Tutorial with optional exercises

This tutorial is designed to get you quickly running with Canupo. The exercises are all optional but will teach you useful tricks.

Step 1: Download and unpack CANUPO, then enter the directory "tutorial". You shall find the following files:

`floor.xyz`	This is the floor sample shown in the overview.
`vegetation.xyz`	This is the vegetation sample shown in the overview.
`scene.xyz`	A scene extract to test the classification.

They are simple text files containing x,y,z coordinates on each line, easy to work with. We recommend you to open them with the very useful CloudCompare free software that can be downloaded at http://www.danielgm.net/cc/.

Exercise: Extract another floor and vegetation sample from the scene using the scissors tool from CloudCompare.

Step 2: Choose a relevant set of scales. Think at how 1D, 2D or 3D the object looks like at various scales. In this case, above 20cm, the vegetation already looks like a big 3D ball, while the floor looks flat. We ideally want to use low scales for refining the classification (ex: detecting a patch of floor below a vegetation branch), but the floor is rippled so getting too low (<5cm) may introduce spurious 3D components. All this is approximate as it should be, rely on your intuition and do not worry, the classifiers are robust enough to small variations.

Now, extract the floor and vegetation multiscale features using canupo. Open a terminal (Unix) or dos command line (Windows) in the tutorial directory, and type the following two lines (shall work on both systems):

```
../canupo 0.04:0.02:0.2 : floor.xyz floor.xyz floor.msc
../canupo 0.04:0.02:0.2 : vegetation.xyz vegetation.xyz vegetation.msc
```

This computes the multiscale features from 4cm to 20cm every 2cm on both samples. To understand why names are duplicated type `../canupo` without arguments to see help, and read the article. This is related to core points, a reduced set of reference points at which to perform the computations when the data is too large. Here we compute the features on each point in the sample.

Exercise 1: Run the `../resample` tool without arguments for help. Then generate a set of core points for the scene.xyz file, using a spatial subsampling of 4 cm. Once this is done run the `../canupo` command on the scene, using its core points for faster computations (compare to the time used without core points).

Exercise 2: Run the `../density` tool without arguments for help. Then generate density plots for the msc files that you generated. Look at how the vegetation and floor look like at different scales

Step 3: Build a classifier from the samples with the following command:

```
../suggest_classifier_lda proposal.svg : vegetation.msc - floor.msc
```

Type `../suggest_classifier_lda` without arguments for help on the syntax. Basically, you are telling the program to write a classifier in the file `proposal.svg` that shall separate the two classes with only one sample in each class (one of vegetation, one of floor). Look at the overview sheet in the previous section for a snapshot of SVG file. You shall notice that two blobs are separated by a line. The blob on the left represents all the floor points, the one on the right all the

vegetation points. You can now verify that the blobs are well separated. Read the article to understand how the transformation was performed.

Exercise 1: The SVG files are vector graphics file (and they are also text files). We recommend you to open the `proposal.svg` *file with the excellent Inkscape free software that can be downloaded at* http://www.inkscape.org/ .

Exercise 2: If you generated the multiscale features for the whole scene in Step 2, then use them as unlabelled information for improving the classifier. Refer to the help provided by `../suggest_classifier_lda` *without argument.Compare the resulting SVG files.*

Exercise 3: Run `../suggest_classifier_svm` *without arguments for help. Then run it instead of the LDA classifier in the example. Change the SVM search grid cross-validation size (the N parameter) and see how this affects the quality of the generated classifier. In practice the LDA classifier works as well if not better than the SVM and it is much faster.*

Step 4: Validate the classifier

```
../validate_classifier proposal.svg classifier.prm
```

This step simply means that you are happy with the default classifier.

Exercise: Edit the decision boundary path in the SVG file with Inkscape before validating it. For example by adding and moving a few points in the path. You can easily obtain powerful non-linear classifiers this way. This is also convenient if you want to favor one class over the other, for example here you may move the decision boundary away from the vegetation sample. Not really useful here, but in situations where the two blobs overlap, this is an easy way to make unbalanced classifiers (ex: you may want to be sure to remove all the vegetation even if you loose a bit more of floor at each vegetation patch base).

