



Analiza Badań Okulograficznych i Symulacja Ruchu Oka

Massive-Scale Gaze Analytics and Eye Movement Simulation

Andrzej T. Duchowski

School of Computing
Clemson University
Clemson, SC, USA

SWPS Seminar 2014: December 16, 2014



CLEMSON UNIVERSITY
EYE TRACKING LAB
SCHOOL OF COMPUTING

CLEMSON
UNIVERSITY

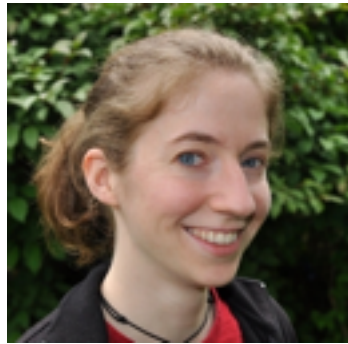
Abstract



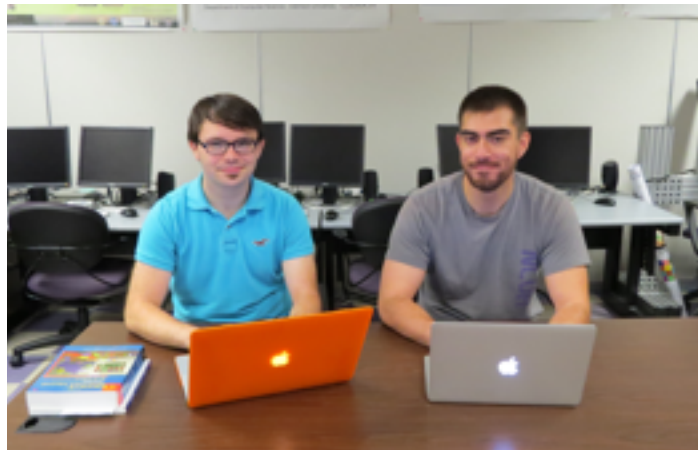
In this talk I will go over two projects we are currently engaged in at Clemson. The first deals with parallelization of gaze analytics where computation is conducted on the Palmetto High Performance Computing cluster. The second deals with eye movement simulation. The goal of the latter is to eventually generate gaze data ground truth with which to validate different filtering approaches.

W tym wykładzie przedstawię dwa projekty nad którymi pracuje w Clemson. Pierwszy dotyczy równoczesnej analizie okulograficznych danych gdzie obliczenia są robione na klasterze Palmetto. Drugi dotyczy symulacji ruchu oka gdzie cel jest wyprodukowanie danych jako podstawa na uzasadnienie lub weryfikację różnych filtrów cyfrowych.

Acknowledgements



- Co-PIs and Graduate Students
 - Don H. House
 - Sophie Jörg
 - Takumi Bolte, Darrell Best, et al.



- Funding
 - Partially supported by NSF



- Donations
 - NVidia's equipment donations



- SWPS invitation

Acknowledgements



- Collaborators:
 - Krzysztof Krejtz
 - Information Processing Inst., Warsaw, Poland
 - Iza Krejtz
 - University of Social Sciences and Humanities, Warsaw, Poland
 - Agnieszka Szarkowska
 - University of Warsaw, Poland
 - Radosław Mantiuk, Bartosz Bazyluk
 - West Pomeranian Univ of Technology, Szczecin, Poland



Changing technology



Arrington



ASL



Dikablis



EyeLink



Positive
Science



FaceLab



mirametrix



SMI



Tobii

Increasing affordability



Pupil Labs
(€380-800)



The Eye Tribe
(\$99)

Personal installation



 **THEEYETRIBE**



CLEMSON UNIVERSITY
EYE TRACKING LAB
SCHOOL OF COMPUTING

CLEMSON
UNIVERSITY

Classroom installation



CLEMSON UNIVERSITY
EYE TRACKING LAB
SCHOOL OF COMPUTING

CLEMSON[®]
UNIVERSITY

Classroom installation



CLEMSON UNIVERSITY
EYE TRACKING LAB
SCHOOL OF COMPUTING

CLEMSON[®]
UNIVERSITY



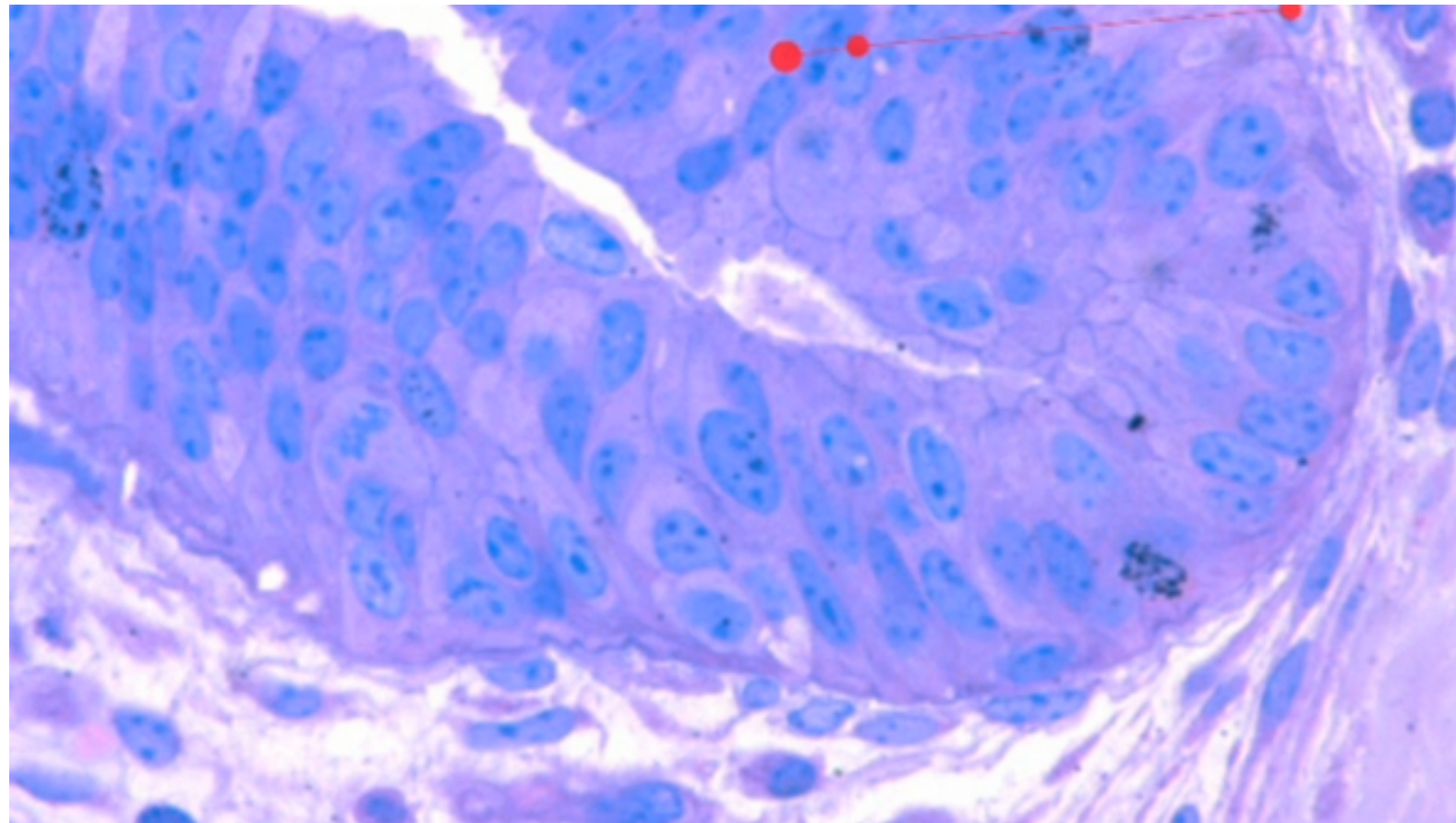
Niedawne Projekty



CLEMSON UNIVERSITY
EYE TRACKING LAB
SCHOOL OF COMPUTING

CLEMSON[®]
UNIVERSITY

- Gaze-Augmented Think-Aloud (Vitak et al., 2012):
- Tested effectiveness of expert's gaze atop video
 - scanpath needs to demonstrate **expert's search strategy**

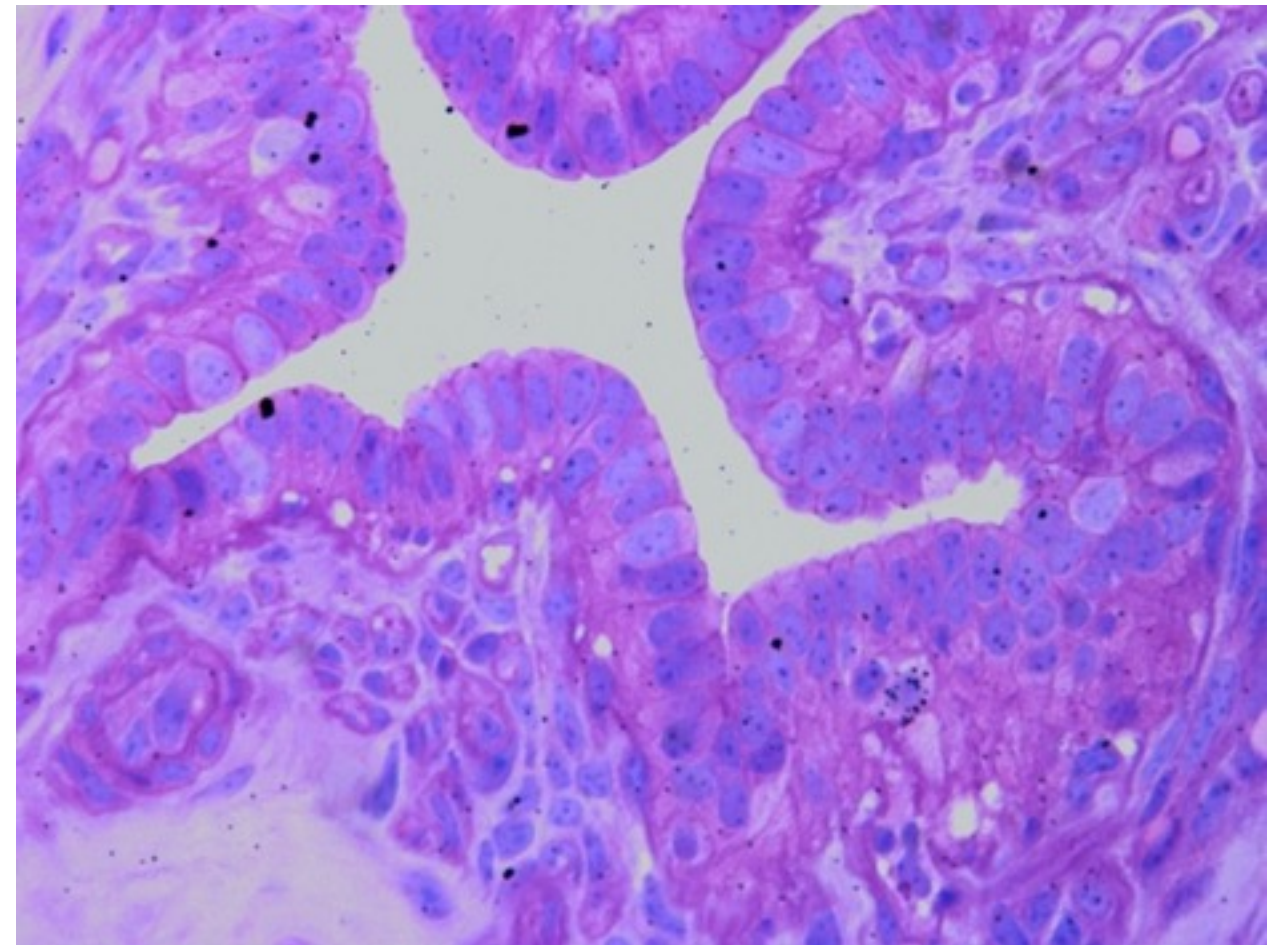
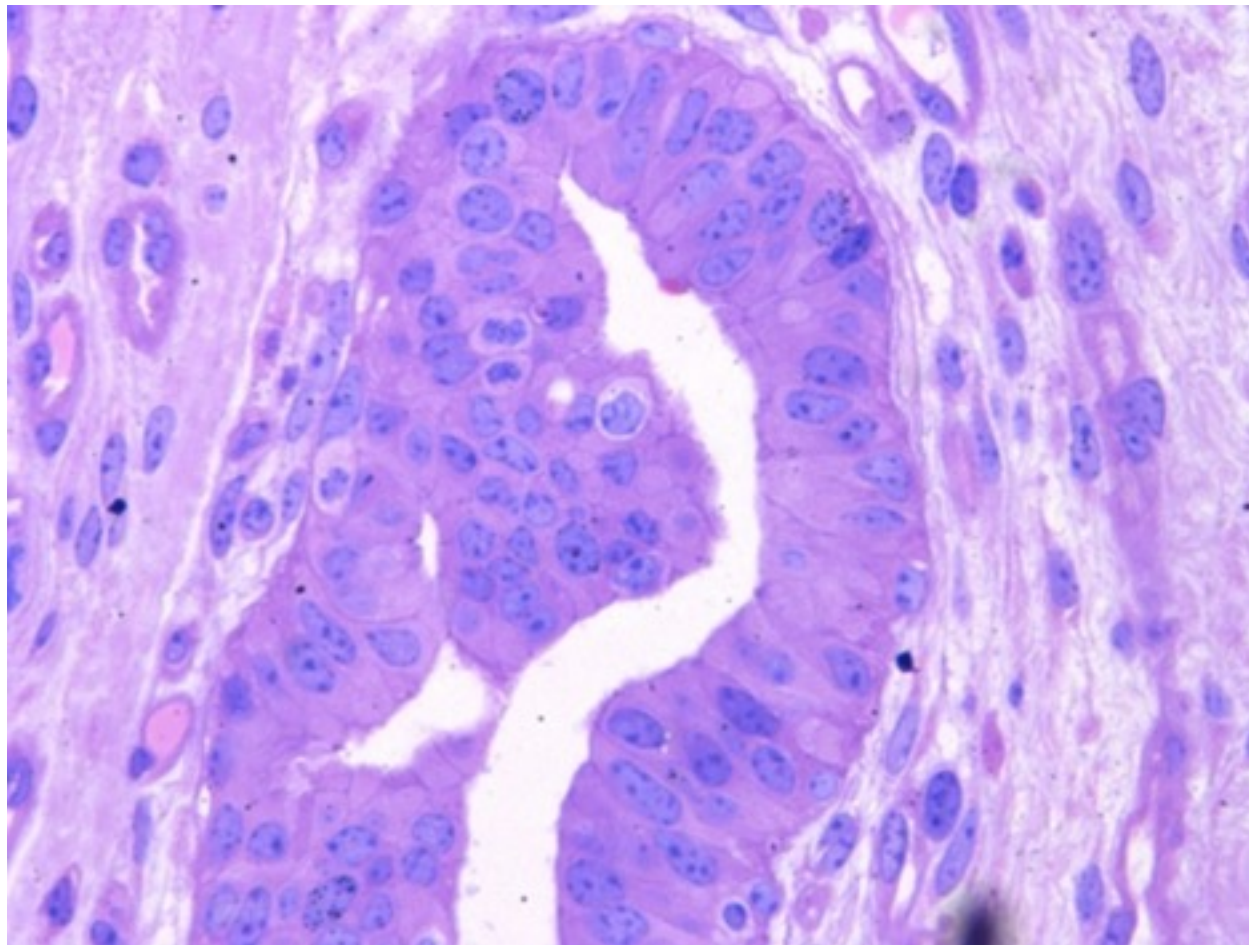


Example stimuli & task: identify immuno-gold (BrdU) stained cross-sections of bovine mammary tissue.

GATA as an aid to learning



- Vitak et al. (2012) developed histological training:
 - **S**taining, **T**issue types, **A**rtifacts, **M**agnification, **P**lane of section



Example stimuli & task: identify immuno-gold (BrdU) stained cross-sections of bovine mammary tissue.

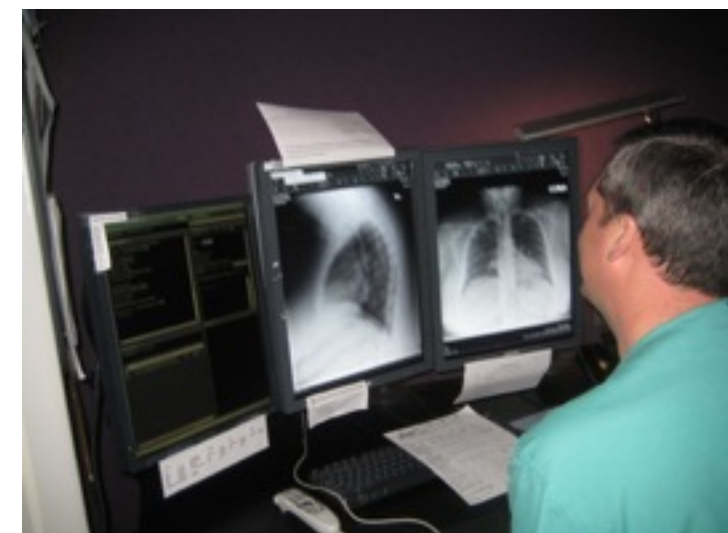
GATA as an aid to learning



- Results on accuracy:
 - no significant effect of GATA on true positive rate (hits)
($F(1,28) = 1.21, p = 0.28, n.s.$)
 - BUT...
 - significant effect of GATA on false positive rate (error)
($F(1,27) = 10.15, p < 0.01$)
 - GATA helped users reduce making mistakes**

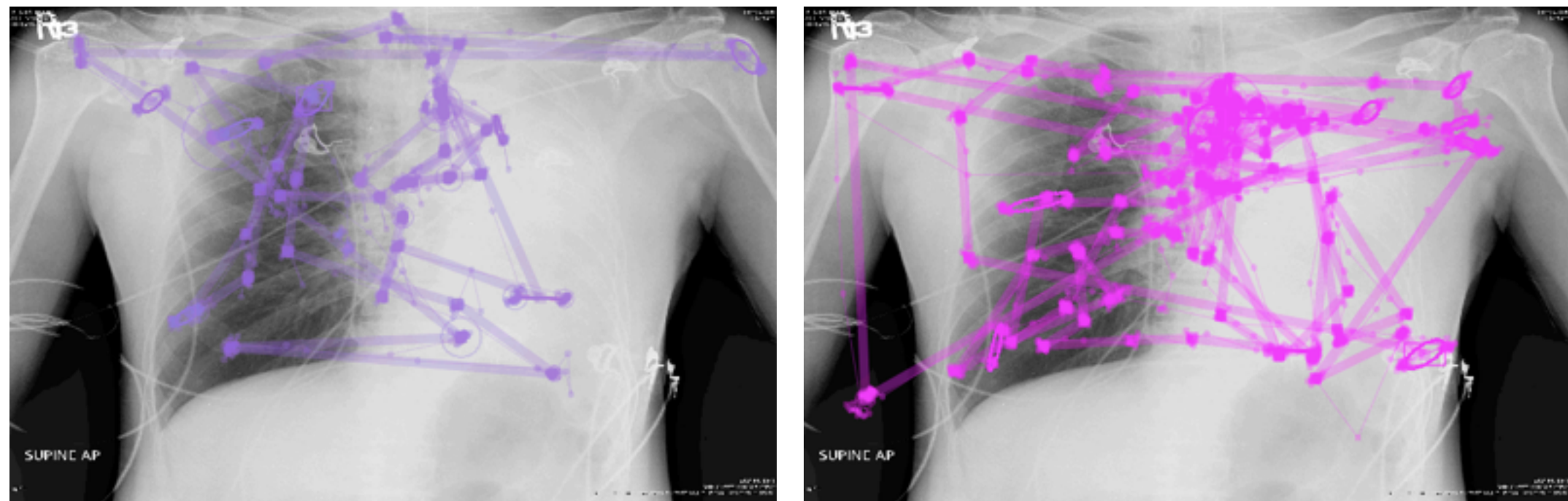


- Gaze-Augmented Think-Aloud (Vitak et al., 2012):
 - simpler than more elaborate visual codifications
 - e.g., Sadasivan et al. (2005)
 - scanpath indicates what to look for **and what to avoid**
 - accompanying verbalization explains visual guidance
 - initial ethnographic data collected from radiologists



Current training at local hospital: from left to right, **intern**, **resident**, **radiologist**.

- Visual search:
 - well-defined strategies have been developed by experts
 - e.g., top-down cognitive strategies based on experience
 - Chest X-Ray (CXR) inspection:
 - **A**irway, **B**ones, **C**ardiac silhouette, **D**iaphragms, **E**xternal tissue, ...



Expert (left) and novice (right) scanpaths over abnormal CXR.

