به نام هستی بخش

گزارش پروژه

درس علوم اعصاب محاسباتی

نویسنده: حمیدرضا ابوئی

شماره دانشجویی: ۴۰۲۶۱۷۵۰۹

استاد: دکتر دلیری

تیر ۱۴۰۳

توضیحات دیتاست:

این مجموعه داده شامل ضبط‌های خارج سلولی و رفتار ماکاک در آزمایش‌هایی است که نقش قشر پیش حرکتی[1] (PMd) و قشر حرکتی اولیه[2] (M1) را در یک کار رسیدن[3] متوالی بررسی می‌کنند. در این آزمایش، میمون یک مکان نما روی صفحه را کنترل می کرد و برای جابجایی آن مکان نما به هدف دستیابی مشخص شده، با چندین هدف ارائه شده در هر آزمایش، پاداش دریافت کرد. حداقل نیازهای سینماتیکی برای حرکات رسیدن وجود داشت (مثلاً زمان‌های نگه‌داری بسیار کوتاه)، به این معنی که میمون معمولاً یک سری نسبتاً صاف از رسیدن را ایجاد می‌کرد. متغیرهای رفتاری ثبت شده شامل موقعیت، سرعت و شتاب هستند. دسترسی به مکان های هدف و زمان های تقریبی ارائه گنجانده شده است. ضبط‌های الکتروفیزیولوژیک با آرایه‌های چند الکترودی یوتا از دو میمون جمع‌آوری شد، در مجموع چهار جلسه. برای هر جلسه ده‌ها واحد کاملاً جدا شده از قشر حرکتی اولیه (M1) و قشر پیش حرکتی پشتی (PMd) وجود دارد.

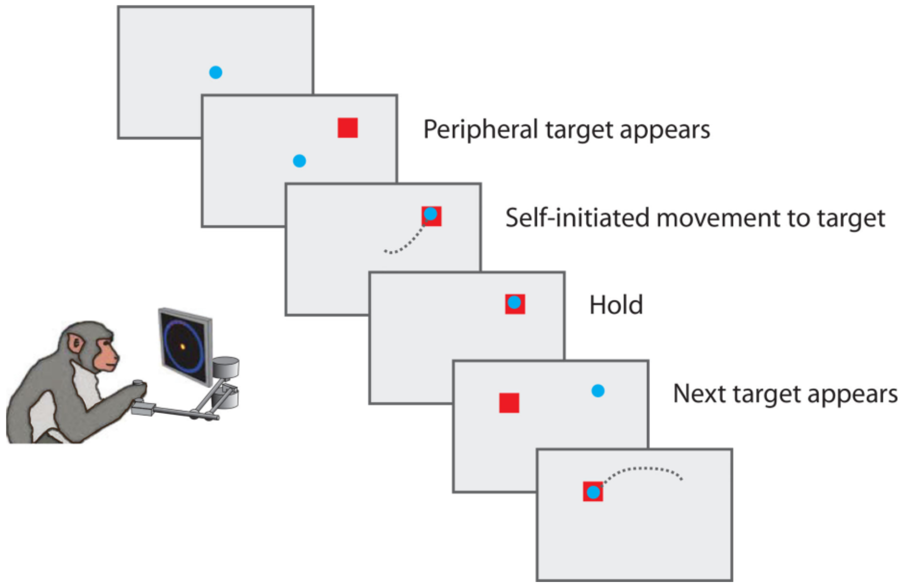
توضیح آزمایش

دو میمون (Monkey M, MM؛ Monkey T, MT) یک کار رسیدن را انجام دادند که در آن مکان نما کامپیوتر را با استفاده از حرکات بازو کنترل کردند (شکل ۱). میمون در حالی که یک دستکاری مسطح[4] دو پیوندی را کار می کرد، روی یک صندلی نخستی نشسته بود. حرکات دست به یک صفحه افقی در فضای کاری ۲۰ در ۲۰ سانتی متر محدود می شود. در این کار، یک نشانه بصری روی صفحه (مربع ۲ سانتی متر) مکان هدف را برای هر دستیابی مشخص می کرد و پس از انجام یک سری چهار تا رسیدن صحیح به اهداف، به میمون جایزه داده شد. مکان هدف به صورت شبه تصادفی انتخاب شد تا در داخل یک حلقه (شعاع = ۵-۱۵ سانتی متر، زاویه = ۳۶۰ درجه) در مرکز هدف فعلی قرار گیرد.

زمان نمایش هدف برای اولین دستیابی هر آزمایش (آغاز سری، پس از استراحت) با رسیدن به دو تا چهار (وسیله میانی) متفاوت بود. برای اولین دستیابی، هدف ارائه شد و به میمون اجازه داده شد بدون یک دستور دوره تاخیر حرکت کند. برای فاصله های دو تا چهار، میمون هدف بعدی را با نگه داشتن مکان نما برای مدت کوتاهی در یک جعبه ۲ سانتی متری در ۲ سانتی متری در مرکز هدف فعلی آغاز کرد. پس از رسیدن به هدف فعلی، هدف بعدی ۱۰۰ میلی‌ثانیه بعد فعال شد (نکته: نمایش داده نمی‌شود). زمان ظاهر شدن هدف بعدی دقیقاً ثبت نشد، اما با آزمایش‌های بعدی مشخص شد که به طور متوسط ۹۶ میلی‌ثانیه پس از شروع به کار ظاهر می‌شود. بنابراین، به طور متوسط، هدف بعدی تقریباً ۲۰۰ میلی ثانیه پس از رسیدن میمون به هدف فعلی ظاهر شد. علاوه بر این، یک دوره توقف تحمیلی ۱۰۰ میلی ثانیه وجود داشت که همزمان با شروع به هدف بعدی آغاز شد. بنابراین، بین زمانی که میمون به هدف فعلی رسید و زمانی که اجازه داشت حرکت بعدی را آغاز کند، یک فاصله زمانی ۲۰۰ میلی ثانیه وجود داشت. این دوره کوتاه نگهداشتن باعث شد که میمون‌ها با نزدیک شدن به هدف، سرعت خود را کاهش دهند. در عمل، این کار منجر به یک سری حرکات نسبتاً نرم بازو با فاصله متغیر شد.

---

[1] premotor cortex
[2] primary motor cortex
[3] reach
[4] two-link planar manipulandum



شکل ۱ شماتیک آزمایش رسیدن

کد به دو قسمت تقسیم می‌شود. بخش اول نشست‌های مختلف (۴ نشست) را آماده سازی برای خواندن می‌کند و کد دوم تمام مراحل پردازش را در بر دارد.

بخش اول به شرح زیر است:

```
clc;
clear;
close all;

%% Start reading data1
main_file_reader_flag = 1;
fname = 'MM_S1_raw.mat'; % Monkey M (MM), session 1; PMd and M1
code1
clear;
main_file_reader_flag = 1;

fname = 'MT_S1_raw.mat'; % Monkey T (MT), session 1; PMd only
code1
clear;
main_file_reader_flag = 1;

fname = 'MT_S2_raw.mat'; % Monkey T (MT), session 2; PMd only
code1
clear;
main_file_reader_flag = 1;

fname = 'MT_S3_raw.mat'; % Monkey T (MT), session 3; PMd only
code1

clear main_file_reader_flag
```

