Keysens

vAlgorithms development

Technical note

Document title: technical note - *vAlgorithms* development
Document number: 1005
Revision: 04
Date: June 2018

Keysens - Machine Vision
Poligono Ecce Homo nave 4. 12530 Burriana, Spain
info@keysens.com
www.keysens.com

Contents

1 Introduction

Keysens vAlgorithms **are a collection of functions written in C and contained in a DLL. They implement machine vision and data processing algorithms.** ***Keysens*** **software like** ***vDevelop*** **and the runtimes load several algorithms DLLs and** ***algorithms description files*** **contained in the algorithms directory,** `alg`**.**

For building machine vision applications one makes a project with ***vDevelop.*** **Projects consists of several parameters for camera settings and communications with installation devices like robots, PLCs and HMIs, and the most important, an algorithms script, a list of algorithms that will be executed sequentially.**

Every algorithm in the script uses the output data from the previous algorithm, does its computations and makes its results available to the next algorithm. So, every algorithm has one conceptual data type as input and one conceptual data type as output. The data type that an algorithm provides as output has to be compatible with the data type the next algorithm needs as input.

For building new libraries of algorithms one has to prepare a new DLL. New algorithms are compiled and included in the new DLL. There has to be one function for each algorithm, usually in a separate file. All these functions have the same parameters, so the prototypes of the algorithms functions are the same. The names of the functions have to follow a convention. The exact prototype and the values contained in the parameters will be explained below.

Furthermore, there has to be an ***algorithms description file*** **that contains valuable information about each algorithm for the programs that use them. This is a text file that follows a convention and basically contains keywords specifying the algorithms characteristics, like the acronym, the input and output conceptual data types, title, description, the parameters and their values (minimum, maximum, default), indications about if the algorithms**

provides results that have to be displayed graphically, like points, lines, rectangles, circles, etc. The *algorithms description file* format and all the keywords will be explained below.

Images in *Keysens vAlgorithms* are passed to the algorithms functions using the *OpenCV* image data type, and algorithms are encouraged to use *OpenCV* functions when possible, so there must be a version of *OpenCV* installed in your computer. Version 2.4.13 is recommended.

2 Conceptual data types

The conceptual data types for input and output can be one of these:

Data type	Meaning
NUL	Not specified data type, can be any data type. Used only by algorithms that recover previous results.
RAW	A colour image in a colour space, usually RGB but it can be another. It is a 3-channel image. If the image is grey-level, it is still a 3-channel image with the same information in the three bands, so it appears as a grey-level image.
LBL	A one-channel image. It can be a grey-level image or contain labels that usually mean object or background, but they can mean anything the developer designs. There are 256 labels, from 0 to 255.
DAT	A matrix of real numbers. The size of the matrix, rows and columns, is determined by the algorithm at execution time and may vary at each execution. DAT types are used as detected objects, each row being one object and each column being one object feature (centre x, centre y, area, etc.).
REG	A collection of regions, that is, subsets of the image. Regions usually represent neighbour pixels that have characteristics in common, like a similar grey level. From regions many features can be computed, like area, centre, elongation, perimeter, extension, moments, etc.

Although an algorithm provides only one conceptual data type as input or output, their values can actually have many data (many lines, for example, in a lines detection algorithm).

3 Compatible conceptual data types

When building the algorithms script with *vDevelop* one can insert, delete and substitute algorithms. These operations are allowed only if the output data type of every algorithm is compatible with the input data type of the next one. Besides, when inserting and substituting, *vDevelop* shows a list filled only with the compatible algorithms. `NUL` is compatible with all data types, the remaining data types are only compatible with the same data type.

Data type	Compatible with
NUL	NUL, RAW, LBL, DAT, REG
RAW	RAW
LBL	LBL
DAT	DAT
REG	REG

4 Writing a DLL of vision algorithms

DLLs are written in C using a programming development environment like *Microsoft Visual Studio*, *Bloodshed Dev-C++*, Qt and others.

In the *Keysens vAlgorithms* standard, the DLL should always have two files, `dllmain.c` and `dll.h`. `dllmain.c` has one function, `VersionName`, that returns a C character string with the DLL name (and version or any information of 80 characters maximum). *vDevelop* and the runtime programs call this function after loading the DLL and display the DLL name.

There is another C character string defined, `_result`, that is used for all algorithms to put an information before finishing its execution. This information is displayed by *vDevelop* after executing the function of the algorithm. The runtime programs do not display this information for not to fill its log window, since it is a runtime program that executes algorithm scripts at frame rate, continuously. The algorithms developed by *Keysens* write the algorithm name and the values of the algorithm parameters in this string.

