LOADING TOOLBOXES

```
ft_defaults
bml_defaults
format long
clear
```

DEFINING PATHS

```
SUBJECT='DM10XX';
SESSION='intraop';

% this first part of this script is not task-specific
% TASK='lombard';

PATH_DATASET = 'Y:\DBS';
PATH_DER = [PATH_DATASET filesep 'derivatives'];
PATH_DER_SUB = [PATH_DER filesep 'sub-' SUBJECT];
PATH_PREPROC = [PATH_DER_SUB filesep 'preproc'];
PATH_ANNOT = [PATH_DER_SUB filesep 'annot'];
PATH_SRC = [PATH_DATASET filesep 'sourcedata'];
PATH_SRC_SUB = [PATH_SRC filesep 'sub-' SUBJECT];
PATH_RIPPLE = [PATH_SRC_SUB filesep 'ses-intraop' filesep 'ripple'];
PATH_ANALOG = [PATH_SRC_SUB filesep 'ses-intraop' filesep 'analog'];
PATH_ALHAOMEGA = [PATH_SRC_SUB filesep 'ses-intraop' filesep 'alphaomega']; %path to ne
PATH_ALHAOMEGA_MAT = [PATH_ALHAOMEGA]; %path to neuroomega's converted mat files.
PATH_AUDIO_DER = [PATH_DER_SUB filesep 'aec']; %path to acoustic echo cancelling folde
PATHS_AUDIO_SRC = strcat(PATH_SRC_SUB,filesep,{'ses-training';'ses-preop';'ses-intraop
PATHS_TASK = strcat(PATH_SRC_SUB,filesep,{'ses-training';'ses-preop';'ses-intraop'},fi

% loading annotation tables
events_files = bml_annot_read_tsv([PATH_ANNOT filesep 'sub-' SUBJECT '_events-files.ts
```

```
Error using readtable
Unable to find or open 'Y:\DBS/derivatives/sub-DM10XX/annot/sub-DM10XX_events-files.tsv'. Check the
path and filename or file permissions.

Error in bml_annot_read_tsv (line 23)
annot = readtable(filename,varargin{:});
```

```
% events_files.path = strrep(events_files.path,'/Volumes/Nexus4','Y:');
session = bml_annot_read_tsv([PATH_ANNOT filesep 'sub-' SUBJECT '_sessions.tsv']);
runs = bml_annot_read_tsv([PATH_ANNOT filesep 'sub-' SUBJECT '_runs.tsv']);
cd(PATH_PREPROC)
```

