Interactive CEDAR tutorial—Part 1

Mathis Richter

January 9, 2019

1 Detection of simulated input

Create a two-dimensional field (`NeuralField`). This field may represent a two-dimensional surface, like a table top. Peaks in the field represent objects on that surface.

The field receives three inputs, which we simulate, for now, by 2D Gaussian functions (`GaussInput`) at different positions.

Tune the field so that it forms a stabilized peak at every input location.

2 Camera input & field regimes

We will now replace the simulated input with input from a live camera. Use a `NetReader` step to read activation from the network on port "`caren/activation`". Run that activation through a `StaticGain` step, which enables you to vary the overall strength of the activation; then connect the output of that step to the field.

First, tune the field to create a peak for every object in the scene. Once that works, try (slowly) moving the objects in the scene and observe whether the peaks follow the moving input.

Second, create a copy of the field (`Ctrl+D`) which then also receives input from the camera. Tune this field to have working memory about objects in the scene. That is, once peaks have formed, you should be able to quickly remove an object without the corresponding peak decaying. Once that works, try again to slowly move an object in the scene. The working memory peak should follow the moving input.

Third, create another copy of the field which then also receives input from the camera. Tune this field to make a selection decision, that is, it should only form a peak at a single position, even when multiple objects are in the scene. Make sure that the field is not in a working memory regime, that is, the peak should disappear when you remove the object from the scene. Play with adding and removing objects to and from the scene. At the position of which object does the peak form? Is it always the same object? Why do you think that is? Can you observe hysteresis in the field? If you are wondering, at this point, how activation is generated from the camera image, you are asking the right question.

3 Image preprocessing & 3D fields

We will now have a look at the preprocessing of the camera image. Create another `NetReader` step, this time with the port “`caren/camera`”. This will give you the raw camera image, scaled down to a resolution of 30x40 pixels.

Create a sequence of image processing steps that creates three-dimensional activation from the images. First, transform the color space from RGB (red, green, blue) to HSV (hue, saturation, value) using the `ColorConversion` step. What does that do? What is the HSV color space? Split the image into its three channels, hue, saturation, and value (`ChannelSplit`). Next, add a `RateMatrixToSpaceCode` step and feed the hue-channel into it as its first input (bin map). As its second input (values), use the saturation-channel, but apply a `Threshold` step on it to filter out weakly saturated areas. In the parameters of the `RateMatrixToSpaceCode` step, set the *number of bins* to 8 and the *upper limit* to 255. Try to understand what exactly that step does. Making a drawing and talking to your partner usually helps.

Feed the output of that sequence of steps into a `StaticGain` step and from there into a three-dimensional field (size of the field: 30x40x8). Tune the field so that it creates a peak for every object in the scene. Have a look at the plot of this field and make sure you understand it! Can you read off the spatial position and the color of the different objects in the scene from the activation values?

4 Projections

We will now tune the three-dimensional field to make a selection decision for an object of a certain color. For this, we first need a one-dimensional field (the color field; size 8), which spans the color-dimension. This color field will receive a Gaussian input (`GaussInput`) and create a peak based on that input. Choose the position of the Gaussian input so that it specifies a color you would like the model to pick. Now, using a `Projection` step and a `StaticGain` step, connect the color field to the three-dimensional field. When this is working, you should only see a thin layer of the three-dimensional field being affected by input from the color field. Why is that? Why is the thin layer of increased activation at this position in the three-dimensional field and not somewhere else? Make sure you understand exactly how the two fields are connected and what the `Projection` step does! Tune the strength of the input and the parameters of the three-dimensional field so that it only creates a peak when input from the camera overlaps with input from the color field. Play with the Gaussian input to the color field and select different objects in the scene.

Create a new two-dimensional field (size 30x40) and, using both a `Projection` step and a `StaticGain` step, connect the output of the three-dimensional field to the new two-dimensional field. For the projection, choose the mode `Maximum`. Set up the connection strength so that there is a peak in the two-dimensional field whenever there is a peak in the three-dimensional field. What does the `Projection` step do in this case?

The two-dimensional field can be interpreted as the spatial attention of the small model we created. You could also create a peak here from a Gaussian input and select certain positions in the scene. The one-dimensional field can be interpreted as a feature attention (color) of the model. The three-dimensional field binds the features of color and space together.