Image Processing and Pattern Recognition

Assignment 3

November 12, 2012

Submission: Send report and implementation zipped (*[surname].zip*) to `bvme@icg.tugraz.at`.
Deadline: November 26, 2012, 23:59
Note: This assignment will be implemented in C++ using OpenCV 2.4. For informations about downloading and installing, as well as a thorough documentation we refer you to the project's homepage at `http://opencv.willowgarage.com`.
Explanations regarding the provided framework can be found at the end of this specification document.

Texture Classification

In this assignment you have to implement the texture classification algorithm by Varma and Zissermann [1], [2]. The algorithm is divided in a learning stage and a classification stage.
First, filter responses are generated and clustered via the k-means algorithm. For every class, that should be learned, select x images and compute the l filter responses, where l is the number of filter kernels in the filter bank. If the image is of size $m \times n$, then you get $m \cdot n$ feature vectors per image and $x \cdot m \cdot n$ feature vectors per class. All feature vectors of a class are clustered using the k-means algorithm and the resulting cluster centers are collected as *textons* in a dictionary.
Next, you have to train texture models. Each training image of a class is convolved with the filter bank and the responses are labeled with the nearest textons of the dictionary using the Euclidean distance. The frequency of each texton is counted in a histogram, which forms the model. Therefore, you get one model per training image.
To classify a new image, you again have to compute the filter response and the texton histogram and compare it with all model histograms using the χ^2 distance (already implemented in OpenCV). The class associated with the smallest distance is assigned to the image (region).
Note: Each image - for dictionary learning, model training and classification - has to be normalized to zero mean and a unit standard deviation. Use $x = 5$ images for dictionary learning and learn three classes - see Framework Information.

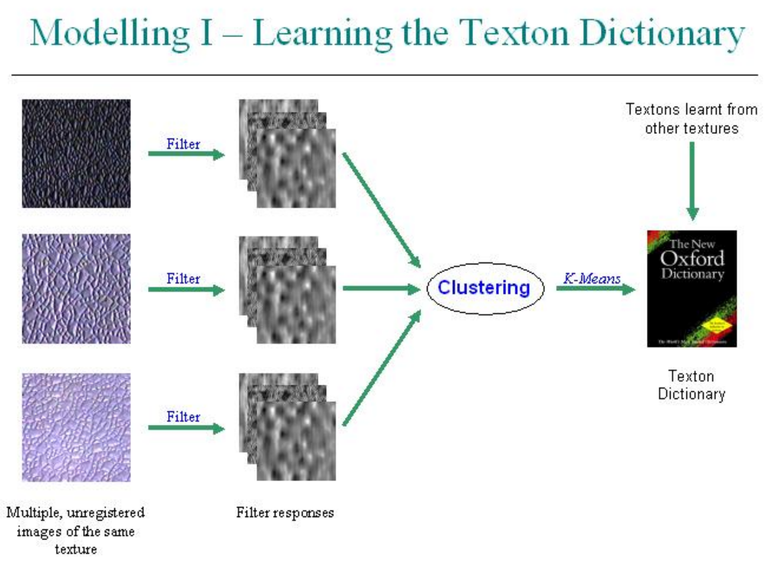
Modelling I – Learning the Texton Dictionary
Filter
Filter
Filter
Clustering
K-Means
Textons learnt from other textures
The New Oxford Dictionary
Texton Dictionary
Multiple, unregistered images of the same texture
Filter responses

Figure 1: Learning the dictionary from training images. Taken from `http://www.robots.ox.ac.uk/~vgg/research/texclass/with.html`

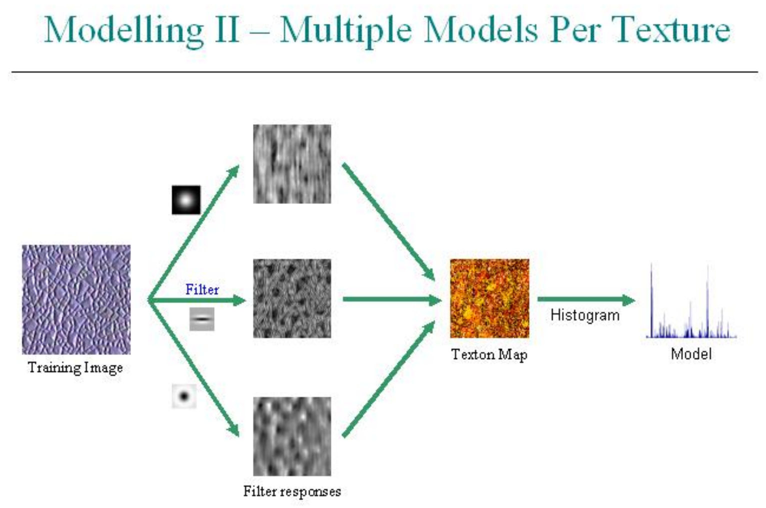

Figure 2: Learning a model from a training image. Taken from `http://www.robots.ox.ac.uk/~vgg/research/texclass/with.html`