LOADING HEADER INFORMATION

```
cfg=[];
cfg.path=PATH_RIPPLE;
cfg.pattern='*.nev';
cfg.filetype='trellis.nev';
```

```
info_nev=bml_info_raw(cfg);
info_nev.basename = strrep(info_nev.name,'.nev','');

cfg=[];
cfg.path=PATH_RIPPLE;
cfg.pattern='*.ns5';
cfg.filetype='trellis.ns5';
info_ns5=bml_info_raw(cfg);
info_ns5.basename = strrep(info_ns5.name,'.ns5','');

cfg=[];
cfg.path=PATH_RIPPLE;
cfg.pattern='*.nf3';
cfg.filetype='trellis.nf3';
info_nf3=bml_info_raw(cfg);
info_nf3.chantype(:) = {'hires'};
info_nf3.basename = strrep(info_nf3.name,'.nf3','');

cfg=[];
cfg.path=PATH_RIPPLE;
cfg.pattern='*.nf6';
cfg.filetype='trellis.nf6';
info_nf6=bml_info_raw(cfg);
info_nf6.chantype(:) = {'hifreq'};
info_nf6.basename = strrep(info_nf6.name,'.nf6','');

cfg=[];
cfg.mpx_path=PATH_ALHAOMEGA;
cfg.path=PATH_ALHAOMEGA_MAT;
cfg.filetype='neuroomega.mat';
cfg.chantype='analog';
info_neuroomega=bml_neuroomega_info_raw(cfg);
info_neuroomega.basename = strrep(info_neuroomega.name,'.mat','');

info_wav_src=[];
info_wav_src_renamed=[];
for i=1:length(PATHS_AUDIO_SRC)
    cfg=[];
    cfg.path=PATHS_AUDIO_SRC{i};
    cfg.pattern='ZOOM*/*.wav';
    cfg.filetype='audio.wav';
    info_wav_src=bml_annot_rowbind(info_wav_src,bml_info_raw(cfg));

cfg.pattern='sub-*.wav';
    info_wav_src_renamed=bml_annot_rowbind(info_wav_src_renamed,bml_info_raw(cfg));
end

info_wav_src.basename = strrep(info_wav_src.name,'.WAV','');
info_wav_src_renamed.basename = strrep(info_wav_src_renamed.name,'.wav','');
cfg=[];
```

```
cfg.path=PATH_AUDIO_DER;
cfg.pattern='*.wav';
cfg.filetype='audio.wav';
info_wav=bml_info_raw(cfg);
info_wav.basename = strrep(info_wav.name,'.wav','');
info_wav = info_wav(~ismember(info_wav.name,info_wav_src_renamed.name),:);
info_wav = bml_annot_rowbind(info_wav_src_renamed,info_wav);


% Saving files information table with OS times
info_orig = bml_annot_rowbind(info_ns5,info_nf3,info_nf6,info_nev,info_neuroomega,info_
bml_annot_write_tsv(info_orig, [PATH_ANNOT filesep 'sub-' SUBJECT '_ses-' SESSION '_da
```

ALIGNING TRELLIS FILES TO EVENTS BY SCRIPTS (NO TIME WARPING)

looping through nev files

```
info_nev.delta_t(:) = nan;
info_nev.min_cost(:) = nan;
info_nev.n(:) = nan;
for i=1:height(info_nev)
    %selecting events files that correspond to this nev file. Doing this by
    %coarse time

    nev_events_files = bml_annot_filter(events_files,info_nev(i,:));
    if ~isempty(nev_events_files)
        task_events = table();
        for j = 1:height(nev_events_files)
            % changed LB 20231019, sync file and lombard file couldn't be
            % concatenated because sync.stim_file was all nans
            events_tmp = bml_annot_read_tsv(fullfile(nev_events_files.path{j},nev_event
            if isnumeric(events_tmp.stim_file) && all(isnan(events_tmp.stim_file)); ev
            task_events = bml_annot_rowbind(task_events, events_tmp);
        end
        %loading events from nev file
        E = ft_read_event(fullfile(info_nev.folder{i},info_nev.name{i}),'detectflank',
        hdr = ft_read_header(fullfile(info_nev.folder{i},info_nev.name{i}));
        cfg=[]; cfg.Fs = 1; %already in seconds
        cfg.starts = info_nev.starts(i);
        nev = bml_event2annot(cfg,E);
        nev.value = nev.value - 1;
        nev.sample = round(nev.starts .* hdr.Fs);

        % plot(nev.starts,nev.starts - task_events.starts)

        %Aligning blackrock to presenation task to get GTC. No timewarping.
        cfg=[];
        cfg.scan = 100;
        [slave_delta_t, min_cost, warpfactor] = bml_timealign_annot(cfg, task_events,
```

```
        info_nev.delta_t(i)=slave_delta_t;
        info_nev.min_cost(i)=min_cost;
        info_nev.n(i)=height(nev);

        fprintf('%f sec, min cost = %f\n', slave_delta_t, min_cost)
    end
end
%adjusting ripple files
info_nev.delta_t(ismissing(info_nev.delta_t)) = 0;
cfg.keys = 'basename';
info_ns5 = bml_annot_left_join(cfg,info_ns5,info_nev(:,{'basename','delta_t'}));
info_nf3 = bml_annot_left_join(cfg,info_nf3,info_nev(:,{'basename','delta_t'}));
info_nf6 = bml_annot_left_join(cfg,info_nf6,info_nev(:,{'basename','delta_t'}));
info_nev.starts = info_nev.starts + info_nev.delta_t;
info_nev.ends = info_nev.ends + info_nev.delta_t;
info_ns5.starts = info_ns5.starts + info_ns5.delta_t;
info_ns5.ends = info_ns5.ends + info_ns5.delta_t;
info_nf3.starts = info_nf3.starts + info_nf3.delta_t;
info_nf3.ends = info_nf3.ends + info_nf3.delta_t;
info_nf6.starts = info_nf6.starts + info_nf6.delta_t;
info_nf6.ends = info_nf6.ends + info_nf6.delta_t;
roi_nev = bml_roi_table(info_nev,'trellis');
roi_ns5 = bml_roi_table(info_ns5,'trellis');
roi_nf3 = bml_roi_table(info_nf3,'trellis');
roi_nf6 = bml_roi_table(info_nf6,'trellis');
roi_ripple = bml_annot_rowbind(roi_nev, roi_ns5, roi_nf3, roi_nf6);
%ripple time after solid translation to GTC is master time
sync_ripple = roi_ripple;
bml_annot_write_tsv(sync_ripple, [PATH_ANNOT filesep 'sub-' SUBJECT '_ses-' SESSION  '
```

