Bar sweep analysis

Table of contents

0.1 Breakdown of `process_bars_p2`

This function reads in the frame data:

```
f_data = Log.ADC.Volts(1, :); % frame data
```

and then the voltage data:

```
v_data = Log.ADC.Volts(2, :)*10; % voltage data
```

This is the data across the entire experiment, the three repetitions of the two flash stimuli and the two bar stimuli. The voltage data is multiplied by 10 because during the data acquisition it is downsampled by a factor of 10. (TODO: ADD WHY).

[PNG - 0001] ^General plot of f_data and v_data

0.1.1 Parsing the bar data

The function `src/analysis/protocol2/pipeline/parse_bar_data` is used to parse the frame data to understand when the moving bar stimuli were presented.

1. **Find the range of timepoints during which all of the bar stimuli for each repetition were being presented. This includes both the slow and fast bars.**

Firstly, using the value of the parameter `on_off`, which refers to whether bright ('on') or dark (off') stimuli were used during the protocol, the parameter `drop_at_end` is set to either '-200' for `on_off` == 'on' or '-100' for `on_off` == 'off'.

Then `idx` is set to the timepoints for which the difference in the frame position is equal to `drop_at_end`. This happens both once during the 4 pixel flashes and for the last timepoint of the last 6 pixel flash. The 1st, 3rd and 5th values are removed to remove the timepoints during the 4 pixel flashes and so only the timepoints corresponding to the end of the 6 pixel flashes remain. This is useful because this is the stimulus before the bar stimuli start.

```
idx = find(diff_f_data == drop_at_end);
idx([1,3,5]) = [];
```

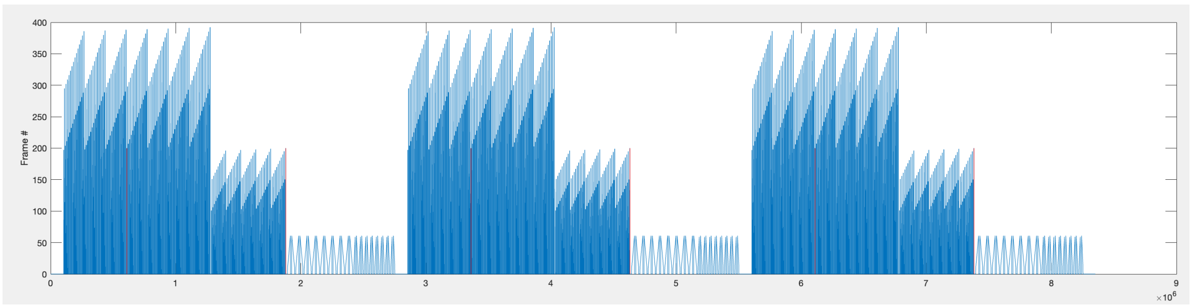

Figure 1: MATLAB figure of the frame position data over the entire P2 experiment (blue) with vertical lines indicating the 6 timepoints in the variable 'idx' (red).

Since each flash stimulus is followed by a 440ms gap, and each bar stimulus is preceded by a 1000ms gap, there is a ~1440ms period before the first bar stimulus being presented and the last of the 6 pixel flashes being shown.

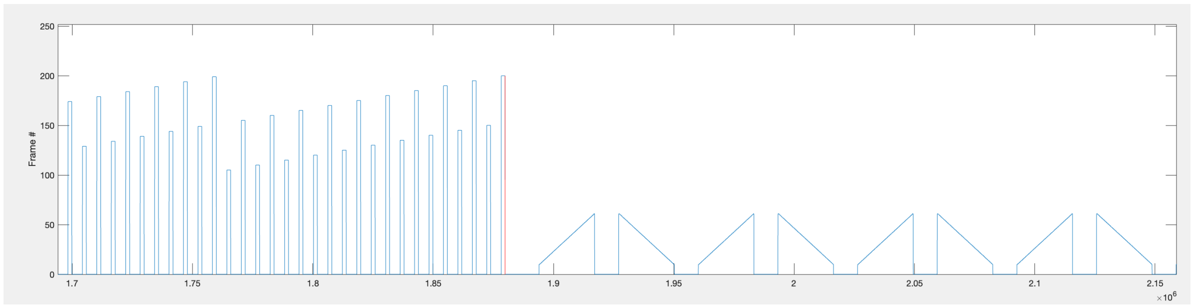

Figure 2: Zoomed in view of the figure above showing only the second timepoint in 'idx' that corresponds to the end of the last 6 pixel flash in Rep 1.

