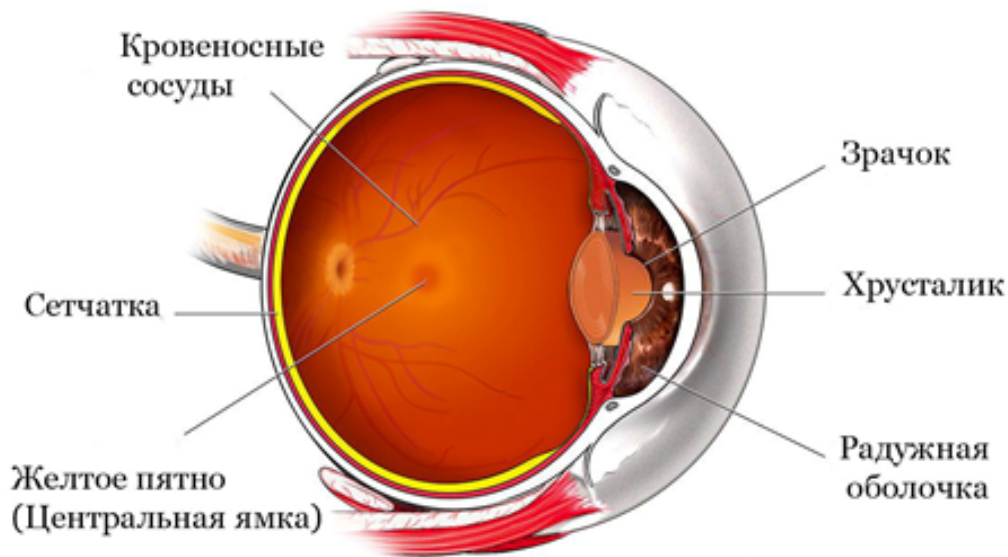


# Применение айтрекеров для юзабилити-исследований ПО в GNU/Linux

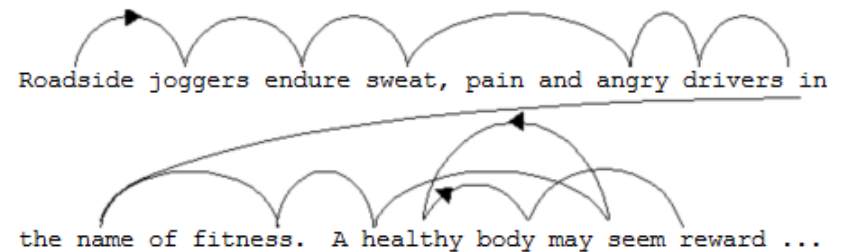
А.В. Дубицкий, Д.А. Костюк,  
А.А. Маркина, С.А. Фомин

# Как человек видит



Фоторецепторы сетчатки глаза делятся на два типа – колбочки и палочки. Основная масса колбочек сосредоточена в центральной части сетчатки, называемой жёлтым пятном или макулой.

