# How to scare a fish (school)

# with (a) Python

Andrej Warkentin
@awakenting
PyData Berlin 2018
07.07.2018

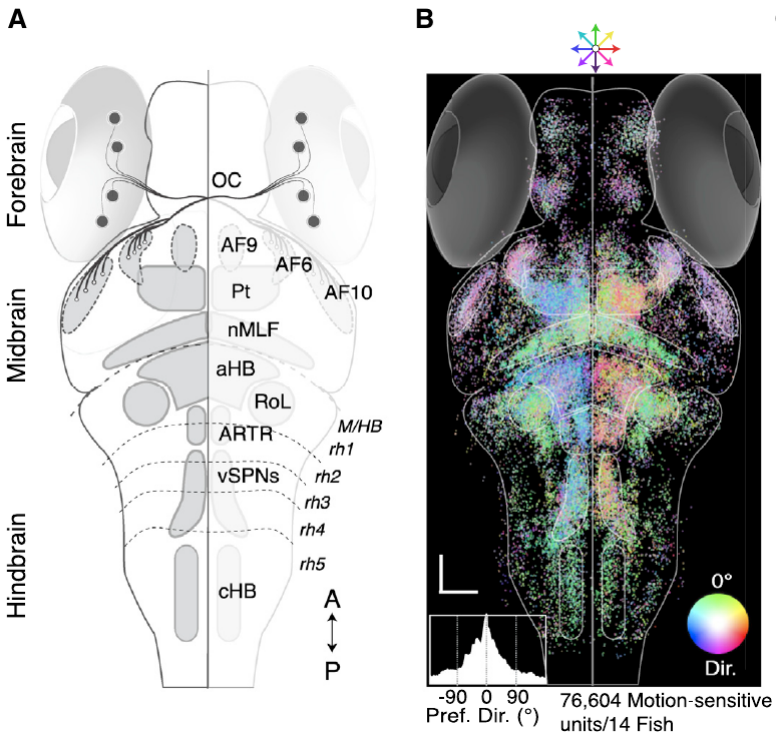
## Outline

- what I did
  - neuronal model for startle responses in fish
  - coupling of neuronal model with collective behavior model

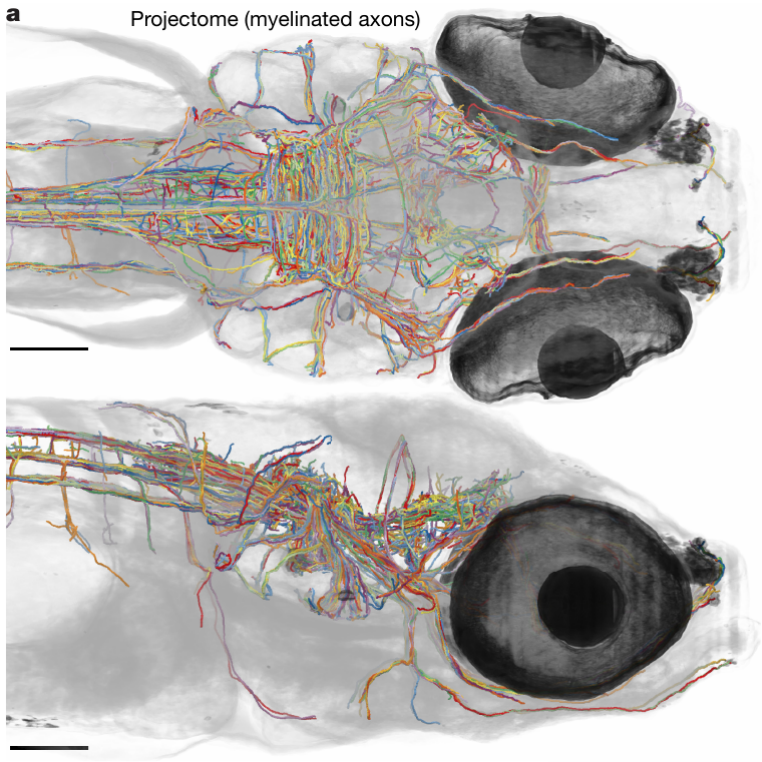
- how I did it
  - and with which python packages