Ultimately, the data will be broken up into chunks of the same length with 1000ms before the bar stimulus and until 900ms after the end of the bar stimulus, but first we want to find the range of timepoints during which ALL bar stimuli (both slow and fast) are presented for each repetition.

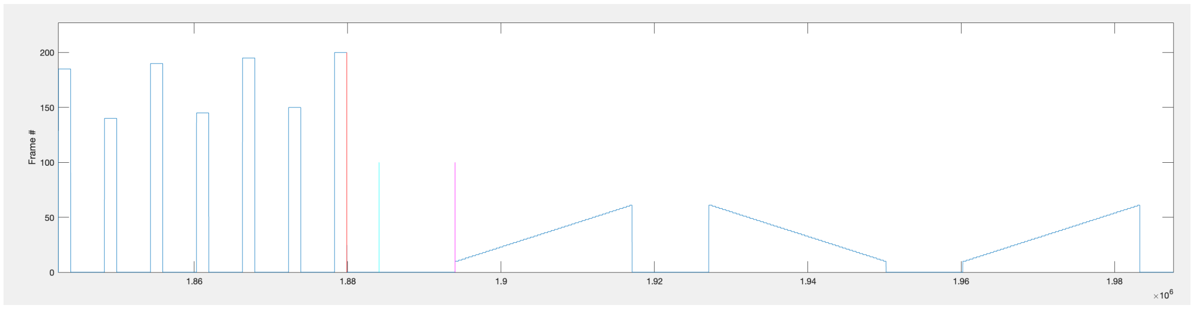

Figure 3: Zoomed in view of frame position data. (Red) Last frame of the last flash of the 6 pixel flashes from Rep 1. (Cyan) Time point after 440ms gap after the last flash and 1000ms before bar stimulus starts. (Magenta) First frame of the first bar stimulus in Rep 1.

The vertical bars are the timepoints that are found in the code. The two timepoints that represent the start and end of the first repetition of bar stimuli are defined as `rep1_rng` and are represented graphically by the magenta vertical bar and the green vertical bar. This range excludes the interval periods before the stimuli start and after the stimuli stop.

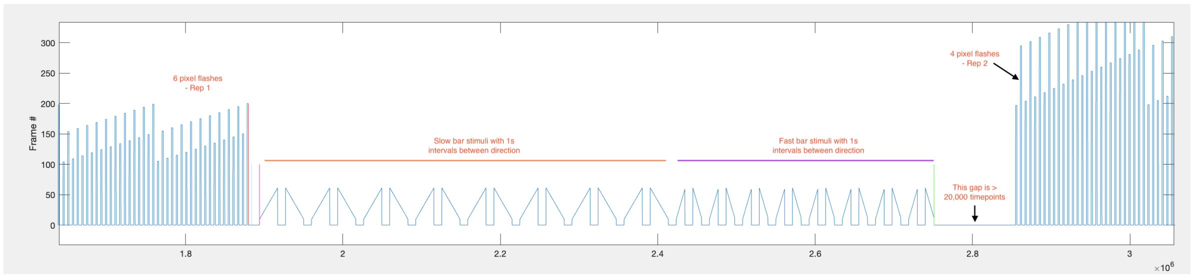

Figure 4: Overview of how the timing of the moving bar stimuli are found using the frame position data. The figure shows the end of the 6 pixel flashes, both bar stimuli and the start of the second repetition of the 4 pixel size flashes.

This is the code that was used to plot the cyan, magenta and green lines:

```
plot([idx(1)+gap_between_flash_and_bars idx(1)+gap_between_flash_and_bars], [0 100], 'c') %
plot([start_f1 start_f1], [0 100], 'm') % first frame of the first moving bar stimulus.
plot([end_f1 end_f1], [0 100], 'g') % last frame of the last moving bar stimulus of the repe
```