کدهای فایل اصلی به شرح زیر است:

```
is_main_reader_executed = exist("main_file_reader_flag","var");
if ~is_main_reader_executed
    clc
    clear;
    close all;
    fname = 'MM_S1_raw.mat'; % Monkey M (MM), session 1; PMd and M1

end
%% Load data

fpath = './../data/data_and_scripts/source_data/raw/';
ipath = './../images/';
% fname = 'MM_S1_raw.mat'; % Monkey M (MM), session 1; PMd and M1
% fname = 'MT_S1_raw.mat'; % Monkey T (MT), session 1; PMd only
% fname = 'MT_S2_raw.mat'; % Monkey T (MT), session 2; PMd only
% fname = 'MT_S3_raw.mat'; % Monkey T (MT), session 3; PMd only

if (strcmp(fname,'MM_S1_raw.mat'))
    img_header_name = "MM_S1_";
elseif (strcmp(fname,'MT_S1_raw.mat'))
    img_header_name = "MT_S1_";
elseif (strcmp(fname,'MT_S2_raw.mat'))
    img_header_name = "MT_S2_";
else
    img_header_name = "MT_S3_";
end


load([fpath fname])

alldays.tt = trial_table;

% Check for existence of M1 data - only one animal (MM) has it
M1_present = exist('M1','var');

%% User inputs

dt = .01; % in seconds. .01 = 10ms, .02 = 20ms, etc

time_end_extend = 0.4; % in seconds; time after end of reach to continue grabbing 
data
time_before_extend = -0.3; % in seconds; time before reach start to start grabbing 
data; needs to be negative to go backwards


%% Initialize

blocks = 1; % there is only one block in this data set
correct_trials_only = 0; % All included trials are correct by default; this is legacy
col_start_time = 1; % column # in the table with experimental data for time of trial 
start; user shouldn't change this
```

```
col_end_time = 22;  % column # in the table with experimental data for time of trial
end; user shouldn't change this

bl_trials{1} = ones(size(trial_table,1),1);

% Initialize M1_units and PMd_units
PMd_units = cell(length(PMd.units),1);
if M1_present
    M1_units = cell(length(M1.units),1);
end

%% Extract data


% For PMd units
for i = 1:length(PMd.units)
    PMd_units{i} = PMd.units(i).ts;
end

% For M1 units
if M1_present
    for i = 1:length(M1.units)
        M1_units{i} = M1.units(i).ts;
    end
end

kin.pos(:,1) = cont.t;
kin.pos(:,2:3) = cont.pos;
kin.vel(:,1) = cont.t;
kin.vel(:,2:3) = cont.vel;
kin.acc(:,1) = cont.t;
kin.acc(:,2:3) = cont.acc;


% Extract neural data
tic

num_units_PMd = length(PMd_units);
if M1_present
    num_units_M1 = length(M1_units);
end

max_spk_time = max(kin.pos(:,1)); % Set max timestamp to collect spikes

edges = 0:dt:(max_spk_time + dt); % Defines edges for histogram; used for binning
data later

% Initialize variables that will contain spikes
neural_data_temp_PMd = nan(num_units_PMd,length(edges)-1);
if M1_present
    neural_data_temp_M1 = nan(num_units_M1,length(edges)-1);
end

% Bin data using histcounts()
for nrn_num = 1:num_units_PMd
```

```
    neural_data_temp_PMd(nrn_num,:) = histcounts(PMd_units{nrn_num},edges);
    disp(['PMd unit: ' num2str(nrn_num) '/' num2str(num_units_PMd)])
end

if M1_present
    for nrn_num = 1:num_units_M1
        neural_data_temp_M1(nrn_num,:) = histcounts(M1_units{nrn_num},edges);
        disp(['M1 unit: ' num2str(nrn_num) '/' num2str(num_units_M1)])
    end
end

% IMPORTANT:
% Spikes collected from 0 ms onward; Kinematics from 1000ms onward
% This gets rid of the spikes before kinematics  are collected
% I.e., with 10ms bins, 1000ms = 100 bins * 10ms/bin

num_bins_discard = round(1/dt); % number of bins to discard before kinematics are
colletcted
bin_start = num_bins_discard + 1; % bin to start keeping track of data

neural_data_temp_PMd = neural_data_temp_PMd(:,bin_start:end);
if M1_present
    neural_data_temp_M1 = neural_data_temp_M1(:,bin_start:end);
end

disp(['Neural data extracted: ' num2str(toc) ' seconds'])

% Extract kinematic data
tic

kin_ts = downsample(kin.pos(:,1),round(1000*dt));  % This doesn't need to be filtered
(it's just timestamps)

% Decimate (smooth+downsample, rather than just downsample)
x_pos = decimate(kin.pos(:,2),round(1000*dt)); % Sampling rate of kinematics is 1
kHz. thus, decimate so bin size is same as dt
y_pos = decimate(kin.pos(:,3),round(1000*dt));
x_vel = decimate(kin.vel(:,2),round(1000*dt));
y_vel = decimate(kin.vel(:,3),round(1000*dt));
x_acc = decimate(kin.acc(:,2),round(1000*dt));
y_acc = decimate(kin.acc(:,3),round(1000*dt));

disp(['Kinematic data extracted: ' num2str(toc) ' seconds'])

% Make timestamps
ts = edges(1:(end-1));
ts = ts(bin_start:end); % Discard first second as above

%% Arrange data by *trial*

clear Data; clear Data2; clear Data_trials

for block_idx = 1:length(blocks)
    block_num = blocks(block_idx);
    trials = find(bl_trials{block_num}==1);
```

```
    % Initialize
    num_trials = sum(bl_trials{block_num});
    Data(block_num).kinematics = cell(num_trials,1);
    Data(block_num).neural_data_M1 = cell(num_trials,1);
    Data(block_num).neural_data_PMd = cell(num_trials,1);
    Data(block_num).block_info = alldays(block_num).tt(trials,:);
    Data(block_num).trials = bl_trials{block_num};

    num_units_PMd = length(PMd_units);
    if M1_present
        num_units_M1 = length(M1_units);
    end

    % Arrange data by trial
    for trial_idx = 1:length(trials)
        trial_num = trials(trial_idx);

        disp(['Writing data for trial: ' num2str(trial_num)])

        tr_start = alldays(block_num).tt(trial_num,col_start_time) +
time_before_extend; % Grab data before trial start
        tr_end = alldays(block_num).tt(trial_num,col_end_time) + time_end_extend; %
Grab data after trial end
        tr_bins = logical((ts>=tr_start).*(ts<=tr_end)); % Find bins between tr_start
and tr_end

        % Take neural data from between tr_start and tr_end
        Data(block_num).neural_data_PMd{trial_idx} = neural_data_temp_PMd(:,tr_bins);
        if M1_present
            Data(block_num).neural_data_M1{trial_idx} =
neural_data_temp_M1(:,tr_bins);
        end

        % Take kinematics from between tr_start and tr_end
        Data(block_num).kinematics{trial_idx} = [x_pos(tr_bins), y_pos(tr_bins), ...
                                    x_vel(tr_bins), y_vel(tr_bins), ...
                                    x_acc(tr_bins), y_acc(tr_bins), ...
                                    kin_ts(tr_bins)];

        % Get timestamps (imposed by me) to make sure they match those of
        % the kinematics
        Data(block_num).timestamps{trial_idx,1} = ts(tr_bins)';

    end
end

Data_trials = Data; % rename

%% Arrange data by *reach* - data has multiple reaches per trial; break trials up


% More user inputs
min_reach_len = 2; % in cm
time_before_cue = -0.3; % Amount of time before target comes on to grab data (in sec)
```

```
max_reach_time = round(1.4/dt) + ceil(abs(time_before_cue)/dt); % Max time for reach,
in bins
spd_thresh = 8; % in cm/sec; This is different from the speed threshold in
visualize_data.m; this helps to define the end of a reach
buff = 0.3; % Velocity often non-zero when cue comes on. Look forward at least this
much to find end of reach (in sec)
end_buff = 0.3; % Allows reach end to be a little later than the official end of
trial. Just to be a little more permissive (larger data window)
pd_lag = 0.096; % Photodetector wasn't used, so "cue on" is the command signal, not
the detection signal. *Average* lag in Miller lab is 96 ms. Exact lag varies from
trial to trial. See data description document for more information.

% Initialization
% Data2 = Data;
Data2 = struct;
idx = 1;

for tr = 1:num_trials
        reaches = find(~isnan(Data.block_info(tr,[3 8 13 18]))); % Find successful
reaches in this trial. Unsuccessful reach will have a nan in corresponding column 3
(reach 1), 8 (2), 13 (3), 18 (4)
        num_reach = length(reaches);
        tr_end = Data.block_info(tr,col_end_time);

        ts = Data.kinematics{tr}(:,end); % Based upon kinematic time stamps

        x_vel = Data.kinematics{tr}(:,3);
        y_vel = Data.kinematics{tr}(:,4);
        x_pos = Data.kinematics{tr}(:,1);
        y_pos = Data.kinematics{tr}(:,2);

        spd = sqrt(x_vel.^2 + y_vel.^2);

        for reach_idx = 1:num_reach
            reach = reaches(reach_idx);

            idx_cue_on = 2 + 5*(reach - 1); % Find column in trial table which
denotes time of target appearance

            % If reach wasn't completed or was invalid for some reason
            % (Columns referenced here will be NaN if reach is invalid)
            if isnan(Data.block_info(tr,idx_cue_on+1)) ||
isnan(Data.block_info(tr,idx_cue_on+2))
                continue
            end

            % Correct for command signal lag
            cue_on = Data.block_info(tr,idx_cue_on);
            cue_on = cue_on + pd_lag;

            % Correction: for reaches 2-4, target is displayed 100ms before
            % time in trial table
            if reach > 1
                cue_on = cue_on - .1; % Get the actual time from trial table;
subtract 100 ms for correction
```

```
            end

            wind_st = cue_on + time_before_cue; % When the data window starts; grabs
data before target appears

            [~,idx_cue_on2] = min(abs(ts-cue_on)); % Find time bin when cue comes on

            % Determine end time for each reach
            if reach < 4 % For reaches 1-3 on a given trial; these will be followed
by another reach
                % Find time of: min velocity before next go cue
                idx_cue_on_next = 2 + 5*(reach);
                cue_on_next = Data.block_info(tr,idx_cue_on_next);

                % For reach to end: (this is permissive)
                % 1) slow (falls below speed threshold), and
                % 2) a certain minimum time after cue onset must have elapsed (buff)
                % 3) end has to be before the next reach starts plus buff
                cond_reach_end = logical((spd < spd_thresh).*(ts > (cue_on +
buff)).*(ts < (cue_on_next + buff)));

                % If conditions not met, skip
                if sum(cond_reach_end) == 0
                    continue
                end

                % Within times meeting conditions, find one with minimum speed
                cond_reach_end_nan = 1.*cond_reach_end; % Initialize and convert to
scalar array
                cond_reach_end_nan(cond_reach_end_nan < 1) = nan; % Make the bins not
meeting conditions above nan
                [~,idx_reach_end] = min(spd.*cond_reach_end_nan);
                reach_end = ts(idx_reach_end);


            else
                % Find time of min velocity after last reach. Conditions:
                % 1) slow (falls below speed threshold), and
                % 2) a certain minimum time after cue onset must have elapsed (buff)
                % 3) end has to be before the trial ends plus buff
                cond_reach_end = logical((spd < spd_thresh).*(ts > (cue_on +
buff)).*(ts < (tr_end + end_buff)));

                % If conditions not met, skip
                if sum(cond_reach_end) == 0
                    continue
                end

                % Within times meeting conditions, find one with minimum speed
                cond_reach_end_nan = 1.*cond_reach_end; % initialize and convert to
scalar array
                cond_reach_end_nan(cond_reach_end_nan < 1) = nan;
                [~,idx_reach_end] = min(spd.*cond_reach_end_nan);
                reach_end = ts(idx_reach_end);
```

```
            end


            % Define window of time to save
            wind_reach = logical((ts>=wind_st).*(ts<=reach_end));


            % Add meta-data
            Data2.trial_num{idx,1} = tr;
            Data2.reach_num{idx,1} = reach;
            Data2.reach_st{idx,1} = cue_on; % Time of cue on used as a proxy for when
reach starts. It's approximate.
            Data2.cue_on{idx,1} = cue_on;
            Data2.reach_end{idx,1} = reach_end;
            Data2.reach_pos_st{idx,1} = Data.kinematics{tr}(idx_cue_on2,1:2);
            Data2.reach_pos_end{idx,1} = Data.kinematics{tr}(idx_reach_end,1:2);
            Data2.avgspeed{idx,1}=mean(spd);
            delta_pos = Data2.reach_pos_end{idx,1} - Data2.reach_pos_st{idx,1};
            [Data2.reach_dir{idx,1}, Data2.reach_len{idx,1}] =
cart2pol(delta_pos(1),delta_pos(2));

            idx_target_on = 1 + ceil(abs(time_before_cue)/dt); % Target is on in bin
1 unless extra time before is added, in which case it's on after that extra time
            temp = zeros(sum(wind_reach),1); temp(idx_target_on) = 1;
            Data2.target_on{idx,1} = temp;

            % Copy stuff
            Data2.kinematics{idx,1} = Data.kinematics{tr}(wind_reach,:);
            Data2.neural_data_PMd{idx,1} = Data.neural_data_PMd{tr}(:,wind_reach);
            if M1_present
                Data2.neural_data_M1{idx,1} = Data.neural_data_M1{tr}(:,wind_reach);
            end
            Data2.block_info = Data.block_info;
            Data2.time_window{idx,1} = wind_reach;
            Data2.timestamps{idx,1} = Data.timestamps{tr}(wind_reach);

            % Exclude reach if it doesn't meet requirements
            if (Data2.reach_len{idx} < min_reach_len) || (sum(wind_reach) >
max_reach_time)
                continue
            end

            % Increment reach index
            idx = idx + 1;

        end
end

% Rename variables
clear Data
Data = Data2; clear Data2

disp('Data cleaning completed.')
disp('Reach data located in variable: Data.')
disp('Trial data located in variable: Data_trials.')
```

```
%% Show all routes
figure();
all_kinematics=[];
x_reach_pos_end = nan(length(Data.kinematics));
y_reach_pos_end = nan(length(Data.kinematics));
for tr = 1:length(Data.kinematics)
    all_kinematics = [all_kinematics; Data.kinematics{tr,:}];
    a = Data.reach_pos_end{tr,:};
    x_reach_pos_end(tr) = a(1);
    y_reach_pos_end(tr) = a(2);
end

x = all_kinematics(:,1);
y = all_kinematics(:,2);
plot(x,y);
title("All routes monkey went and reach destinations");
xlabel("x");
ylabel("y");
hold on;

scatter(x_reach_pos_end,y_reach_pos_end);
legend("animal pos","reach pos")
saveas(gcf,strcat(ipath,img_header_name,"all_routes_pos.png"))

%% Show Some data
figure();
subplot(2,2,1);
% h=heatmap(Data.neural_data_M1{1, 1}, ...
% 'GridVisible','off');
% h.XData=Data.timestamps{1,1};
% h.XLabel = "time";
% h.YLabel = "channel";

if M1_present
    imshow(Data.neural_data_M1{1, 1})
    xlabel("time");
    ylabel("channel");
    title("neural Data M1");
else
    subplot(2,2,1);
    scatter(Data.kinematics{1, 1}(:,5),Data.kinematics{1, 1}(:,6))
    title("kinematic Acceleration")
    xlabel("x");
    ylabel("y");
end

subplot(2,2,2);
scatter(Data.kinematics{1, 1}(:,1),Data.kinematics{1, 1}(:,2))
title("kinematic position")
xlabel("x");
ylabel("y");
hold on;
scatter(Data.reach_pos_st{1,1}(1),Data.reach_pos_st{1,1}(2),'black');
scatter(Data.reach_pos_end{1,1}(1),Data.reach_pos_end{1,1}(2),'red');
```

```
subplot(2,2,3);
imshow(Data.neural_data_PMd{1, 1})
xlabel("time");
ylabel("channel");
title("neural Data PMd");

subplot(2,2,4);
plot(Data.kinematics{1, 1}(:,3),Data.kinematics{1, 1}(:,4));
title("Kinematic Velocity")
xlabel("x");
ylabel("y");
saveas(gcf,strcat(ipath,img_header_name,"overview_1.png"))

trial_num_display = 161;
figure();

if M1_present
    subplot(2,3,1);
    imshow(Data.neural_data_M1{trial_num_display, 1})
    xlabel("time");
    ylabel("channel");
    title("neural Data M1");

    subplot(2,3,2);
    imshow(Data.neural_data_PMd{trial_num_display, 1})
    xlabel("time");
    ylabel("channel");
    title("neural Data PMd");

    subplot(2,3,3);
    scatter(Data.kinematics{trial_num_display,
1}(:,1),Data.kinematics{trial_num_display, 1}(:,2))
    title(strcat("kinematic
position|",sprintf("dir=%.2f,len=%.2f",Data.reach_dir{trial_num_display,1},Data.reach
_len{trial_num_display,1})))
    xlabel("x");
    ylabel("y");
    hold on;

scatter(Data.reach_pos_st{trial_num_display,1}(1),Data.reach_pos_st{trial_num_display
,1}(2),'black');

scatter(Data.reach_pos_end{trial_num_display,1}(1),Data.reach_pos_end{trial_num_displ
ay,1}(2),'red');

    subplot(2,3,4);
    plot(movmean(sum(Data.neural_data_M1{trial_num_display, 1}),15));
    xlabel("time");
    % ylabel("channel");
    title("neural Data M1 firing rate");

    subplot(2,3,5);
    plot(movmean(sum(Data.neural_data_PMd{trial_num_display, 1}),15))
    xlabel("time");
```

```
    % ylabel("channel");
    title("neural Data PMd firing rate");

    subplot(2,3,6);
    plot(Data.kinematics{trial_num_display,
1}(:,3),Data.kinematics{trial_num_display, 1}(:,4));
    title("Kinematic Velocity")
    xlabel("x");
    ylabel("y");
else
    subplot(2,2,1);
    imshow(Data.neural_data_PMd{trial_num_display, 1})
    xlabel("time");
    ylabel("channel");
    title("neural Data PMd");

    subplot(2,2,2);
    scatter(Data.kinematics{trial_num_display,
1}(:,1),Data.kinematics{trial_num_display, 1}(:,2))
    title(strcat("kinematic
position|",sprintf("dir=%.2f,len=%.2f",Data.reach_dir{trial_num_display,1},Data.reach
_len{trial_num_display,1})))
    xlabel("x");
    ylabel("y");
    hold on;

scatter(Data.reach_pos_st{trial_num_display,1}(1),Data.reach_pos_st{trial_num_display
,1}(2),'black');

scatter(Data.reach_pos_end{trial_num_display,1}(1),Data.reach_pos_end{trial_num_displ
ay,1}(2),'red');

    subplot(2,2,3);
    plot(movmean(sum(Data.neural_data_PMd{trial_num_display, 1}),15))
    xlabel("time");
    % ylabel("channel");
    title("neural Data PMd firing rate");

    subplot(2,2,4);
    plot(Data.kinematics{trial_num_display,
1}(:,3),Data.kinematics{trial_num_display, 1}(:,4));
    title("Kinematic Velocity")
    xlabel("x");
    ylabel("y");

end
saveas(gcf,strcat(ipath,img_header_name,"overview_",int2str(trial_num_display),".png"
))

% %% Try fitting firing rate with
% trial_num_Processing = 182;
% fr_m1 = movmean(sum(Data.neural_data_PMd{trial_num_Processing, 1}),5);
% fr_PMd = movmean(sum(Data.neural_data_PMd{trial_num_Processing, 1}),5);
% xpos = Data.kinematics{trial_num_Processing, 1}(:,1);
% ypos = Data.kinematics{trial_num_Processing, 1}(:,2);
```

```
%
% linearCoef = polyfit(fr_m1,xpos,3);
% linearFit = polyval(linearCoef,fr_m1);
% figure()
% plot(fr_m1,xpos,'s', fr_m1,linearFit,'r-')
% xlabel('neural_m1'); ylabel('x_pos');


%% Show Direction and length
figure();
polarscatter([Data.reach_dir{:}],[Data.reach_len{:}]);
title("Direction and length")
saveas(gcf,strcat(ipath,img_header_name,"polar_dir_len.png"))

%% calculate all firings in each trials and show tuning curve

% Data2 = struct;
%
% idx = 1;
% for tr = 1:num_trials
%     reaches = find(~isnan(Data.block_info(tr,[3 8 13 18]))); % Find successful
reaches in this trial. Unsuccessful reach will have a nan in corresponding column 3
(reach 1), 8 (2), 13 (3), 18 (4)
%     num_reach = length(reaches);
%     tr_end = Data.block_info(tr,col_end_time);
%
%     ts = Data.kinematics{tr}(:,end); % Based upon kinematic time stamps
%
%     x_vel = Data.kinematics{tr}(:,3);
%     y_vel = Data.kinematics{tr}(:,4);
%     x_pos = Data.kinematics{tr}(:,1);
%     y_pos = Data.kinematics{tr}(:,2);
%     x_acc = Data.kinematics{tr}(:,5);
%     y_acc = Data.kinematics{tr}(:,6);
%
%     spd = sqrt(x_vel.^2 + y_vel.^2);
%
%     for reach_idx = 1:num_reach
%         disp(idx);
%          Data2.average_firing_rate_PMd{idx,1} =
mean(mean((Data.neural_data_PMd{idx,1})));
%          Data2.average_firing_rate_M1{idx,1} =
mean(mean((Data.neural_data_M1{idx,1})));
%
%
%         % Exclude reach if it doesn't meet requirements
%         if (Data.reach_len{idx} < min_reach_len) || (sum(wind_reach) >
max_reach_time)
%             continue
%         end
%         idx = idx + 1;
%     end
%
%
% end
```

```
for trial = 1:length(Data.time_window)

     Data.average_firing_rate_PMd{trial,1} =
(mean((Data.neural_data_PMd{trial,1}),2));
     if M1_present
        Data.average_firing_rate_M1{trial,1} =
(mean((Data.neural_data_M1{trial,1}),2));
     end

end

%% Plot Tuning curves
% select_neuron = 13;
select_neuron = nan;

if (strcmp(fname,'MM_S1_raw.mat'))
    selected_data_to_plot=105;
    selected_data_to_plot2=276;
    [~,minI] = min(mean([Data.average_firing_rate_PMd{:}]));
    [~,maxI] = max(mean([Data.average_firing_rate_PMd{:}]));
    selected_data_to_plot3 = minI;
    selected_data_to_plot4 = maxI;
    [~,minI] = min(mean([Data.average_firing_rate_M1{:}]));
    [~,maxI] = max(mean([Data.average_firing_rate_M1{:}]));
    selected_data_to_plot5 = minI;
    selected_data_to_plot6 = maxI;
elseif (strcmp(fname,'MT_S1_raw.mat'))
    selected_data_to_plot=171;
    selected_data_to_plot2=201;
elseif (strcmp(fname,'MT_S2_raw.mat'))
    selected_data_to_plot=309;
    selected_data_to_plot2=410;
else
    selected_data_to_plot = 91;
    selected_data_to_plot2 = 60;
end

if isnan(select_neuron)
    figure();
    x = ([Data.reach_dir{:}]);
    y = mean([Data.average_firing_rate_PMd{:}]);
    f = fit(x.',y.','gauss1');
    plot(f,x,y)
    hold on;
    plot(x(selected_data_to_plot),y(selected_data_to_plot),'o');
    plot(x(selected_data_to_plot2),y(selected_data_to_plot2),'o');
    if exist("selected_data_to_plot3","var")
        plot(x(selected_data_to_plot3),y(selected_data_to_plot3),'o');
        plot(x(selected_data_to_plot4),y(selected_data_to_plot4),'o');
    end
    xlabel("reaching Direction");
    ylabel("average firing rate")
    title("tuning curve, firing rate PMd vs reach direction")
    saveas(gcf,strcat(ipath,img_header_name,"fit_gauss_PMd_tunning_curve.png"))
```

```
    if M1_present
        figure();
        x = [Data.reach_dir{:}];
        y = mean([Data.average_firing_rate_M1{:}]);
        f = fit(x.',y.','gauss1');
        plot(f,x,y)
        hold on;
        plot(x(selected_data_to_plot),y(selected_data_to_plot),'o');
        plot(x(selected_data_to_plot2),y(selected_data_to_plot2),'o');
        if exist("selected_data_to_plot3","var")
            plot(x(selected_data_to_plot5),y(selected_data_to_plot5),'o');
            plot(x(selected_data_to_plot6),y(selected_data_to_plot6),'o');
        end
        xlabel("reaching Direction");
        ylabel("average firing rate");
        title("tuning curve, firing rate m1 vs reach direction");
        saveas(gcf,strcat(ipath,img_header_name,"fit_gauss_M1_tunning_curve.png"))

    end

    figure();
    if M1_present
        subplot(2,2,1);
        imshow(Data.neural_data_M1{selected_data_to_plot, 1});
        xlabel("time");
        ylabel("channel");
        title(strcat("neural Data M1",sprintf("
|Dir%.2f",Data.reach_dir{selected_data_to_plot,1})));

        subplot(2,2,2);
        imshow(Data.neural_data_PMd{selected_data_to_plot, 1})
        xlabel("time");
        ylabel("channel");
        title("neural Data PMd");

        subplot(2,2,3);
        imshow(Data.neural_data_M1{selected_data_to_plot2, 1})
        xlabel("time");
        ylabel("channel");
        title(strcat("neural Data M1",sprintf("
|Dir%.2f",Data.reach_dir{selected_data_to_plot2,1})));

        subplot(2,2,4);
        imshow(Data.neural_data_PMd{selected_data_to_plot2, 1})
        xlabel("time");
        ylabel("channel");
        title("neural Data PMd");

        saveas(gcf,strcat(ipath,img_header_name,"firing_rate.png"))

        figure();
        subplot(2,2,1);
        imshow(Data.neural_data_M1{selected_data_to_plot5, 1});
        xlabel("time");
```

```
        ylabel("channel");
        title(strcat("Minfr neural Data M1",sprintf("
|Dir%.2f",Data.reach_dir{selected_data_to_plot5,1})));

        subplot(2,2,2);
        imshow(Data.neural_data_PMd{selected_data_to_plot3, 1})
        xlabel("time");
        ylabel("channel");
        title(strcat("minfr neural Data PMd",sprintf("
|Dir%.2f",Data.reach_dir{selected_data_to_plot5,1})));

        subplot(2,2,3);
        imshow(Data.neural_data_M1{selected_data_to_plot6, 1})
        xlabel("time");
        ylabel("channel");
        title(strcat("maxfr neural Data M1",sprintf("
|Dir%.2f",Data.reach_dir{selected_data_to_plot6,1})));

        subplot(2,2,4);
        imshow(Data.neural_data_PMd{selected_data_to_plot4, 1})
        xlabel("time");
        ylabel("channel");
        title(strcat("maxfr neural Data PMd",sprintf("
|Dir%.2f",Data.reach_dir{selected_data_to_plot5,1})));
        saveas(gcf,strcat(ipath,img_header_name,"firing_rate_min_max.png"))

    else
        subplot(2,1,1);
        imshow(Data.neural_data_PMd{selected_data_to_plot, 1})
        xlabel("time");
        ylabel("channel");
        title(strcat("neural Data PMd",sprintf("
|Dir%.2f",Data.reach_dir{selected_data_to_plot,1})));

        subplot(2,1,2);
        imshow(Data.neural_data_PMd{selected_data_to_plot2, 1})
        xlabel("time");
        ylabel("channel");
        title(strcat("neural Data PMd",sprintf("
|Dir%.2f",Data.reach_dir{selected_data_to_plot2,1})));
        saveas(gcf,strcat(ipath,img_header_name,"firing_rate.png"))

    end

%     figure();
%     x = [Data.avgspeed{:}];
%     y = mean([Data.average_firing_rate_PMd{:}]);
%     f = fit(x.',y.','gauss1');
%     plot(f,x,y)
%     xlabel("average reaching speed");
%     ylabel("average firing rate")
%     title("tuning curve, firing rate PMd vs average reach speed")

%     if M1_present
%         figure();
```