ADDING TIME FROM ORIGINAL WAV FILES TO DERIVED WAV FILES

```
info_wav_src_times = info_wav_src(endsWith(info_wav_src.name,'LR.WAV'),:);
info_wav_src_times.duration_round = round(info_wav_src_times.duration,1);
info_wav_src_times.starts_src = info_wav_src_times.starts;
info_wav_src_times.date_src = info_wav_src_times.date;
info_wav_src_times.datenum_src = info_wav_src_times.datenum;
info_wav.duration_round = round(info_wav.duration,1);
%info_wav = info_wav(:,setdiff(info_wav.Properties.VariableNames,info_date_vars))

cfg=[];
cfg.keys={'duration_round'};
cfg.select = {'starts_src','date_src','datenum_src'};
info_wav = bml_annot_left_join(cfg,info_wav,info_wav_src_times);
info_wav.starts = info_wav.starts_src;
info_wav.date = info_wav.date_src;
info_wav.datenum = info_wav.datenum_src;
info_wav.ends = info_wav.starts + info_wav.duration;
info_wav.id = [];
info_wav = bml_annot_table(info_wav);
```

COARSE ALIGNMENT

```
roi_neuroomega = bml_roi_table(info_neuroomega,'neuroomega');
roi_wav = bml_roi_table(info_wav,'audio');

roi = bml_annot_rowbind(roi_neuroomega, roi_ripple, roi_wav);
% plot initial coarse alignment
figure;
cfg=[];
cfg.facet="filetype";
bml_annot_plot(cfg,roi,'LineWidth',3)
```

FIND LANDMARKS IN AUDIO

The time in the zoom recorder is not guaranteed to have the same reference across runs (battery, change, time resets, etc). Might need to mark landmarks for every run.

```
%selecting files to explore from runs table
landmarks_task = 'lombard'; % which task will be used to find audio landmarks?
task_runs = runs(strcmp(runs.task, landmarks_task) & strcmp(runs.session,'intraop'),:)

i = 2; % MANUALLY select which run index to find landmarks in

fprintf('run idx =%d\n',task_runs.run(i))
cfg=[]; %ns5
cfg.roi=roi_ns5(contains(roi_ns5.name, sprintf('%s',string(task_runs.trellis(i)))),:);
cfg.channel='analog 2'; %'analog 2'=stimulus audio, 'analog 30'=produced audio
ns5_audio = bml_load_continuous(cfg);
cfg.roi.name
bml_praat(ns5_audio);

cfg=[]; %ZoomH6
cfg.roi=roi_wav(contains(roi_wav.name, ['sub-' SUBJECT '_ses-intraop_task-' landmarks_
wav_audio = bml_load_continuous(cfg);
cfg.roi.name
bml_praat(wav_audio);

cfg=[]; %alphaomega
cfg.roi=roi_neuroomega(contains(roi_neuroomega.name,sprintf('%.3f',lombard_runs.MER_de
cfg.roi=cfg.roi(2, :); % you might have to tweak this number manually
wav_audio = bml_load_continuous(cfg);
cfg.roi.name
bml_praat(wav_audio);
```