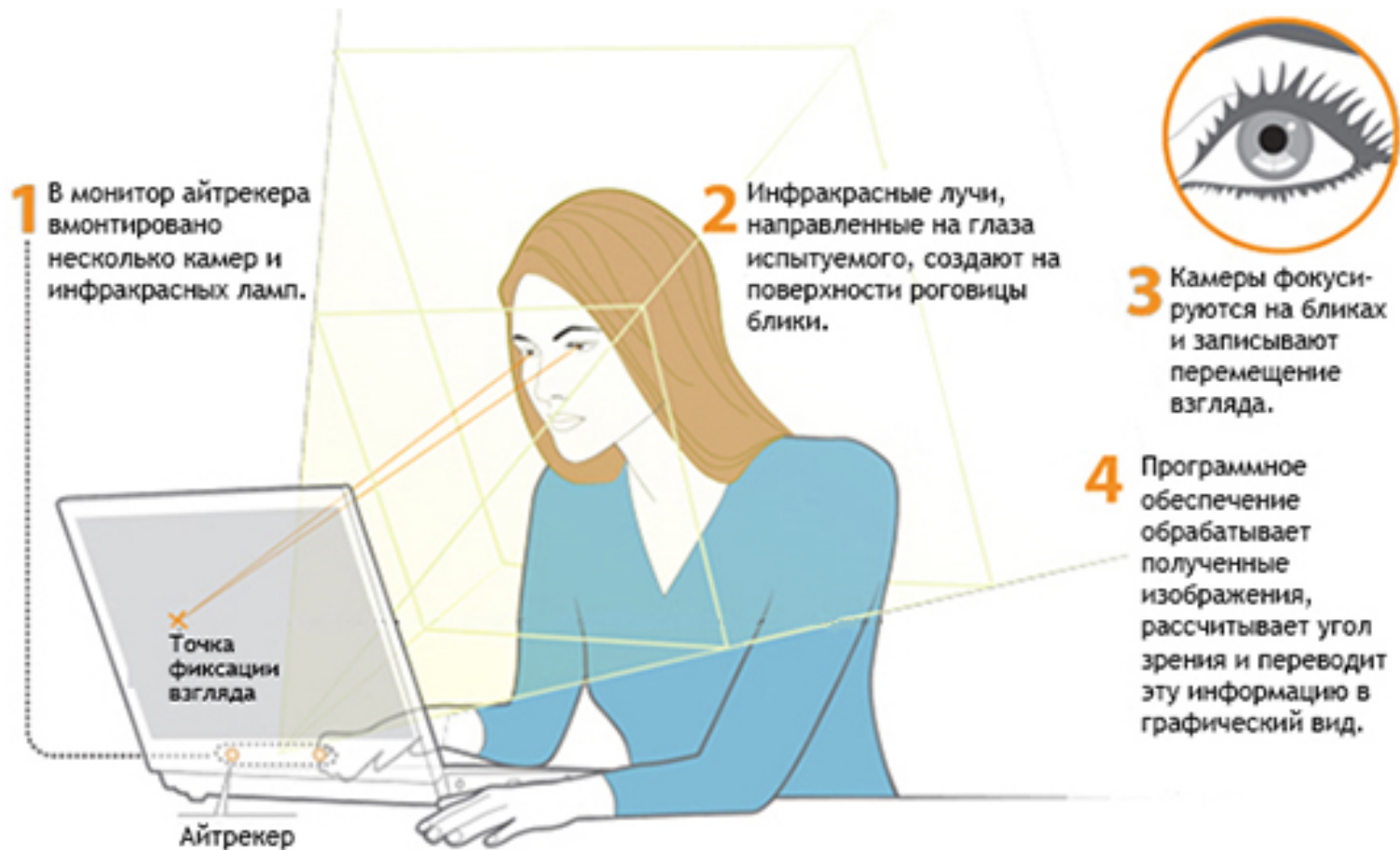
При работе с текстом за одну саккаду считывается 7 – 9 букв, а продолжительность фиксации составляет около 250 миллисекунд



# Применение в UX-исследованиях

- Причины UX проблем  
(заметность элементов, точки фокуса внимания, ментальная нагрузка и отвлечения)
- Особенности поведения пользователей  
(стратегии визуального поиска, паттерны чтения и сканирования)
- Сравнение решений

