Read Me - from ROI selection to movie creation

1 Goal of the 5 scripts

The goal of the Python scripts named P1 to P5 is to detect regions of interest (ROIs) on GCaMP6s and tdTomato fluorescence images, to allow the user to correct the ROIs found automatically and to compute movies containing the behavior images from the fly, the fluorescence images, the optic flow measurements from the ball and the %DR/R traces from left and right neurons calculated based on the selected ROIs.

2 Setup of the Environment

To run these 5 Python custom scripts the user needs to install Python (currently running with version 2.7.10) and the following libraries (note that most of them will come with your installation of Python) :

- Numpy (v 1.13.1)
- OpenCV (v 3.3.0)
- Matplotlib (v 1.3.1)
- PIL (v 4.2.1)
- Skimage (v 0.13.0)
- cPickle (v 1.71)

The script runs on the versions mentioned next to the libraries. Click on the libraries name to access their documentation.

3 Folder Organisation

The data should be organised as presented below to run the 5 ROI selection scripts. One folder is created per experiment and should contain :

- *a behavior_imgs folder* containing all the behavior frames and a .csv file containing the acquisition time of each behavior frame
- *an .h5 file* containing all the information from the computers during the experiment. The script will extract, from this file, the timing of the fluorescence frames but also the information to align the behavior frames and the optic flow measurements to the fluorescence images.
- *a registered folder* containing the three .tif files with the fluorescence images from GCaMP6s and tdTomato.
- *an opflow.txt file* that contains the vector informations and timing of the optic flow measurements

- *a timeStamps.txt file* that contains the acquisition timing of the pulse signal. The pulse signal is the signal that coordinates the two computers and transfers the information to the second computer that the first one is acquiring the behavior frames.

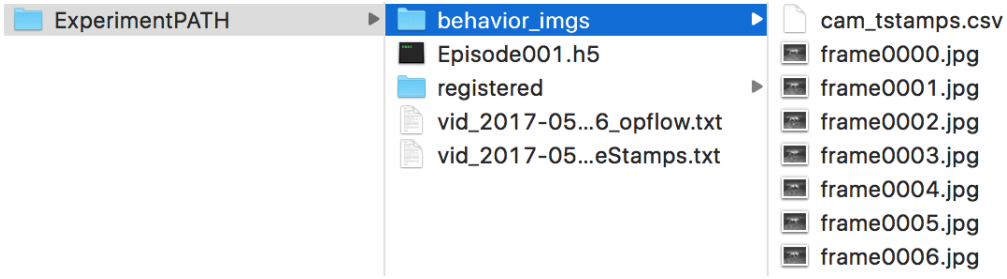

Figure 1: Folder organisation - Part 1

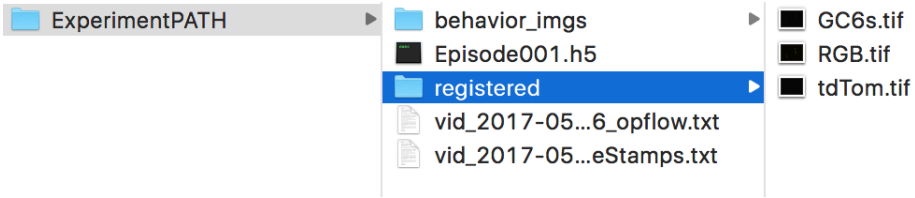

Figure 2: Folder organisation - Part 2

4 Running Scripts P1 to P5

The 5 Python scripts have to be run one after the other to perform the ROI detection, correction and to create the summary movie. In the 5 scripts you will need to set the path to your experiment folder (the one presented in the previous section) as the « dataDir » path (see figure below). Once set in the 5 Python scripts, you can run the first script P1 from your terminal.

Figure 3: dataDir Path

4.1 P1 - ROI autodetection

When P1 script is run, it asks the user which channel he wants to use for the analysis as presented in fig 4.

```
Do you want to use both TdTom & GCaMP6s (RG), or GCaMP6s channel alone (G) or TdTom channel alone (R) ?
Keyin the channels you want to use to detect the ROI (eg.,'R' or 'RG')
->
```

Figure 4: Fluorescence channel selection

The first fluorescence frame with all the detected contours is presented to the user. The user has to select the appropriate ROI for left and right neurons.

