OBJECT-BASED IMAGE ANALYSIS - FINAL EXAMINATION

Written examination for the OBIA online course. It comprises some practical assignments to be conducted in the eCognition software (full version in GI_Lab or trial (version 9.1, last version which allows to save projects)) and questions about the results.

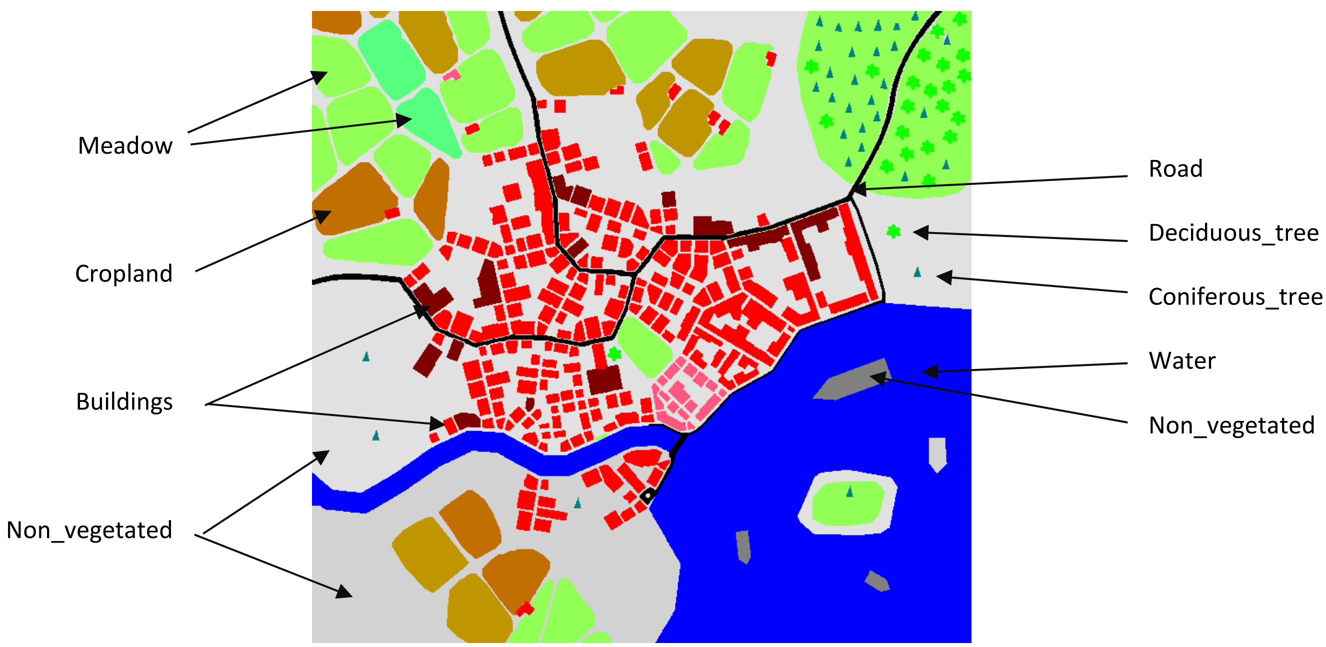
Submission: *To complete the practical part (🖱) of the exam, create an eCognition project file *.dpr where requested. Results of Question 2 should be written into a text document (PDF or *.doc). Upload the results when you are finished. Results are automatically checked in Blackboard for plagiarism; please make sure to cite your sources.*

Note: Do not forget to frequently save your work as well as intermediate results (maybe even giving them different file names). This way you always have a backup in case the program crashes!

🖱 QUESTION 1 (40 POINTS)

OBJECT-BASED CLASSIFICATION

Open **Artifical_landscape_UTM_01_2025.tif**. The image is neither an aerial image nor a satellite image - it is an artificial landscape (UTM projection, 0.2m spatial resolution, RGB bands). On first view, you can distinguish different potential classes like water, buildings, streets etc. Perform an object-based image analysis and **classify the image using the following eight classes (use in your project the class names as indicated in the screenshot below):**



Start with a first segmentation using adequate parameters (test different scale parameters), select representative samples for each class (since the colours per class are not much varying, a few samples will be enough) and perform a sample-based (supervised) image classification. Mind the right choice of features for the computation of the feature space; **select from different classification algorithms (e.g., SVM or Random Trees etc) and classify the image.**

[10 points]

You will maybe not be able to separate some meadow and deciduous tree objects, due to the similar colour. You can correct that in the second part of the exam.
If you are satisfied with the result, save the project under the following name:

Your_surname_sample_based.dpr

<u>Knowledge-based classification: Based on this first classification,</u> you should now conduct a rule-based classification to differentiate / aggregate the initial classes using (your) expert knowledge. The following classes should be differentiated (in brackets, some hints how to define the classes). It is recommended to use the same order in defining the classes, since some of them are build-upon each other. If some **deciduous trees and some of the meadow objects could not be separated** in the initial classification, try to solve the problem by e.g., using an area threshold first.
It is also recommended, that you merge (dissolve) all objects belonging to the same class and share a border (use the "merge region" algorithm for that with the parameter "merge: by class" in eCognition 10.x, in the trial version 9.1 the "merge region" algorithm must be executed for each class selected in the class filter).
Additional classes should be derived as follows:

[10 points]

- **Forested_Grassland** (meadow with more than one tree in it)
- **River** (water with a different shape than a lake)
- **Lake** (water with a different shape than a river)
- **Urban_tree** (trees close to a village, i.e. in close distance to more than one building)
- **Farmhouse** (buildings **directly connected** to grassland or cropland – try to implement this in one algorithm chaining the two conditions with "OR")
- **Ships** (Non_vegetated class with a relative border to water/lake of 100%)
- **Island** (bare soil, trees and grassland, surrounded by water. Have a look at the algorithm "find enclosed by class" and its documentation to solve this problem).
- **Freighter** (the largest of the three ships should be classified as a freighter– try to find a way to make the rule generic and without absolute values. There are several solutions possible to find the largest object in a domain, either with a specific algorithm or with a specific feature).

Try to **group similar classes** (group tab of the class hierarchy) into self-defined main classes where it makes sense (e.g., Lake and River can be grouped in a parent class "Water" etc.).

<u>Possible approach</u>: You can copy the level above the existing one and work there or you can execute the knowledge-based rules in the same level. To use most of the object-related parameters make sure, that you merge objects of the same class first, if your segmentation did oversegment the objects.
Now you have to look for *object-features*, which could be used to determine the semantic classes (e.g., class related features like "Rel. border to class xy", object-shape related ones etc.). Use for example the "assign class" algorithm in your rule-set to apply the different object-features. Alternatively, you can also make use of fuzzy "membership functions" of the single classes to define their properties, followed by the use of the "hierarchical classification" algorithm in the rule-set.

<u>Feel free to add additional classes, if necessary</u>. **<u>Avoid as much as possible to use fixed thresholds for the classification</u>**, e.g. fixed thresholds of colours are not meaningful and not transferable in this artificial image.

Hierarchical classification

[10 points]

Create a new level above the existing one(s). **Use a very large-scale parameter**, so that the trees and the surrounding grassland are within one object. Test several scape parameters and compactness settings. Derive on this level the following classes by using hierarchical relationships between the highest level and the objects classified on the level below:

- **Coniferous_forest** (combination of single trees and the surrounding grassland area - coniferous trees are dominating, e.g. number of coniferous trees > number of deciduous trees in the lower level. Here you should use class related feature across the image object hierarchy)
- **Deciduous_forest** (combination of single trees and the surrounding grassland area - deciduous trees are dominating. Here you should use class related feature across the image object hierarchy)

[10 points]

The work process must be **comprehensive and repeatable**. Thus, use the *Process Tree* that displays the single processes in a reasonable manner, i.e., keep all processes in the *Process Tree*, which were used for the classification and in the correct order of usage. **The process tree should be in procedural order to be executable in one pass.** If you are satisfied with the result, save the project under the following name:

Your_surname_expert_based.dpr.

When done, add both eCognition projects (*.dpr) to your zip-file.

Note: *In addition to the exercises and the material available in the Blackboard course, you should consult the eCognition User Guide as well as the eCognition Reference book, if needed (both accessible via the software interface > help menu). In the Reference book, you will find detailed description of features, algorithms and processes.*

+ QUESTION 2 (15 POINTS [5 + 5 + 5])

OBJECT STATISTICS

[15 points]

a) Describe shortly (3-5 sentences), why most of the classes of the knowledge-based classification cannot be addressed in a pixel-based approach, **give three specific examples of your solution.**

b) Select the appropriate ***Feature*** in your final project and figure out the length (**in map coordinates - not in pixels, check the context menu for the feature to change the units**) for the class: **Freighter**

Select the appropriate ***Object Feature*** in your final project and figure out the ***Elliptic fit*** values (geometry feature) for the classes **River** and **Lake.** Describe briefly how the **Elliptic fit** is calculated and what the meaning of the values is. Consult the eCognition User Guide for a more comprehensive understanding. Write down the values in your text document and explain the value differences between the two objects. In which case the value gets 0?

c) Create a customized feature reflecting the following formula for an $REDNESS$ **index** in your final project:

$$REDNESS = \frac{(Mean\ Layer\ 1 - Mean\ Layer\ 2)}{(Mean\ Layer\ 1 + Mean\ Layer\ 2)}$$

- What threshold of the $REDNESS$ index could be used to differentiate between buildings and the other classes/objects? For this you can make use e.g. of the "update range" view in eCognition, as explained in the "objects features" exercise.
- Can you explain why the "water" and "road" objects have an undefined REDNESS value?

Write down the value of the threshold and your explanation for the undefined objects in your text document and **save the customized feature within your final eCognition project.**