`dll.h` defines names for building the DLL, includes *OpenCV* header file, makes the `_result` string available to all files, declares the `VersionName` function prototype, and finally declares two data structures that are used by the algorithms for passing values of a certain type:

- **`REGIONS_STRUCT`: used by algorithms working with REG data type as input or output for passing detected regions in *OpenCV* style.**
- **`DATA_STRUCT`: used by algorithms working with DAT data type as input or output for passing matrices with numeric values of any number of rows and columns.**

`REGIONS_STRUCT` and `DATA_STRUCT` are the C data types where conceptual data types REG and DAT are stored. Figure 1 shows the `dllmain.c` file listing and Figure 2 the `dll.h` file listing.

```
#include "dll.h"

char _version[80] = "My Vision Algorithms DLL version";
char _result[10240] = "";

extern "C" ALGORITHMSSHARED_EXPORT char *VersionName()
{
    return _version;
}
```

Figure 1: `dllmain.c`

Every algorithm has to be written in a C function in a separate file. The function should have the following prototype:

```
extern "C" ALGORITHMSSHARED_EXPORT char * execute_ALGORITHM_ACRONYM(
    IplImage *inImage,void *inResource,
    IplImage *modelImage,void *modelResource,
    IplImage *outImage,void *outResource,int *pars)
```

Where `ALGORITHM_ACRONYM` has to be substituted by the actual algorithm acronym.

As seen, an algorithm may have an input image and an input resource, a model image and a model resource, and an output image and an output resource. These data is not always present in all algorithms. If not present, it has the C value `NULL`. The images are defined as `IplImage` type, that is the *OpenCV* data structure for images. The resources are defined as `void` pointers because, depending on the conceptual data type of the algorithm, they will point to a `REGIONS_STRUCT` object or to a `DATA_STRUCT` object.

```
#ifndef DLL_H
#define DLL_H

#include "user_algorithms_global.h"

#include <cv.h>

#ifndef M_PI
#define M_PI                3.14159265358979323846
#endif

#define OBJECT_LABEL        255
#define SECOND_OBJECT_LABEL 127
#define NON_OBJECT_LABEL      0

extern char _result[10240];

typedef struct REGIONS_STRUCT
{
    CvMemStorage* storage;
    CvSeq* contours;
    int number;
} REGIONS_STRUCT;

typedef struct DATA_STRUCT
{
    int rows,cols;
    double *data; //array of rows x cols elements
} DATA_STRUCT;

#endif // DLL_H
```

Figure 2: `dll.h`

Depending on the behaviour of the algorithm defined with the keywords in the *algorithms description file*, an algorithm may pass a "model" image contained in a file and a model resource also contained in a file. The name of these files containing models will be explained later.

The array `pars` contains the algorithm numeric parameters as defined in the *algorithms description file*, filled with the values of the algorithm entry in the project being executed. The first four values are always the region of interest set in the program executing the algorithm at that moment. The region of interest should be used to process only that part of the images in algorithms that actually process images. The rest of the values in `pars` depend on each algorithm and are defined in the *algorithms description file*. There is no indication of the number of values because each algorithm expects a certain number of values, again as defined in the *algorithms description file*.

A sample algorithm function written in a file such as AlgorithmFoo.cpp may have the aspect shown in Figure 3.

5 Execution of algorithm scripts

***vDevelop* and the runtime programs execute one algorithm after another following the algorithms script of their current vision project. For that, some simple rules are followed:**

- **The output of one algorithm is passed as input to the following one.**

```
#include "dll.h"

#include <windows.h>
#include <stdio.h>
#include <stdlib.h>

#include <cv.h>

extern "C" ALGORITHMSSHARED_EXPORT char * execute_ALGORITHM_ACRONYM(
    IplImage *inImage,void *inResource,
    IplImage *modelImage,void *modelResource,
    IplImage *outImage,void *outResource,int *pars)
{
    int width = outImage->width;
    int height = outImage->height;

    int roi_col = pars[0];
    int roi_row = pars[1];
    int roi_width = pars[2];
    int roi_height = pars[3];

    //process images and/or data, provide results in outImage and/or outResource
    ...

    sprintf(_result,"ALGORITHM_ACRONYM: roi: %d %d %d %d,
        roi_col,roi_row,roi_width,roi_height);
    return _result;
}
```