# Схема работы айтрекера



# Сравнение решений — крайние случаи

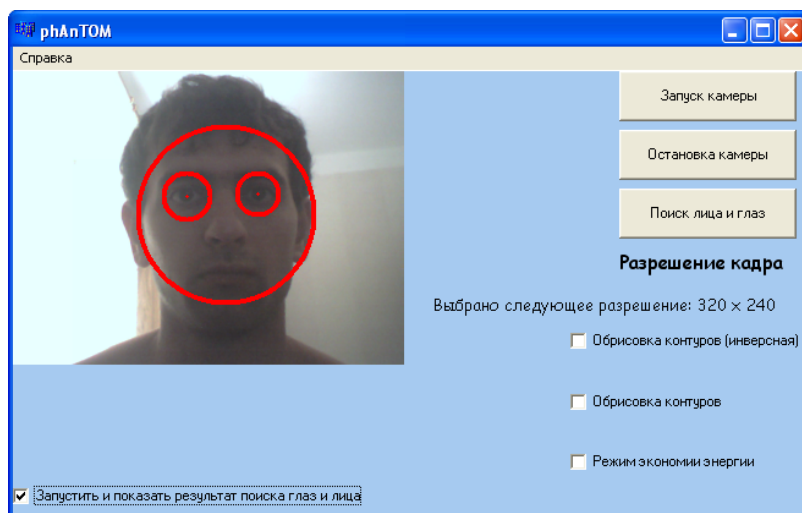
- Профессиональные айтрекеры:



Особенности на примере профессиональных решений от tobii:

- Носимые либо дистанционные (монтируются под экраном в поле зрения пользователя)
- Ценник от \$ 10 000
- Часто совместимы с Linux и MacOS
  - например, отдают данные по протоколу TCP/IP

- Программные решения на основе веб-камеры:



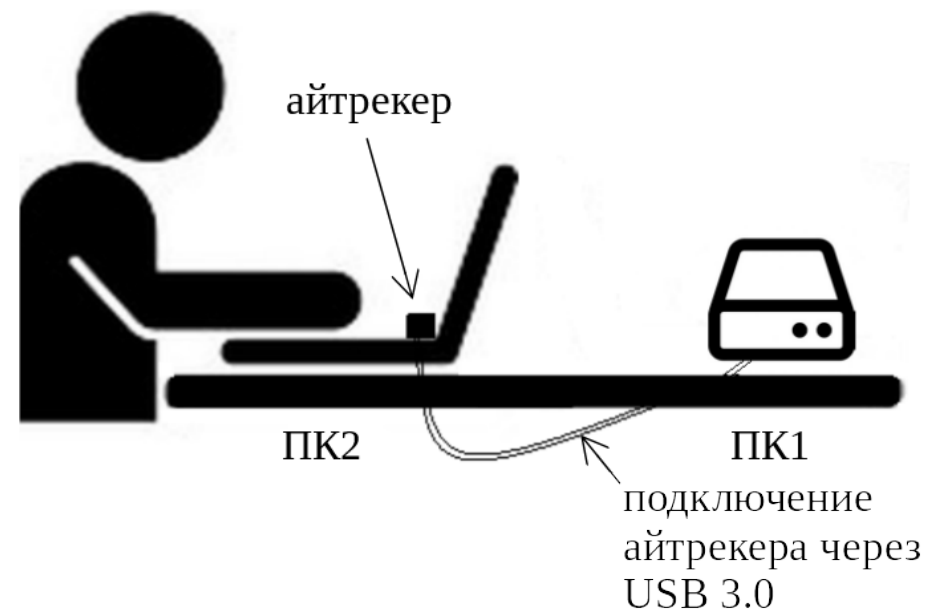
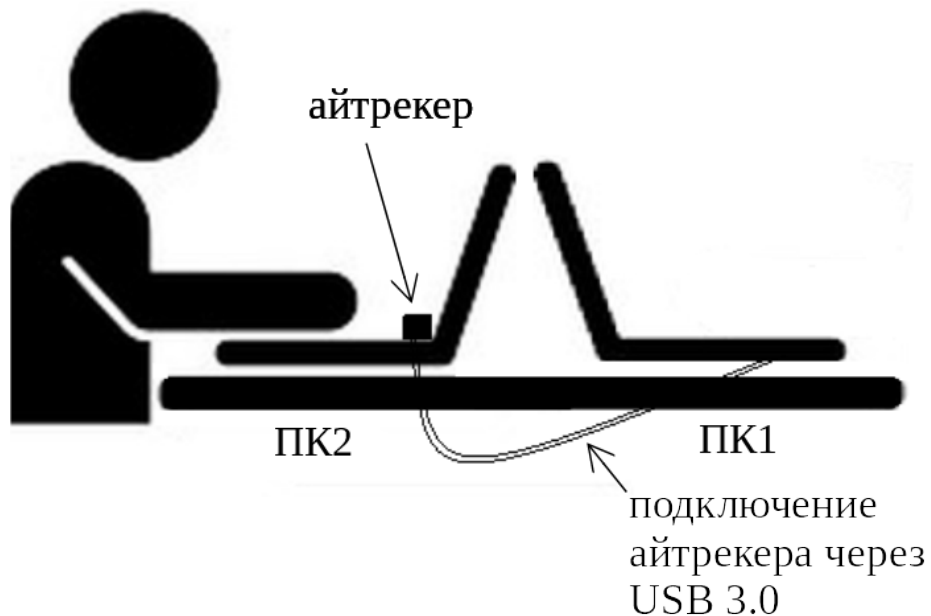
Ряд свободных программных проектов нацелен на позиционирование взгляда с помощью веб-камеры:

- eviacam, TrackEye, WebGazer.js, ...
- Написаны на C#, Delphi, JavaScript
- Обычно распознают лицо в видео-потоке с помощью библиотеки OpenCV (иногда — с помощью собственных алгоритмов)
- Позиционируют взгляд с разрешением, вдвое ниже разрешения веб-камеры, не отслеживают поворот головы

# Айтрекеры Tobii потребительского сегмента:



Из-за проблем с SDK (Linux-версия есть, но её нет)  
схемы подключения выглядят так:



# Получаемые данные

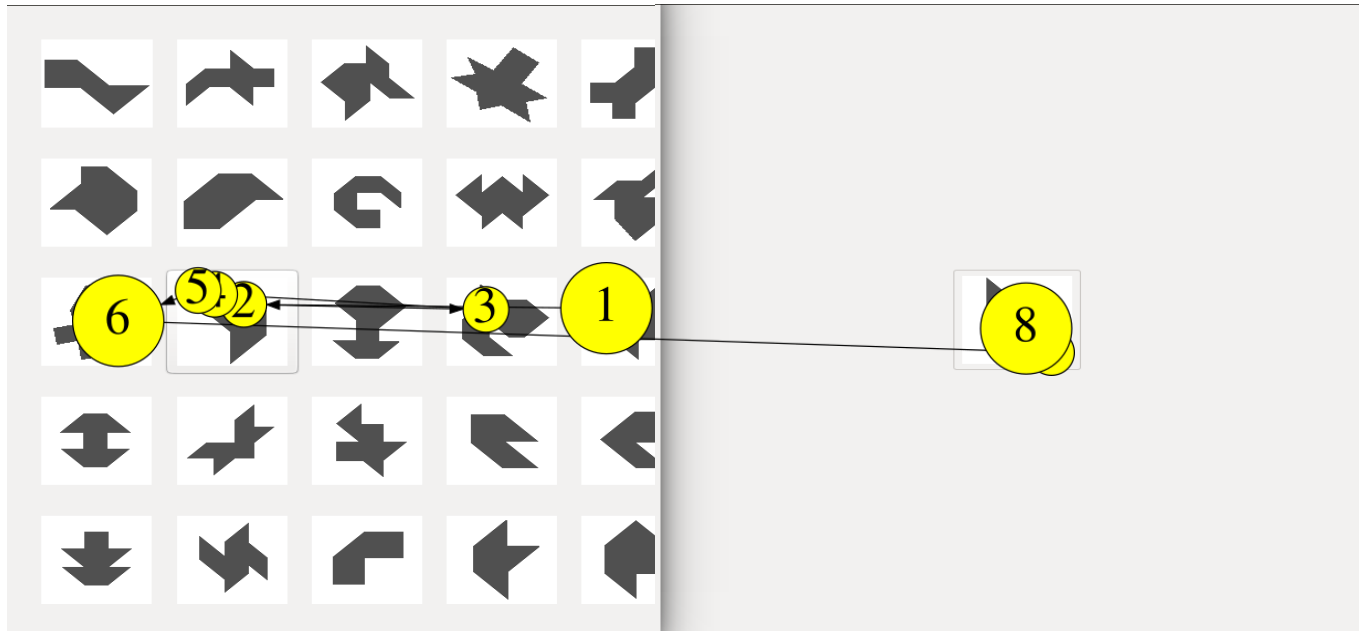
- Число фиксаций
- Длительность каждой фиксации
- Время до первой фиксации
- Длительность первой фиксации
- Общее число фиксаций
- Общее время фиксаций
- Скорость саккад
- Длина саккад
- Скорость просмотра
- Диаметр зрачка
- Сколько было фиксаций до посещения зоны интереса

