# Image Analysis: OCR-classification

## Goal

The goal with this assignment is for you to develop a sense for

1. how image analysis problems are solved with MatLab,
2. how important it is to solve the problem step by step and verifying that each step is working before proceeding.

## Background

You work as a top secret agent for a government agency and you have found strange messages scribbled on Post-It notes. The messages are numbers and you believe that the numbers are some sort of code. A summer worker has scanned the Post-It notes and left you with a huge number of files. The problem is that you can not start your state of the art code-cracking software with images as input. It can only handle strings of characters and you definitely do not want to manually feed the numbers into the system. So it has become your task to write an OCR[1]-software that given the images extract the numbers.

If you succeed you will be highly rewarded, if you fail the safety of the nation can be threatened.... and, more importantly, you will not be highly rewarded.

## Task

Your task is to implement an **automatic** [2] OCR-software. This mean that your solution should be able to extract numbers, classify them and return a string of numbers. It is important that you keep the grouping of the numbers the same as they appear in the image since it is believed that the position of a number is an important feature in the code.

In order to work properly with other software your system **must**[3] have the interface

```
≫s=OCR(Image);
```

Where `Image` is an image obtained from `[Image,Facit]=ReadNumberImage(P,N);` and `s` is a string of numbers in groups of three separated with a space character. The function `ReadNumberImage` can be downloaded from the home page of the course. From the home page you can also find a function `OCR_test(P)` that given a path `P` to a directory of images evaluates your `OCR`-function. For more information on `ReadNumberImage` and `OCR_test` download them and type `help ReadNumberImage` or `help OCR_test` at the MatLab-prompt. The scanned images can also be found on the home page of the course.

[1] Optical Character Recognition

[2] Meaning that you shall not have to change anything in the code of your solution in order to classify different images. You are allowed to store parameters necessary for the classification on disc.

[3] Absolutely no exception to this syntax will be tolerated

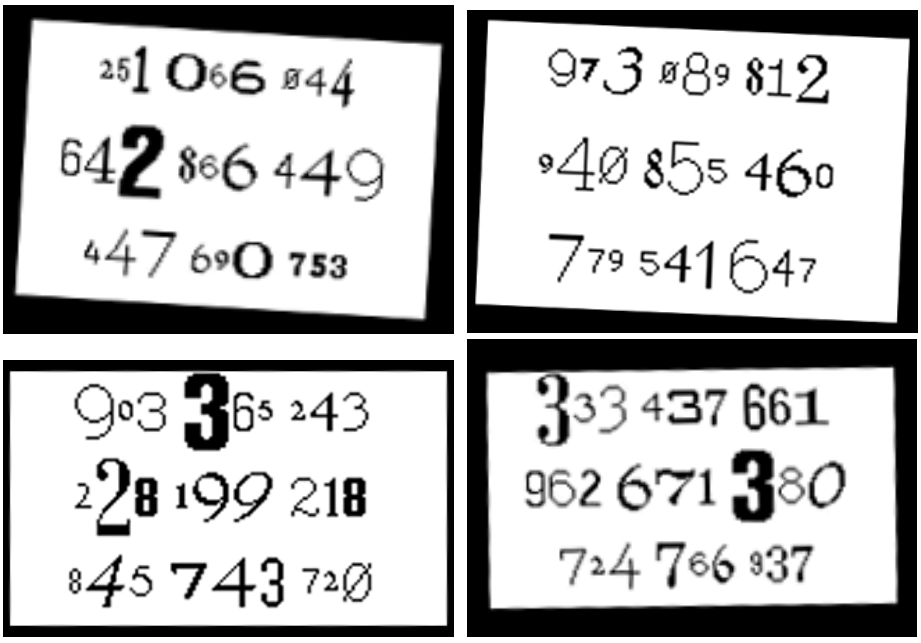
It is recommended that you solve the problem in the following steps

