Project Number:	FP6-516059
Document number:	DR_FRESH_WP2_2.4.3
Document Title:	Specification for identification and recognition of symbols and associated software prototype tools
Date:	31/01/07

Abstract

This report contains the specifications and documentation of the symbol recognition software.

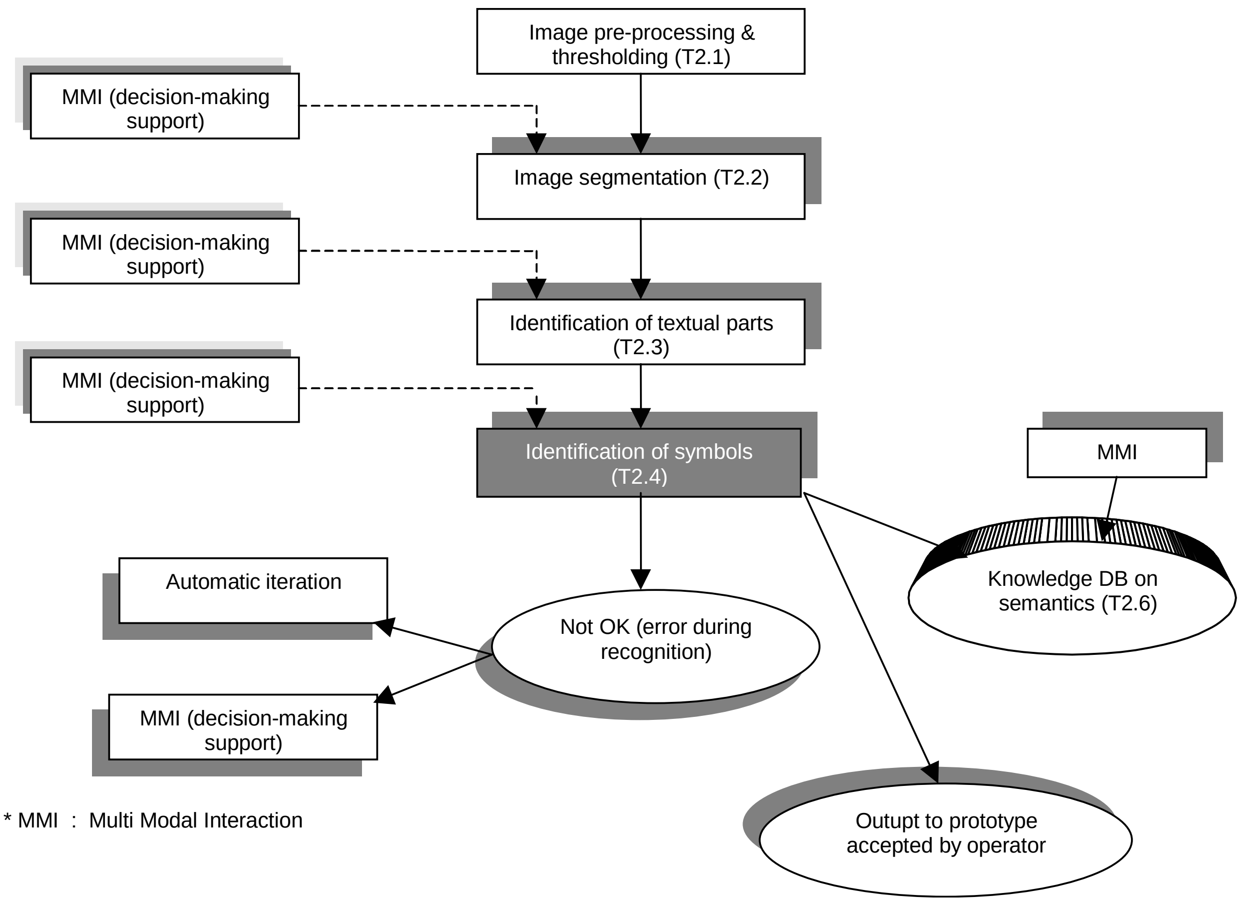
Keyword List WP2, symbol, recognition, software, integration, specifications, documentation