# Визуализация результатов эксперимента

- Gaze plot - граф, отражающий последовательность фиксаций взгляда
  - Метка узла — его номер в последовательности фиксаций
  - Размер узла — длительность фиксации
- Heat map или тепловая карта
  - Показывает разной температурой цвета распределение плотности фиксаций по изображению наблюдаемого объекта
  - С учётом длительности фиксации или без
  - Разновидность - fog map — наоборот, размывает части изображения, на которые меньше смотрели



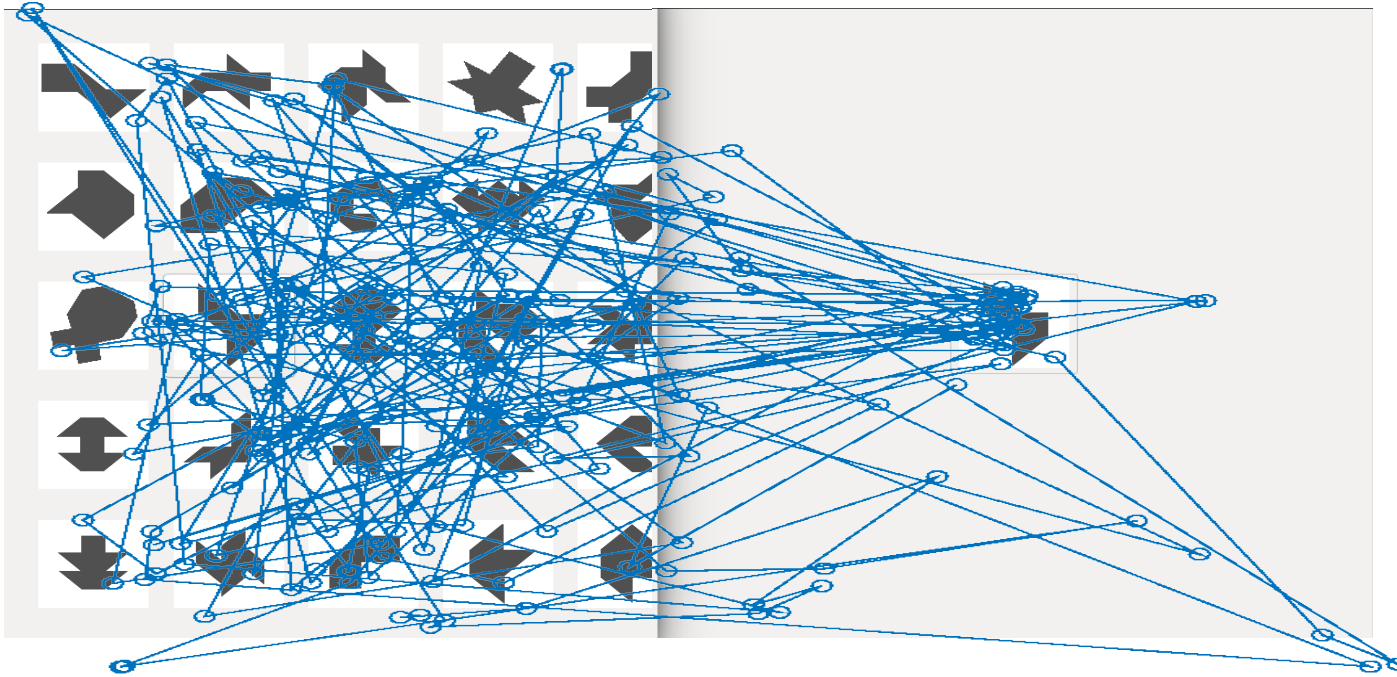
# Gaze Plot с помощью graphviz



```
digraph G {
  outputorder=edgesfirst
  node [shape=circle style=filled
  fillcolor=yellow fontsize=40
  fixedsize=true width=0.5]
  2 [pos = "137,375!"]
  3 [pos = "328,371!"]
  4 [pos = "114,383!"]
  5 [pos = "101,386!"]
  7 [pos = "774,337!"]
  node [width=1]
  1 [pos = "423,372!"]
  6 [pos = "38,362!"]
  8 [pos = "754,356!"]
  1 -> 2
  2 -> 3
```