SAP 2014



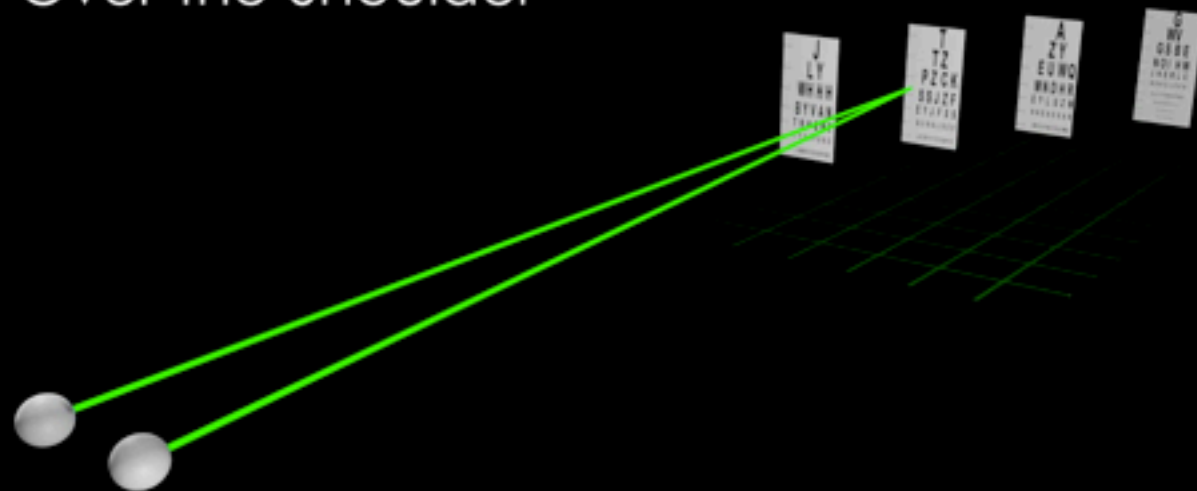
Left Eye



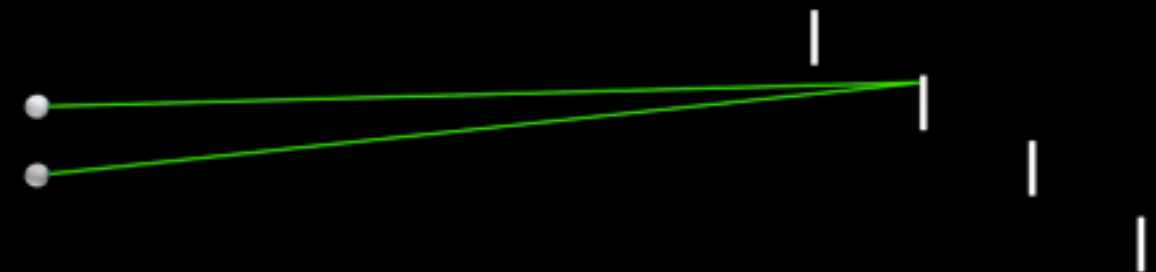
Right Eye

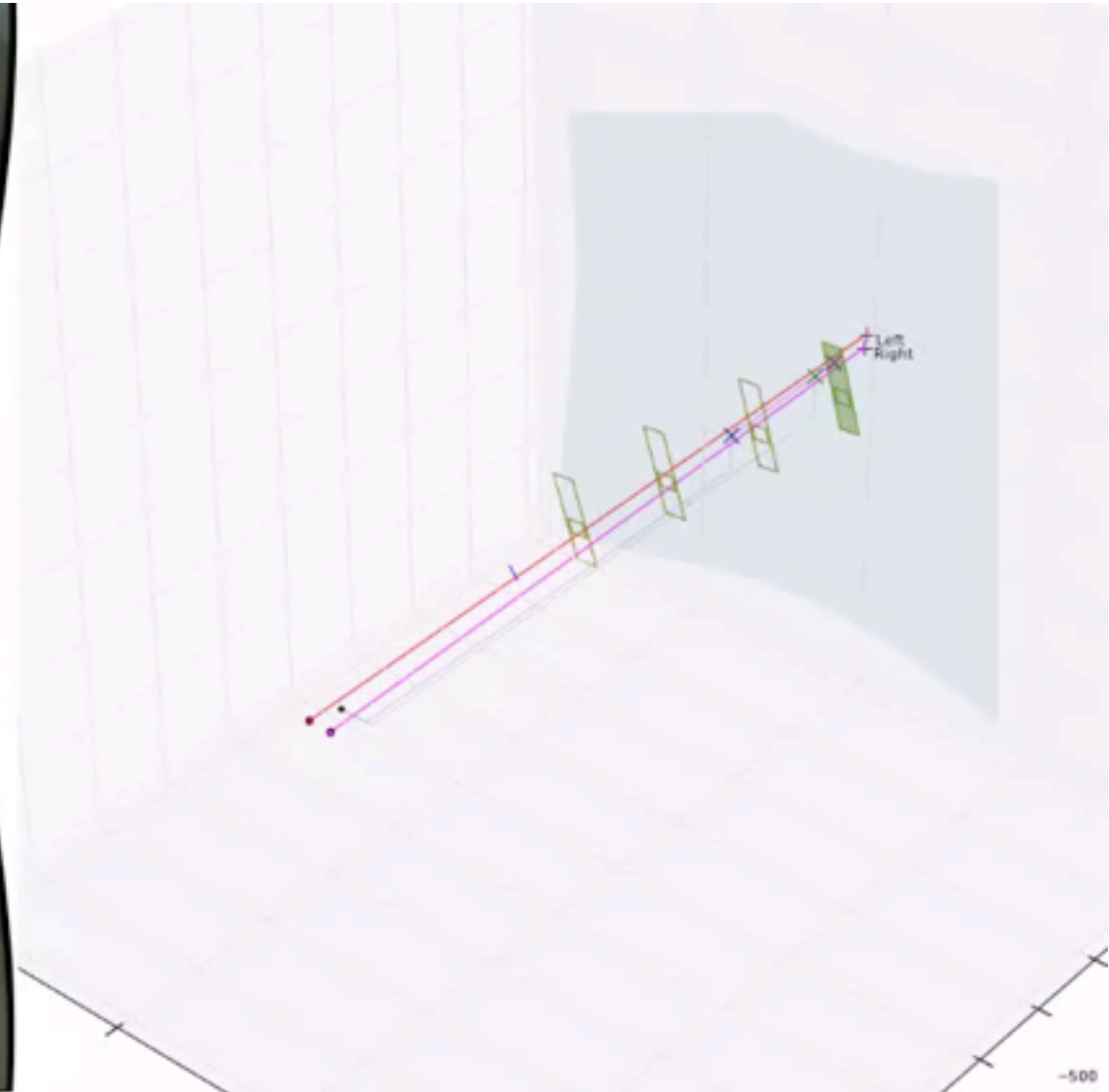
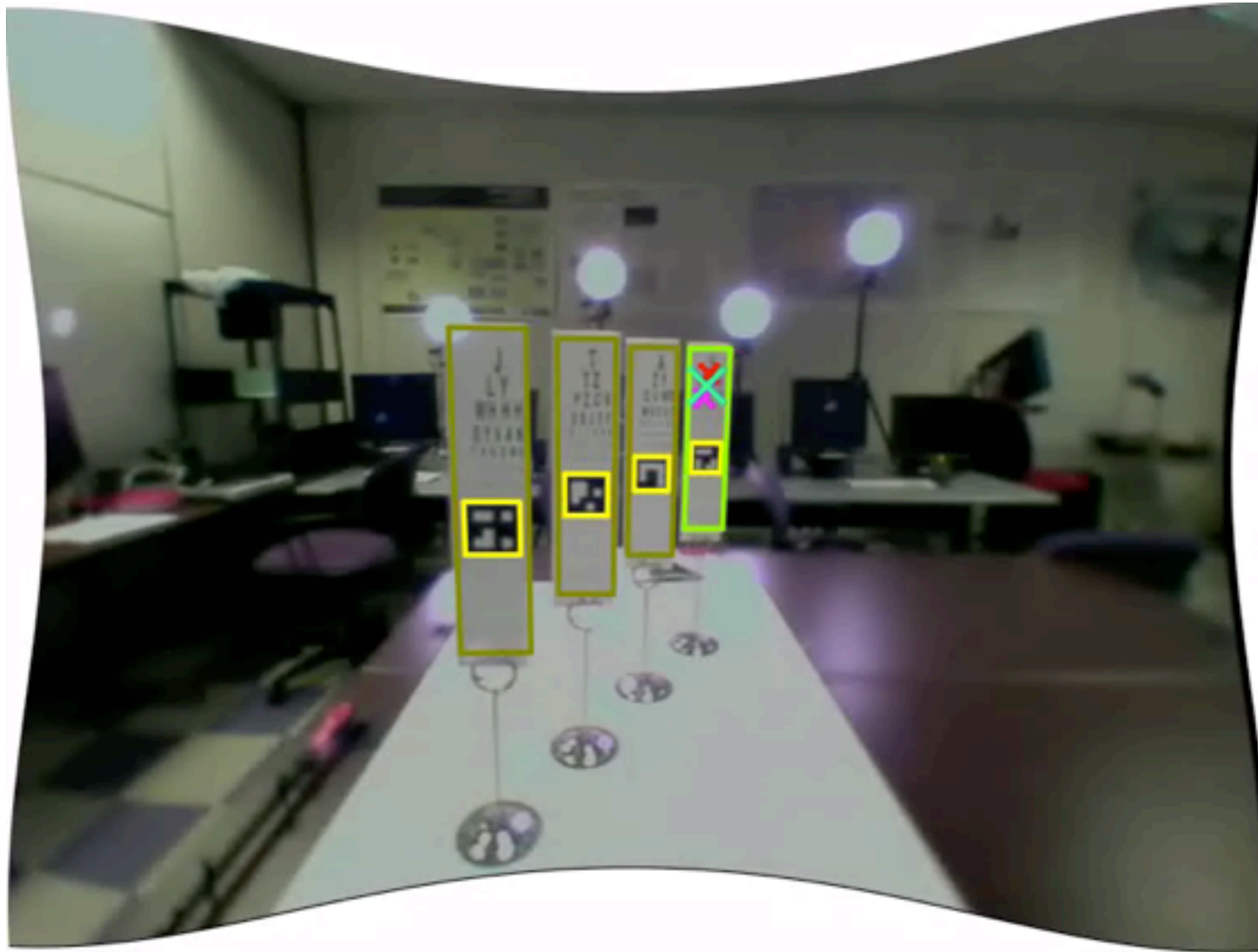


Over-the-Shoulder



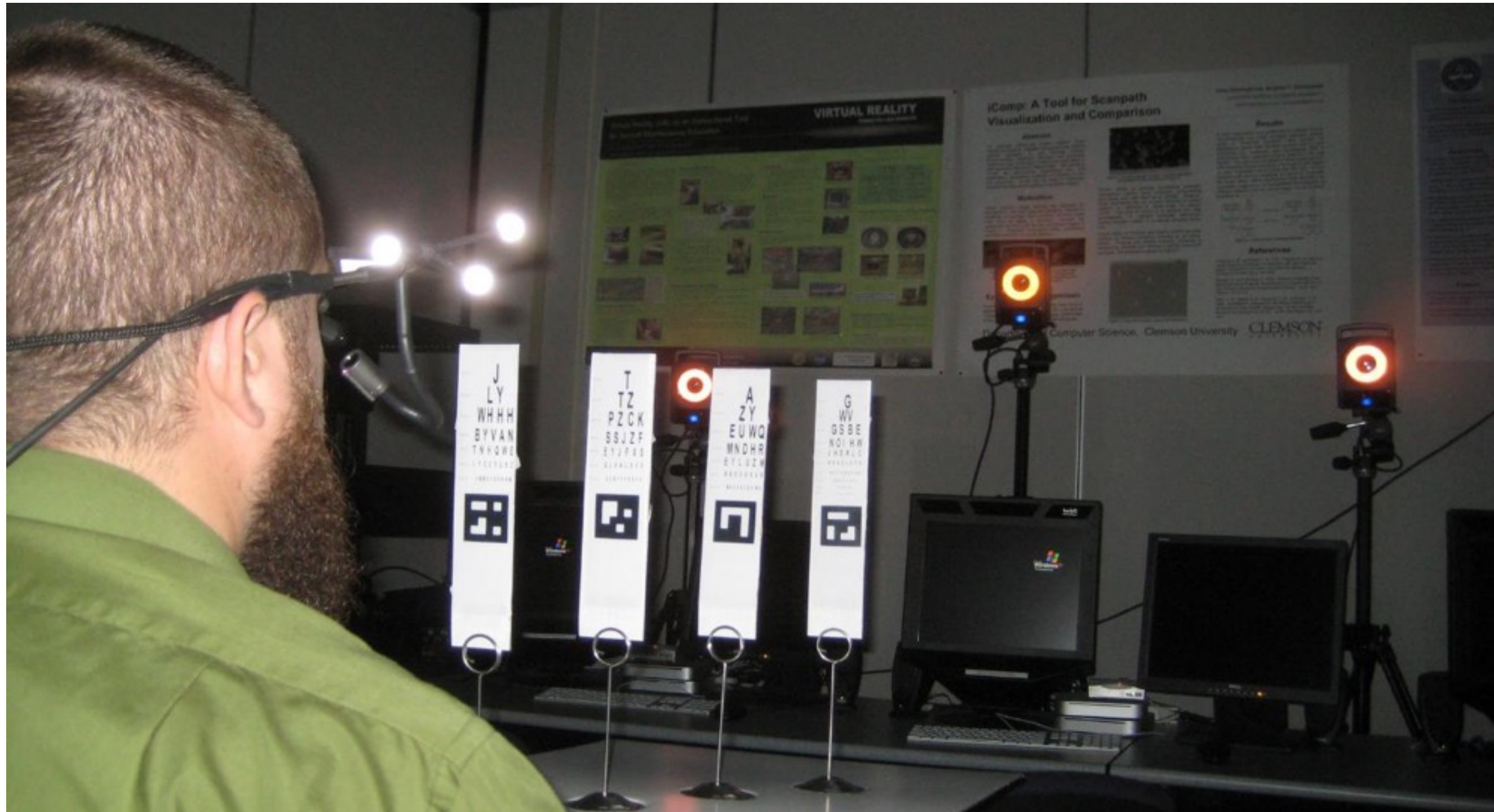
Top Orthographic



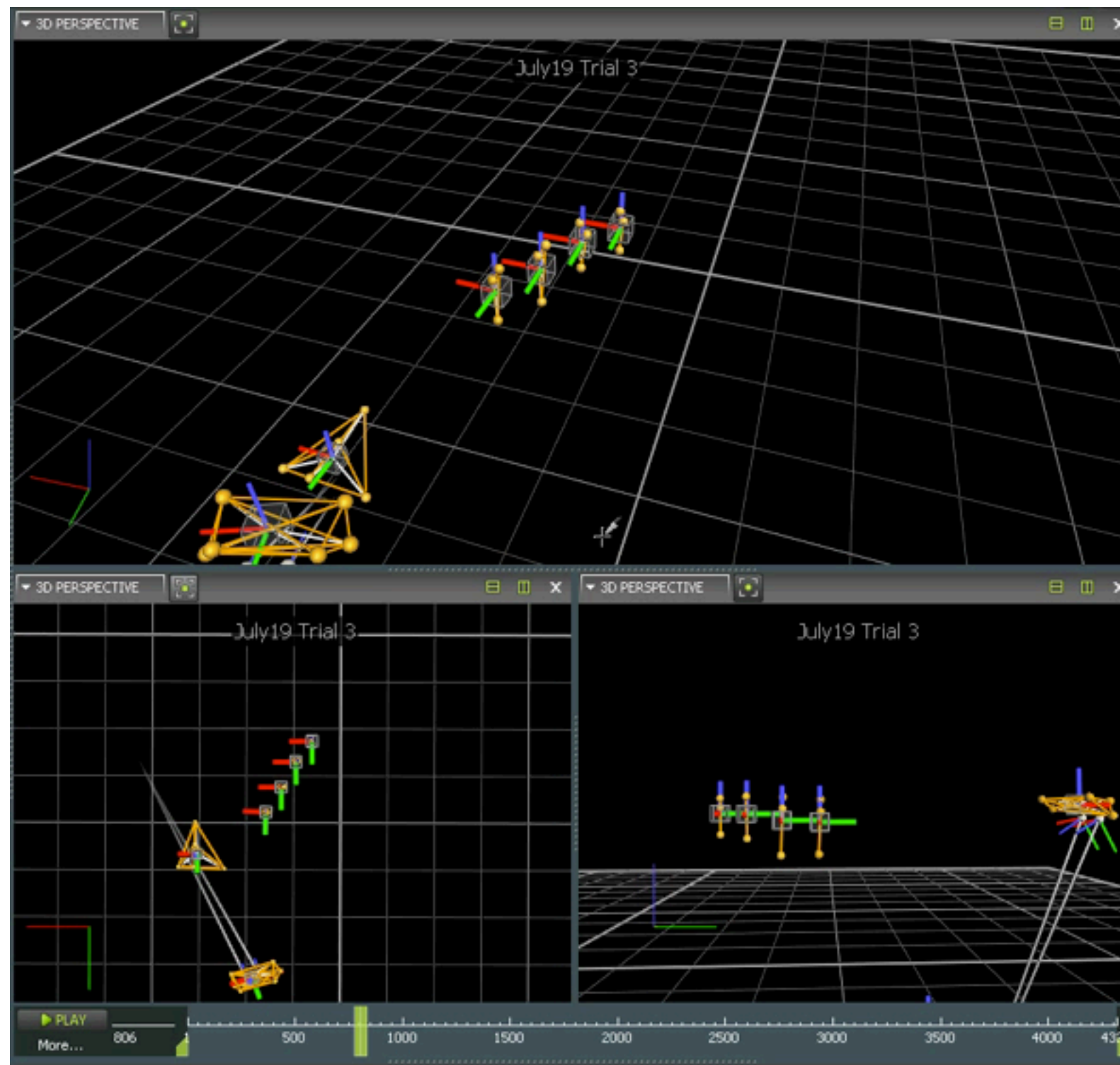


Video source: Duchowski et al., 2014, video courtesy of Lech Świrski, University of Cambridge.

Working with Vicon



Working with Vicon





Bieżąca Praca



CLEMSON UNIVERSITY
EYE TRACKING LAB
SCHOOL OF COMPUTING

CLEMSON[®]
UNIVERSITY



Massive-Scale Gaze Analytics



CLEMSON UNIVERSITY
EYE TRACKING LAB
SCHOOL OF COMPUTING

CLEMSON[®]
UNIVERSITY

Massive-scale gaze analytics

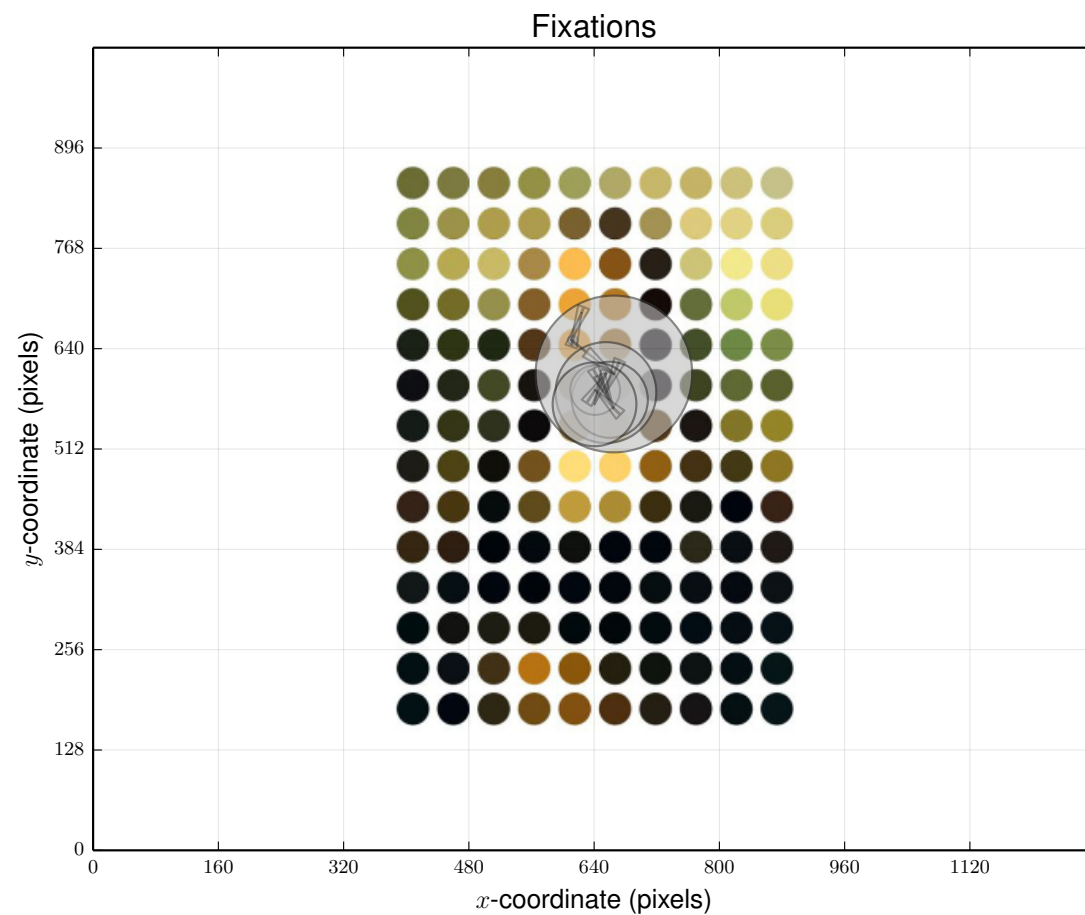


- Eye tracking data is voluminous
- Consider the following experiment stimulus:

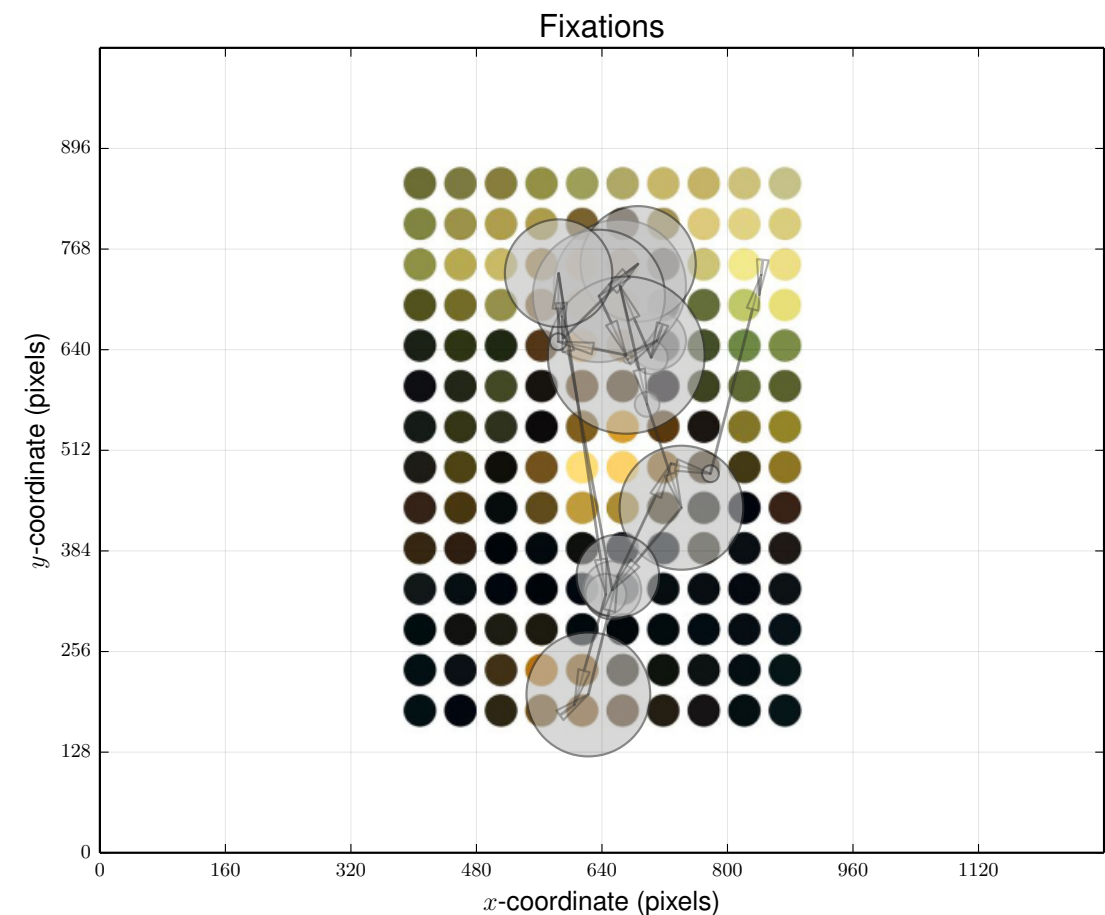


- Hypothesis:
 - gaze of those who recognize *puntos1* differs from those who do not

Fixations



VIEWER WHO RECOGNIZED IMAGE



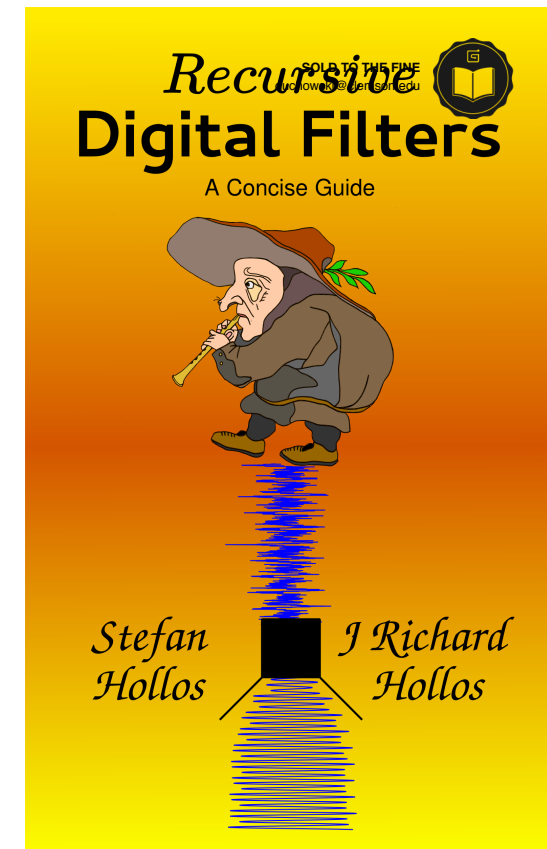
VIEWER WHO DID NOT RECOGNIZE

- Goal is to:
 - detect fixations (via filtering)
 - compare fixations (via a quantifiable metric)

Raw gaze data filtering



- Velocity-based filters:
 - differentiate position signal, yielding velocity
 - velocity threshold used to detect saccades
 - no ring buffer needed, just convolution
- Infinite Impulse Response (IIR) filter:
 - Finite Impulse Response (FIR) - “memory”

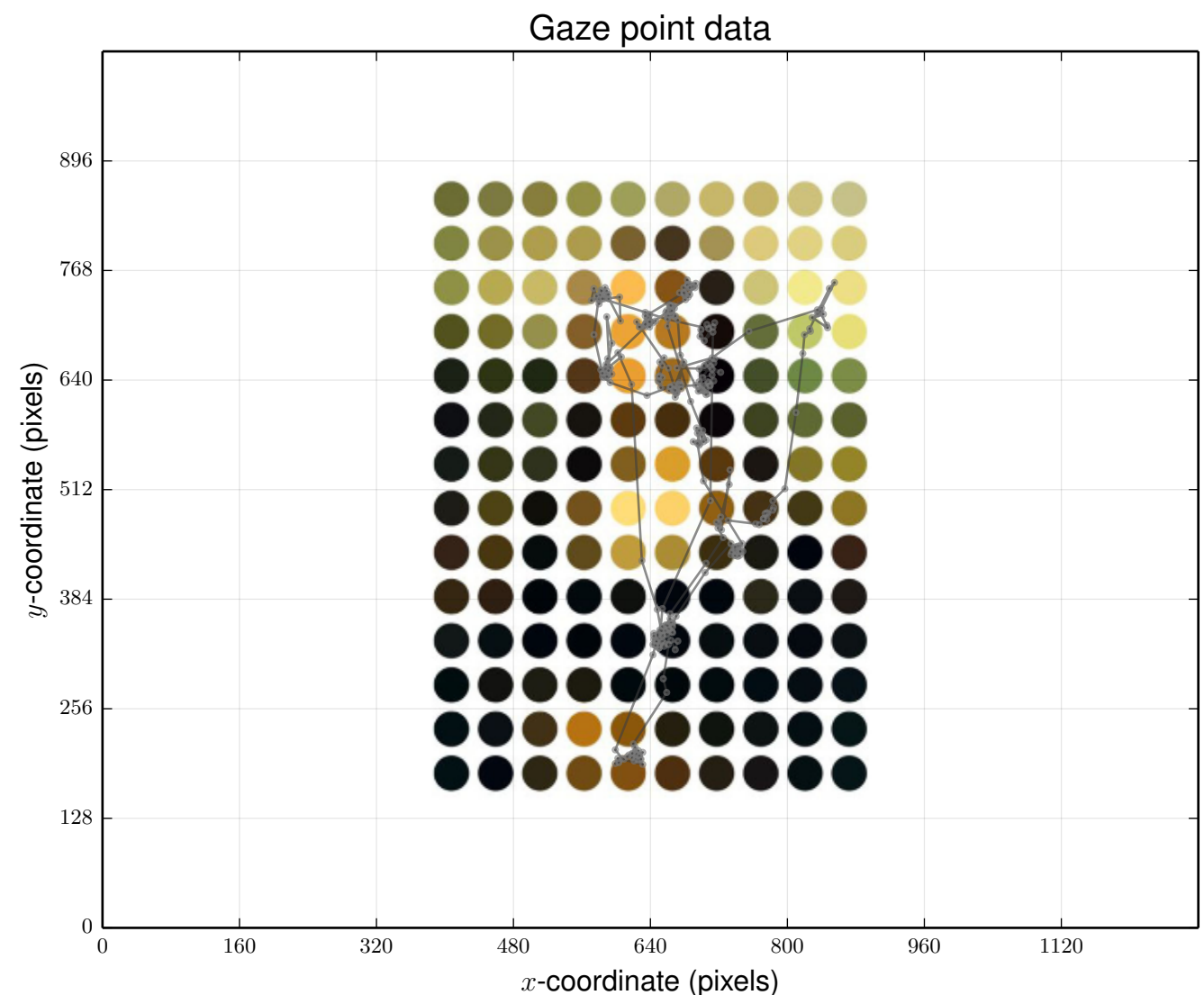


$$\dot{x}_n^s(t) = 1/(\Delta t^s) \left(\sum_{i=-p}^p h_i^{t,s} x_{n-i} - \sum_{i=-q}^q g_i^{t,s} \dot{x}_{n-i} \right)$$