## A neuronal model for visually evoked startle responses in schooling fish

## Why fish?

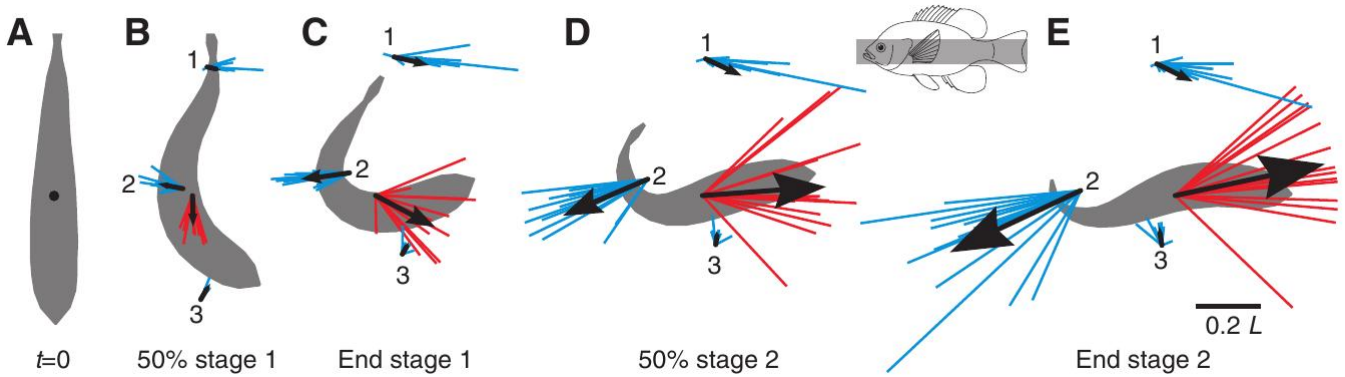



Naumann et al. 2016



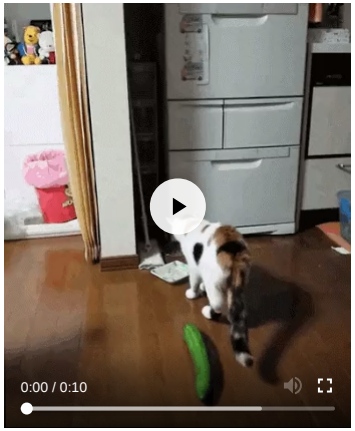

Hildebrand et al. 2017

## Startle Response?



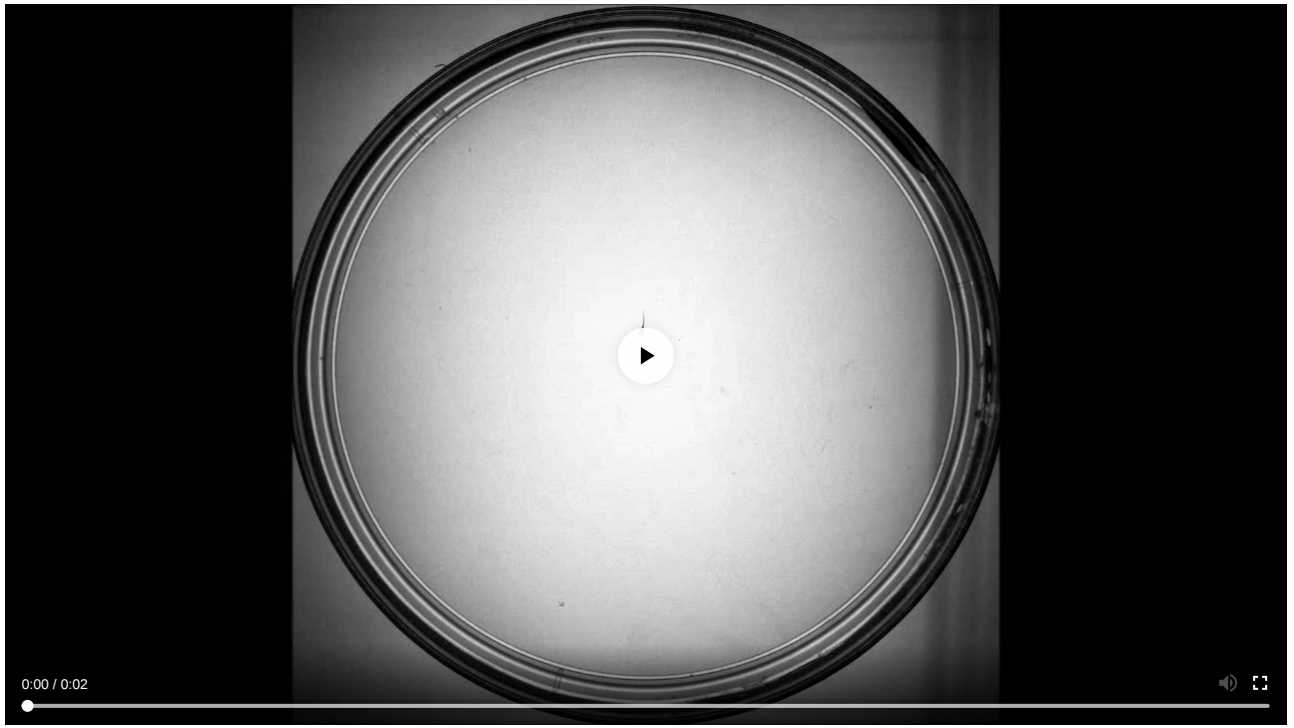