- Применим для статического отображения при небольшом числе фиксации
  - однако большой лог айтрекера различим только в пошаговом режиме
- Зато несложно сделать скрипт, автоматически превращающий отметки времени в длительность и переводящий наборы координат айтрекера в формат graphviz

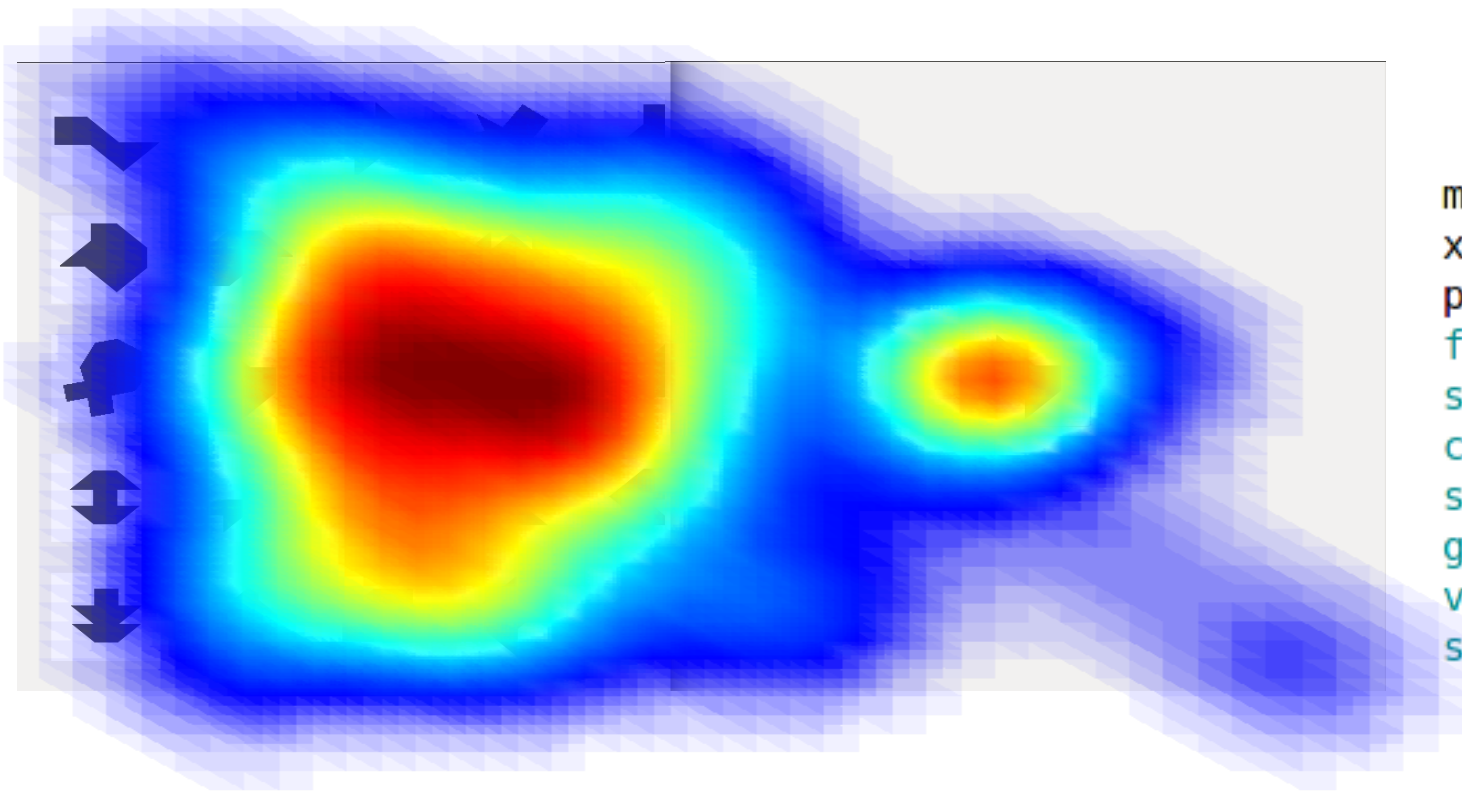
# С помощью Octave (простая визуализация)