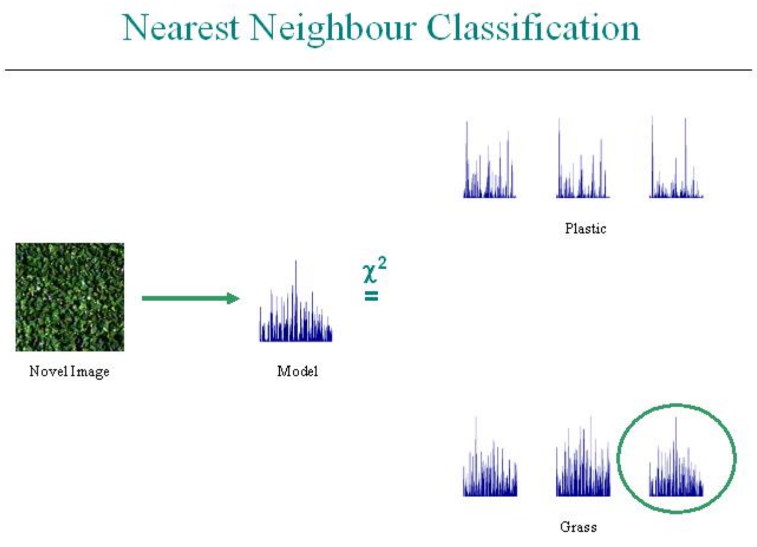

Figure 3: Classify an unseen image. Taken from `http://www.robots.ox.ac.uk/~vgg/research/texclass/with.html`

Database

Download the VisTex database[1] (Download it from `http://www.icg.tugraz.at/courses/lv710.081/info/VisTex.tar.gz`). Use the classes in the reference directory to build the dictionary and to train the model. For classification try different scenes. Use own images also.

Filter responses

We use the Schmid Filter Bank [3] to compute the filter responses. It is defined as follows:

$$F(r, \sigma, \tau) = \cos\left(\frac{\pi \tau r}{\sigma}\right) e^{-\frac{r^2}{2\sigma^2}}$$

with $(\sigma, \tau) \in \{(2,1), (4,1), (4,2), (6,1), (6,2), (6,3), (8,1), (8,2), (8,3), (10,1), (10,2), (10,3), (10,4)\}$. The parameter r defines the support. To normalize each filter kernel you have to subtract the mean of the kernel and divide it by the sum of the absolute kernel values. Therefore, we get $l = 13$.

Experiments and report

Experiment with different classes and report the classification output of several scene images. Try at least three different compositions of classes and four different scenes.

[1] `http://vismod.media.mit.edu/vismod/imagery/VisionTexture/`

Report also problems which arised and how you solved them. Describe also how well the classification works and how you could improve it.

Bonus: Reducing the number of models

The number of models grows linear with the size of training images and this results in a weak runtime of the classification. Varma et al. presents in [2] two approaches to reduce the number of histograms. For this task you have to implement the method *Selection to determine texture models* **or** *Pose normalization* to get all points.

Framework Information

We provide a simple framework which can be built using CMake (`http://www.cmake.org/`). The steps to build and run the framework under Unix-based operating systems are

```
> cd /path/to/framework
> cmake .
> make
> ./learning pathToReferenceDirectory class1 class2 class3 pathToImage2Classify
```

The framework provides several hints on where to start implementing (watch out for the inline documentation). You may adapt the framework at will (add/remove files/classes/functions/...), however, ensure that your submission can be compiled by invoking `cmake/make` on a system where OpenCV 2.4 is installed.
Using additional libraries is not required/allowed. Please ensure that your submission contains only files needed to build and run your implementation (source files, CMakeLists.txt - no binaries, debug files, images).
Note: The framework uses Unix/Linux depending code to iterate files in a directory.

References

[1] M. Varma and A. Zisserman. A statistical approach to texture classification from single images. *International Journal of Computer Vision*, 62(1):61–81, 2005.

[2] M. Varma and A. Zisserman. Classifying images of materials: Achieving viewpoint and illumination independence. In *Proceedings of the European Conference on Computer Vision*, 2002.

[3] C. Schmid. Constructing models for content-based image retrieval. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, 2001.