**In Scanner Motion Tracking**

Data Acquisition and Processing

Meric Ozturk

[mrc.ozturk@gmail.com](mailto:mrc.ozturk@gmail.com)

Edward Cui

<cui23327@gmail.com>

Table of Contents

[Introduction 3](#_Toc392082886)

[Setting up for Image Acquisition: Outdated 4](#_Toc392082887)

[Analyzing a Motion Tracked Run 6](#_Toc392082888)

[Post-processing & Signal Characterization 15](#_Toc392082889)

[Appendix A: Script Breakdown 17](#_Toc392082890)

[Image Registration 17](#_Toc392082891)

[Signal Characterization 18](#_Toc392082892)

[Appendix B: Measuring Intercentroid Distance 19](#_Toc392082893)

[Appendix C: Optical Markers 20](#_Toc392082894)

# Introduction

This head tracking system was developed in order to capture in-scanner subject movement for magnetic resonance spectroscopy scans in which structure-based image registration is not possible. The system is comprised of two steps: image capture, which happens at the time of scan, and image analysis/post-processing, which happens sometime after scan through a suite of MATLAB scripts.

Image registration is accomplished through the use of optical markers (see **Appendix C**) taped on to a pair of scanner-safe goggles. Because the region of interest is close to the eye, we are able to use the same camera used for eye tracking without modification. To capture images, the composite feed from the eye tracking camera is put through a video digitizer; the resulting video stream is sampled at 5 frames per second through a third party application installed on the BIOPAC computer and resulting images are saved as jpegs to a specified folder.

Movement covariates are extracted from the captured images through the following algorithm:

1. Convert images to binary through a user provided binary threshold.
2. Automatically detect the two fiducial markers through Hausdorff distance. The resulting images will be cropped and saved.
3. Find fiduciary edges through an edge finding algorithm.
4. Find fiduciary centers and calculate displacement across time.
5. Calculate the number of pixels per millimeter in image space between the centroids of the two square markers. Then distance calculated based on number of pixels can be converted to displacement and speed in millimeters.

The algorithm is capable of tracking all three degrees of two-dimensional motion. For a well-captured image with good contrast and focus, expected error in the XY axes is on the order of a fifth of a millimeter; for angular motion, we expect around 3 degrees of noise. Because the subject’s angular motion is severely limited by the head coil, often the angular motion axes will consist of pure noise and can be thrown away.

**A note on data structure**: Image data for each subject should be located under *{project\_name}/subjects/***subject\_name**/**movement**. Images for a single subject and scan session should be further subdivided into runs. The MATLAB code assumes all frames of images in the given directory belong to a single run.

Setting up for Image Acquisition: Outdated

**Note: Do not adjust any part of the eye tracking camera before having Dennis give you an explanation of the camera’s parts and function.**

Capturing images for motion analysis requires the following hardware:



Motion Tracking Goggles

Composite to USB Video Digitizer

Composite / S-Video Attachment

Coax Cable with Composite Adaptor

To prepare the hardware for image capture:

1. Turn on the eye tracker control unit and the camera power supply. Boot up the BIOPAC computer if it’s not already running.

Power supply to the eye tracker camera.

The eye tracker’s control unit.

1. Attach the composite cable attachment to the USB video digitizer and plug it in to one of the USB slots on the BIOPAC machine. Start motion capture software **CaptureFlux** (shortcut on the desktop). **Under the “Video Source” option, select “Hauppauge”. Under “Audio Source”, select “No Audio”. Under “Capture Format” select “Image”. Under “images / sec”, enter “5”.**



1. On the back of the eye tracker control unit, there is a T-junction labeled with orange tape (pic). **Disconnect the orange coax cable** **and plug in your coax cable**. **Attach the composite adaptor** to the other end of the cable, and **plug into the yellow composite video jack of the USB video digitizer.**
2. Go to the scanner room and **turn on the camera bulb and preview monitor**. Place the **all-pass attenuating filter** (NOT the IR filter) in front of the bulb. Focus the camera, **making sure the black squares have crisp edges**. Instruct the subject to move until the **goggles are centered in-frame and the 10mm feature is visible**. Adjust the camera’s aperture to **maximize contrast between the black rectangles and white background** of the goggles. Go back to the control room and **confirm the feed is visible on the BIOPAC computer**. **Turn off the preview monitor** to limit EMF interference**.**
3. **Confirm software settings** (on top right, 6th button) for CaptureFlux match up with the settings below:

* Under “DV”: Frequency = 4800; Standard = NTSC; Ratio = 16x9; check “Keep Native Freq.”; uncheck “Create scn file”
* Under “Images”: choose “jpeg”; quality at 100; check “Always deinterlace”

1. **Confirm that video overlay of time/date is turned off** (on top right, 5th button).