CONTENTS

1. PURPOSE, SCOPE AND OVERVIEW OF THIS DOCUMENT

This document contains the software documentation of the symbol recognition process. This document completes document D2.4.1 “Intermediate report with first interactive prototype”. It makes reference to document D2.4.2 for common sections. The WP2 intends to develop a recognition tool that transfers paper plans in reconstructed wiring information. It is based on the generic recognition framework as described in the associated sub-activities: the overall process is depicted in the following chart, on which we can identify Task 2.4 of WP2 :

Figure 1 : Task 2.4 in WP2



The Identification of symbols (or Symbol Recognition Task) has as main inputs:

- Small parts of scanned wiring plans, called Source Images, resulting from Image Segmentation Task (T2.2). These symbols are given in Win32 BMP format.
- A family of models, called Reference Images, which can be attached to some parts of Semantic Knowledge DB (T2.6). Same format as previous item: Win32 BMP. The main objective of our software is that the source images are recognized among the reference images.

2. SOFTWARE REQUIREMENTS

In this section we describe more detailed functionalities related to the global architecture of the Symbol Recognition Software. This architecture reproduces four main processes:

- Computation of reference image signatures.
- Computation of source image signatures.
- For each source image, computation of its distance with each reference image.
- For each source image, sorting of the reference images in ascending order of their distance from the source image.

Beside these specific calculative functionalities related to descriptor computation, the software is composed of a set of utility functions:

- Configuration of input parameters.
- Scan of reference images and of source images.
- Conversion of image format.
- Generation of intermediate results for study.
- Generation of final results.

All these functions will make this method for symbol recognition possible. The quality of this method depends on two needs:

- a method allowing for fast integration of new descriptors
- some functions allowing integration with external software.

The list of functions that are going to be described is:

- Calculative functions: Image signature composition of geometric descriptors.
- Utility functions:
  - Configuration of input parameters.
  - Scan of reference images and of source images.
  - Conversion of image format.
  - Generation of intermediate results for study.
  - Generation of final results.
- Integration functions:
  - Rapid integration of new descriptors.
  - Integration with external software.

2.1. CALCULATIVE FUNCTIONS

Please refer to document D2.4.2 “Final report on symbol recognition with evaluation of performances” for details on descriptor computation and distance measurements.

2.1.1. IMAGE SIGNATURE COMPOSITION OF GEOMETRIC DESCRIPTORS

An image signature is a vector where each coordinate is the result of a descriptor as described in D2.4.2. These geometric image signatures are vectors of non-negative integers. The order of descriptors used inside the vector is the same for all image signatures.

Image signatures can be divided into two sets:

- Source images which are those to be recognized.
- Reference images which form the reference for the recognition process.

Image signatures can be obtained from:

- Direct processing of each descriptor.

Since the descriptor name is given each time inside the signature file, the order of them is free: the software will order itself in a same way for all images. When an image signature has been created from direct processing, a CSV image signature file is saved with the same format as before.

2.2. UTILITY FUNCTIONS

2.2.1. CONFIGURATION OF INPUT PARAMETERS

Input parameters of the software are saved in an INI file named rec.ini. We do not need multilevel structure, so INI file format has been chosen rather than XML file format because of its simplicity and parsers performance. Moreover this format is portable.
There are a general section and one section per descriptor. The parameters of a descriptor section are described in D2.4.2.

2.2.2. SCAN OF REFERENCE IMAGES AND OF SOURCE IMAGES

The scan of directories should be done in a portable way (Windows, Linux). Unfortunately no standard functions exits for this in both platforms so the choice has been made to use POSIX functions “opendir”, “readdir” and “closedir” with structure data “dirent”. These functions are implemented with the specific functions available on MS Windows platform. This gives a transparent way for application to scan directories.

In order to recognize image file of a directory in a portable way, the choice to use the function “imageFormat” of the class QImage of the Qt library has been made. This function indicates an error if the current file cannot be read or if the image format is not recognized.

2.2.3. CONVERSION OF IMAGE FORMAT

Image files are loaded by the corresponding Qt function: the constructor of the class QImage. We have developed a function (named “convertQImage2BinaryImage”) to convert Qt::QImage objects into Qgar::BinaryImage objects.

2.2.4. GENERATION OF INTERMEDIATE RESULTS FOR STUDY

Another conversion function (named “convertBinaryImage2QImage”) is available for converting Qgar::BinaryImage objects to Qt::QImage. This function is especially used for generating intermediate images for study in descriptor functions.

