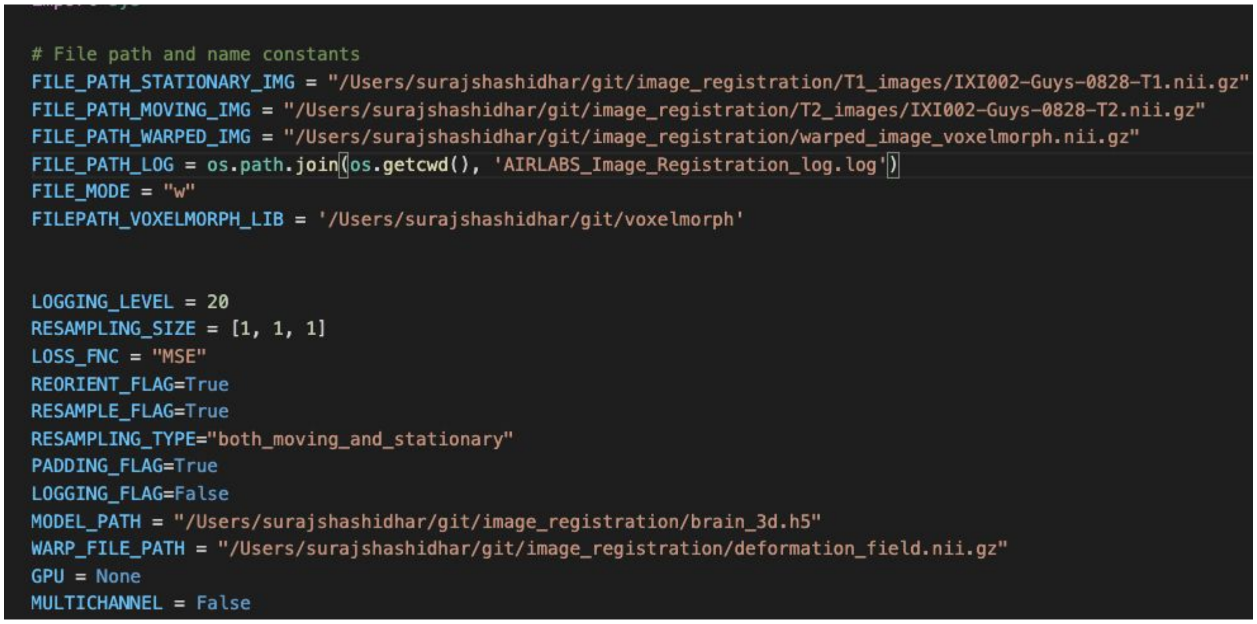
File processing and preprocessing activities are put into img_processing.py file, more preprocessing can be added here without much changes to other files.

Current preprocessing pipeline consists of reading nifti files, reorienting images to standard Right Anterior Superior (RAS) orientation, resampling images to 1mm isotropic or resampling moving image to match that of stationary image, padding to get images of same dimension, saving warped image into nifti files. The name of the methods reflect the above steps. More preprocessing can be added if required. Each step of preprocessing can be turned on/off by a flag and some parameters such as resampling voxel sizes can be changed.


```
class ImageprocessingUtils:

    (parameter) output_warped_image_file_path: Any  oving_image_file_path,
    output_warped_image_file_path, reorient_flag, resample_flag, resampling_type,  resampling_size,
    padding_flag, logging_flag, log = None):⋯

    def read_input_images(self):⋯


    def reorient_images(self):⋯


    def resample_image(self):⋯


    def find_correct_shape(self,input_shapes, nb_vol_max_dim=32):⋯


    def convert_tensor_to_nifti(self, warped_img_tnsr, transformation, displacement):⋯


    def save_warped_image(self, warped_nifti_img):⋯


    def pad_nifti_img(self):⋯
```