You are now ready to capture images. **To start grabbing images for a run**:

* Choose the image save dir (**use one dir / run**). Chosen from 2nd button, top right.
* Go to the **Intervalometer** (4th button, top right). On the bottom right, make sure input boxes read “6 frames”. This will **sample at 5Hz** since NTSC is 30 frames/sec.
* To start /stop acquisition, click on the big Intervalometer button. **Make sure to synchronize your acquisition’s start and stop times with the scanner.**

**Clean up: Plug orange coax cable back into control unit, turn off bulb, camera and control unit power supply, and note session duration on the camera log.**

Motion covariates extracted from these images are only as good as the images themselves. To ensure data integrity during analysis and post-processing, make sure you are always capturing the best quality image possible. Image quality is most affected by:

* **Lighting**: Because the unfiltered camera bulb is too bright for comfort, we have to use the attenuating all-pass filter to decrease light intensity. Make sure you are using the correct filter; the other filter will pass no visible light and greatly degrade image quality. We’ve also found that **turning on the in-scanner lights greatly increases image quality**.
* **Aperture**: The camera’s aperture controls how much light enters the camera. If the aperture is too closed, the entire image will be dark, degrading fiduciary contrast. If it is too open, the entire image will appear washed out, degrading fiduciary contrast. Try to adjust the aperture to maximize fiduciary contrast.
* **Focus:** A well-focused image will have sharp, crisp edges. An unfocused image will introduce noise into your movement covariates.

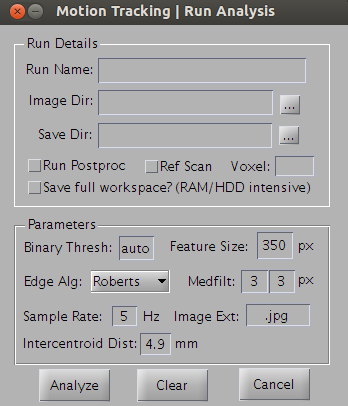
Analyzing a Motion Tracked Run

The image\_regsuite of scripts is used to extract movement covariates from run images and is comprised of a main script the user interacts with, and supporting scripts called by the main script as needed. The main script is in ./image\_reg/imreg\_preproc.m

The rest of the code is in the ./image\_reg/imreg\_code/subdirectory; see **Appendix A** for descriptions of what each of these scripts is responsible for.

**How to analyze images from a run:**

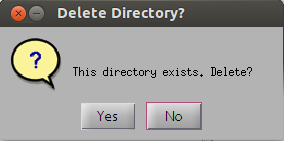
1. Run MATLAB and navigate to image\_reg*.*This path should be MATLAB’s working directory whenever starting a new analysis.
2. Type imreg\_preproc into the command window. The below GUI will open:



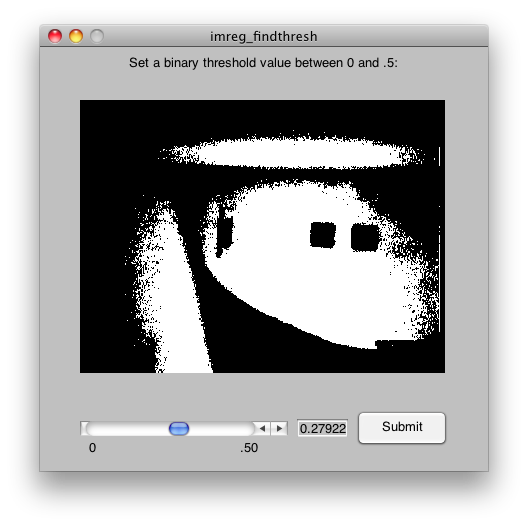
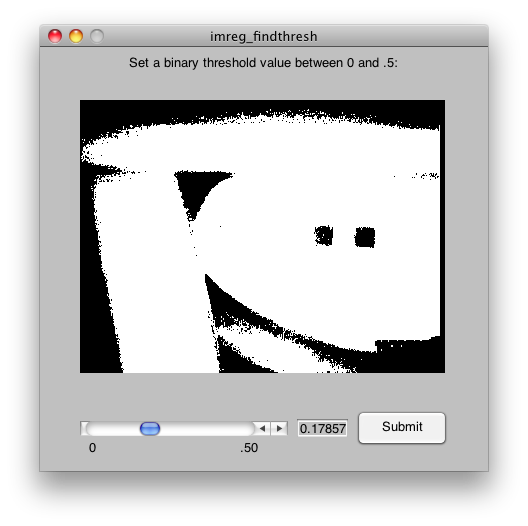
The GUI makes available the following parameters. **Required are in red:**