Signal processing



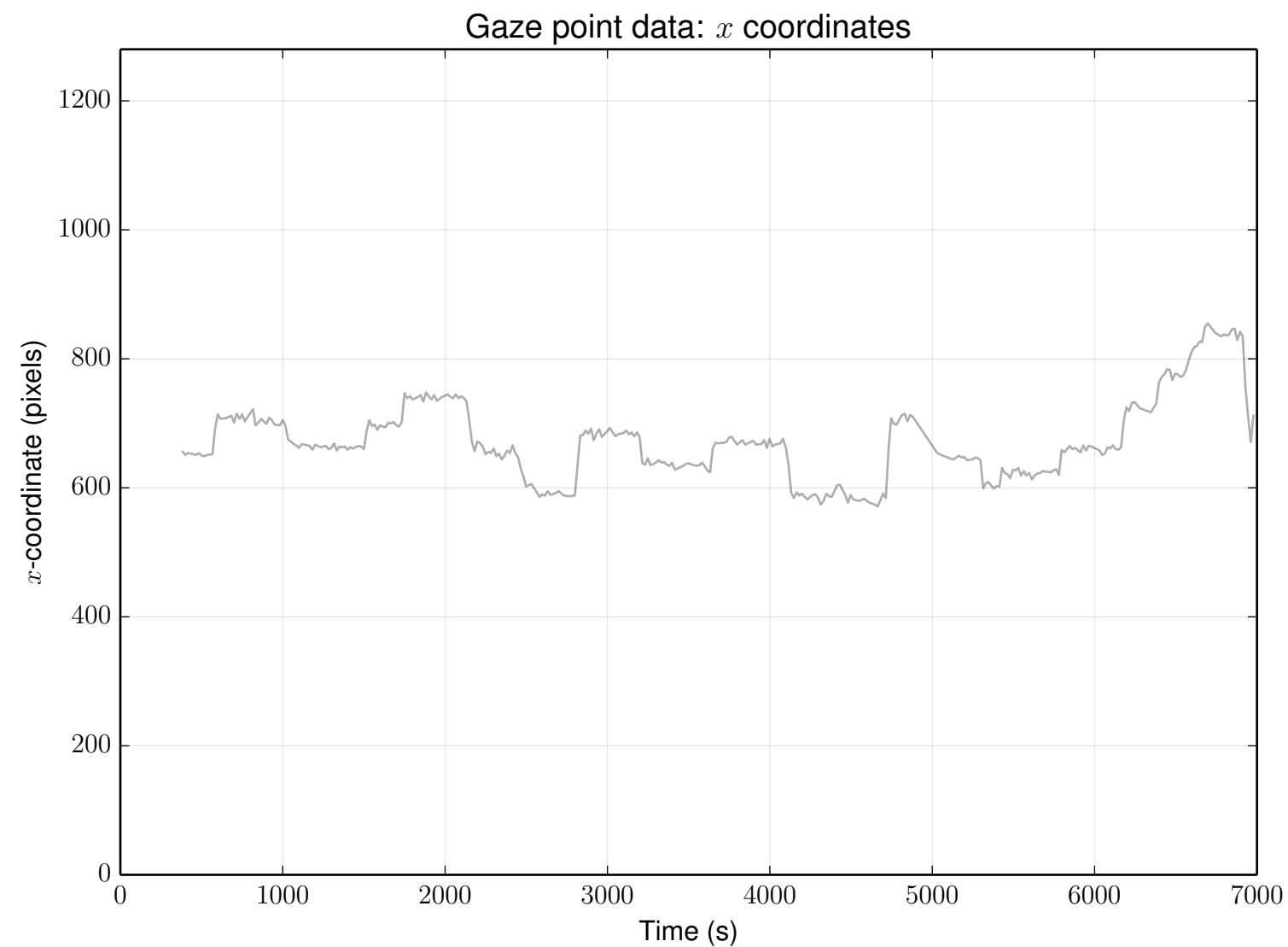
- To detect fixations, need to filter data, via:
 - position-variance (dispersion-based) filter, or
 - velocity-based filter
- Python “pipeline”
 - for more info, see:
Duchowski et al., 2014
(ET4S)
- Pipeline starts with:
 - raw gaze point data



Signal processing



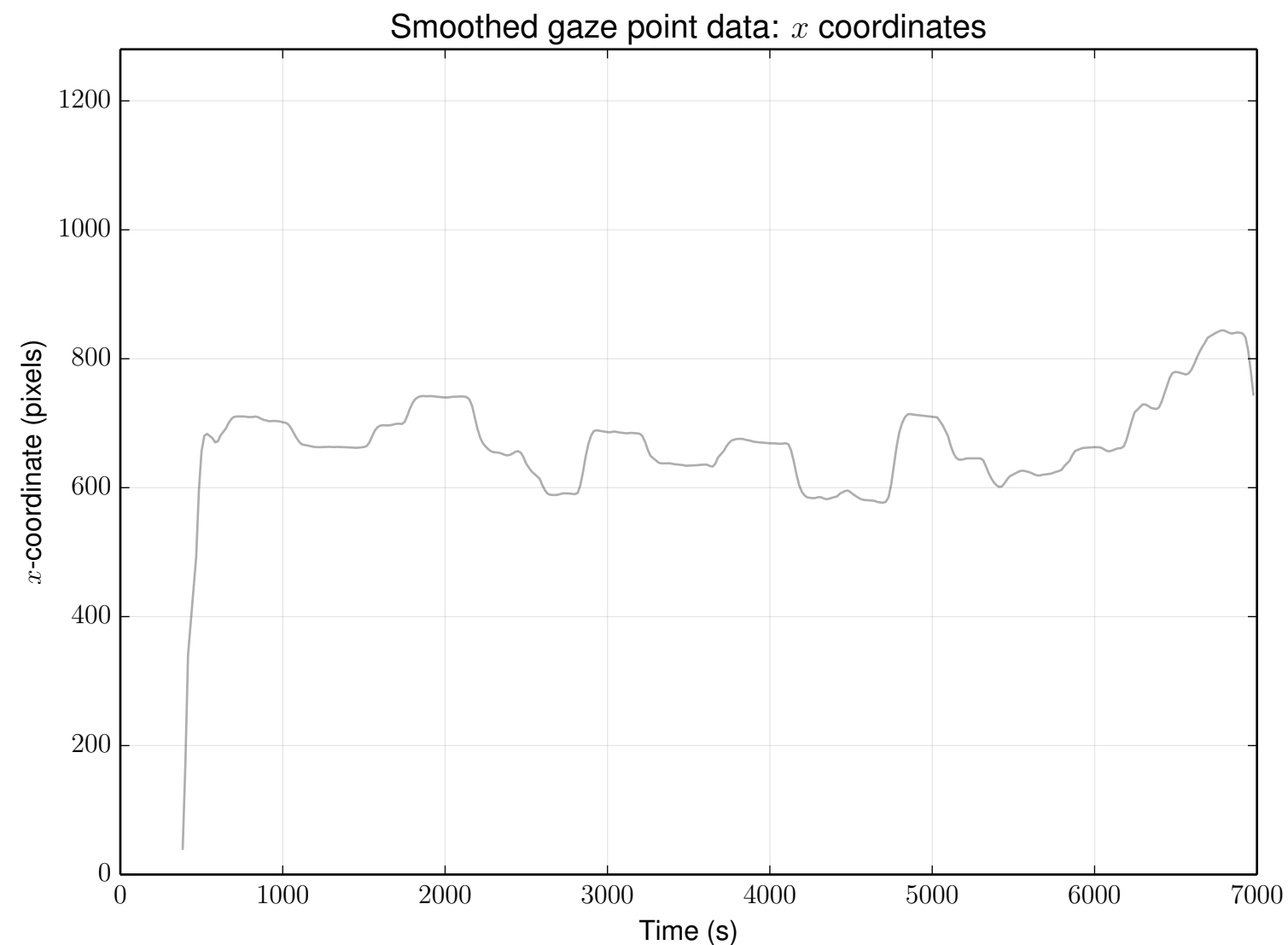
- Proceed with:
 - raw gaze point data
 - x -coordinate vs. time



Signal processing



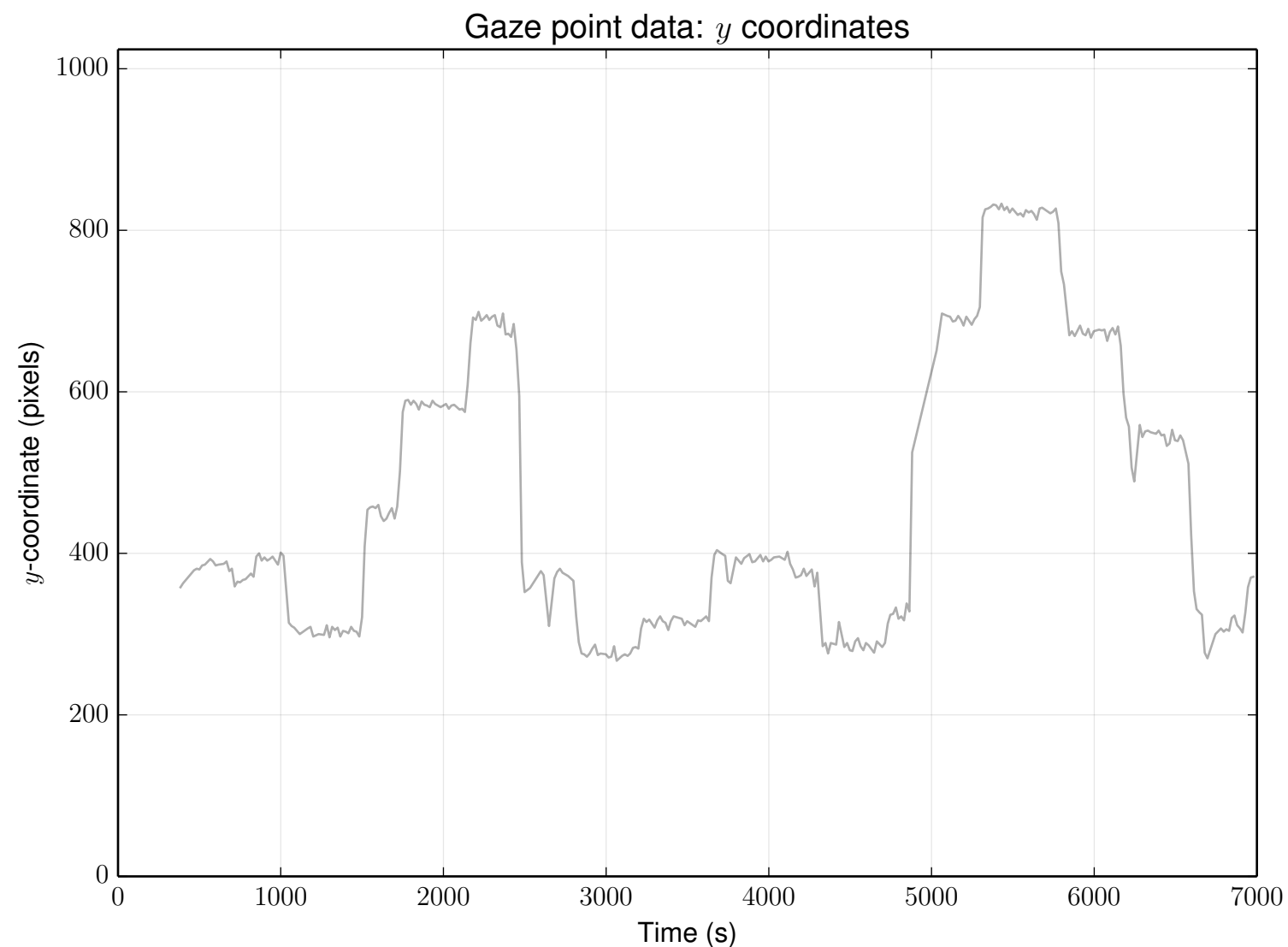
- Proceed with:
 - raw gaze point data
 - x -coordinate vs. time
 - smoothed x -coord



Signal processing



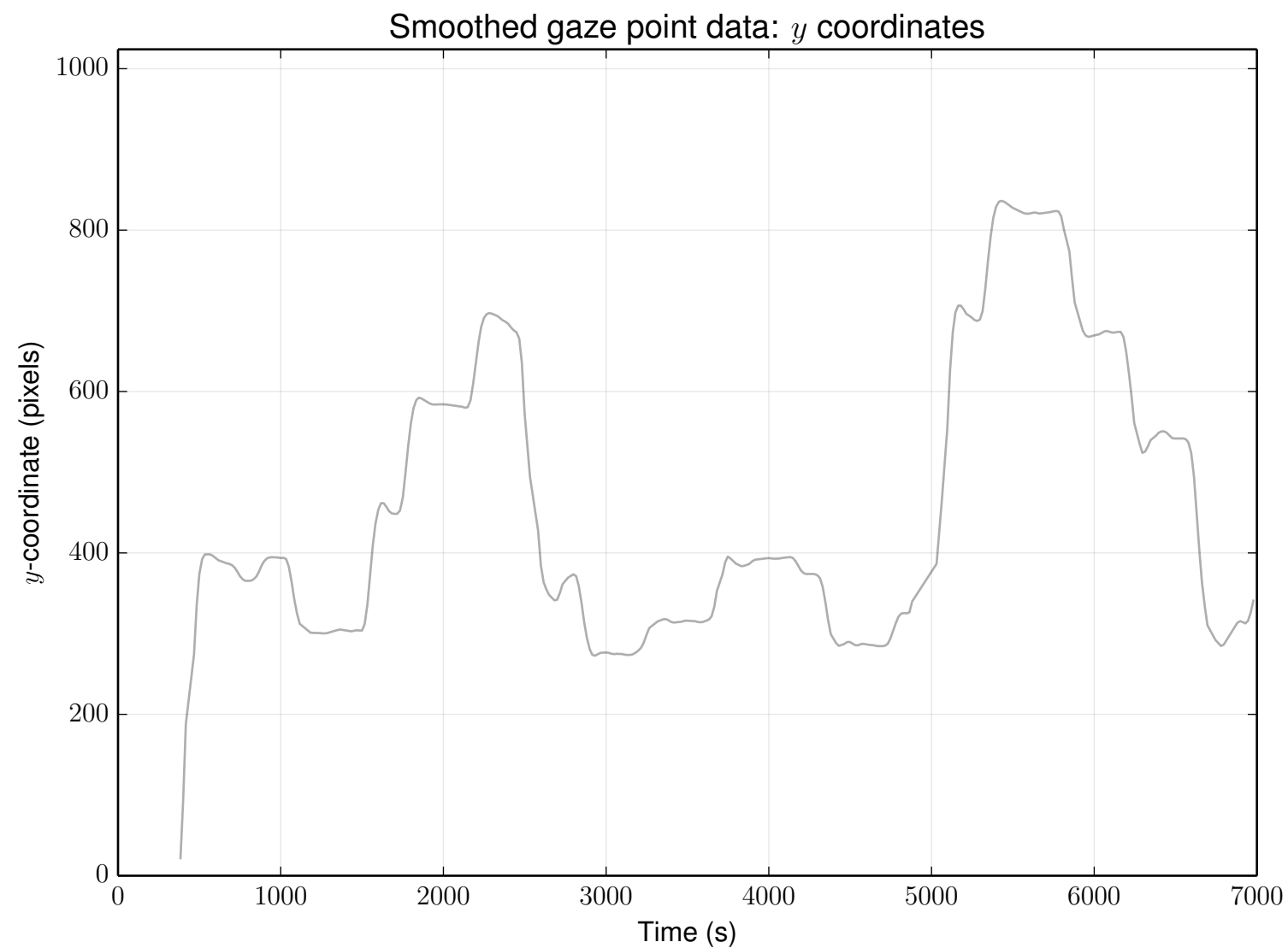
- Proceed with:
 - raw gaze point data
 - x -coordinate vs. time
 - smoothed x -coord
 - y -coordinate vs. time



Signal processing



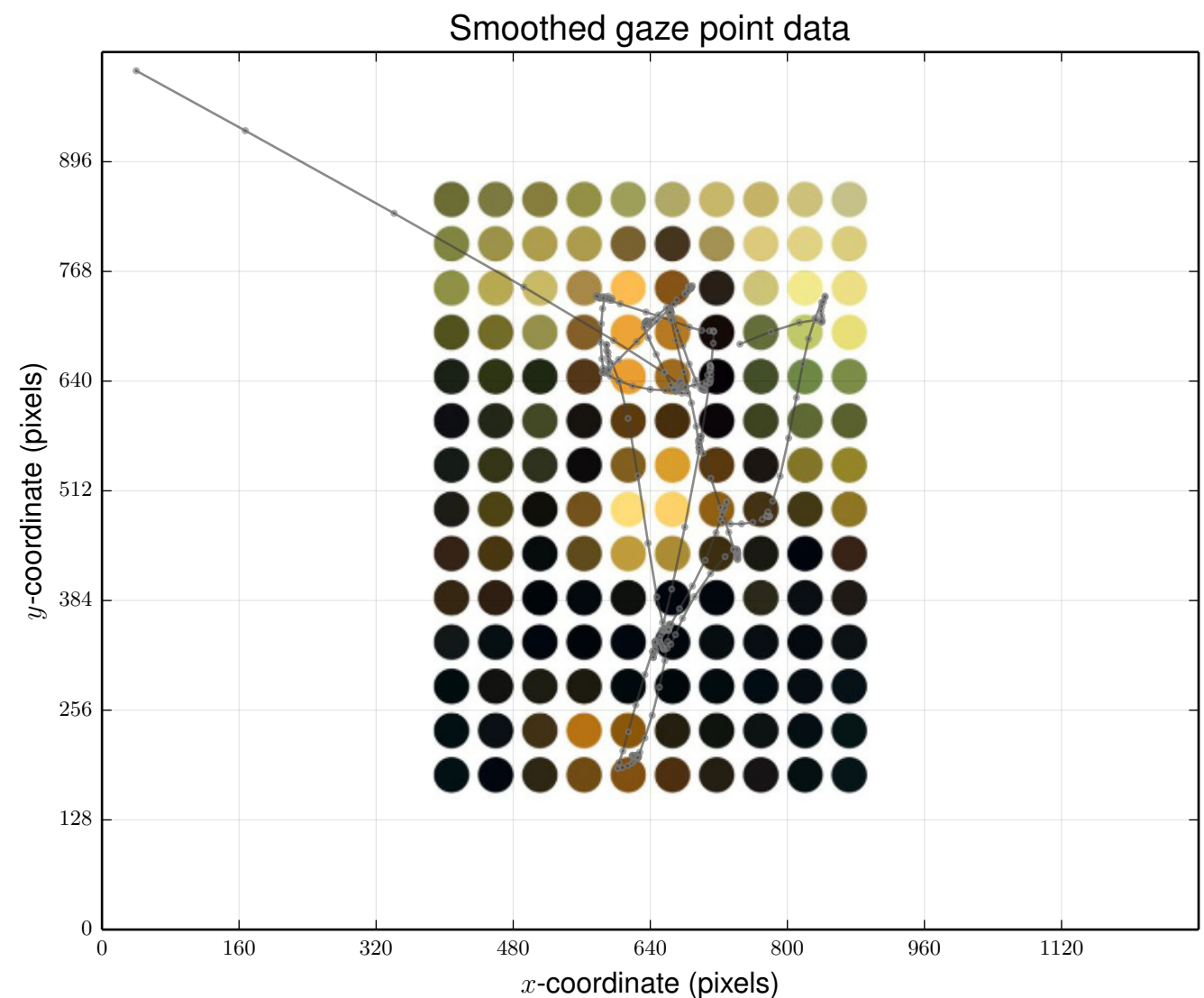
- Proceed with:
 - raw gaze point data
 - x -coordinate vs. time
 - smoothed x -coord
 - y -coordinate vs. time
 - smoothed y -coord



Signal processing



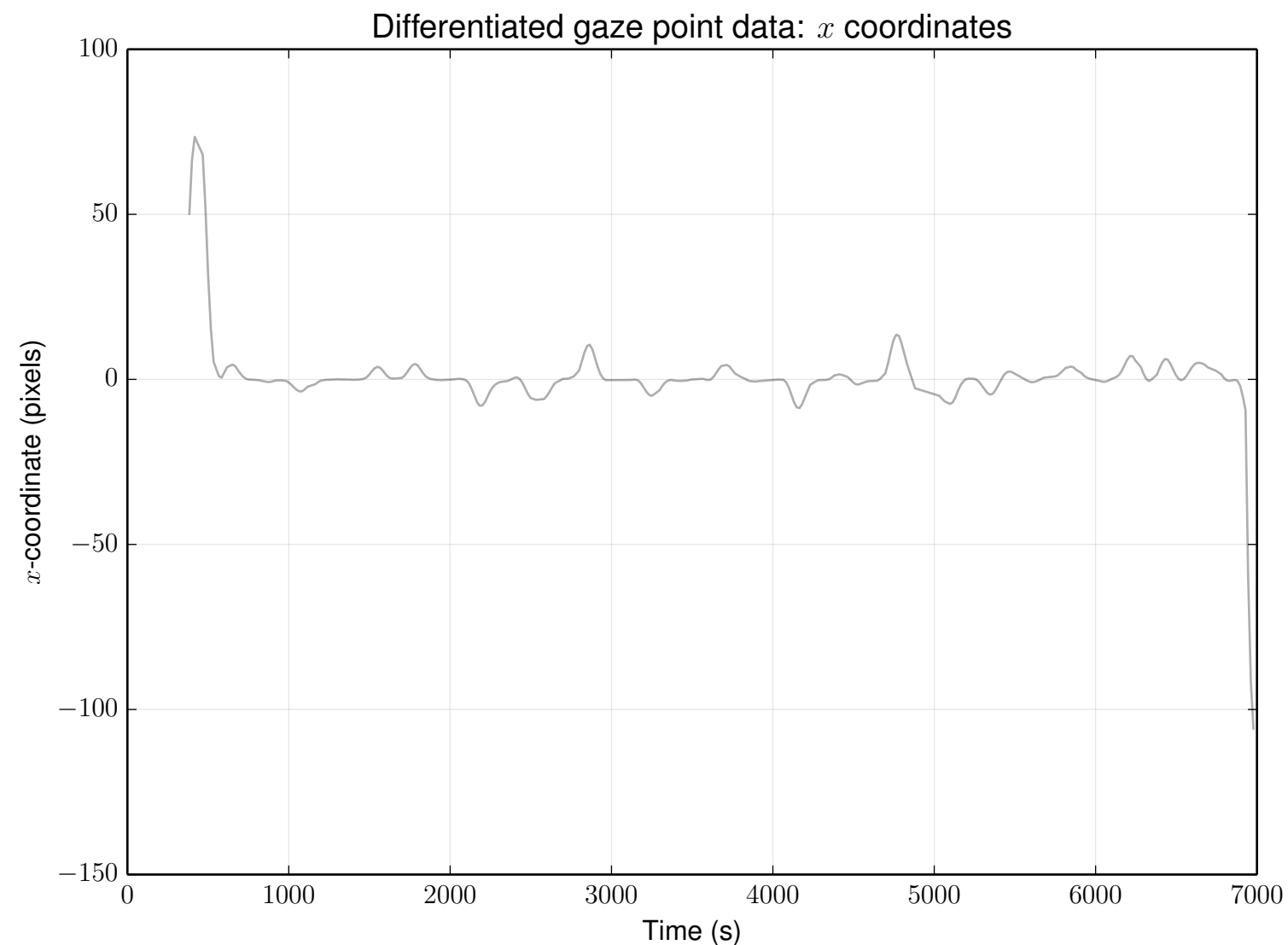
- Proceed with:
 - raw gaze point data
 - x -coordinate vs. time
 - smoothed x -coord
 - y -coordinate vs. time
 - smoothed y -coord
 - smoothed 2D data