In fact, different types of intermediate results can be generated for study:

- Intermediate images.
- Log messages in different streams.
- A HTML file which recapitulates intermediate and final results.

2.2.5. GENERATION OF FINAL RESULTS

Final results are the signature file of the reference images and the matching results file both as CSV files. Their format is described in specification chapters Image signature composition of geometric descriptors and Image distance calculation from geometric descriptors respectively.

2.3. INTERFACE FUNCTIONS

2.3.1. INTEGRATION OF NEW DESCRIPTORS

Because several descriptors have to be developed and integrated into Rec software, their integration must be fast and easy.

2.3.2. INTERFACE WITH EXTERNAL SOFTWARE

Performance, facility to interface and multilanguage are important criteria to choose the right type of interface.. A standalone mode with a specific executable using this DLL is proposed.

2.3.3. INTERFACE WITH 2D CAD DATA

There have been some exchanges between WP2 and WP3 on how to make use of symbol recognition capabilities developed in WP2 for recognizing symbols found in 2D vectorial data provided by CAD systems.

WP3's "black box" converts Autocad data to Pivot. We suppose that, whatever blocks are present in the 2D vectorial file, they are found in this structure as a node in a tree. This is an advantage compared to bitmap drawings, as we have an explicit grouping of a number of graphical elements. On the other hand, the block does not necessarily correspond to a symbol; it can even be part of a symbol only, or anything else.

Interaction with WP2 tools are made from the Pivot editor. This editor is the most appropriate location for handling Pivot trees. We propose that the combinatorics of grouping several blocks into a candidate symbol be done within the Pivot editor:

- either because the user delineates a set of graphical features and says "recognize this symbol".
- or WP3 develops tools on its own to analyze block overlaps, graphical/geometrical continuities etc. to automatically delineate subsets of the Pivot tree which are to be submitted to symbol recognition.

Once we have a subset of the tree which **may** be a symbol, this can be submitted to WP2 symbol recognition tools. These tools are based on a combination of image-based features and geometric information.

This is a proposal for what information is to be provided to WP2 tools:

- geometric processing of basic geometrical elements is done by WP3 tools
- the geometrical elements resulting of the previous step are provided as a list of elements with all their characteristics
- the text present in the subset is provided with all its characteristics, including positioning
- the basic shapes directly present in the Autocad data are provided with all their characteristics
- the hierarchical structure (blocks) is provided

- in addition, a bitmap of the subset under consideration is also provided to WP2, so that WP2 tools can compute image-based features

With this information, WP2 tools can process the Pivot subset under consideration and send back to Pivot editor an ordered list of possible symbols with their recognition score.

One question remains open, that of the final matching to decide on the correct symbol

- in the present state of progress, the final matching is still partly based on image/template matching in WP2, and therefore not necessarily the most suitable for checking the actual symbol in the vectorial data used in WP3.
- as we are within an editor, the user keeps of course the option of choosing the correct symbol from the list

We have to consider various options here. This is in great part covered by what is called the semantic reference base in WP2. Intermediate draft of Deliverable D.2.6.2 deals with these problems, which are about how we represent our knowledge about the symbols in Fresh. The final matching question is directly related to the way knowledge and references are represented.

3. SOFTWARE DESIGN

3.1. SYSTEM ENVIRONMENT, DESIGN APPROACH AND STRATEGIES

Two important software contexts have to be taken into account to design this software:

- The use of the free (LGPL V2.1 licence) Qgar library. As seen in the function specifications, we often use the Qgar library functions to process complex or common algorithms.
- The interface with external software in order to allow the whole WP2 Fresh process.

Some difficulties were coming from these two contexts:

- The Qgar library runs on Linux in a C++ development environment.
- The external software (see task T2.2 of WP2) is running on MS Windows and is written with another programming language (Delphi).

To find the best compromise for software design and development, the following choices have been made:

- Development on current release of MS Windows, to facilitate integration with external software.
- Programming in C++ language to facilitate the use of functions and complex structures needed for Graphics Recognition present in the Qgar library.

The software can be developed on the free C++ IDE with Win 32 native release of GCC compilator (MINGW) or on the commercial IDE, MS VisualStudio .NET 2003 (7.1). These products have been retained after test of several others C++ environments.

