Implementation of biomedical image processing algorithms

PFC0071-2003

Wouter De Raeve
Wouter.DeRaeve@mail.be

June 19, 2003

Abstract

The easy way to design image processing algorithms in the Matlab programming environment needs to be compared with alternative ways. The drawbacks of the Matlab codes are dependency of the Matlab's engine and often slow execution. This implies that the developed algorithms can only be used within the original environment. Usage in, for example a hospital is either difficult or expensive. Moreover, the executation of the Matlab's algorithms is usually slower than those that would be developed in a programming language as C/C++. Those languages also carry the advantage to be easily ported between different platforms. The project consists of trying to convert the Matlab-algorithms into stand-alone applications, and to compare the different methods available to achieve this.

Preamble

This project was only made possible thanks to the support of some people and friends who I hold dear and to whom I owe a great deal. In particular I would like to thank:

- *Carlos Platero* and *dr. ir. Luc De Backer* for allowing me to complete this final project work.
- *My parents* for giving me the opportunity to go study abroad. This was a lifetime experience, and without their support and kindness none of this was possible.
- *Greet* for letting me go for four months and for all the trust and support she gave me.
- *Sylvie* for being a better guide I could ever be.
- *My friends* for accepting me who I am, for showing what friendship means, and for joining me in the many unforgettable moments they have given me.
- *My companions* in Madrid of the GVA, for touring me around the city and showing me Madrid outside the school.
- *Jeroen* for standing up with my constant nagging and for his comforting words when needed.
- *Ilse&Julie* for introducing us to Madrid, and for helping out the first difficult days, as well as joining in our great activities.
- *4EL* and especially *De bende van Oostmalle* for making the years at the Kaho Sint-Lieven as pleasant as possible.

A last word goes out to Frank Louwers, for hosting our pictures on `http://fotos.frankbruno.be`.

Contents

Chapter 1

Introduction

1.1 Goal of the project

The main goal of this final project is to study the possibilities of converting image processing algorithms, written in the Matlab-environment, to stand-alone applications [1], so they can be distributed without the requirement of Matlab or any other software on the system. This project will try to compare different possibilities, of which one will probably be used in the future at the GVA[2] lab. Therefore this project will also try to be a reference guide to beginning users as well as a problem-solving guide.

1.2 Global positioning of the project

The project takes place in the EUITI [3], part of the UPM [4]. The lab is part of the ELAI [5]. This project is one of many produced by the *Grupo Visión Artificial* (GVA), which is led by Carlos Platero Dueñas. GVA has since long time been working on acquisition, processing, visualization and calculation of images, medical images in particular. Currently, we're working for the *Hospital Ramón y Cajal de Madrid*, *the Fundación para la Hepatitis Virales* and the *Universidad Autonoma de Madrid, Facultad de Biologia*. Several projects are being completed by the time of writing. This is a short, incomplete list of projects running:

- Conversion of Matlab-code to C++
- Acceleration of image-processing through computer clustering
- Development of Matlab Graphical User Interfaces

[1] independent of used operating system
[2] Grupo Visión Artificial
[3] Escuela Universitaria de Ingeniería Técnica Industrial
[4] Universidad Politécnica de Madrid
[5] Electrónica, Automática e Informática Industrial

- Image transport through Java Dicom-toolkit
- Reconstruction of 3D-images out of 2D-slices

The project comes into the larger framework of image processing, and therefore should be seen in that context.

1.3 Structure and emphasis of this book

This book is the report of the project performed from the beginning of March until the end of June. It will explain the possibilities of converting the code written in Matlab to C++. The idea behind this book was not to produce a report as is, but as well try to act as a reference guide for future development. Therefore it will not only contain a comparison between the possibilities. A great part of this book will deal with how to use these possibilities and which problems you might encounter when developing through these methods. Therefore it will contain an installation guide for several platforms, small examples to be able to test a successful installation or to use as a starting point, as well as a short description and possible solution to installation, developing, compiling and running errors you might encounter throughout the process.

Chapter 2

Matlab & Matlab Compiler

2.1 Matlab

2.1.1 What is Matlab?

2.1.1.1 General description

MATLAB® is a high-performance language for technical computing. It integrates computation, visualization, and programming in an easy-to-use environment where problems and solutions are expressed in familiar mathematical notation. Typical uses include:

- Math and computation
- Algorithm development
- Modeling, simulation, and prototyping
- Data analysis, exploration, and visualization
- Scientific and engineering graphics
- Application development, including graphical user interface building

MATLAB is an interactive system whose basic data element is an array that does not require dimensioning. This allows you to solve many technical computing problems, especially those with matrix and vector formulations, in a fraction of the time it would take to write a program in a scalar noninteractive language such as C or Fortran.

The name MATLAB stands for matrix laboratory. MATLAB was originally written to provide easy access to matrix software developed by the LINPACK and EISPACK projects. Today, MATLAB uses software developed by the LAPACK and ARPACK projects, which together represent the state-of-the-art in software for matrix computation.

MATLAB has evolved over a period of years with input from many users. In university environments, it is the standard instructional tool for introductory and advanced courses in mathematics, engineering, and science. In industry, MATLAB is the tool of choice for high-productivity research, development, and analysis.

MATLAB features a family of application-specific solutions called toolboxes. Very important to most users of MATLAB, toolboxes allow you to learn and apply specialized technology. Toolboxes are comprehensive collections of MATLAB functions (M-files) that extend the MATLAB environment to solve particular classes of problems. Areas in which toolboxes are available include signal processing, control systems, neural networks, fuzzy logic, wavelets, simulation, and many others.

2.1.1.2 Where to get it?

Matlab is produced by the Mathworks-firm. Their homepage can be found at `http://www.mathworks.com`. Currently release 13 is the most up-to-date version of Matlab, version 6.5. Matlab is not cheap, but special student versions are available too, but yet with restrictions that severely limit the usage, especially when working with large data structures such as images. Therefore a full version is highly recommended. You can also download a trial version of the program.

2.1.2 Image processing with Matlab

Matlab provides built-in functions for image-processing. The Image Processing Toolbox extends the MATLAB(R) computing environment to provide functions and interactive tools for enhancing and analysing digital images an developing image processing algorithms. In addition, it facilitates the learning and teaching of image processing techniques in both academic and research settings.

Together, MATLAB and the Image Processing Toolbox provide scientists, researchers, and engineers with a diverse, flexible set of tool for solving complex imaging problems in disciplines such as aerospace/defence, astronomy, remote sensing, medical and scientific imaging, and materials science. Most functions are implemented in the open MATLAB language, letting you explore and customize existing toolbox algorithms or develop your own.

You can use the toolbox for the restoration of noisy or degraded images, image enhancement for improved intelligibility, blob analysis, and extraction and analysis of image data with image statistics and transforms, as well as to develop complete solutions to challenging image processing problems that involve multidimensional data sets.

Some of its key features are:

- Neighborhood and block operation

- 2-D filters, linear filtering, and filter design
- Image analysis, including pixel, region, and feature statistics and measurement
- Interactive GUI for control point selection
- Binary and grayscale morphology
- Spatial transformation
- Image registration, segmentation, and deblurring
- FFT, DCT, and radon transform
- Region-of-interest processing
- Multidimensional image processing
- DICOM import and export in addition to supported file formats in MATLAB
- Color space conversions

Matlab's Image Processing Toolbox provides fast ways to perform:

- Data Import and Export
- Image Analysis and Enhancement
- Image Manipulation
- Visualization

Latest version of the Image Processing toolbox is version 3. The homepage can be found at:
`http://www.mathworks.com/products/imageprocessing/`.

2.2 Matlab Compiler: General

2.2.1 Overview

2.2.1.1 What?

The MATLAB Compiler enables you to convert many MATLAB applications containing math, graphics, and graphical user interfaces to stand-alone C and C++ code. This code can be generated as stand-alone executables, shared libraries, or dynamic link libraries (dll's) for use on any MATLAB supported platform. The MATLAB Compiler has three components: the Compiler, the Math Library, and the Graphics Library.

You can take advantage of the MATLAB environment to quickly develop and prototype your applications and then use the MATLAB Compiler to automatically convert them into C or C++ source code. The MATLAB Compiler eliminates the tedious manual translation process and reduces development time for applications that run outside the MATLAB environment.

No thorough knowledge of C/C++ is needed, although a basic knowledge is indeed essential, as well as experience with the Microsoft Visual C/C++ Developing Environment, if development with that compiler is intended.

Some of its key features are:

- Lets you embed MATLAB based algorithms into C and C++ applications and distribute them freely
- Enables you to include MATLAB math, graphics, and GUIs for end-user applications
- Compiles many toolbox M-files for inclusion in your stand-alone applications
- Protects proprietary algorithms to prevent users from modifying code
- Allows you to compile, edit, and run your application from within Microsoft Visual Studio

Yet there are some restrictions. The Mathworks-website [1] mentions the following:

- M-files containing objects
- Calls to the MATLAB Java interface
- M-files calls to eval or input containing workspace variables

Yet during the project more restrictions and difficulties arose, as will be discussed later.

The Matlab Compiler's home page can be found at
`http://www.mathworks.com/products/compiler/`

We have used the most recent version 3.0.

2.2.1.2 Why?

There are three main reasons to compile M-files:

1. To speed them up
2. To hide proprietary algorithms
3. To create stand-alone external applications

[1] http://www.mathworks.com

Compiled C or C++ code typically runs faster than its M-file equivalents because:

- Compiled code usually runs faster than interpreted code
- C or C++ code can contain simpler data types than M-files. The MATLAB interpreter assumes that all variables in M-files are matrices. By contrast, the MATLAB Compiler declares some C or C++ variables as simpler data types, such as scalar integers; a C or C++ compiler can take advantage of these simpler data types to produce faster code. For instance, the code to add two scalar integers executes much faster than the code to add two matrices.
- C can avoid unnecessary array boundary checking. The MATLAB interpreter always checks array boundaries whenever an M-file assigns a new value to an array. By contrast, you can tell the MATLAB Compiler not to generate this array-boundary checking code in the C code. (Note that the i switch, which controls this, is not available in C++.)
- C or C++ can avoid unnecessary memory allocation overhead that the MATLAB interpreter performs.

Compilation is most likely to speed up M-file functions that:

- Contain loops
- Contain variables that the MATLAB Compiler views as integer or real scalars
- Operate on real data only

Compilation is not likely to speed up M-file functions that:

- Are heavily vectorized
- Spend most of their time in MATLAB's built-in indexing, math, or graphics functions

MATLAB M-files are ASCII text files that anyone can view and modify. MEX-files are binary files. Shipping MEX-files or stand-alone applications instead of M-files hides proprietary algorithms and prevents modification of your M-files.

2.2.2 How the compiler works

The Matlab COMPILER does not only convert written matlab functions into C/C++-code, but as well into so called MEX-files, COM-libraries,... But since the project only dealt with converting to stand-alone programs through C/C++-code, I will only deal with those.

Depending on the option used, Matlab can convert user m-files into C-code or into C++-code. Using the appropriate option, the Matlab Compiler also generates additional files in order to be able to distribute the algorithm as a stand-alone program.

The following example to show how the Matlab Compiler works is taken from the Compiler reference guide.

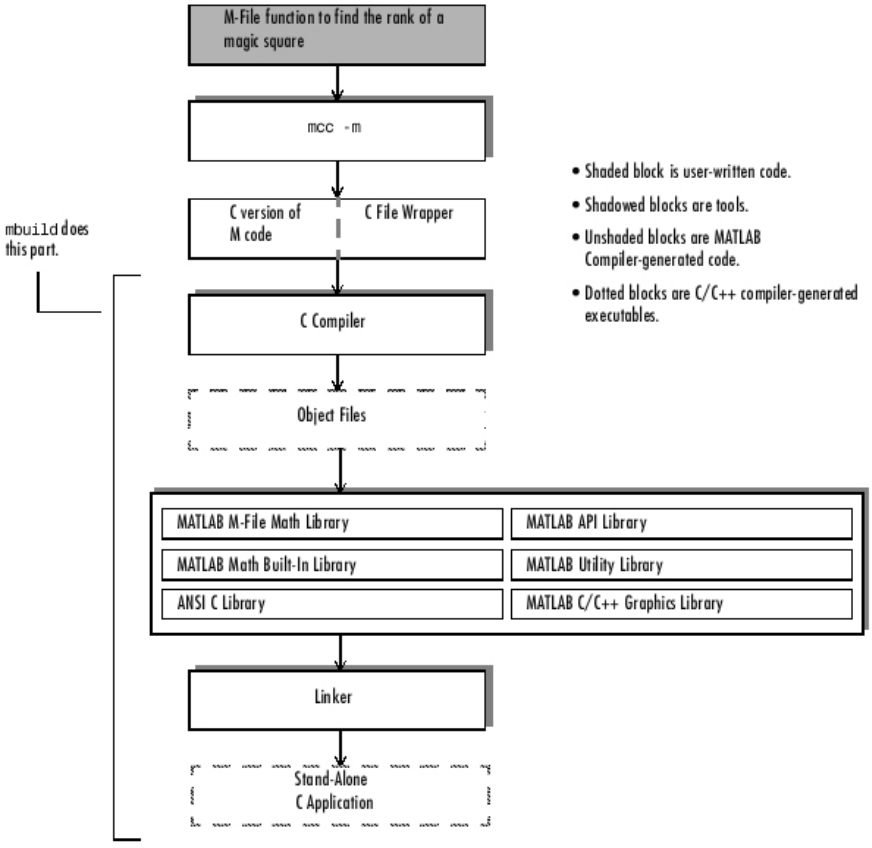


Figure 2.1: How the Compiler Works

The MATLAB Compiler, when invoked with the -m macro option, translates input M-files into C/C++ source code that is usable in any of the supported executable types. The Compiler also produces the required wrapper

file suitable for a stand-alone application. Then, your ANSI C compiler or C++ compiler compiles these C/C++ source code files and the resulting object files are linked against the MATLAB C/C++ Math and Graphics Libraries, which are included with the MATLAB Compiler.

2.2.3 Limitations and restrictions

The following limitations are the limitations and restrictions as mentioned by Mathworks. Later on in the book a few other problems and limitations will be discussed. Therefore this part should not be considered as foolproof.

Almost all functionality, available in Matlab is supported by the latest version of the Matlab Compiler, yet there still are some restrictions.

This version of the Compiler cannot compile

- Script M-files
- M-files that use objects
- Calls to the MATLAB Java interface
- M-files that use input or eval to manipulate workspace variables[2]
- M-files that use exist with two input arguments, for example:

  ```
  exist('foo','var')
  ```

  The single variable form works for filenames and functions only.
- M-files that dynamically name variables to be loaded or saved. This example is disallowed by the Compiler:

  ```
  x= 'f';
  load('foo.mat',x);
  ```

- M-files that load text files, for example:

  ```
  load -ascii sampling1
  ```

The Compiler cannot compile built-in MATLAB functions (functions such as *eig* have no M-file, so they can't be compiled). Note, however, that most of these functions are available to you because they are in the MATLAB Math Built-in Library (libmatlb).

In addition, the Compiler does not honor conditional global and persistent declarations. It treats global and persistent as declarations. For example:

[2]input and eval calls that do not use workspace variables will compile and execute properly

```
if (y==3)
persistent x
else
x = 3;
end
```

2.2.3.1 Stand-alone Applications

The previous rules also apply to stand-alone applications, yet there are some other limitations that should be accounted for when developing. The following table, as presented by Mathworks, gives a list of incompatible functions.

add_block	add_line	applescript	assignin
callstats	close_system	cputime	dbclear
dbcont	dbdown	dbquit	dbstack
dbstatus	dbstep	dbstop	dbtype
dbup	delete_block	delete_line	diary
echo	edt	errorstat	errortrap
evalin	fields	fschange	functionscalled
get_param	hcreate	help	home
hregister	inferiorto	inmem	isglobal
isjava	isruntime	java	javaArray
javaMethod	javaObject	keyboard	linmod
lookfor	macprint	mactools	methods
mislocked	mlock	more	munlock
new_system	open_system	pack	pfile
rehash	runtime	set_param	sim
simget	simset	sldebug	str2func
superiorto	system_dependent	trmginput	type
vms	what	which	who
whos			

Table 2.1: Unsupported functions in Stand-alone mode

2.2.3.2 Fixing Callback Problems: Missing Functions

The Mathworks-documents also mentions problems concerning so called callback problems. Since I did not run into such problems, I will not go into this further, but instead would direct you to the Compiler Reference documents available on the Mathworks website[3].

[3] http://www.mathworks.com/products/compiler

2.3 Matlab Compiler: Installation

2.3.1 Windows Installation

This section of the book will describe the installation and testing procedure for using Matlab Compiler in conjunction with the Windows-operating system. It will especially focus on installing Matlab Compiler when Microsoft Visual C++ version 6.0 is installed, yet where possible a short description on how to install it with other c/c++-compilers will be given.

2.3.1.1 System Requirements

No other requirements are necessary than those needed by Matlab itself. The Compiler can not be installed if Matlab 6.5 Release 13 is not yet installed on the system. In order to install Matlab itself, I direct you to the Mathworks-website[4], where detailed description is available.

In order to successfully be able to create a working stand-alone application, a supported ANSI C/C++ compiler is necessary.

According to the Mathworks-website, the following compilers are supported[5]:

- Lcc C version 2.4 (included with MATLAB). This is a C only compiler; it does not work with C++.
- Watcom C/C++ versions 10.6 and 11.0
- Borland C++ versions 5.3, 5.4, 5.5, 5.6, and free 5.5. (You may see references to these compilers as Borland C++Builder versions 3.0, 4.0, 5.0, and 6.0.) For more information on the free Borland compiler and its associated command line tools, see `http://community.borland.com`
- Microsoft Visual C/C++ (MSVC) versions 5.0, 6.0, and 7.0

Since I only worked with Microsoft Visual C/C++ version 6.0, the main focus of the book will lay on this compiler, which has its advantages and disadvantages (see later).

Important! If you want to use the Matlab Compiler with Microsoft Visual C/C++, it is important that Microsoft Visual C/C++ is installed into its default folder, since Matlab Compiler relies on parts of this compiler[6].

Attention! Applications generated by the Matlab Compiler in Windows are 32 bit applications, only usable in a 32 bit Windows-environment, and will therefore not run in other environments such as DOS or Linux.