Signal processing



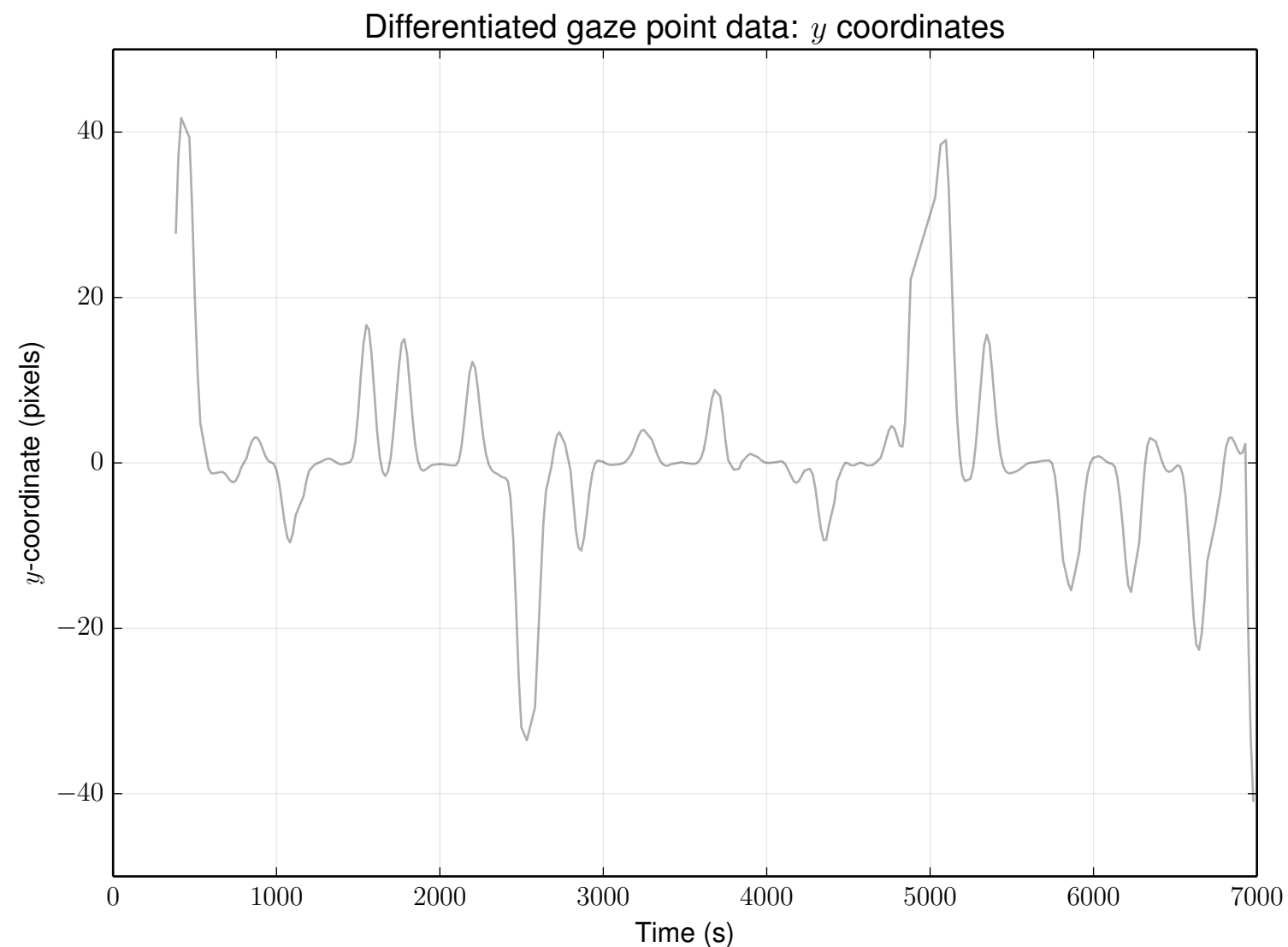
- Proceed with:
 - raw gaze point data
 - x -coordinate vs. time
 - smoothed x -coord
 - y -coordinate vs. time
 - smoothed y -coord
 - smoothed 2D data
 - differentiated x -coord



Signal processing



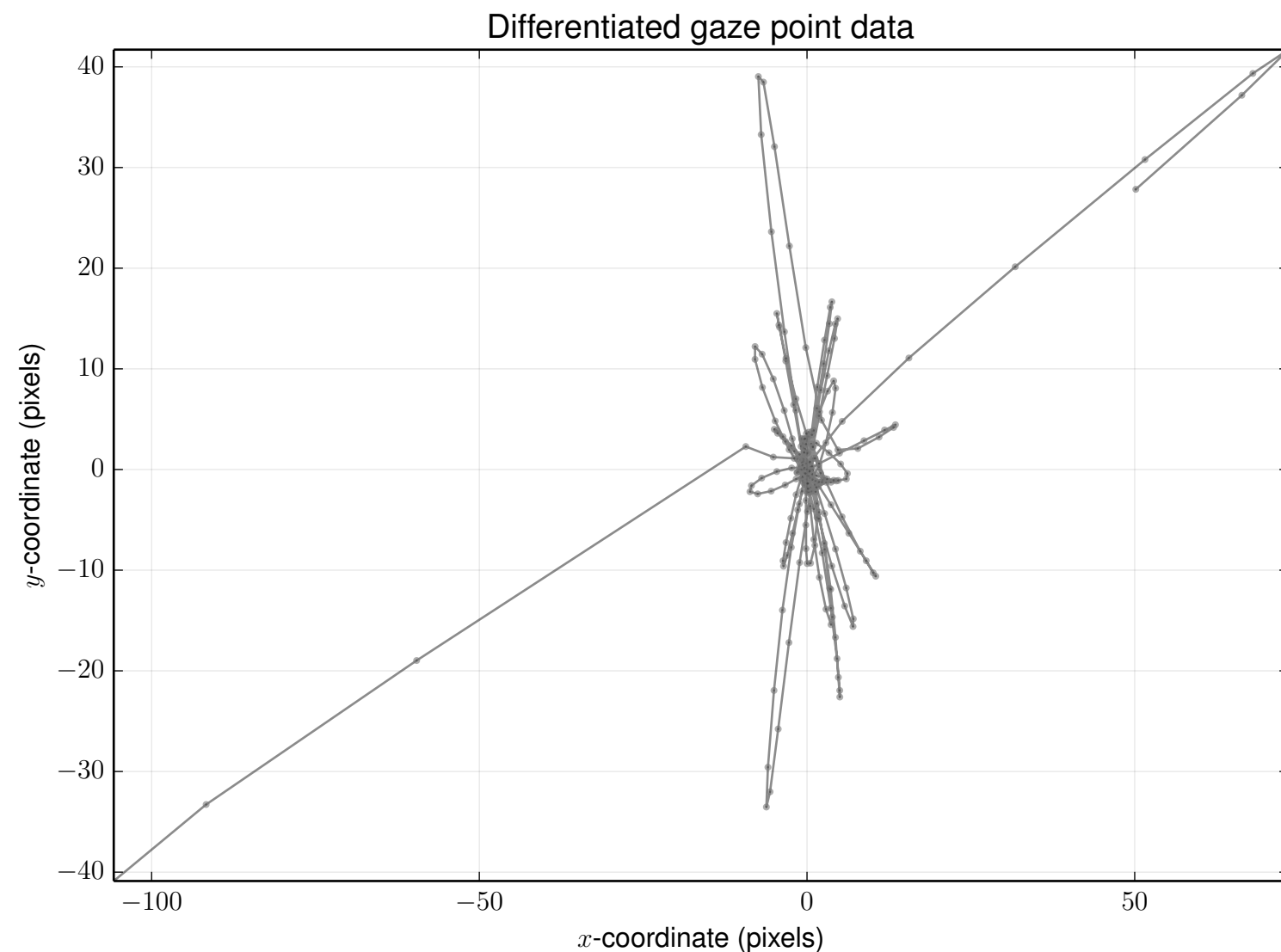
- Proceed with:
 - raw gaze point data
 - x -coordinate vs. time
 - smoothed x -coord
 - y -coordinate vs. time
 - smoothed y -coord
 - smoothed 2D data
 - differentiated x -coord
 - differentiated y -coord



Signal processing



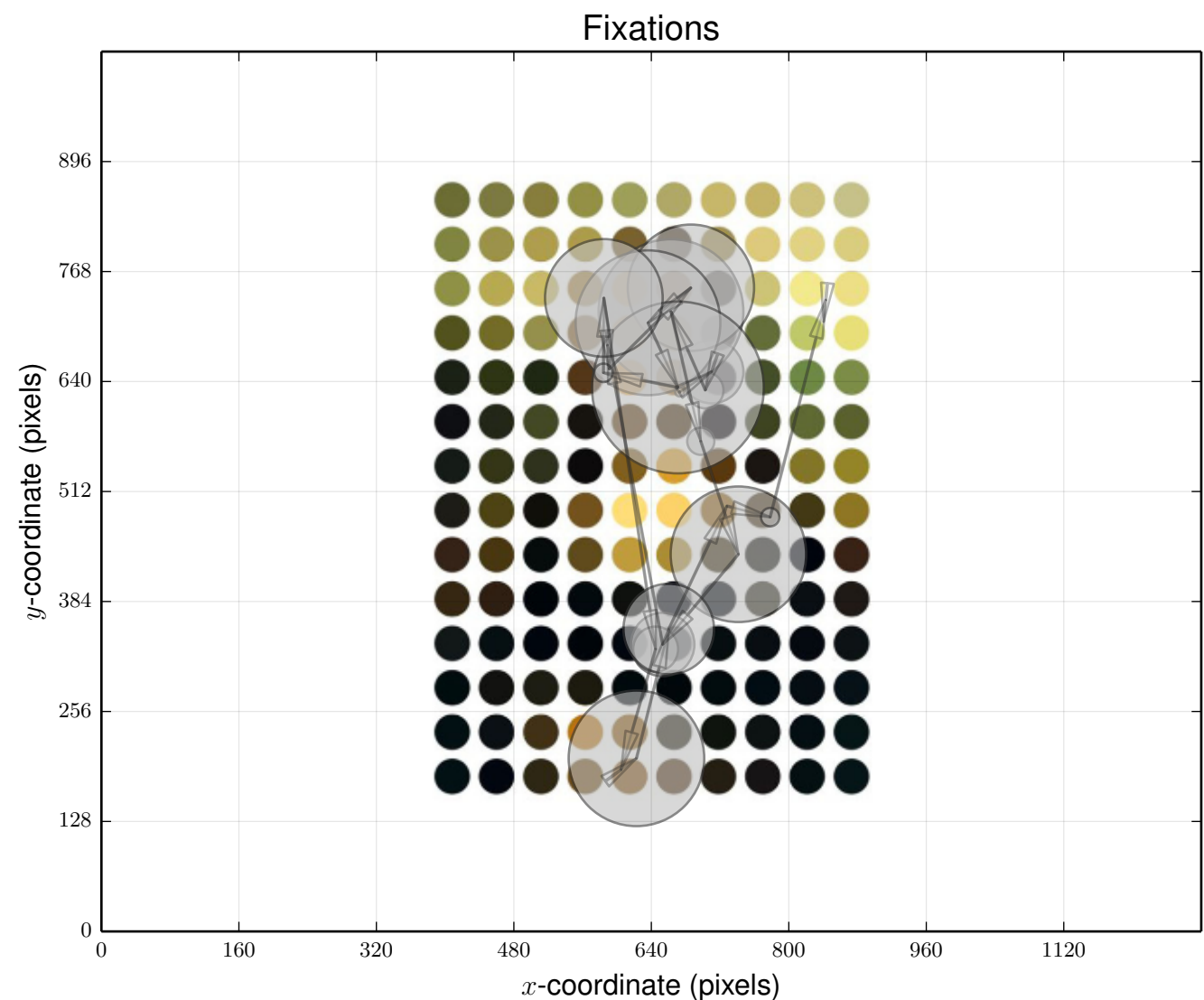
- Proceed with:
 - raw gaze point data
 - x -coordinate vs. time
 - smoothed x -coord
 - y -coordinate vs. time
 - smoothed y -coord
 - smoothed 2D data
 - differentiated x -coord
 - differentiated y -coord
 - differentiated 2D data



Signal processing



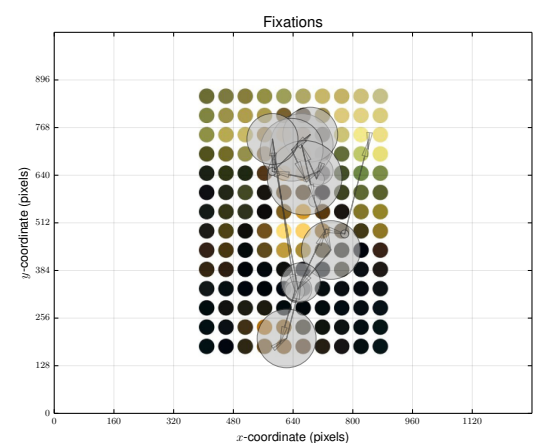
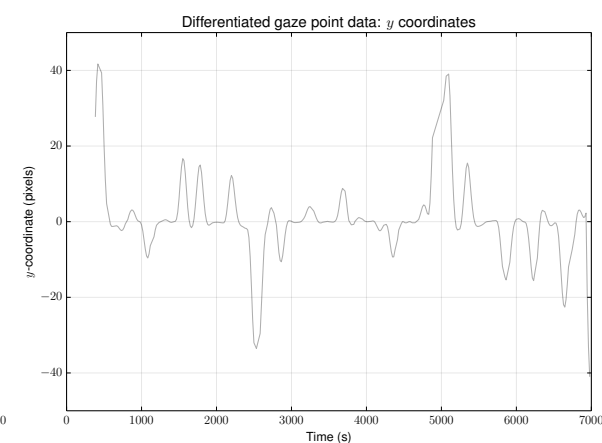
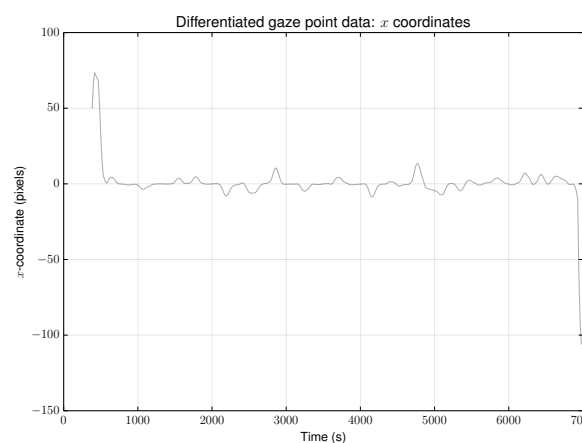
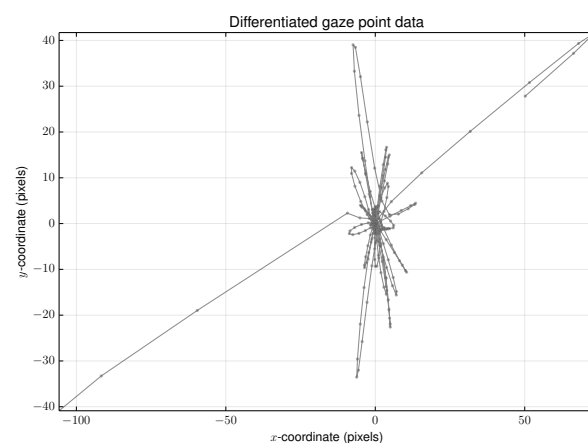
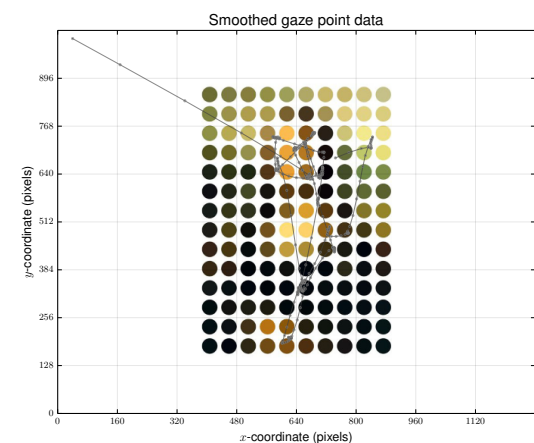
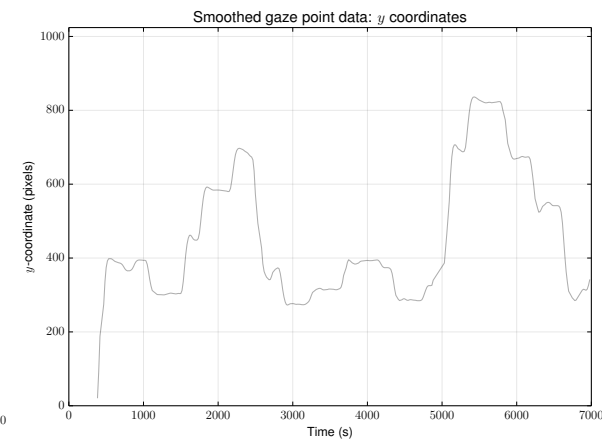
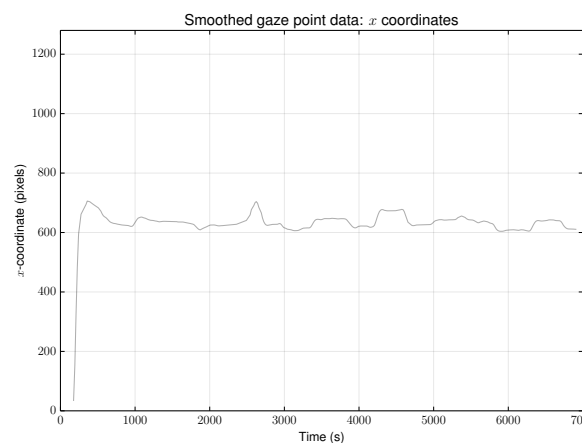
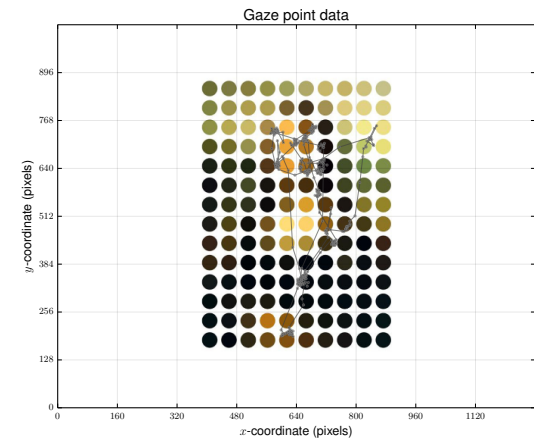
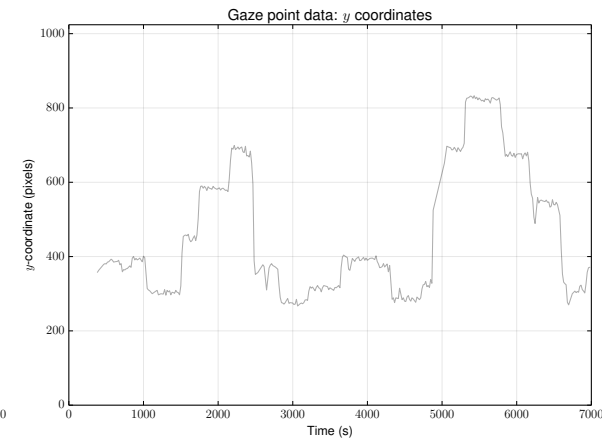
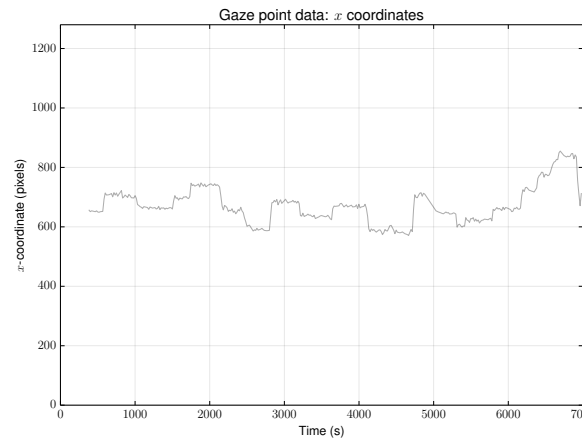
- Proceed with:
 - raw gaze point data
 - x -coordinate vs. time
 - smoothed x -coord
 - y -coordinate vs. time
 - smoothed y -coord
 - smoothed 2D data
 - differentiated x -coord
 - differentiated y -coord
 - differentiated 2D data
 - 2D fixations



Plotting



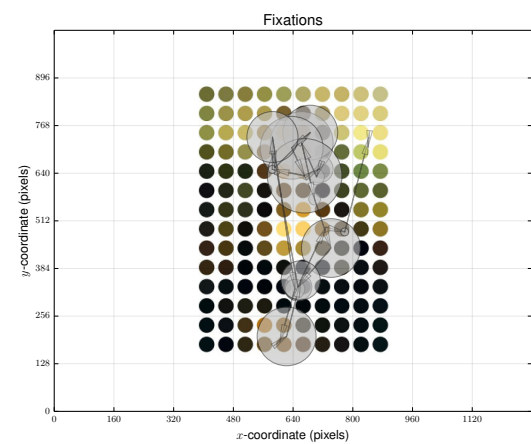
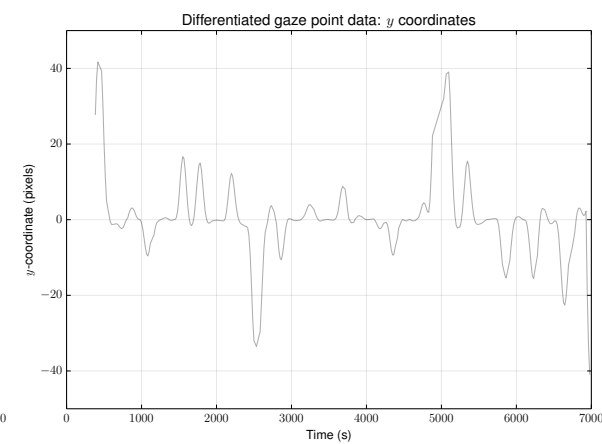
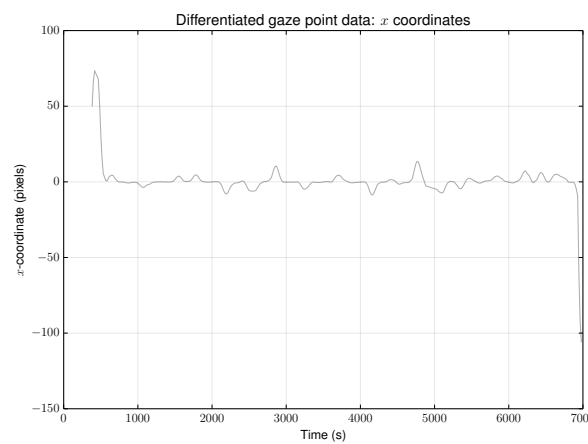
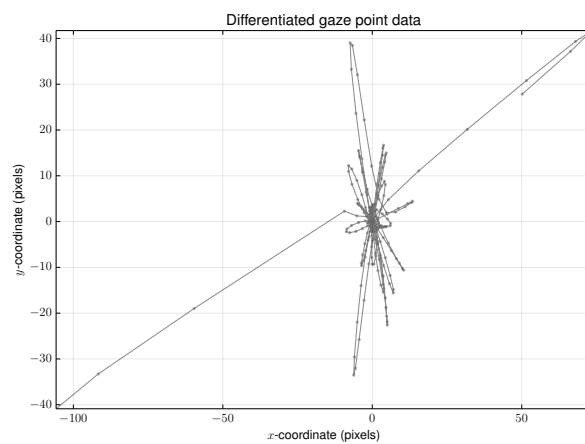
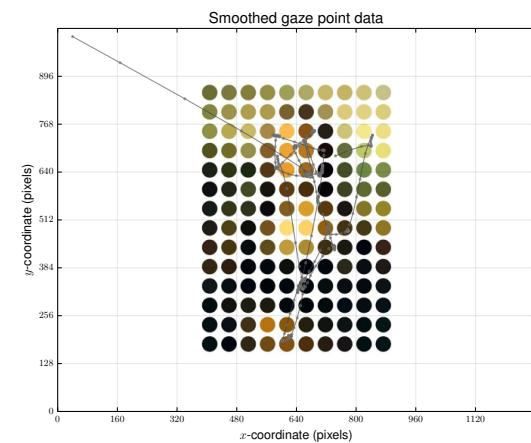
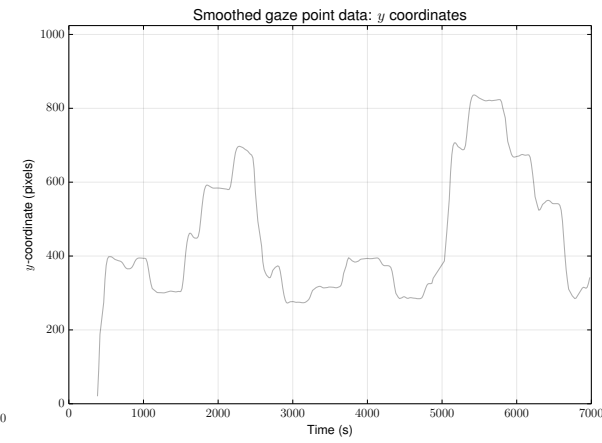
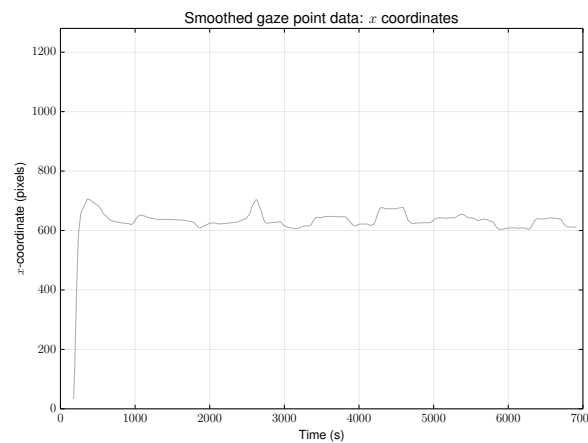
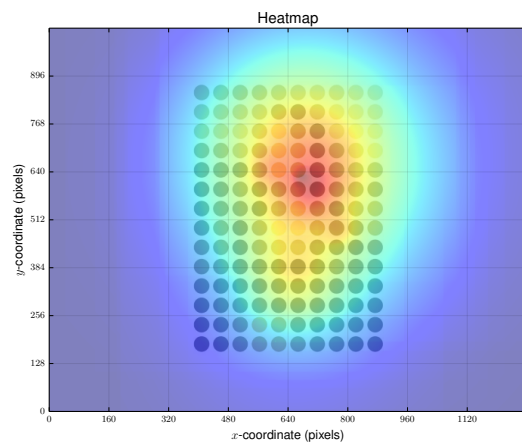
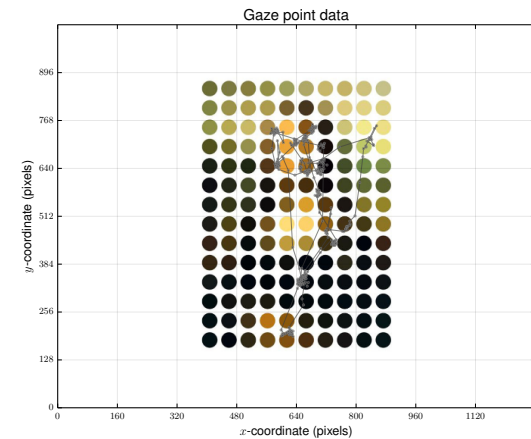
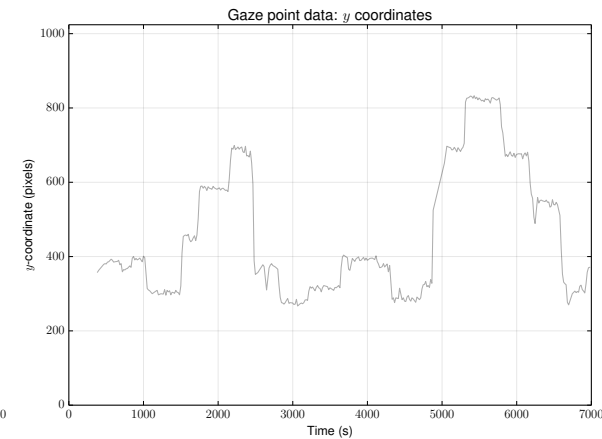
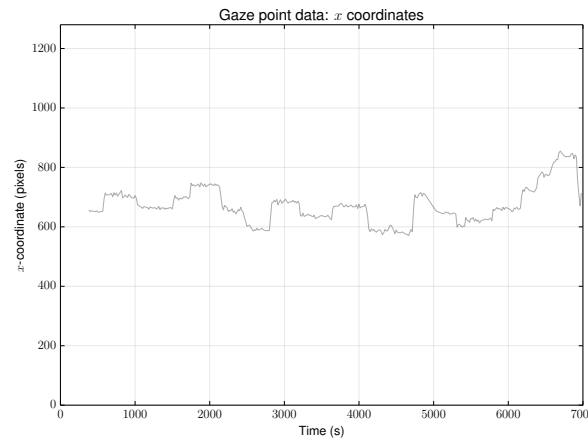
- 10 plots per trial
- 3 trials per user
- 30 users
- 900 plots total