* **Run Name**: This determines the name of the analysis folder to be created. Name it with a run identifier and ‘analysis’ (e.g. run\_{number}\_analysis). The program will append the date to the run name to create the folder name.
* **Image Dir**: Directory the images are stored in. Click the ellipses on the right to open the directory chooser. Point to the run directory the images are stored in (./subject\_name/movement/run\_name).
* **Save Dir:** Directory in which we will save the analysis folder. Click the ellipses on the right to open the directory chooser. Point the save directory to the folder movement data is stored in (./subject\_name/movement).
* **Run Postproc:** Checking this box tells the script that this is the last run for this scan session, and that you would like to open the post-processing script after analysis ends. The post-processing script can also be opened from the MATLAB command line (see the section on post processing).
* **Ref Scan:** Is this scan a reference scan? Usually this run corresponds to a structural scan that helps localize the brain region to put voxels for later MRS runs. Make sure to check this box when processing a run corresponding to a structural scan.
* **Voxel:** which voxel is localized/used in current run? Enter the brain region used, e.g. v1, mfg, etc.
* **Save Workspace**: If checked, this will save every variable in the workspace to the analysis folder. Because it will save images at every stage of the analysis, it is extremely memory and hard disk intensive. If left unchecked, it will only save the results variables from the workspace. There is no need to check this box when running an analysis; it is around for debugging reasons.
* **Binary Thresh**: The binary threshold used in converting the grayscale images to black and white. Notice the default entry is “auto”; this will allow you to use the binary thresholding tool. Leaving this option on “auto” is generally your best bet. There is not really a good number for thresholds, as the image quality can vary quite a bit. See the following section for more detail.
* **Feature Size:** Specifies the area (in pixels) an image feature must be to be considered “not noise”.
* **Edge Alg**: Specifies the edge finding algorithm to be used by the imreg\_edges script. The default algorithm is generally fine to use.
* **Medfilt:** This specifies the smoothing kernel, using ‘medfilt2’ function. The first value is for rows (horizontal), and the second value is for columns (vertiacal). Default is 3x3 smoothing. This usually helps reduce artifacts in the image frames.
* **Sample Rate**: Specifies how many images are captured per second. If you did not change anything in the acquisition (5 frames per second), you can leave this as default.
* **Image Ext**: image extension that the script will look for in the “Image Dir” folder and load. Default is ‘.jpg’.
* **Intercentroid Dist**: the distance in millimeters between the two squares in the fiducial marker, measured by a caliper. Current default is 4.9mm.

1. After clicking “Analyze”, a project directory is going to be created under “Save Dir”, with run name and date appended. If the directory already exists, a pop-up dialogue is going to show up and ask you if you want to delete the existing directory and/or to reprocess.



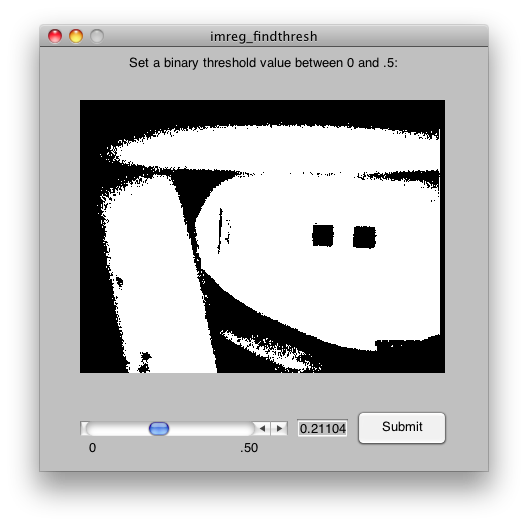
1. Next, you will need to choose a binary threshold using the binary threshold tool:



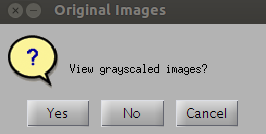
The threshold tool features a GUI slider, which can take on a thresholding value between 0 and 0.5. Dragging the slider or clicking the left and right arrow next to the slider changes the image in real time; adjust the slider until an acceptable image is found. General rule of thumb: we want the two dark squares to be as uniform and dark as possible, but we want them to be surrounded by a sea of white, with little noise.

We can see from the image on the left (above) that the threshold was set too low; a lower threshold makes a more selective image, and in this case the threshold is so selective it has begun to deteriorate our features of interest (especially the right square). Conversely, the right image has a threshold that is too high; we can see a lot of noise creeping in.

Here is an example of a well thresholded image. Click “Submit” button once you find a good threshold.

