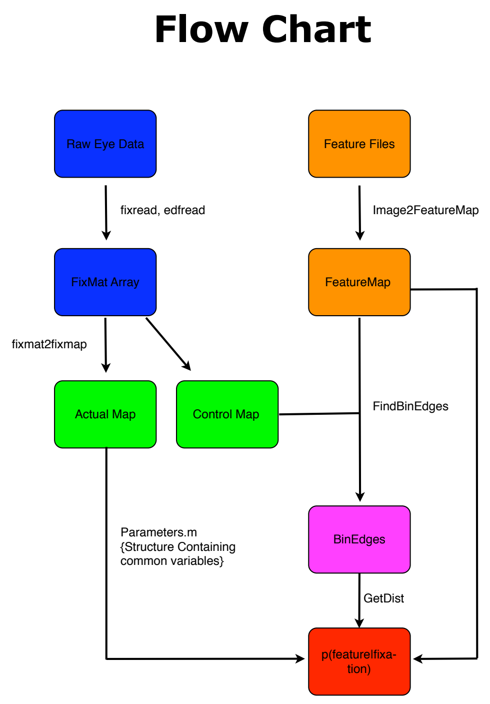
0.1 General Overview

The set of functions present in this toolbox are used to compute posterior probabilities: probability of fixation given feature percentiles. The detailed explanation of the steps can be found in Frank Schumann's Master Thesis. In Figure 1, please find the function names which are responsible of different steps leading to posterior probabilities.



0.2 General Requirements

You need to respect to some folder naming conventions in order to use the toolbox without much problems. The folders you must have created before using the toolbox are listed below. It is recommendede that you first create a project folder ($) where all the following folders will be stored.

$/FeatureMaps In this folder, feature maps of different images must be stored. Each feature map, such as Luminance contrast maps, mean luminance maps etc. must have their own folders. Here is an example FeatureMap folder organization.

```
drwxr-xr-x 2 sonat ustaff 8192 2007-06-01 20:44 RG
drwxr-xr-x 2 sonat ustaff 8192 2007-06-02 02:18 RG_C_Radius_23
drwxr-xr-x 2 sonat ustaff 8192 2007-06-01 23:08 RG_C_Radius_45
drwxr-xr-x 2 sonat ustaff 8192 2007-06-02 01:28 RG_M_Radius_23
drwxr-xr-x 2 sonat ustaff 8192 2007-06-02 00:18 RG_M_Radius_45
drwxr-xr-x 2 sonat ustaff 8192 2007-06-01 20:44 SAT
drwxr-xr-x 2 sonat ustaff 8192 2007-06-02 03:02 SAT_C_Radius_23
drwxr-xr-x 2 sonat ustaff 8192 2007-06-01 23:03 SAT_C_Radius_45
drwxr-xr-x 2 sonat ustaff 8192 2007-06-02 01:56 SAT_M_Radius_23
drwxr-xr-x 2 sonat ustaff 8192 2007-06-02 00:23 SAT_M_Radius_45
```

```
drwxr-xr-x 2 sonat ustaff 8192 2007-06-01 20:44 YB
drwxr-xr-x 2 sonat ustaff 8192 2007-06-02 03:46 YB_C_Radius_23
drwxr-xr-x 2 sonat ustaff 8192 2007-06-02 00:08 YB_C_Radius_45
drwxr-xr-x 2 sonat ustaff 8192 2007-06-02 02:23 YB_M_Radius_23
drwxr-xr-x 2 sonat ustaff 8192 2007-06-02 01:05 YB_M_Radius_45
```

Within each of the feature folders the featuremaps of each stimulus image must be stored as a single `.mat` file. An example and prefered organization is the following:

```
-rw-r–r– 1 sonat ustaff 8904993 2007-06-10 21:09 image_001.mat
-rw-r–r– 1 sonat ustaff 8912429 2007-06-10 21:09 image_002.mat
-rw-r–r– 1 sonat ustaff 8900295 2007-06-10 21:09 image_003.mat
-rw-r–r– 1 sonat ustaff 8910572 2007-06-10 21:09 image_004.mat
-rw-r–r– 1 sonat ustaff 8945344 2007-06-10 21:09 image_005.mat .
.
.
-rw-r–r– 1 sonat ustaff 8898754 2007-06-10 21:26 image_254.mat
-rw-r–r– 1 sonat ustaff 8896367 2007-06-10 21:26 image_255.mat
```

Note that there is not a distinction on the file names concerning the conditions. In the example above the images 1 to 64 and 65 to 128 ... belongs to different conditions. However this is not explicitely coded in the file names in the case presented above.