[4] http://www.mathworks.com/support/topic/install/index.shtml

[5] A completer list can be found here: http://www.mathworks.com/support/technotes/1600/1601.shtml

[6] To be more precise: the *VC98*-folder should not be changed!

2.3.1.2 Installation of the Compiler

The Matlab Compiler comes with the original installation CD as provided by Mathworks when buying Matlab. The following steps only apply when the Compiler is not yet installed on the system. You can check whether the Compiler is already installed by typing `ver` at the Matlab command prompt. The Matlab Compiler and its version should appear in the list:

```
MATLAB Compiler                    Version 3.0        (R13)
```

When Matlab Compiler does not appear in the list, the following steps should be performed:

1. insert Mathworks Installation-disc, the welcome dialog automatically shows up
2. Enter your Personal License Password (PLP), which was sent by e-mail
3. Click yes after reading the license agreement
4. Enter your name and company name
5. This is the most important step.
   (a) Specify your current MATLAB installation directory as the installation directory
   (b) Deselect all but the product (or products) you want to add. By default, the installer lists all the products you are licensed to install, preselected for installation, not just the new products. In our case this means that the Matlab Compiler should be installed, as well as the Matlab C/C++-Math and the Matlab C/C++ Graphics library.

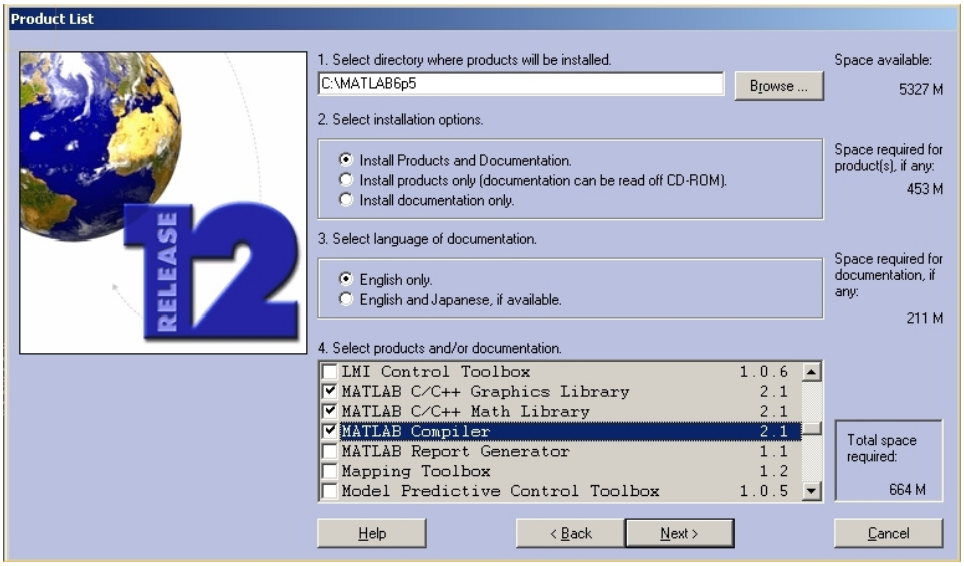


Figure 2.2: Compiler Installation Windows

2.3.1.3 Configuring the Compiler

In this step you will configure the Matlab Compiler to be used with the compiler installed on your system. This step assumes at least one supported compiler is installed on the system. It also assumes that you have sufficient read/write permissions on the computer.

As our goal is to create stand-alone applications only the part dealing with that conversion would be necessary, but in order to set up a more complete environment, and to be able to use all the functions within our programming environment, it is important to configure the creation of MEX-files[7] as well.

Configuring Compiler for creation of MEX-files

- In the Matlab Command Window, type `mex -setup`. The following question will be asked:

  ```
  Please choose your compiler for building external
  interface (MEX) files:

  Would you like mex to locate installed compilers [y]/n?
  ```

- Answer with 'y', and a list of available supported compilers on the system will be shown, such as the following example:

  ```
  Select a compiler:
  [1] Lcc C version 2.4 in C:\MATLAB6P5\sys\lcc
  ```

[7]MATLAB Executable

```
[2] Microsoft Visual C/C++ version 6.0 in
C:\Archivos de programa\Microsoft Visual Studio

[0] None

Compiler:
```

- Select the appropriate compiler, in my case the correct answer would be 2. Verification will be asked:

```
Please verify your choices:

Compiler: Microsoft Visual C/C++ 6.0
Location: C:\Archivos de programa\Microsoft Visual Studio

Are these correct?([y]/n):
```

- If the choice is indeed correct, type 'y'. The configuration will now begin. If all turns out well, a text very similar to the following should be shown:

```
The default options file:
"C:\WINNT\Profiles\username
\Application Data\MathWorks\MATLAB\R13\mexopts.bat" is
being updated...
Installing the MATLAB Visual Studio add-in ...

  Updated ...
```

- Now the Compiler is successfully configured.

Configuring Compiler to create stand-alone applications

In order to be able to create stand-alone applications, Mathworks has developed an utility to easily customize the configuration and build process. This tool, which can be invoked by typing `mbuild` at the command line is the easiest way to configure the Matlab Compiler to create stand-alone applications.

Mbuild can be used as a complete command-line controlled utility, but when using Microsoft Visual C/C++, no knowledge of the available options is necessary, only a basic configuration.

The configuration for mbuild is very similar to that of the one above for the creation of mex-files. I will here only shortly restate the necessary steps, and clarify where things are different.

- invoke `mbuild -setup` at the matlab command line
- select 'y' in order to let Matlab search for available compilers
- choose the correct compiler (ic Microsoft Visual C/C++)
- Press 'y' when you have selected the correct compiler
- If all finishes well, you'll get basically the same message as before
- Now you have to save the new info into the Matlab Path-library. You can do this by invoking the following commands on the Matlab commond prompt:

```
cd (prefdir)
mccsavepath;
```

Now the Matlab Compiler itself is fully configured.

2.3.1.4 Testing Installation within Matlab

The actual core of the C/C++-code generation by matlab, is provided by the command *mcc*. But since the rest of this book will be dealing with the creation of applications using the Microsoft Visual C/C++ environment, where thorough knowledge of the options of this command is unnecessary , I will only use it this time to test the installation of the Compiler.

Testing of the installation can be done by following the following steps:

1. In the Matlab environment, go to the following folder:
   `C:\MATLAB6p5\extern\examples\cmath`
2. Make sure the ex.c-files are available (1-6).
3. Test by invoking `mbuild ex1.c`
4. If no errors occured, a exe-file was created which can be run
5. If any errors did occur, check *Initial troubleshooting* below
6. In the Matlab environment, go to the following folder:
   `C:\MATLAB6p5\extern\examples\compiler`
7. Make sure the file `hello.m` is in the folder
8. Test it by entering `hello` at the Matlab command prompt
9. Test the Compiler by invoking the following command:

   ```
   mcc -m hello
   ```

10. If the Compiler is installed and configured correctly, the following files should have been generated:

    ```
    hello.c       hello.h       hello_main.c
    hello.exe     hello.m
    ```

11. If this gives errors, see below *Initial troubleshooting*.

Initial Troubleshooting

If for any error with these examples, try going through the installation procedure again.

If the error occurring is not license-related, and you have tried reinstalling, check the Matlab Compiler Reference guide for further trouble-shooting.

The Guide can be found here:
`http://www.mathworks.com/access/helpdesk/help/pdf\_doc/compiler/compiler3.pdf`

The sections applicable are:

- Mex Troubleshooting, page 61
- Troubleshooting the Compiler, page 63 and 111
- Troubleshooting mbuild, page 109

2.3.1.5 Installation of Microsoft Visual C/C++ Add-In

MathWorks has provided the Matlab Compiler with built-in additional support for the Microsoft Visual C/C++ developing environment. This through a plug-in, which after installation is available in the developing environment, and which lets you easily integrate m-files into a Visual C/C++-project.

If you have already performed the previous two installations, in short being *mex -setup* and *mbuild -setup*, and you have selected the Microsoft Visual C/C++ compiler, the necessary files are already installed.

If not I will here repeat a short retake:

1. To build MEX-files with the add-in for Visual Studio, run the following command at the MATLAB command prompt:

   ```
   mex -setup
   ```

   Follow the menus and choose either Microsoft Visual C/C++ 5.0 or 6.0. This configures mex to use the selected Microsoft compiler and also installs the necessary add-in files in your Microsoft Visual C/C++ directories.

2. To build stand-alone applications with the MATLAB add-in for Visual Studio (requires the MATLAB Compiler and the MATLAB C/C++ Math Libraries), run the following command at the MATLAB command prompt:

   ```
   mbuild -setup
   ```

   Follow the menus and choose either Microsoft Visual C/C++ 5.0 or 6.0. This configures mbuild to use the selected Microsoft compiler and also installs the necessary add-in files into your Microsoft Visual C/C++ directories. (It is not a problem if these overlap with the files installed by the mex -setup command.)

3. For either mex or stand-alone support, you should also run the following commands at the MATLAB prompt:

   ```
   cd(prefdir); mccsavepath;
   ```

   These commands save your current MATLAB path to a file named mccpath in your user preferences directory. (Type prefdir to see the name of your user preferences directory.)

   This step is necessary because the MATLAB add-in for Visual Studio runs outside of the MATLAB environment, so it would have no way to determine your MATLAB path. If you add directories to your MATLAB path and want them to be visible to the MATLAB add-in, rerun the cd and mccsavepath commands shown in this step and replace prefdir with the desired pathname. *This is necessary when you use external functions!*

Now you are ready to add the plug-in into the Microsoft Visual C/C++ Developing Environment. The plug-in will appear as a toolbar in the environment, as well be able to select a Matlab Project in the New Project Dialog box.

I will here provide a step-by-step procedure.

1. Select **Tools** - **Customize** from the MSVC[8] menu.

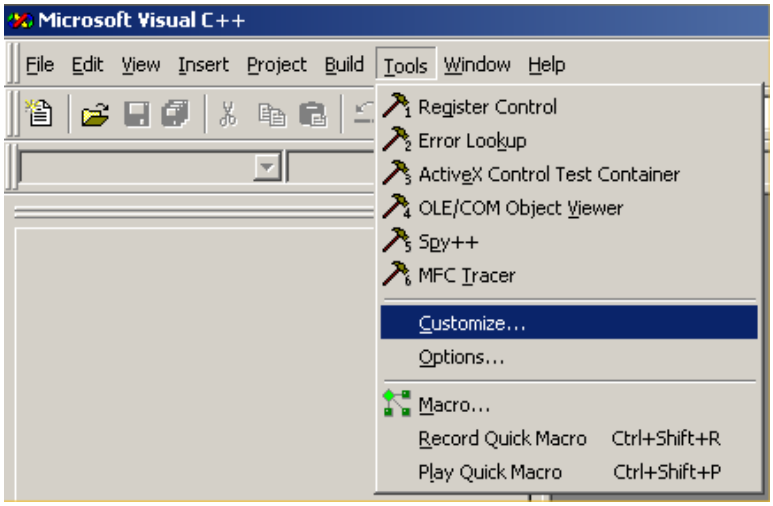


Figure 2.3: Tools-Customize

2. Click on the Add-ins and Macro Files tab.

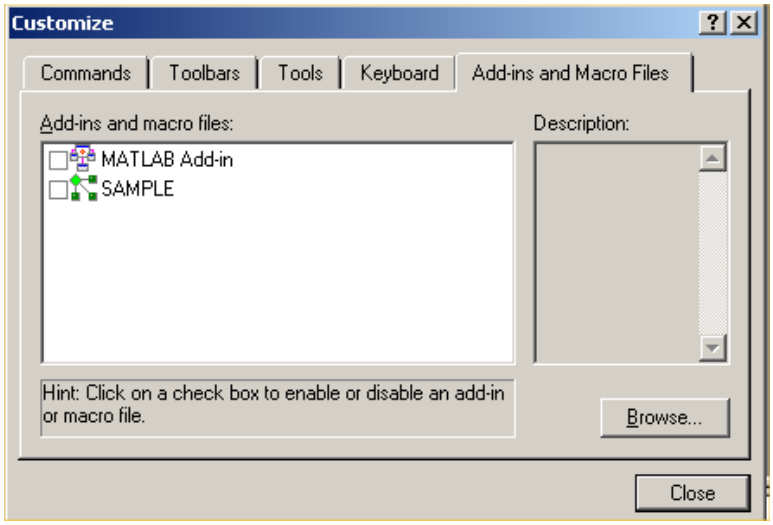


Figure 2.4: Add-ins Macro Files

3. Select MATLAB for Visual Studio on the Add-ins and Macro Files list and click Close. The floating MATLAB add-in for Visual Studio toolbar appears. Selecting MATLAB for Visual Studio directs MSVC to automatically load the add-in when you start MSVC again.

[8] I will call Microsoft Visual C/C++ 6.0 MSVC from now on

Figure 2.5: Add-In toolbar

Now should be able to select a Matlab Project from the New Projects dialog, as is shown in the picture below:

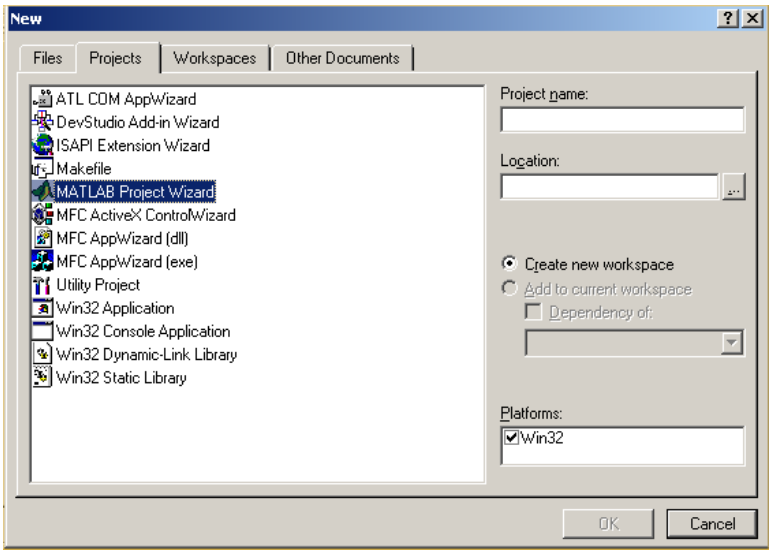


Figure 2.6: New Matlab Project 1

When you see this screen, the installation of the plug-in was successful. The following section will test the installation.

2.3.1.6 Testing of Microsoft Visual C/C++ Add-In

Before trying to test the integration with Microsoft Visual C/C++, it is advisable to test the working of the Matlab Compiler within the Matlab environment, as is described in section 2.3.1.3 on page 15.

In order to test the correct installation, it is advisable to first try to compile and run one of the examples available.

I will here provide a step by step testing procedure:

1. Create a new Matlab Project by selecting File - New - Projects - Matlab Project Wizard and give the project a name, eg 'test'.

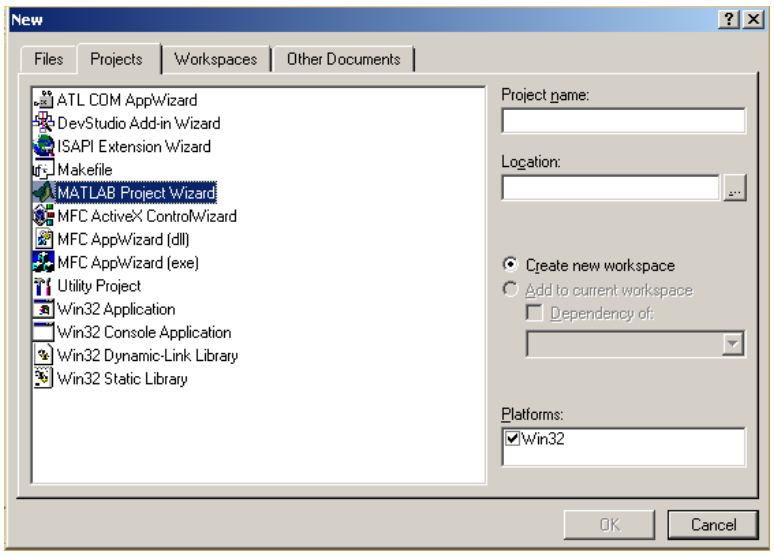


Figure 2.7: New Matlab Project 2

2. Select the language you would like to use, at this moment both languages can be chosen. When developing later on, problems can occur when selecting one or the other, in that case the choice will be important. See the section on *Incompatible functions* for more information on this. Other options should be in the default state, as these are the ones necessary for most applications.

Figure 2.8: Matlab Project Wizard

3. When pressing finish, you will be able to select your source m-file. For this example, we will use the file 'flames.m', available in the folder: *C:\MATLAB6p1\extern\examples\sgl*

4. It will then show a report file, produced by the Matlab Compiler. If no errors turn up in this section, you can proceed. With higher complexity programs, errors could occur here, which will be discussed later on.

5. Now you can build the application by pressing the build button in MSVC. It will then start compiling all the necessary files. The result should look like this:

```
----------Configuration: test - Win32 Debug--------
Compiling...
flames.cpp
hot.cpp
flames_mainhg.cpp
Linking...

test.exe - 0 error(s), 0 warning(s)
```

6. Copy the file 'flames.mat' to the folder of your MSVC-project.

You can now run the program by invoking it through a command-prompt. An animation should appear.

2.4 Developing in Windows/MSVC

2.4.1 Preparing your m-file

Before trying to compile any of the functions you have written, make sure you have installed and configured the Compiler correctly. This can be checked by compiling any of the examples, as described in the testing section 2.3.1.3 on page 15.

These should compile and run properly before continuing.

This section will describe how to prepare your m-files, so that successful compilation is more likely. Following these restrictions will save already a great deal of problems. Although the Matlab Compiler provides a very easy and fast creation of stand-alone applications, it is most of the times not a click-and-go operation. This book is intended to prevent and solve as much problems as possible.

2.4.1.1 Input Restrictions

When trying to develop a stand-alone application, this is one of the aspects that most likely will need adjusting. In regular matlab-functions, as provided by m-files, any kind and number of inputs can be used. This makes every function very flexible and easy to understand from the outer world. People do not need to know anything of the internal elements of the functions. A good help function on the meaning and type of inputs is the only essential in order to use the function properly.

In the development of a stand-alone application, that can be run from the command prompt, some of this flexibility and transparency will have to be given up. Because of this, probably more knowledge of the internal functioning will be necessary, although a thorough help file and explanation of the function should be sufficient.