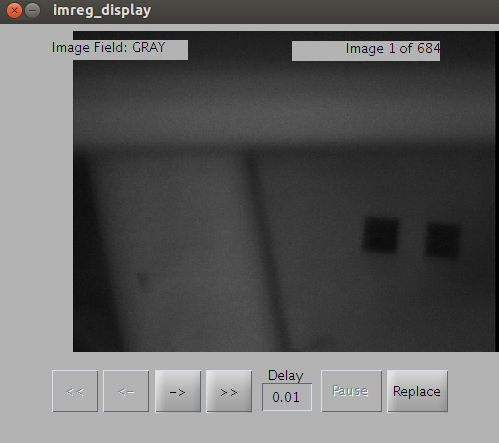


1. Now, the script is going to load all the images into the memory, which may take a while depends on the size of the image and the computer. After the loading, a pop-up dialogue will show up and ask you to view the loaded gray-scale image. It is usually not necessary, although it could be helpful to identify artifacts during reprocessing. If this is the first run, click on “No” to continue.



Click on “Yes” will open an image viewer.

**A quick primer on the image viewer**: Clicking the single arrows (“<-”and “->”) will cycle through the images one by one, while clicking the double arrows (“<<” and “>>”) will cycle through the images automatically with a delay between images given in the “delay” text box. Cycling can be stopped by clicking on the “Pause” button. Clicking “Replace” will replace the current image with the previous image; if you have terrible, unsalvageable frames or frames where the subject moved off camera, you can use this functionality to smooth over these unsalvageable frames.



1. Next, the script will automatically segment the squares. This usually takes a while if the images are large. At the end of this step, a pop-up dialogue will show up and ask you to view the segmented images. Definitely click on “Yes”, as this step is an important checkpoint and later calculations of movement parameters entirely depend on whether this step is successful or not.

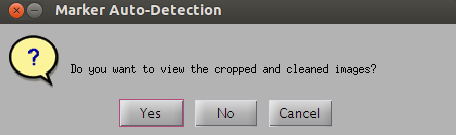
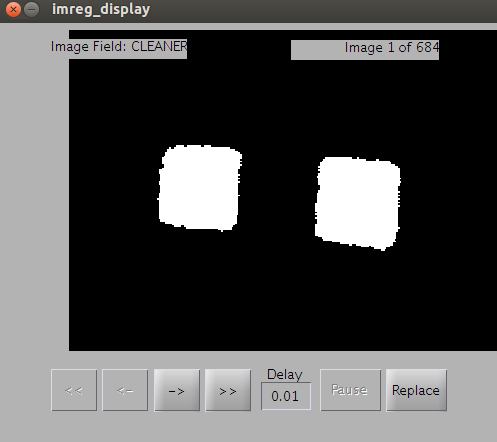
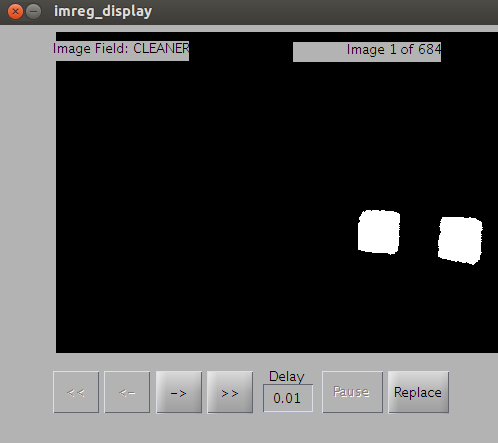


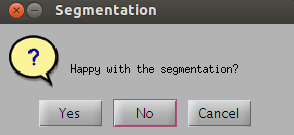
Image viewer will show up again and display the segmented binary image as the following:



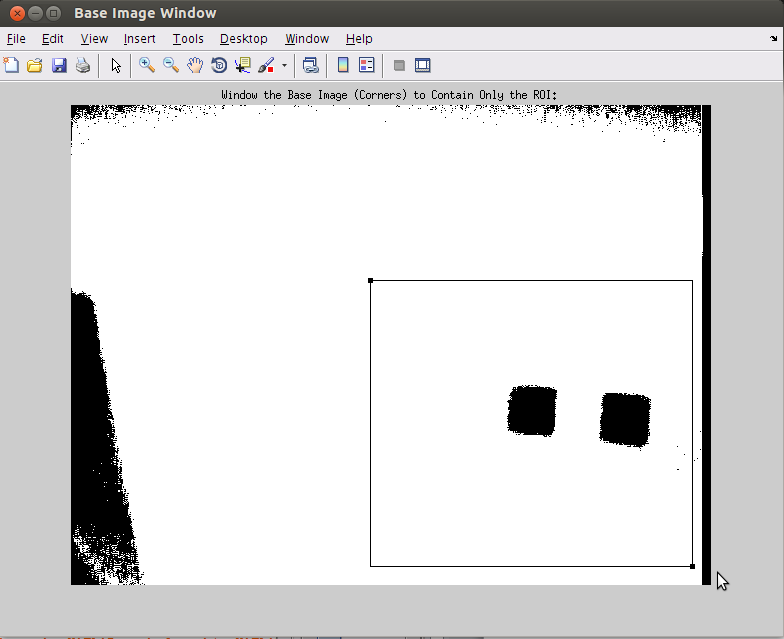
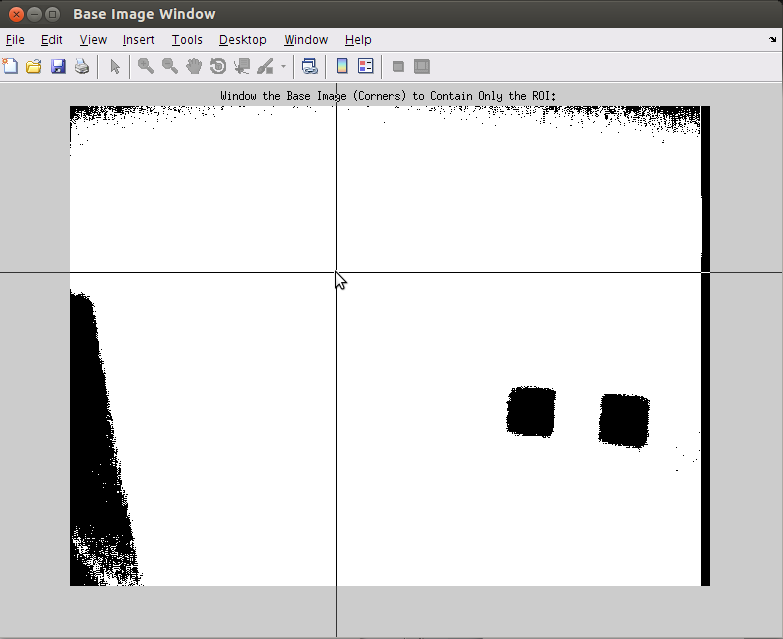
Cycle through to make sure all the frames ONLY have the two squares and no other artifacts. *Here is a rule of thumb that may be helpful: upon seeing the first frame as shown above, if the two squares are around the center of the image, it is likely that there will be no artifacts in other frames. If the squares somehow cluster at a corner, whereas large area of the frame is empty/black, then it usually means there are artifacts in later frames (shown in the figure below). But note that this would assume that the subject did not move a lot. If the subject moved a lot, then this wouldn’t be true. In general, we expect subject not moving too much.* **Regardless of the heuristics, ALWAYS cycle through to examine every frame.**



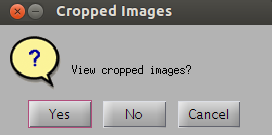
1. Next, a pop-up dialogue will show up and ask you to confirm whether the segmentation is good or not. If the segmentation is done correctly, choose “Yes”. If not, choose “No”.



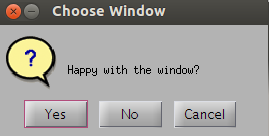
Upon choosing “No”, a window will show up and ask you crop the image based on the first frame.



A pop-up dialogue will show up and ask you to view the cropped gray-scale image. Choose “Yes” and see if all the markers are captured within the gray-scale image.

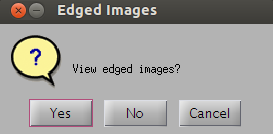
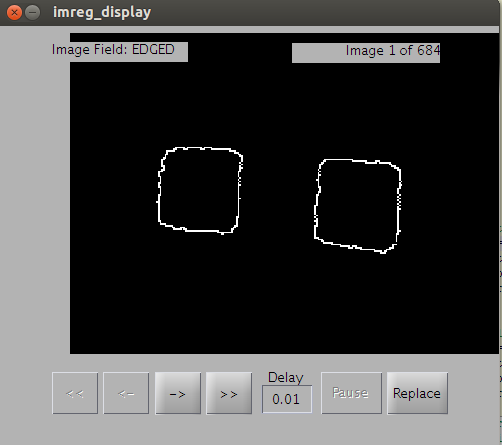