Tytell and Lauder 2008

## Startle response of a cat to a model predator:

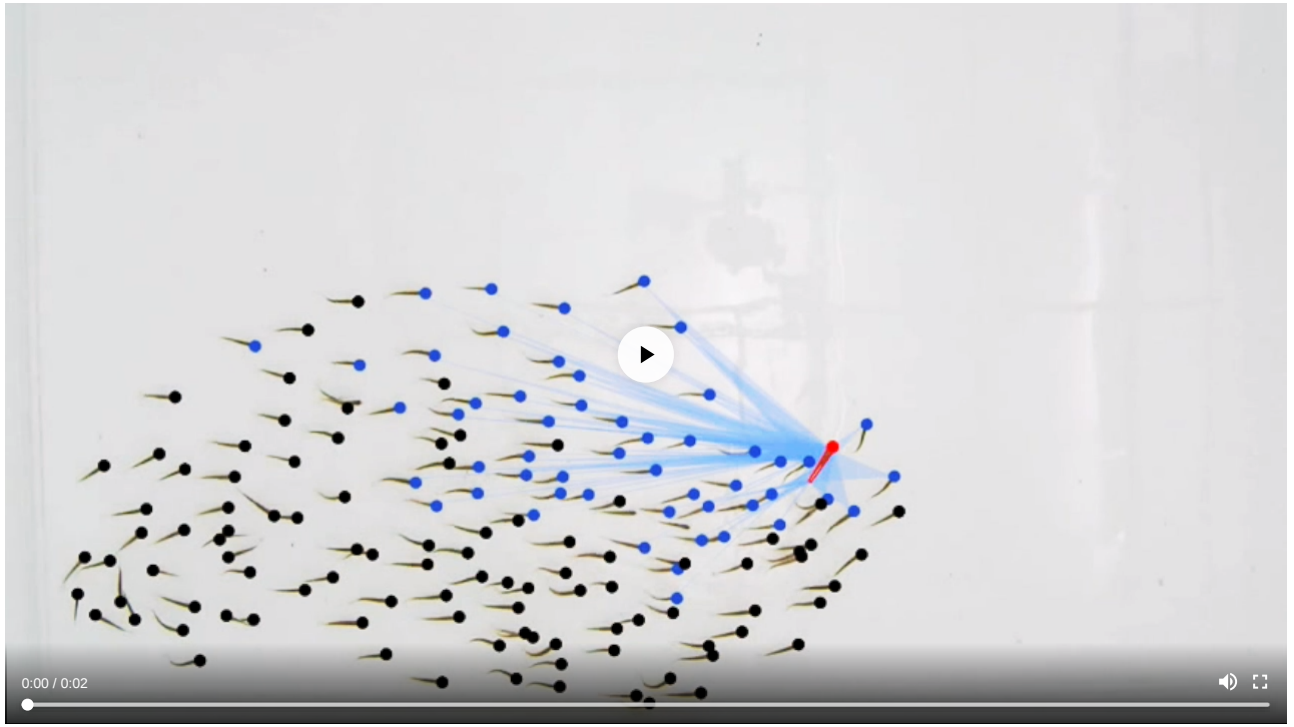



source: https://redd.it/3cmrj5

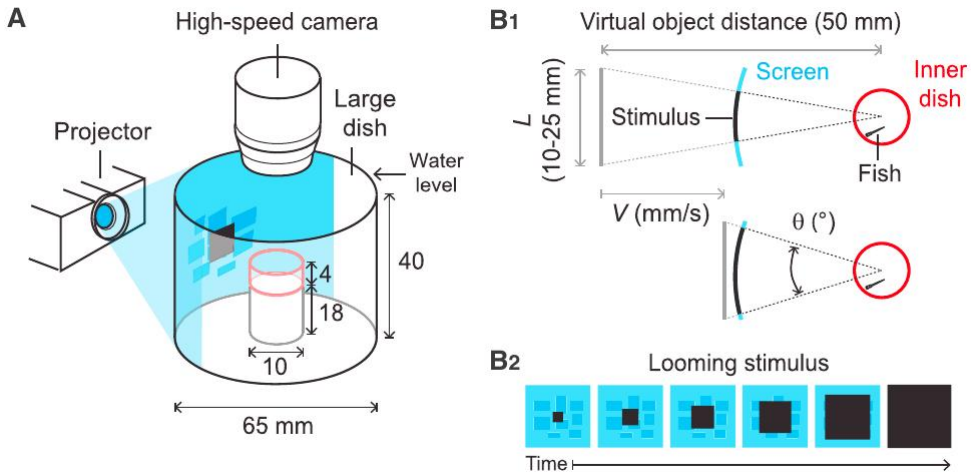
## Startle response of a fish:



**In a fish school:**

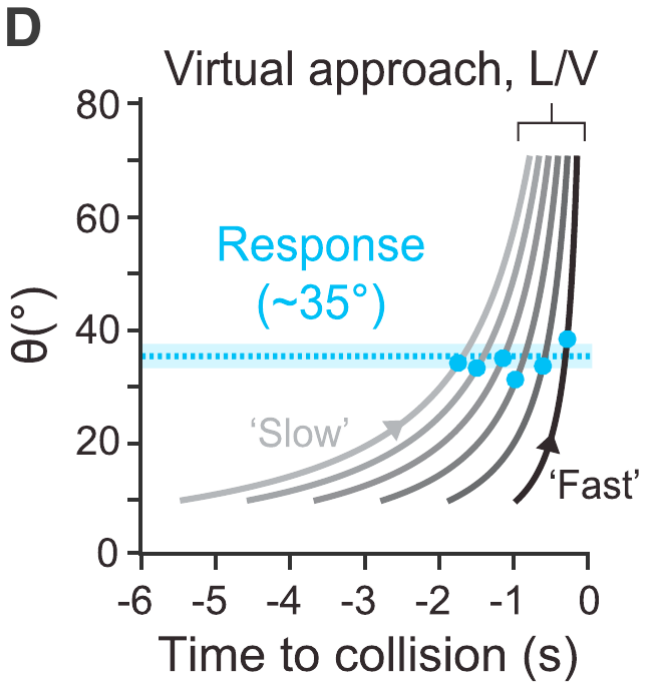



**Visually evoked startle responses**

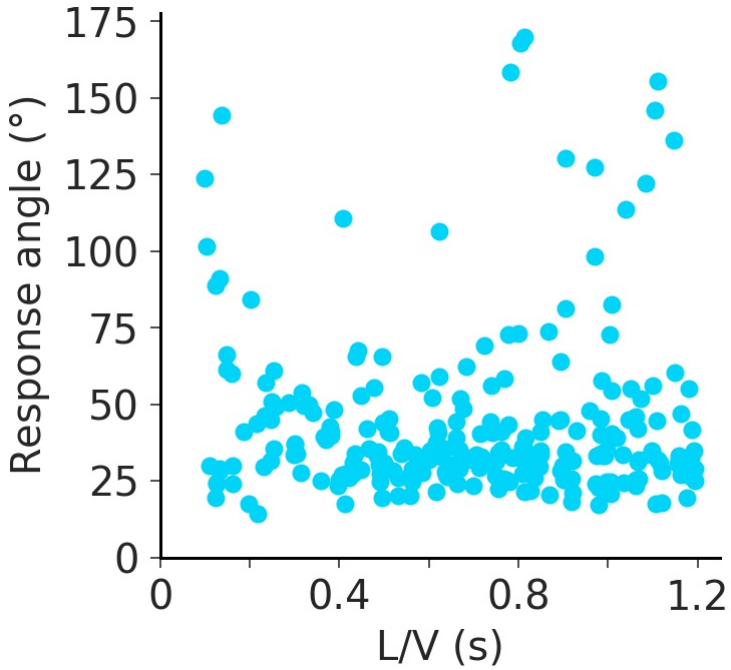



Bhattacharyya et al. 2017

**Experimental response properties**



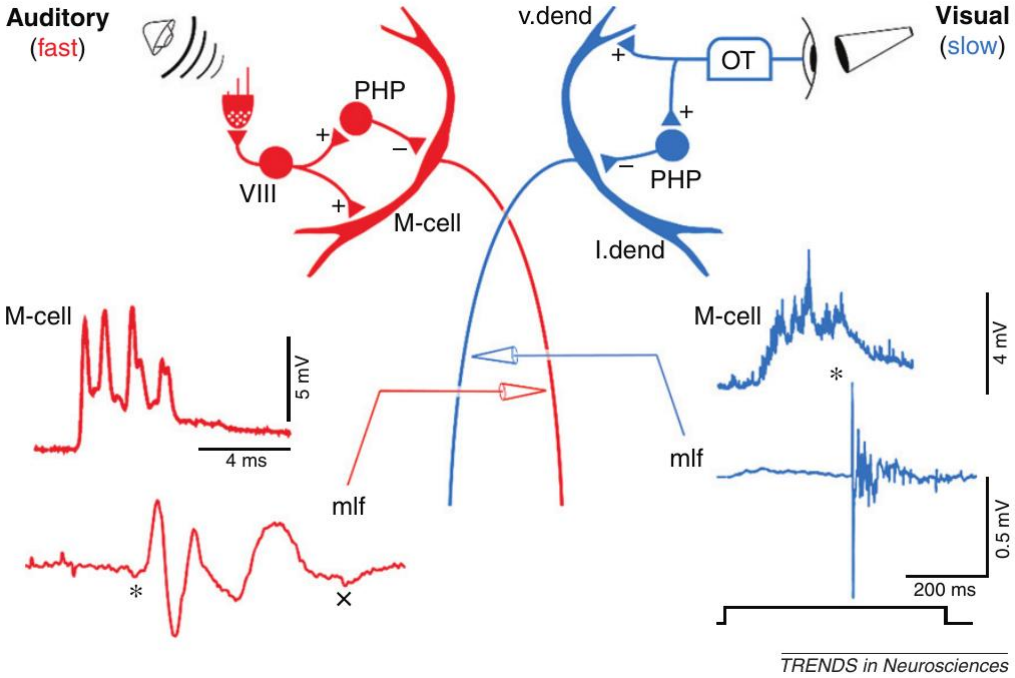

Bhattacharyya et al. 2017



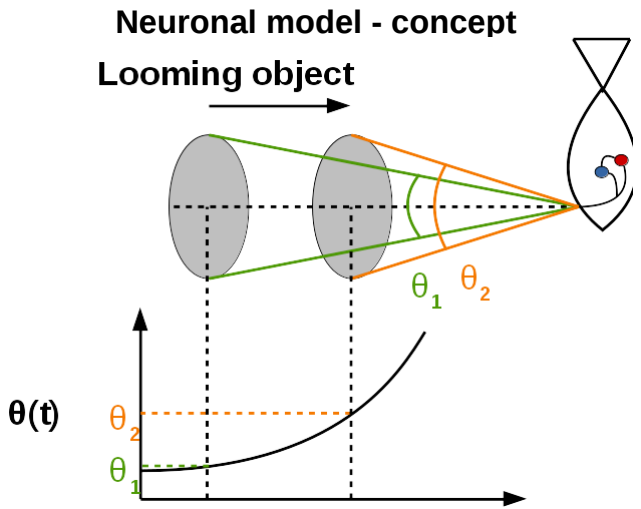

Bhattacharyya et al. 2017