Create table manually

- eg, **sub-DM10XX_ses-intraop_task-lombard_annot-audio-landmarks.tsv**

- find EXACTLY ONE landmark per data stream-eg, don't find landmarks for two Lombard runs
- task run filetype name landmark_time

ADJUST ROIs ACCORDING TO LANDMARKS

```
landmarks = bml_annot_read_tsv([PATH_ANNOT filesep 'sub-' SUBJECT '_ses-intraop_task-'


%get current t0 for landmark files
landmarks.t0 = bml_map(landmarks.name,roi.name,roi.starts,NaN);

% adding reference to transform to global time coordinates
landmarks.landmark_gtc = landmarks.landmark_time + landmarks.t0;
landmarks_wide = unstack(landmarks(:,{'run','filetype','landmark_gtc'}),'landmark_gtc'

% calculating corrections
landmarks_wide.neuroomega_delta = landmarks_wide.trellis_ns5 - landmarks_wide.neuroome
landmarks_wide.audio_delta = landmarks_wide.trellis_ns5 - landmarks_wide.audio_wav;


%correcting info tables using ns5 as master time
cinfo_neuroomega=info_neuroomega;
cinfo_neuroomega.starts = info_neuroomega.starts + nanmean(landmarks_wide.neuroomega_d
cinfo_neuroomega.ends = info_neuroomega.ends + nanmean(landmarks_wide.neuroomega_delta

cinfo_wav=info_wav;
cinfo_wav.starts = info_wav.starts + nanmean(landmarks_wide.audio_delta);
cinfo_wav.ends = info_wav.ends + nanmean(landmarks_wide.audio_delta);

% transforming to ROI tables
roi_neuroomega = bml_roi_table(cinfo_neuroomega,'neuroomega');
roi_wav = bml_roi_table(cinfo_wav,'audio');

% merging roi tables
roi = bml_annot_rowbind(roi_neuroomega, roi_ripple, roi_wav);
```

PLOT AND VERIFY COARSE ALIGNMENTS. SAVE FIGURE

```
f=figure();
cfg=[];
cfg.facet="filetype";
bml_annot_plot(cfg,roi,'LineWidth',3)
saveas(f,['figures/sub-' SUBJECT '_ses-' SESSION '_proto-A04-coarse-sync.png']);
bml_annot_write_tsv(roi, [PATH_ANNOT filesep 'sub-' SUBJECT '_ses-' SESSION '_data-fil
```

DETECT STRETCHES CONSTANT DEPTH IN ALPHA OMEGA

```
cfg=[];
cfg.criterion = @(x) length(unique(x.depth))==1 && abs((max(x.ends)-min(x.starts))-sum
neuro_cons_depth = bml_annot_consolidate(cfg,roi_neuroomega);
ao_session = neuro_cons_depth(neuro_cons_depth.duration > 30,:);
ao_session = ao_session(:,{'starts','ends','depth'});
bml_annot_write_tsv(ao_session,[PATH_ANNOT filesep 'sub-' SUBJECT '_ses-' SESSION '_sta
```