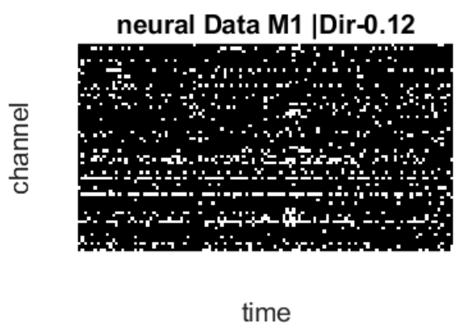
```
%         x = [Data.reach_len{:}];
%         y = mean([Data.average_firing_rate_M1{:}]);
%         f = fit(x.',y.','poly3');
%         plot(f,x,y)
%         xlabel("reaching length");
%         ylabel("average firing rate")
%         title("tuning curve, firing rate M1 vs reach length")
%     end
%
%     figure();
%     x = ([Data.reach_dir{:}]);
%     y = mean([Data.average_firing_rate_PMd{:}]);
%     f = fit(x.',y.','gauss1');
%     plot(f,x,y)
%     xlabel("reaching Direction");
%     ylabel("average firing rate")
%     title("tuning curve, firing rate PMd vs reach direction")
%
%     if M1_present
%         figure();
%         x = [Data.reach_dir{:}];
%         y = mean([Data.average_firing_rate_M1{:}]);
%         f = fit(x.',y.','gauss1');
%         plot(f,x,y)
%         xlabel("reaching Direction");
%         ylabel("average firing rate")
%         title("tuning curve, firing rate m1 vs reach direction")
%     end


else

    figure();
    x = [Data.reach_dir{:}];
    y = [Data.average_firing_rate_PMd{:}];
    f = fit(x.',y(select_neuron,:).','gauss1');
    plot(f,x,y)
    xlabel("reaching Direction");
    ylabel("average firing rate")
    title("tuning curve, firing rate PMd vs reach direction")

saveas(gcf,strcat(ipath,img_header_name,"fit_gauss_PMd_tunning_curve_single_neuron.pn
g"))

    if M1_present
        figure();
        x = [Data.reach_dir{:}];
        y = [Data.average_firing_rate_M1{:}];
        f = fit(x.',y(select_neuron,:).','gauss1');
        plot(f,x,y)
        xlabel("reaching Direction");
        ylabel("average firing rate")
        title("tuning curve, firing rate m1 vs reach direction")
```

