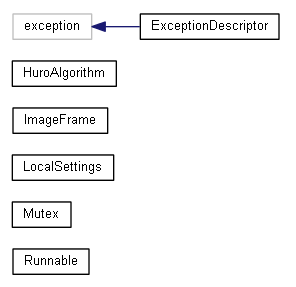
# Outline

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

Our system uses on the one hand global image features, which describe an image as a whole so which have the ability to generalize an entire object with a single vector. And the other hand it uses local features, which represent image patches, and which are computed at multiple points in the image and are consequently more robust to occlusion. However, they may require specialized classification algorithms.

# Class hierarchy

## General classes

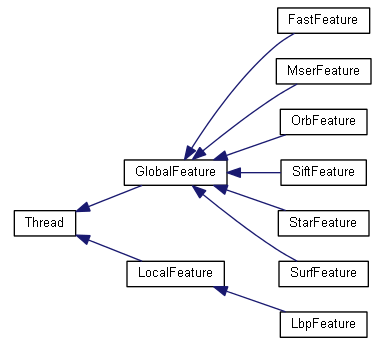


* Exception class whose instances store an error message, name of a source file, function and line number.
* HuroAlgorithm, which manages the whole feature extraction process. I will explain it a little bit more later.
* Image frame class which is an interface for media handling (it forwards the incoming media streams to the feature extractors as an OpenCV Mat object). Its construction is still an open question.
* LocalSettings: Singleton settings manager class. Every parameter of the feature extractors can be tuned through this class. (No need to recompile the sources.)
  + Those feature extractors, what we want to apply, can be set also here.
* 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. And the new thread will enter in the implemented Run() method (every feature extractor has to overload it).

### HuroAlgorithm

This class contains the main algorithm, and also resource owner for accessing images (I mean controlling/encoding the incoming data) and manager of feature extractor pools. So it stores, initializes and starts all of the local-, and global feature extractors, and has a member through which the multimedia data are streamed. The resulted feature vectors are accessible through an instance of HuroAlgorithm.

## Feature extractors



The general feature extractors (local and global) are derived from thread super class. The notation is reversed here; I need to refractor the names. These can be executed at the same time 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.