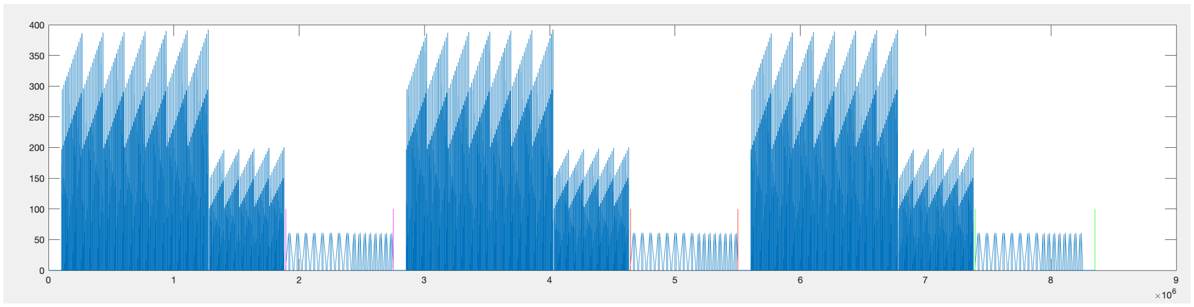

Figure 5: Range of timepoints for the bar stimuli in rep1 (magenta lines), rep2 (red lines) and rep3 (green lines). The third rep uses the last frame of the experiment as the end of it's range. It doesn't matter that this includes the interval time at the end because the actual start and stop times of the bar stimuli are found within these ranges in the next step.

2. **Find the timepoints for when each individual bar stimulus starts and stops.**

Now that we have extracted the range of timepoints during which all of the bar stimuli of one repetition are presented, we now want to extract the timepoints for each individual bar stimulus (one sweep of the bar) within each repetition. To do this, the difference between frame positions are used again. This time we are looking for the timepoints when the frame position changes from 0 (the background interval frame) and a frame when the bar is being presented. For all moving bar stimuli, the bar first appears around frame 10, therefore an absolute difference of >9 in frame position was used to find the timepoints when the bar starts and stops.

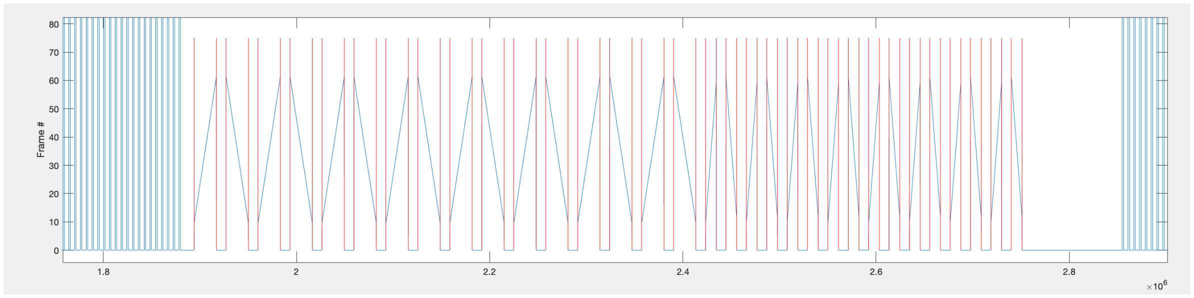

Figure 6: Frame position over one repetition of all of the moving bar stimuli. Red vertical lines indicate the start and end of each individual flash sweep. These values are stored within 'idxs_all{1,1}'

3. **Create the cell array `data` which combines the voltage data for each bar stimulus including 9000ms before and after each sweep of the bar.**

The indices found above and stored in `idxs_all` are then used to extract the relevant voltage data for each bar sweep. Since there is a 1000ms gap between each bar sweep (added in July 2025), 900ms is added before and after the bar sweep timepoints to see the baseline voltage between each sweep.

[PNG - 0008] ^ Frame position (blue) and the magenta vertical lines show the range of data that is included for the first bar and the cyan vertical lines show the range of data to include for the second bar. These ranges include 900ms before and after the bar stimulus itself.

This data is combined into the [32 x 4] cell array `data`. The voltage data for the slow bar stimuli are in rows 1:16 and the voltage data for the fast bar stimuli are in rows 17:32. Each cell contains a linear array of the voltage over the 900ms before the bar stimulus, the bar stimulus presentation and 900ms after the bar stimulus. This means that the slow bar stimuli have voltage data arrays of roughly [1 x 41100] timepoints (10kHz acquisition - 0.9s pre, 2.3s stim, 0.9s post = ~4.1s total) and the fast stimuli of [1 x 29100] timepoints (10kHz acquisition - 0.9s pre, 1.1s stim, 0.9s post = ~2.9s total). The first three columns contain the data for each repetition to each condition adn the fourth column is the mean across the three repetitions. The size of the linear arrays are trimmed to the length of the shortest repetition, concatenated and then averaged using `mean()`.