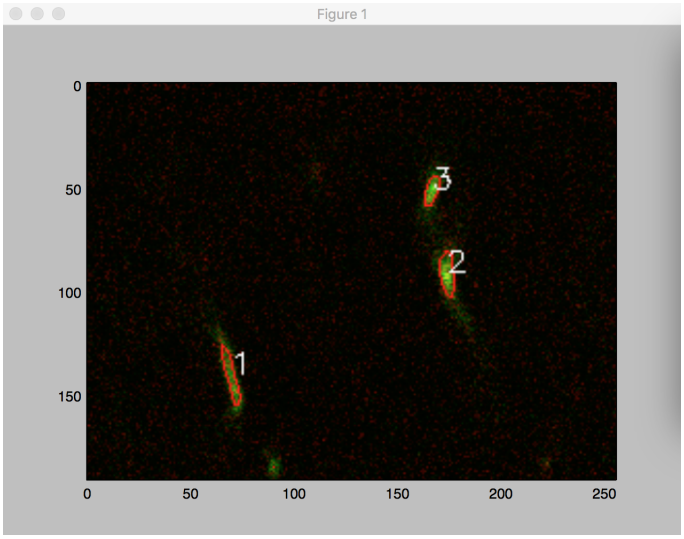

Figure 5: Reference frame

```
Keyin the channels you want to use to detect the ROI (eg.,'R' or 'RG')
->RG
which is L ROI?
key in number of the ROI ->
```

Figure 6: Reference frame - ROI selection

```
which is L ROI?
key in number of the ROI ->1
which is R ROI ?
key in number of the ROI ->
```

Figure 7: Reference frame - ROI selection

The script finds all the left and right ROIs on the other fluorescence frames based on the object detection within the current frame being processed and cross-correlation with the first reference frame where the good ROIs were selected by the user. This script usually runs in about 1 minute. Once it is done, a new folder named output is added to the experiment folder. The output folder contains three folders :

- *cropROI_auto folder* storing the cropped fluorescence images with left and right ROI drawn on them

- *GC6_auto folder* storing the fluorescence values from the left and right ROIs but also the %DR/R for left and right neurons.
- *ROI_auto folder* that contains the non-cropped fluorescence images with left and right ROI drawn on them

4.2 P2 - ROI Manual Correction

The automatic detection of the regions of interest is now completed, script P2 can be run. This script allows the user to observe the ROI detected on the fluorescence frames in the frame range of his choice. The user is asked to enter the frame range he wants to review as presented on figure 8.

```
>>>Reading GC_orig_auto...
('open len', 650)
>>>Import auto-selected images...
('***', 649, ' images for review in total (from', 0, '-', 650, ').***')
Keyin the range of frame for correction (eg.,'25-66','0-350' or 'all')
->all
```

Figure 8: Frame Range

Each frame from the ROI_auto folder created in P1 is presented to the user for a visual check. For each frame the user can decide whether the ROIs are well detected or not. The user also has the possibility to go back to the previous frame to « recorrect » it.

```
('There are', 9, 'frames to fix')
frame from ROI Usr Corrected
('image in review - frame number ', 641)
-----------------------------------
If wrong, keyin 'n' or 'N'.
If correct, press 'Enter'.
Back to previous img, press 'b' or 'B'.
Pause and come back later, press 'p' or 'P'.
-----------------------------------
->
```

Figure 9: User Selection

If the ROIs are not correctly detected on the frame, the same procedure for the reference frame in P1 is used and all objects detected within the frame are presented to the user for ROIs selection as presented in fig 10.

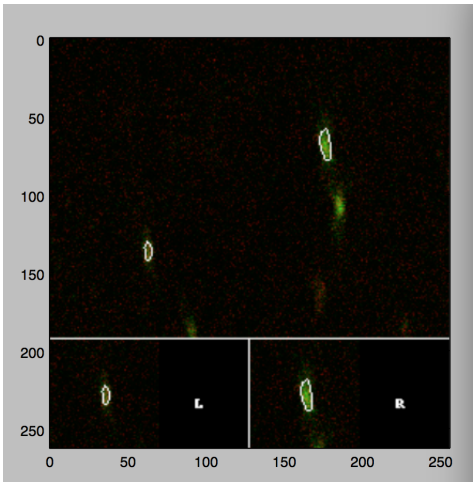

Figure 10: Contours Detected

If the ROIs presented to the user are still not correct (wrong shape or ROI not detected), the user has the opportunity to enter « +X » and this frame will be stored to go through a manual drawing procedure of the ROI in P3 script.