Some secondary or dependant choices from previous have been made:

- The use of the free (LGPL V2.1 licence) Qt 3.3.5 library for image format conversion and for generation of intermediate images for test only. So the Qt library is not used in calculative functions but only in utility functions. Qt is a very portable and performing graphics library.
- The use of the standard library STL for C++ development which offers a lot of containers and very performing iterators and data manipulation algorithms.
- The possibility to run the software in a standalone manner to facilitate test and development and the possibility to interface the software with another.
- The input and output data file formats, which are:
  - INI file to save configurations data of the software and to initialize it.
  - Semicolon-separated CSV file format for reading and saving image signatures, and for saving image matching results.

3.2. SYSTEM STRUCTURE

The global diagram illustrating what has been described previousliy is given below:

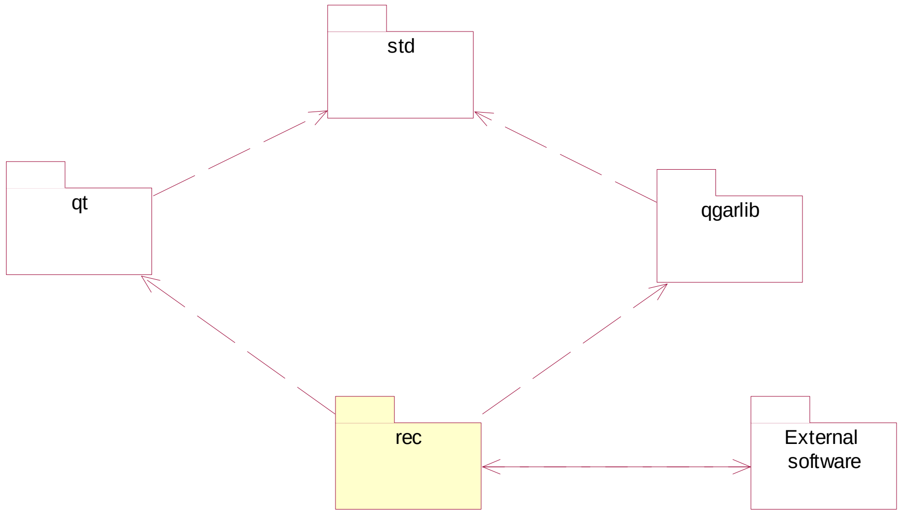


Figure 2: High-level component structure

The dependency between the Rec component and qt, qgarlib and std components have already been explained.

3.3. INTERFACE DESIGN WITH AN EXTERNAL SOFTWARE

The interface for using the Rec software is formed by a DLL file in order to allow external software programmed in another language to configure and launch the recognition process. The interface allows only standard data types to be exchanged in order to allow interfacing with different programming language. Thus instance of object classes are not exchange directly: an integer identifier is rather given to retrieve objects.

The general interactions are given in the following diagram:

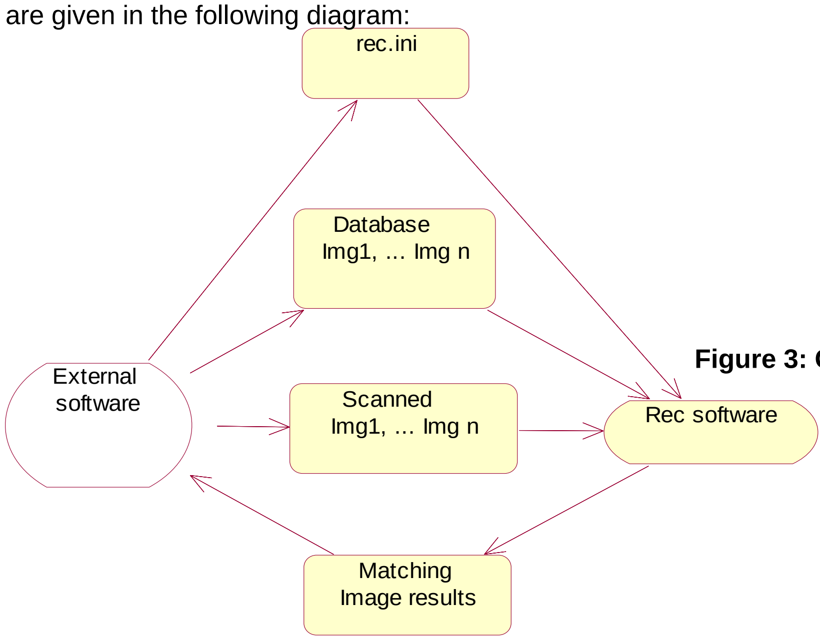


Figure 3: General interactions

3.4. PROCESS DESIGN

All the files concern calculative functions and are optional according to user choice to integrate or not some descriptors.