[PNG - 0009] ^ Example `data` cell array.

The cell array `data` is returned by the function `parse_bar_data` and is used for plotting and analysing the responses to the bar stimuli.

0.1.2 Plotting the bar data

1. **Circular timeseries plot with central polar plot for both speeds. (`src/analysis/plotting/plot_timeseries_polar_bars`)**

This function plots the timeseries voltage data per condition for each repetition in light grey and the mean response across conditions in light blue for the fast stimuli and dark blue for the slow stimuli. These timeseries plots are positioned in a circle, with each plots position corresponding to the direction in which the bar stimulus was moving. For instance, the plot at the very top of the circle (N position on a compass) corresponds to the response of the cell to a horizontal bar moving from bottom to top. Whereas, the plot at the right of the circle (E position on a compass) corresponds to the response of the cell to a vertical bar moving from left to right. These timeseries plots include the 900ms of interval beforehand and 900ms after the end of the bar stimulus. Thin vertical black lines are added onto the subplot to show these times.

A polar plot is positioned in the centre of the timeseries plots. To generate the polar plot, the maximum of the mean response of the cell to each direction is calculated, the median voltage across the entire recording is subtracted from this value, then it is used as the magnitude of the polar plot. Specifically, the mean response across all repetitions for each condition is extracted. This data is then trimmed to exclde the 0.9s interval before and the last 0.7s of the interval after the flash. Then, the 98th percentile value from this trimmed data is found. This value is added to the array `max_v` which is the size [n_conditions, n_speeds] - so in this case [16 2]. The minimum value (2nd percentile value) in the second half of this trimmed data is stored in a similar way into the array `min_v`. These arrays are returned by the plotting function.

[PNG - 0010] ^ Circular timeseries + polar plot. Two colours = different speeds.

2. **Polar plot with vector sum resultant angle arrow (`src/analysis/plotting/plot_polar_with_arro`**

Plots the same polar plot as the one in the centre of `plot_timeseries_polar_bars` but as a full figure in itself. This function first calls the function `src/analysis/analyse_bar_DS/vector_sum_polar` to find the `resultant_angle` of the vector sum of the polar plot and then adds an arrow pointing in this `resultant_angle` on top of the polar plot. The arrow is hard coded to have a fixed magnitude of 30. This is because the polar plots use the median voltage subtracted peak voltage values and given the current results the rlim of [0 30] fits most data.

[PNG - 0011] ^ polar plot with resultant angle arrow

3. **Heatmap of max values per direction (`src/analysis/plotting/plot_heatmap_bars`)**

This function takes in the array `max_v` (size:[n_conditions, n_speeds]) and produces a heatmap of these values.

[PNG - 0012] ^ Heatmap plot of max_v

0.1.3 Calculating metrics from the bar data

The responses to the moving bar stimuli are then further processed to the calculate:

- How symmetric the polar plots are.
- The direction selectivity index (vector sum method and PD-ND method).

These are calculated for both the slow and the fast speeds and all of these metrics are stroed in a struct `bar_results` which is saved as the file `peak_vals....mat` in the `results_folder`. The steps to generate these metrics are outlined below:

1. **The timeseries data is reordered using `src/analysis/helper/align_data_by_seq_angles`**

Initially, the cell array `data` has the rows (conditions) ordered by the order in which they are presented which flows through each orientation in one direction and then the opposite direction. This function simply reorders the rows in `data` so that the rows correspond to sequential angles (i.e. 0, 1/16pi, 2/16pi etc..).

2. **Find the PD and then reorder the data again so that the PD is always positioned to pi/2 (up). (`src/analysis/helper/find_PD_and_order_idx`)**

Here, the preferred direction (resultant angle of the vector sum of the repsonse) is calculated again. (TODO - update this code so that it uses the same function as above…), and it finds which of the 16 directions is closest to the resultant angle of the vector sum. It then shifts the responses in this position to be aligned to pi/2 ('N' on the polar plot) and rearranges the other responses accordingly.