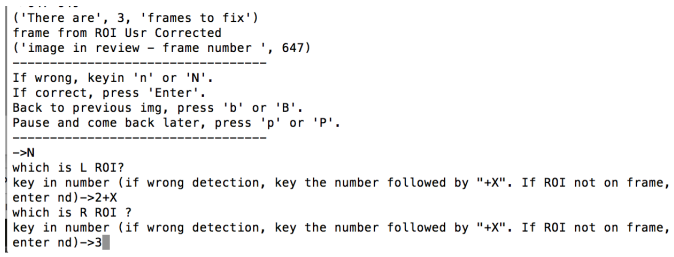

Figure 11: Wrong Detection

At the end of script P2, 4 new folders are created within the output folder:

- *cropROI_UsrCorrected folder* contains all the cropped fluorescence images (corrected with P2 and automatically selected for the frames that did not need correction) with left and right ROIs drawn on them
- *GC6_UsrCorrected folder* contains all the raw fluorescence (corrected with P2 and automatically selected for the frames that did not need correction) from left and right ROIs but also the %DR/R for left and right neurons.
- *ROI_UsrCorrected folder* that contains the non-cropped fluorescence images (corrected with P2 and automatically selected for the frames that did not need correction) with left and right ROIs drawn on them
- *Frame#_UsrSelected* that is used only if the P2 script is paused to store the frames that still need to be fixed in a txt file.

If the contours presented to the user during this correction were still not satisfying, the user has the opportunity to run script P3. P3 will open the text file created in the GC6_UsrCorrected folder and named : wrongROIShape.txt. This text file contains the frame numbers for which the user added the "+X" when reviewing the frames in script P2.

4.3 P3 - ROI Manual Draw

The script opens the frames to fix and presents them one by one to the user. The user has the opportunity to draw 2 ellipses as the left and right regions of interest. The user has to select the fluorescence channel he wants to use to detect and draw the ROI and the frames from this channel are presented one by one.

The user needs to position two centers of the two ellipses that are going to be drawn on the frame. The left ellipse center must be drawn with the left mouse click and the right ellipse center must be drawn with the right mouse click. Once the two centers are selected, the user must add one mouse click (left or right). Figure 12 presents the situation - the user has the opportunity to confirm the center selection or to go through the centers selection again.

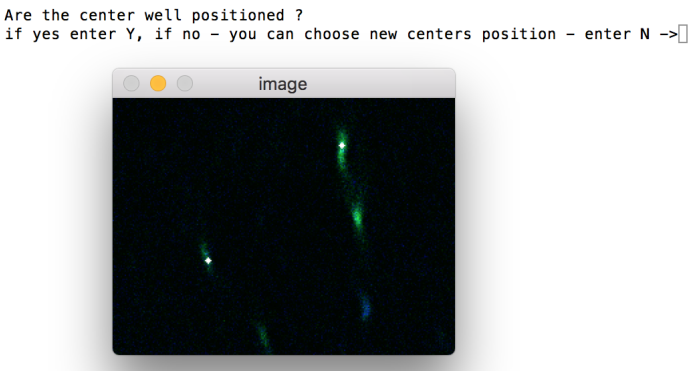

Figure 12: Ellipse Dimensions

Once the ellipses centers are well positioned, the user has to enter the ellipses dimensions. The script will then draw the two ellipses and the user has the opportunity to go through the dimensions selection again.

Once the user is satisfied with the ellipses drawn on the frame, the next frame to be corrected is presented. At the end of the P3 script, no new folder is added to the output folder. The ROI drawn on the fluorescence frames are directly changed in the cropROI_UsrCorrected folder and in the ROI_UsrCorrected folder. New text files are added in the GC6_UsrCorrected folder containing the new fluorescence values calculated from script P3 (which is a mix of the corrected frames in script P3 and corrected frames in script P2).

4.4 P4 - Alignment

Once the scripts P1 to P3 have been run, all the ROIs should be well detected. The alignment script needs to be run to align the optic flow measurements with the fluorescence images and %DR/R. This script uses the .h5 file, the csv file within the behavior_img folder, the optic flow and cam_stamps text files and the fluorescence text files from GC6_UsrCorrected folder. This script runs in more or less 20 seconds. Once again don't forget to change the « dataDir » path to the one of your experiment folder. Once the script is done, a new .p file is added to the output directory. It is a dictionary that contains all the data aligned that will be used to run script P5.