The missing of this flexibility and transparency can be solved as well by developing a gui[9] or shell around the main program, which will only be the algorithm itself. In that case, no knowledge at all of the algorithm is needed, the user is not aware of the existence of the algorithm in its present form.

A simple example will clarify.

The following example takes two inputs, in1 and in2, which are supposed to be integers. Then the content of these inputs is shown, as well as the sum of these inputs. The Matlab code for this simple function:

```
function in(in1,in2)

in1
in2
in1+in2
```

And the result:

```
>> in(5,10);

in1 =
     5
in2 =
```

[9] Graphical User Interface

```
    10
ans =
    15
```

Which is of course correct.

When compiling this function within MSVC[10], the resulting file is called *input.exe*, determined by the name given to the project, not by the name of the function.

Normally you would then expect that the following command at the command-prompt would give the proper results:

```
input 5 10
```

This because of the way we are used to working with DOS, where every command carries its inputs through the command-line. For example:

```
copy sourcefile destinationfile
```

Yet this returns a strange and incorrect result:

```
C:\Wouter\VC++6projects\examples\input>input 5 10

in1 =

5


in2 =

10


ans =

   102   101
```

Which is very strange, because it seemed to have loaded the inputs correctly, as shown by the correct displaying, but the sum is calculated incorrectly. This is because the inputs are treated as strings, not as integers.

Solution

The solution to this problem is very simple yet also very uncomfortable: not using inputs. But inputs are inevitable in a function, so way around needs to be used.

A solution one might think of, is using the built-in "load" function in Matlab. Yet this function can not be used because of a bug in the Matlab Compiler. See more in the section about incompatible functions about this problem.

A viable, yet not ideal solution which I have used in developing the stand-alone applications, is the following. Matlab offers I/O[11] functions very similar to those available in C. Functions as to open/read/write/close files. In this lies the solution

[10] Microsoft Visual C/C++

[11] Input/Output

used. Instead of giving the inputs as arguments at the command line, the inputs are passed through a file. Then the Matlab functions to read from files is used to read the inputs into their respective variables. Yet the file name of the file that contains the values of the inputs cannot be passed either. Therefore this file name will be a fixed string, and is hardcoded into the program. Thus after compilation, the file name of that file is chosen and cannot be altered.

Advantages:

- Inputs possible

Disadvantages:

- Non-ideal workaround
- Not flexible (fixed filename)
- Non-transparent (user needs to know what inputs the algorithm expects)

However, this workaround is only necessary in the main function! This means only the function that will be the resulting executable. Any functions called within the main function can carry inputs as they were originally intended to.

Thus in our Matlab-code the following adjustments are made:

```
function in()

fid=fopen('input.txt','r');
in=fscanf(fid,'%i\n')
fclose(fid);

in1=in(1,1)
in2=in(2,1)

in1+in2
```

The fixed filename where it will read its variables from is called 'input.txt', but can of course be any filename. A short description of the used functions:

fopen Opens the file for reading, indicated by the 'r'-switch

fscanf Reads formatted from the file identifier fid into the variable

fclose Closes the file

The file 'input.txt' will contain the values:

```
5
10
```

This gives the correct result:

```
C:\Wouter\VC__6P~1\examples\in2>in2

in =

     5
```

```
    10


in1 =

     5


in2 =

    10


ans =

    15
```

Yet this solution is meant when the inputs are digits. When the inputs are strings, such as filenames, the adjustment to the algorithm is even smaller:

```
fid=fopen('input.txt','r');
in1=fgetl(fid);
in2=fgetl(fid);
fclose(fid);
```

The variables will now be strings in the matlab environment.

If you have adjusted your code like this, it is necessary that the file *input.txt* is located in the working directory, that is in the directory your executable is located. If the executable cannot find any of the files it has to open, whether it is a text-file, an image or something else, the following error will be displayed:

```
Exception!  File: handler.cpp, Line: 73
    Invalid file identifier.
```

This is the first thing to keep in mind when trying to compile. In short:

- No inputs in main function
- Workaround by using fixed text-file
- Adjusting matlab code by using I/O functions

2.4.1.2 Output Restrictions

Next to the input restrictions, the restrictions imposed by the Compiler on possible outputs, thus the return values of the functions, are more logic and easier to understand, but evenly uncomfortable as those implied on the inputs.

When writing algorithms in Matlab-code, it is not necessary to know in advance which values you will return to the main program, nor of which type they are. This way it is possible to return integers, doubles, strings, arrays or even combinations of them. Thus a function like
function [ImgIn,ImgOut,ContVirus,Err] = ContVirusC(), which would return an

array, containing two other arrays and two integers, would impose no problems. These returned values can then be used within the matlab environment to perform other actions.

Yet when developing a stand-alone application, that is restricted by the possibilities the operating system, ic DOS[12] offers, such possibility nor exists nor is necessary.

Not necessary, because the main reason for multiple returns in the matlab environment is the possibility to reuse those variables with other functions, within the matlab environment. Yet the stand-alone application is outside that environment, and therefore those other functions can not be used. Even when those other functions would as well be compiled into a stand-alone application, this would not be necessary, since the functions can not take inputs as they would have in matlab, as explained in the first section of this chapter.

Yet often the results are needed for later reference, or for usage within other programs or algorithms. This means that there has to be a way to save the results of the algorithms, which would normally be saved by returning them to the main program, in an other way so that they remain accessible when the application or algorithm has finished.

The solution for this problem depends on which value(s) the algorithm returns. When there's only one return, and it's an integer within the limits of 255, no adjustment to the algorithm needs to be made, since DOS can get one return from a program. Yet DOS does not see this as a real variable, but as an error code. Therefore this solution can only be used when the algorithm is used within another program that can catch the error return code, such as a gui written in Java.

An example will make this clear:

```
function out=out()

out=15;
```

This of course returns the value 15. When running this function in matlab, a variable with value 15 can be used within the matlab environment.

```
>> val=out

val =

    15
```

When compiling this function in MSVC, and running it in a DOS-prompt, we get no result, since it cannot be inserted into a variable:

```
C:\Wouter\VC++6projects\examples\out>out

C:\Wouter\VC++6projects\examples\out>
```

The result the function returned can be found in the errorlevel code, which can be displayed by typing *echo %errorlevel%* at the command prompt:

[12] Although we are using windows, the application generated by the compiler is basically a dos command-line based application, and therefore subsides to the restrictions dos implies

```
C:\Wouter\VC++6projects\examples\out>out

C:\Wouter\VC++6projects\examples\out>echo %errorlevel%
15
```

Yet this is not a good solution, because

1. You need an external program that catches the errorlevel code
2. The errorlevel code is limited to 255, so results bigger than 255 can not be stored

In the end this a faulty workaround, and therefore following solution should always be followed.

The solution needed when multiple return values, other than integers are needed lies in the method that is used by any program since long time to save data, files. The possibilities matlab offers for I/O with files makes it possible to save the data to a file. A full description on the possibilities Matlab offers on File I/O can be found at the Mathworks website[13].

For the algorithms used in the GVA-lab, which for now only return arrays of integers and image files, only a the following functions are needed:

imwrite Writes image arrays to a file

fprintf Writes ascii data to a text file

Yet this has some consequenses. The user that would like to use the results from the algorithm later on, has to know the exact format in which the results were written to the outputfile. This problem can be solved by issuing a clear description on how a certain variable type is written.

- Images will be written in the JPG[14]-format.
- Arrays of integers/doubles will be written as following into the text file (eg a 4x3-array)

  ```
  1 54 244 12
  48 47 12 31
  12 14 1 36
  ```

Every application will create its own output file, most of the time a text file. The filename of this file can be hardcoded, chosen by the developer, or can be chosen by the user, which of course then should be considered as an input, with the consequenses as described in the first section of this chapter.

To summarize:

- No outputs as in the Matlab-environment
- Single integer output smaller than 255 can be treated as errorlevel code in dos (not recommended)
- Return values or results should be stored in a file
- Results file can be hardcoded or chosen by the user

[13] http://www.mathworks.com/support/product/ML/tech-note/io/page1.shtml
[14] Joint Picture Expert Group

2.4.1.3 Documented Limitations

Before trying to compile your algorithms, check your code to find in your algorithm any functions that might be in the list of documented restrictions imposed by the Matlab Compiler. For a list of those functions, see section 2.2.3 on page 9.

2.4.1.4 Incompatible functions

Later on in this book a list of incompatible functions, functions that will generate errors during conversion, compile or runtime, is given. Also the solution to the problems with these functions is given.

Therefore it is advisable as a prevention rule to already perform the actions stated in the solutions. The solutions stated there are applicable to a wide range of functions, and thus can help prevent a lot of problems.

An example of an algorithm that was prepared fully for conversion can be found in the back of the book. There you will find the original algorithm, thus containing the code as it was originally run in the Matlab-environment, plus the algorithm, adjusted to be correctly converted. The code also contains comments, referring to any problems that occurred converting this algorithm, plus a short solution. The full solution can then be found in the respective parts of the book.

2.4.2 General developing tips

A lot of problems might occur when trying to convert the matlab-algorithms using the Matlab Compiler. The following sections will try to facilitate the development by issuing tips and providing a list of incompatible functions.

2.4.2.1 Successful installation

Make sure the installation was successfully finished. The installation procedure can be found at section 2.3.1 on page 11 and the testing procedure can be found at section 2.3.1.3 on page 15.

2.4.2.2 Preparation

Before compiling, remember that there are a few limitations to the compiler and some oddities might occur. Try preparing your matlab code by eliminating errors by going through these limitations. The section describing this can be found at 2.4.1 on page 21.

I will here provide a short list of these limitations:

inputs Matlab doesn't accept inputs as you would put them in the command line in DOS. Therefore use files to load your data.

outputs Neither does the Compiler support return values as you're used to in the Matlab Environment. Again the solution here lies in the usage of files to store your data.

documented limitations Mathworks has documented and listed a few of the limitations of the Matlab Compiler. Make sure none of your functions is listed there. These limitations can be found in this book at section 2.2.3 on page 9.

2.4.2.3 Usage of modules

This is a rule that applies to any development of software/algorithms. Whenever trying to develop an algorithm, try developing using re-usable modules. Meaning, when writing algorithms that often use the same set of routines, try combining those routines into a new function. Remember, within your algorithm, there are no restrictions concerning the input/outputs.

This has multiple advantages. First, it makes your code clearer, making it easier to find where a possible error occurs. It also makes it more transparent for future viewers to understand the working of the algorithm. And it enlarges the chance one of the modules created can be used in the future, thus saving developing time and costs.

Secondly, it rules out a lot of possible errors when compiling. Meaning, when using a function or module that already was used before in a successfully compiled project, the error occurring in the new project cannot be caused by that particular module.

An example of this is reading data from a textfile. As will be described later on in this book. The Matlab Compiler does not support the usage of the function *load* when trying to read ascii-files. Instead, you have to write your own methods to read the file, using low-level I/O-functions. Instead of writing this set of routines over and over again for every algorithms, it is advisable to write a separate function that reads all data from a file, which then can be used in every future algorithm.

2.4.2.4 Step-by-step developing

It is very unlikely when first trying to convert the matlab-code to C/C++-code, you will get a flawless compilation. Most of the times an error will occur, and it is not always easy nor very clear which function or which part causes the error. Thus the following method provides a time consuming but effective solution to this problem.

When the algorithm is prepared as described earlier in this book, it is advisable, when a full compilation turns out to be unsuccessful, to try compiling step-by-step.

Divide the algorithm into different parts, and try compiling first the first part. When this is successful, you can add the second part and so on. If a failure occurs at one part, try compiling that part line by line. That way it will be easy to locate the problem and then find a possible solution or work-around for it.

2.4.3 List of incompatible functions

When developing stand-alone applications using the Matlab Compiler, it will soon turn out not to be a click-and-go type of situation, which you would expect reading the glossary Mathworks presents on its Compiler. Unfortunately a lot of problems have occurred during the converting of the presented algorithms. A lot of these problems were repetitive, since a lot of the algorithms used at the GVA, use the same functions and methods of writing.

This section of the book will try to present a list of functions that will produce errors during compile-or run-time. This is only a list of the functions I came across to when converting the algorithms given to me during the time of this project. Therefore this list can not at any point be considered as complete or failsafe. Yet it can be considered as a reference to some of the problems that might occur. As well

it should be considered as a dynamic part of this book, since future developments might give new problems, to which the solutions will be put here as well.

Another thing to consider is that although some functions might not be listed here, but indeed were used in the algorithms, and therefore were considered as successful, could give problems in your case. This because the solution provided for one function, might as well be the solution to a problem with another function. This is especially the case with the solutions presented here: *Strel* Section 2.4.3.7 on page 41 and here: *Check* not found* in section 2.4.3.7 on page 42.

It is also often that one function uses another function, that is actually the base of the problem. An example of this is the function *Edge* that uses the *Imlincomb*-function. This will most of the times be visible at the error reported. Therefore the solution can be found at that function. Thus, although the erroneous function might not be in this list, it is advisable to try one of the solutions presented here.

I have tried to put as much as structure into this section as possible, so it would be possible for future readers to have a quick reference.

Before continuing, it is off course necessary that the reason for the error or problem lies in the usage of one of these functions. This means that successful installation should have been tested, as described in section 2.3.1.3 on page 15.

2.4.3.1 Edge

The function *edges* finds edges in an intensity image.

edge takes an intensity image I as its input, and returns a binary image BW of the same size as I, with 1's where the function finds edges in I and 0's elsewhere.

edge supports six different edge-finding methods:

- The Sobel method finds edges using the Sobel approximation to the derivative.
- The Prewitt method finds edges using the Prewitt approximation to the derivative.
- The Roberts method finds edges using the Roberts approximation to the derivative.
- The Laplacian of Gaussian method finds edges by looking for zero crossings after filtering I with a Laplacian of Gaussian filter.
- The zero-cross method finds edges by looking for zero crossings after filtering I with a filter you specify.
- The Canny method finds edges by looking for local maxima of the gradient of I. The gradient is calculated using the derivative of a Gaussian filter. The method uses two thresholds, to detect strong and weak edges, and includes the weak edges in the output only if they are connected to strong edges. This method is therefore less likely than the others to be "fooled" by noise, and more likely to detect true weak edges.

Type of error Compile-error when using C++

example program

```
ImgEnt = imread('cameraman.tif');
```

```
ImgBorder=edge(ImgEnt(:,:),'canny',.4,2);
imshow(ImgBorder)
```

Matlab output The Matlab output is the following picture:

Figure 2.9: Result of Edge-function

The problem occurs when trying to compile the project when using C++-code:

```
imlincomb.cpp
c:\edgetest\imlincomb.cpp(578) : error C2065:
'mlxClassname' : undeclared identifier
Error executing cl.exe.

edgetest.exe - 1 error(s), 0 warning(s)
```

It is clearly to be seen that the problem lies in the usage of the function *imlincomb*. Refer to the section on *imlincomb* to solve the problem.

2.4.3.2 Imdilate

The function *imdilate* dilates the image.

imdilate imdilate(IM,SE) dilates the grayscale, binary, or packed binary image IM, returning the dilated image, IM2. The argument SE is a structuring element object, or array of structuring element objects, returned by the strel function.

Here it can already be seen that the problems occurring when using this function are caused by the necessary usage of the *strel* function. Refer to the section on the *strel*-function on how to solve the problem.

2.4.3.3 Imfeature

The function *imfeature* computes feature measurements for image regions.

imfeature stats = imfeature(L,measurements) computes a set of measurements for each labeled region in the label matrix L. Positive integer elements of L correspond to different regions. For example, the set of elements of L equal to 1 corresponds to region 1; the set of elements of L equal to 2 corresponds to region 2; and so on. stats is a structure array of length max(L(:)). The fields of the structure array denote different measurements for each region, as specified by measurements. *measurements* can be a comma-separated list of strings, a cell array containing strings, the single string 'all', or the single string 'basic'. The set of valid measurement strings is included in the following table:

'Area'	'Image'	'EulerNumber'
'Centroid'	'FilledImage'	'Extrema'
'BoundingBox'	'FilledArea'	'EquivDiameter'
'MajorAxisLength'	'ConvexHull'	'Solidity'
'MinorAxisLength'	'ConvexImage'	'Extent'
'Eccentricity'	'ConvexArea'	'PixelList'
'Orientation'		

Table 2.2: Imfeature measurements strings

Type of error Incorrect result

example program

```
function imfeattest()


ImgEnt=imread('cameraman.tif');

ImgBorde=edge(ImgEnt(:,:),'canny',.4,2);
ImgObj = (ImgEnt(:,:)>150) & (~ImgBorde);

ImgEtiq=bwlabel(ImgObj);
stat = imfeature (ImgEtiq,'Area');
stat(1).Area
```

Matlab output

```
ans =

       29332
```

This function converts, compiles and builds well, but when running it in the dos-command prompt, the wrong result is shown:

```
C:\imfeattest>imfeattest

ans =

     []
```

Solution The solution to this problem was not so easily to be found. Some other people did have the same problem, for example here: `http://mathforum.org/epigone/comp.soft-sys.matlab/prelkalkhex/uy9tzp8l5.fsf@mail.mathworks.com` where is stated that the solution to the problem should be found here: `http://www.mathworks.com/support/solutions/data/27239.shtml`. Yet that address is no longer valid. Therefore another solution needed to be found.

When going through the manual of *imfeature* the following was stated:

```
Note    This function is obsolete and may be removed
in future versions. Use regionprops instead.
```

It seems the imfeature-function has been replaced by a new function called *regionprops*.

```
STATS = regionprops(L,properties) measures a set of properties for
each labeled region in the label matrix L.
```

Thus in the next attempt the *imfeature* function was replaced by the *regionprops*-function.

```
stat = regionprops (ImgEtiq,'Area');
```

which gives the same result as before:
After restarting the project, the correct result is displayed:

```
C:\imfeattest>imfeattest

ans =

       29332
```

2.4.3.4 Imlincomb

The function *imlincomb* computes a linear combination of images.

imlincomb Z = imlincomb(K1,A1,K2,A2,...,Kn,An) computes `K1*A1 + K2*A2 + ... + Kn*An` where K1, K2, through Kn are real, double scalars and A1, A2, through An are real, nonsparse, numeric arrays with the same class and size. Z has the same class and size as A1.

