# iqr Tutorial 05-Camera-Visual classification

iqr tutorials will provide you with a practical introduction to using the neural simulation software and give you an insight into the principles of connectionist modeling. They are intended to be complementary to the detailed operation manual.
The home page of iqr is at iqr.sourceforge.net. Up to date information, documentation and tips and tricks ca be found in the iqr wiki (sourceforge.net/p/iqr/wiki/Home/).
The repository of iqr packages is here: sourceforge.net/projects/iqr/files/. iqr is open source software. You can browser the entire source code of iqr here: http://sourceforge.net/p/iqr/code/HEAD/tree/. Please contribute to iqr by reporting bug (sourceforge.net/p/iqr/bugs/) and requesting features (sourceforge.net/p/iqr/feature-requests/).

## Aims

- In this tutorial we will learn to discriminate a group input into two different classes. We will apply this operation to real time video in order to obtain a 2-color output.
- The result of this exercise is implemented in the file “classification.iqr”. However, the file is not complete. As an exercise, you have to set the missing parameters while reading the tutorial.

## Introduction

For the next tutorials we will use a camera and the iqr “video module” (please check the Tutorial “Camera-Basic interfacing” for more details). In general we can consider a camera as a matrix of sensors; each pixel gives the reading of a particular sensor in terms of amount of light; if we use a 160x120 camera output we are dealing with almost 20 thousand individual sensors.

When we use a sensor device, the first step is to check the range of its output signal. In iqr we can this by setting the output of the sensor module to a neuron group of linear thresholds and then check for the output in the space plot. The min and max values in the bar of the space plot will give us an idea of the approximate range.

Most information that we can get via sensors is expressed as numerical values in a range. For instance, the value of pixel in a monochrome image indicates the luminosity level in a region of the image. In some cases this may be encoded by an integer number between 0 and 255, thus providing 256 different possible values. In the case of the output of our video module the values are encoded as decimal numbers between 0 and 1, and to fulfill your curiosity, there are 256 different values as well (256 is an important number for computers).

In some situation though we may be interested in reducing all this amount of information, 256 or more different values, to just two: 0 and 1. And with 0 and 1 we may be encoding dark or clear, too close or far away, noisy or silent, etc.

This is one of the easiest operation that we can apply to the data given by a sensor, it consists in setting a threshold and consider that only values above that threshold correspond to the “active” class or 1.

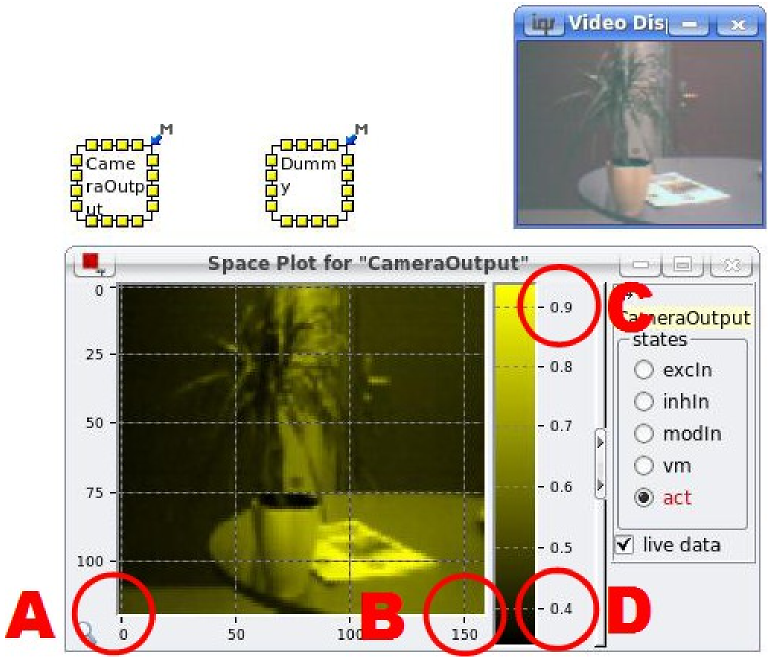

*Figure 1 Output of the camera seen as a "space plot" of the group "CameraOutput". The group has approx. 100 rows (A) and approx. 150 columns (B) (the true values are 120 and 160). From the upper (C) and lower (D) values of the "color bar" we see that the maximum value mapped is between 0.9 and 1.0 and the minimum between 0.3 and 0.4*

## *Example: Discrimination between classes of spots in an image*

Let's think of a task where we have to search for a white object in an image, for instance a football or a (pale) face. If we are provided with a monochrome camera, a first step towards finding a white ball could be to discriminate clear spots form darker ones. We can do this by setting a threshold.