LOAD MASTER EVENTS

```
%loading master events from all nev files
master_events = table();
for i=1:height(info_nev)

    E=ft_read_event(fullfile(info_nev.folder{i},info_nev.name{i}),'detectflank','both'
    hdr = ft_read_header(fullfile(info_nev.folder{i},info_nev.name{i}));
    cfg=[]; cfg.Fs = 1; %already in seconds
    cfg.starts = info_nev.starts(i);
    master_events_i = bml_event2annot(cfg,E);
    if ~isempty(master_events_i)
        master_events_i.value = master_events_i.value - 1;
        master_events_i.sample = round(master_events_i.starts .* hdr.Fs);
        master_events_i.file_id(:) = i;
        master_events = bml_annot_rowbind(master_events,master_events_i);
    end

end

% filter events
master_events.tmp = [2^15;abs(diff(master_events.value))];
master_events = master_events(master_events.tmp >= 2^9,:);
```

SYNCHRONIZING EVENT FILES TO RIPPLE EVENTS

Will inherit exact timing from ripple. looping through events_files

```
events_files.delta_t(:)=nan;
events_files.min_cost(:)=nan;
events_files.warpfactor(:)=nan;
events_files.n(:)=nan;
for i=1:height(events_files)
    %selecting nev files that correspond to this events_files.
    nev_events_files = bml_annot_filter(roi_nev,bml_annot_extend(events_files(i,:),eve

    if ~isempty(nev_events_files)
        %nev_events_files = nev_events_files(2,:);
        task_events = bml_annot_read_tsv(fullfile(events_files.path{i},events_files.fi
        %loading events from nev file
        nev = table();
        for j = 1:height(nev_events_files)
```

```
            E = ft_read_event(fullfile(nev_events_files.folder{j},nev_events_files.name
            cfg=[]; cfg.Fs = 1; %already in seconds
            cfg.starts = nev_events_files.t1(j) - (nev_events_files.s1(j)-1)./nev_event
            nev_j = bml_event2annot(cfg,E);
            nev_j.value = nev_j.value - 1;
            nev_j = bml_annot_filter(nev_j,bml_annot_extend(events_files(i,:),1));
            nev = bml_annot_rowbind(nev,nev_j);
        end

        % plot(nev.starts,nev.starts - task_events.starts)

        %Aligning events to ripple
        cfg=[];
        cfg.scan = 1;
        cfg.timewarp = true;
        [slave_delta_t, min_cost, warpfactor] = bml_timealign_annot(cfg, nev, task_even
        n = height(task_events);
        events_files.delta_t(i)=slave_delta_t;
        events_files.min_cost(i)=min_cost;
        events_files.warpfactor(i)=warpfactor;
        events_files.n(i)=n;

        if (height(nev) == height(task_events)) && all(nev.value == task_events.value)
            task_events.starts = nev.starts;
            task_events.ends = task_events.starts + task_events.duration;
            bml_annot_write_tsv(task_events,[PATH_ANNOT filesep strrep(events_files.fi
            fprintf('sync %s, delta_t=%f, avg cost=%f, warpfactor=%f \n',events_files.

        elseif (height(nev) < height(task_events)) && height(nev_events_files)==2 % ev

            %applying timealignment to events as fallback strategy
            tbar = mean(task_events.starts);
            task_events.starts_corrected = (task_events.starts - tbar) .* warpfactor +

            %finding alingment between events
            [nev_idxs, task_events_idxs] = find_mincost_indices(nev.value, task_events

            %getting time from nev according to alignemnt (when available)
            task_events.starts = bml_map(1:height(task_events),task_events_idxs,nev.st

            %using fallback times when nev times not available
            task_events.starts(isnan(task_events.starts))=task_events.starts_corrected

            bml_annot_write_tsv(task_events,[PATH_ANNOT filesep strrep(events_files.fi
            fprintf('sync %s, delta_t=%f, avg cost=%f, warpfactor=%f (maximizing match
        end
    end
end
```