Type of error Compile-error

example program

```
I = imread('rice.tif');
J = imread('cameraman.tif');
K=imlincomb(2,I,2,J);
imshow(K)
```

Matlab output The output of this function is the picture shown below:

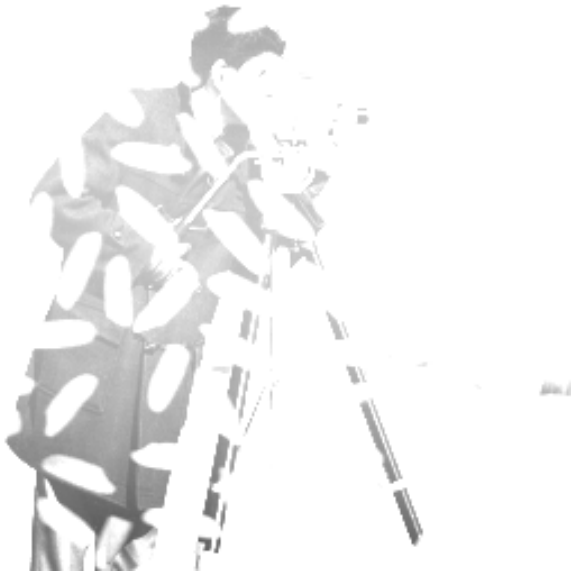


Figure 2.10: Imlincomb-result

This function is used quite often, and most of the times by other functions of the Matlab Image Processing toolbox. Functions that use *imlincomb* are:

- edge
- imadd
- imsubtract
- ...

The problem with this function only occurs when trying to convert to C++-code, not with C-code.

Conversion works flawlessly, but when trying to compile, the following error is shown:

```
imlincomb.cpp
c:\imlincombtest\imlincomb.cpp(578) :
error C2065: 'mlxClassname' : undeclared identifier
```

Again, the "solution", or better workaround, can be found on the Mathworks Website: `http://www.mathworks.com/support/solutions/data/32633.shtml`.

```
Solution:

This is a bug in the MATLAB Compiler 3.0 (R13) that has been
reported to our development staff for further investigation.

As a workaorund, please try generating a C stand-alone
instead of a C++ stand-alone.
```

When trying to compile into C-code, the following compile-messages are shown:

```
c:\imlincombtest\imread.c(1199) : warning C4761:
integral size mismatch in argument; conversion supplied
checkinput.c
c:\imlincombtest\imshow.c(1744) : warning C4761:
integral size mismatch in argument; conversion supplied
checknargin.c
checkstrs.c
images_private_imlincombc_mex_interface.c
iptgetpref.c
truesize.c
c:\imlincombtest\iptgetpref.c(461) : warning C4761:
integral size mismatch in argument; conversion supplied
c:\imlincombtest\iptgetpref.c(618) : warning C4761:
integral size mismatch in argument; conversion supplied
images_private_num2ordinal.c
c:\imlincombtest\truesize.c(1117) : warning C4761:
integral size mismatch in argument; conversion supplied

Linking...

imlincombtest.exe - 0 error(s), 5 warning(s)
```

In spite of the warnings, the program runs fine, that is, if the images used ('rice.tif','cameraman.tif') are copied to the working directory.

Thus, as a conclusion, it can be said that for now, it is better to convert into C-code, and not into C++-code. During the project, I did not occur any problems when changing to C-code.

2.4.3.5 Load

The function *load* loads workspace variables from disk.

load S = load(...) returns the contents of a MAT-file in the variable S. If the file is a MAT-file, S is a struct containing fields that match the variables in retrieved. When the file contains ASCII data, S is a double-precision array.

Options on the type of file that is loaded, can be given by an option switch. Possibilities are:

- *-mat* forces load to treat the file as a MAT-file, regardless of file extension. With -mat, load returns an error if the file is not a MAT-file.

- *-ascii* forces load to treat the file as an ascii-file, regardless of file extension. With -ascii, load returns an error if the file is not numeric text.

Type of error Runtime-error

example program

```
function loadtest()

s=load('input.txt','-ascii')
```

This forces load to treat the file 'input.txt' as an ascii-file. This isn't really necessary, since a file with the extension 'txt' is automatically treated as an ascii-file. 'input.txt' is a text file with the following contents:

```
5
10
```

matlab output

```
s =

     5
    10
```

This function, which provides an easy way to load in variables, gives errors when trying to run the compiled program. Conversion, compilation and building otherwise are flawless.

```
C:\loadtest>loadtest
ERROR: Can't open file C:\loadtest\input.txt.

EXITING
```

Yet, the file is in the working directory:

```
C:\loadtest>dir input.txt
 El volumen de la unidad C no tiene etiqueta.
 El número de serie del volumen es: BC5F-9EE5

 Directorio de C:\loadtest

24/04/2003  17:36                 5 input.txt
               1 archivos              5 bytes
               0 dirs   3.551.293.440 bytes libres
```

The problem lies again in a bug in the Matlab Compiler. This is stated clearly at the Mathworks website:
`http://www.mathworks.com/support/solutions/data/24103.shtml`.

```
This is a bug in the MATLAB Compiler where the compiled LOAD
function is unable to read ASCII files. This problem has been
reported to our development staff to be addressed in a future
release of MATLAB.
```

```
We suggest saving your data in a MAT file that can be read in
using LOAD in your stand-alone application. Please note that
if you take the return value of a LOAD command into a variable,
the data contained in the variable will be in a structure format.
Thus, you will have to refer to the particular variable
within the structure.

If you would like to use an ASCII file, please use the low
level File I/O commands like FSCANF.
```

The workaround suggested on the website is a possible workaround for MAT-files. But since we are working with plain ascii-files, that workaround is not possible. The file we would like to read is created by the external program *ejemplo2*, which is a plain dos-application which saves the results into a text file.

Therefore another solution is needed. As stated on the website, the usage of the low-level file I/O commands is necessary.

For a general guide on file I/O, refer to the Mathworks-website:
`http://www.mathworks.com/support/tech-notes/1600/1602.shtml`

The solution presented here is a general workaround around this problem, and consists of writing a function that is able to read in data from an ascii-file, as it would be done by the *load*-function within the Matlab environment.

loadascii

The function, which will be called *loadascii*, takes the filename of the ascii-file as input, and returns the matlab variable.

The source code for the function is pretty simple and straightforward:

```
function a = loadAscii(filename)
    fid = fopen(filename);
    if fid==-1
       error([filename ': File does not exist']);
    end
    numlines = 0;
    try
       while(  1 )
           tline = fgetl(fid);
           if ~ischar(tline), break, end
           numlines=numlines+1;
    end
       fseek(fid, 0, 'bof');
       a = fscanf(fid,'%g',[numlines inf]);
       [r c] = size(a);
       % The next two steps are required since MATLAB
       % reads the data in column major format
       % and fscanf reads data in row major format
       a = reshape(a,c,r);
       a = a';
       fclose(fid);
    catch
       fclose(fid);
```

```
        error([filename ': Error while reading file']);
    end
```

This function can be used every time trying to read in an ascii-file into a variable, and is therefore easier than to write your own routines every time.

A short explanation of the function. First, it checks whether the given file, which is known by the string-variable *filename*, exists. If not, it returns an error code.Then, it performs a loop as long as valid data is read in. It reads is data line by line. The data obtained there serves only to know the number of valid lines in the file. The actual reading is then performed outside the loop by the *fscanf*-statement. Due to the difference in interpreting data between Matlab and the fscanf-function, a reshape-statement is necessary. This reshapes the array a, and changes the number of columns to rows and vice-versa. Then, the array is transported to get the correct result.

Our little example can now be rewritten as:

```
function loadtest()

s=loadAscii('input.txt')
```

The loadAscii-function has to be in the same folder as the loadtest-function!

Now after restarting the project, the correct result is displayed:

```
C:\loadtest>loadtest

s =

     5
    10
```

2.4.3.6 Save

The *Save*-function is a bit odd in this series of incompatible functions. The function does not give any conversion problems, nor any compiling or linking errors or oddities. Yet, it turned out to act strangely when using it to store data into a text file, and then reusing it again in another program. This problem occurred with the specific algorithm used during this project. It might be that in other situations, no problems occur, yet it is advisable to use the solution here provided, since it seems to be more foolproof than the *Save*-function itself.

save saves workspace variables on disk.

save save filename var1 var2 ... saves only the specified workspace variables in filename.mat. Use the * wildcard to save only those variables that match the specified pattern. For example, save('A*') saves all variables that start with A. Options can be given by using several switches, listed in the table below:

option Argument	Result: How Data is Stored
-append	The specified MAT-file, appended to the end
-ascii	8-digit ASCII format
-ascii -double	16-digit ASCII format
-ascii -tabs	delimits with tabs
-ascii -double -tabs	16-digit ASCII format, tab delimited
-mat	Binary MAT-file form (default)
-v4	A format that MATLAB version 4 can open

Table 2.3: Options for "save"

The one we are particularly interested in is the *-ascii* option. This would allow us to easily load the stored variables again in an external program. But Mathworks imposes some restrictions on the usage of the storing of ascii-files:

- Each variable to be saved must be either a two dimensional double array or a two dimensional character array. Saving a complex double array causes the imaginary part of the data to be lost, as MATLAB cannot load non-numeric data ('i').
- In order to be able to read the file with the MATLAB *load* function, all of the variables must have the same number of columns. If you are using a program other than MATLAB to read the saved data this restriction can be relaxed.
- Each MATLAB character in a character array is converted to a floating point number equal to its internal ASCII code and written out as a floating point number string. There is no information in the save file that indicates whether the value was originally a number or a character.
- The values of all variables saved merge into a single variable that takes the name of the ASCII file (minus any extension). Therefore, it is advisable to save only one variable at a time.

Yet when using the compiler, even when keeping in mind those restrictions, problems may occur. I will show the possible problem with the program used in the algorithm.

This is the Matlab-code for the program:

```
ImgEnt = imread('cameraman.tif');

Hist = imhist(ImgEnt(:,:));
save hist.txt Hist -ASCII
dos('ejemplo2 hist.txt ResUmb');
```

This save the results of the histogram-function *imhist* to a file called hist.txt, which is then again read by an external program called **ejemplo2.exe** which calculates the threshold-values of the images, and saves those in the file called **ResUmb**.

When executing this code in Matlab, the following results are obtained:

hist.txt (only first 15 lines)

```
1.0000000e+000
  2.8000000e+002
  1.2290000e+003
  9.5200000e+002
  1.2160000e+003
  8.2900000e+002
  8.2400000e+002
  7.2200000e+002
  7.5800000e+002
  8.1700000e+002
  9.8000000e+002
  8.3800000e+002
  1.1720000e+003
  1.2080000e+003
  1.3530000e+003
```

ResUmb

```
108
109
108
29
99
36
140
175
184
167
110
0
7
1
```

So far, no problems. Now when the function is converted to a stand-alone program, and then executed, the following error happens:

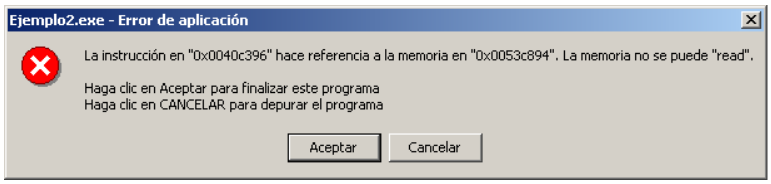


Figure 2.11: Savetest-error

This is not an error caused by the Matlab-program, but a "regular" windows memory error. This leads to the suspicion that the external program ejemplo2.exe

might be the cause of the problem. Yet it has run perfectly within the matlab environment. Therefore the problem lies in the save-function.

After a closer look it can be seen that the save-function is not well converted. The output-file of the save-function, hist.txt, is not saved correctly. When opening the hist.txt file, the following can be seen:

```
MATLAB 5.0 MAT-file, Platform: PCWIN,
Created on: Thu May 08 18:00:45 2003
```

Which is of course completely different from the result within the Matlab-environment. This causes the program "ejemplo2" to read in faulty results, and crash, hence generating the shown error dialog.

Thus we have to write our own routine of saving the histogram, or any other variable. This can be done by using the Matlabs internal I/O-routines.

De array hist is an array of 1 column, of which every element is saved on a new line in the hist.txt-file. Thus a simple loop, for the whole size of hist does the trick. Every time one value is taken from hist, and written on a new line in the file.

This is the code:

```
fid = fopen('hist.txt','w');
for i=1:size(Hist,1)
    fprintf(fid,'%i\n',Hist(i));
end
fclose(fid);
```

A little explanation:

fid = fopen('hist.txt','w'); Creates a file identifier, with the option 'write', if the file doesn't exist, a new file will be created, as in our case.

for i=1:size(Hist,1) Perform the loop for the whole size of Hist, thus taking every element.

fprintf(fid,'%in',Hist(i)); Write the value as an integer (%i), followed by a new-line character (
n). Of course other types, such as doubles or floats can be used.

fclose(fid); Close the file-identifier, so the file becomes accessible again.

Now the program runs fine and the correct result is calculated.

2.4.3.7 Strel

The function *strel* creates a morphological structuring element, used when opening and performing morphological functions on images.

strel SE = strel(shape,parameters) creates a structuring element, SE, of the type specified by shape, listed in the table below.

Flat Structuring Elements	
'arbitrary'	'pair'
'diamond'	'periodicline'
'disk'	'rectangle'
'line'	'square'
'octagon'	
Nonflat Structuring Elements	
'arbitrary'	'ball'

Table 2.4: Shapes by strel

Type of error Runtime-error

example program

```
function streltest()

se=strel('square',3)
```

Matlab output

```
se =

Flat STREL object containing 9 neighbors.

Neighborhood:
     1     1     1
     1     1     1
     1     1     1
```

This function which is very often used and therefore of great importance, converts and compiles without errors, but gives an error during runtime.

A part of the problem can already be seen during conversion-time:

```
Warning: File: streltest Line: 3 Column: 4
The MATLAB Compiler does not currently support MATLAB object-
oriented programming. References to the method "strel" will produce
a run-time error.
```

Compilation runs without problems, but when trying to run the built executable the following problem occurs:

```
Exception!  File: handler.cpp, Line: 73
Undefined function or variable 'strel'.
```

Solution The solution to this problem is given on the Mathworks website.

It seems the Matlab Compiler can not find the necessary files for the correct conversion, therefore a set of files needs to be copied to a general accessible folder.
`http://www.mathworks.com/support/solutions/data/29113.shtml`

```
Solution:

If you are using any morphological functions in the M-file that you
are trying to compile, please copy all the files from:

toolbox/images/images/@strel

into your working directory, or somewhere on the MATLAB path so that
the compiler can see them. Please Note: This is not a general way to
compile user-defined classes (this is a specific work-around
provided for the Image Processing Toolbox).
```

The files involved in this operation are:

```
Contents.m
disp.m
display.m
getheight.m
getneighbors.m
getnhood.m
getsequence.m
isflat.m
reflect.m
strel.m
translate.m
```

These files need to be copied to the "working directory" meaning to the directory where the main algorithm that uses the strel-functions, is situated.

After doing this and reconverting/recompiling/rebuilding the project, another problem occurs, as is seen during conversion-time:

```
Warning: File: strel Line: 652 Column: 5
References to "checkinput" will produce a run-time error because it
is an undefined function or variable.
(and others alike)
```

and during runtime:

```
Exception!  File: handler.cpp, Line: 73
Undefined function or variable 'checknargin'.
```

The solution to this new problem can again be found on the Mathworks-website:
`http://www.mathworks.com/support/solutions/data/32474.shtml`

```
This error message occurs because the MATLAB Compiler does not
locate these files properly.

To work around this problem, move all the files starting with
"check" from the Image Processing Toolbox 3.2 (R13) private
directory into the parent directory of that private directory.
That is, copy these files:
```

```
$MATLAB\toolbox\images\images\private\checkconn.m
$MATLAB\toolbox\images\images\private\checkinput.m
$MATLAB\toolbox\images\images\private\checknargin.m
$MATLAB\toolbox\images\images\private\checkstrs.m


to this location:


$MATLAB\toolbox\images\images
```

The Mathworks website also offers a immediate method to do this.

```
movefile([matlabroot '\toolbox\images\images\private\check*'],
[matlabroot '\toolbox\images\images\'])
rehash toolbox
```

After copying these files, and restarting the project, finally the correct result is obtained.

2.4.4 Distributing the applications

After successfully converting the Matlab-code into C/C++-code, and then compiling and building them into an executable exe-file, we can now consider the final stage, what it was meant to be, in casu a stand-alone application. First, it is important to define what a stand-alone application is.

In our project, a stand-alone algorithm is an algorithm that can be run independently from any installed program, shell or gui. This means that the program must also run when Matlab is not installed on the system, or when there is no gui available for the algorithm.

The second objective is already completed. Since every algorithm is treated as a separate one, every algorithm can be run from any command prompt, that is if all other necessities are available, such as libraries (see later) or files (input, images, ...).

The following of this section can be mainly be revised in brief here:
`http://www.mathworks.com/support/tech-notes/1600/1606.shtml`

For the first adjective, some larger arrangements need to be made.

2.4.4.1 Installing standard Matlab Libraries

All the basic Matlab-libraries need to be installed on the system, in order to make any program work on any system. MathWorks provides an installer to perform this action. It is necessary that this installer is executed, although it appears it is only copying files, merely copying files does not work. The installer can be found here: `$MATLAB\extern\lib\win32\mglinstaller.exe` This file should be distributed along with any executable created by the Matlab Compiler. This file should then be installed on the target computer. The installation is pretty straightforward, yet I will provide here a description.

Where After double clicking, two files will be extracted, and you will be asked where to install the libraries. If you only need one algorithm, which is in the folder where you started the installer, you can select the default folder by pressing ENTER, otherwise, which is recommendable, you can choose to install the libraries in any path you like.

Checking This will extract two folders into the folder you have selected. These folders are named *bin* and *toolbox*.

Adding path In this part you will need to add the path of the libraries to the system's path variable, so the system can find the new libraries. You can add the path as following:

- Start-Control Panel-System
- tabpage Advanced, Environment variables
- In the Systems Variable-section, locate the path-line and press Edit
- At the end, add a semi-colon, followed by the path of the libraries, followed by `\bin\win32`. For example, if you installed the libraries to the folder *c:\matlibs*, the path line should be changed by adding *c:\matlibs\bin\win32*, as shown in the picture.

