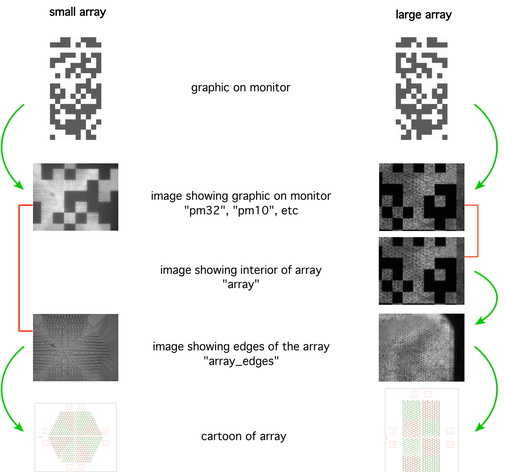
How to align the monitor with the array

This document explains how to use photographic mapping images to compute the alignment between the monitor and the array. If photographic mapping images are unavailable, the alignment can also be obtained from clicked points that approximate the corners of the array, but these are less accurate (see examples/monitor_alignment.m for an example of the procedure).

There are two routes to alignment, depending on the size of the array.



In these examples, images connected by a green arrow are ones for which the user selects alignment points. Images with a red connection are assumed to be in the same coordinates as each other. (Images A and B are in the "same coordinates" if each pixel of A represents the same physical x-y location as the corresponding pixel of B.)

There are three steps required for photographic mapping:

1.Identify photographic mapping images

For a small array, the user will specify at least two images, "pm#" (# = 32, 10, or 2) and "array_edges". Note that each value of # can have its own image. For a large array, the user must also specify the image "array" (more on this later).

These images are most easily specified as paths in an Index file. If relative, the paths are assumed to begin with `server_data_path`.

```
:photographic-mapping-pm32 "2007-09-18-4/slidebook/Image 2"
:photographic-mapping-pm2 "2007-09-18-4/slidebook/Image 3"
:photographic-mapping-array-edges "2007-09-18-4/slidebook/Image 1"
```

For those wary of atavism, images can also be loaded directly into datarun.

```
datarun.piece.photographic_mapping.images.pm# = <image>;
datarun.piece.photographic_mapping.images.array_edges = <image>;
```

2. select alignment points

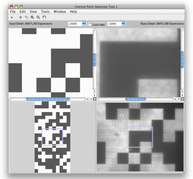
Once images are specified, execute this code:

```
datarun = compute_monitor_to_array_transformation(datarun);
```

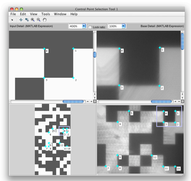
A GUI will appear for the user to select alignment points between the various images. The GUI is documented in the matlab help for the function cpselect.

The user's task is to select corresponding points in the two images. Start out by setting the zoom level of each image to make them approximately the same size. It's best to click on corners or intersections. First click in one image, then click the corresponding location in the other image. Points can be adjusted after clicking. The user must select at least 4 points in each pair of images. When finished, close the GUI window. A new GUI will appear for each pair of images to be aligned.

newborn GUI



after selecting points



The user must enter at least 4 points. Otherwise the code will crash, and all progress will be lost. Don't forget!

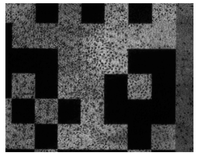
Be gentle with the GUI. Fast dragging movements or closing it too soon after it opens can cause it to crash. This is because Mathworks despises Macs.

To review the clicked points, or modify them, run the function again. The previously selected points will appear by default. To leave them untouched, close the GUI window. If you modify points but don't like your changes, cancel the execution in the command window.

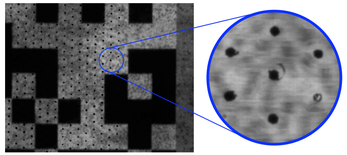
If multiple pm# images are specified in datarun, the user can choose which one to use for alignment with the argument "which_pm". One possible strategy for high res alignment is to first do a first pass using "pm32" (the default), then a second pass using "pm2".

For a large array, the photographic mapping image of the monitor ("pm#") is often too zoomed to see any corner of the array. This makes it difficult or impossible to figure out which electrode is which. To solve this problem, specify an intermediate image "array". This image is in the same coordinates as "pm#". For the image "array_edges", use a zoomed out image which shows at least one corner of the array. In most cases, there will be obvious landmarks (e.g. bad electrodes) to align "array_edges" with "array".

pm32

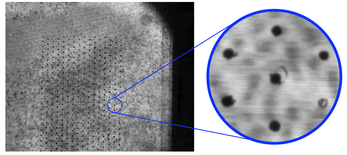


array

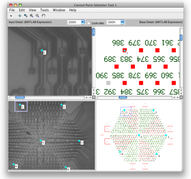


landmark to align array and array_edges

array_edges



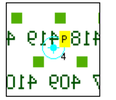
When the array image to array cartoon alignment appears, the image and the cartoon will be in the same orientation (assuming the rig and optical-path-direction are entered correctly). The user should click on corresponding electrodes in the two images. It's easiest to select corner electrodes.



corner electrodes are easy!

prediction tool

good prediction



bad prediction

The best strategy is to select three electrodes, then use the GUI's prediction tool to predict points. After selecting a few corner electrodes, click an interior electrode in the image of the array, and see where it is predicted to land in the array cartoon. If it lands squarely on an electrode, this indicates the previously clicked points are probably correct. If instead it lands between electrodes, there was probably an error in previous alignment points. If the predicted point looks good, it should still be tweaked to align perfectly with the array cartoon. Note that predicted points, if untweaked, do NOT count towards the four point minimum.

3. save the alignment

The alignment points can be saved by calling:

```
save_monitor_alignment(datarun);
```

and subsequently loaded by calling:

```
datarun = load_monitor_alignment(datarun);
```