Eye Tracking

- Calculate Fixation Points
- Visualisation
- Next Steps

Calculate Fixation Points

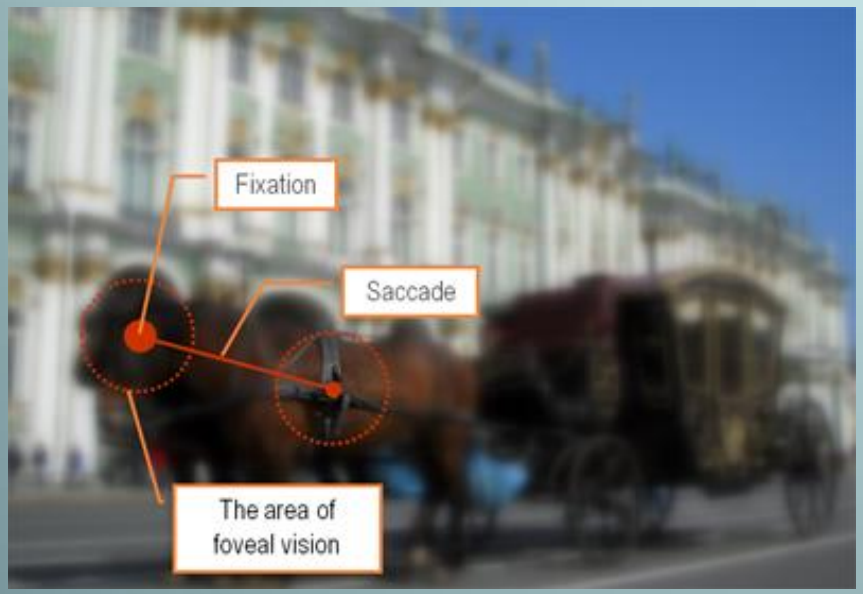
How to extract fixations from rare eye-tracking data?

- [1] Salvucci, Dario D., and Joseph H. Goldberg. "Identifying fixations and saccades in eye-tracking protocols." *Proceedings of the 2000 symposium on Eye tracking research & applications*. ACM, 2000.
- [2] Nyström, Marcus, and Kenneth Holmqvist. "An adaptive algorithm for fixation, saccade, and glissade detection in eyetracking data." *Behavior research methods* 42.1 (2010): 188-204.
- [3] Duchowski, Andrew. *Eye tracking methodology: Theory and practice*. Vol. 373. Springer Science & Business Media, 2007.
- …

➢ Implemented an easy proximity based algorithm:

- Drawback: last 2 points in Input are not computed
- Minimum duration time set to 200 ms

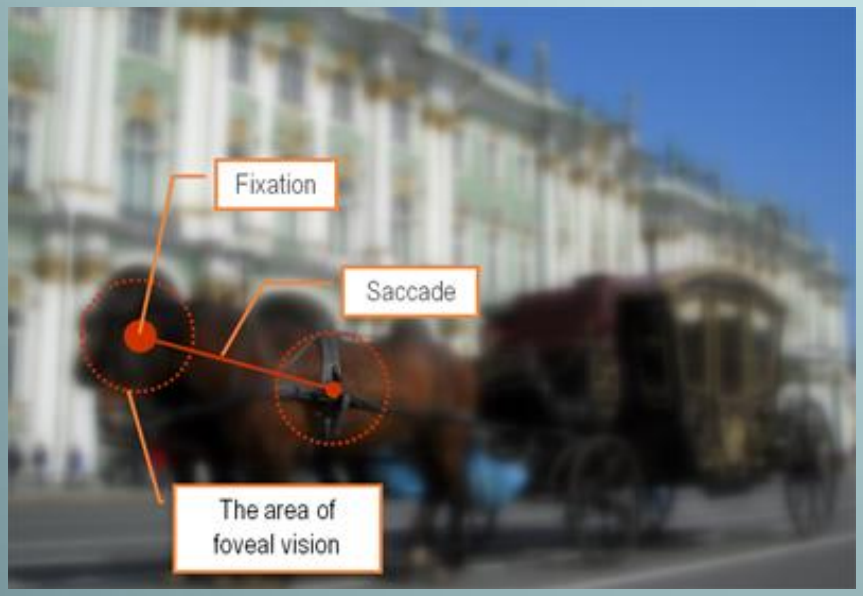
Calculate Fixation Points



http://eyetracking.ch/wissen/was-ist-eye-tracking/

- "**Fixations** are eye movements that stabilize the retina over a stationary object of interest."[3]
- The goal of eye movement signal analysis is to **characterize the signal** in terms of salient eye movements, i.e., **saccades and fixations** (and possibly smooth pursuits). Typically, the analysis task is to **locate regions** where the signal average changes abruptly indicating the end of a fixation and the onset of a saccade and then again assumes a stationary characteristic indicating the beginning of a new fixation.[3]

Calculate Fixation Points



http://eyetracking.ch/wissen/was-ist-eye-tracking/

Pseudo Code:

For all points:

If (Dist p1p2< $Dist_{max}$) {
    add Points to gazeList }
Else {
    If (fixation duration>Dur_{min}){
        add fixation to fixList}
    clear(gazeList);
}
return fixList;

Visualisation

- HTML & JavaScript
- Communication between Python and JavaScript:
  - create JSON-File with all relevant data:

```
var fixList =[ {
                "x":546.5245259602865,
                "y":325.5212673611111,
                "stop":0.5864257294097115,
                "duration":0.3437668068954735 6,
                "start":0.2426589225142379 3
              }, {…}]
```

Visualisation

- 2 Layer:
  - Background Youtube Video in autoplay
  - Foreground JS-Canvas with Circles at fixation points
  - 2 different circles: mean duration calculated -> decides the colour and size of the point
  - Clear canvas after fixation points

Next Steps

- Improve the fixation algorithm
- Optimize & extend the visualisation:
  - Design
  - Synchronisation
- Send the data as a input-stream
- Questions & Suggestions?