Plotting



- Forgot heatmaps!
- 11 plots
- $11 \times 90 = 990$ plots



Parallel processing



- How to process the data efficiently?
 - $30 \times 3 = 90$ raw data files to start with
 - processing each file 11 times
- Sequentially, 990 steps...split work across 90 machines
- 91 machines actually (need one to act as *foreman*)
- Use supercomputer cluster!
 - *Palmetto* @ Clemson
 - off-campus at research park
 - *halo2, grom* @ UW
 - Uniwersytet Warszawski



interdyscyplinarne centrum
modelowania matematycznego
i komputerowego



CLEMSON UNIVERSITY
EYE TRACKING LAB
SCHOOL OF COMPUTING



- icm maszyny
 - Hydra
 - Nostromo
 - Boreasz
 - Halo2
 - Notos



Hydra



Nostromo



Boreasz



Halo2



Notos

halo2 cluster



- halo2
 - Sun Constellation
 - batch, interactive
- 512 compute nodes
 - 16 cores each
 - 8192 cores total
- Interconnects:
 - InfiniBand
- Job queuing: Torque
 - OpenPBS enhanced
- OS: Linux x86_64



interdyscyplinarne centrum
modelowania matematycznego
i komputerowego



CLEMSON UNIVERSITY
EYE TRACKING LAB
SCHOOL OF COMPUTING

CLEMSON[®]
UNIVERSITY

Poland steps up supercomputing



- Gdańsk, Kraków to get fastest supercomputers in Poland
 - AGH: Prometheus will be #1 in Poland, #30 in the world



See: <http://www.hpcwire.com/2014/10/30/poland-steps-supercomputing-game/>

Palmetto cluster



- Ranked #66 on Top 500
 - #4 among US academic research clusters
- 1,978 compute nodes
 - 20,728 cores
 - 598 NVidia Tesla GPUs
- Interconnects:
 - InfiniBand, Myrinet
- Job queuing: PBS Pro
- OS: Linux

CLEMSON[®]
U N I V E R S I T Y



Parallel programming



- Split processing up using MPI API
 - MPI handles interconnect between CPU cores
 - cores could be distributed on same node or distant nodes
 - can use MPI in C/C++ code or Python (`mpi4py`)
 - *foreman/workers* (producer/consumers) programming model

```
if rank == 0: # foreman (code courtesy of Ed Duffy @ Clemson)
    for file in list:
        comm.recv(source=MPI.ANY_SOURCE, tag=TAG_WORK_NEED, status=status)
        comm.send((file,), dest=status.source, tag=TAG_WORK_TODO)
    for i in range(1, ncores):
        status = MPI.Status()
        comm.recv(source=MPI.ANY_SOURCE, tag=TAG_WORK_NEED, status=status)
        comm.send((0,), dest=status.source, tag=TAG_WORK_DONE)
```

Parallel programming



- Think in parallel:
 - foreman waits for next available worker, sends filename
 - when all files processed, send each worker tag to quit
 - workers send foreman “ready” ticket when nothing to do

```
else: # worker (code courtesy of Ed Duffy @ Clemson)
    while True:
        comm.send((0,), dest=0, tag=TAG_WORK_NEED)
        file, = comm.recv(source=0, tag=MPI.ANY_TAG, status=status)
        if status.tag == TAG_WORK_DONE:
            break
        # process file
```


Scheduling



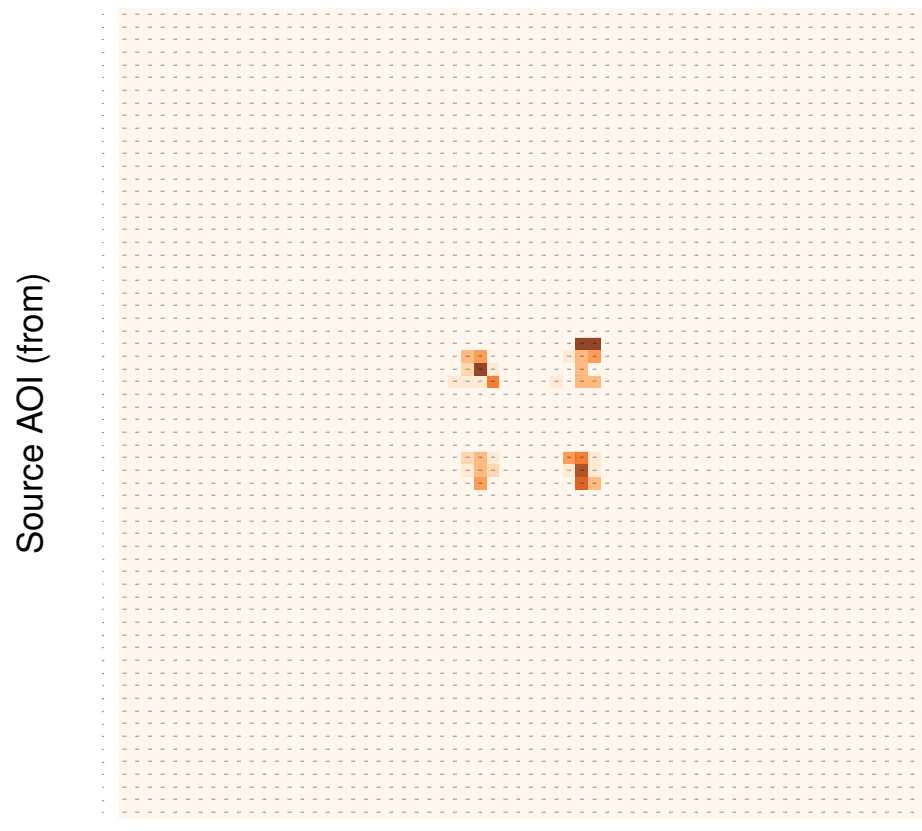
- Once program is written, how to run on cluster?
- Depends on scheduler (e.g., PBS Pro or Torque)
- PBS Pro:

```
qsub -l select=91:ncpus=1:mpiprocs=1 -l walltime=1:00:00
```

```
mpiexec -n 91 python ./src/mpi_script.py
```

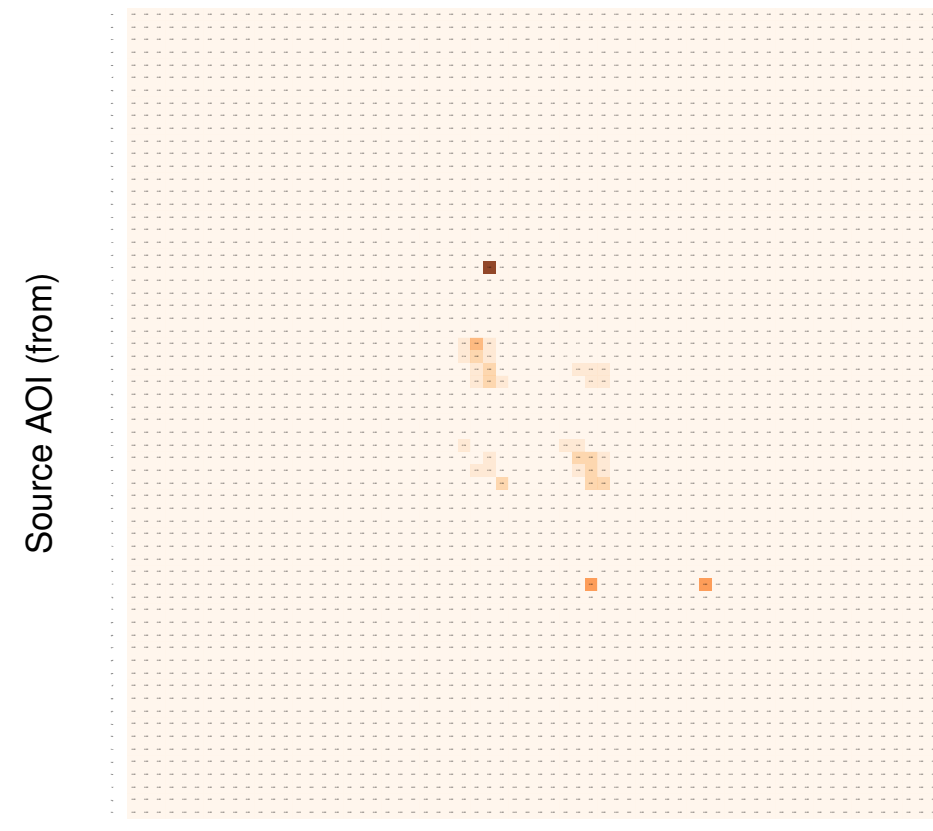
- Queuing syntax will be different, but idea is the same
- Can run in batch mode or interactively
- Number of nodes requested depends on queue, job...

Results



Destination AOI (to)

VIEWER WHO RECOGNIZED IMAGE

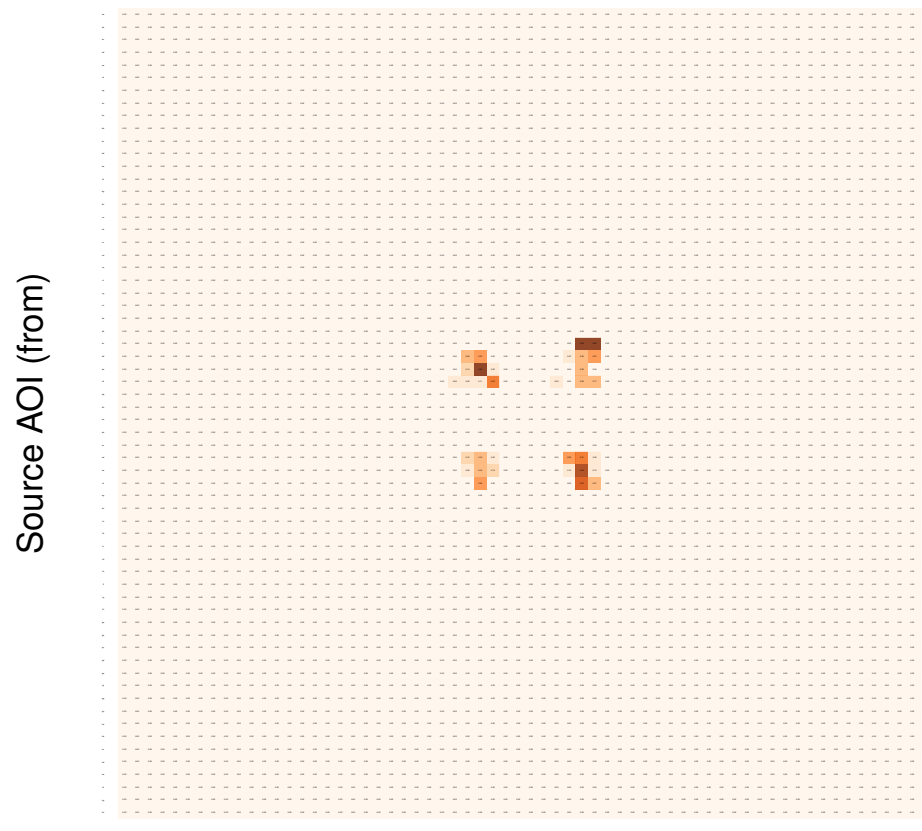


Destination AOI (to)

VIEWER WHO DID NOT RECOGNIZE

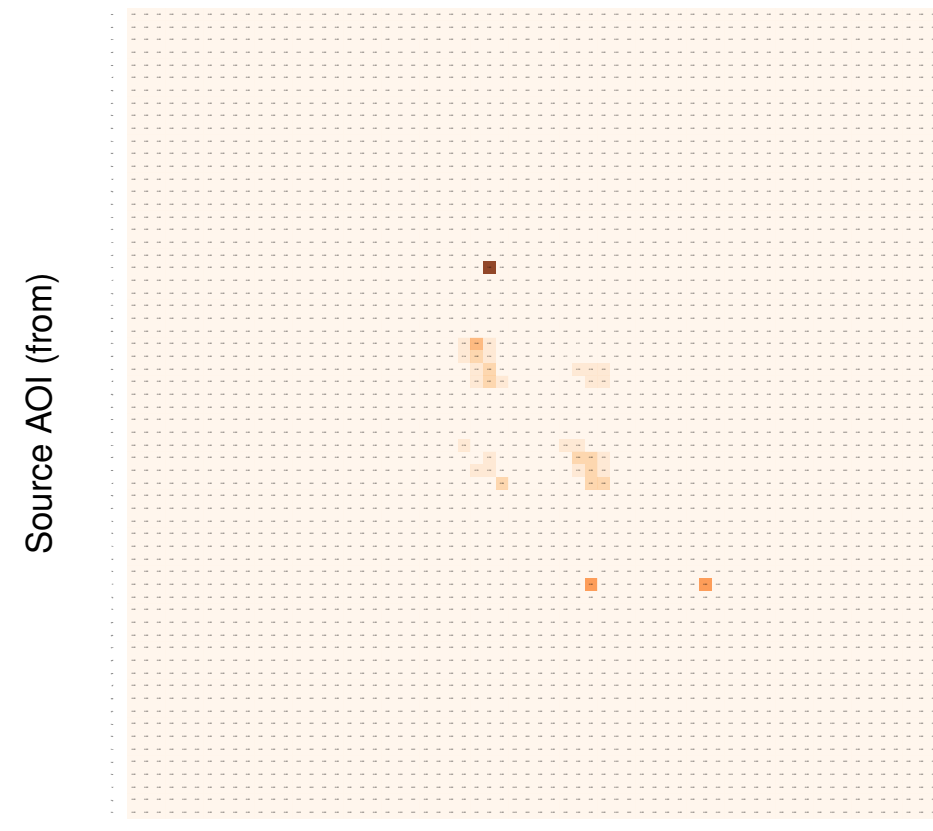
- Parallel processing produces plots, data files
- Need to collate data for statistical analysis (in R)
- In this example, compute transition matrices

Results



Destination AOI (to)

VIEWER WHO RECOGNIZED IMAGE



Destination AOI (to)

VIEWER WHO DID NOT RECOGNIZE

- Complexity of process can be measured by entropy:

$$H_t = - \sum_{i \in S} \pi_i \sum_{j \in S} p_{ij} \log p_{ij}$$

Transition and Stationary Entropies



$$H_t = - \sum_{i \in \mathcal{S}} \pi_i \sum_{j \in \mathcal{S}} p_{ij} \log p_{ij}$$

$$H_s = - \sum_{i \in \mathcal{S}} \pi_i \log \pi_i$$

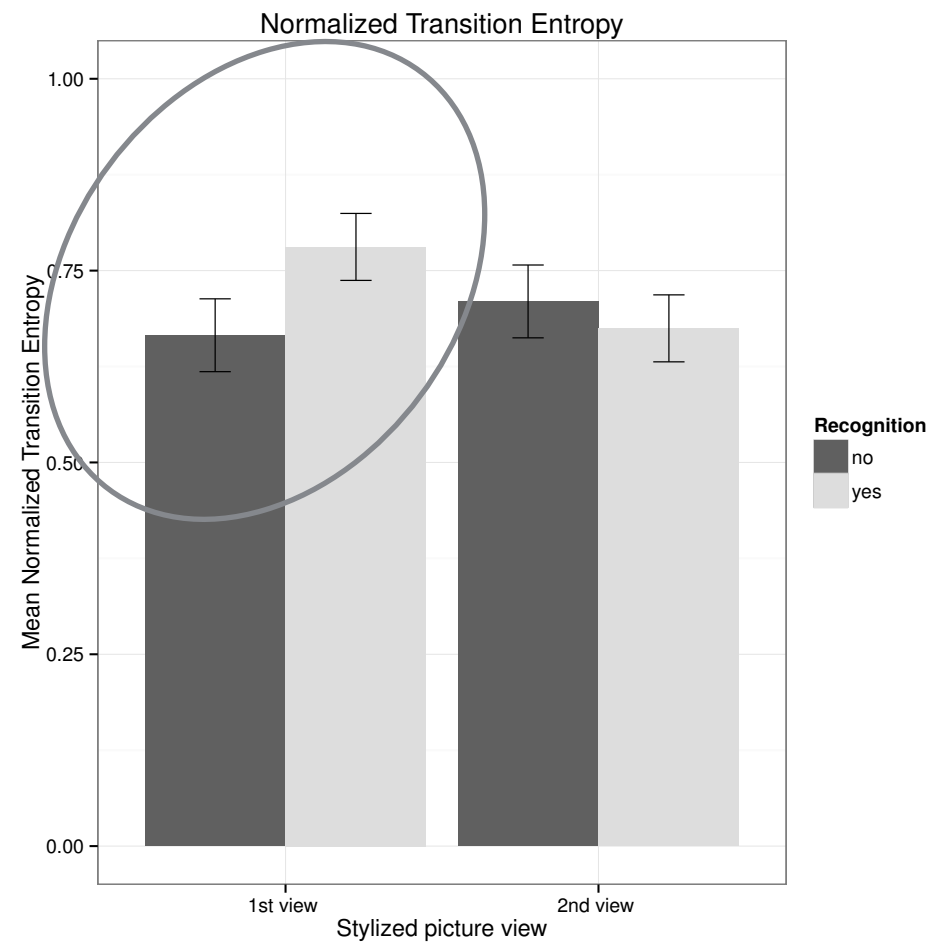
$$\mathbf{P} = \begin{pmatrix} 0.78 & 0.06 & 0.16 \\ 0.18 & 0.46 & 0.36 \\ 0.15 & 0.09 & 0.76 \end{pmatrix}, \quad \pi = (0.42, 0.14, 0.44)$$

$$H_t = 0.73$$

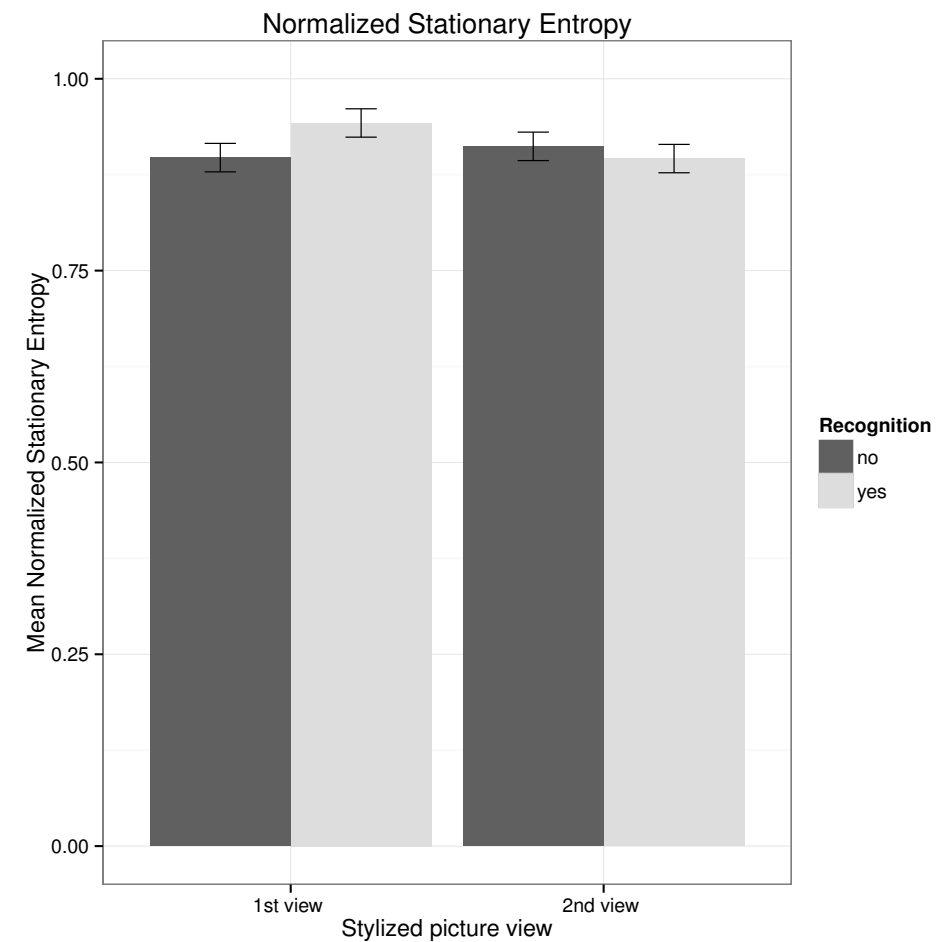
$$H_s = 1.00$$

- A higher value of entropy means “surprise!”
- Stationary entropy considers entropy in the long run

Results



TRANSITION ENTROPY



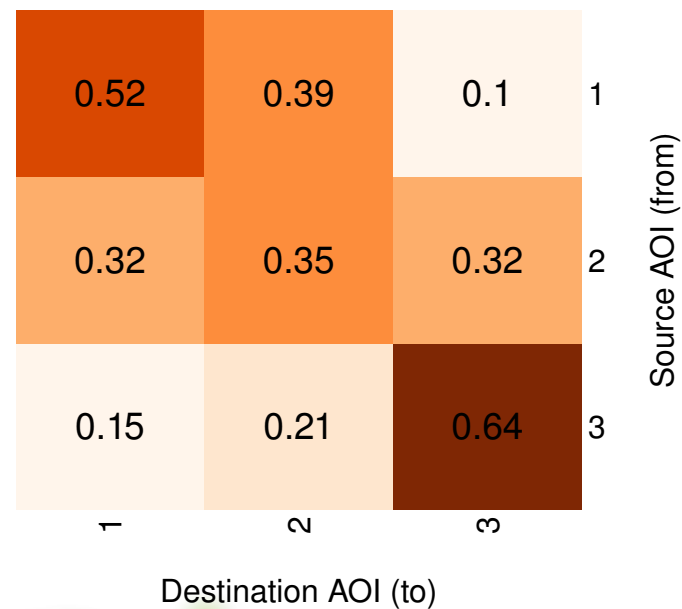
STATIONARY ENTROPY

- Those who recognized image showed higher transition entropy than those who did not ($F(1,28)=8.77$, $p < 0.01$)
- Similarly for stationary entropy

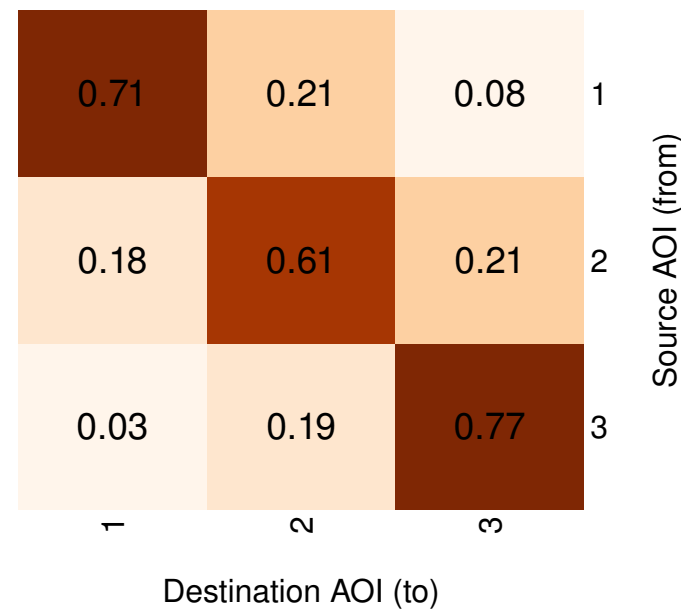
Example (ETRA 2014)