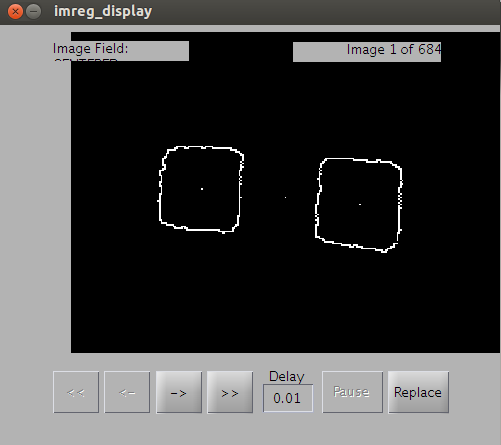
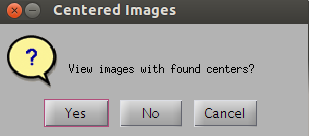
After viewing the cropped images, another diaglogue will show up and ask you to confirm whether all the frames capture the fiducial marker. Choose “Yes” if so; choose “No” to re-crop the images.



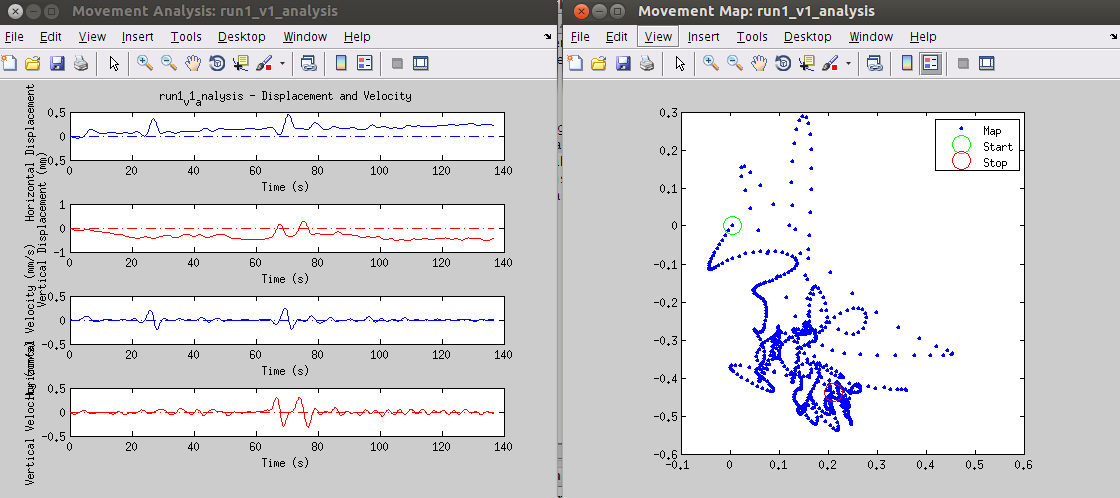
The images will be re-segmented and the process will be repeated between Step 6 and 7 until the user confirm that the segmentation is good.

1. If the segmentation has been successful, the script will proceed to the next step and find edges and centroids of the fiducial marker. This allows the user to visually confirm that the position of the head captured by the markers.



Usually, it is not necessary to view the edged images, as the edges can be seen also in the images with centers dotted.

1. Finally, the script will calculate displacement, as well as its first derivative (velocity), of movement. The final results are presented in two ways. In the figure below, the left figure shows a time series plot of each movement parameter (displacement and velocity) at vertical and horizontal direction. The figure on the right shows a foot-print of the fiducial marker on the 2D surface, with starting point circled green, and stopping point circled red.
2. After a completed analysis, the analysis folder should have the following items in it:

* **log.txt**: The log displays the status of the analysis at each stage and also lists parameters for the run.
* **movement\_graph.fig**: This MATLAB figure displays the movement covariates in graph form.
* **movement\_map**.fig: This MATLAB figure displays the foot-print trace of movement.
* **movement\_graph**.**jpg**: An exported image of the MATLAB figure above. You will need to create this .jpg yourself after editing the MATLAB figure to your needs.
* **movement\_map**.jpg: similar to movement\_graph.jpg
* **movement\_values.csv**: A comma-separated value file of the movement covariates so they can be opened in Excel. The columns are separated as follows:

|  |  |  |  |
| --- | --- | --- | --- |
| Time | X Axis (horizontal) Movement | Y Axis (vertical) Movement | Rotational Movement |

The time vector is in seconds, the x and y vectors are in mm, and the rotational movement is in degrees.

* **short\_workspace.mat**: A limited MATLAB workspace which contains the run parameters structure (params), processed images structure (images), and all of the movement covariates.

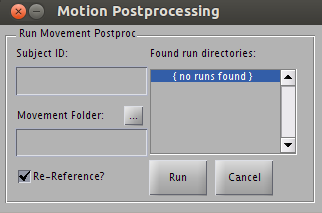
Congratulations, you’ve successfully completed a run! Each run with around 1700 images should take about 5~10 minutes (if image frames are free of artifacts, it might take less than that time), depends on the size of the image and the computer used.