$/BinEdges Within this folder the bin edges will be stored as `.mat` files. (If you dont know what it is 'binedges', please refer to the master thesis of Frank Schumann). Each bin edges data will have a unique filename specifying the parameters that were used during its computation. If you would like to compute the binedges of the Luminance Contrast features, of condition Pink Noise, with a given smoothening factor, within the first half of the presentation etc. etc. all this will be stored on the filename, thus leading to somewhat strange filenames. However these filenames will be very handy when we would like to know with which parameters a given bin edges has been computed.

$/PostDist By using each bin edge file and its corresponding feature map we compute the Posterior probabilities (that is probability of fixation given a feature value or feature percentile). These are stored in this folder. The file names within this directory are exactly the same as the filenames within the BinEdges folder.

Except the featuremap folder, users do not need to actively maintain the $/BinEdges and $/PostDist folders. Because the content of these folders are created by functions which know how to choose their filenames. However you have to take that everything is perfectly fine within the $/FeatureMap directory. That is all the folder dedicated for different features within the folder $/FeatureMap must have the same number of naming conventions, and there must be no images missing etc. etc.

The toolbox relies exclusively on the usage of the standard *fixmat* format to make anything meaningful with the fixation data. If you dont know what is the fixmat format please see Johannes Steger's instructions in stud.ip (*edfread and fixread cheatsheet, with install instructions*). However you must add to the fixmat a field called `.rect` containing the size of the images shown. In case your images are 960 times 1280, then you would need to add the `.rect` field in the following way:
fixmat.rect = [0 960 0 1280]

0.3 List of Basic Functions

Here you find a short description of what the functions are doing. More help is available on their help section, just type `help {functionname}` in matlab command line.

0.3.1 Feature Map Related Functions

These functions computes the *contrast* and *mean* maps of the images shown. They save their results in specific folders which have their unique names formed according to their input arguments. You may use these functions together with different *base features*. For example if you would like to compute the *luminance contrast map* of a set of images you would need to run `GetContrastMaps` function on the Luminance base feature. Luminance base feature is nothing but the luminance values of the images. You may, by using the same function compute the *red-green color contrast* of a set of images by using red-green base feature. Depending on what you feed to these functions they produce different feature maps but the underlying computations are the same.

- `GetMeanMaps(r,varargin)`

  Uses `LuminanceMap2.m`. `R` is the diamater of the circular (not gaussian) patch. `VARARGIN` is the name of the feature folder where the mean luminance maps must be computed. For example, below three feature maps are given (these are some of the folders which are located in the $/FeatureMap folder).

  $/FeatureMap/LUM

  $/FeatureMap/LUM_M_Radius_23

  $/FeatureMap/LUM_M_Radius_45

  The folder LUM contains the raw images shown to the subjects during the experiments, that is why it is referred as a *base feature*. That is reason why they are called Luminance maps within this context although it is not true that these are feature maps in the strict sense. The function `GetMeanMaps` when used in the following way

  ```
  >> GetMeanMaps(23,'LUM');
  ```

  it opens each of the images within the directory LUM and computes the mean luminance maps and saves the results in a folder named LUM_M_Radius_23. _M_ holds for 'mean'.

- `GetContrastMaps(r,varargin)`

  Relies on `ContrastMap2.m`. It is very similar to `GetMeanMaps` function in the way it operates however instead of measuring the mean luminance maps, it computes the contrast maps (Uses _C_ instead of _M_ while creating the feature specific folder. Just keep in mind that

  ```
   >> GetContrastMaps(23,'LUM')
  ```

  would compute the contrast maps of the images and save the results in folder LUM_C_Radius_23. A second run of the same function on the resulting folder

  ```
  >> GetContrastMaps(120 , 'LUM_C_Radius_23' )
  ```

  would compute the contrast map of the contrast maps stored in folder LUM_C_Radius_23 to the folder LUM_C_Radius_23_C_Radius_120. This is the way how to obtain texture maps.