[PNG - 0013] ^ polar plot with the PD repositioned to be in the pi/2 direction

This code also uses the functions:

- `src/analysis/helper/compute_FWHM` to calculate the width of the polar plot at half of the maximum response.
- `src/analysis/helper/compute_circular_var` to compute the circular variance. This function returns a number between 0 and 1. 0 would imply that the cell only responds in one direction and has very sharp tuning, 1 would imply the cell responds uniformly to all directions and has very broad tuning.
- The function `circ_vmpar` from the MATLAB Circular Statistics Toolbox. This function estimates the parameters of a von Mises distribution and returns `thetahat` (preferred direction) and `kappa` (concentration parameter).

These metrics are found for the bar stimuli moving at both speeds.

The data that is eventually saved includes `data`, `data_ordered` - ordered by angle and `data_aligned` - data ordered with PD in the pi/2 position. The order `ord` in which to rearrange the initial `data` structure into the version with the PD in the pi/2 position is also saved, as well as `d_slow` which is the max_val data per direction reordered so that the PD response is in the pi/2 position and the struct `bar_results` which contains all of these metrics for the stimuli at both speeds.

`resultant_angle` - the angle of the preferred direction (PD) of the cell is returned by the overall script `process_bars_p2`.

0.1.4 Breakdown of variables in `process_bars_p2`

- `max_v` - 98th percentile value of the raw voltage during the presentation of each bar stimulus. This is calculated within `plot_timeseries_polar_bars` and fed into the function `find_PD_and_order_idx`. Within `find_PD_and_order_idx`, the median voltage is then subtracted from this array to return `responses`.
- `responses` - median subtracted `max_v`.
- `magnitude` - normalised between 0 and 1. Magnitude of vector sum.
- `d_slow/d_fast` - angle and median-subtracted voltage responses to each direction. These angles are now shifted so that the maximum result is aligned with pi/2, not the actual direction in which the bar was moving. The response data is unaffected though and can be used for quantification of the tuning width.
- `cv` - circular variance of the responses. If CV = 0, all responses are in one direction (sharp tuning). If CV = 1, responses are uniformly spread (broad tuning).
- `thetahat` and `kappa` - parameters of the von Mises distribution fitted to the data. `thetahat` is the preferred direction (PD) and `kappa` is the concentration parameter (higher values = sharper tuning).