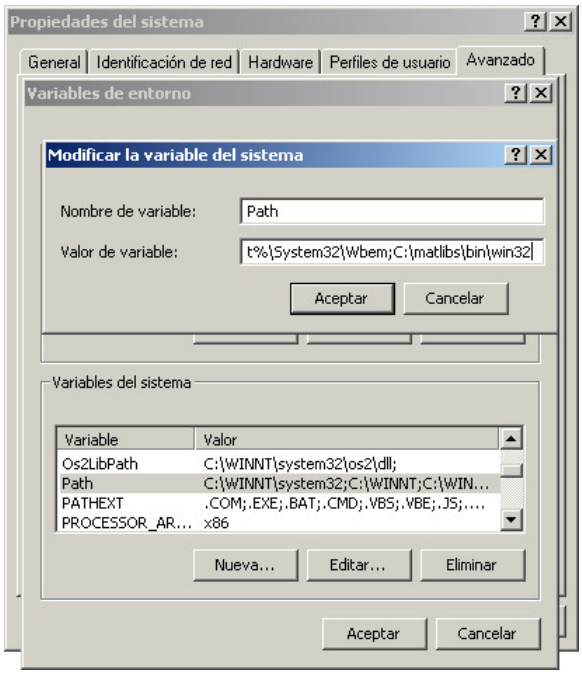


Figure 2.12: Path variable adjustment

2.4.4.2 Copying (optional) binairies

If there are any, all files within the folder *bin*, created in the working directory, should be distributed along with the algorithm.

2.4.4.3 Any custom MEX-files your application uses

This is the least clear and also the most tricky section. On the Mathworks, only is stated as in the title: *Any custom MEX files your application uses.*

Yet it is difficult to know which those are. One possibility is to look in the working directory of the program at the generated C/C++-files. Most of them will represent a dll. For example *applylut_mex_interface.c* represents *applylut.dll.* The other possibility is a trial-and-error procedure. In this, you try executing the algorithm from the command prompt. If an error occurs, it will be in the line of the following:

```
ERROR: Failed to find MEX-File on path : dataread.dll
```

This directly points out which dll is missing. Locate this dll in your Matlab-folder, and copy it to the working directory of the program, and retry.

Yet also a slightly different error can occur.

```
Failed to find MEX-File on path : images/private/imfilter_mex.dll
```

Copying this dll does not affect this error. The explanation and solution can be found on the Mathworks website:
`http://www.mathworks.com/support/solutions/data/32983.shtml`

They also provide a solution:

```
Solution:

The current fix to this problem is to recreate the MATLAB
path on the deployment computer.

For example, consider the DLL:

C:\matlab6p5\toolbox\images\images\private\imhistc.dll.

On the deployment computer, you will need to create the entire tree:

c:\matlab6p5\toolbox\images\images\private\imhistc.dll.
```

Thus on the target computer, we need to create that specific tree exactly as on the source computer, and copy the libraries to the target computer.

These are the libraries involved:

```
bwfillc.dll
bwlabel1.dll
bwlabel2.dll
bwlabelnmex.dll
castc.dll
cq.dll
ddist.dll
dicom_decode_rle.dll
dicom_decode_rle_segment.dll
```

```
dilbw.dll
ditherc.dll
erobw.dll
eucdist2.dll
grayto16.dll
grayto8.dll
grayxform.dll
imdivmex.dll
imfilter_mex.dll
imhistc.dll
imlincombc.dll
immultmex.dll
inv_lwm.dll
inv_piecewiselinear.dll
iptregistry.dll
morphmex.dll
mwguidgen.dll
nnsearch.dll
ordf.dll
radonc.dll
resampsep.dll
watershed_vs.dll
```

After copying all these libraries, the algorithm should run fine.

2.4.4.4 Summary

Thus, in short the following steps are necessary when distributing an algorithm:

- Install Matlab Runtime Libraries (mglinstaller.exe)
- Copy files from
  bin-folder to target computer
- Copy any other needed libraries to target computer (including privates)

2.5 Ad/Disadvantages Matlab Compiler

2.5.1 Advantages

In this section I will try to summarize and describe the advantages the Matlab Compiler offers. I will go through some of the advantages the Mathworks-company states, as well as some of my own experiences. I will not only give here the advantages, but critically look at them whether they are achieved or not.

First, let us go through the advantages according to Mathworks.

2.5.1.1 Advantages according to Mathworks

The Matlab Compiler gives according to the Mathworks website and the documentation of the Compiler the following advantages:

- Faster execution

- Easy distribution
- To hide algorithms

Faster execution is an advantage that might seem important nowadays. Mathworks states that compiled functions should run faster than the original. There are four reasons to this. The first reason is that every time an algorithm is run within the matlab environment, the code needs to be interpreted by the Matlab engine, and converted to machine language so the processor can execute it. Although the current engine and wide instruction set of current processors have limited this time, it could be a difference. Yet, if it would make a difference, it would be unnoticeable compared to the overall execution time of the algorithms used. The second reason is that Matlab treats all variables as matrices, whether it is a simple integer or a complex array. The code in C/C++ does not act this way. It often uses simpler structures, such as integers, when a more complex structure is necessary. Since our algorithms use all kinds of structures, this advantage can hardly be noticed. The third reason is that the generated C-code (not C++-code) can be set up not to check array boundaries. Matlab always performs this check. Yet to insure stability it is advisable to leave this check in the generated code, so no speed advantage can be found here. The fourth deals with a better memory allocation used in C/C++ than in Matlab. No real details can be said about this.

Overall, it can not really be said that the compilation of the programs offers a significant increase in execution time. In some cases the compiled file turned out to be even slower. Yet in other cases a speed increase might be noticeable. Mathworks also points this out. They state that compilation is most likely to speed up algorithms using loops, integers or other simple data structures. Yet, it is not likely to speed up algorithms using complex vectors, Matlab's indexing or graphics functions. Since in our case most of the time we are dealing with the latter case, it is not so remarkable no speed increase was noticed.

Easy distribution was one of the main reasons for this project. The free distribution as a stand-alone application without the necessity of Matlab, makes it easy to use the algorithms in an external environment, such as in our case a hospital or a university. So distributing your stand-alone applications is indeed possible. It could be considered as easy, since once a package is set for an algorithm, it can be installed on every pc in the same way. Yet it could also be considered as un-easy and complicated, since creating the packages is far from transparent or reusable. Although the included installer provides a large deal of the needed libraries, the necessity to adjust the system path manually, and the necessity to add additional libraries, which are to be found through a primitive manner, makes the distribution a part on which the Compiler can improve in the future. In the end though, a package could be created using an installer, which then could be easily distributed, so none of this hassle would be necessary.

Hiding of algorithms is obtained easily. Since the built executable can be distributed without the necessity of any source files, none of the internal mechanism of the algorithm is available to the outer world. Although the distributing of source code should be cheered at, when hours and hours of developing

time and big investments were necessary, this is the only way to be able to gain some of its costs.0

2.5.1.2 Other Advantages

The following advantages are some of the advantages I saw out of experience of working with the Compiler. Some of these are already in respect to other possibilities of implementing these algorithms.

Fast developing When all problems are solved with the compiler, and a bit of knowledge of the generated errors and problems is at hand, converting an algorithm using the Matlab Compiler can become a matter of minutes. This is a great deal less than when using external libraries. In that case you basically need to rebuild the algorithm from scratch, translating every line of Matlab-code into a similar function provided by the library. Often these functions use different type of arguments than the function in Matlab, which makes it even more difficult to reproduce the results.

Easy testing It is easy to test the converted product. Since there is always the source algorithm available within the Matlab Environment, it is easy to control the results obtained by the algorithm.

Easy installation The installation of the Compiler is very straightforward and well documented, and should not impose any problems, when following the guidelines. Installing the Compiler after an installation of Matlab does not impose any problems either.

Easy integration with MSVC The possibility to add an add-in to the Microsoft Visual C++ developing environment, and the ability to easily start a project based on a Matlab-file from within the MSVC-environment, makes the compiler very easy to use, without having to know off the parameters of the conversion.

2.5.2 Disadvantages

In this part the main disadvantages of the Matlab Compiler that I ran into during the project will be discussed. These are all disadvantages out of experience, since of course Mathworks does not provide any disadvantages as is. The limitations and restriction on the functionality of the Compiler, as given by Mathworks, could be considered as disadvantages, but since the usage of these functions in a stand-alone version is often unnecessary, and in our algorithms never used, it is my opinion that these do not really count as a disadvantage.

Undocumented limitations There are some limitations to the Compiler that are not mentioned. As can be seen in the section 2.4.3 on page 28 a lot of functions carry out problems. The solution to these problems can almost always be found on the Mathworks-website, so they are aware of them, but yet nothing is mentioned in the manual.

Incompatible functions There seem to be quite some functions that have difficulties to be converted correctly. This is a disadvantage to the Compiler, since for every algorithm, you need to find out which is the erroneous function, and then find a solution. This can be time-consuming, but as already

stated before, it seems that when a number of solutions is applied, a lot of problems are avoided and solved. Therefore it is advisable to first apply these solutions in advance, to avoid a few problems.

Code preparation As stated in the section "Preparing your m-file" on page 21, a few adjustments need to be made to the original algorithm, in order to ensure proper converting. These adjustments could take some time, but when an example is at hand, it's often a matter of copy-paste.

2.6 Comparison Matlab/Compiled algorithms

2.6.1 General notices

An important part of the entire process of the conversion, and which was one of the reasons to start studying on the conversion possibilities, was to make a comparison on the result of the different methods.

It is necessary to review the both otherwise identical algorithms, being one written and run within the Matlab environment, the other one being the converted C/C++-version. It is needed to review these algorithms on several fields that serve the main purpose, being offering a fast, cheap and easy-to-use solution for the people at the hospital to process their images. An other important factor is the ease for the programmer, or how to obtain the final result that will be distributed to the distinct hospitals.

Another aspect that could be considered as important is the transportability to several platforms. In the end, it might be interesting to develop executables or code that can be run on several platforms, mainly being Windows and Linux. This because both offer their own advantages, making further research on both platforms possible, thus not limiting the options by restraining one of them.

2.6.2 Speed comparison

One of the main reasons for starting this project was the possible increase in speed that could be obtained by converting the matlab algorithms to C/C++-code. This was important, since some of the algorithms can be quite time-consuming, especially when using large sets of images, or in the future, when developing will be extended to the usage of three-dimensional objects. The importance of this factor does not only reflect in this project, but also in other projects running in the GVA, such as the clustering of computers.

The reasons why the conversion from the Matlab-algorithms to C/C++-code could deliver a decrease of execution time were already discussed at section 2.5.1.1 on page 47, but will be repeated here in short:

Interpretation Every time an algorithm is run within the Matlab-environment, the code needs to be interpreted by Matlab, and converted to assembly code which can then be run by the processor. This is not necessary when you have C++/C-code, which, once built, generates an executable which does not have to be interpreted again.

Variables Matlab treats every variable, even an integer, as matrices. This creates overhead that is speed-restraining, which is not necessary in C/C++, where all data types can be used.

Array boundaries Matlab always checks for array boundaries, in order not to go out of these arrays and thus cause errors. This can be turned of if wanted in C++/C, in order to gain speed. Yet this is not advisable since it could lead to unwanted crashes.

Memory allocation Memory allocation provided by the C/C++-compiler is supposed to be better than that of the Matlab interpreter. This could gain a speed increase as well.

In order to test the difference in execution time, three algorithms were taken and were processed on a set of images. These specific algorithms were taken, since these were the one requested to be converted by the end of the project. These three algorithms would be the ones sent to the hospitals, and which would be integrated into a graphical user interface for automatic processing.

The algorithms in particular are:

Neuro Blastomas A very time-consuming algorithm that takes two images at a time in order to calculate numbers of viruses.

FHV Virus Algorithm to detect the number of viruses in one image

FHV Cellulas Algorithm to count the cells of an enhanced image

2.6.2.1 Neuro Blastomas

This is one of the more time-consuming algorithms. It is used to detect the number of bad cells in two images. The first image contains the original cells, and the second contains a selected part of marked cells. An example of these images can be seen below:

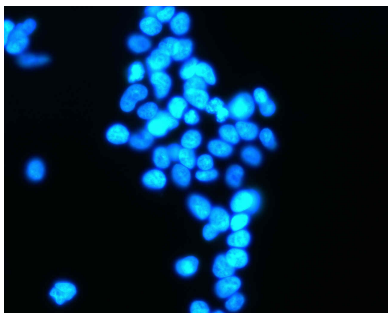


Figure 2.13: Original cells

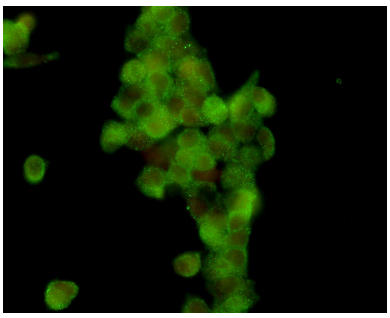


Figure 2.14: Marked cells

The algorithms uses mainly morphological functions, of which the following are the most typical for the algorithm:

- Edge-detection through 'canny'-algorithm
- Image Dilation
- Image Histogram
- Image Subtraction
- Image Region Properties calculation

The algorithm was processed on 24 images, meaning 12 sets of 2 images, since the algorithm takes 2 images at a time.

The following table presents the execution times in the different environments: [15]

	OS	Number of images	Total execution time	Avg time /set	Avg time /image
MATLAB	Win32	24 (12x2)	2:33	12,82s	6,41s
C	Win32	24 (12x2)	2:42	13,84	6,95s
MATLAB	Linux	24 (12x2)	2:10	10,9s	5,45s
C	Linux	24 (12x2)	2:13	11,1s	5,55s

Table 2.5: Execution times NB

Thus, in the end, it can be seen that in this case, there is not a real gain in speed. The difference of 0,1s can as well be less the next time the algorithm would be processed.

The explanation of this lack of increase in speed lays in one of the main functions of the algorithm. The most time consuming function here is the *edge-detection*-function, which uses the *canny*-method to calculate the edges in the image. Although the other functions do offer some speed increase, this speed increase is

[15] OS: Operating System

zeroed out by the edge-detection, which takes longer in C/C++ than in the Matlab-environment.

A remark on this is that this is not the only case with this function. When trying to implement this function using *Vigra* (see later), this function turned out to be very slow in C/C++ as well. Basically it could be concluded that the implementation and handling of this function in Matlab is far superior than those in C/C++.

2.6.2.2 FHV Virus

This is the first of the two algorithms used in a particular hospital. As the algorithm it is quite straightforward, and not as time consuming.

The main functions used:

- Image Histogram
- Thresholding
- Image labeling

The simplicity of the algorithm makes it fast to execute. It only takes one image to compute. An example of such an image is presented below:

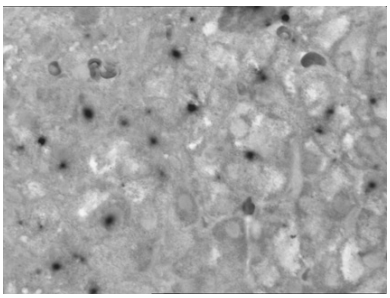


Figure 2.15: Example of FHV Virus image

The following table presents the execution times in the different environments:

	OS	Number of images	Total execution time	Avg time /set	Avg time /image
MATLAB	Win32	8	1,2s	150ms	150ms
C	Win32	8	1,02s	127ms	127ms
MATLAB	Linux	8	1,12s	140ms	140ms
C	Linux	8	0,88s	110ms	110ms

Table 2.6: Execution times FHV Virus

In this case a difference can be seen. Although it seems very small, in average a gain of 13% can be seen. Here we can see some of the speed advantages the conversion to C/C++ can give.

2.6.2.3 FHV Cellulas

The second of the two algorithms counts the cells of an image. Again it is quite a straightforward algorithm, and differs from the previous only in a few parameters.

Thus again the same main functions can be found: The main functions used:

- Image Histogram
- Thresholding
- Image labeling

An image example:

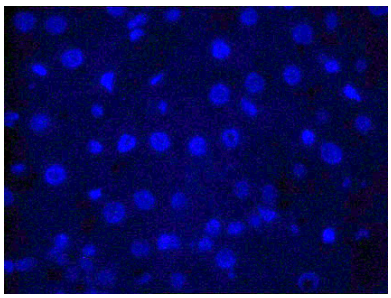


Figure 2.16: FHV Cellulas Example

The following table presents the execution times in the different environments:

	OS	Number of images	Total execution time	Avg time /set	Avg time /image
MATLAB	Win32	8	1,6s	203ms	203ms
C	Win32	8	1,2s	153ms	153ms
MATLAB	Linux	8	1,44s	180ms	180ms
C	Linux	8	1,12s	140ms	140ms

Table 2.7: Execution times FHV Cellulas

This time an even greater amount of speed increase can be noticed. The difference is now up to 45%.

2.6.2.4 Conclusion

As a conclusion it can be said that the conversion of the algorithms to C/C++ can indeed cause a speed increase in the execution. Yet it is also to be noted that this might be very depending on the type of algorithm and the functions used. It appears that some functions even cause a longer execution time. The *Canny Edge-detection* is a good example of this.

2.6.3 Programmer's choice

Another aspect that might be considered as important is the ease for the programmer to develop his applications. In the comparison between the algorithm within the Matlab environment, and the built algorithm in the dos-environment, this discussion is somewhat pointless.

Pointless, since every built algorithm is based on an algorithm in Matlab, thus always extra will be needed to make the conversion. Yet it is to be noted, that, using the plug-in for the Microsoft Visual Studio, and following the notes in the book earlier on, describing the possible problems and precautions, the conversion can be considered as an easy job, only taking a few minutes.

2.6.4 Transportability between platforms

Another important feature of an application might be its easy transportability between platforms. When speaking of platforms, the main can be considered to be Windows and Linux, since those are most widely spread and available.

Here a clear distinction can be found between the two environments. Since Matlab also exists for Linux, and all functions available in Windows are also available on that platform, the algorithms developed in Matlab do not need any adjustment when porting them for usage in Linux. The only thing that might need to be changed is static path variables, if present, since path and file handling is somewhat different between the two platforms.

This does not count for the built algorithm. This even counts in two phases. First, during the conversion of a m-file, thus a matlab-algorithm, several Matlab-libraries are used in order to correctly convert the functions. These libraries are different in Windows and in Linux. Secondly, when linking the C++/C-code in the compiler, the compiler is set to compile executables for the one or the other platform. These executables can not be executed in the other platform.