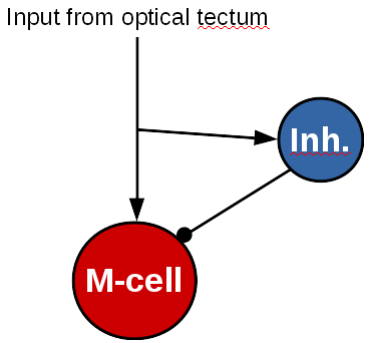
**Neuronal model**

**Motivation: The Mauthner cell**




Pfaff et. al 2012






## Neuronal model- equations

**Input:**

$$I(t) = f(\theta(t))$$

$$\theta(t) = 2 \cdot \arctan(\frac{L/2}{distance})$$

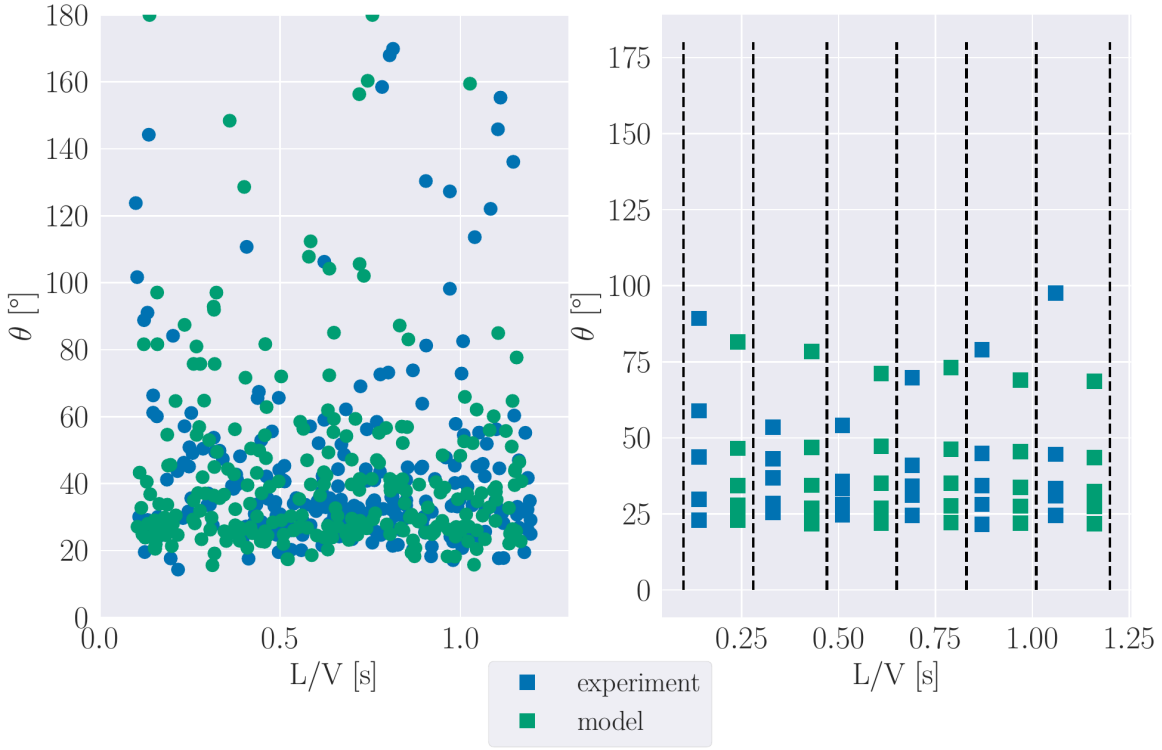
**Inhibitory population:**

$$\tau_\rho \frac{d\rho}{dt} = -(\rho(t) - \rho_0) + a_\rho I(t) + \eta_\rho(t)$$