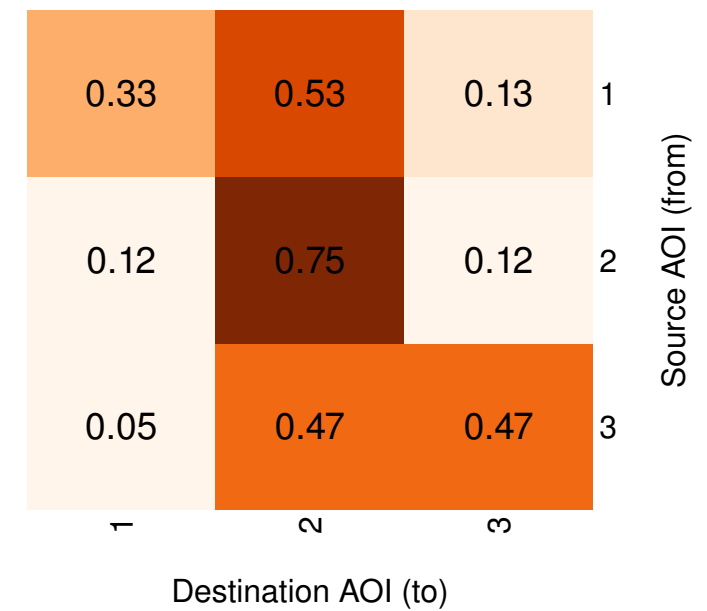
$H_t = 0.98, H_s = 1.10$



$H_t = 0.77, H_s = 1.09$



$H_t = 0.80, H_s = 0.88$



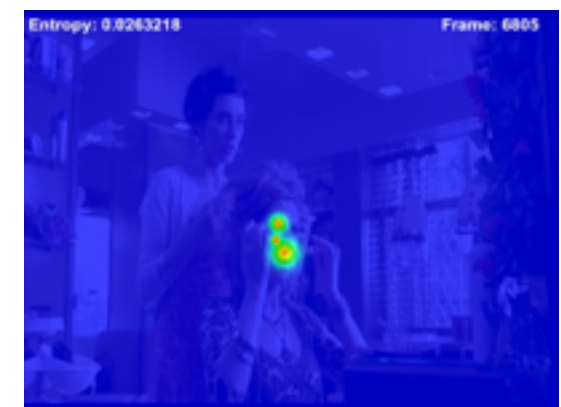
Distributed computing



- Above example distributes work among compute nodes
- Can combine MPI parallelization with GPU processing
- Example: gaze over subtitles vs. voice-over (dubbing)



SUBTITLES



VOICE-OVER

Distributed rendering



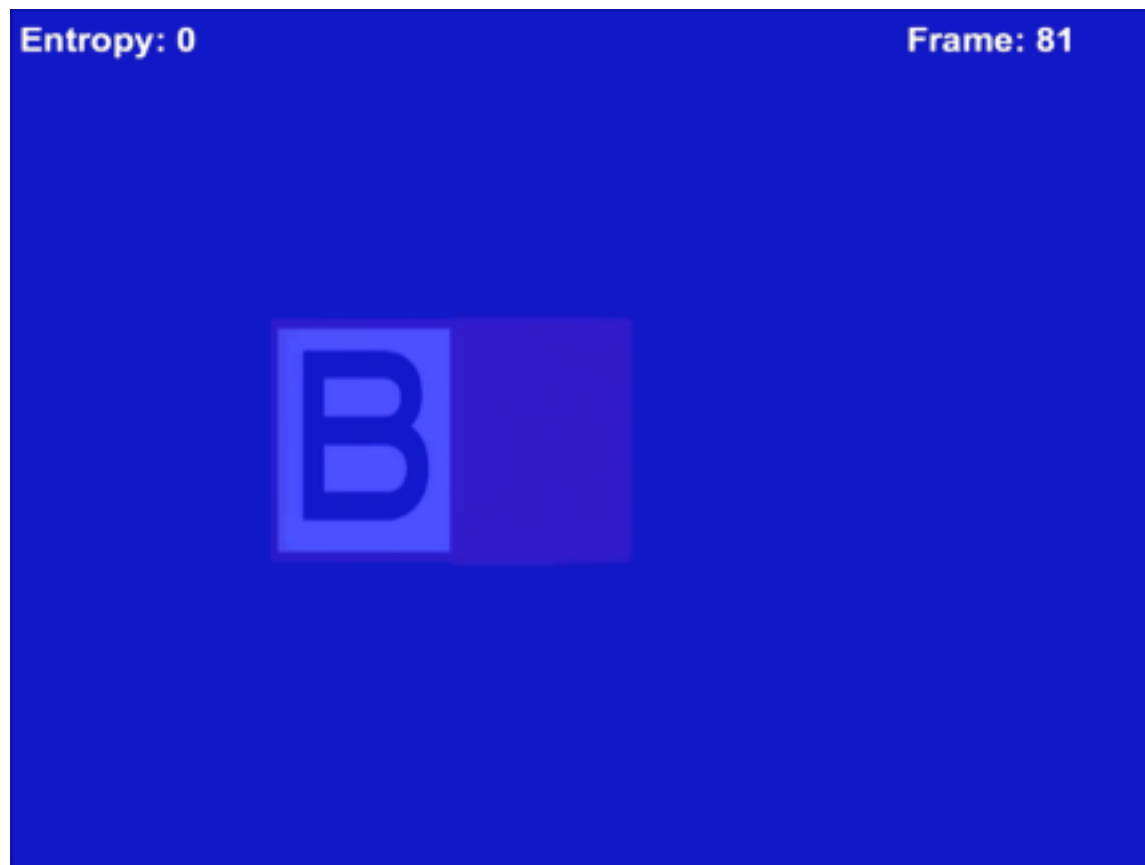
- Use GPU to render heatmaps (OpenCL)

```
__kernel
void gaussian(int2 dim, float2 pos, float sigma, __global float *gauss)
{
    // Get the work-item's unique ID
    int2 idx = {get_global_id(1), get_global_id(0)};
    float sigma2sq = -2.0 * sigma * sigma;

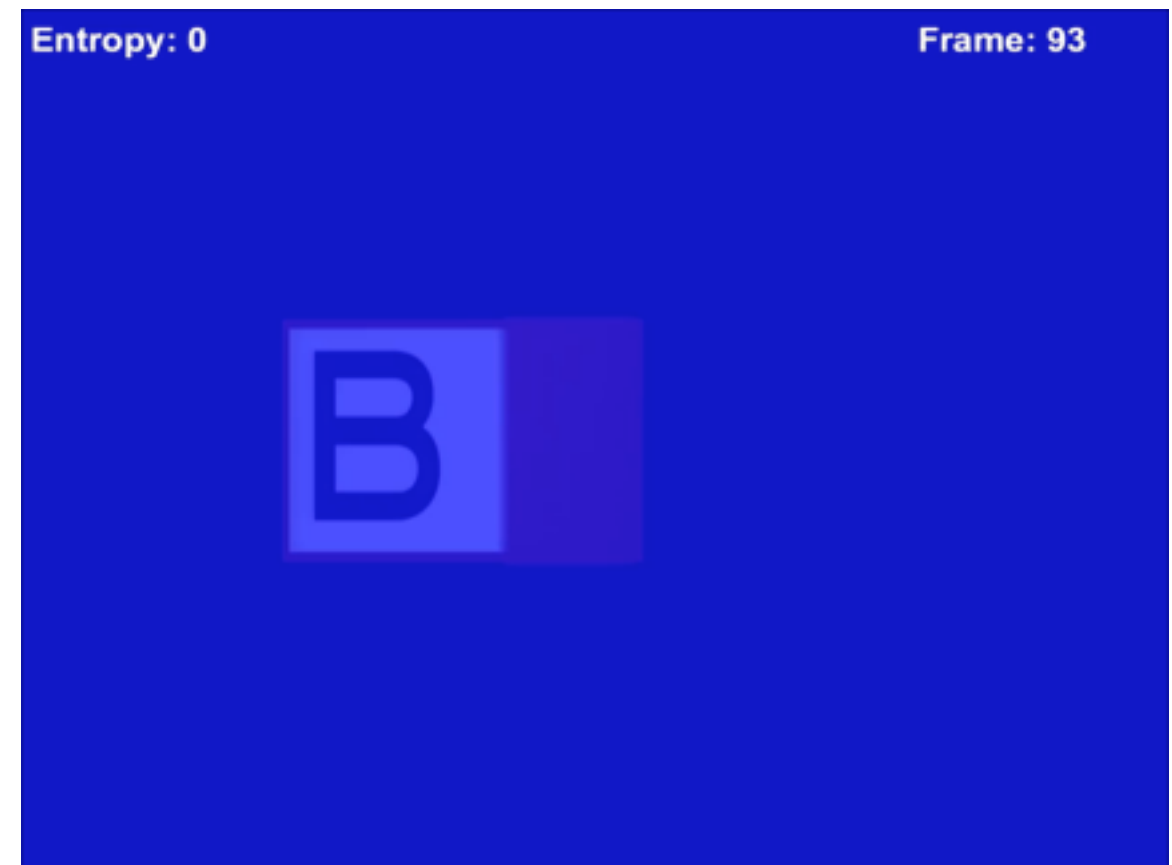
    float sx = idx.x-pos.x;
    float sy = idx.y-pos.y;
    float res = exp((sx*sx+sy*sy)/sigma2sq + (0.0));
    gauss[idx.y*dim.x+idx.x] += res;
}
```

- see Duchowski et al. (2012)

Heatmap rendering



SUBTITLES



VOICEOVER

- How to compare gaze atop video?

Heatmap comparison



- Time-sliced approach:
 - aggregate scanpaths for w video frames



SUBTITLES



VOICE-OVER

- Compute similarity metric:
 - use heatmap entropy?
 - use heatmap difference entropy?
 - Nearest-Neighbor Index?

$$H_f = - \sum_{i \in 256} p_i \log p_i$$

Nearest-Neighbor Index

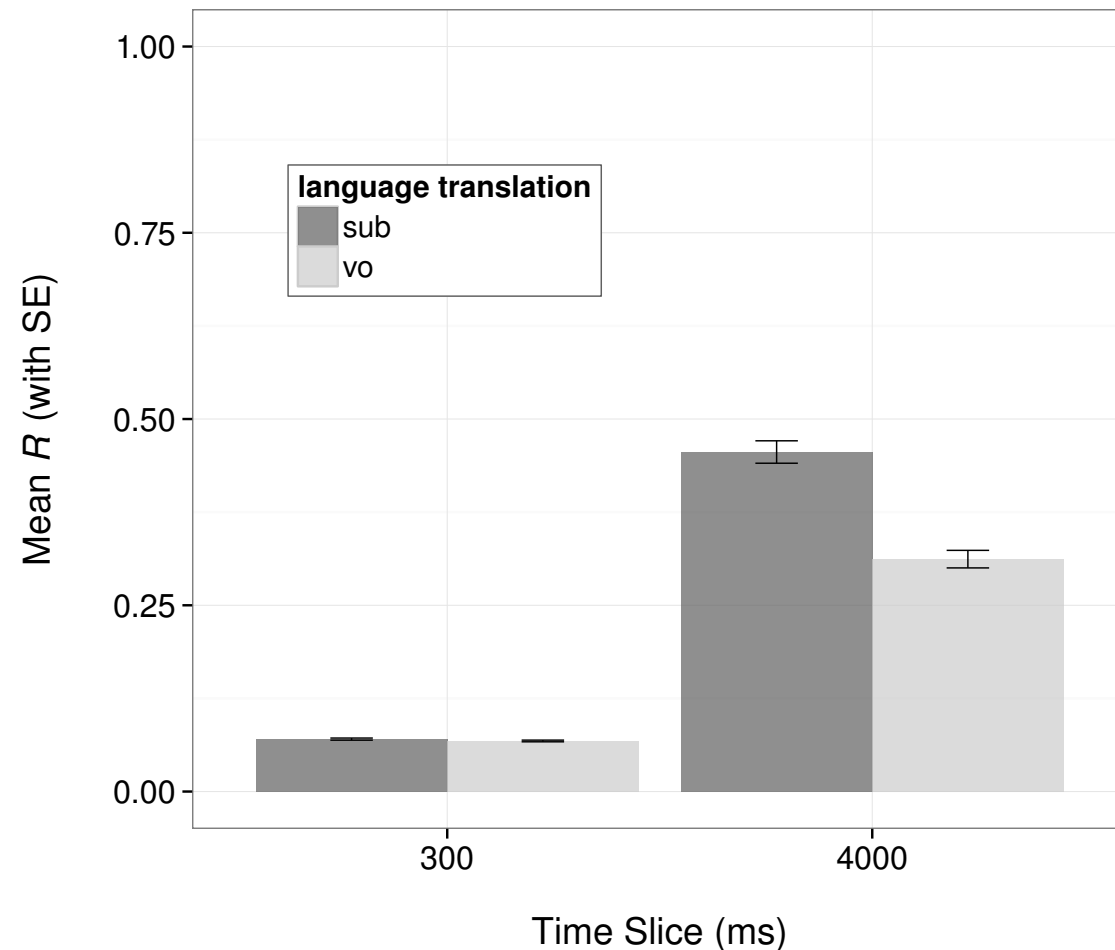


- Given n fixations per time slice,
 - compute r_i , distance from i^{th} fixation to its nearest neighbor
 - Nearest Neighbor Index
$$R = \frac{2\sqrt{\rho}}{n} \sum_i^n r_i$$
 - where ρ is the density of the observed distribution, $\rho = n/A$
 - with A denoting observation area
- Use *kd*-tree for fast nearest-neighbor query
- NNI properties:
 - $R > 1$, ordered distribution (e.g., hexagonal grid)
 - $R = 1$, random distribution
 - $R < 1$, clustered
 - $R = 0$, maximum aggregation (singularity)

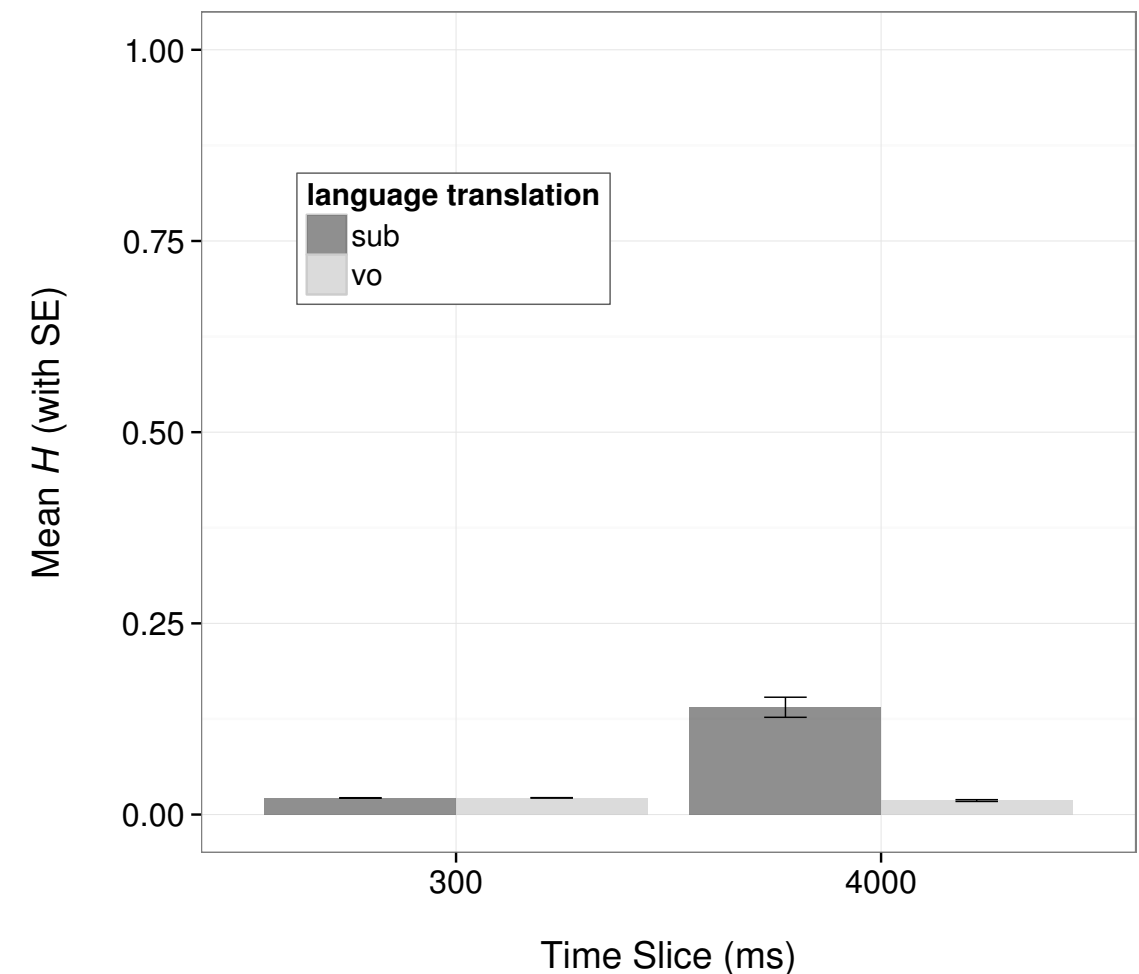
Preliminary results



Nearest Neighbor Index ($R = \frac{2\sqrt{p}}{n} \sum_i^n r_i$)



Heatmap Entropy ($H = - \sum_{i \in 256} p_i \log(p_i)$)



- Significant effect of subtitles on gaze distribution
- But only for larger time slice (4 sec)



Eye Movement Simulation



CLEMSON UNIVERSITY
EYE TRACKING LAB
SCHOOL OF COMPUTING

CLEMSON[®]
UNIVERSITY

Eye movement simulation



- Given:
 - primary gaze vector $\mathbf{g}_p$ and reference gaze direction $\mathbf{g}_e$
 - $\mathbf{V}_e$ the bisector of $\mathbf{g}_p$ and $\mathbf{g}_e$ that is $\perp$ to displacement plane
- Rotation between $\mathbf{g}_p$ and $\mathbf{g}_e$ is expressed by quaternion

$$\mathbf{q} = [\mathbf{V}_e \cdot \mathbf{g}_e, (-\mathbf{V}_e \times \mathbf{g}_e)]$$

- Quaternion $\mathbf{q}$ is a vector with four components

$$\mathbf{q} = (q_0, q_\tau, q_V, q_H)$$

- with q_0 the scalar, and q_τ, q_V , and q_H the torsional, vertical, and horizontal components, respectively

Eye movement simulation

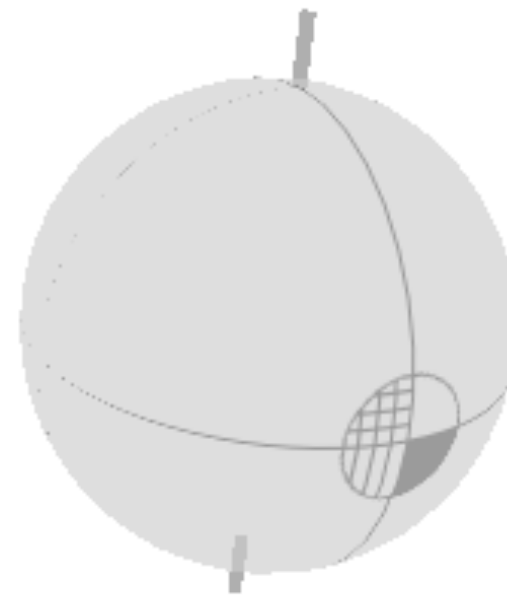


- Assume:
 - displacement plane is (initially) aligned with Listing's plane

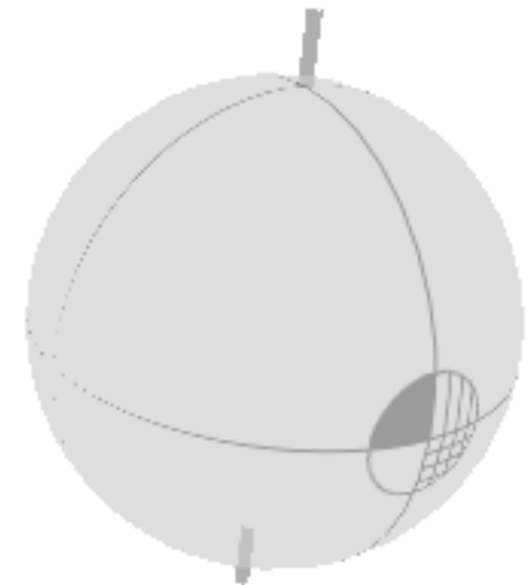


- Modeled eyeball just points down $(0, 0, 1)$ z -axis

Eyeball rotation



CORRECT



IMPLAUSIBLE

- Use Listing's and Donder's laws:
 - convenient simplification & physiologically seen constraint
 - $q_{\tau} = 0$, negligible visible torsion (when head is fixed)

Adhering to Listing's law



- Establish “look point” in space, $\hat{\mathbf{x}}_k = (\hat{p}_{k,x}, \hat{p}_{k,y}, \hat{p}_{k,z})$
- Set gaze direction vector $\mathbf{g}_r$ to point at reference point $\hat{\mathbf{x}}_k$
- Specify rotation quaternion's plane as (f, f_V, f_H)
- Express q_τ as a function of q_H and q_V :

$$q_\tau = f + f_V q_V + f_H q_H$$

- If $f \neq 0$, $\mathbf{q} = (q_0, q_\tau, q_V, q_H)$ does not satisfy Listing's law
- Quaternion $\mathbf{e} = (\sqrt{1 - f^2}, f, 0, 0)$ does, however

Adhering to Listing's law



- To force adherence to Listing's law, right-multiply $\mathbf{q}$ by:

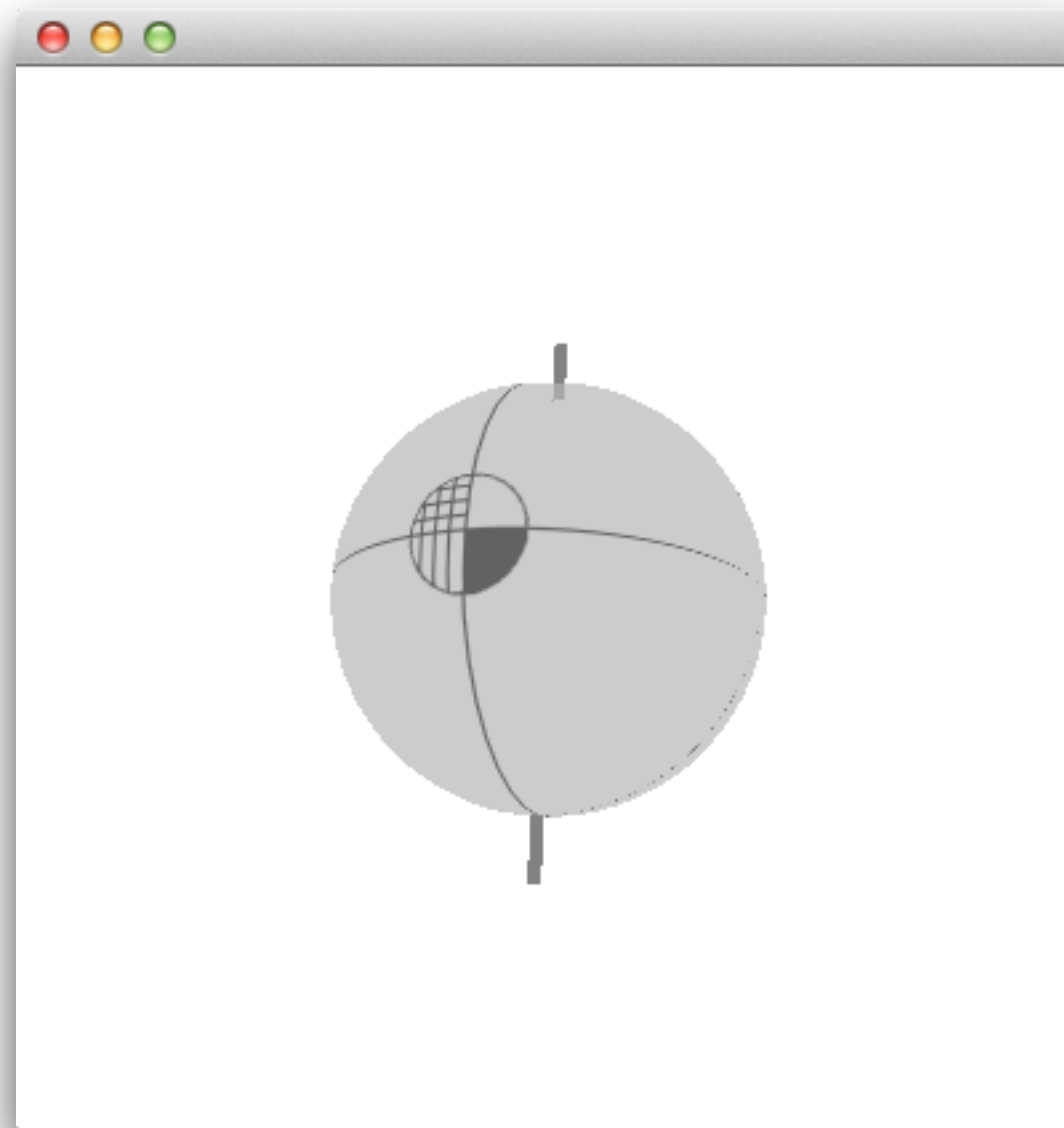
$$\mathbf{e}^{-1} = \left(\sqrt{1 - f^2}, -f, 0, 0 \right)$$

- This adjusts rotation quaternion's torsional component
- To find Listing's plane, compute normal vector:
 - specify quaternion of primary position relative to $\mathbf{e}$ as:

$$\mathbf{p} = (V_1, V_0, -V_3, V_2)$$

- where $V = (1, -f_V, -f_H) / |(1, -f_V, -f_H)|$
- Quaternion $\mathbf{q}$ is then left-multiplied by $\mathbf{p}^{-1}$ giving $\mathbf{p}^{-1}\mathbf{q}\mathbf{e}^{-1}$
(the corrected rotation quaternion adhering to Listing's law)

DEMO

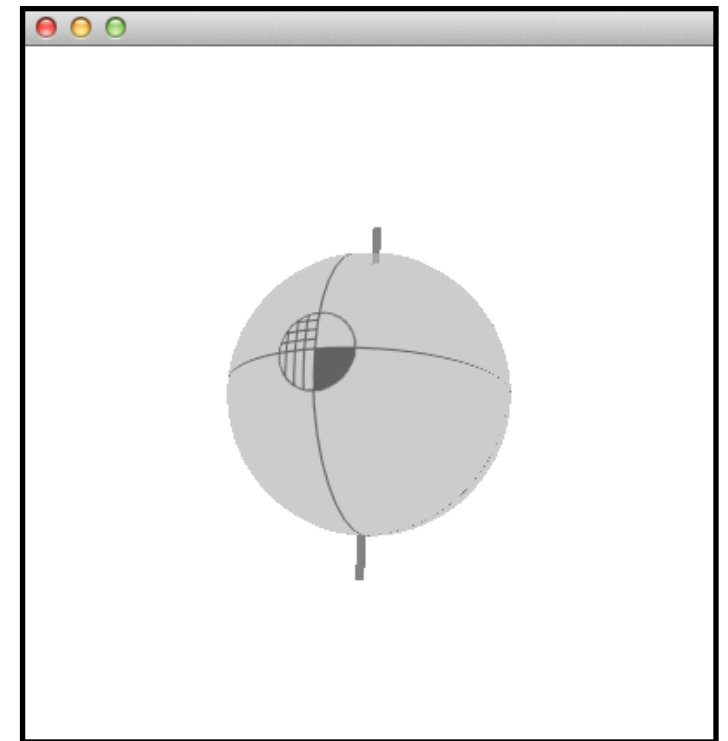


`rainbow/research/emm/src/qtEye`

Problems...