Thus in this part, the Matlab algorithm is in the advantage. Yet the Compiler also exists in Linux. Although the ease-of-use, mainly provided by the plug-in for Microsoft Visual C++, is absolutely absent, and Linux might be more difficult to work with, in the end it is possible to convert the algorithms to executables in the Linux environment as well.

When considering a GUI, to facilitate the usage of the algorithm, the picture is somewhat different. Here the graphical user interfaced that would be developed in Matlab, is not compatible with Linux. Since the gui accesses functions, for example to obtain the contents of a folder, only available in Windows, porting the gui to Linux takes an extensive amount of work. A GUI, developed in any other, por fundo platform independent developing language, such as Swing in Java or QT in C++, would need none or only minor adjustments in order to work correctly on the other platform.

2.6.5 Other remarks

The points mentioned above are the main comparison subjects on the two running environments. Yet other aspects might be considered as important as well. Especially those from the point of view of the future users of the algorithms, ic the people at the different hospitals the GVA is working for.

For those people other aspects than ease-of-development or portability might be of importance. In this section a short notice on some of these possible aspects will be given.

2.6.5.1 Price

For the people at the hospital this probably is, except of course the accuracy and correctness of the results, one of the main factors. Here it is that the strength of a converted algorithm comes at hand.

If a hospital would like to run the algorithm from within the Matlab-environment, they would need to acquire the following:

Matlab 6.5 R13	2650eur
Matlab Signal Processing Toolbox	1100eur
Matlab Image Processing Toolbox	1250eur

Table 2.8: Matlab Requirements

This is for only one user, and without going into pirate software, this would be an immensive amount of money to be spent.

On the other hand, using compiled algorithms, which can be distributed as stand-alone applications, without the necessity of Matlab, this could be considered as free, knowing that the GVA does not charge the hospitals. In this case only the previous products are needed, plus the Matlab Compiler.

Therefore it is clear that using stand-alone applications is a lot cheaper for the hospitals than acquiring the Matlab Software itself.

2.6.5.2 Ease of use

For the people at the hospital, it is necessary that the algorithms are easy to use, without needing to know any of the inner of the algorithms.

Considering the algorithms within the Matlab-environment, this is hardly the case, the user needs to know what kind of inputs are expected, and how to get the output. Nonetheless this is basic Matlab knowledge, and with some simple instructions it can be clarified. Also Matlab offers the ability to construct a graphical user interface, invoking the algorithm. The development of this gui takes some time, but makes it for the users easier to use the algorithm, plus it makes it possible to process a set of images automatically. Thus the user friendliness can be augmented substantially.

The Stand-Alone application as is, is not as user-friendly either. The user needs to create an input file that contains any data needed by the algorithm, such as the name of an image, plus the user needs to go into the dos-command box to run the algorithm. But also here a solution is possible. Again, a lot of ways exist to build a GUI which provides the same functionality or more as would be possible in Matlab. It is possible to use programming languages as Java (Swing) or C+ (QT) to create platform independent GUI's.

Chapter 3

Vigra

3.1 What is Vigra?

3.1.1 General Description

VIGRA stands for "Vision with Generic Algorithms". It's a novel computer vision library that puts its main emphasize on customizable algorithms and data structures. By using template techniques similar to those in the C++ Standard Template Library, you can easily adapt any VIGRA component to the needs of your application, without thereby giving up execution speed. VIGRA was mainly implemented by Ullrich Köthe.

In order to fully understand the VIGRA-libraries, and if you're willing to adjust, improve or write new functions, it is advisable to check the following documentation:

Reusable Software in Computer Vision in: B. Jähne, H. Haußecker, P. Geißler: "Handbook on Computer Vision and Applications", volume 3, Acadamic Press, 1999

STL-Style Generic Programming with Images in: C++ Report Magazine 12(1), January 2000

Note: Yet for this project none of this is necessary, we will deal only with the available functions.

The Vigra Homepage can be found here:
`http://kogs-www.informatik.uni-hamburg.de/~koethe/vigra/`

3.1.2 Features

The list of features Vigra offers is quite long:

Images

- templated image data structures for arbitrary pixel types
- pre-instantiated images with many different scalar and vector valued pixel types
- 2-dimensional image iterators, adapters for various image subsets

- input/output of many image file formats: Windows BMP, GIF, JPEG, PNM, Sun Raster, TIFF (including 32bit integer, float, and double), Khoros VIFF

Image Processing

- STL-style image processing algorithms with functors (e.g. arithmetic and algebraic operations, gamma correction, contrast adaptation, thresholding), arbitrary regions of interest using mask images
- image resizing using resampling, linear interpolation, or spline interpolation
- automated functor creation using expression templates
- color space conversions: RGB, R'G'B', XYZ, L*a*b*, L*u*v*, Y'PbPr, Y'CbCr, Y'IQ, and Y'UV
- Fourier transform (via fftw)

Filters

- 2-dimensional and separable convolution, Gaussian filters and their derivatives, Laplacian of Gaussian etc.
- recursive filters, exponential filters
- non-linear diffusion (adaptive filters)
- distance transform (Manhattan, Euclidean, Checker Board norms)
- morphological filters and median (disk structuring elements)
- Loy/Zelinsky symmetry transform
- Gabor filters

Segmentation

- edge detectors: Canny, zero crossings, Shen-Castan
- corner detectors: corner response function, Beaudet, Rohr and Förstner corner detectors
- region growing: seeded region growing, watershed algorithm

Image Analysis

- connected components labeling
- detection of local minima/maxima (including plateaus)
- region statistics

3.2 Installation of Vigra under Windows

3.2.1 Downloads

In order to work with the Vigra libraries in conjunction with JPEG and TIFF-files, some files need to be downloaded. I will try as much as possible to inform the version of the needed files. This because of problems that occur when using different/newer versions of the software.

It is advisable to create a base directory in which the downloaded files will be stored.

- Vigra Package The current version of the Vigra library package is version 1.1.6. This can be downloaded from the Vigra Homepage (`http://kogs-www.informatik.uni-hamburg.de/~koethe/vigra/`) or directly from this link: `http://kogs-www.informatik.uni-hamburg.de/~koethe/vigra/vigra1.1.6.tar.gz`.
- TIFF To have support for the TIFF-image format, it is necessary to download the needed library files and include files for this format. The homepage can be found here: `http://www.libtiff.org/`. All necessary files can be found here: `http://gnuwin32.sourceforge.net/packages/tiff.htm`. The version used during the project was version **3.5.7**. The following files are needed:
  1. Binaries - tiff-3.5.7-1-bin.zip
  2. Libraries - tiff-3.5.7-1-lib.zip
- FFTW These files are needed for some of the functions of Vigra. If these functions are not used, these files are not necessary, yet it is advisable to download and install them. The functions in casu are Fast Fourier Transform-functions. The homepage of the FFTW-libraries is located here: `http://www.fftw.org/`. The libraries used in the project had version **2.1.3**. Yet in order to avoid having to build the libraries yourself, pre-built libraries can be found here: `http://claymore.engineer.gvsu.edu/\~steriana/software\_fftw.html`. The FFTW for Win32 are needed, which can be downloaded directly here: `http://claymore.engineer.gvsu.edu/\~steriana/fftw213.libs.zip`.
- ZLIB ZLib is a library needed when compressing/decompressing image files, such as is the case with JPEG-files. Therefore, when working with JPEG-files, these libraries are needed too. The hompage is located at: `http://www.gzip.org/zlib/`.They can be found at `http://gnuwin32.sourceforge.net/packages/zlib.htm`. The version used in the project was version **1.1.4**. Yet again, a pre-built library can be found, and is advisable: `http://www.winimage.com/zLibDll/zlib114dll.zip`.
- JPEG Since most of the images worked with during this project are JPEG-files, the libraries were needed too. The homepage of the open-source JPEG-libraries is located here: `http://www.ijg.org`. Of this, several files are needed, and it is important to get the correct files, since there might be errors using other versions.

1. Binaries - jpeg-6b-1-bin.zip - 153 712 bytes - `http://gnuwin32.sourceforge.net/downlinks/jpeg-bin-zip.htm`
2. Libraries - libjpeg.lib - 900 758 bytes - `http://www.stillhq.com/cgi-bin/cvsweb/panda/contrib/libjpeg/libjpeg.lib`
3. Include files - jpegsrc.v6b.tar.gz - 613 261 bytes - `http://ijg.org/files/jpegsrc.v6b.tar.gz`
4. Windows DLL - libjpeg.dll - 151 685 bytes (rename jpeg-62.dll)

3.2.2 Installing downloads

After downloading the necessary files, the proper files need to be extracted and system settings need to be updated.

It is advisable to create a folder in which all the libraries, except those of vigra, will be installed. Since all libraries reside under the GNU-license (see license) (or similar), a folder *gnuwin32* was created for this purpose.

3.2.2.1 FFTW Libraries

When you followed the instructions in the download-section, you should have the *fftw213.libs.zip*-file in your download folder.

Open the file with Winzip, and extract the files to a destination folder, eg `C:\Archivos de programa\GnuWin32\FFTW\lib`.

The following files should now be located in the folder:

```
FFTW2dll.dll
FFTW2dll.exp
FFTW2dll.lib
FFTW2st.lib
RFFTW2dll.dll
RFFTW2dll.exp
RFFTW2dll.lib
RFFTW2st.lib
```

3.2.2.2 TIFF Libraries

The following files should be downloaded into your folder:

```
tiff-3.5.7-1-lib.zip
tiff-3.5.7-1-bin.zip
```

Both files should be extracted into a proper folder, such as TIFF, while assuring the option "use folder names" within winzip is checked.

After extracting, the following folders should have been created:

```
bin
contrib
include
lib
man
manifest
```

3.2.2.3 ZLIB Libraries

Again, extract the file you downloaded, *zlib114dll.zip* to a proper folder, eg *LIB*.

The following files/folders should have been extracted:

```
dll16
dll32
static32

ioapi.h
readme.txt
unzip.h
zconf.h
zip.h
zlib.h
```

3.2.2.4 JPEG-Libraries

The installing of these files is somewhat different in some cases:

jpeg-6b-1-bin.zip should be extracted into a proper folder, preferably called JPEG. A folder called *bin* will be created. Assure the file *jpeg-62.dll* is available.

libjpeg.lib should be copied into a folder called *lib* into the JPEG-folder.

jpegsrc.v6b.tar.gz should be extracted into the JPEG-folder. A folder called *jpeg-6b* will be created

libjpeg.dll is the same as *jpeg-62.dll*. Therefore, the *jpeg-62.dll* can be copied to libjpeg.dll in the same folder.

3.2.2.5 Vigra-Libraries

Extract the file *vigra1.1.6.tar.gz* into a chosen folder.

3.2.2.6 Updating MSVC Directories

In order to make it possible for Microsoft Visual C/C++ to find the new libraries and source files, the directory settings need to be adjusted.

The directory settings can be found at: Tools-Options-Directories.

$installpath is the path where you extracted all the libraries. If you chose as suggested, 4 folders should be there:

```
FFTW
ZLIB
JPEG
TIFF
```

The following directories should be added:

Include files

- $installpath\JPEG\JPEG-6B (several header/source-files)

- $installpath\TIFF\INCLUDE (tiff.h and others)
- $vigrapath\VIGRA1.1.6\INCLUDE (Vigra and Irix-folders)

Library files

- $installpath\JPEG\LIB (libjpeg.lib)
- $installpath\TIFF\LIB (libtiff.lib)
- $installpath\FFTW\LIB (FFTW2dll.lib)

3.2.2.7 Updating Windows path Variable

Windows needs several Dynamic Link Libraries (DLL's) in order to run the executables generated using the Vigra Libraries. These DLL's are mainly concerning the usage of JPEG and TIFF-files.

If you do not know how to update the Windows Path Variable, check the section on *distributing Matlab stand-alone applications* on section 2.4.4.1 on page 44.

The following paths should be added to the Windows Path-variable:

- $installpath\FFTW\LIB (several dll's)
- $installpath\ZLIB\DLL32 (zlib.dll)
- $installpath\JPEG\BIN (jpeg-62.dll and libjpeg.dll)
- $installpath\TIFF\BIN (libtiff.dll)

3.2.2.8 Handling Microsoft Visual C/C++ 6.0 Compiler Bug

Please note that Microsoft has fixed all of the problems described below in Visual Studio.NET, so the following description is relevant only if you are using the earlier 6.0 version.

The Microsoft Visual C++ compiler, version 6.0 (which we shall call VC++ 6.0) fails to match the C++ standard in several ways that prevent some of the code in in the Vigra C++ library from compiling properly. A deal of these unmatching code problems can already be solved by installing the latest service pack available. This service pack, currently version 5, can be downloaded from the Microsoft website: `http://msdn.microsoft.com/vstudio/downloads/updates/sp/vs6/sp5/default.aspx`.

The problem influencing the Vigra-libraries is the following. The MS library does not define the min and max algorithms, which should be found in the *¡algorithm¿*-header.

The problem occurs when trying to compile certain functions, error as the following might occur:

```
c:\vigra\vigra1.1.6\include\vigra\inspectimage.hxx(949) :
error C2039: 'min' : is not a member of 'std'
c:\vigra\vigra1.1.6\include\vigra\inspectimage.hxx(949) :
error C2065: 'min' : undeclared identifier
```

The workaround we use is to define a new header file, say minmax.h, which we include in any file that uses these functions.

This header file should look like this:

```
#ifndef␣GUARD_minmax_H
#define␣GUARD_minmax_H
#ifdef␣_MSC_VER
//␣needed␣to␣cope␣with␣bug␣in␣MS␣library:
//␣it␣fails␣to␣define␣min/max

template␣<class␣T>␣inline␣T␣max(const␣T&␣a,␣const␣T&␣b)
{

return␣(a␣>␣b)␣?␣a␣:␣b;
}

template␣<class␣T>␣inline␣T␣min(const␣T&␣a,␣const␣T&␣b)
{

return␣(a␣<␣b)␣?␣a␣:␣b;
}

#endif

#endif
```

This file should be copied to the include folder of vigra, for example `C:\Vigra\vigra1.1.6\include\vigra`.

The errors mentioned all occur in another header-file of Vigra, named **inspectimage.hxx** which is located in the Vigra-include directory. This file should be adjusted as following. In the appendix the full source-code of the sections that need to be changed can be found so you can copy it from there if needed.

1. Include *minmax.h* by adding `#include "vigra/minmax.h"` to *inspectimage.hxx*
2. Change every instance of `std::min` and `std::max` to `min` respectively `max`

3.2.3 Testing Installation

In order to test the correct installation and set-up, open the Vigra Workspace, which is located in the *vigra\src*-folder.

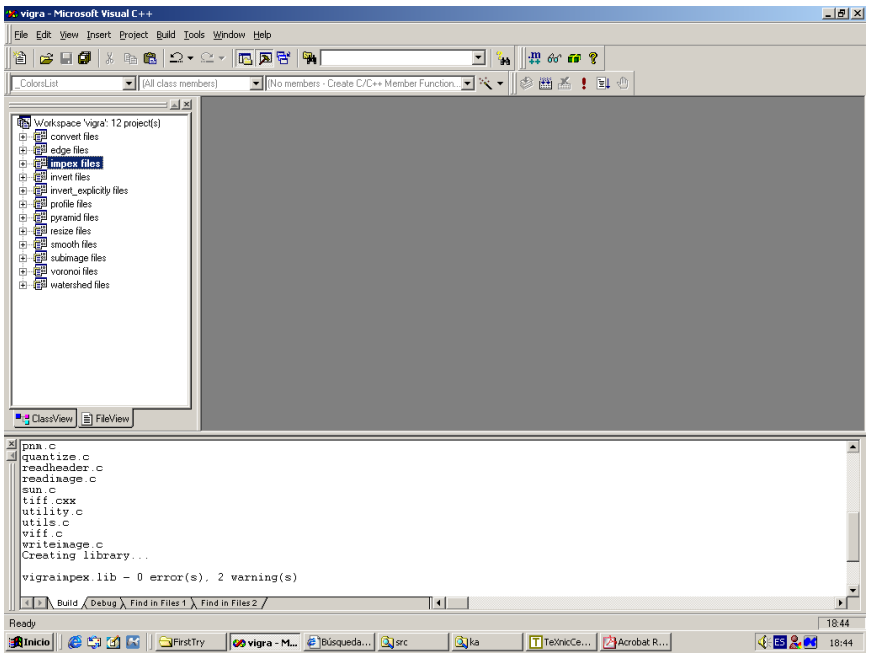


Figure 3.1: Vigra Workspace

Set the **impex**-project as active project, and compile. If all is ok, the building will succeed, but you might get warnings, as shown in the picture. This doesn't affect the correct operation of vigra. It is important that this compilation is successful.

Then, compile, for example, the **convert**-project. This should also succeed. You should then have a file called *convert.exe* in the *examples*-folder.

Running this file without any arguments should give the following output:

```
Usage: C:\Vigra\vigra1.1.6\src\examples\convert.exe infile outfile
(supported formats: BMP GIF JPEG PBM PGM PPM P7 PNM SUN TIFF VIFF )
```

Running this file with arguments included should work fine too:

```
C:\Vigra\vigra1.1.6\src\examples>convert c:\wouter\images\out.jpg
c:\wouter\images\test.bmp

C:\Vigra\vigra1.1.6\src\examples>
```

As a final test, set the function edge as active project, and try to compile it. If the previous worked, this should compile without errors. If some errors do occur, check the solution in section 3.2.2.8 on page 61

3.3 New Vigra-project

3.3.1 Testfile

It is now possible to create a new project and start working using the Vigra-libraries for C.