Figure 3: `AlgorithmFoo.c`

- **The programs have a global variable containing the current value of the region of interest. This is passed to every algorithm as the first four values of the `pars` array (x coordinate of the top-left corner, y coordinate, width and height). That is, they are passed at positions 0 to 3 in the `pars` array. An algorithm may indicate the programs to modify the value of the region of interest, this is done with keywords `"@setroi"` and `"@prevroi"`.**
- **The algorithms numeric parameters are passed starting at position four in the `pars` array.**
- **The programs store all the results of all algorithms in case they are needed by a future algorithm in the script.**

6 The algorithms description file

As mentioned, besides the DLL containing the algorithms code, there has to be an *algorithms description file* that indicates the programs that use the DLL about the algorithms characteristics: parameters, conceptual data types for input and output, type of results it provides, etc. A sample listing of an *algorithms description file* with three algorithms can be seen in Figure 4.

The format of the *algorithms description files* is as follows:

- **Lines starting with # are comments.**
- **Every algorithm description starts with "START ALGORITHM" and end with "END".**

- **Between "START ALGORITHM" and "END" there are lines of keywords and their values.**

All keywords start with character `@`. Keywords have values after the keyword name, this is always a string but numbers may be extracted from the string. Any keyword may appear or not in the description. If it does not appear, its value is an empty string. A list of all keywords grouped by its use, a brief description and values can be seen in Figure 5.

When a keyword has one or more numbers as values, the numbers should be separated by a space. When a number can be 0 or 1, one means true, that is, the keyword should be defined with value 1 to take effect.

Following, the keyword effects are explained with more detail, including the format of the result that an algorithm with that keyword has to provide. Further on, when we refer to *the programs* we mean *vDevelop* and the runtime programs, and when we refer to *the algorithms*, we refer to any vision algorithm written with the *Keysens* approach.

6.1 Keywords for algorithm description

@acronym

The name of the algorithm as should appear in the algorithm function of the DLL. Usually acronyms are in capital letters. If they consist of several words, they are separated by _. Example: `LINES_DETECTION`.

@input

Conceptual data type of the algorithm input. It can be `NUL`, `IMG`, `LBL`, `DAT`, `REG`. It is used by the programs to chain compatible types in the algorithms script.

@output

Conceptual data type of the algorithm output. It can be `IMG`, `LBL`, `DAT`, `REG`. It is used by the programs to chain compatible types in the algorithms script and to allocate resources for each algorithm.

@title

Algorithm title. It is displayed by the programs.

@description

Algorithm description. It is displayed by the programs.

@icon

Algorithm icon. It has to be a `pgn` file of size 24×24. It is displayed by the programs.

6.2 Keywords for algorithm parameters

@parameters

Algorithm parameters names. It is a string where names are separated by commas. Algorithm parameters are numbered starting from one.

@values

Algorithm parameters values. A string with integer numbers separated by spaces. The programs put the values in the array `pars` when calling the algorithm function.

@defvalues

Algorithm parameters default values. A string with integer numbers separated by spaces. The programs can restore the parameters values to their default values.

@minvalues

Algorithm parameters minimum values. A string with integer numbers separated by spaces. The programs check that the parameters values are not smaller than the minimum values.

@maxvalues

Algorithm parameters maximum values. A string with integer numbers separated by spaces. The programs check that the parameters values are not bigger than the maximum values.

@editable

Tells the programs whether the parameters values can be modified by the user. If the value is 1 they can be modified.

@mouse

Tells the programs whether the parameters values can be modified by the user with the mouse. A value greater than 0 indicates that this parameter number and the next three, four in total, will be modified when editing parameters and dragging the mouse. The first parameter is the mouse column in the image, the second one is the mouse row, the third one is the width of the rectangle defined by the mouse when dragging and the fourth one is the height.
The numbers are in image coordinates independently of the image being displayed with a zoom or displayed in different areas of the program window.

6.3 Keywords for drawing algorithm results

@lines

If the value is 1 indicates to the programs that the result represent lines and should be drawn over the original image when the algorithm is selected.
The algorithm is expected to provide a DAT output where every row represents a line. The columns represent:

- **line distance to the top-left corner of the image,**
- **line orientation in degrees relative to the horizontal.**

@segments

If the value is 1 indicates to the programs that the result represent segments and should be drawn over the original image when the algorithm is selected.