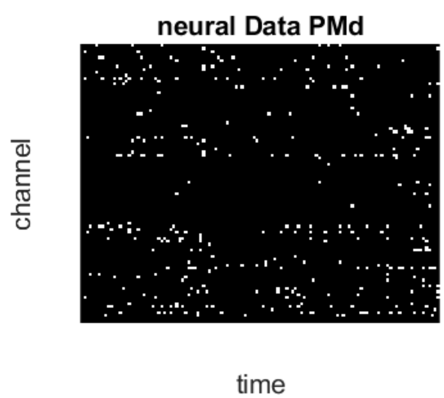
```
saveas(gcf,strcat(ipath,img_header_name,"fit_gauss_M1_tunning_curve_single_neuron.png
"))
    end

    figure();
    x = [Data.reach_len{:}];
    y = [Data.average_firing_rate_PMd{:}];
    f = fit(x.',y(select_neuron,:).','poly3');
    plot(f,x,y)
    xlabel("reaching length");
    ylabel("average firing rate")
    title("tuning curve, firing rate PMd vs reach length")

saveas(gcf,strcat(ipath,img_header_name,"fit_gauss_PMd_poly_len_tunning_curve_single_
neuron.png"))

    if M1_present
        figure();
        x = [Data.reach_len{:}];
        y = [Data.average_firing_rate_M1{:}];
        f = fit(x.',y(select_neuron,:).','poly3');
        plot(f,x,y)
        xlabel("reaching length");
        ylabel("average firing rate")
        title("tuning curve, firing rate M1 vs reach length")

saveas(gcf,strcat(ipath,img_header_name,"fit_gauss_M1_poly_len_tunning_curve_single_n
euron.png"))

    end

end
```

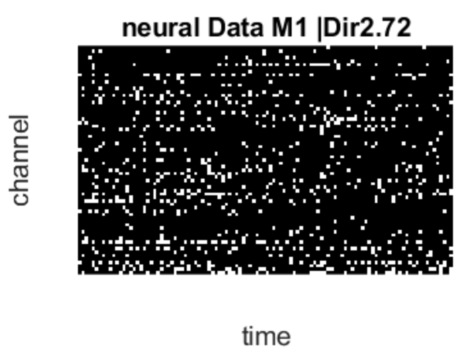
همچنین تصاویر به دست آمده نیز به شرح زیر می‌باشد

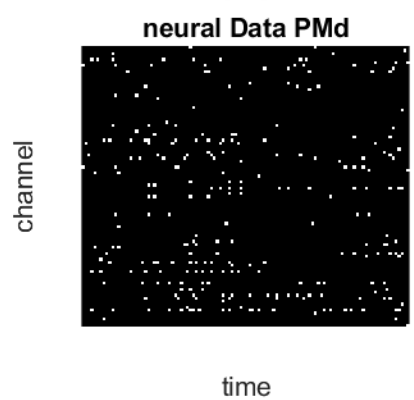
تصاویر MM-S1:

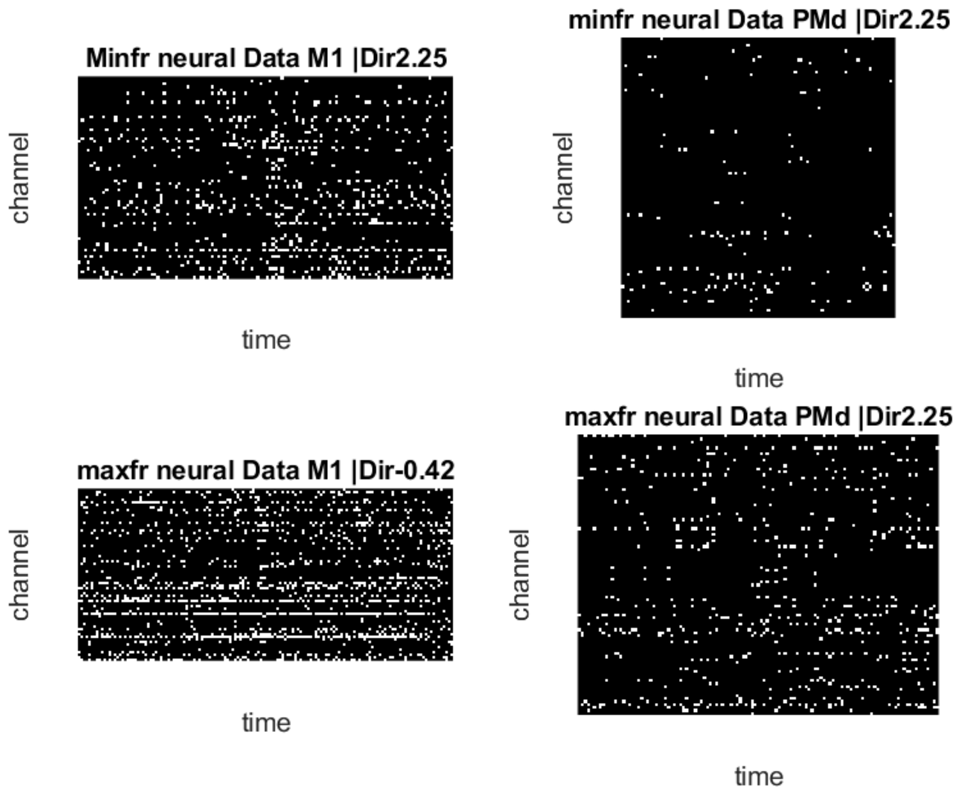

All routes monkey went and reach destinations
animal pos
reach pos
y
x
12
10
8
6
4
2
0
-2
-4
-6
-8
-15
-10
-5
0
5
10