Parametrization

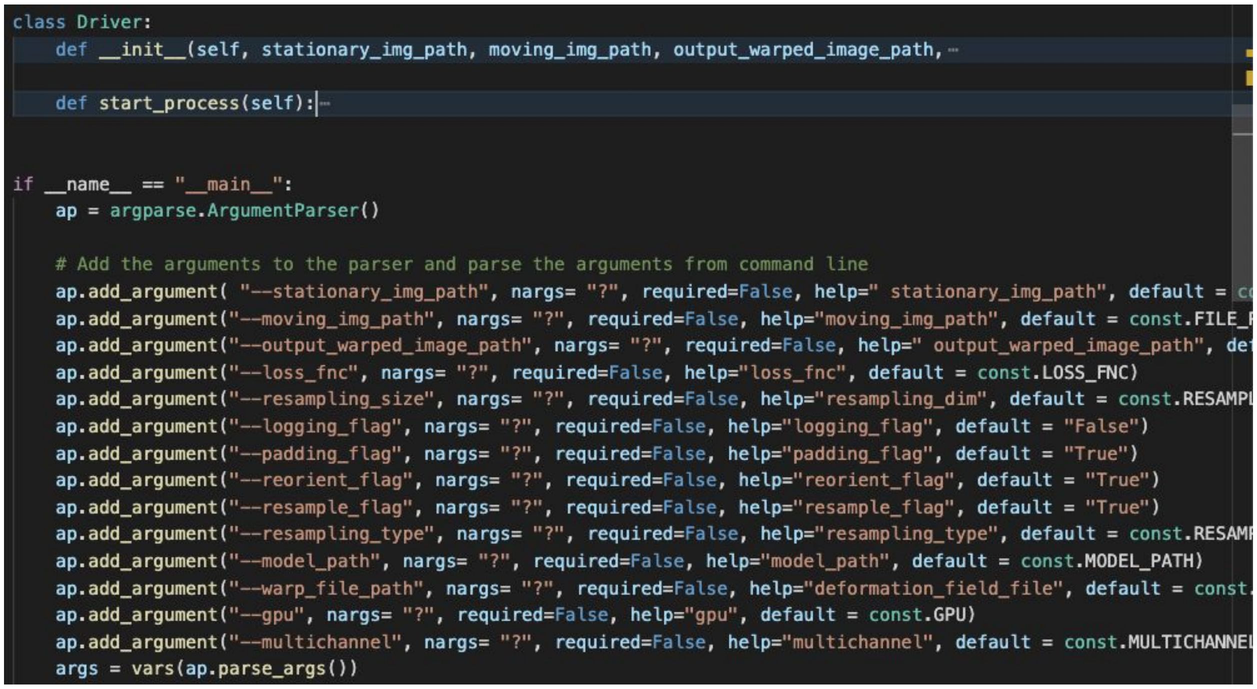
Application constants and default values to be used by the application are parameterized in dl_airlabs_constants.py file. This consists of default values of application and command line parameters if not given



```
# File path and name constants
FILE_PATH_STATIONARY_IMG = "/Users/surajshashidhar/git/image_registration/T1_images/IXI002-Guys-0828-T1.nii.gz"
FILE_PATH_MOVING_IMG = "/Users/surajshashidhar/git/image_registration/T2_images/IXI002-Guys-0828-T2.nii.gz"
FILE_PATH_WARPED_IMG = "/Users/surajshashidhar/git/image_registration/warped_image_voxelmorph.nii.gz"
FILE_PATH_LOG = os.path.join(os.getcwd(), 'AIRLABS_Image_Registration_log.log')
FILE_MODE = "w"
FILEPATH_VOXELMORPH_LIB = '/Users/surajshashidhar/git/voxelmorph'


LOGGING_LEVEL = 20
RESAMPLING_SIZE = [1, 1, 1]
LOSS_FNC = "MSE"
REORIENT_FLAG=True
RESAMPLE_FLAG=True
RESAMPLING_TYPE="both_moving_and_stationary"
PADDING_FLAG=True
LOGGING_FLAG=False
MODEL_PATH = "/Users/surajshashidhar/git/image_registration/brain_3d.h5"
WARP_FILE_PATH = "/Users/surajshashidhar/git/image_registration/deformation_field.nii.gz"
GPU = None
MULTICHANNEL = False
```

Main Driver

The main driver file driver.py takes in command line parameters such as image paths, preprocessing types and orchestrates the registration process. If parameters are not passed from command line application would take default parameters from constants file.



```
class Driver:
    def __init__(self, stationary_img_path, moving_img_path, output_warped_image_path,⋯

    def start_process(self):⋯


if __name__ == "__main__":
    ap = argparse.ArgumentParser()

    # Add the arguments to the parser and parse the arguments from command line
    ap.add_argument( "--stationary_img_path", nargs= "?", required=False, help=" stationary_img_path", default = co
    ap.add_argument("--moving_img_path", nargs= "?", required=False, help="moving_img_path", default = const.FILE_F
    ap.add_argument("--output_warped_image_path", nargs= "?", required=False, help=" output_warped_image_path", def
    ap.add_argument("--loss_fnc", nargs= "?", required=False, help="loss_fnc", default = const.LOSS_FNC)
    ap.add_argument("--resampling_size", nargs= "?", required=False, help="resampling_dim", default = const.RESAMPL
    ap.add_argument("--logging_flag", nargs= "?", required=False, help="logging_flag", default = "False")
    ap.add_argument("--padding_flag", nargs= "?", required=False, help="padding_flag", default = "True")
    ap.add_argument("--reorient_flag", nargs= "?", required=False, help="reorient_flag", default = "True")
    ap.add_argument("--resample_flag", nargs= "?", required=False, help="resample_flag", default = "True")
    ap.add_argument("--resampling_type", nargs= "?", required=False, help="resampling_type", default = const.RESAMP
    ap.add_argument("--model_path", nargs= "?", required=False, help="model_path", default = const.MODEL_PATH)
    ap.add_argument("--warp_file_path", nargs= "?", required=False, help="deformation_field_file", default = const.
    ap.add_argument("--gpu", nargs= "?", required=False, help="gpu", default = const.GPU)
    ap.add_argument("--multichannel", nargs= "?", required=False, help="multichannel", default = const.MULTICHANNEL
    args = vars(ap.parse_args())
```