SYNCHRONIZE ZOOM WITH EVENTS USING DIGITAL TRIGGERS

```
TASKS = {'sync', 'lombard', 'smsl'};

cfg =[];
cfg.master_events = master_events;
cfg.roi = roi_wav(contains(roi_wav.name,'_events.wav') & contains(roi_wav.name,TASKS)
cfg.timewarp = true;
cfg.diagnostic_plot = true;
cfg.timetol = 0.001;
cfg.coarsetol = 0.001;
cfg.restrict_master_by = 'file_id';
sync_wav = bml_sync_audio_event(cfg);

savefig(['./figures' filesep 'sub-' SUBJECT '_ses-intraop_proto-A04-fine-sync-audio_'
    char(string(datestr(now, 'YYYYmmDDMM'))) '.fig'])

bml_annot_write_tsv(sync_wav,[PATH_ANNOT filesep 'sub-' SUBJECT '_ses-' SESSION '_task

%not synching the calibration
sync_audio_cons = sync_wav(~contains(sync_wav.name,'calibration'),:);
info_wav = info_wav(~contains(info_wav.name,'calibration'),:);

% matching files by basename
tmp = regexp(sync_audio_cons.name,'^(sub-[\w]*_ses-[\w]*_task-[\w]*_run-[0-9]*)_[\w-\.
sync_audio_cons.basename  = [tmp{:}]';
tmp = regexp(info_wav.name,'^(sub-[\w]*_ses-[\w]*_task-[\w]*_run-[0-9]*)_[\w-\.]*$','t
info_wav.basename  = [tmp{:}]';
tmp=regexp(info_wav.name,'^sub-[\w]*_ses-[\w]*_task-[\w]*_run-[0-9]*_(recording-)?([a-
info_wav.chantype=cellfun(@(x) x(2),tmp);

% transfering sync to other wav files
sync_audio_all = table();
for i=1:height(sync_audio_cons)
    sel_info_wav = info_wav(strcmp(info_wav.basename,sync_audio_cons.basename(i)),:);
    sync_audio_i = repmat(sync_audio_cons(i,:),[height(sel_info_wav),1]);
    sync_audio_i.name = sel_info_wav.name;
    sync_audio_i.nSamples = sel_info_wav.nSamples;
    sync_audio_i.chantype = sel_info_wav.chantype;
    %sync_audio_i.fullfile = [];
    sync_audio_all = bml_annot_rowbind(sync_audio_all,sync_audio_i);
end

bml_annot_write_tsv(sync_audio_all,[PATH_ANNOT filesep 'sub-' SUBJECT '_ses-' SESSION
```

SYNCHRONIZE NEUROOMEGA FILE WITH EVENTS USING DIGITAL TRIGGERS

```
cfg = [];
cfg.master_events = master_events;
cfg.roi = roi_neuroomega;
cfg.timewarp = true;
cfg.diagnostic_plot = true;
cfg.timetol = 0.0003;
cfg.coarsetol = 0.001;
sync_ao = bml_sync_neuroomega_event(cfg);

bml_annot_write_tsv(sync_ao,[PATH_ANNOT filesep 'sub-' SUBJECT '_ses-' SESSION '_sync-
```

COMBINING SYNC ENTRIES INTO UNIQUE TABLE

```
sync = bml_annot_rowbind(sync_audio_all, sync_ripple, sync_ao);
%sync = bml_annot_rowbind(sync_audio_all, sync_ripple);
bml_annot_write_tsv(sync,[PATH_ANNOT filesep 'sub-' SUBJECT '_ses-' SESSION '_sync.tsv
```

VERIFYING ALL TABLES HAVE BEEN WRITTEN PROPERLY

```
sync_alphaomega = bml_annot_read_tsv([PATH_ANNOT filesep 'sub-' SUBJECT '_ses-intraop_
sync_ripple = bml_annot_read_tsv([PATH_ANNOT filesep 'sub-' SUBJECT '_ses-intraop_task
sync_wav = bml_annot_read_tsv([PATH_ANNOT filesep 'sub-' SUBJECT '_ses-intraop_task-lo
sync_wav.date_src=[];
info_wav =  bml_annot_read_tsv([PATH_ANNOT filesep 'sub-' SUBJECT '_ses-intraop_task-l
info_wav = info_wav(info_wav.filetype=="audio.wav",:);
```