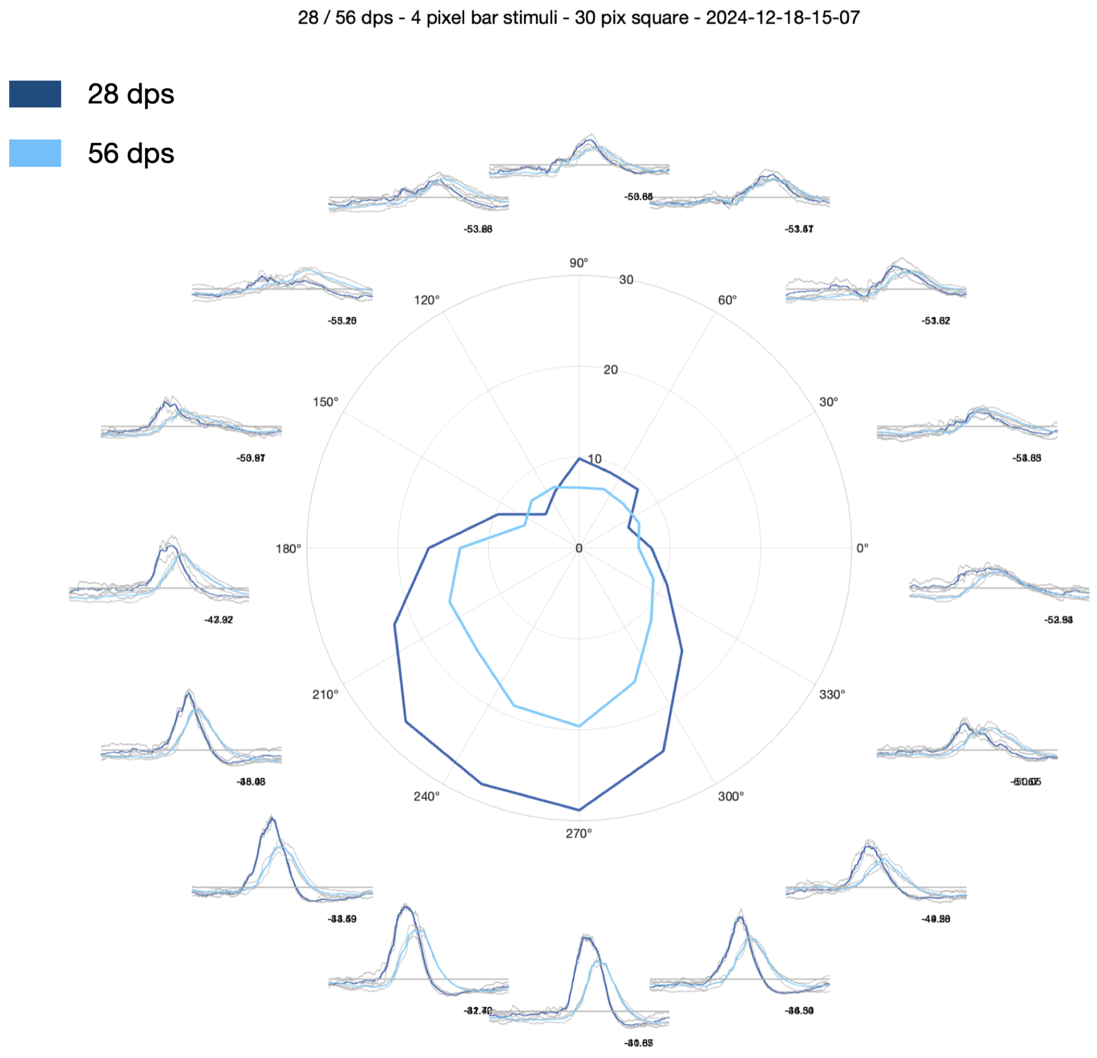

Figure 7: Example polar plot of bar sweep responses in a control RNAi fly to P2 in Dec 2024.

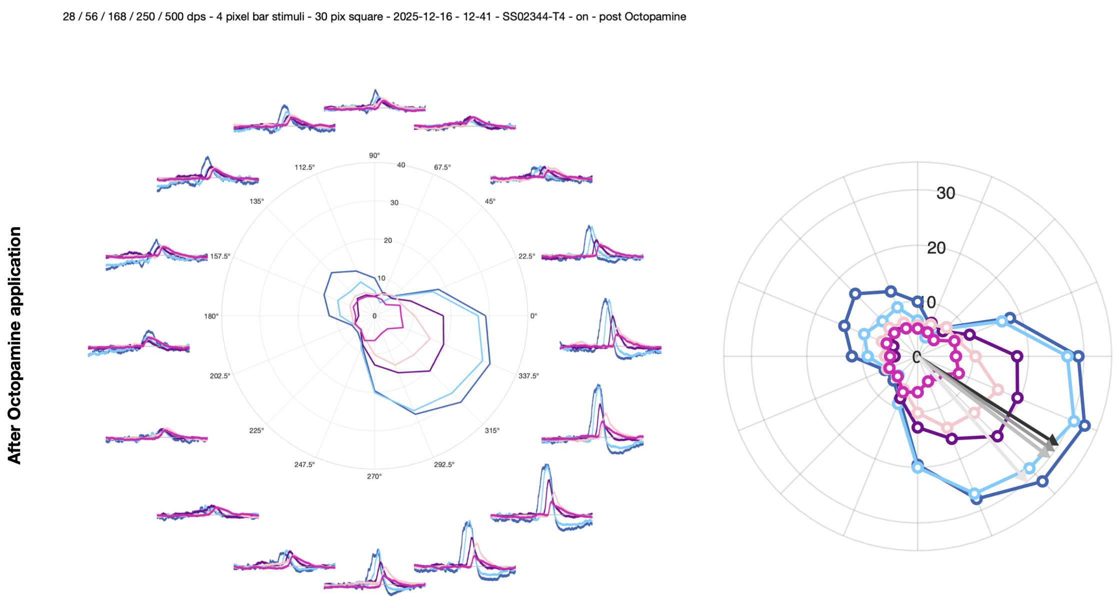

Figure 8: Example polar plot of a T4 cell after Octopamine application showing the responses to the 5 different bar sweep speeds in the P2 version used in Dec 2025.