0.3.2 Helper Functions

- `FeatMap = Images2FeatMap(path,ImIndex,varargin)]`

  It is used to load feature maps to the working space.

  Usage:

  ```
  >> Images2FeatMap(['$/FeatureMaps/' FeatureName ],1:10)
  ```

  With this usage it loads the first 10 images to the memory. It is handy to use this function with `GetFeatures` as it provides a easy way to refer to different feature folders. As a default behavior it opens the individual feature maps as they are stored without any further modification. varargin can be used to specify a given amount (in pixels) to be cropped from the all of the edges. That if varargin is equal to 200, then 200 pixels are cropped, leading a reduction of 400 pixel of size in a given cardinal direction.

- `[p,a,d]=CircularPatch(r)`

  Computes a circular mask. This is used by functions which are responsible of feature map comptation.

- `[map,nPix]=CropperCleaner(map,varargin)`

  This function is used to remove the zeros of the zero-padding or to crop a specified amount of pixels from the 2D feature maps. Feature maps have always their size smaller then the original images. This is due to the fact that it is not possible to compute the feature values at the edges of the images. That's why they are saved after zero padding with a necessary amount of zeros so that the size of the feature maps matches to the size of the original images. With this function you can easily manipulate the zeros padding.

- `p = GetParameters`

  This is one of the most important functions. It provides all the parameters that are necessary to make any further computations. It contains the following fields (the values which are shown below are arbitrary):

  - `p.FWHM: 45`
    This is the Full Width Half Maximum (in pixels) of the gaussian smoothening kernel. To correctly use this you must know how much degree a pixel corresponds in your setup.
  - `p.CropAmount:  0`
    This is the number of pixels that a feature map must be cropped from all of its edges. This is necessary to remove the edge artefacts of feature maps.
  - `p.kernel:  [117x117 double]`
    This is the smoothening gaussian kernel. It is computed according to the value of the FWHM field.
  - `p.nBin:  20` Number of bins.
  - `p.start:  0`
  - `p.end:  6500`
    Start and end values (in ms) determine the interval of the stimulus presentation that must be used for the analysis. For example if it is equal to [0 2000] only the fixation within the first 2 seconds are included.
  - `p.nBS: 0`
    Number of bootstraping.
  - `p.fix:  2`
    The first valid fixation index. If it is equal to 2, then the first fixation of all trials are excluded from the analysis.
  - `p.subjects:  [1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25]`
    The indices of the valid subjects.
  - `p.folder:  [1x134 char]`
    This is the tag of the current parameters used. It is unique for this parameter structure array and it will be used to save the derived data such as bin edges and posterior distributions. It is created by calling `Param2Folder` function.
  - `p.ffcommand:  'fix' [1x999 double] 'start' [0] 'end' [6500] 'subject' [1x25 double]`
    This field will be used in `GetDist` function.

  This function when used without any input arguments simply returns all the default values. However you may overwrite the default values by entering first the name of the field and second its new value. For example

  ```
  >> GetParameters('FWHM',90);
  ```

  would overwrite the FWHM defaults value with the entered value 90. The `P` structure array will be used ubiquitously during the analysis. It will be passed to many other functions and specify the necessary parameters.

- `folder=Param2Folder(p)`

  It is used to generate a tag specific to the current state of parameters. Used mainly with `GetParameters.m`

- `f = GetFeatures`

  With this function you can easily get a list of feature folder you have. It will be very handy when you will need to enter feature folder names to different functions.

  `>> f = GetFeatures;`

  Returns all the features you have in your FeatureMap folderas a cell of string arrays.

  `>> f = GetFeatures('LUM*');`

  Returns all the features which starts with the string 'LUM'.

- `FixMap=fixmat2fixmap(fixmat,kernel,CropAmount,varargin)`

  This function is used to create fixation maps. The varargin inputs are passed to `SelectFix` function.

  `>> fixmap = fixmat2fixmap(fixmat,p.kernel,0);`

  `FIXMAP` is the fixation map computed with all of the fixations which is present in `FIXMAT`. To smoothen the binary map, It uses the kernel contained in `P` structure array (usually obtained by calling `GetParameters`). It applies no cropping as the last input is zero.