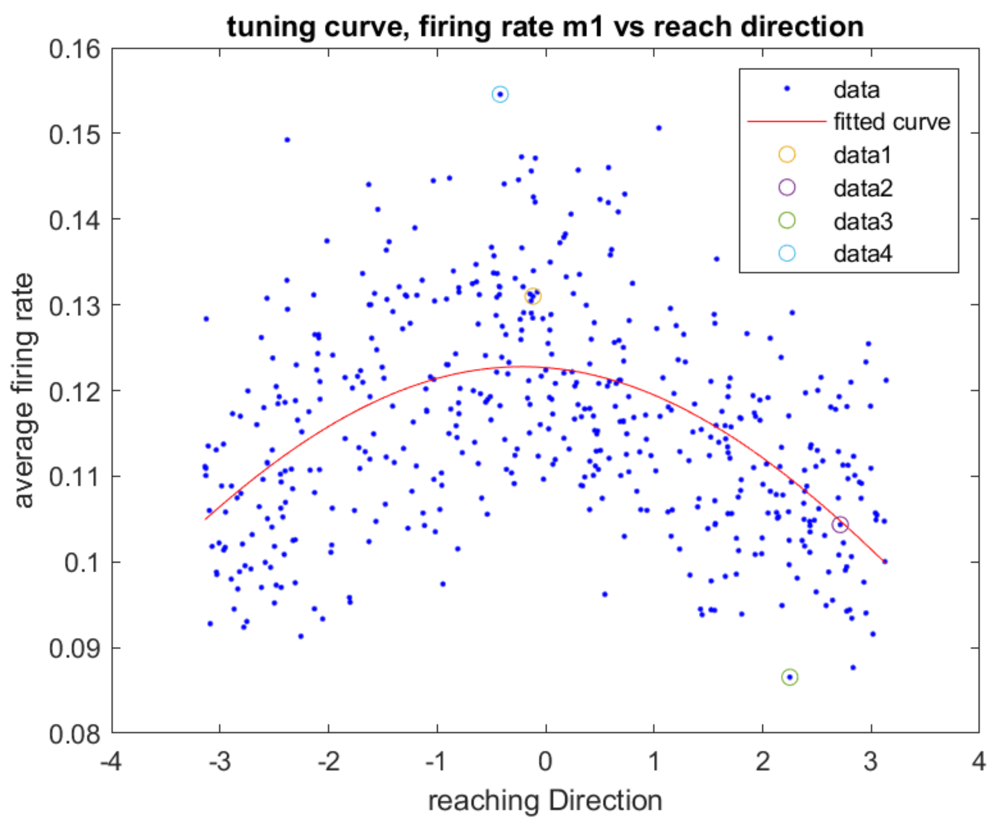

neural Data M1 |Dir-0.12
channel
time

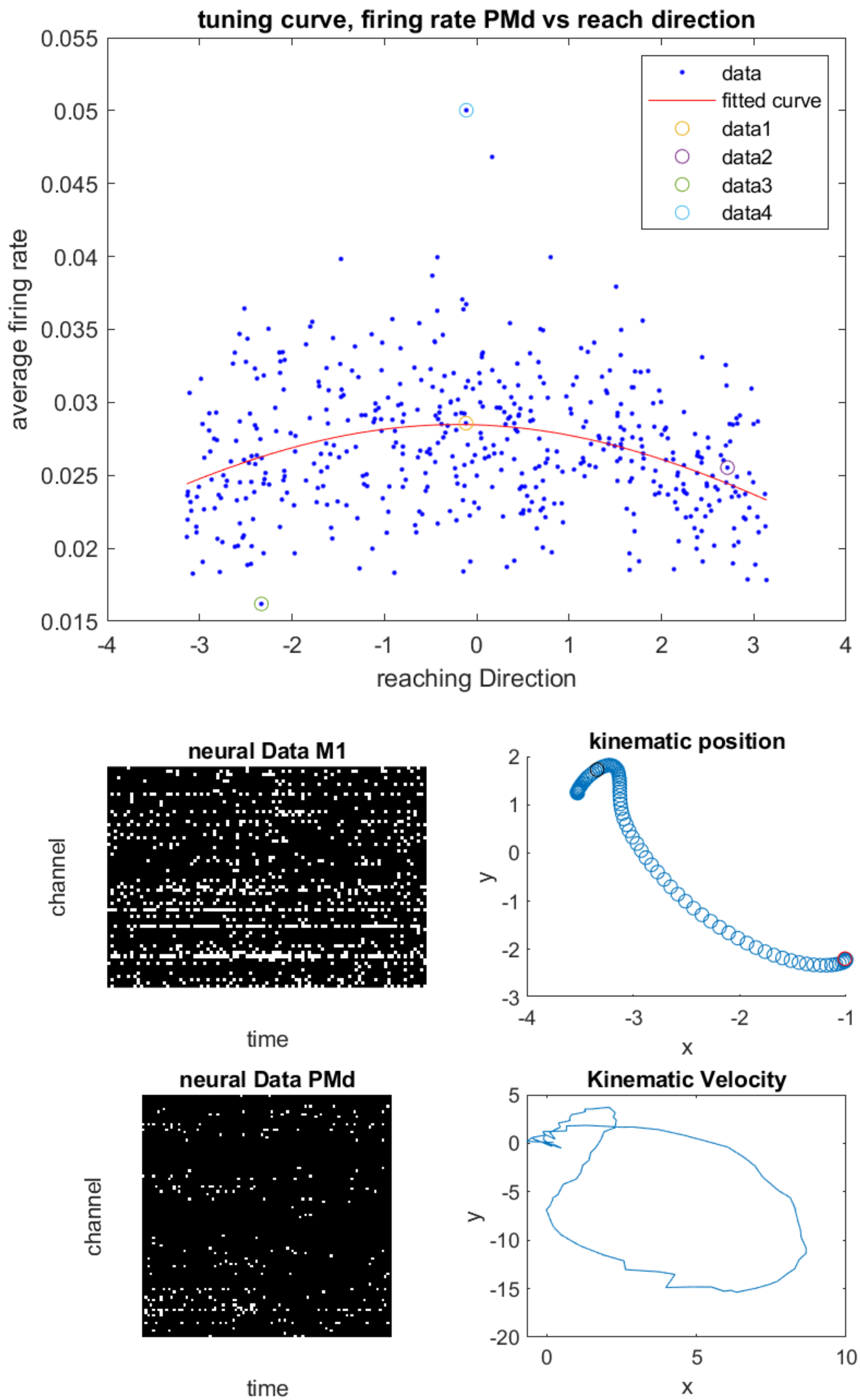

neural Data PMd
channel
time

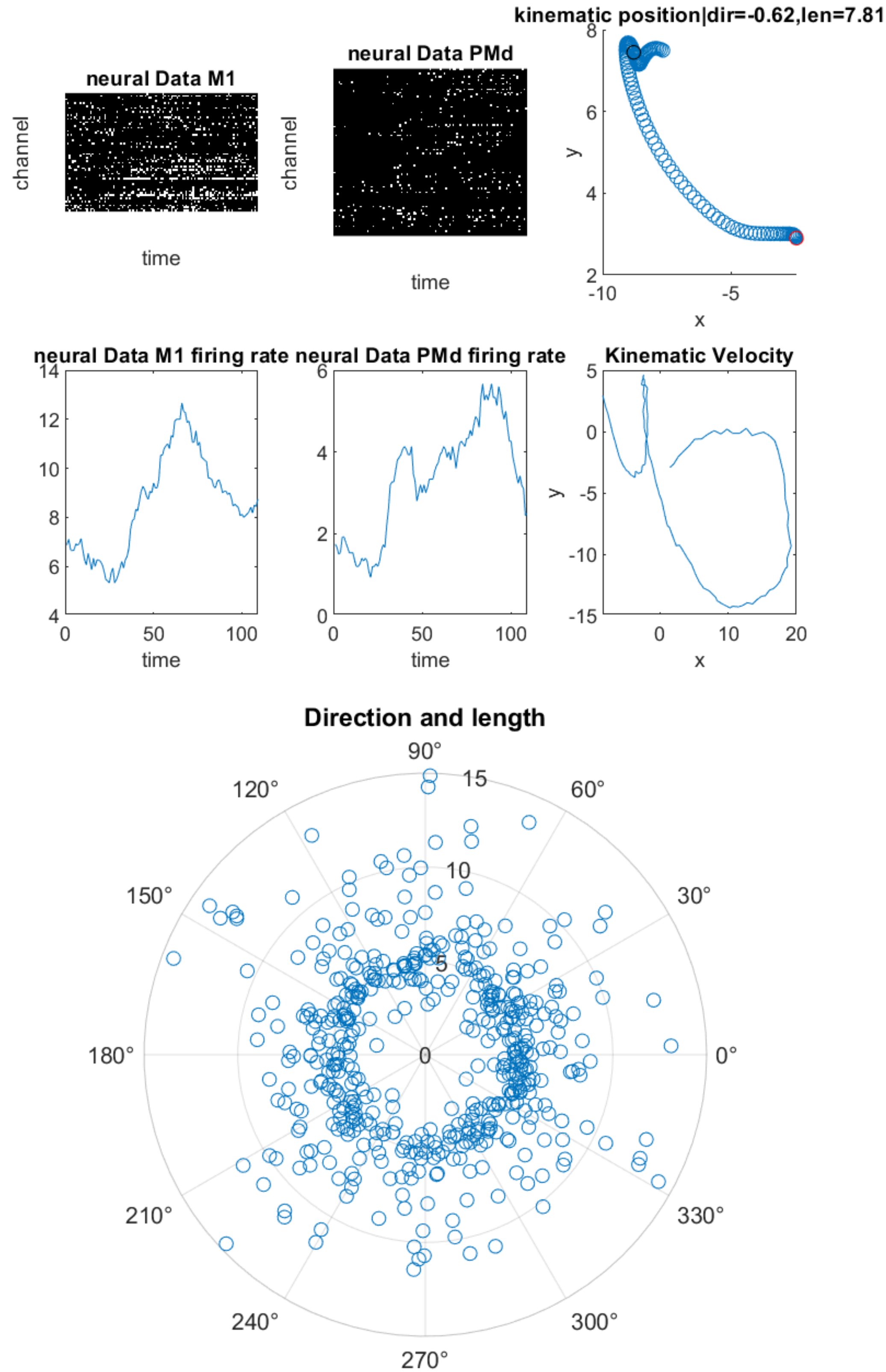

neural Data M1 |Dir2.72
channel
time


neural Data PMd
channel
time


Minfr neural Data M1 |Dir2.25
channel
time
minfr neural Data PMd |Dir2.25
channel
time
maxfr neural Data M1 |Dir-0.42
channel
time
maxfr neural Data PMd |Dir2.25
channel
time

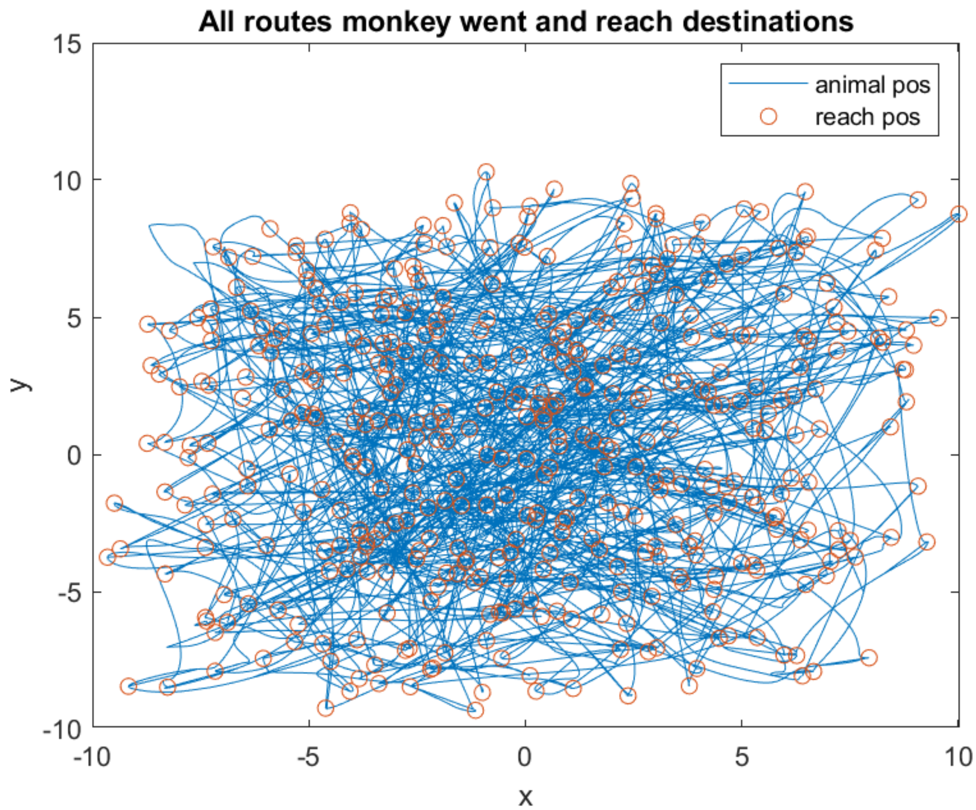

tuning curve, firing rate m1 vs reach direction
average firing rate
reaching Direction
0.16
0.15
0.14
0.13
0.12
0.11
0.1
0.09
0.08
-4
-3
-2
-1
0
1
2
3
4
data
fitted curve
data1
data2
data3
data4


tuning curve, firing rate PMd vs reach direction
average firing rate
reaching Direction
data
fitted curve
data1
data2
data3
data4
neural Data M1
channel
time
kinematic position
y
x
neural Data PMd
channel
time
Kinematic Velocity
y
x

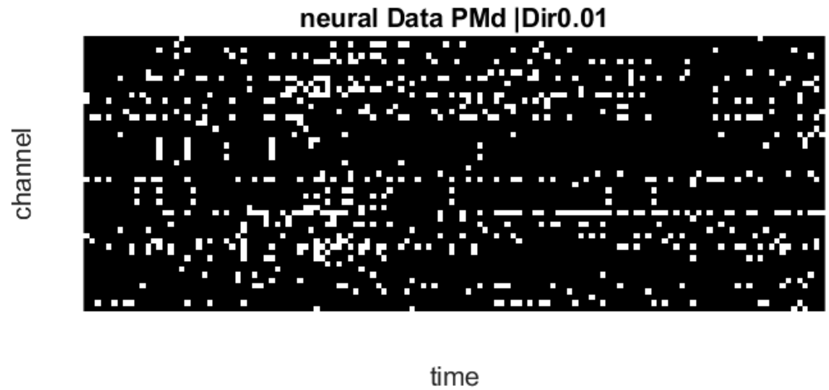


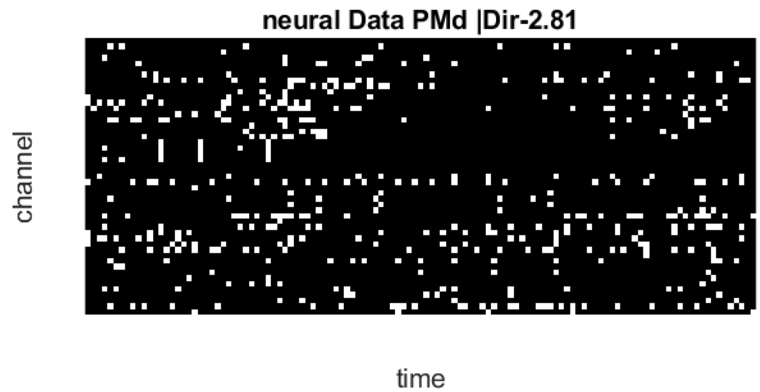
neural Data M1
channel
time
neural Data PMd
channel
time
kinematic position|dir=-0.62,len=7.81
y
x
neural Data M1 firing rate
time
neural Data PMd firing rate
time
Kinematic Velocity
y
x
Direction and length
90°
120°
60°
150°
30°
180°
0°
210°
330°
240°
300°
270°

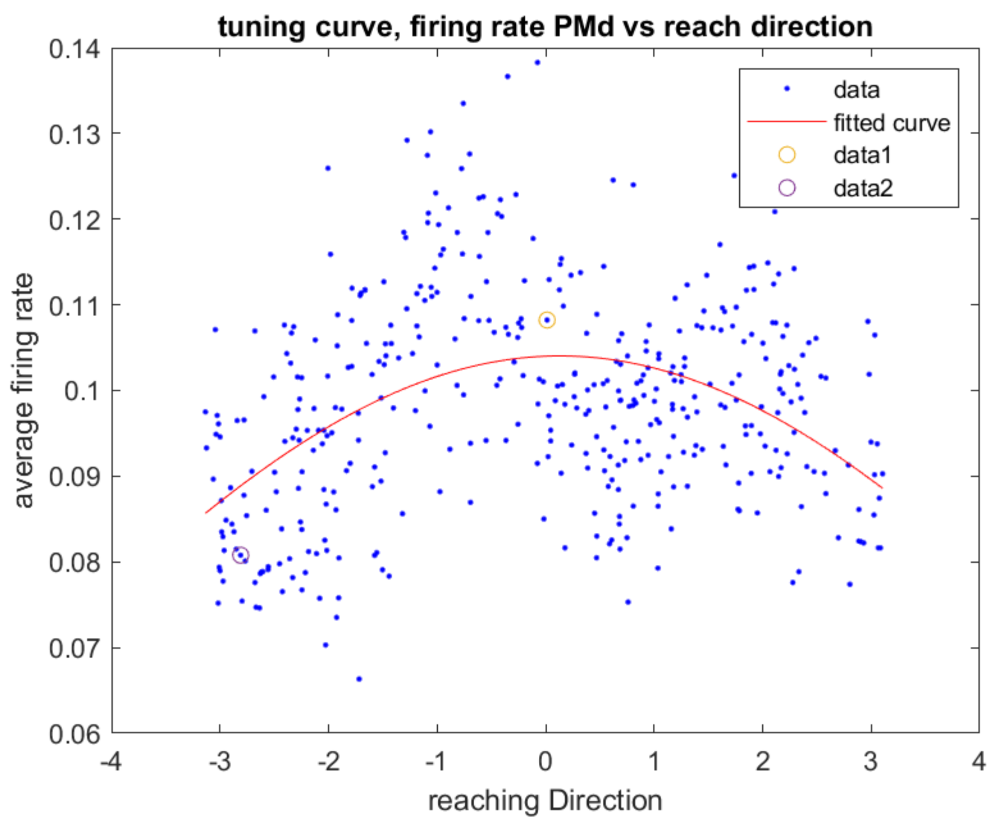
تصاویر MT-S1:

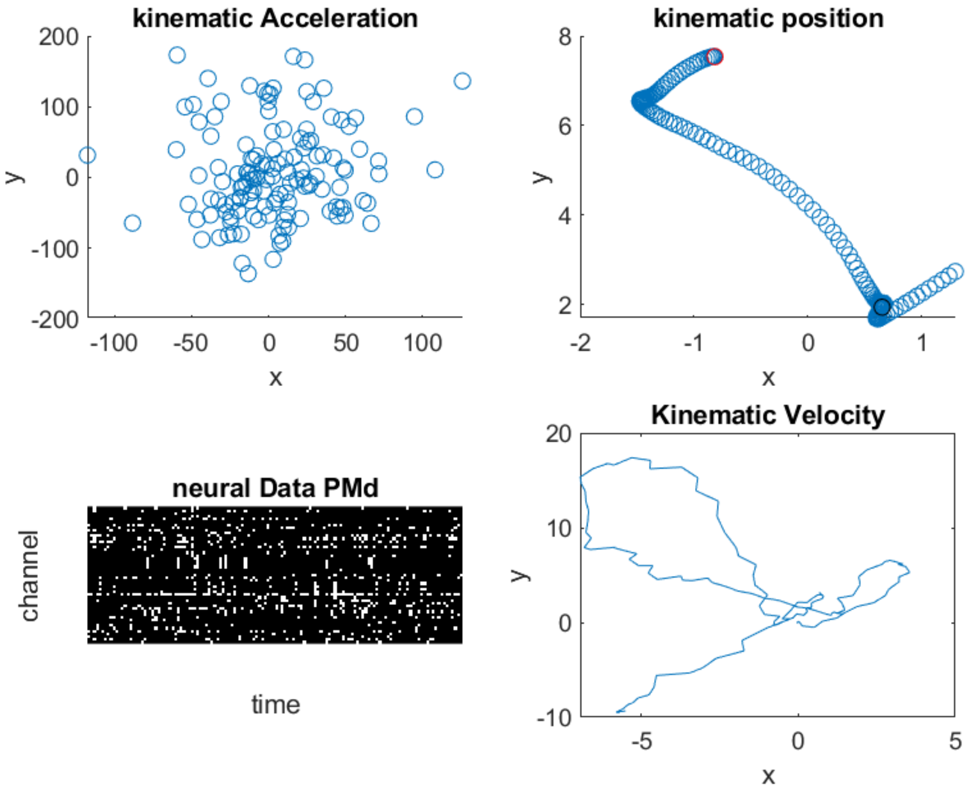


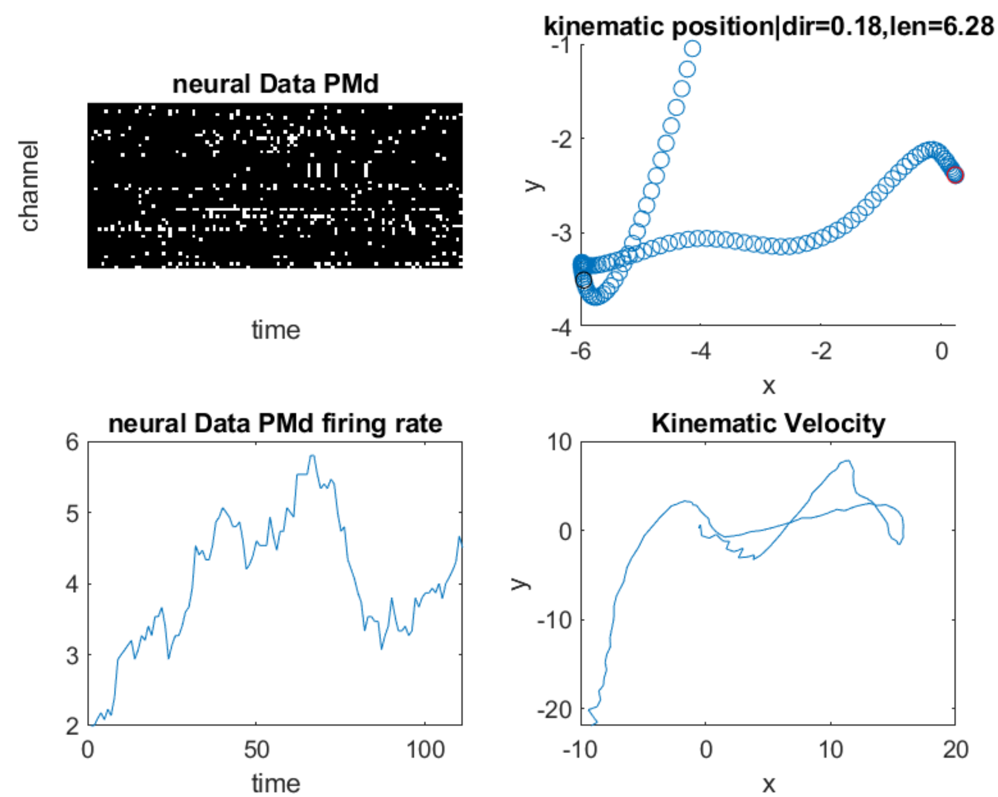
All routes monkey went and reach destinations
animal pos
reach pos
15
10
5
0
-5
-10
-10
-5
0
5
10
y
x


neural Data PMd |Dir0.01
channel
time

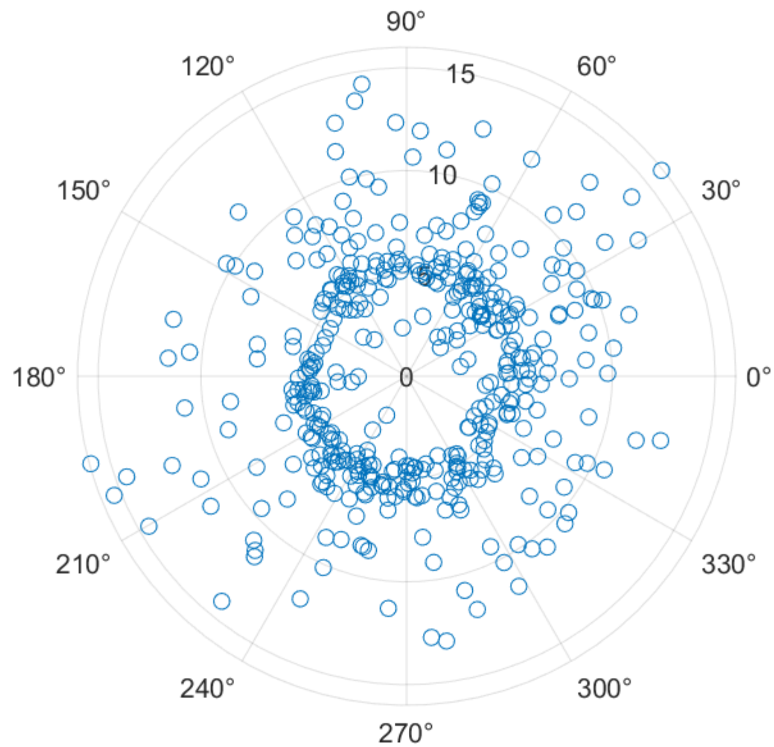

neural Data PMd |Dir-2.81
channel
time


tuning curve, firing rate PMd vs reach direction
data
fitted curve
data1
data2
average firing rate
reaching Direction
0.14
0.13
0.12
0.11
0.1
0.09
0.08
0.07
0.06
-4
-3
-2
-1
0
1
2
3
4

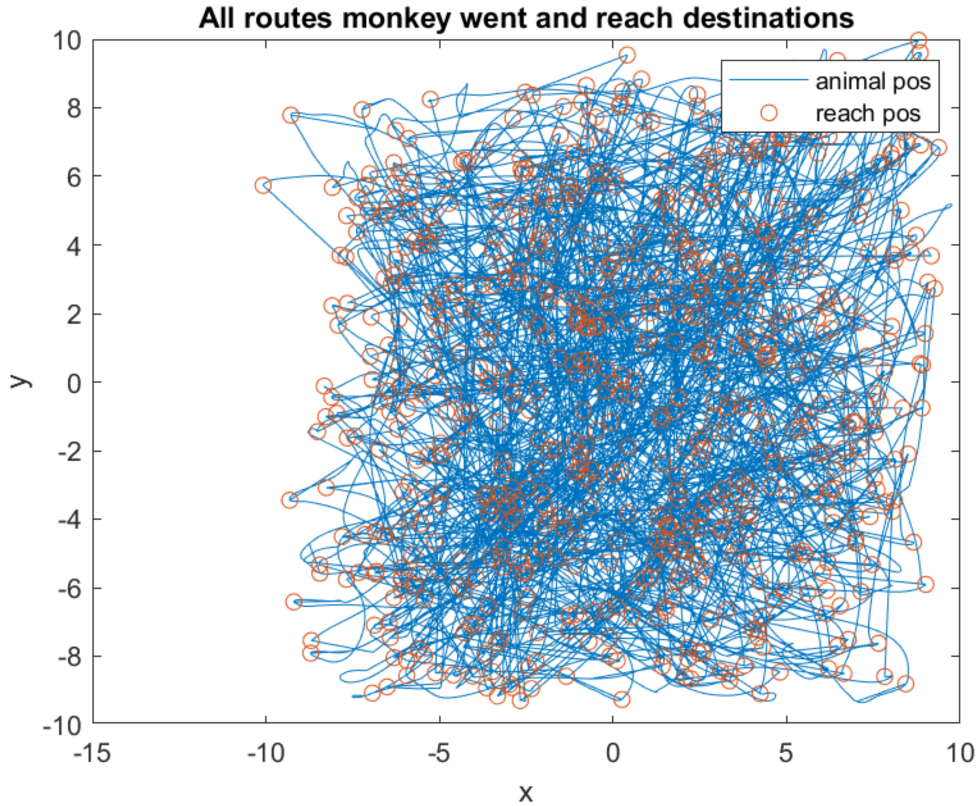

kinematic Acceleration
kinematic position
neural Data PMd
Kinematic Velocity
channel
time
x
y


kinematic position|dir=0.18,len=6.28
neural Data PMd
neural Data PMd firing rate
Kinematic Velocity
channel
time
x
y