- Rotations seem to be working, but...
 - need Listing's plane visualization
 - what happens when plane rotates? (we get torsion ✓)
 - what about vergence? (need two eyes)
- Need to automate rotation animation...
 - need to get eyeball to saccade by itself
- **How about a stochastic model?**



Stochastic modeling



- To model eyeball rotation, animate look point in 2D:

$$\hat{\mathbf{x}}_k = (\hat{p}_{k,x}, \hat{p}_{k,y}, \hat{p}_{k,z}) = (\hat{p}_{k,x}, \hat{p}_{k,y}, 0)$$

- First attempt:

- model next fixation at randomly generated location,

$$\hat{p}_{k,x} = \mathcal{N}(0.5, 0.1), \quad \hat{p}_{k,y} = \mathcal{N}(0.5, 0.1)$$

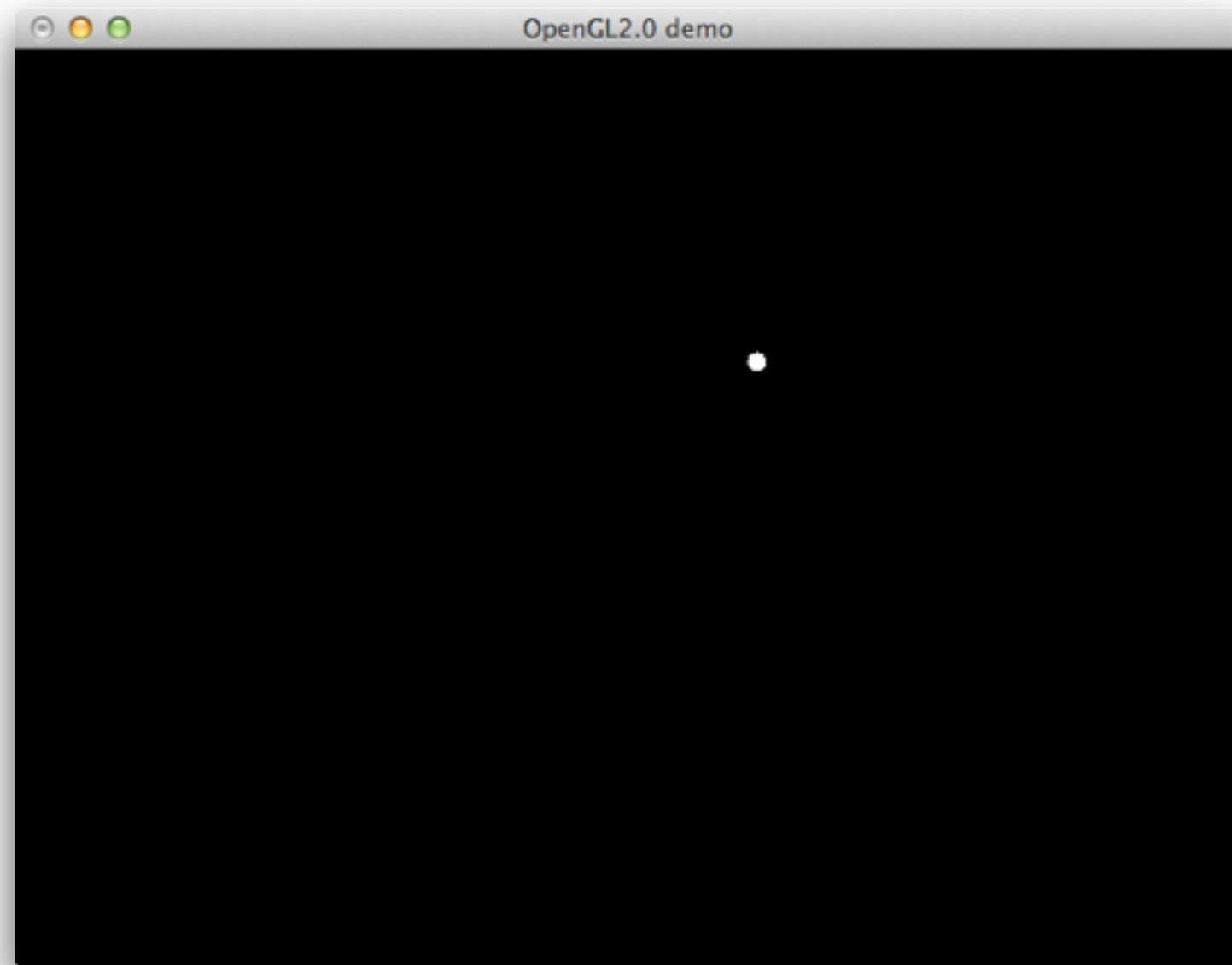
- governed by Normal distribution (normalized coordinates)

- fixation duration modeled by Poisson distribution
(range: [150,650] ms)

- gaze jitter: perturbation modeled by Normal distribution:

$$\hat{p}_{k,x} + \mathcal{N}(0.0, \Delta x), \quad \hat{p}_{k,y} + \mathcal{N}(0.0, \Delta y)$$

DEMO



`rainbow/research/emm/src/eyesim/C++/simple`



CLEMSON UNIVERSITY
EYE TRACKING LAB
SCHOOL OF COMPUTING

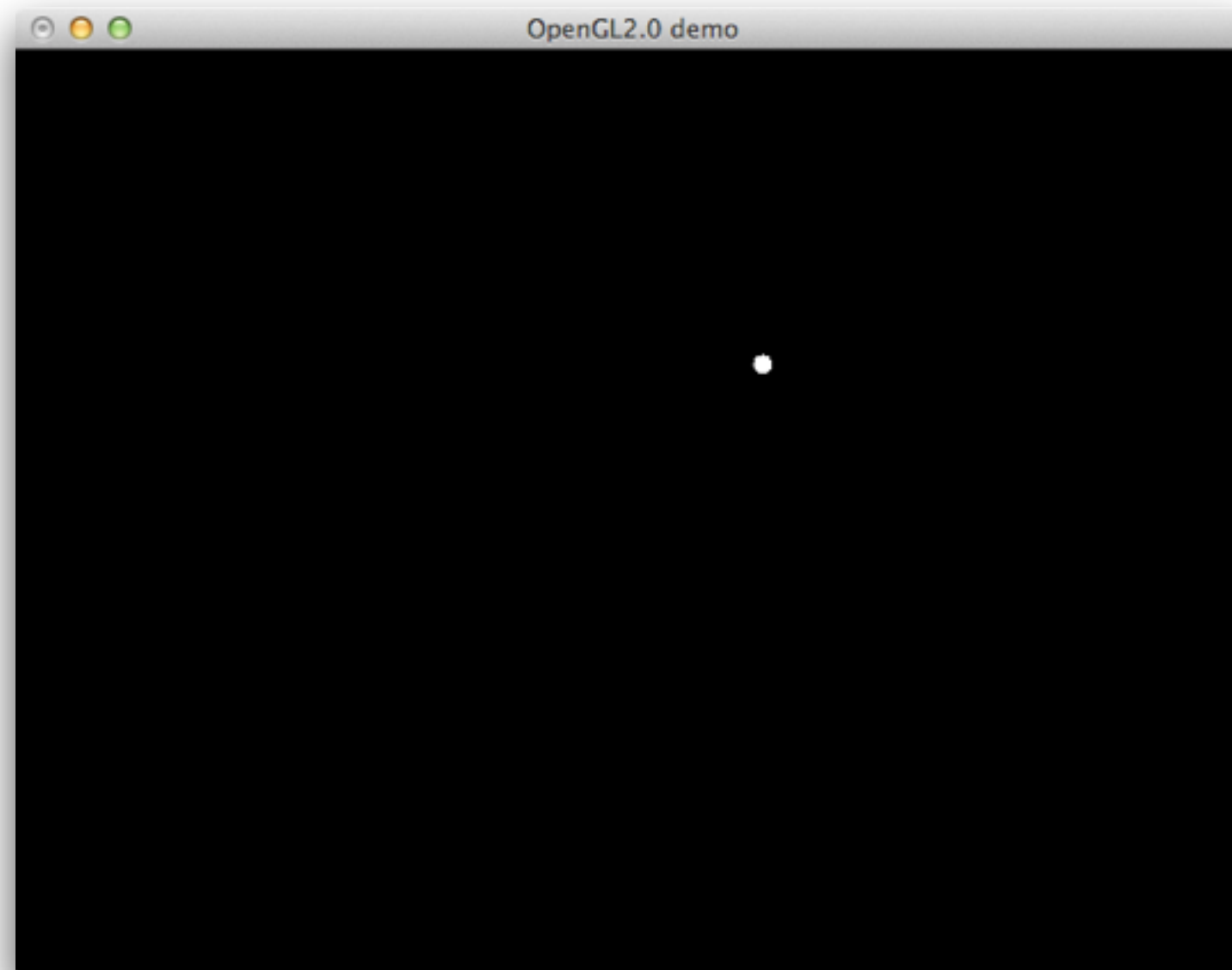
CLEMSON
UNIVERSITY

Stochastic modeling



- Saccades are generally stereotyped:
 - relationship between saccade amplitude θ and duration Δt
$$\Delta t = 2.2\theta + 21$$
 - known as the *main sequence* (for saccades up to ~ 20 deg.)
 - fairly intuitive: large saccades (rotations) performed faster
- Second attempt:
 - next fixation determined by modeling saccadic amplitude
$$\theta = \mathcal{P}(\lambda), \quad \lambda = 5^\circ$$
 - fixation duration modeled via main sequence
 - direction of saccade randomly generated, s.t., $0^\circ \leq \theta \leq 360^\circ$

DEMO



`rainbow/research/emm/src/eyesim/C++/sequence`



CLEMSON UNIVERSITY
EYE TRACKING LAB
SCHOOL OF COMPUTING

CLEMSON
UNIVERSITY

Stochastic modeling



- To model eyeball rotation, animate look point in 2D:

$$\hat{\mathbf{x}}_k = (\hat{p}_{k,x}, \hat{p}_{k,y}, \hat{p}_{k,z}) = (\hat{p}_{k,x}, \hat{p}_{k,y}, 0)$$

- Third attempt:

- model next fixation with randomly generated amplitude,

$$\hat{p}_{k,x} = \mathcal{N}(0, \sqrt{5}) + 17, \quad \hat{p}_{k,y} = \mathcal{N}(0, \sqrt{5}) + 17$$

- governed by Normal distribution (degrees visual angle)

- fixation duration modeled by main sequence

- gaze jitter: perturbation modeled by pink noise:

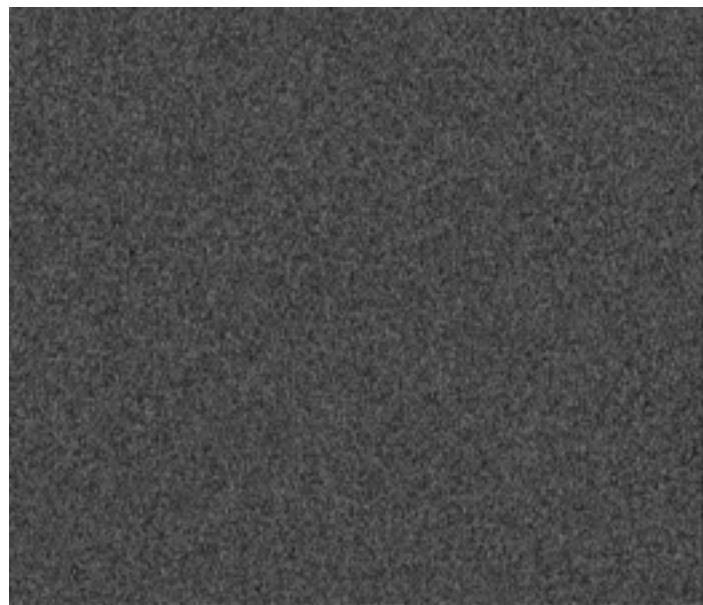
$$\hat{p}_{k,x} + H_{n,x}(z), \quad \hat{p}_{k,y} + H_{n,y}(z)$$

Pink noise

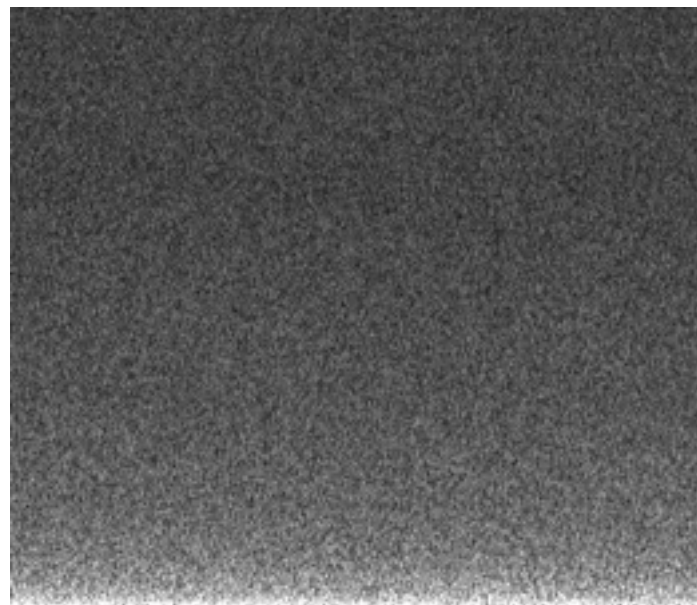


$$H_n(z) = \prod_{k=1}^n \frac{z - b_k}{z + b_k} \quad \text{where} \quad b_k = \cos \left(\frac{2\pi k}{2n + 1} \right)$$

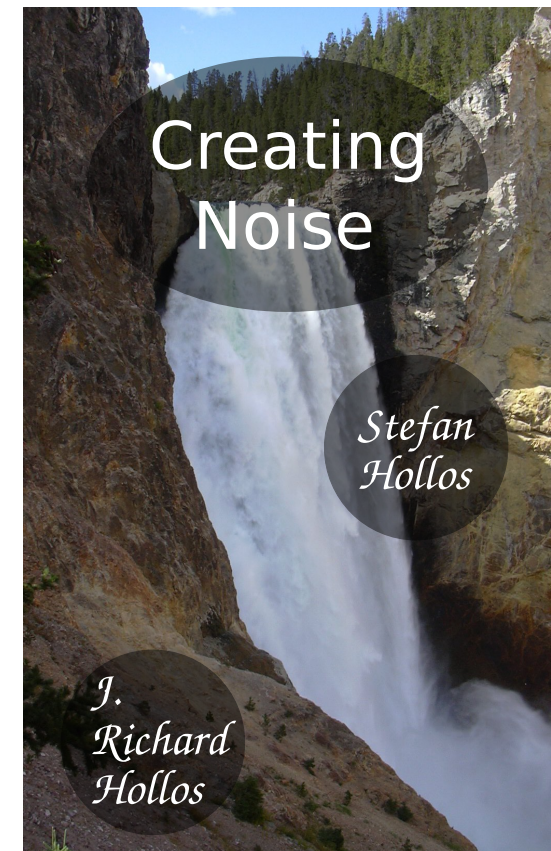
- Known as $1/f$ noise since most of the energy is concentrated in lower spectrum



white noise



pink noise

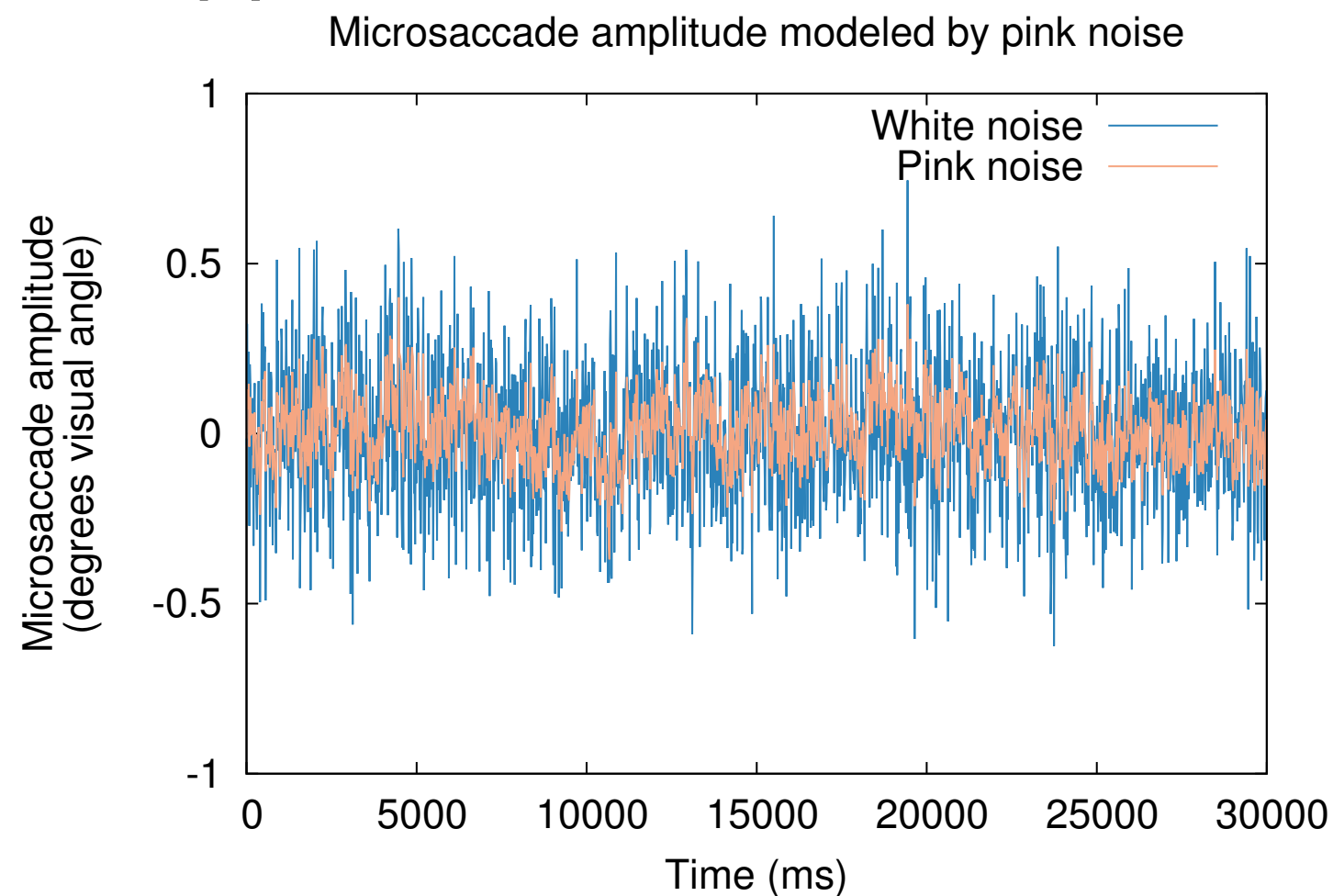


Pink noise

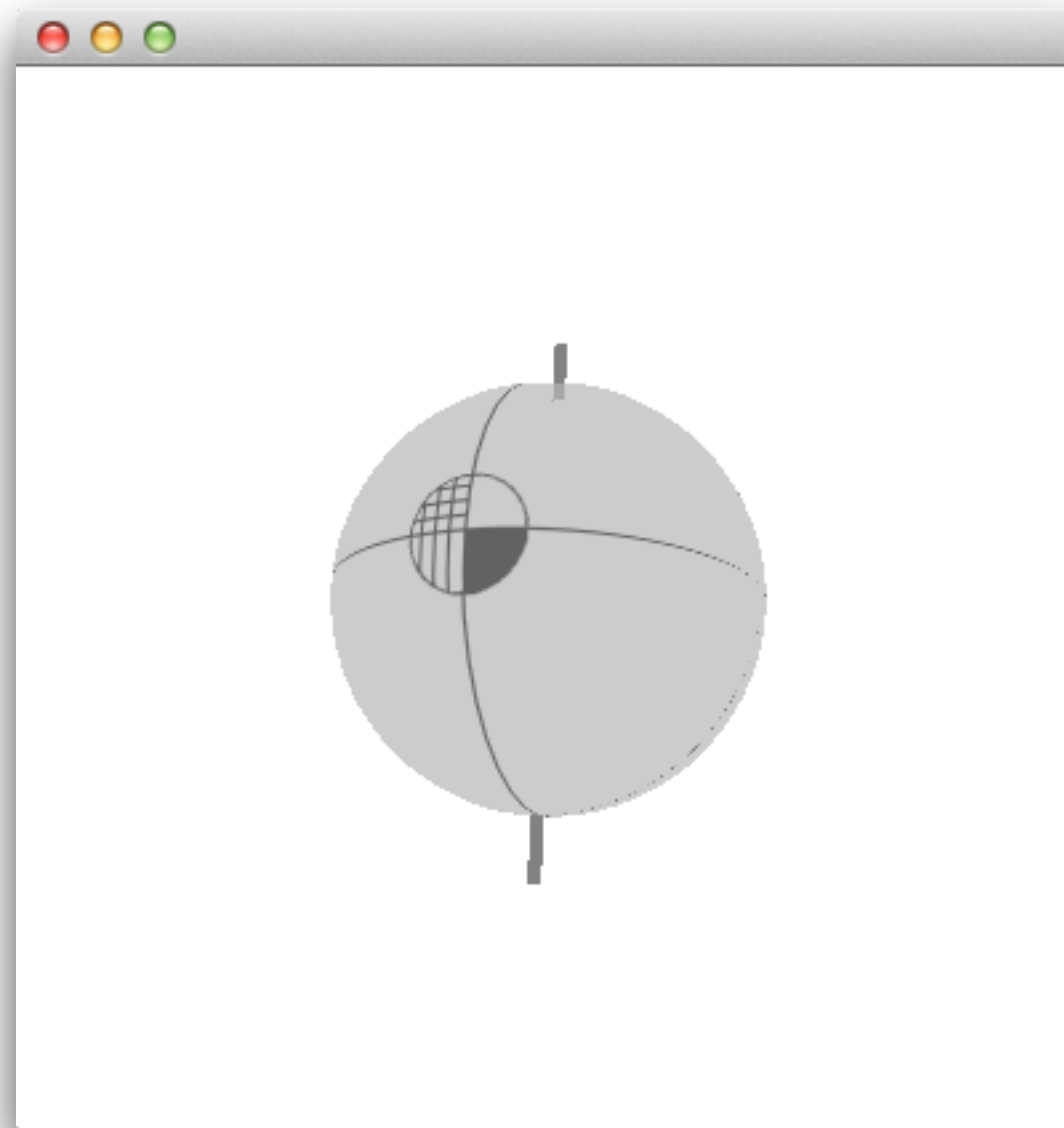


$$H_n(z) = \frac{1}{G_n(\alpha)} \prod_{k=1}^n \frac{z - q_k}{z + p_k}, \quad G_n(\alpha) = \prod_{k=1}^n \frac{\alpha + a_k}{a_k \alpha + 1}$$