Quite likely, the main object that your camera will find in its visual field is your face. With such working material one of the games you can play is to search for the threshold value that better isolates your face from the background. Based on luminance (gray level), this will be an easy task if, for instance, your skin color is clearer than the background (if your skin color is darker that the background you have to jump to the "Negative Image" exercise).

## Implementation

### Processes

To implement this operation in iqr we only need one process with its type set to "video module". We have to select the "output" of the camera to monochrome ("Process properties", tab "output", check box "monochrome").

We will need two groups "CameraOutput" and "dummy". We will send to "camera output" the information we want to process and to "dummy" the one we discard. To see that, check the properties of the process "camera" in the tab "module to groups". See that the "Grayscale" output is sent to "CameraOutput".

### Groups

The iqr implementation of this tutorial is quite minimalistic, as said above we need two groups, but only one of them ("CameraOutput") is involved in the classification (Figure 1).

Both groups size has to match the camera resolution: 160x120. This is set in the group “properties”, section “topology”, for the “topologyRect” (press the “edit" button) the values 160 and 120. Whenever we refer to a groups "size" we refer to this "width" and "height". For the group "dummy" we do not care of anything else.

The group “cameraOutput” does the job. First of all, recall that we want to give only two kinds of outputs: 0 and 1. The neuron type “Integrate and Fire" does it, when it "spikes" it outputs "1". When it is not spiking it outputs 0. Therefore all we have to do is set the appropriate properties:

- “persistence” and “probability”: unless otherwise stated, is safe to set these values to “0" and “1" in most of the cases (this holds for “Integrate and Fire”, “Linear threshold” and “Sigmoid neurons”.
- "membrane potential reset": if "persistence" is set to 0 this parameter has no effect, but again, set this value to "0" unless you have good reasons to do otherwise.
- "threshold": this is the only parameter that matters for this exercise. By setting its value we decide the extent of the "active are" that we will see in the image.

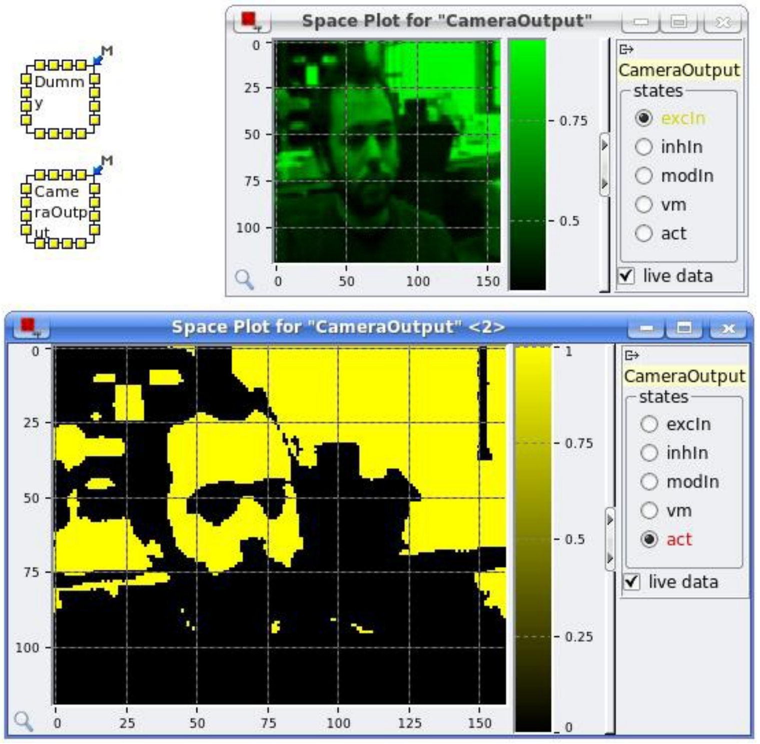

*Figure 2: Face of the bearded author of this exercise with a messy office landscape as background. The output of the camera appears in green in the top right. The yellow space plot shows the result of the thresholding process.*

As you will see, iqr does a good job in processing the data at real time. Keep in mind that you CAN modify parameter values while an iqr model is running, you can not modify the neuron types, connections, etc. Changes to parameters are effective once you press the apply button. In this exercise you can modify the "threshold" value up and down until you get your favorite output. Test the «0» and «1» cases.

### *Links to other domains*

#### Mathematics

In mathematics the operation performed by this module is known as Heaviside function.

#### Photoshop

If you ever worked with an image processing software, like Adobe Photoshop, you can find this manipulation in the program. Indeed, in Photoshop its name "threshold" and it does exactly the same as our exercise. The difference is that what you do in Photoshop for one image, in iqr  you do it to a stream of images in real time.

## Exercises

- Complete the iqr module with the missing parameters. Report them.
- Change the Integrate and Fire neuron to a Linear threshold neuron ("persistence" and "probability" set to 0 and 1 respectively). Set the threshold to the same value you used for the “Integrate-and-Fire” neuron. Describe the result.