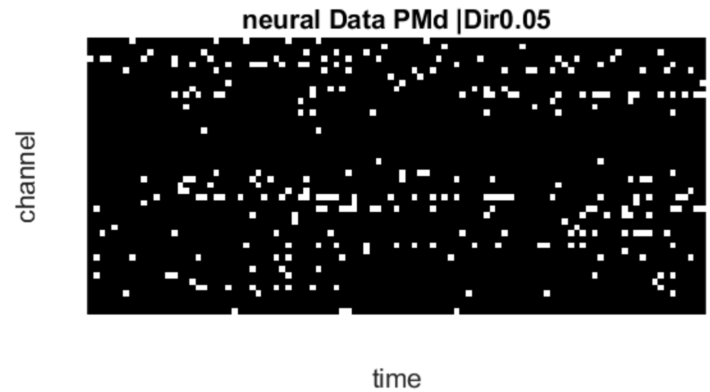
Direction and length

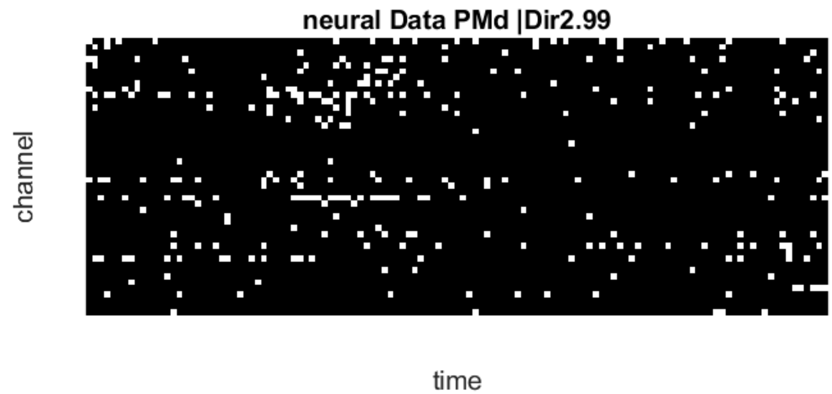


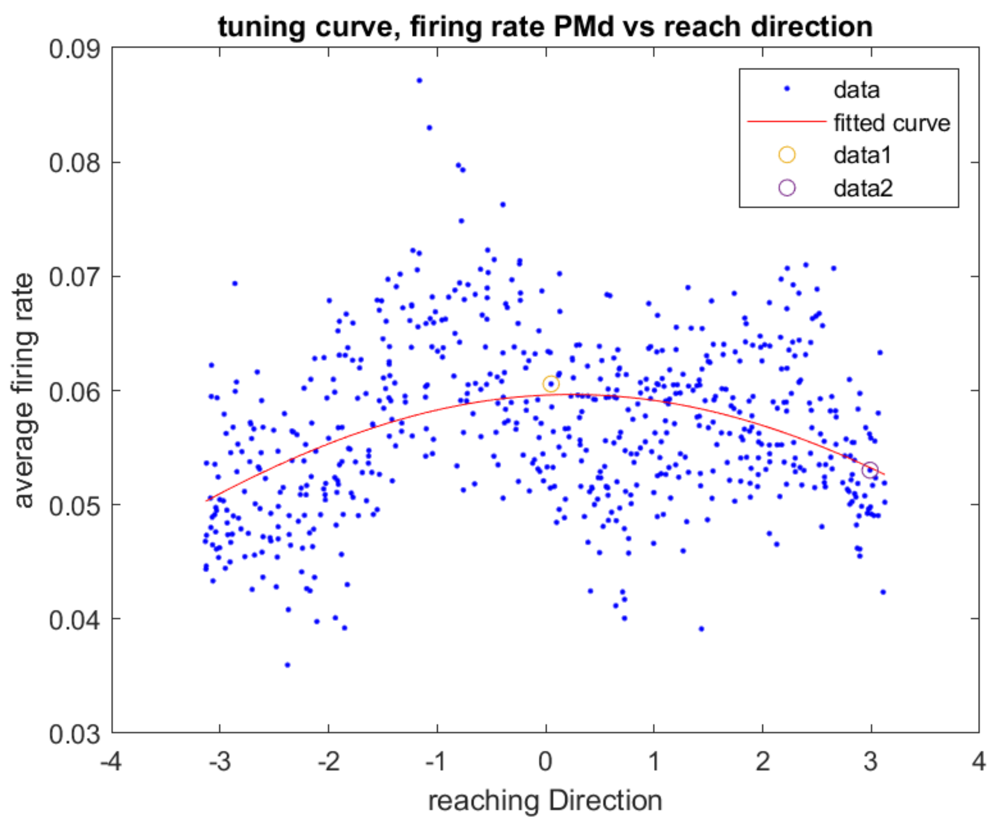
تصاویر MT-S2:

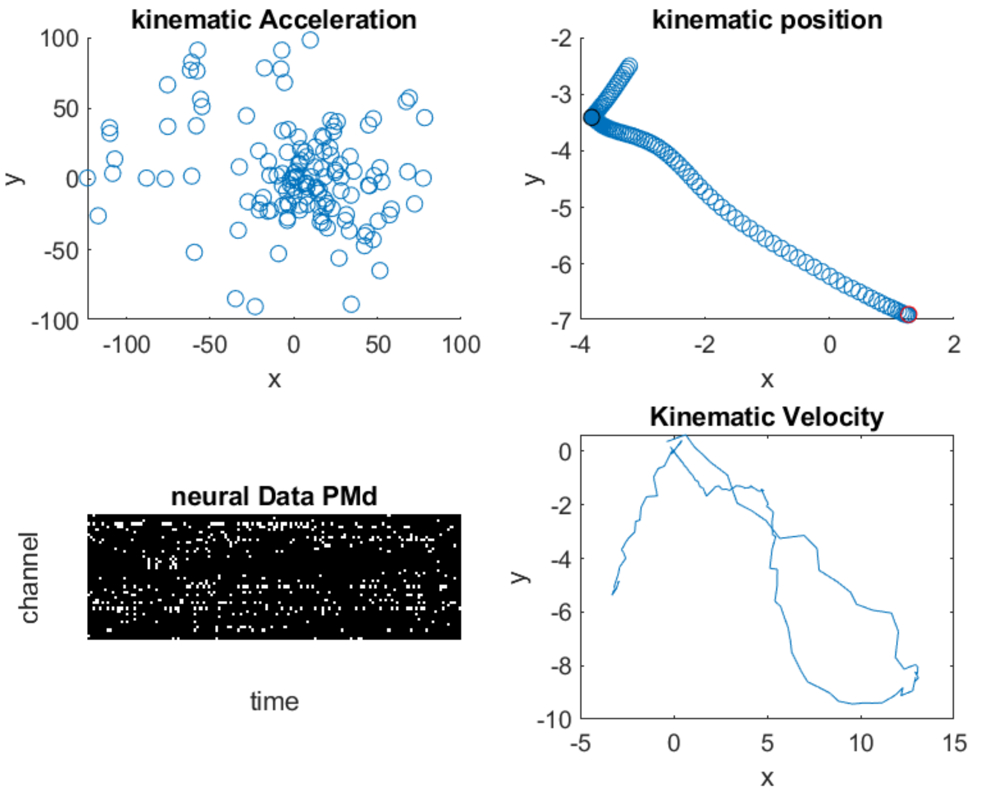


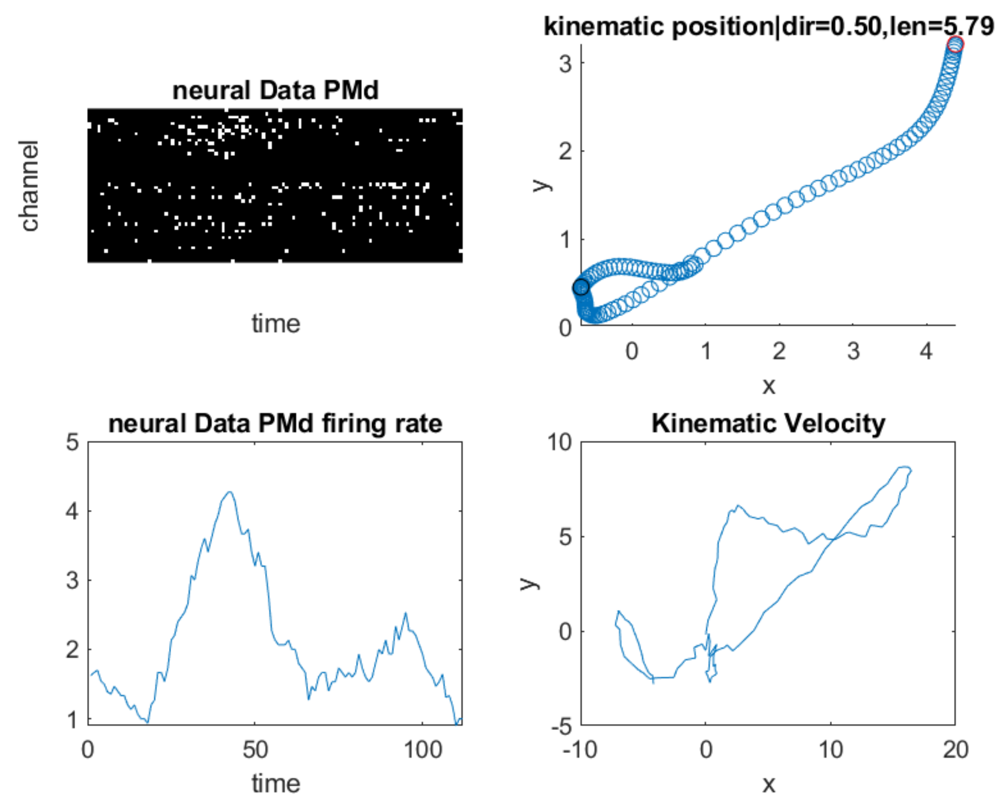
All routes monkey went and reach destinations
animal pos
reach pos
y
x
10
8
6
4
2
0
-2
-4
-6
-8
-10
-15
-10
-5
0
5
10

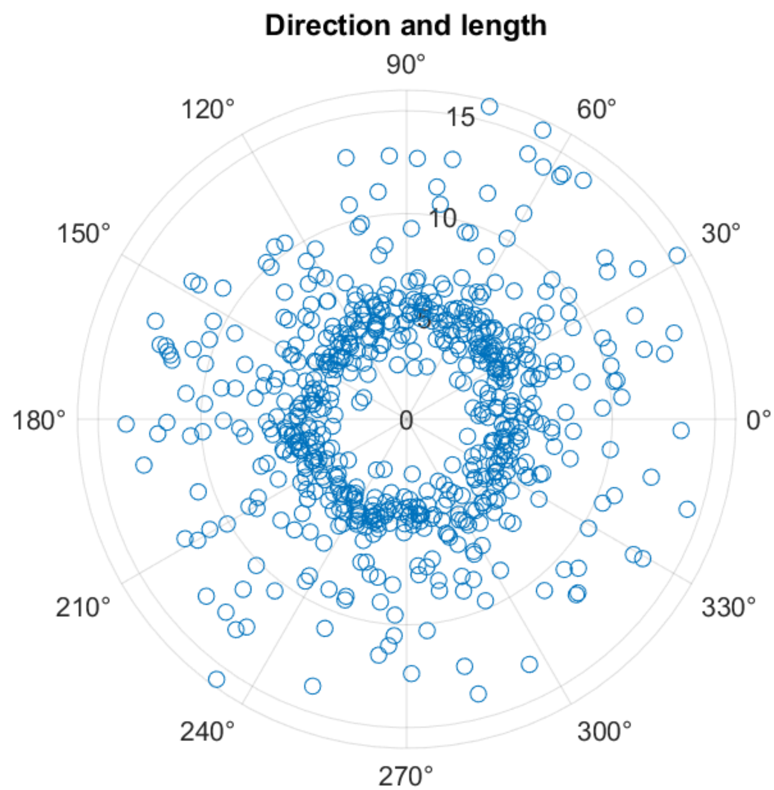

neural Data PMd |Dir0.05
channel
time


neural Data PMd |Dir2.99
channel
time

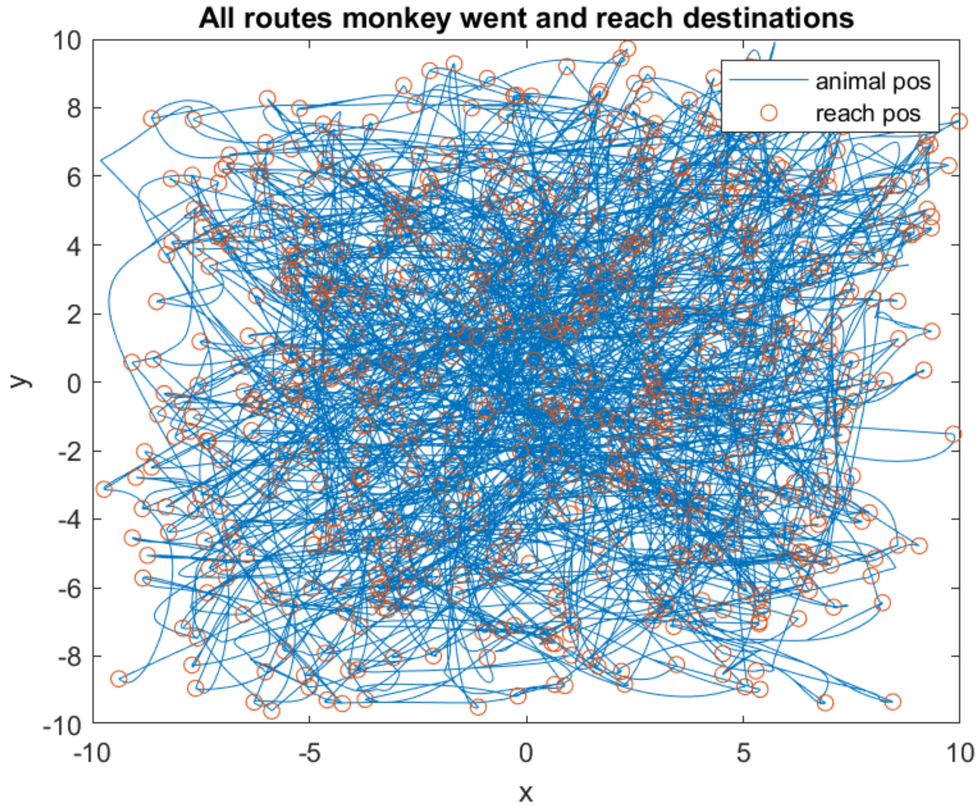

tuning curve, firing rate PMd vs reach direction
data
fitted curve
data1
data2
0.09
0.08
0.07
0.06
0.05
0.04
0.03
-4
-3
-2
-1
0
1
2
3
4
average firing rate
reaching Direction

