AINT308: Problem with Saliency Code

Phil Culverhouse, James Rogers
CRNS, University of Plymouth
UK

The Itti & Koch model of saliency (See Fig. 1) employs saccadic eye control. The OWL has saccadic control because of the steerable eyes (see Rogers et al. 2017). In addition, the OWL has neck rotate so we have augmented the Itti & Koch model to possess an egocentric view of the OWL's world using the neck 160° pan capability. See PlymouthOWLRobot-attention-model-example.mkv video example in the Github archive (https://github.com/pculverhouse/plymouth-owl).

The Pan View window shows the individual saccade gazes overlaid on a world-view map that is egocentric to the OWL. It is not updated properly, as the OWL moves it gaze around its world driven by the saliency map. The default saliency map has one feature map encoded for low pass filtered image intensity. As can be seen from Fig.1 the saliency model has more feature maps contributing to the overall saliency analysis of the scene. Unfortunately, the more feature maps that are added to the software implementation, the slower the model runs. This has an impact on delays within the camera video stream.

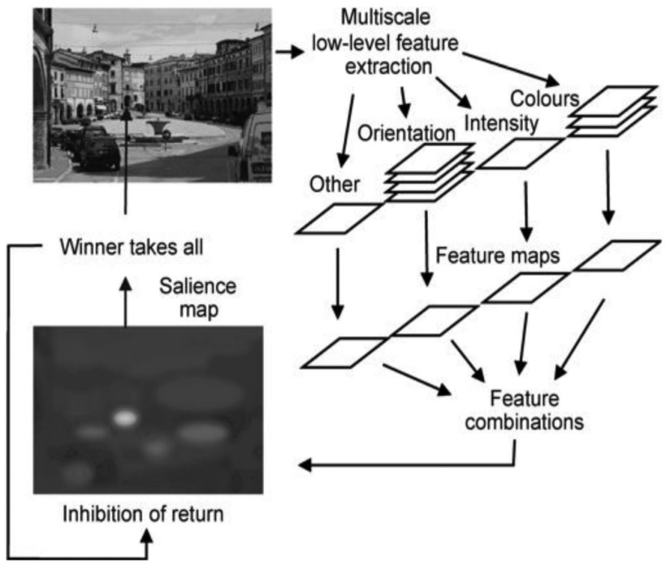

Figure 1. Schematic of Itti's salience model. Figure from Tatler et al. (2011); redrawn from Itti and Koch (2000).

We have observed an issue under Win10 where the video stream is buffered by the OS and read by OpenCV VideoCapture(). The issue is that the Salience code reads a video frame from the OWL video stream and processes it. It determines where to move the eyes to, moves the eyes, and takes another frame from the video stream.

But, the video stream is fully buffered by the OS, so as not to drop frames (the norm for applications like Netflix for example). But our application needs to flush the video buffer between frames processed, so as to be reading an up to date frame from the OWL and not an old one that holds a photo of somewhere the OWL used to be looking at a few seconds ago!

Unfortunately, there is no buffer flush function in VideoCapture devices. So we have to emulate the flush function. Check out lines 176-180 from salience.cpp, shown in Code Snippet 1.

```
     //============Capture Frame=========
    //for(int f=0;f<15;++f){
       if (!cap.read(Frame)){
         cout  << "Could not open the input video: " << source << endl;
            }
    //}
```

Code Snippet 1: reading from video stream

You will see that we have commented out a for loop around the read frame. Once you have a saliency map running, you will need to uncomment this loop control to allow the buffer to be emptied before you take the next photo.

You could replace cap.read() with cap.grab() to grab a set of frames from the buffer, and then just process the last one for colour etc. by calling cap.retrieve(), which is more efficient.

Only play with this adjustment once you are happy that your feature maps are complete. Since adding more maps, and other processing will slow the program down and cause the pseudo-flushing not to work fully.

References

Rogers J, Page-Bailey K and Culverhouse PF. (2017) The Plymouth OWL, a new open-source robot for stereo depth using vergence and disparity. Internal report 2017-01: Centre for Robotics & Neural Systems, University of Plymouth, 2017. https://www.plymouth.ac.uk/uploads/production/document/path/9/9273/OWL-paper-short.pdf

Tatler, Hayhoe, Land & Ballard (2011) Eye guidance in natural vision: Reinterpreting salience. J Vis. 2011; 11(5). DOI:10.1167/11.5.5.
(see https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3134223).