The algorithm is expected to provide a DAT output where every row represents a segment. The output should have six columns that represent:

- **line distance to the top-left corner of the image,**
- **line orientation in degrees relative to the horizontal,**
- **x coordinate of the first segment point,**
- **y coordinate of the first segment point,**
- **x coordinate of the second segment point,**
- **y coordinate of the second segment point.**

@circles

If the value is 1 indicates to the programs that the result represent circles and should be drawn over the original image when the algorithm is selected.
The algorithm is expected to provide a DAT output where every row represents a circle. There should be three columns that represent:

- **x coordinate of the centre of the circle,**
- **y coordinate of the centre,**
- **the circle radius.**

@rectangles

If the value is 1 indicates to the programs that the result represent rectangles and should be drawn over the original image when the algorithm is selected.
The algorithm is expected to provide a DAT output where every row represents a rectangle. There should be eleven columns that represent:

- **x coordinate of the centre of the rectangle,**
- **y coordinate of the centre,**
- **orientation of the centre,**
- **x coordinate of the rectangle corner point,**
- **y coordinate of the first corner,**
- **x coordinate of the second corner,**
- **y coordinate of the second corner,**
- **x coordinate of the third corner,**
- **y coordinate of the third corner,**
- **x coordinate of the fourth corner,**
- **y coordinate of the fourth corner.**

The corners have to in clockwise or counter-clockwise order because lines are drawn from one corner to the next and from the last one to the first one.

@points

If the value is 1 indicates to the programs that the result represent points and should be drawn over the original image when the algorithm is selected.
The algorithm is expected to provide a DAT output where every row represents a point. There should be two or three columns, the third one being optional. columns represent:

- **x coordinate of the point,**
- **y coordinate,**
- **orientation of the point.**

@orderedpoints

If the value is 1 indicates to the programs that the result represent points and should be drawn over the original image when the algorithm is selected.
The output has to as explained for keyword `@points`, but the points, besides being drawn, are joint by lines indicating their order. The first point is indicated by a bigger circle.

@rois

If the value is 1 indicates to the programs that the result represent regions of interest (ROIs) and should be drawn over the original image when the algorithm is selected.
The algorithm is expected to provide a DAT output where every row represents a ROI. There should be four columns that represent:

- **x coordinate of the top-left corner of the ROI,**
- **y coordinate of the top-left corner,**
- **width,**
- **height.**

6.4 Keywords for passing previous results to algorithms

@previmage

If the value is 1 indicates to the programs that the first algorithm parameter refers to a previous algorithm number (starting from one in the script). The image (if there is one) provided as a result by that previous algorithm will is passed as the input image to this algorithm.
If the immediately previous algorithm provides also an image as result, this image is passed as the model image.
The parameters that indicate the algorithm uses a previous image, a previous DAT resource, a previous REG resource, a model image in a file or a model data file, should be the first algorithm parameters and should be in the following order:

- **The number of the previous image if "@previmage 1" is specified in the algorithms description file.**
- **The number of the previous DAT resource if "@prevdata 1" is specified.**
- **The number of the previous REG resource if "@prevregions 1" is specified.**
- **The number of a model image file if "@modelimage 1" is specified.**
- **The number of a data file if "@modeldata 1" is specified.**

@prevdata

If the value is 1 indicates to the programs that one algorithm parameter indicates a previous algorithm number (starting from one in the script). The DAT result (if there is one) provided as a result by that previous algorithm will be passed as input resource to this algorithm.
If the immediately previous algorithm provides also a REG or DAT result, this resource is passed as the model resource. The algorithm parameter that indicates the previous DAT result to be used is as explained in keyword `@previmage`.

@prevregions

If the value is 1 indicates to the programs that one algorithm parameter indicates a previous algorithm number (starting from one in the script). The REG result (if there is one) provided as a result by that previous algorithm will be passed as input resource to this algorithm.
If the immediately previous algorithm provides also a REG or DAT result, this resource is passed as the model resource. The algorithm parameter that indicates the previous REG result to be used is as explained in keyword `@previmage`.

@combinerows

If the value is 1 indicates to the programs that the DAT resource to be presented as input to this algorithm has to be constructed from combining the rows of the DAT outputs of several previous algorithms, indicated in the algorithm parameters. Algorithms to be combined are indicated with their order number in the algorithms script, starting from one. An algorithm using this keyword has access to all the DAT results of the combined algorithms. If the column numbers of the combined algorithms do not match, the biggest column number is used and values are filled with zeros when needed.