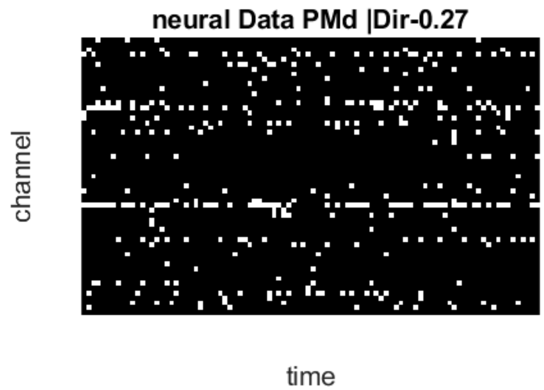

kinematic Acceleration
kinematic position
neural Data PMd
Kinematic Velocity
channel
time
x
y

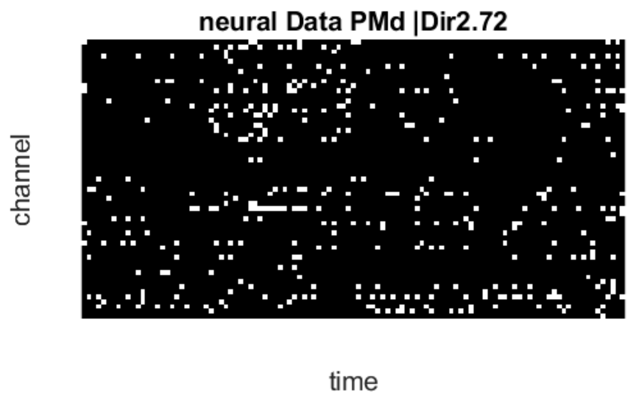

neural Data PMd
kinematic position|dir=0.50,len=5.79
neural Data PMd firing rate
Kinematic Velocity
channel
time
x
y

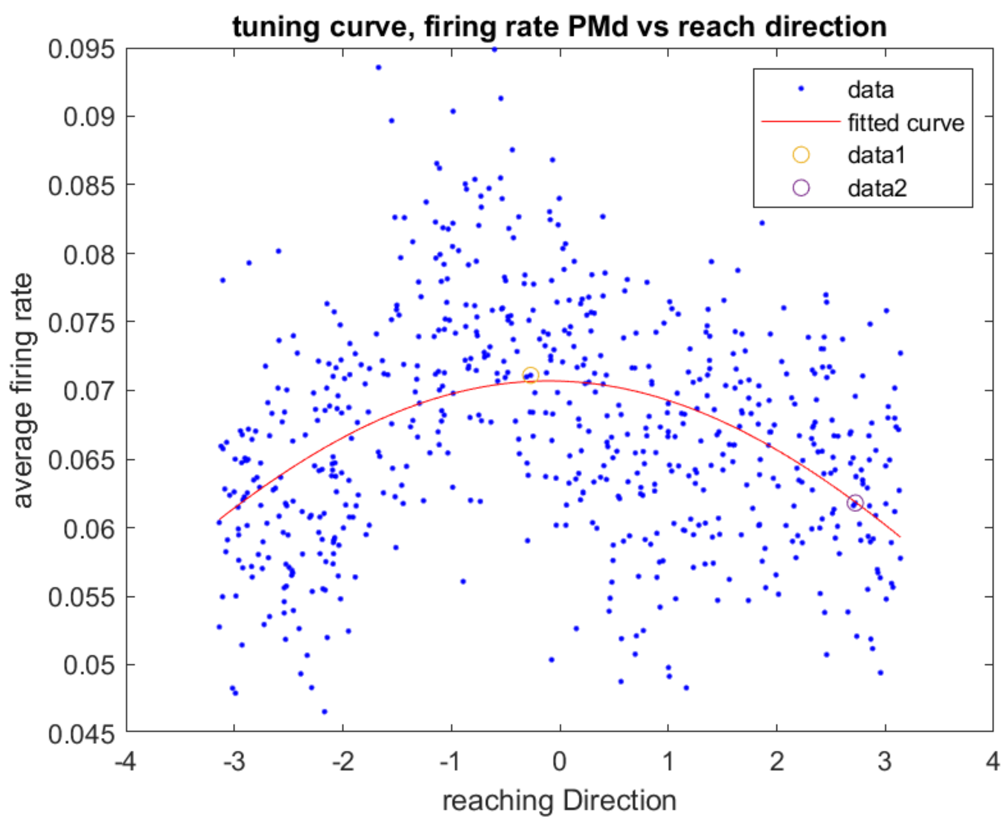

Direction and length
90°
60°
30°
0°
330°
300°
270°
240°
210°
180°
150°
120°
15
10
5
0

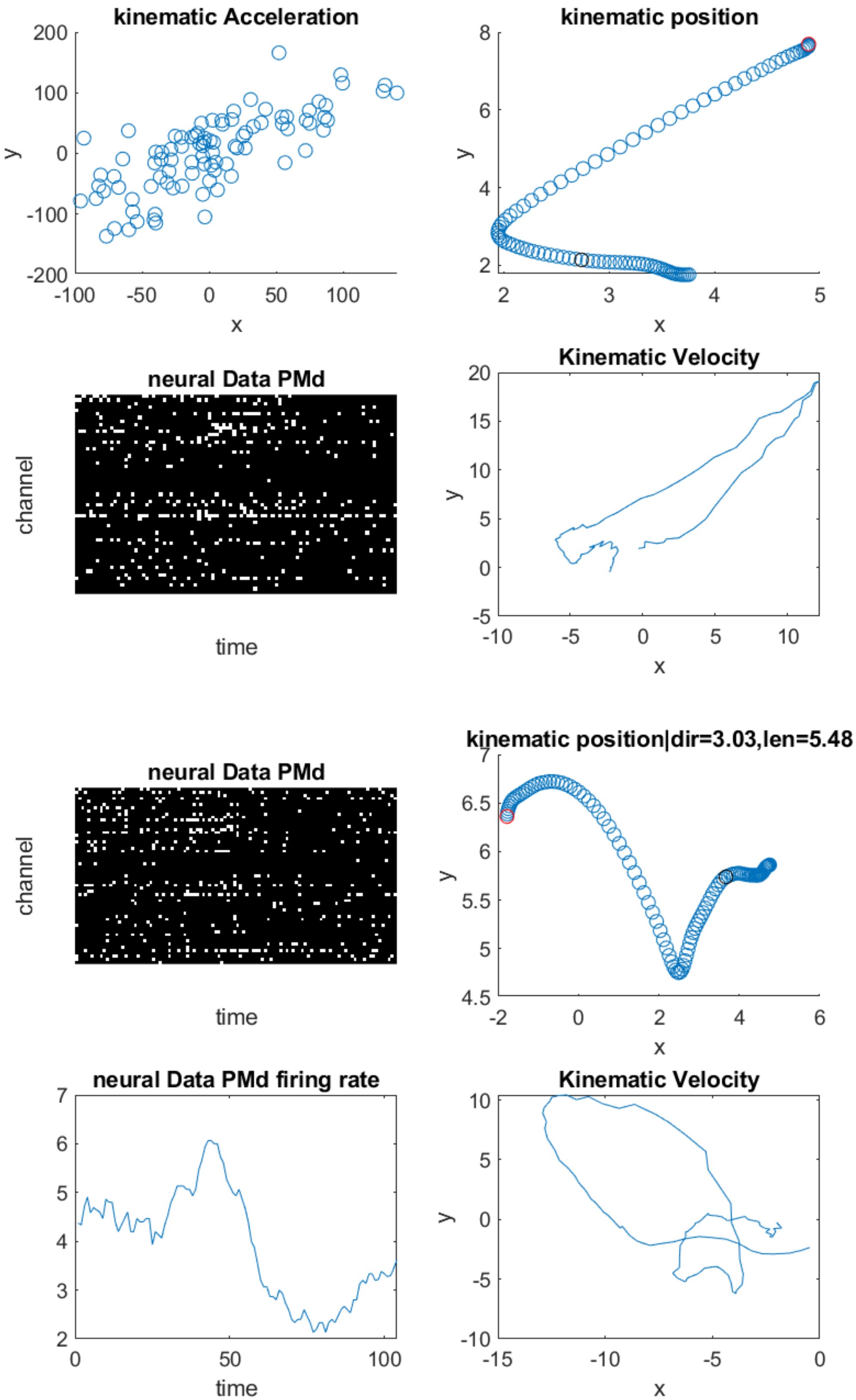
تصاویر MT-S3:

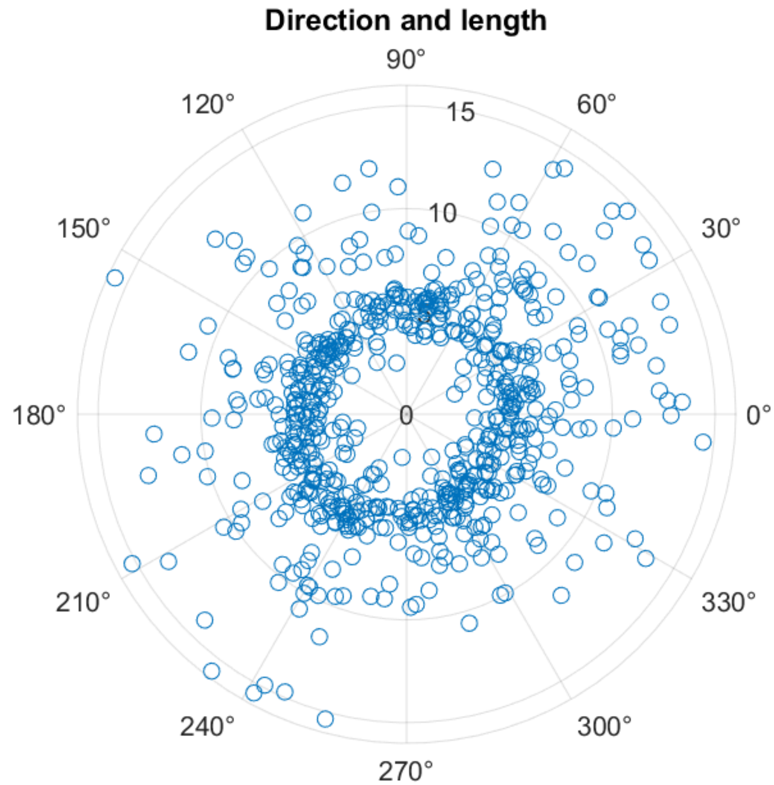

All routes monkey went and reach destinations
animal pos
reach pos
10
8
6
4
2
0
-2
-4
-6
-8
-10
y
-10
-5
0
5
10
x


neural Data PMd |Dir-0.27
channel
time


neural Data PMd |Dir2.72
channel
time


tuning curve, firing rate PMd vs reach direction
data
fitted curve
data1
data2
average firing rate
0.095
0.09
0.085
0.08
0.075
0.07
0.065
0.06
0.055
0.05
0.045
-4
-3
-2
-1
0
1
2
3
4
reaching Direction


kinematic Acceleration
200
100
0
-100
-200
y
-100
-50
0
50
100
x
kinematic position
8
6
4
2
y
2
3
4
5
x
neural Data PMd
channel
time
Kinematic Velocity
20
15
10
5
0
-5
y
-10
-5
0
5
10
x
neural Data PMd
channel
time
kinematic position|dir=3.03,len=5.48
7
6.5
6
5.5
5
4.5
y
-2
0
2
4
6
x
neural Data PMd firing rate
7
6
5
4
3
2
0
50
100
time
Kinematic Velocity
10
5
0
-5
-10
y
-15
-10
-5
0
x


Direction and length
90°
120°
60°
150°
30°
180°
0°
210°
330°
240°
300°
270°
15
10
0

لینک دیتاست:

https://crcns.org/data-sets/motor-cortex/pmd-1/about-pmd-1