EXTRACTION OF FEATURES FROM A SET OF IMAGES

==========

```
@{
 author = {António Miguel Baptista Videira},
 title = {Feature Extraction},
 year = {2013},
}
```

INTRODUCTION.

.DESCRIPTOR OF LOW-LEVEL FEATURES

==========

Color Histogram: The default RGB color Histogram is extracted by discretization of the colors in the image into 32 bins, and counting the number of image pixels in each bin. A bin correspond to part of the color intensity spectrum.

Edge Histogram**:** Represent the spatial distribution of five types of edges. Four directional edges and one non-directional. Dividing the image into 16(4x4) non-overlapping sub-images and for each sub-image a histogram with five edge bins is generated, resulting into a descriptor with 5*16 = 80 attributes.

Tamura Features**:** Extracts the features describing the coarseness, contrast and directionality. The first two (Contrast and Coarseness) are represent by single values, while directionality is split into 16 bins. This result in a vector of 18 attributes.
In this case for computation speed, the coarseness value is set to zero.
To get the value of coarseness is necessary to pass an argument to the executable.

Gabor Features: First we pass the image through a filter bank of Gabor filters. The Gabor descriptor is formed as a vector of means and standard deviations of filter responses with different scales and orientations.

All the attributes are normalized between 0 and 1.

.SIFT FEATURES

==========

Since the features are vectors of real numbers it does not make sense to construct a dictionary from all the features obtained from the training set directly, because it would most likely be of overwhelming size. An intermediary step has to be to find a limited number of feature vectors which represent the feature space well, for constructing a dictionary.

After a few research I saw that was often done by k-means clustering, an iterative algorithm for finding clusters in data. After the dictionary has been constructed new images can be described by extracting features from them and matching them with the features in the dictionary which are closest.

The first thing to do is using BoW model to represent images to construct a dictionary. After creating the BOW KMeans Trainer object we call the cluster method to run k-means on the descriptors and obtain the cluster centers. And in the end we will have a Matrix that contains the BoW representation of the image which can now be used with a classification algorithm.

COMPILING AND TESTING.

Run the makefile:
$ make
And then run the exe:
$./main <option> <folder> <xmlfile> <Coarseness>

<option> 1 - For Extraction of the descriptor containing the 166 variables.
2 - To get the Sift descriptor.

<folder> Path to the folder containing the images. e.g. "/home/USER/images"
Or just put the folder containing the images in the folder of the exe.

<xmlfile> File to save the features extracted (e.g. Features.xml).
If it does not exists, is create.

<Coarseness> 1 to extract the coarseness value from the images.
0 the coarseness value is set to zero.

./main 1 image Features.xml 0 //Extract the 166 descriptor(with the coarseness = 0)
./main 1 image Features.xml 1 //Extract the 166 descriptor(with the coarseness)
./main 2 image Features.xml //Extract the SIFT descriptor