@combinerows

If the value is 1 indicates to the programs that the DAT resource to be presented as input to this algorithm has to be constructed from combining the columns of the DAT outputs of several previous algorithms, indicated in the algorithm parameters. Algorithms to be combined are indicated with their order number in the algorithms script, starting from one. An algorithm using this keyword has access to all the DAT results of the combined algorithms. If the row numbers of the combined algorithms do not match, the biggest row number is used and values are filled with zeros when needed.

6.5 Keywords for passing models contained in files

@modelimage

If the value is 1 indicates to the programs that one algorithm parameter indicates the name of the file to be used as model image. Model image files have names as `model999.ppm`, where `999` indicates a positive integer number.
The algorithm parameter that indicates the model image file to be used is as explained in keyword `@previmage`.

@modeldata

If the value is 1 indicates to the programs that one algorithm parameter indicates the name of the file to be used as model data. Model data files have names as `model999.dat`, where

`999` indicates a positive integer number. Data files are plain text files containing a matrix in this format:

```
[number_of_rows  number_of_columns]
row one values separated by spaces
row two values separated by spaces
...
```

The algorithm parameter that indicates the model data file to be used is as explained in keyword `@previmage`.

6.6 Keywords for the algorithm to set a regions of interest

@setroi

If the value is 1 indicates to the programs that the first row of the DAT result represent a region of interest (ROI) and has to be used as the current region of interest for the next algorithm. The algorithm should provide the new ROI in a DAT result, in the first row and first four columns. These four columns represent:

- x coordinate of the top-left corner of the ROI,
- y coordinate of the top-left corner,
- width,
- height.

@prevroi

If the value is 1 indicates to the programs that the region of interest (ROI) has to be set to the provided value before executing this algorithm. The algorithm should have at least four parameters, the first four parameters represent the ROI like this:

- x coordinate of the top-left corner of the new ROI,
- y coordinate of the top-left corner,
- width,
- height.

6.7 Keywords for changing the default resource allocation of the algorithm

@outdata

If the value is 1 indicates to the programs that the algorithm will also provide an output resource of type DAT although its output is of type RAW or LBL. So the programs will allocate resources for this DAT result.

@outimage

If the value is 1 indicates to the programs that the algorithm will also provide an output image of type RAW, a three channels image, although its output is not of type RAW or LBL. So the programs will allocate this output image and will display it when the algorithm is selected. This output image will not be passed as input image to the next algorithm.

@noimage

If the value is 1 indicates to the programs that no image resource has to be allocated for this algorithm although its output is RAW or LBL. Instead, the programs will pass to this algorithm the output image of the previous algorithm.

@noresource

If the value is 1 indicates to the programs that no DAT or REG resource has to be allocated for this algorithm although its output is DAT or REG. Instead, the programs will pass to this algorithm the output resource of the previous algorithm. This keyword is used together with `@prevdata` or `@prevregions`, that make the programs present the output of a previous algorithm as input resource.

6.8 Keywords for algorithms that may end the execution of the script

@endnodata

If the value is 1 indicates to the programs to finish the execution of the algorithms script is the algorithm has a result of type DAT with zero rows, that is, if there is no data in the result.

@endondata

If the value is 1 indicates to the programs to finish the execution of the algorithms script is the algorithm has a result of type DAT with any data, that is, with number of rows and columns greater that zero.

@endonfalse

If the value is 1 indicates to the programs to finish the execution of the algorithms script if the algorithm has a result of type DAT that has a value zero (false) in the first element of the matrix, that is, element at row one and column one.

@endontrue

If the value is 1 indicates to the programs to finish the execution of the algorithms script if the algorithm has a result of type DAT that has a value different from zero (true) in the first element of the matrix, that is, element at row one and column one.

6.9 Keywords for algorithms that accumulate results from several executions and send data statistics

@nosend

If the value is 1 indicates to the programs not to send any data on then local network after processing this algorithm, although this algorithm finishes the execution of the algorithms script. Usually results will be sent from another algorithm when some condition are met.

@sendacc

If the value is 1 indicates to the programs to send the global data resource, computed as accumulated data results. It is sent if this algorithm finishes normally. The algorithm may

finish the script if it has keywords like @endnodata, @endondata, @endontrue, @endonfalse, that may be used in combination with @sendacc.

@statsdata

If the value is 1 indicates to the programs to accumulate this data results to a global data resource, computing the addition, the mean and the maximum values of the first column of the results. The global resource is initialized if needed, its dimensions are 3 columns and the same number of rows as the results of this algorithm. If the number of rows changes, it is initialized again.