To make sure the source file is not the cause of the problems, I will here post an example-file which should compile fine. This is an exact copy of the **convert.cxx**-file, which is a good base for every algorithm, since it only reads and writes the images to the correct image type.

convert.cxx

```
/************************************************************************/
/*                                                                      */
/*               Copyright 1998-2002 by Ullrich Koethe                  */
/*       Cognitive Systems Group, University of Hamburg, Germany        */
/*                                                                      */
/*    This file is part of the VIGRA computer vision library.           */
/*    ( Version 1.1.6, Oct 10 2002 )                                    */
/*    You may use, modify, and distribute this software according       */
/*    to the terms stated in the LICENSE file included in               */
/*    the VIGRA distribution.                                           */
/*                                                                      */
/*    The VIGRA Website is                                              */
/*        http://kogs-www.informatik.uni-hamburg.de/~koethe/vigra/      */
/*    Please direct questions, bug reports, and contributions to        */
/*        koethe@informatik.uni-hamburg.de                              */
/*                                                                      */
/*  THIS SOFTWARE IS PROVIDED AS IS AND WITHOUT ANY EXPRESS OR          */
/*  IMPLIED WARRANTIES, INCLUDING, WITHOUT LIMITATION, THE IMPLIED      */
/*  WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. */
/*                                                                      */
/************************************************************************/


#include <iostream>
#include "vigra/stdimage.hxx"
#include "vigra/impex.hxx"

using namespace vigra;
// MSVC doesn't support Koenig lookup


int main(int argc, char ** argv)
{
    if(argc != 3)
    {
        std::cout << "Usage: " << argv[0] << " infile outfile" << std::endl;
        std::cout << "(supported formats: " << vigra::impexListFormats() << ")" << std::endl;

        return 1;
    }

    try
    {
        // read image given as first argument
        // file type is determined automatically
        vigra::ImageImportInfo info(argv[1]);

        if(info.isGrayscale())
        {
            // create a gray scale image of appropriate size
            vigra::BImage in(info.width(), info.height());

            // import the image just read
            importImage(info, destImage(in));

            // write the image to the file given as second argument
            // the file type will be determined from the file name's extension
            exportImage(srcImageRange(in), vigra::ImageExportInfo(argv[2]));
        }
        else
        {
            // create a RGB image of appropriate size
            vigra::BRGBImage in(info.width(), info.height());
```

```
            // import the image just read
            importImage(info, destImage(in));

            // write the image to the file given as second argument
            // the file type will be determined from the file name's extension
            exportImage(srcImageRange(in), vigra::ImageExportInfo(argv[2]));
        }
    }
    catch (vigra::StdException & e)
    {
        // catch any errors that might have occured and print their reason
        std::cout << e.what() << std::endl;
        return 1;
    }

    return 0;
}
```

3.3.2 Creating project

3.3.2.1 Empty Project

In order to start with the most general project, you can start with an empty project in Microsoft Visual C/C++. This can be done by selecting "File-New", and then selecting the "Win32 Console Application" in the Projects-tab. Select a name for the project and press OK.

3.3.2.2 Add Vigra

To be able to correctly link the developed code, the Vigra base library file should be included. This file was created by the compilation of the impex-project in the Vigra-workspace. See section 3.2.3 on page 62 for more info on this.

The compilation was successful if a file called **vigraimpex.lib** was created in the also newly created *lib/windows*-folder. Thus, for example, the file should be located in a folder called `C:\Vigra\vigra1.1.6\lib\windows`.

This folder should be added to the MSVC's list of library directories. This can be done by selecting "Tools-Options", the tab "directories","library files" and there adding the folder in question.

3.3.2.3 Create Standard Project

The following will create the standard code, which is standard, because it basically just opens and then writes again the file to the filenames given at the command prompt.

Select "File-New-C/C++ Sourcefile", give it a name and press OK. Then copy the code from the file **convert.cxx** which you can find in the *Vigra/Src/Examples*-folder. Or which you can find here at section 3.3.1 on page 64.

3.3.2.4 Change Project Settings

Also the project settings need to be adjusted. The vigraimpex, jpeg and tiff libraries need to be included in the project.

This can be done by selecting "Project-Settings" from the Menu, and then going to the Link-tab. The following three files need to be added in the "Modules"-field:

- vigraimpex.lib
- libtiff.lib
- libjpeg.lib

This is shown in the picture:

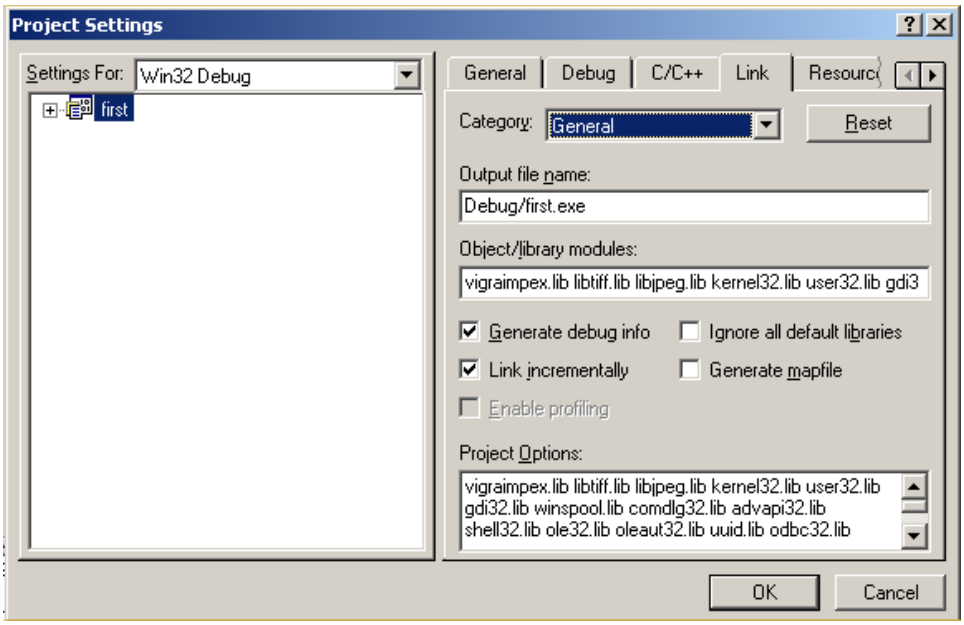


Figure 3.2: Vigra Project Settings

After compiling this successfully, Vigra can be used completely.

3.4 Why the Vigra-project was postponed

At this point of the project, after the successful installation and testing of the Vigra-libraries, it was decided by the GVA to postpone or call off this part of the project. There are several reasons for this decision.

- First, the conversion of the algorithms on both platforms had to be finished by the end of June, meaning that for the developing of the

algorithms with Vigra, only three and a half weeks were left. This tight schedual, considering the rather time-consuming development using Vigra, was the main reason for the postponing.

- Secondly, the Vigra-libraries did not turn out to be what they were expected too. Vigra is a very powerfull image processing library, but a lot of things needed to be implemented. Vigra uses extensively of templates, which makes the code very extensible, yet for a beginning programmer quite difficult to use. Also, if a similar function exists in the Vigra libraries, the global point of view of the function is different than that in Matlab. This makes it necessary to extensively test the accordance between the different parameters of the functions, which is time-consuming. This implies that for every line of code in Matlab a thorough research is needed, and therefor is not a five-minute task.

- Thirdly, the functions that were converted to Vigra, and that were able to produce similar results as in Matlab, turned out to be slower than their counterparts in Matlab. Since the execution time was one of the reasons to convert the code to C/C++, this was a major disadvantage of Vigra. Since for only one function execution time differed a lot, in an algorithm using multiple functions, the results would have been even worse.

- Last, after the conversion to both platforms, a GUI needed to be made to make the processing of the images easier and more convenient. The intention is to make this GUI using QT, a platform independent GUI-designer for C++. Yet this project, performed by another student in the school, did not advance as wanted. Therefore a faster method to make the GUI was needed. The time that came clear by postponing Vigra was used to create this GUI in Java.

Chapter 4

GUI

4.1 Why the GUI

The final stage of this project consisted out of developing a GUI[1] for the processing of the algorithms. Although all the converted algorithms can be seen as stand-alone applications, and can thus be ran without the necessity of any other program, a GUI has several advantages.

A GUI makes it possible for a non-expert user to easily use the algorithms. He does not need to know any of the functioning of the algorithm, nor does he need to know what kind of input is expected, and what kind of output is produced. It is only necessary to know how the gui itself works.

The usage of a graphical interface also makes it possible to automate the process. The command-line based stand-alone application has the disadvantage that it can only work on one image, or on a pair of images at the same time. Most of the times a whole set of images needs to be processed, and thus invoking the stand-alone application for every image separately would be time-consuming and inefficient. One possibility is to adjust the algorithm for the usage of multiple images, but this would require a thorough reimplementation. A gui makes it easy to do this, since a gui can take care of the sequential processing of the images. The user only needs to press "start" in order to begin the processing.

The last reason is that in the Matlab-environment already a gui was developed. The extension of this gui to a platform independent language is merely a logical step since also the Matlab-algorithms were converted. Also in a test-phase, the people at the hospital were introduced with the GUI in Matlab. Designing a similar GUI would make the step to the new system easier.

[1] Graphical User Interface

4.2 Developing environment

4.2.1 Java Depeloping Kit

Java was chosen as programming language for the GUI. The reason behind this is that Java provides a fast and easily understandable way to create GUI's. The developing time of the gui using Java is significantly shorter than when another option, of which QT[2] is an example, would have been chosen. This because of the available knowledge of this language and developing, as well as the easy adaptability towards the future. Java's home page is located at `http://java.sun.com`. The version of the JDK used in this project was version 1.4.1.

4.2.2 Borland Jbuilder

As developing environment Borland JBuilder 8.0 Personal was chosen. The homepage of Borland can be found at `http://www.borland.com`. This version has limited features, but is free to use and is sufficient for our needs. This environment was chosen because of the programmer's experience with this environment, which made clear that an easy and fast development of the GUI was possible.

4.3 Java

In this section I will give a short description on what is Java.

4.3.1 Java the trinity

Java as a whole consists of three distinct, but interlocking, parts:

- an object-oriented language
- a virtual machine, which provides binary portability across platforms
- a platform, or core library, which provides portable system and GUI services

Very little in Java is completely new, but part of its power is the way that it brings together all these ideas into a single development platform.

4.3.2 Java the language

Java is an excellent object-oriented language that has borrowed many of the good bits from previous languages without carrying too much baggage along with them.

[2] `http://www.trolltech.com/products/qt`

It is a general-purpose language; you could write pretty much anything in it. Sun's javac compiler is written only in Java, for eample. It's better for some things than others, of course. I'll address that later.

Syntactically, Java resembles C for the most part, and has pretty standard C looping constructs and operators. Architecturally, it resembles SmallTalk more than C++, since it includes garbage collection, and (except for primitives) everything is a first-class object.

Java is stricter than C in some respects; for example you must use a boolean expression in conditionals, not integers or pointers or what-have-you. On the whole, however, it manages to be both cleaner and more forgiving than C without being as annoying as, say, Pascal to do real work in. It seems to already be replacing Pascal and C as a first-language in universities; the language is set up so that a good compiler can catch most of the stupid mistakes that trip up youngsters, and its robust design prevents the more bizarre forms of memory allocation and access bugs.

Java was designed from the beginning to prevent the sort of access violations, memory leaks and array indexing errors that comprimise the security and stability of C and C++ applications. Most introductions to Java would say that they did that by doing away with pointers; I would disagree - everything (again, except for primitives) is a pointer in Java, in the sense that it is a reference rather than a pure value. More complex references are planned for Java 1.2 and later, including weak references that don't prevent garbage collection.

But most really severe bugs in C apps come from bad array references, memory management problems, dangling or mangling pointers, etc. None of this is possible in Java without a lot of pretty pathological work. The addition of garbage collection eliminates whole rafts of problems, and bounds checking prevents most of the rest. Wimpy? Perhaps - but then again, having the compiler decide which registers to put stuff in was considered wimpy at one point. As the (cost of programmer time/cost of computer time) ratio gets bigger, doing your own memory management looks less and less attractive.

4.3.3 Java the virtual machine

You can certainly compile java language apps to native binaries, but part of the original idea was binary portability, a much more stringent requirement. This is obviously required when you are sending code over a hetorogenous network and expecting it to work on all manner of hardware and operating systems.

To get binary portability, you have to have an intermediate form, usually referred to as pseudocode or p-code. This is a machine language for a machine that is built in software, that then translates the p-code to real machine code on the fly in some fashion. P-code is nothing new; some

SmallTalk implementations use it, and some microkernel-based operating systems use it. And a software emulator is simply a virtual machine whose p-code is the code for an actual CPU.

A Java VM has to conform to the Java VM specification, which specifies the number and type of registers, endianness, and the size of integers and longs and such. This last prevents Java applications from breaking on assumptions made about the size of things. In addition, Java p-code can be verified at run-time to see if it came from a valid compiler; this prevents weird virii and such from being dumped on an unsuspecting VM to exploit faults in the implementation.

4.3.4 Java the platform

This is the part Microsoft is scared about. There have been cross-platform toolkits. There have been virtual machine environments. But put them together, and you have binary-portable, cross-platform, fully GUI, system and network-capable applications. Instead of locally-installed, client-server, platform-specific applications, you have applets, or Marimba channels, or applications servers that don't care what sort of machine it's serving up applications to, as long as the machine speaks Java. And as more corporate information systems are written in Java, the need to standardize on single business-wide platform diminishes, giving everyone more of a choice for their desktops.

4.4 How to work with the GUI

4.4.1 The GUI in general

The GUI was developed with the original Matlab-GUI in mind. It is attempted to provide the same functionality and system as the GUI developed in Matlab. The GUI in Matlab was developed by David Lopez as a part of his project. The description given here is the general description of the GUI. For the different algorithms, only small parts of the GUI change, mostly supportive messages. The main idea and functioning of the GUI remains unchanged.

For the understanding of the GUI, the following section will describe which options are possible with the GUI, and how to use it.

When the GUI is started, the main screen of the GUI will appear, this will look like the picture below. You can adjust the size of the screen if wanted.

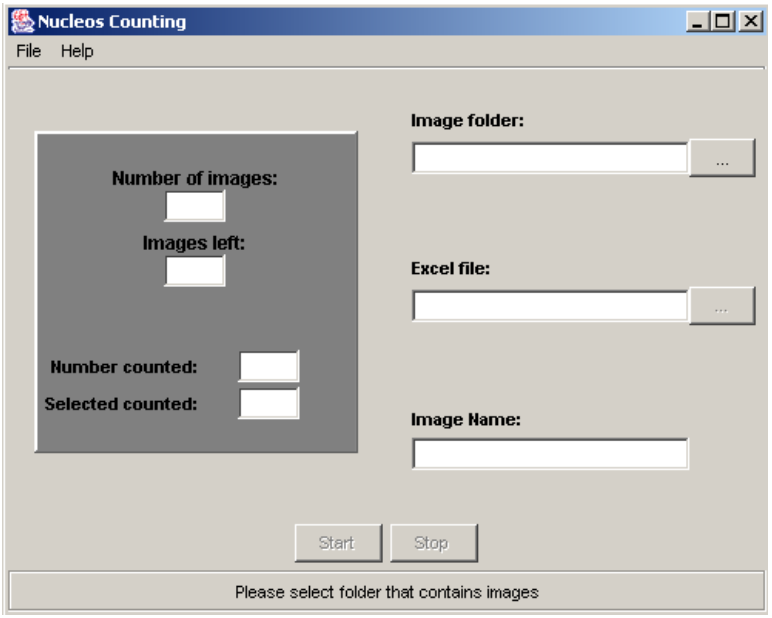


Figure 4.1: GUI Main Screen

The left part of the screen is used to display the results of the processing. The right screen is the selection screen, where the user can choose the settings for the processing. All the settings need to be set in order for the algorithm to work.

Basically, the user needs to perform the following steps:

- Select the folder that contains the images
- Select an excel-file to store the results
- Press start to begin processing

At every step, the status bar will display a message showing which step is required next.

4.4.1.1 Select folder

In this section the user can select the folder which contains the images. Depending on the type of algorithm, the required contents of the folder can differ. The program will check whether the contents of the folder are correct for the algorithm to be used, and will report if this is not the case.

The dialog to select the folder can be opened by pushing the button next to the text field. At the time, this is the only button that works, so no problems should arise. The dialog to select the folder then appears:

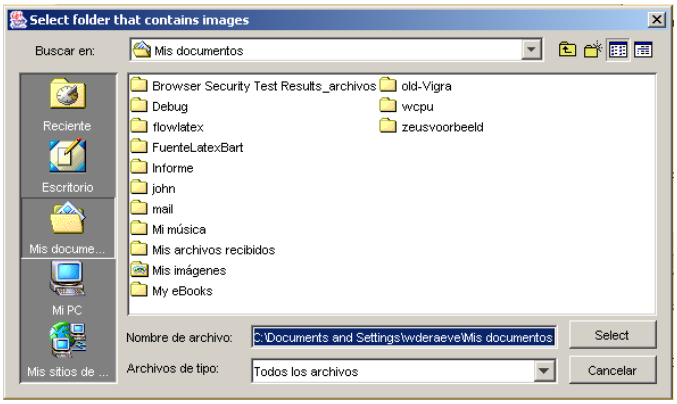


Figure 4.2: Select image folder

You are then able to select the folder containing the images, by going to the directories as normal in a windows-environment.

When a folder is selected that contains no valid images, this means that the folder does not contain images suited for the algorithm, the following error is displayed:

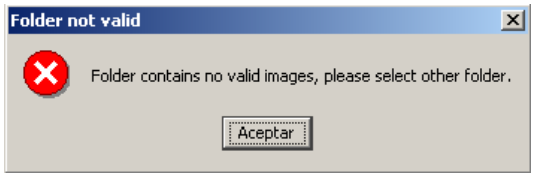


Figure 4.3: Folder contains no valid images

After pressing OK, the user can select a new folder. Another error that may occur is that some of the requirements of the folder for the algorithm are not met. For example, some algorithms take a set of images at a time, for example 2. It is then necessary that there are an whole number of sets available. For example a set consists out of `A1n.jpg` and `A1.jpg`. If thus the number of images ending with `n.jpg` is different than those with `.jpg`, the computing is not possible, and an error like the following will occur:

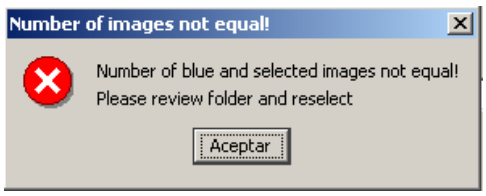


Figure 4.4: Number not valid

The user can now reselect his folder.

If no errors occur, the name of the folder is now shown in the text field, and now it is possible to select the results-file. Also the GUI will be updated, and the number of images or image sets to be processed will be shown.

4.4.1.2 Excel-file

The user can now select the excel-file in which the results of the processing will be stored. Again, a similar screen as the one for choosing the image folder appears, yet now it is possible to only select files with the `xls`-extension, or to type in a new filename with the `xls` extension in the field.