- большой лог айтирекара превращается вот в это...
- поэтому при анализе протяженных тестов обычно используют тепловые карты

```
m=csvread('/tmp/sdf.txt');  
x=m(:,1); y=m(:,2); z=m(:,3);  
plot(x,y,"o-");
```

# Тепловая карта, построенная с помощью Octave



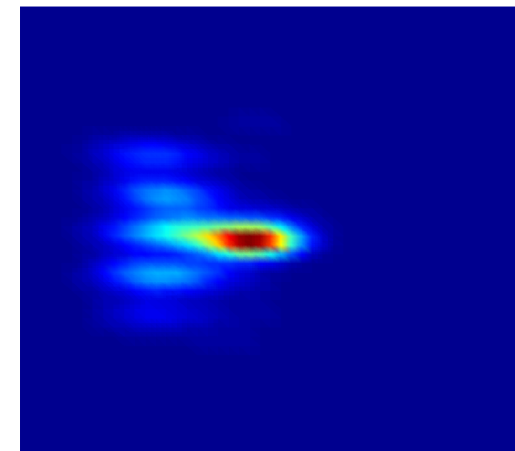
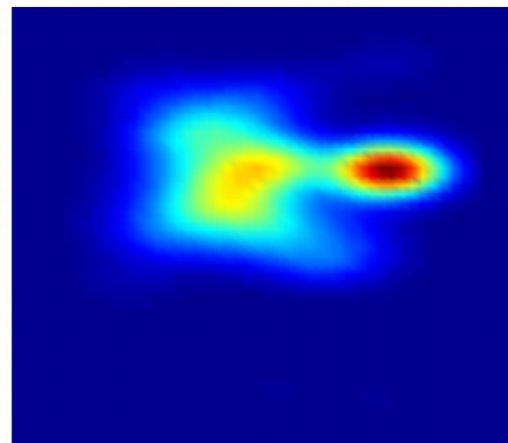
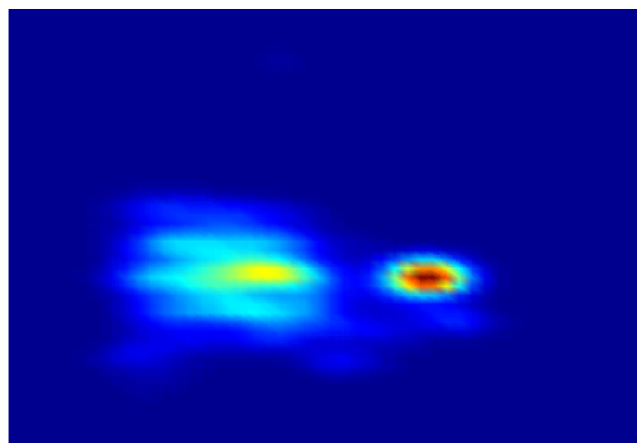
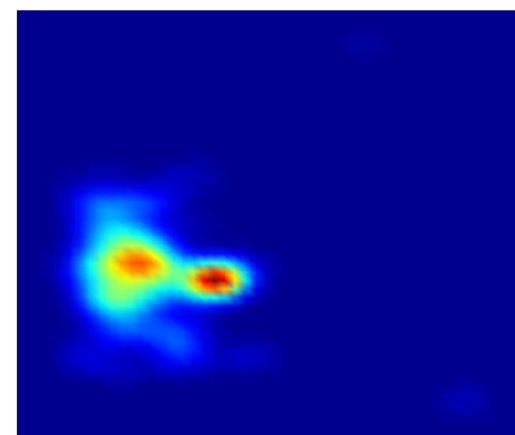
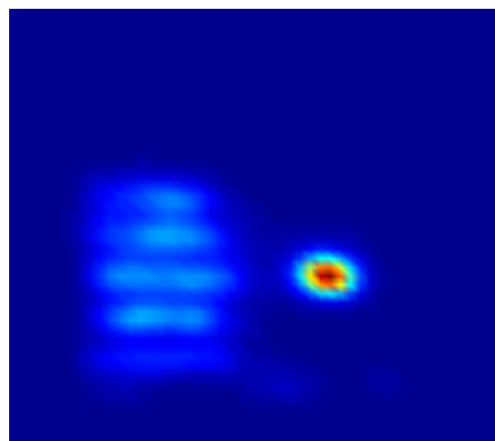
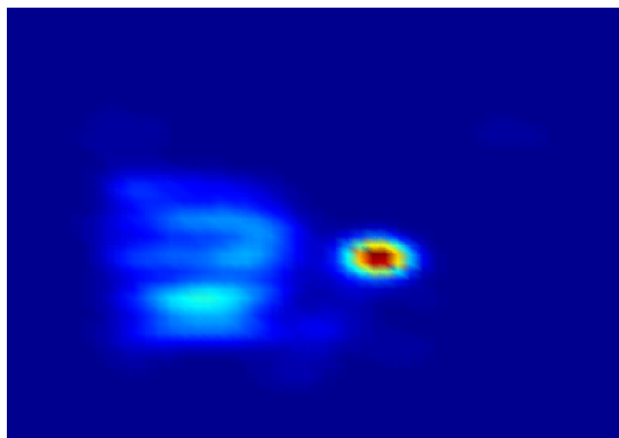
Генерация heat map:

```
m=csvread('sdf.csv');  
x=m(:,1); y=m(:,2);  
p= gkde2([x'; y']);  
figure(1);  
surf(p.x,p.y,p.pdf);  
colormap(jet);  
shading interp;  
grid off; axis off;  
view(2);|  
saveas (1,'sdf.png');
```

В примере использована черная магия, которая называется «бивариантная оценка плотности ядра» (или что-то в этом роде)

[https://github.com/aulloa/Auto\\_Heat\\_Mapper](https://github.com/aulloa/Auto_Heat_Mapper) — пример её использования, создававшийся для matlab — но в упрощённом виде оно работает в octave (см. код справа)

# Варианты одного задания, выполнявшиеся разными участниками эксперимента



Спасибо за внимание