After finishing the last run of a scan, you have the option of running the signal characterization script (“imreg\_postproc”).

Post-processing & Signal Characterization

Post-processing refers to running a signal characterization algorithm on a run’s movement covariates in order to summarize the signal. There are several ways used to calculate the summary. Consider the time series of movement as a high dimensional variable, “Size” indicates Euclidean distance of the time series. “Jaggedness”, norm of the first derivative of the signal, can describe instantaneous movement (velocity). Signal size looks only at the amount of movement, while jaggedness takes into account both signal amplitude and frequency in equal measure. Both, however, assume that, across runs and subjects, there are equal numbers of time points, which may not usually be the case. Hence, there comes root-mean-square (RMS) summary of the time series, which is similar to “size”, but takes the average of the time series point, to remove the effect of duration. .

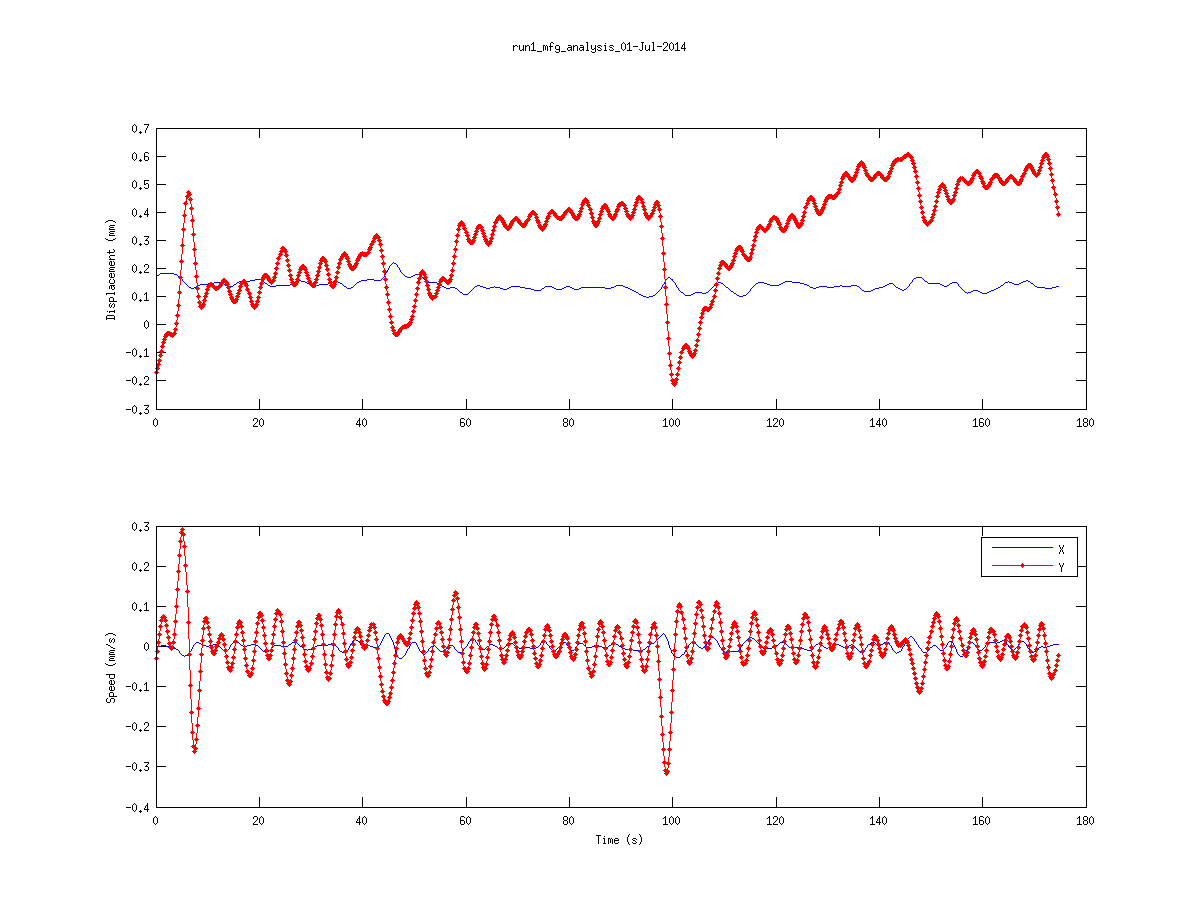
If you are about to analyze the last run of a scan session for movement and would like to run the session for characterization afterwards, check the “Run Postproc” box; imreg\_preproc.m will automatically fill in the right directory name when starting the script. Note that the supporting codes are also in .image\_reg/imreg\_code/. You can also start the script from the command line by calling “imreg\_postproc”- either way this GUI will pop up:



After choosing a directory the script will find all the analysis directories in the movement folder given (it does this by checking to see if they include short\_workspace.mat). Make sure the “Re-Reference?” box is ticked. It should be ticked by default, but it never hurts to double check.