4.5 P5 - Movie Creation

This script computes the frames for the final movie. It uses the dictionary created from P4, the frames from the behavior_img folder and the frames from the cropROI_UsrCorrected folder. A new folder is created in the output folder named : FramesforVideo. It contains all the frames to compute the final movie. The following ffmpeg command can be run from the terminal to compute the final movie :

```
cd YOUR_FRAMES_FOR_VIDEO_FOLDER_PATH
ffmpeg -r 30 -start_number 0086 -i VidFrame%04d.png -c:v libx264
-preset ultrafast -vf format=yuv420p -crf 0 FinalMovie.mp4
```

5 Detailed Explanations of scripts P1 to P5

5.1 P1 - ROI autodetection

Main from start to beginning of the loop
The first fluorescence image is selected as the reference frame. The thresholding function performs several transformation on the image : the dynamic range of pixel value is extended using the full bandwidth of the 8 bit image to augment the contrast in the image. A blur filter from OpenCV library is applied before applying the Otsu Threshold. The output of this threshold is a binary image with the background as one color and all the objects within the image as the other color. A kernel with an erosion factor set in the global variables is then applied using OpenCV library. This erosion function is used to avoid the detection of very small regions of interest. The user should decrease the erosion value if he wants to increase the size of the detected ROIs (and vice versa). The output image is now a binary image with components eroded. This image is given to the connectedComponents function from OpenCV library. This function returns the number of objects detected in the image and the image itself with the background set to 0 and all the different objects set to their respective label value. All the detected contours are then drawn (with the number of their position in the total contours list) on a copy of the reference image and presented to the user with the AskROI() function. This function returns a list containing, at the first position, the number of the contour selected as the left ROI and at the second position, the number of the contour selected as the right ROI. The selected contours are drawn on the fluorescence frames that are going to be stored and the centroid position of the selected contours are calculated. Those centroids are set as the reference centroids : PRCL and PRCR. The fluorescence frames are then cropped and stored with the contours selected by the user. The fluorescence values are calculated in GetAbsFluoValues() by applying a mask containing 0 values at the background positions and 1 values at the contour position. The mean of the pixel value in the ROI are then computed for GCaMP6s and tdTomato and stored in 4 different lists - 2 for left neuron and 2 for right neuron.

The reference frame was therefore processed with the selected ROI from the user and those ROIs were set as the reference ROI for the next part of the script.

Main from beginning of the loop
The script performs a loop on all the fluorescence images. The first step is used to detect the ideal position of the current ROIs based on a cross correlation between the current frame being processed and the reference frame. The Python version of Manuel Guizar's script was used to perform the cross correlation of the two frames. The new optimal position of the ROIs centroid were then computed and set as PRCRC (right centroid) and PRCLC (left centroid). The same thresholding function is applied to the image and an eroded image is given to the findMostProbableContour() function. All the objects within the image are detected using OpenCV library as previously explained. math.hypot was used to compute the Euclidean distance between all the centroids of the ROIs detected in the current frame and the « ideal centroid of the ROI calculated by the cross correlation ». The ROIs having the closer centroids to the ones calculated with the cross correlation were selected. The frames were then processed as previously explained for the reference frame : selected contours were drawn on the fluorescence images and stored, fluorescence values were extracted from ROI selected.

Main after the loop
The %DR/R were calculated once all the fluorescence values were computed from the different ROIs. %DR/R is the ratio of GCaMP6s fluorescence to tdTomato fluorescence normalized by the baseline ratio. The baseline ratio was calculated using a 2.5 seconds time window over

the fluorescence samples. The average of the ratio of GCaMP6s to tdTomato was computed for this 2.5 seconds window over the entire list of fluorescence values from the ROI. The minimum average of the ratios was selected as the baseline ratio R. The %DR/R, raw tdTomato fluorescence and raw GCaMP6s fluorescence of the ROIs were stored in text files.

5.2 P2 - ROI Manual Selection

Main from beginning to while loop
The script checks if it was already run and paused by looking for the « paused » txt file containing the frames to fix. It opens the most up to date fluorescence values. If the P2 script was run and paused, the user has to complete the paused selection and the script will enter in the UIselect==1 condition and will open the paused fluorescence values and number of frames to fix. Once the paused selection completed, the user will have the opportunity to check another range of frames if he wants to.