  `>> fixmap = fixmat2fixmap(fixmat,p.kernel,100,'image',1:10);` Computes a fixation map using images one to 1.

  `>> fixmap = fixmat2fixmap(fixmat,p.kernel,100,'image',1:10,'fix',2:1000);` Computes a fixation map using images 1 to 10 without taking into acount the first fixations. `>> fixmap = fixmat2fixmap(fixmat,1,100,'image',1:10,'fix',2:1000);` Computes a non smoothened map as the kernel is simply 1.

- `fixmat=GetFixMat(i)`

  This function must be adapted according to your setup. It simply load the I^{th} fixmat file. If you have more then one fixmat, it is handy to load to the working space the desired fixmat.

- `be=Load_FindBinEdges(feat,cat,p)`

  It loads the bin edges `BE`, computed with the feature `FEAT`, condition CAT and parameter array `P`. The bin edges files are written to the disk by the meta function `All_FindBinEdges` which calls FindBinEdges.

- `[varargout]=SelectFix(fixmat,varargin)`

  This function is used to filter the fixmat structure array. Example usage:

  `>> fixmat2 = SelectFix(fixmat,'subject',1:10)`

  Removes all the fixations within the fixmat array `FIXMAT` which do not belong to subjects one to ten. That is all the fixation data in `FIXMAT2` would be composed of the data originating from subjects one to ten.

  `>> fixmat = SelectFix(fixmat,'fix',2:1000)`

  Remove all the first fixations.

  `>> fixmat = SelectFix(fixmat,'image',1)`

  Keep only the data done on the first image.

  `>> fixmat = SelectFix(fixmat)`

  Calling the function without any other argument then fixmat would simply remove the fixations which are out of the screen.

0.3.3 Core Functions

You will not need to use these function directly on the command line. They are called by meta function listed in the next section.

- `BinEdges=FindBinEdges(FeatureMap,FixMap,nBins)`

  This function computes the binedges leading to histogram equalization. That is the occurence of the feature values (after being weighted by the central bias) when counted within the bins detected with this function is equal for all the bins.

  `FEATUREMAP` is the result of `Images2FeatMap`. `FIXMAP` is the fixation map, it is the result of `fixmat2fixmap`. Each computation of bin edges needs a weight matrix because not all part of the monitor are likely to be seen, this weight matrix represents the central bias of the subjects. `NBINS` is the total number of bins you would like to devide the feaeture space.

  Usage:

  `>> p = GetParameters;`

  Get the default parameter values.

  `>> fixmap = fixmat2fixmap(fixmat,p.kernel,p.CropAmount);`

  Get the control fixation map.

  `>> [BinEdges] = FindBinEdges(f.data,fixmap,20);`

  Compute the bin edges.

- `res = GetDist(fixmat,p,varargin)`

  This is the longest function and it is responsible of generating the posterior probabilities. It is recommended you understand what is going on inside. It is compatible with N dimensional distributions that is it can accept any number of binedges and feature maps pairs. It computes for N dimensional feature space the posterior probabilities. It is highly recommended that after you use this function you simply check that your control distributions are flat.

- `c=Core(FixMap,varargin)`

  You will never need to call this function on the command line, GetDist heavily relies on this function. See help Core for more information.

0.3.4 Meta Functions

- `All_FindBinEdges(f,p)]` This function computes the bin edges for the features in `F` according to the parameters stored in `P`. `F` contains the output of `GetFeatures`. `P` is the parameter variable, result of `GetParameters`. This function must be adapted to your requirements. In the current version, the conditions limits (i.e. the first and last frames belonging to a condition) are hardcoded on the function. This will probably change in future. For the time being you must enter by hand your own image indices by hand. At line 25 you can see the default condition delimiters: [1 64; 65 128; 129 192; 193 255], this means that the image indices belonging to the first condition are from 1 to 64 and the second condition's images have indices from 65 to 128 and so on. The result are saved in the folder $/BinEdges.

- `[All_GetDist(FileList)]`

  Similar to the previous function, This function computes the posterior probabilities for all the computed bin edges. `FILELIST` is cell array containing the filenames of different bin edges stored in the folder `BinEdges`. The results are saved in $/PostDist folder.