The "run name" given to each run is determined by the first substring of the run folder name, so if your run folder is named run3\_v1\_analysis the run name in the csv will be run3; if it is named run3-v1\_analysis it will be named run3-v1.

Postprocessing will recalculate frame of reference based on structural runs (indicated in preprocessing) and adjust displacement of movement (velocity/speed should not be affected). It will then plot a time series figure for each adjusted run and save them as .png file in a folder “reference\_corrected\_figures” under the movement folder selected, e.g. the following:



Notice that for displacement (top), time series no longer start at 0, but shifted, since we are accounting movement relative to the average position of structural runs. X indicates horizontal movement, and Y indicates vertical movement.

When run, the script creates and saves a .csv file into the movement directory named <subject-id>\_rereferenced\_movement.csv which contains all the movement summary measure run by run, so that data can be easily imported into Excel. It also saves a .mat with some workspace variables in case we have to do something else with them later on. A log file, named “post\_proc\_log.txt” will be saved as well to record messages printed on the screen during postprocessing.

Appendix A: Script Breakdown

Image Registration

**imreg\_preproc.m:** This is the main script; all user interaction occurs with this script and this script is in charge of initializing variables and calling all other scripts. It also manages the runtime log and returns any errors to the user.

**imreg\_gui.m/fig**:Figure and backend code for the main analysis window GUI. The default values shown upon the start of imreg\_preproc can be changed here. Use guide command to open and edit the GUI. Change places in the .m script corresponding to the value boxes as well.

**imreg\_findthresh.m/fig**: Figure and backend code for the binary threshold finder tool.

**imreg\_readims.m**: This function initializes the images container and reads in and grayscales all of the jpeg images available in the images directory, in ASCII order.

**imreg\_display.m/fig**: Figure and backend code for the image display tool.

**imreg\_segment.m**: This function contains the main algorithm to extract double square features from the image.

**imreg\_window.m**: This function presents the user with the first image in the series and allows the user to specify the region of interest. It then crops all of the images down to this ROI; all of the analysis after this point will be run on this ROI. This drastically cuts down on processing time and memory consumption.

**imreg\_edges.m**: This function takes the cleaned images and finds the feature edges using the user-specified edge-finding algorithm.

**imreg\_center.m**: This function takes the binary images and uses kmeans cluster algorithm to find their pixel center locations. The pixel center of the overall image is used to calculate displacement and pixel centers of individual features are used to calculate angular displacement.

**imreg\_px2mm.m**: This function uses inter-centroid distance in number of pixels and in millimeters (specified in the GUI and params structure) to find a conversion factor to convert displacement from number of pixels to millimeters.

**imreg\_calculate.m**:This function is used to calculate translational and rotational displacement from the first image using pixel centers found in imreg\_center.m.

**imreg\_butterfilter.m:** This function applies Butterworth filter to displacement and velocity time series, which reduces noise in the time series.

Signal Characterization

**imreg\_postproc.m**: The main signal characterization script; it calls all other scripts, formats the .csv and .mat outputs and saves both files. It does all folders / runs navigation.

**postproc\_gui.m/fig**: Figure and backend code for the main post-processing GUI.

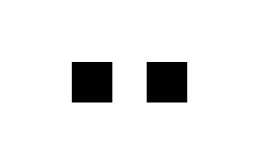
**postproc\_correct\_reference\_disp.m**: Readjust movement to another reference point, usually to a structural run. Velocity/Speed is not touched.

**postproc\_char\_signal.m**:This script runs signal characterization algorithms to output summary measures of movement.

**postproc\_replaceNaN.m**: replace NaN values in a cell to a different value, usually 0.

**postproc\_cell2csv.m**: A MATLAB Exchange script written by Sylvain Fiedler which converts MATLAB cell to a .csv file. Used to write the final .csv file for the worksheet.

# Appendix B: Measuring Intercentroid Distance



X

Y

Use a caliper to measure X and Y shown in the picture. Then, inter-centroid distance

Make sure to measure multiple times and determine the inter-centroid distance by average.

Appendix C: Optical Markers

Macintosh HD:Users:meric:Desktop:goggles4.pdfOptical markers used on the scanner safe goggles are below; for best results, print this page at high quality and resolution. Use double-sided tape to secure to goggles.