The while loop
Every frame in the frame range will be presented to the user. For each frame the user has the opportunity to press enter if the ROIs are well detected, to press P to pause the script if he wants to pause and come back later (temporary files are created and stored to be restored later), to press B if he wants to come back to the previous frame corrected (k value used in the while loop is set to k value from 2 frames before (because at the end of the while loop the k value will be increased by one) or to press N if the ROIs are not correctly detected on the frame (boolTo3 will be set to 1). If N was entered, the script enters the BoolTo3==1 condition. All the contours within the image are detected as performed in script P1 and are presented to the user (refer to P1 detailed information). The user has to select the correct ROIs from the ones that are presented. If none of the ROIs shape presented are correct, the user enters +X and the frame number is stored in a list called WrongDetection. If the left of right ROI is not present on the fluorescence frame, the user can select to set the fluorescence value to NaN by entering « nd ». The script will then enter the boolLN or boolRN == 1 condition (respectively linked to left or right) and will store the fluorescence image with a black image on the part of the missing contour.

Once the while loop is over, the %DR/R are recalculated and the fluorescence values are stored in the new folder named « UsrCorrected » text files.

5.3 P3 - ROI Manual Draw

Main from beginning to while loop
The script opens the frames to fix from wrongROIshape text file. It opens the fluorescence values from the most up to date fluorescences text files and from the paused text files from P3 if the script was run and stopped in the middle. The user has the possibility to select either GCaMP6s channel or tdTomato channel for ellipse drawings with the ChannelSelection() function.

While loop
The script enters the ManualDrawFrame() function and the frame to fix is presented to the user. First he will need to select the center of the ellipses. Two lists are used to store the position of the centroid selected by the user. ValList1 and ValList2. OpenCV library is used to discriminate between left and right mouse click. Left mouse click is used to set the centroid of the left ellipse and right click is used to set the centroid of the right ellipse. The user should perform one left click for left ellipse centroid, one right click for right ellipse centroid and then either perform one left or one right click. If the user performs two right mouse clicks before

performing one left click, the process will start over and the frame will be cleaned. Once the centroids are selected, the user can confirm the good selection of the centroids and perform the dimension selection of the ellipses. Once selected, the ellipses are drawn on a black image and OpenCV library is used to detect the ellipse contours. The contours are drawn on the RGB image and the contours list is returned by the function. The contours are then applied to the fluorescence images and the fluorescence values from the ROIs are extracted and stored in lists as previously detailed in P1 script. Once the while loop is over, %DR/R values are recalculated and stored in new text files.

5.4 P4 - alignment

From getFluoData() to OpenOpflowVector()
The script opens fluorescence data from the most up to date script (it first looks for fluorescence text files computed in P3 script, then in P2 script if the ones from P3 script don't exist,...) and it will let the user know from which files the fluorescence data were used. It then extracts the frameCntr signal, puff signal, piezo signal and step signal from the h5 file created for each experiment. The step signal and frameCntr signal are two lists of the same length and are sent by the same computer. The step signal is a list containing values either equal to 0 or 10. 10 values are set only when the computer creating the step list receives information from the second computer that the behavior frames are being taken. When the behavior frames are processed, the pulse signal is sent to computer number 1 from computer number 2 and the step signal is then moved from 0 to 10. The frame counter is the signal containing the number of the fluorescence image that is being processed. The frame counter increases with the piezo step and 3 piezo steps are corresponding to one frame being processed, therefore the frameCntr signal needs to be divided by 3 to find the fluorescence frame that is currently being processed. Refer to figure 13 for more information on signal alignment. StepOnTrueIdx is a list that contains the index positions at which the step values are above 9 (happening when it receives the signal from computer number 2).

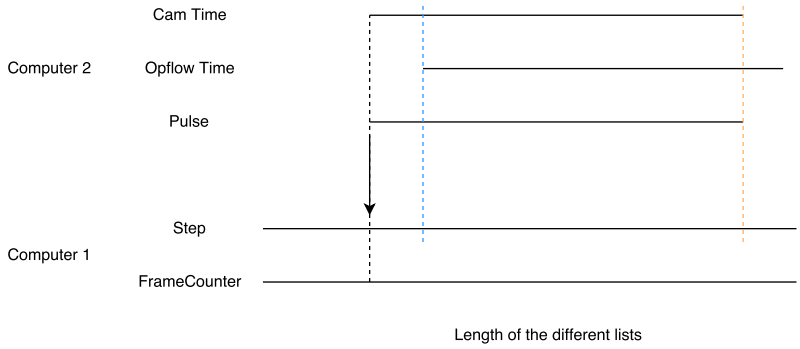

Figure 13: alignment signals

Opflow vector is opened and opflow time vector is managed with the opflowModulo() function. This function is needed because the opflowTimeFile list contains the time information every two positions in the list and the vector information every other two positions in the list. To retrieve only the timing information the time values on every pair positions must be extracted. The three TimeToTimeSyst() functions are used to modify the timing stored in the files to seconds.