@sumdata

If the value is 1 indicates to the programs to sum this data results to a global data resource, computing the addition for every row and column. The global resource is initialized if needed, its dimensions are the same as the results of this algorithm. If the number of rows or columns change, it is initialized again.

7 Comments

If you experience any problems with this document or want to give us feedback, please email us at info@keysens.com.

```
# MY ALGORITHMS DESCRIPTION FILE

# IMAGE MANIPULATIONS

START ALGORITHM
        @acronym EQUALIZE
        @input RAW
        @output RAW
        @title Equalize the histogram of the input image
        @description Converts the input image from RGB to grey level and equalizes ...
        @parameters none
        @defvalues 0
        @minvalues 0
        @maxvalues 0
        @editable 0
        @icon equalize.png
END

START ALGORITHM
        @acronym ROTATE
        @input DAT
        @output RAW
        @title Rotate image
        @description Rotate image. Input data is centre x, centre y and angle. If ...
        @parameters raw image,inverse rotation
        @defvalues 1 0
        @minvalues 1 0
        @maxvalues 100 1
        @editable 1
        @previmage 1
        @icon rotate.png
END

START ALGORITHM
        @acronym SMOOTH
        @input RAW
        @output RAW
        @title Smooth image with mean or Gaussian filter
        @description Image is blurred applying a mean (type=0) or a Gaussian ...
        @parameters columns,rows,type
        @defvalues 5 5 0
        @minvalues 1 1 0
        @maxvalues 29 29 1
        @editable 1
        @icon smooth.png
END
```

Figure 4: `Sample algorithms description file`

Keyword	Value	Brief description
@acronym	string	algorithm name or acronym
@input	string	conceptual input data (NUL, RAW, LBL, DAT, REG)
@output	string	conceptual output data (RAW, LBL, DAT, REG)
@title	string	algorithm title
@description	string	algorithms description
@icon	string	algorithm icon name
@parameters	string	parameter names
@values	numbers	parameter values
@defvalues	numbers	parameter default values
@minvalues	numbers	parameter minimum values
@maxvalues	numbers	parameter maximum values
@editable	number 0 or 1	parameters are editable
@mouse	number	parameters can be modified with the mouse
@lines	number 0 or 1	results are lines and should be drawn
@segments	number 0 or 1	results are segments and should be drawn
@circles	number 0 or 1	results are circles and should be drawn
@rectangles	number 0 or 1	results are rectangles and should be drawn
@points	number 0 or 1	results are points and should be drawn
@orderedpoints	number 0 or 1	results are ordered points and should be drawn
@rois	number 0 or 1	results are regions of interest and should be drawn
@previmage	number	algorithm uses a previous image result (RAW or LBL)
@prevdata	number	algorithm uses a previous DAT result
@prevregions	number	algorithm uses a previous REG result
@combinerows	number 0 or 1	DAT result are rows combined from previous algorithms
@combinecols	number 0 or 1	DAT result are columns combined from previous algorithms
@modelimage	number	algorithms uses a model image
@modeldata	number	algorithms uses a model data file
@setroi	number 0 or 1	algorithm output modifies the region of interest
@prevroi	number 0 or 1	algorithms sets the region of interest before its execution
@outdata	number 0 or 1	algorithm output is not of type DAT but it also creates a DAT output resource
@outimage	number 0 or 1	algorithm output is not of type RAW or LBL but it also creates an output image. This output image is used by the programs only for visualization, it is not passed as input image to the next algorithm.
@noimage	number 0 or 1	algorithm result is an image but it should no create an image, it should copy the one from a previous algorithm
@noresource	number 0 or 1	algorithm result is DAT or REG but it should no create a resource, it should copy the one from a previous algorithm
@endnodata	number 0 or 1	algorithms finishes the script execution if its DAT result has no data
@endondata	number 0 or 1	algorithms finishes the script execution if its DAT result has any data
@endonfalse	number 0 or 1	algorithms finishes the script execution if its DAT result has value zero
@endontrue	number 0 or 1	algorithms finishes the script execution if its DAT result has value different from zero
@nosend	number 0 or 1	algorithms does not send data although it is the last one of the script
@noacc	number 0 or 1	algorithms sends the results from a global data resource that contains data statistics from several executions
@statsdata	number 0 or 1	algorithms tells the programs to compute accumulated statistics from its results to a global data resource
@sumdata	number 0 or 1	algorithms tells the programs to add the values of its results to a global data resource

Figure 5: `Keywords list`