**LIF for Mauthner cell:**

$$\tau_m \frac{dV_m}{dt} = -(V(t) - E_L) + R_m I(t) - \rho(t) + \eta_m(t)$$

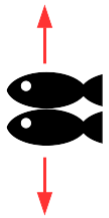
## Model fitting result



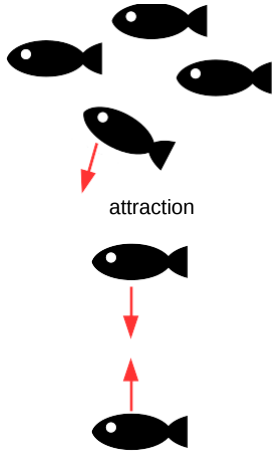

**Collective behavior model**

**Social forces**

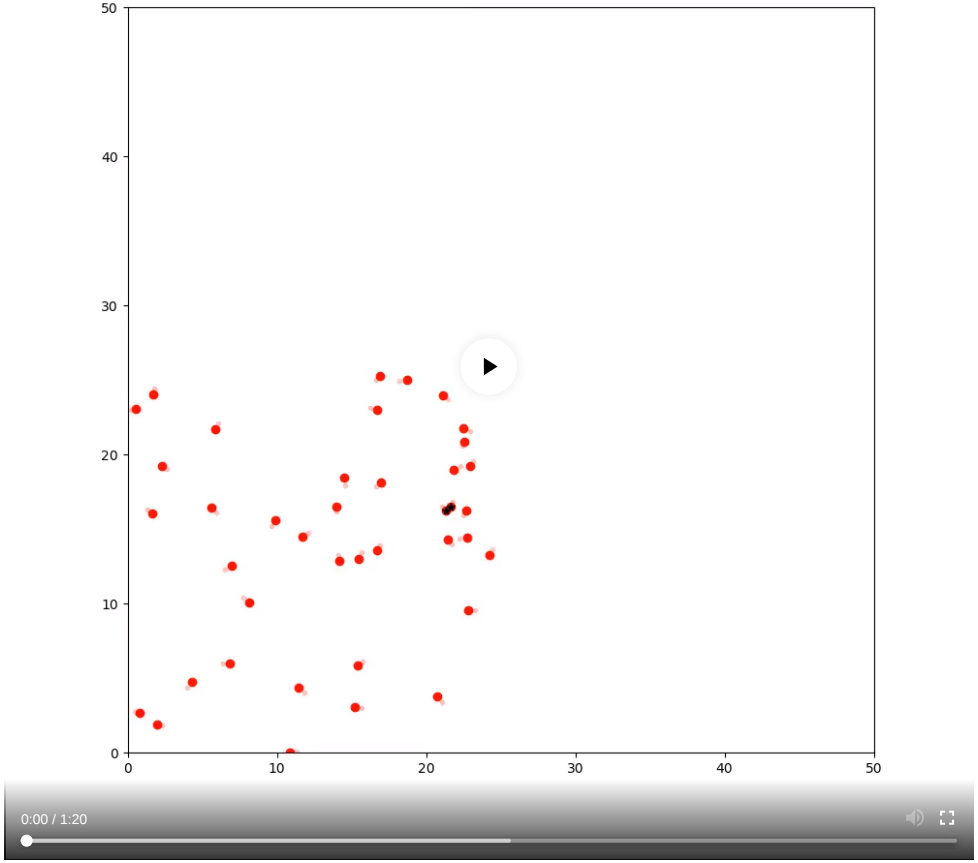
repulsion



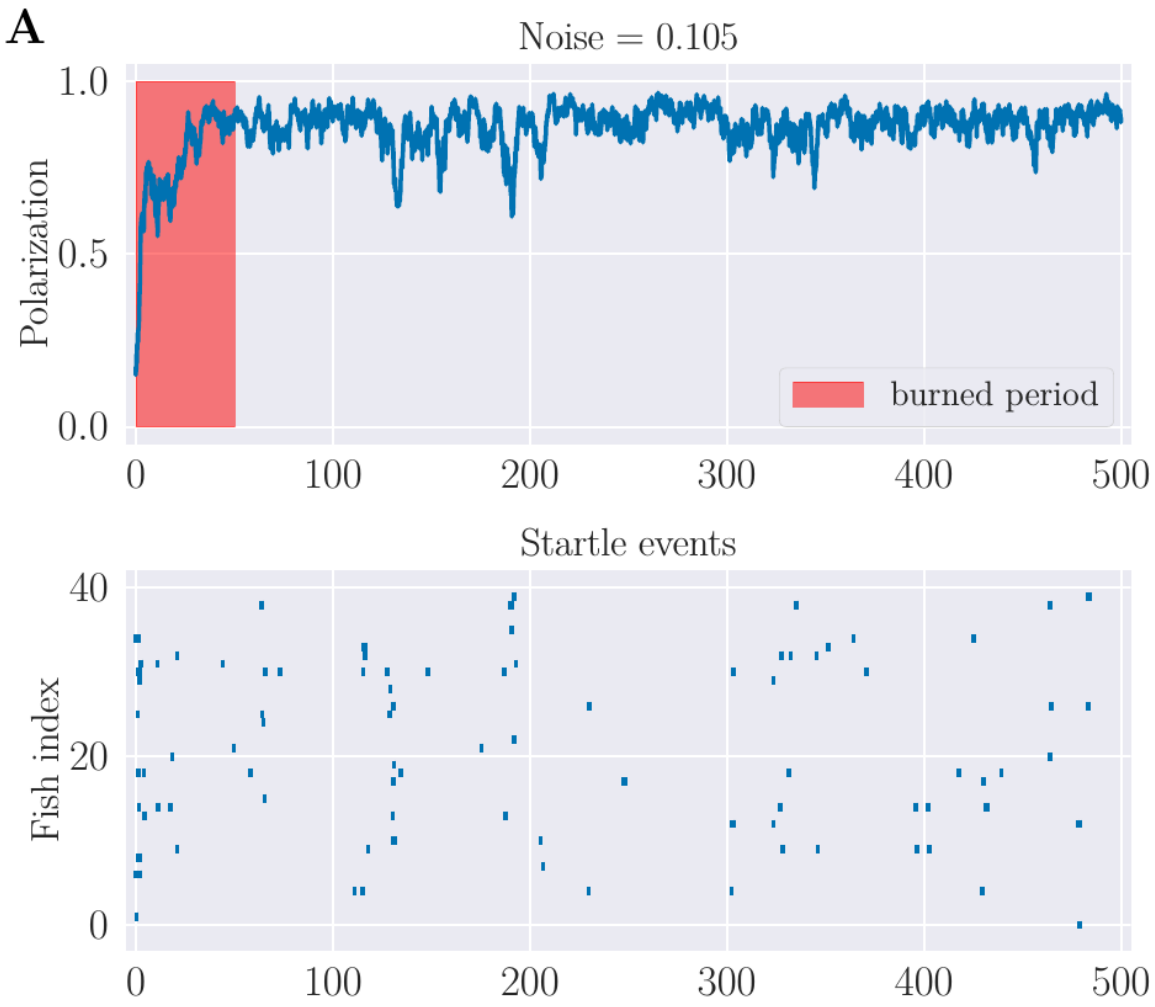
alignment



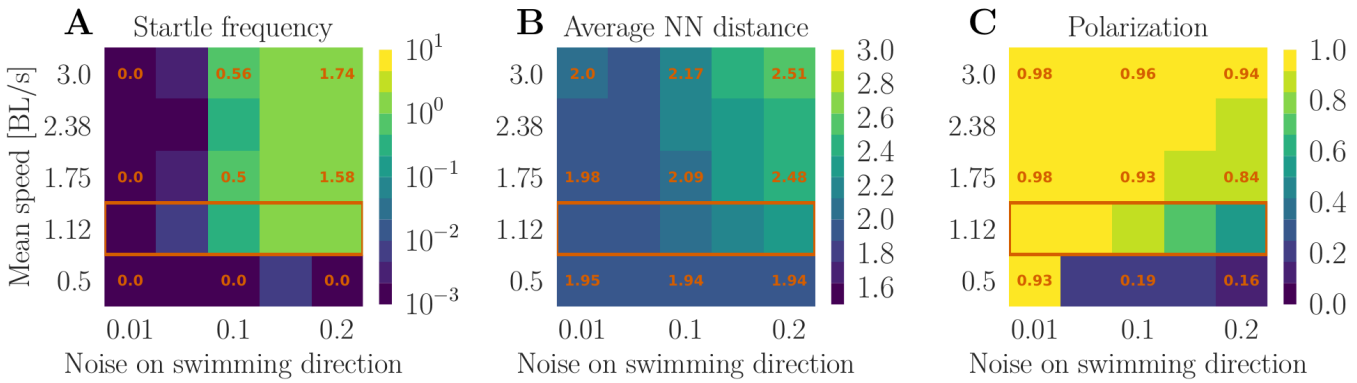

**Simulated fish school**




**Behavioral measurements**




**Parameter exploration**




## How did I do it?

## Accelerating code with numba:

In [5]:
```
from numba import jit

@jit
def jit_ffi_model(lots of parameters):
    # variable initializations
    t = 1
    while t < ntime_steps:
        # model calculations:
        # calculate activation of inhibitory population
        # calculate lif dynamics with inhibitory input

    return time_points, v_m, t_spks, idx_spks, rho_inh
```

## Data storage with pandas

In [7]:
```
import pandas as pd

def run_fit_validation():
    data_cols = ['result1', 'result2']
    data_dict = dict([(col_name, []) for col_name in data_cols])

    for dat_idx in range(nparam_sets):
        for i in range(nreps):
            #calculations of result1 and result2

            result_values = [result1, result2]
            for col, value in zip(data_cols, result_values):
                data_dict[col].append(value)

    fit_df = pd.DataFrame(data_dict)
    fit_df.to_hdf('filepath/fitting_validation_error_v4.hdf5', key='fitting_results', mode='w')
```

## Managing simulations with pypet

In [ ]:
```
from pypet import Environment
from pypet import pypetconstants
from pypet.utils.explore import cartesian_product