3.5. DESCRIPTOR INTEGRATION DESIGN

The following actions allow plugging a new descriptor for example:

- Create empty files
- Put following lines at least

3.6. CODING INTERMEDIATE RESULTS OF A DESCRIPTOR

Two types of intermediate results can be generated in the descriptor source code:

- Log messages which are drawn in a Qt Scrolled Text Window (if WithConsole is set in the INI file), MS Terminal Window (if executable is launched from MS Terminal Window) and always in the rec.log file localized in the same directory as the INI file rec.ini. Use the verbose boolean (already read in class Descriptor) to verify if user has activated or not this option.

Intermediate images can be generated in the output directory (based on the directory specified in the INI file rec.ini as an absolute or relative directory, the software adding a prefix .out to this directory). Use the outputLog Boolean (need to be read) to verify if user has activated or not this option.

4. SOFTWARE TEST PLAN

4.1. EXTERNAL INTERFACE TESTING

The external interface testing is implemented by a small program installed in <INSTALL_DIR>/recBDS3. This program is coded in DELPHI language to show how interfacing the C++ Rec DLL with external software not necessarily coded in the same language. So external interface testing simulates the external software and use Rec software.

In both main programs, there are a loop for managed execution and a loop for standalone execution. These loops can be desactivated at compile time to execute zero, one or many managed or standalone execution separately.

At compile time, the user can edit the source code of the test program and modify the source image to be recognized as well as the reference images from which source image should be recognized.

Success of the text will give an output that is ordered in the same way than the one given by a standalone execution in the HTML ranking table or in the CSV ranking file.

4.2. DESCRIPTORS TESTING

There are two ways to test the descriptor algorithms:

- Log images produced in the directory with the same pathname as the source image extended with an ".out" suffix. Generation of these images can be activated or desactivated per descriptor by setting the parameter "outputLog" in the "rec.ini" file. The geometric characteristics found by descriptor algorithms are highlighted inside intermediate log images by some different colors as showed previously in descriptor specifications. It allows users to decide if geometric characteristics found are pertinent or not. This type of test is visual and qualitative but easy to make.
- Log messages produced in the file "rec.log" at the same place as the executable file.

4.3. SIGNATURE COMPOSITION TESTING

Two tests allow verifying the correct composition of signatures:

- Verifying source and reference image signatures given in the HTML file or CSV files, by comparing each descriptor value with the number of highlighted characteristics shown in log images.
- Launch Rec software with the same value for <srcDir> and <refDir> in order to recognize some images with themselves. This test verifies signature composition, distance calculation and ranking.

4.4. DISTANCE TESTING

Two tests allow verifying the good composition of signatures:

- Launch Rec software to recognize source images with themselves. This test is already described in signature composition testing

- Manually compute the distance between a source image and a reference image and compare it with the value given by the software and print in HTML file or in CSV file.

A function in the HTML file is particularly practical to do this manual calculation. This function allows a direct comparison of signatures between a source image and one or two reference images. This function is available by clicking on a distance value on the ranking array.

In order to do this manual calculation, on should first refer to the appropriate method for distance calculation in configuration (INI file) and apply formulas given in Image distance calculation from geometric descriptors section.

4.5. EVALUATION OF REC SOFTWARE PERFORMANCE

We recall here some results generated by Rec software and interpreted in document D2.4.2. Some complements are described here.
The source symbols (unknown images to be recognized among reference images) are segmented by external software3.

4.5.1. EVALUATION OF REFERENCE IMAGES CLASSIFICATION

Quantifying the overall quality of any approach relies on the comparison of benchmarked data with experimental results. In our case, this consists of first establishing a list of known and their characterizations.

