## Learning to predict where humans look

For many applications in graphics, design, and human computer interaction, it is essential to understand where humans look in a scene. Where eye tracking devices are not a viable option, models of saliency can be used to predict fixation locations. Most saliency approaches are based on bottom-up computation that does not consider top-down image semantics and often does not match actual eye movements. To address this problem, we used eye tracking data of 15 viewers on 1003 images and use this database as training and testing examples to learn a model of saliency based on low, middle and high-level image features. This large database of eye tracking data is publicly available with the paper (Tilke Judd).

http://people.csail.mit.edu/tjudd/WherePeopleLook/index.html

### 1. Eye tracking database

We collected a large database of eye tracking data to allow large-scale quantitative analysis of fixation points and gaze paths and to provide ground truth data for saliency model research. The images, eye tracking data, and accompanying code in Matlab are all available.

In order to extract clusters of fixation locations and see the locations overlaid on the image, do the following:

>> showEyeData ([path to eye data for one user], [path to original images])

```
>> showEyeData('../DATA/hp', '../ALLSTIMULI')
```

You can also run

>> showEyeDataAcrossUsers([path to images], [number of fixations per person to show])

```
>> showEyeDataAcrossUsers('../ALLSTIMULI', 6)
```

This shows the eye tracking center of the eye tracking fixations for all users on a specific image. The fixations are color coded per user and indicate the order of fixations. Press space to continue to the next image.

You can see information about the fixations in 'Eye tracking database\DATA\hp':

i1000978947.DATA.eyeData

This is the raw x and y pixel locations of the eye for every sample that the eye tracker took. An eye tracker recorded their gaze path on a separate computer as they viewed each image at full resolution for 3 seconds (725 sample) separated by 1 second of viewing a gray screen. To separating bottom-up and top-down computation you should divide samples into two groups (First 1.5 S and second 1.5 S).

## 2. saliency model

To get the code for features to work, go to ('Our saliency model\JuddSaliencyModel\JuddSaliencyModel \README.txt') and install the ordered tools.

saliency.m is the code that loads in an image and returns a saliency map. It finds all the necessary features (find*.m files) and then loads the model.mat file to get the weights for combining all the features in a linear model.

As you see in saliency.m, there are 7 groups of FEATURES. You need to uncomment every feature except one; run the saliency.m to create different saliency maps.

What is the features that each FEATURES returns?

## 3. you want to compare saliency maps to fixations

Now we have 7 different saliency maps and 2 fixation maps for each example image. You should calculate ROC between saliency maps and fixations. So we have 2 ROC for each FEATURES that determines which FEATURES is the effect of bottom-up and witch is top-down computation.

For calculation of ROC, see the ('gbvs\gbvs\ readme') line 163.