def run_sim(traj):
    # initialize system parameters
    # initialize prey parameters

    outData, agentData = sw.SingleRun(paraSystem, paraFish)
    store_outdata(traj, outData)

env = Environment(trajectory='looming_swarm',
                  filename=filename,
                  overwrite_file=True,
                  file_title='looming_swarm_simulation',
                  comment='The first exploration',
                  multiproc=True,
                  ncores=6,
                  use_pool=True,  # Our runs are inexpensive we can get rid of overhead
                  )
```

In [ ]:
```
# The environment has created a trajectory container for us
traj = env.trajectory

# Add both parameters
traj.f_add_parameter('N', 40, comment='Number of fish')
traj.f_add_parameter('L', 50, comment='Length of the quadratic field')
traj.f_add_parameter('total_time', 500, comment='time of the simulation in seconds (per defintion)')
traj.f_add_parameter('dt', 0.001, comment='The size of the time step')
# add many more parameters

# Explore the parameters with a cartesian product
traj.f_explore(cartesian_product({'seed': np.arange(200, 203).tolist(),
                                  'speed0': np.linspace(0.5, 3.0, 5).tolist(),
                                  'noisep': np.linspace(0.01, 0.2, 5).tolist(),
                                  'int_type': ['matrix', 'voronoi_matrix'],
                                  'vis_input_method': ['max', 'knn_mean', 'knn_mean_deviate']
                                  }))

# Run the simulation
starttime = time.time()

env.run(run_sim)
```

## Animations with matplotlib

In [ ]:
```
import matplotlib.pyplot as plt
import matplotlib.animation as manimation

def plotSavedAnimation(positionList, startle_list, L, doblit=False, sleepTime=0.01):
    FFMpegWriter = manimation.writers['ffmpeg']
    metadata = dict(title='Movie Test', artist='Matplotlib', comment='Movie support!')
    writer = FFMpegWriter(fps=15, metadata=metadata)

    fig = plt.figure(99, figsize=(10, 10))
    ax = plt.subplot()
    points = ax.plot(x, y, 'ro')[0]
    pointstail = ax.plot(x, y, 'r.', alpha=0.2)[0]
    firings = ax.plot(0, 0, 'k*')[0]

    with writer.saving(fig, "firing_map_test.mp4", 100):
        for step in range(1, len(positionList)):
            # get new positions

            points.set_data(x, y)
            pointstail.set_data(xtail, ytail)
            firings.set_data(firingsx, firingsy)

            writer.grab_frame()
    plt.close(fig)
```

## Summary

- how to scare a fish: show a quickly expanding black disk or square
- we developed a neuronal model that can reproduce experimental behavior
- we can couple the neuronal model with a collective behavior model
- there are many helpful python packages for computational modeling work

# Thanks


- Collective Information Processing group
- Open source software:
  - Numpy
  - Scipy
  - Sklearn
  - Matplotlib
  - Seaborn
  - Ipython
  - Jupyter Notebook
  - Pypet (https://pypet.readthedocs.io/en/latest/ (https://pypet.readthedocs.io/en/latest/))
- Tools for this presentation:
  - Jupyter Notebook
  - RISE (https://github.com/damianavila/RISE (https://github.com/damianavila/RISE))


## Questions ?

Also at github.com/awakenting/master-thesis

Or on twitter: @awakenting