GetStartAndStopVidTime() Function finds the start and stop timing of the final video that is going to be computed. The starting point is the time at which the behavior camera and

the optic flow measurements are taken at the same time (= latest time of the first optic flow measurements and first behavior frame acquisition time).

AdaptDataSampling() function samples the frameCntr, puff and step signals to 1500 data points per second. The StepOnTrue list had to be divided by the length adaptor factor because it is a list that contains the index values. The length adaptor is the ratio of the initial list length to the sampling factor (that is defined as points per second) times the video length in seconds.

GetIdx() function finds the start and stop indexes in the pulse list that correspond to the video start and stop time. The pulse list has the same length as the StepOnTrue list as this one refers to values in the step list that are above 9 and that - as previously explained - are linked to the pulse signal. Once the pulse indexes that correspond to the start and stop timing of the final video are found, the value at the pulse index position in the stepOnTrue list is stored as stepIdx. This stepIdx value refers to the index in the frame counter list of the corresponding fluorescence image.

The timeSec list is an interpolated list from the start timing point of the video to the stop timing point and contains the number of points equal to the sampling factor times the video length in seconds. Start and stop indexes of the opflow signal are also computed. The 3 rotations/second are computed from the optic flow raw measurements and interpolated to 1500 points per second.

GetFluoIdx() function manages the %DR/R fluorescence values. First the start and stop idx of first and last fluorescence values are found from the downsampled frameCntr list. The original %DR/R fluorescence values are interpolated to a list size of length equal to the end idx minus the start idx in the downsampled frameCntr list. At this time, we have an interpolated fluorescence %DR/R list containing the fluorescence values from all the fluorescence images and that is interpolated to the length of the difference between last and first index of the fluorescence images in the downsampled frameCntr list. The GetFluoIdx() function received the stepOnIdx and StepOffIdx that where calculated in getIdx (from the pulse start and stop idx). Therefore the fluorescence final list is taken between those two indexes (the start idx is stepOnIdx - GCStepOnIdx because the StepOnIdx was calculated for the entire downsampled frameCntr list and the current %DR/R was interpolated from the GCStepOnIdx position. If the stepOffIdx is higher than the GCStepOffIdx (which means that we don't have fluorescence values corresponding to the end of the movie), then NaN values are added to the end of the fluorescence trace.

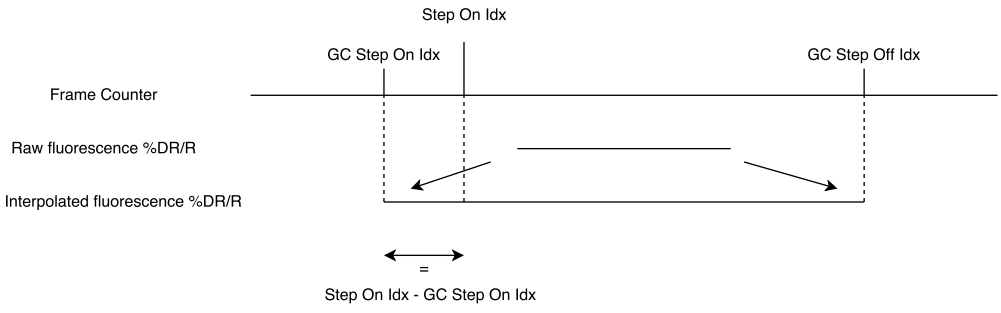

Figure 14: Fluorescence alignment

All the aligned signals are finally stored in a dictionary that will be used to create the frames for the final movie.

5.5 P5 - Movie Frames

The dictionary, fluorescence cropped ROIs and behavior images are first opened. The Puff data is then managed by finding and subtracting the mode value. Indexes of the subtracted puff values that are above 0.1 are stored and are segmented into 4 categories. Those puff categories characterize the puff intensities and will be represented in different red colors depending on the category in the final movie. The flow frame range that represents the number of datapoint to be plotted within 10 seconds frame is calculated and the start and stop idx of the behavior images is also found. The y axis limits are computed for the %DR/R traces and optic flow traces. Finally, the frames for the final movie are created with myFunc() function.