- Download the images and the functions `ReadNumberImage` and `OCR_test` from the home page of the course.
- Store about 20% of the images in a directory called `test` and the rest of the images in a directory called `train`. Store no other files in these two directories.
- Start constructing the `OCR` function in the following steps
  - Restore the image as far as possible. This involves removing disturbances and noise.
  - Even out the intensity of the background (if necessary)
  - Segment the image into connected components (i.e. numbers). Pay attention so that no numbers merge in the segmentation process.
  - Find the rows of numbers in the image and group the numbers accordingly
  - Find the space between groups of numbers and group the accordingly.
  - Extract number by number as separate objects.
  - Postprocess the segmented objects to remove font-specific appearances.
  - Calculate features.
  - Classify the features into numbers (chars) and package them into a string that you return. Pay attention to the format of the string.
- Use the images in the directory `train` when testing what features to use and to estimate parameters of your classifier.
- Evaluate the performance of your system with `OCR_test` on the images in the directory `train`
- Present the results in a report, i.e. convince me that you have understood what you have done and that your solution actually solves the problem that you were given. See the last section for information on what to include in the report.

## Images

The scanned images can be found on the home page of the course. There are several hundreds of images so it should not be a problem to find enough data to train your solution. As suggested earlier, divide the images into two sets and use only one of them for testing and developing your system. The results you present should be from the set of images not used during the development of the system. I will test your system on images that are not available to you.

Examples of images are given below:



Your solution is expected to handle images of the sort shown with a low error rate.

An example of result is the following. When the last of the four images above is presented to the your system, it should return the string `s` in the following format:

```
s='333 437 661 962 671 380 724 766 937'
```

Note that the groups of numbers are kept intact and that the nine groups are presented row-wise. Note also that there is only one space-char between each group of numbers and no space at the beginning or end of the string.

### Hints

Solve the problem in steps, preferably the steps presented earlier.

As you can see, the numbers are printed in different font and with different font-size. This makes it important that you really think about what is characteristic for a specific number. What distinguish a `1` from a `7`? As in many image analysis problem the solution lies in the segmentation of the objects and the features extracted.

Use a simple classifier, for instance a KNN or even IF-THEN-ELSE solutions.

The KNN (K-nearest neighbor) classifier works as follows: For each observation (i.e. connected component) in the training set the desired set of features is calculated. Each connected component is then described as a point in the feature space. To each of the connected components the true class is associated. This can, of course, be obtained via the `Facit` string returned by `ReadNumberImage`. When an unknown number is to be classified the following is performed: The features are calculated and the k-nearest observations from the test set is collected in the feature space. Use the euclidian distance to find the k-nearest points. Assign the most typical class of the k observations collected from the feature space

to the unknown number. In this approach, the parameters that have to be decided are the set of features and the numbers of neighbors that should be considered. For the features, start by looking at the features returned from the `regionprops`-function and for a suitable value for k start with 1.

You could also try to use a neural network as a classifier. For more information see the help associated with the functions `newff,train` and `sim` from the Neural Networks toolbox.

In general, the following MatLab-functions can be useful: `bwlabel, deconvwnr, double, figure, find, fspecial, graythresh, imread, imrotate, max, mean, mean2, ones, regionprops, std, std2, subplot, sum, title` and `zeros`. All morphological operations could be of interest as well.
More information regarding MatLab-functions is obtained by: `help function_name`

A MatLab documentation is shown in your web browser when typing `helpdesk` at the prompt.

## Hand in and judgement

Hand in time is given on the home page of the course.

The report **must**[4] contain:

- The MatLab-code. The code should be well commented.
- Justifications of the chosen solutions. Which solutions have you discarded? Why?
- The classification of at least three different images and probability plots (as created by `OCR_test`) based on at least twenty images.
- A description of the features and the parameters, their values and why you are convinced that there are no better values.

You should be able to interpret new (unseen) images with a probability of correct classification larger that 0.95. The performance for all numbers (zero - nine) should be equal.

Christina Olsén (`colsen@cs.umu.se`) is responsible for the supervision and judgment this assignment. Christina has her office in room MD 426 in the MIT building.

[4]Meaning that a report not adhering to the given points will not be given a pass grade