To achieve this measurement, Rec software computes the number of iso-signature classes (image with same signature) and their size (number of image with same signature in a class) in the HTML file.

4.5.2. EVALUATION OF SOURCE IMAGES CLASSIFICATION

The same type of results is produced in HTML file for source images.

4.5.3. EVALUATION OF RANKING

In order to assess the global recognition quality Rec software computes the ranking of the correct model (candidate image among reference images) for each source image in the HTML file.
Secondly statistics for three type of rank are given:

- Image rank of candidate image which is an interval because of other reference images at the same distance from the source images.
- Iso-signature rank of candidate image which is an interval too for the same reason and because two different reference image signatures can give same distance from source image.

Iso-distance rand of candidate image which is a simple value. Thirdly statistics for three type of iso-distance class size:

- Considering all iso-distance class and number of reference image in these classes.
- Considering only iso-distance class in first rank and number of reference image in these classes.
- Considering only iso-distance class in first rank and number of iso-signature in these classes.

4.5.4. EVALUATION OF TIME PERFORMANCE

The time performance evaluation has been made on a Win32 system computer with an Intel Pentium 4 Mobile at 1.86Ghz core frequency and with a dynamic RAM of 1Go (which is far enough for Rec software itself which takes not more than 32Mo). Time performance is given with and without regeneration of reference image signatures. Total time depends on number of image and of their size.

5. SOFTWARE USER'S GUIDE

The software has two different forms:

- The managed interface when Rec software functions are used by external software..
- The standalone executable when Rec software is launched directly by user without external software.

5.1. INSTALLATION AND INITIALIZATION

The Rec software has been developped on a Win32 platform. No test has been made for Linux platform.

The installation file, as of the writing of this document, is a ZIP file of approximately 45Mo. The content of this ZIP file is explained in section System Structure.

Before starting Rec software, user has to set parameters in the INI file <INSTALL_DIR>/rec.ini. The parameters are described in section Software requirements.

5.2. STARTUP AND TERMINATION

To start Rec software, user executes the file <INSTALL_DIR>/rec.exe without argument. A console window is opened or not depending of compile options or of INI file parameter "WithConsole" for Qt console. If a console window is activated, every log messages activated are printed in this console.

If the user does not close the console window or if there is not console window, the process terminates after some minutes.

If the process terminates abnormally, the message "SOME ERROR OCCURED, SEE ABOVE" is printed at the end of the LOG file "rec.log " and a console window is automatically opened whether or not it is activated.

If the process terminates normally and if no console is activated, no messages are given: the user must verify the file modification date of LOG file for example to know that process has terminated.

If the process terminates normally and if a console is activated, the Qt console is closed automatically or not depending of the INI file parameter "AutoExitIfConsole". If the window console is opened, it is not closed automatically at the end of the process.

5.3. FUNCTIONS AND THEIR OPERATION

Three main functions are available for user interface:

- The INI file rec.ini, to edit in a simple text editor like notepad, that allows user to configure the software (managed or standalone execution).

- A Qt console (or Windows default console) where log messages are printed during execution and allows user to stop the process before its normal termination by closing this window.
- An HTML file named <refDir>_<srcDir>.html, localized in <INSTALL_DIR>. This file includes all results present in CSV files, in LOG file and recall parameters used in INI file. This file includes many statistics for evaluation of the recognition process too. This file gives many interactive functions to analyse results. This function needs an HTML browser, and interactive means are well-known for web users.

5.4. ERROR AND WARNING MESSAGES

A list of error messages that can be occurred and corrective actions have been implemented:

- Concerning Descriptor initialization
- Concerning Signature composition
- Concerning distance computing
- Concerning utility functions

6. CONCLUSION AND PERSPECTIVE

Rec software is based on an algorithm which advantages have already been recalled in document D2.4.2 “Final report on symbol recognition with evaluation of performances”: it is an algorithm completely open and is continuously being improved. Current ongoing work concerns:

- Adding new descriptors :
  - T-junctions
  - Arrow detection
  - Parallel line detection
  - Arc segmentation (cf. published work by JP Salmon)
  - Multiple circle detection
- Adding a global selection algorithm in order to achieve a final, higher recognition rate than the current 95%-99%.