**The Left Ventricle Software**

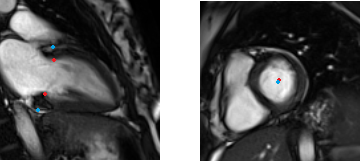
**for 2D Segmentation and 3D Mesh prediction**

**Introduction:**

Given a new MRI scan stored in .mat format, the software automatically makes predictions using pre-trained deep learning models.   
Segmentation CNN was developed by Bai et al, and mesh prediction CNN by us.

Assumption: Require only a partial annotation to localize the LV in a large MRI scan.

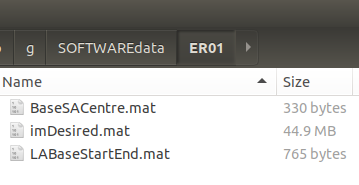
**Partial annotation – 14 points**



Essential points to be clicked:   
epi/endo centre of first SA slice + [epi/endo tops] x [2 points] x [3 LA]

= 2 + 3 \* 4 = 14 per scan

INPUT: 3 files – MRI data (imDesired.mat) + partial annotations in SOFTWAREdata/patient/\*



**Software steps:**

1. **Export Crops**
2. **Predict Segmentation**
3. **Predict 3D Mesh from Segmentation**
4. **Export Crops using EXPORT\_CROPS.py**

The script EXPORT\_CROPS.py applies resize/flip/crop and many other transformations.

**python EXPORT\_CROPS.py patient -f <FRAME> [-a <ANNOT>]**| outputs files in: SOFTWAREdataset/patient/\*

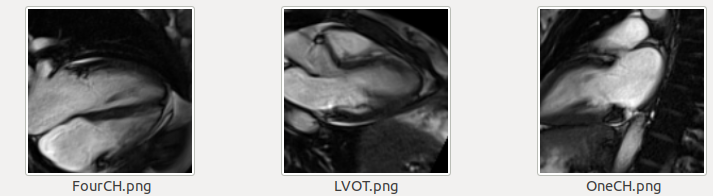
This produces auxiliary TRANSF.p file that contains the transformations done

**<FRAME>** is frame number -- then this can be run for any timeframe.

**The script usually needs to be run two times, as outlined below:**

**First run:**

**python EXPORT\_CROPS.py patient -f 10**



After the first run, one needs to inspect 3 LA slices: FourCH, LVOT, OneCH,

and check whether any of these needs to be mirrored, so as all LV in all images points right-side, and right atrium is at the top.

The usual things to check are:

* 1. FourCH: if right atrium near LV is above the LV. If not: mirror along X-axis
  2. OneCH: if LV points right-side, if not: mirror at Y-axis

So the annotations are one of four: “...”, “X..”, “X.Y”, “..Y”

**These are provided as an optional <ANNOT> argument** - 3 characters in {X, Y, .}

Second run:

**python EXPORT\_CROPS.py patient -f 10 –a X.Y**

**Now all is aligned properly:**



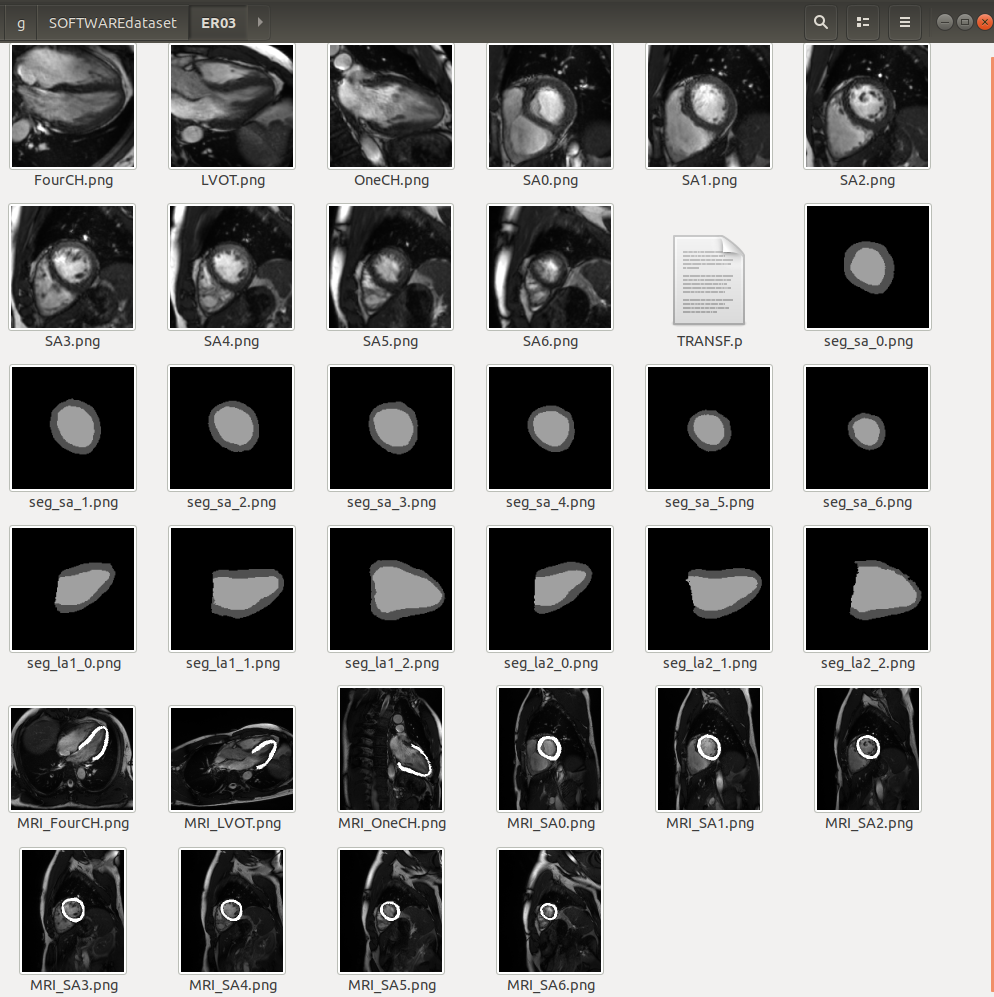
1. **Predict Segmentation using EXPORT\_SEG.py**

The script to run proper segmentation networks (SA/LA segmentation networks, SA + two versions of LA):

**python ./EXPORT\_SEG.py patient   
|** outputs files in: SOFTWAREdataset/patient/\*

**INPUT**: crops of images

**OUTPUT**:   
predicted segmentation SA, 2 x LA (la1 – for the CNN, la2 – without top wall),   
and segmentations in MRI coordinates (SA + LA). Pixel brightness 255 represents the segmentation, and the image values are lover  
  
**EXAMPLE** **FOLDER with OUTPUTS**:



1. **Predict 3D mesh from segmentations   
   using EXPORT\_3DMESH.py**

**SCRIPT:**  ./EXPORT\_3DMESH patient –t centr/rot

**|** outputs files in: SOFTWAREmesh/patient/\*

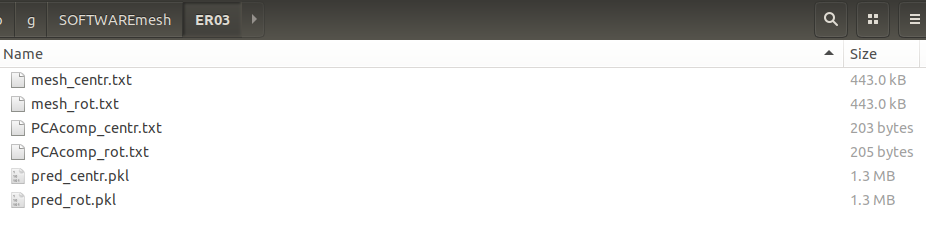
The script makes predictions for patient, for two data type alignments:

Centred or Procrustes-rotated (See paper for details)

The trained centr/rot networks and respective PCAs are stored in /nets/ folder.

This exports predicted mesh, component values, and also PCA parameters for convenience in Python format, and in addition the 3D mesh and PCA representation in text file.

Files produced for both versions:



Components: saved in PCAcomp\_ .txt file – 8 values for 8 components.

Mesh: 17k+ vertex coordinates are save in mesh\_ files.

In addition pickle file pred\_ contains these predictions plus the PCA parameters:

[patient\_name, Components, Mesh], [mean\_mesh, data\_mean, data\_std, PCA\_matrix]

**EXAMPLE RUN for patient ER03 (frame 13, rotated alignment):**

python EXPORT\_CROPS.py ER03 -f 13 –a ...

python EXPORT\_SEG.py ER03

python EXPORT\_3DMESH.py ER03 -t rot

The networks were trained using GPU, but these codes for single patient predictions run at the same speed on a CPU by default. GPU number can be chosen by adding –g 0 option for GPU number 0 to the EXPORT\_SEG.py and EXPORT\_3DMESH.py commands.