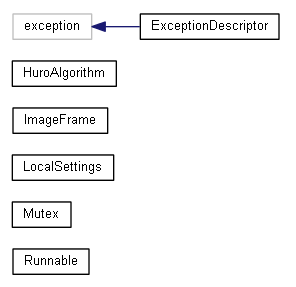
# Outline

This application concentrate only for extracting interesting points from images, the feature matching and indexing is not the part of it. We propose the base of a feature extraction system by a multiprocessing scheme. The first stage is comprised to extract simultaneously efficient local and global image descriptors to get a small number of bestcandidate images from the HURO database.

Most object recognition systems tend to use either global image features, which describe an image as a whole; or local features, which represent image patches. Global features have the ability to generalize an entire object with a single vector. Consequently, their use in standard classification techniques is straightforward. Local features, on the other hand, are computed at multiple points in the image and are consequently more robust to occlusion and clutter. However, they may require specialized classification algorithms to handle cases in which there are a variable number of feature vectors per image.

# Class hierarchy

## General classes

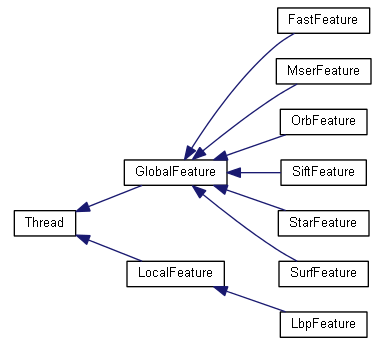


* Exception class whose instances store an error message, source file, function and line number.
* HuroAlgorithm, which manages the whole feature extraction process.
* Image frame class which is an interface for media handling (it forwards the incoming media streams to the feature extractors as OpenCV Mat objects).
* LocalSettings: Singleton settings manager class.
* The other classes are for handling threads. Runnable classes can be derived. Threads can be started and stopped with the Start() and Join() methods respectively. The new thread will enter the implemented Run() method.

### HuroAlgorithm

This class contains the main algorithm, and also resource owner for accessing images and for feature extractor pools. So it stores all of the local-, and global feature extractors, and has a member through which the multimedia data are streamed.

## Feature extractors



The general feature extractors (local and global) are derived from thread super class. These can be executed simultaneously for efficient processing. These two classes store the vector of keypoints (in case of local feature extractors) or a single vector of values (in case of global feature extractors) and forwards they for future processing – feature matching.

The description of feature extractors can be found in our first HURO report, so I don’t wish to talk more about these.

### Global and local feature extractor classes

A little explanation to the general feature extractors:

Everyone runs in its own thread, and has its own feature vector – vector<double> in case of global feature extractors; and vector<cv::KeyPoint> in case of local feature extractors, where cv::KeyPoint is an inner structure of OpenCV (strores the coordinates, size and direction of the local feature vector).

Both of these have some virtual methods (implemented by the concrete feature extractors), like:

* LoadSettingsFromFileStorage(): method for loading algorithm specific settings from the given storage.
* DrawFeatres(): Method for drawing the extracted features.
* Process(): Method for each feature extraction algorithm.