Step 5: Classify the scene. First, generate the multiscale representation for the whole scene if you did not do it already as an exercise in Step 2. Note that you can read the list of scales directly from the classifier file:

```
../canupo classifier.prm : scene.xyz scene.xyz scene.msc
```

Then classify the whole scene:

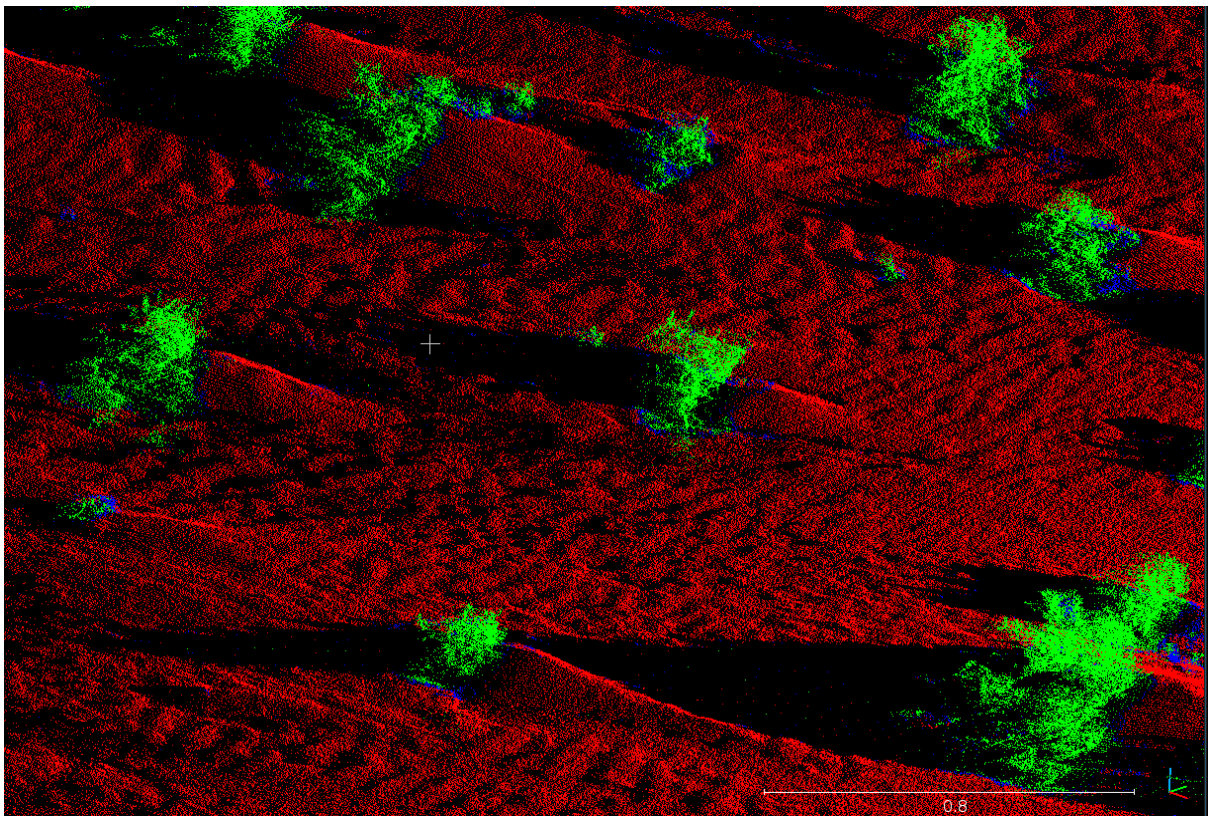
```
../classify classifier.prm scene.xyz scene.msc result.xyz
```

Open the `result.xyz` file with CloudCompare and specify in the open dialog that the fourth column is a “scalar>0” field. This will give each class a distinct color.

Exercise 1: Add a 0.95 confidence threshold, simply by adding 0.95 at the end of the classify command line above. Open the result file again. You shall notice that a third color was added for regions that cannot be classified with more than 95% confidence. Note how these are precisely at the base of the vegetation patches, and also how even small patches within riddles are correctly recognized.

Exercise 2: Use the scene core points from Step 2, excercise 1. Compare the computation time, but also zoom on a classified vegetation patch in CloudCompare in order to understand what tradoff was made: classification of each scene point is now assigned to that of the nearest core point.

Exercise 3: Use the `../filter` *utility in order to remove all the vegetation from the classified scene, using a condition on the fourth column that contain the class of each point.*



An extract of the example scene in this tutorial. The vegetation is displayed in green, the soil in red, zones for which the classification confidence is less than 95% are shown in blue. Note the riddles and the downstream sediment accumulation. This is a snapshot using Cloud Compare of the result you shall obtain at Step 5, Excercise 1.