- where $G_n(\alpha)$ is the n^{th} order approximation to an ideal pink noise filter
- Filter gives pink noise when fed Gaussian (white) noise



DEMO



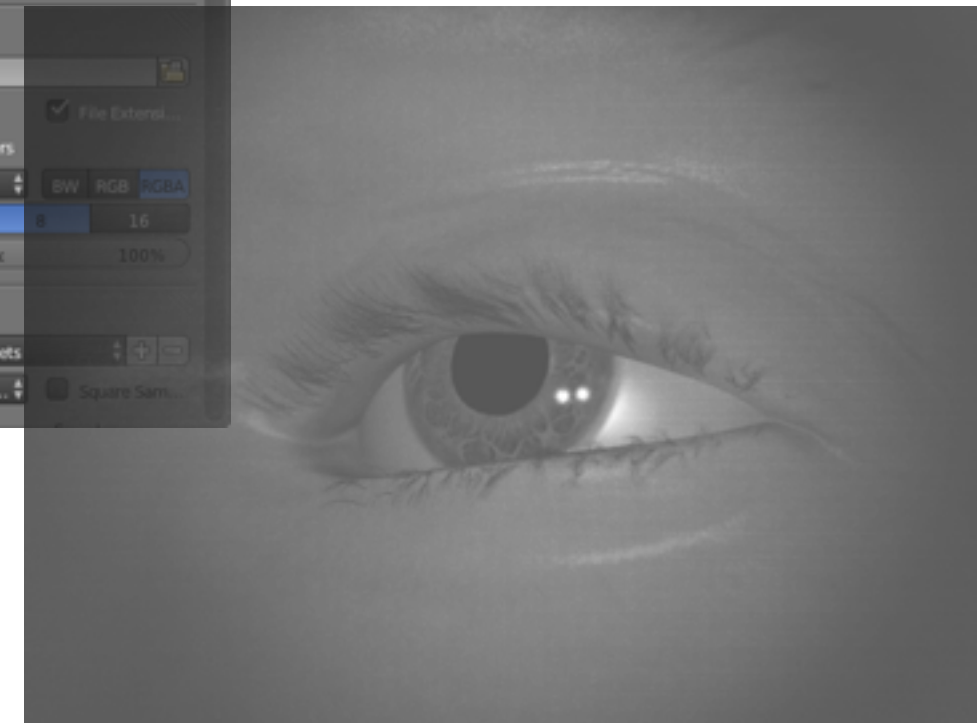
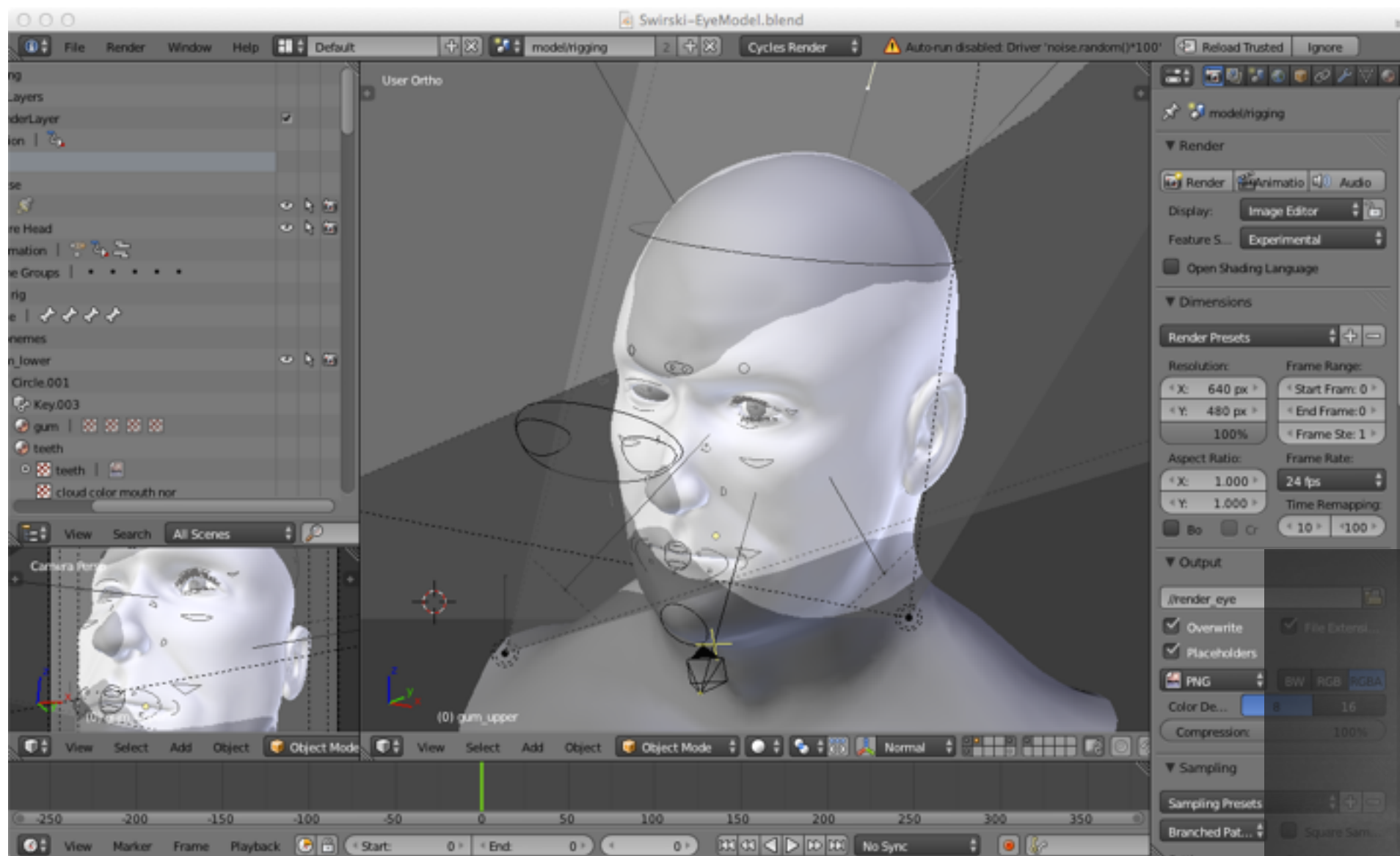
`rainbow/research/emm/src/qtEye`

Next steps...



- Impose 2D fixation sequence, e.g.,
 - 5- or 9-point calibration
 - reading model (recent spate of papers on this)
- Validate fixation detection algorithms:
 - will they detect 5- or 9-point calibration fixations?
- Validate stochastic model:
 - how large should fixation jitter be?
- Apply 2D model to 3D rotation simulation

Apply to 3D rendering



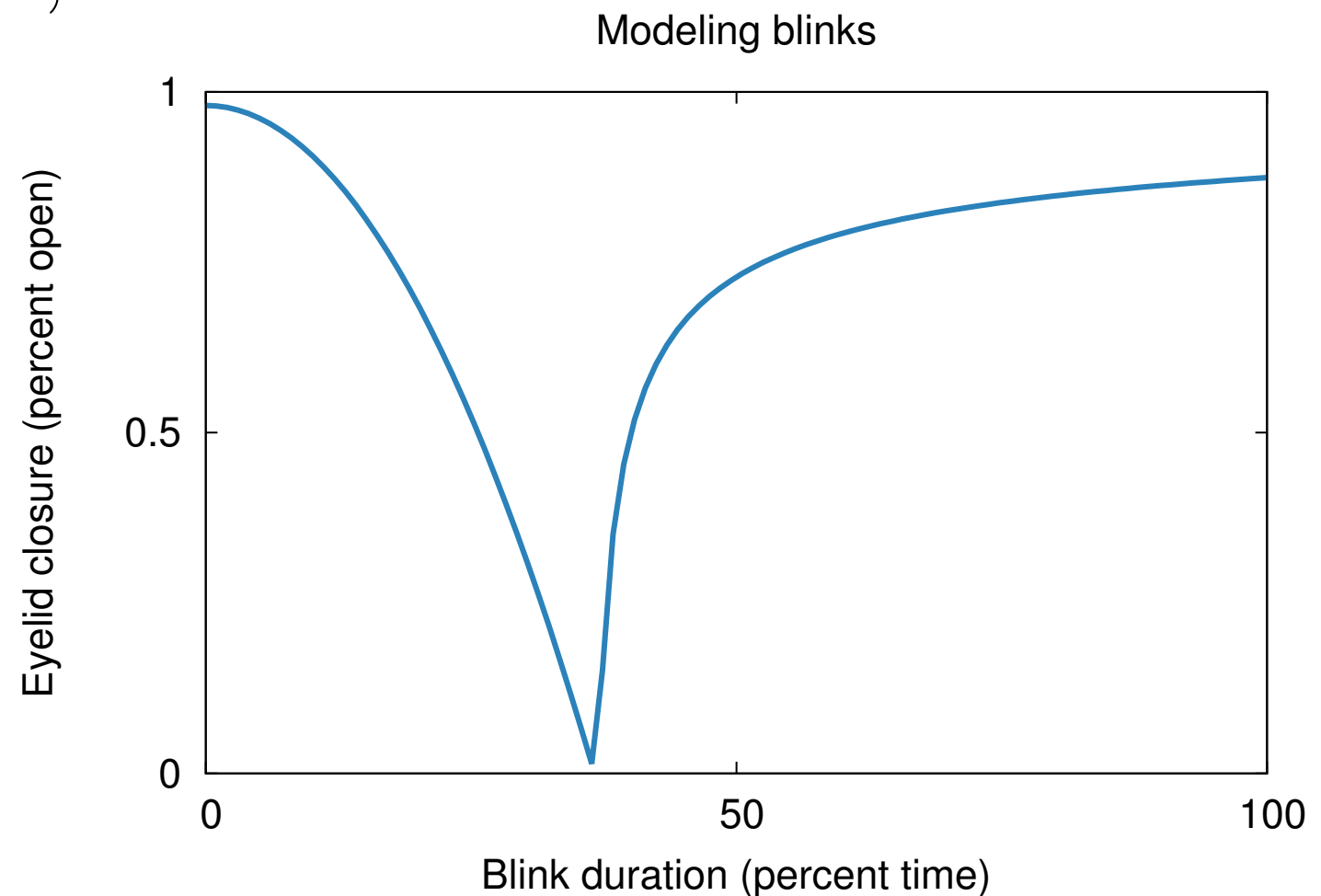
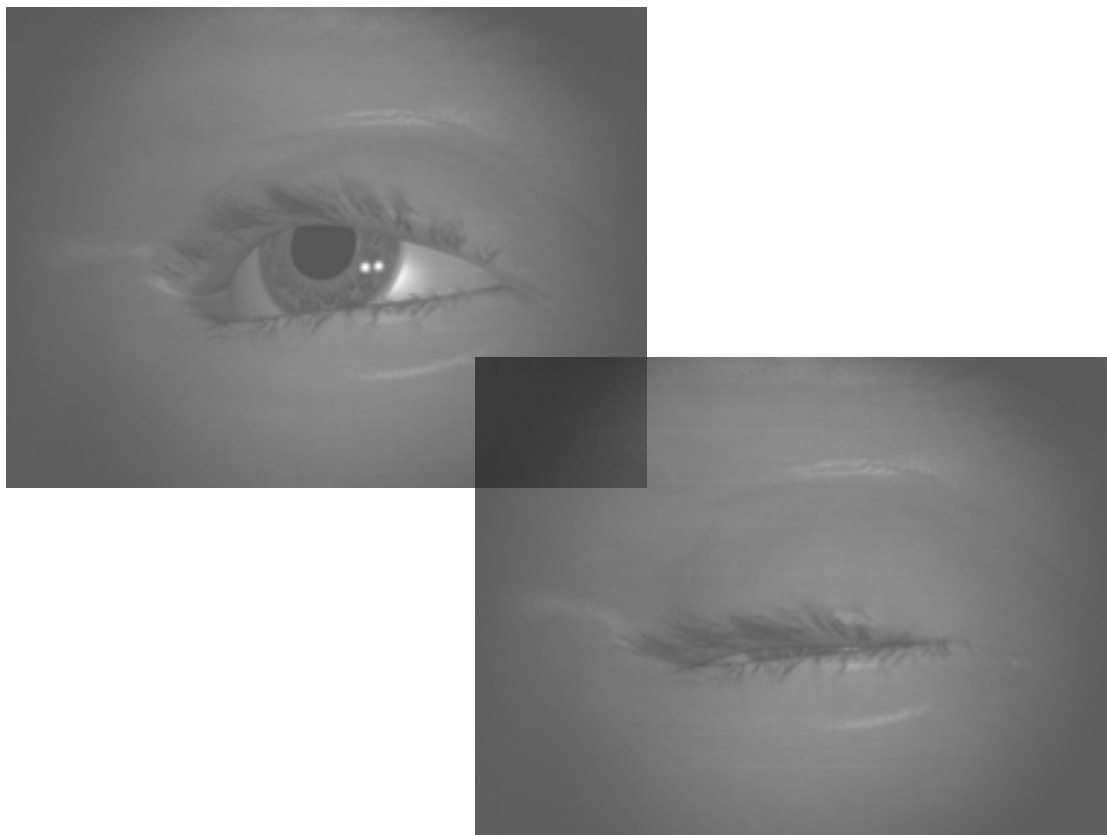
- Lech Świrski's Blender model

Modeling blinks



- Eyelid motion is faster on the way down

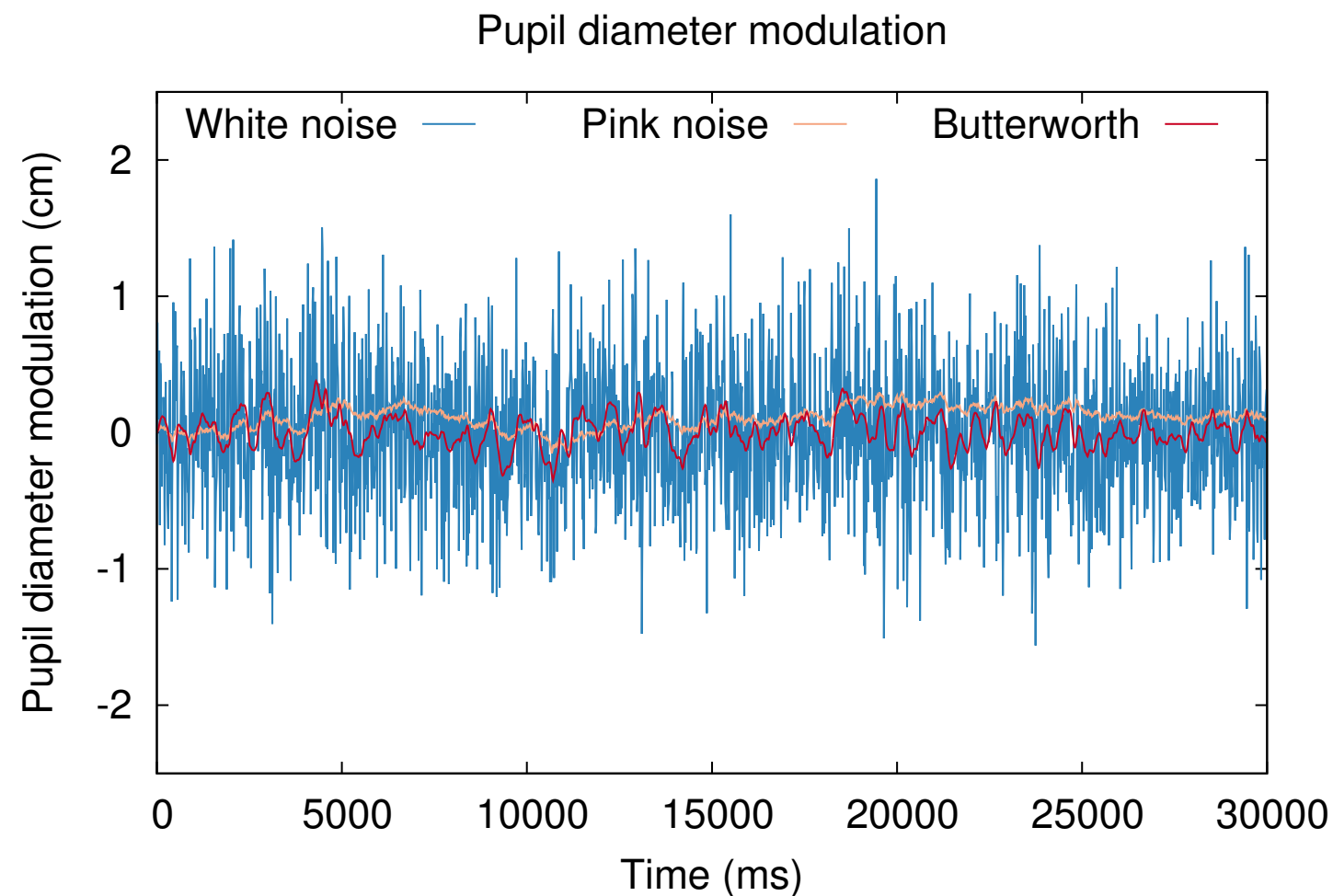
$$C(t) = \begin{cases} a - \left(\frac{t}{\mu}\right)^2, & t \leq \mu \\ b - e^{-c \log(t - \mu + 1)}, & \text{otherwise} \end{cases}$$



Modeling pupil diameter



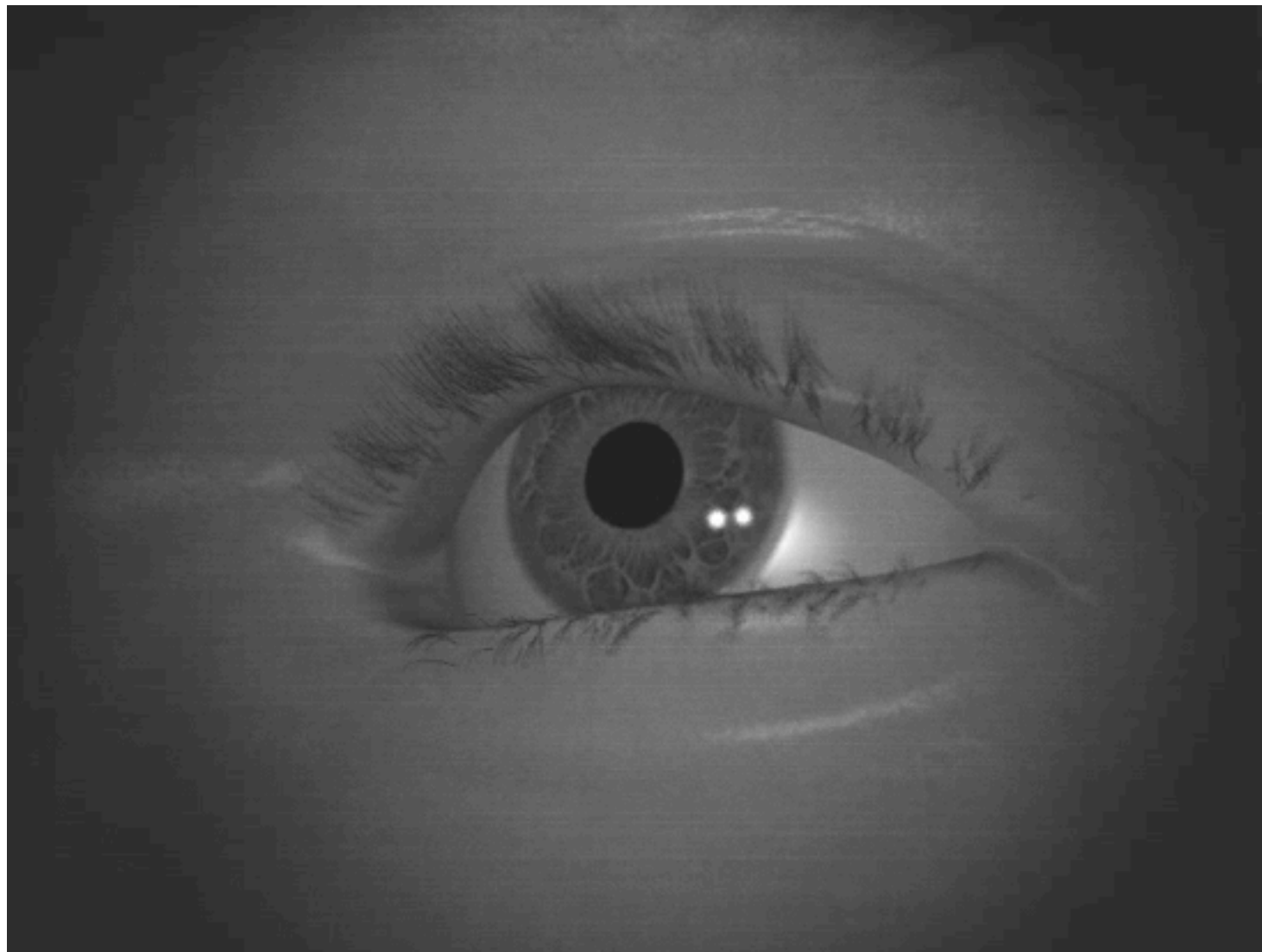
- Pupil diameter continually undergoes small fluctuations
 - even in steady illumination
 - major component is random noise in 0.05—0.3 Hz range
- Can once again use pink noise
- Result gives correlated noise rather than purely random (white noise)



3D rendering



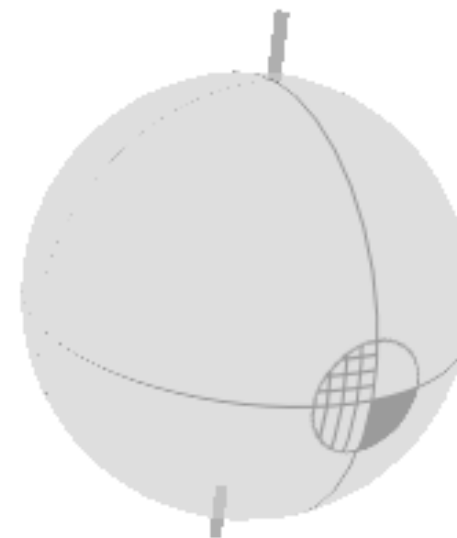
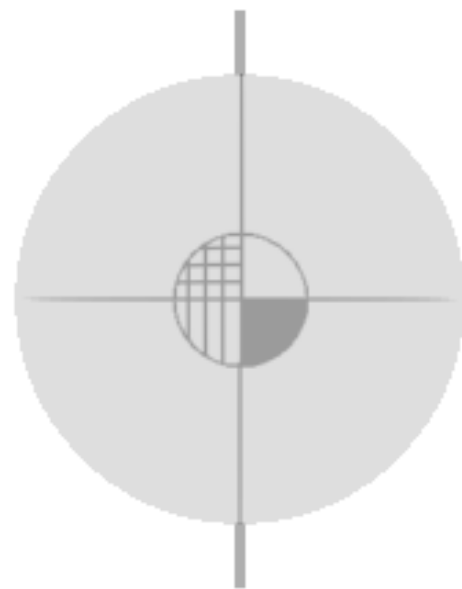
- Each frame takes 2:00 to render, use Palmetto again!
- Using 501 cores this time...custom “render farm”



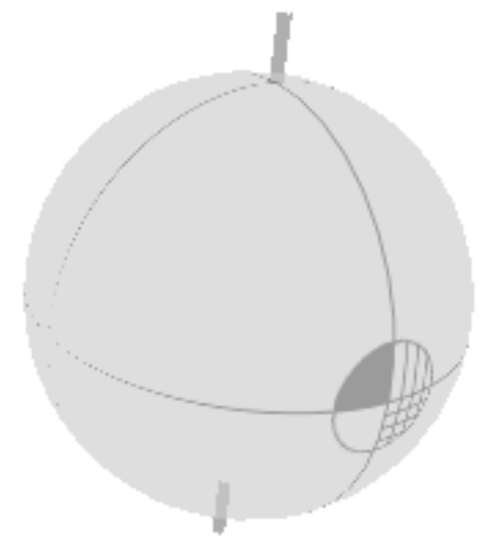


You're Invited: PhD Positions in Visual Computing

- Two open PhD positions at Clemson's Visual Computing
- Both funded (NSF grants):
 - Simulation and Mesh Generation (Josh Levine)
 - Gaze Modeling and Perception (Sophie Jörg & me)



CORRECT



IMPLAUSIBLE

Please RSVP!

sjoerg@clemson.edu



CLEMSON UNIVERSITY
EYE TRACKING LAB
SCHOOL OF COMPUTING

CLEMSON
UNIVERSITY

Q & A



- Thank you!
- Questions?

A few select references...



- Clark, P. J. and Evans, F. C. 1954. Distance to Nearest Neighbor as a Measure of Spatial Relationships in Populations, *Ecology* 35(4), 445—453.
- Duchowski, A. T., Price, M. M., Meyer, M., and Orero, P. 2012. Aggregate Gaze Visualization with Real-time Heatmaps, In *Proceedings of the Symposium on Eye Tracking Research and Applications (ETRA '12)*, ACM.
- Duchowski, A. T., Babu, S. V., Bertrand, J., and Krejtz, K. 2014. Gaze Analytics for Unity 3D Integration: Signal Filtering and Analysis, In *Proceedings of Eye Tracking for Spatial Research (ET4S)*, Sep.23, Vienna, Austria.
- Hollos, S., and Hollos, J. R., 2014. *Creating Noise*. Exstrom Laboratories, LLC, Longmont, CO, April.
- Hollos, S. and Hollos, J. R., 2014. *Recursive Digital Filters: A Concise Guide*. Exstrom Laboratories, LLC, Longmont, CO, April.