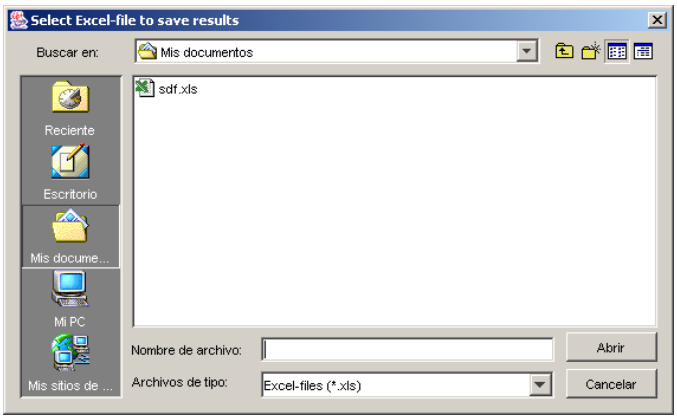


Figure 4.5: Choose Excel-file

The user now has two possibilities:

- Select an existing file
- Create a new file

When the first option is chosen, and thus an existing file is selected in the dialog, the existing file will be deleted. The user is reported with this fact with a warning message:

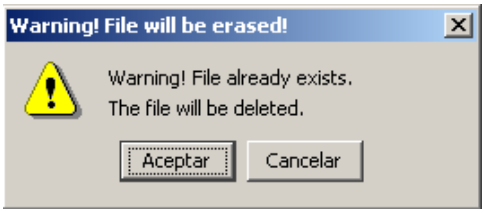


Figure 4.6: File will be deleted.

If the user chooses to accept this, the file will be deleted. Otherwise it is possible to enter a new filename or select another file.

The second option is to type in a filename in the text field. It is required to fill in a filename that carries the xls-extension. If a filename with another extension is filled in, the following error-message will be displayed:

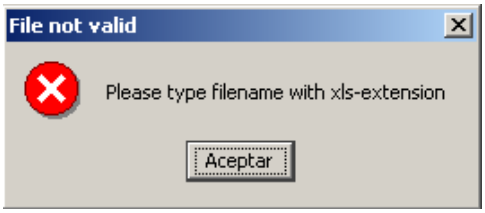


Figure 4.7: Excel extension necessary

The user can now select a new file. If a file without extension is inserted, the extension *xls* will be automatically added.

When no errors occur, the text field will show where the results file will be stored. It is now possible to start processing.

4.4.1.3 Processing

The processing can be started by pressing the start button. During the processing, the set of images is being calculated, and the results of the algorithm are written to the excel file chosen before. Also the time of the calculation is kept, so it is possible to get a view on the time it took to process the images.

After the processing, the status bar will show the time it took to process the set of images, as well as the average time per set of images:

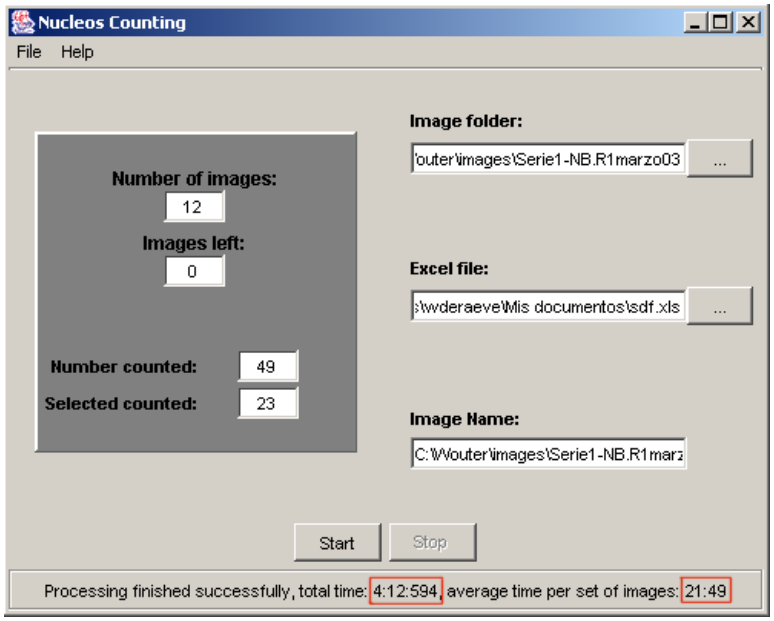


Figure 4.8: Results of processing

4.4.2 FHV GUI

The GUI has another version, created for the FHV algorithms, which is completely similar except that the user has the option to choose the algorithm through a dropdown-box, as you can see below:

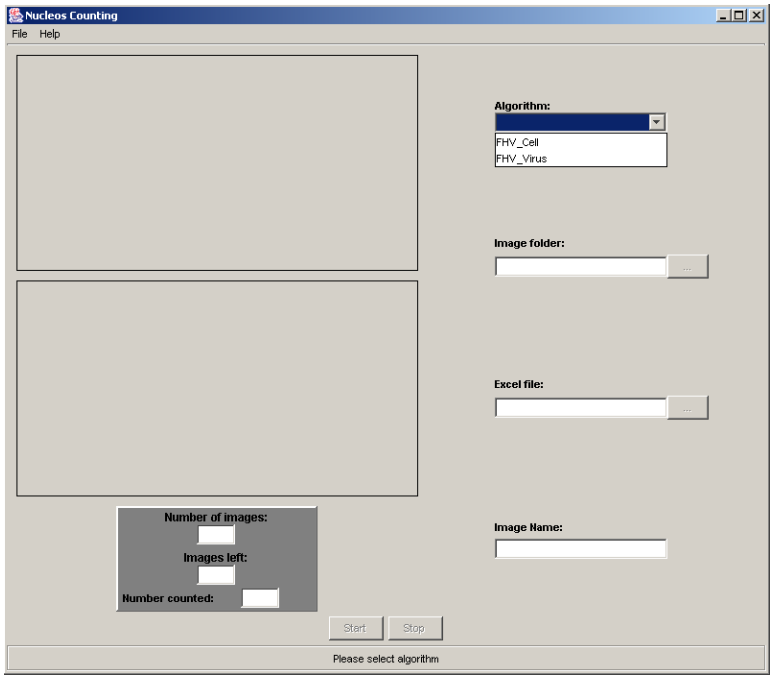


Figure 4.9: GUI for FHV Algorithms with Visualization

After selecting the algorithm, the process is exactly equal as the one described above. This algorithm/gui also creates the resulting images, and displays them. After processing one image, the user is given the possibility to accept or deny the result, or to stop the processing altogether, which resets all settings and allows the user to reselect the options.

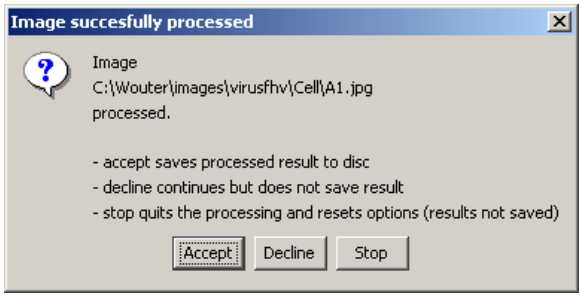


Figure 4.10: Accept/Decline/Stop-dialog FHV

4.4.3 GUI in Linux

The gui in Linux is completely identical as the one available and described in Windows.

4.5 Some used classes

In this section I will present the set of classes used outside the standard Java classes. These are thus not the classes for making the gui itself, such as provided by Swing in Java, but the classes used to perform certain actions.

4.5.1 Utils

This class has only one method, called getExtension(File f) which returns the extension of a certain File.

The method is pretty straightforward, it takes the File as input and returns the extension of the File as a String.

```
import java.io.File;

public class Utils
{
  /**
    * Get the extension of a file.
    */
  public static String getExtension(File f)
  {
        String ext = null;
        String s = f.getName();
        //gets index of "." that seperates filename from extension
        int i = s.lastIndexOf('.');

        if (i > 0 &&  i < s.length() - 1) {
            ext = s.substring(i+1).toLowerCase();
            //get only extension
        }
        return ext;
  }
}
```

4.5.2 ExecFileFilter

This class extends the FileFilter class. It makes sure that only files with the xls extension are shown or can be selected.

In order to make a FileFilter, a class extending the FileFilter-class needs to be declared, since the FileFilter class is abstract. This class needs to implement at least the *accept(File f)*-method from FileFilter. The FileFilter-class reference can be found here: `http://java.sun.com/j2se/1.4.1/docs/api/javax/swing/filechooser/FileFilter.html`.

There are two methods that can be implemented. Although only the *accept()*-class needs to be implemented, it is advisable to implement *getDescription()* as well.

4.5.2.1 accept(File f)

This function states whether a certain file is accepted. If the file should be accepted, the function should return true, otherwise false.

The function uses the method *getExtension* from the class *Utils* to get the extension of the File.

Since only files with the extension *xls* are accepted, *accept* only returns true when the extension is correct.

```
public boolean accept(File f)
  {
    String extension=Utils.getExtension(f);
    if (extension != null)
    {
      if (extension.equals("xls"))
        return true;
      else
        return false;
    }
    return false;
```

4.5.2.2 getDescription()

This function returns a description of the FileFilter, in our case `Excel-files (*.xls)`. This function is used within a dialog to select files, and is shown in the dropdown list.

4.5.2.3 Full code

```
import javax.swing.filechooser.*;
import java.io.File;

public class ExcelFileFilter extends javax.swing.filechooser.FileFilter
{
  final String excelfileext="xls";

  public ExcelFileFilter()
  {
  }

  public boolean accept(File f)
```

```
  {
    String extension=Utils.getExtension(f);
    if (extension != null)
    {
      if (extension.equals("xls"))
        return true;
      else
        return false;
    }
    return false;
  }
  public String getDescription()
  {
    return "Excel-files (*.xls)";
  }
}
```

4.5.3 StreamGobbler

This class is need in order to correctly use the *Runtime.exec()*-function, which runs a program outside the Java program. This program is a new Process within the environment.

Because some native platforms only provide limited buffer size for standard input and output streams, failure to promptly write the input stream or read the output stream of the subprocess may cause the subprocess to block, and even deadlock. Since our program produces output, itis necessary to capture this output.

The output produced by the process, thus the external program, is a Stream, which can be obtained by the *getInputStream* and *getErrorStream*-methods. These streams should then be buffered to make sure all output can be caught. Since a BufferedReader can only be created through a Reader, first a Reader needs to be declared.

For more info on the subject, see this link: `http://www.javaworld.com/javaworld/jw-12-2000/jw-1229-traps.html`.

4.5.3.1 Full code

```
import java.io.*;

class StreamGobbler extends Thread
{
    InputStream is;
    String type;
```

```
    StreamGobbler(InputStream is, String type)
    {
        this.is = is;
        this.type = type;
    }

    public void run()
    {
        try
        {
            InputStreamReader isr = new InputStreamReader(is);
            BufferedReader br = new BufferedReader(isr);
            String line=null;
            while ( (line = br.readLine()) != null)
                System.out.println(type + ">" + line);
            br.close();
            isr.close();
            } catch (IOException ioe)
              {
                ioe.printStackTrace();
              }
    }
}
```

4.5.4 OS_MillisConvertor

This class provides a way to convert milliseconds to weeks, days, minutes, seconds and milliseconds, and provides methods to print these.

This is useful, since a timer in Java works with milliseconds, and in order to display a more understandable and clearer visualisation, a display in minutes, etc is easier.

The original class is written by Joe Sam Shirah. The main explanation and source code can be found here: `http://www.conceptgo.com/articles/MillisConverter.html`.

I have added a method which prints the days, hours,... but with only a : between the several parts. This method, which is called *formatEndStringSimple*, which is based on the method *formatEndString*.

4.5.4.1 formatEndStringSimple

```
public StringBuffer formatEndStringSimple(
                        boolean bPrintRemaining )
  {
    StringBuffer sb = new StringBuffer( 50 );
```

```
    bPrintRemaining = bPrintRemaining ?
                      bPrintRemaining : ( iHours != 0 );
    if( bPrintRemaining )
    {
      sb.append( ":");
      sb.append( iHours );
    }

    bPrintRemaining = bPrintRemaining ?
                      bPrintRemaining : ( iMinutes != 0 );
    if( bPrintRemaining )
    {
      sb.append(":");
      sb.append( iMinutes );
    }

    bPrintRemaining = bPrintRemaining ?
                      bPrintRemaining : ( iSeconds != 0 );
    if( bPrintRemaining )
    {
      sb.append( ":");
      sb.append( iSeconds );
    }

    bPrintRemaining = bPrintRemaining ?
                      bPrintRemaining : ( iMillis != 0 );
    if( bPrintRemaining )
    {
      sb.append( ":");
      sb.append( iMillis );
    }

    if (sb.indexOf(":")==0)
    {
      return new StringBuffer(sb.substring(1));
    }
    else return sb;
  }
```

Chapter 5

Used Tools

5.1 Borland JBuilder 8.0

Since the decision was taking to develop the GUI in the Java programming language, because of the fast developing time, and the transportability between platforms, a development environment was useful.

As integrated development environment Borland's JBuilder 8 Personal edition was chosen. JBuilder's webpage can be found here: `http://www.borland.com/jbuilder/index.html`. There were several reasons to choose this developing environment:

- Ease-of-use
- Fast developing
- Free
- Programmer's Experience

The Personal version, which is sufficient for our needs, is free to use and can be downloaded from the Borland JBuilder website. It provides an easy way to develop graphical user interfaces and applications using the Java programming language. Using JBuilder, no detailed knowledge of the Java Swing elements is needed, although a minor knowledge is advised. Through this environment it is possible to develop a graphical user interface or application swifter than using line-based programming.

5.1.1 Downloading and Installation

5.1.1.1 Download

Borland JBuilder 8 Personal edition, which is free to used, can be downloaded from Borland's Website. You can start from the main page, or

go directly to the JBuilder download page, which is located here: `http://www.borland.com/products/downloads/download_jbuilder.html`.

There the Personal edition should be chosen. You will be asked to fill in some personal details in order to be able to obtain a valid key to activate the software. This key will be sent by email.

It is also advisable to download the documentation and samples available on the website. These can be found here: `http://info.borland.com/techpubs/jbuilder/index8.html`. If wanted, and which is advisable, the documentation can be integrated with the JBuilder IDE. In order to obtain this, the documentation pack needs to be downloaded. The direct links to these packs are the following:

Documentation `ftp://ftpc.borland.com/pub/jbuilder/jb8personal/jb8docs.zip`

Samples `ftp://ftpc.borland.com/pub/jbuilder/jb8personal/jb8samples.zip`

5.1.1.2 Installation

JBuilder's installation is straightforward and easy. Only the following steps are needed:

1. unpack the downloaded zip-file
2. run the executable (usually called 'per_install.exe')
3. accept all license notices
4. choose a folder that contains no spaces (JBuilder will not run otherwise)

The documentation and samples installation is very likewise. Again unpack the zip-file, run the install file, select your JBuilder folder, and press OK.

5.1.2 Getting started

Getting started with JBuilder is quite straightforward. I will post here a short tutorial on how to start a first project within JBuilder.

1. Start a new project by selecting 'New Project' from the 'File'-menu. The following screen will appear. Select a name and press Finish. The other steps in the project can be left on their default setting.

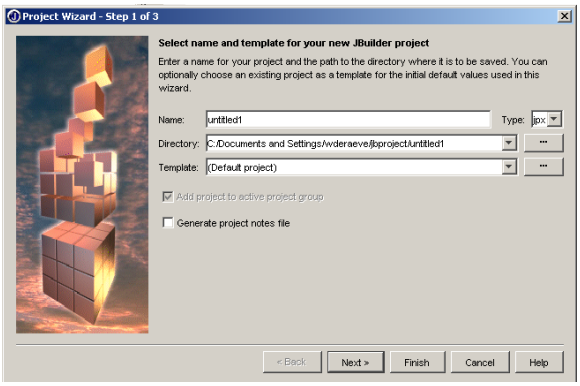


Figure 5.1: Step 1 of creating a new Project

2. Start a new application by selecting 'Application' from the 'General'-tab in the 'File"New'-menu. Fill in a name for the package. This is the package under which all files will be stored. Also give a name to your application. Then press next.

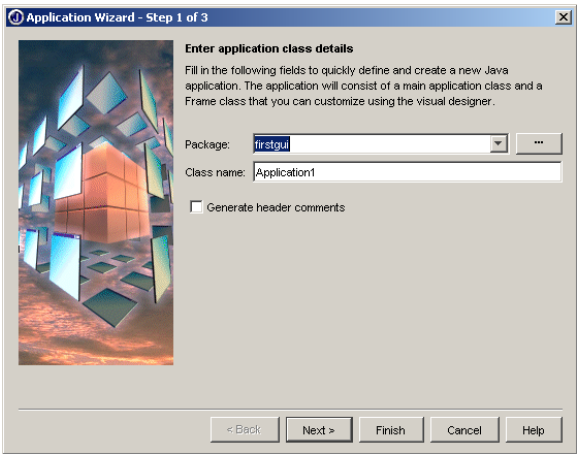
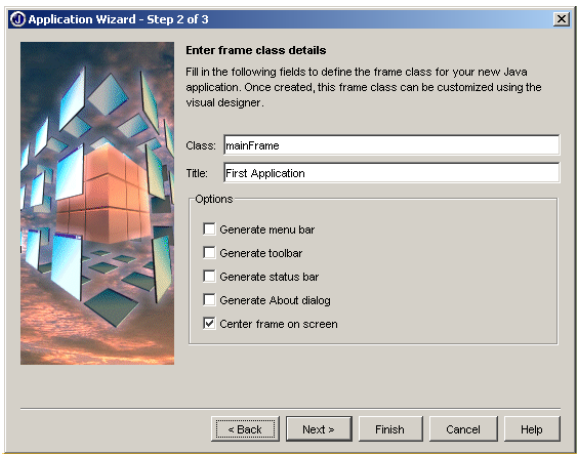


Figure 5.2: Step 1 of the New Application

3. Fill in a name for the frame. This is the main frame of the graphical user interface. Also give a title to the frame. This is what you will see in the bar at the top of the screen. Most of the times this is the name of your application. You can then select which items JBuilder should create automatically. If you want to start with a bald gui, only select the last box. After filling in these details, press finish.



After doing this, you will have created a first application on which you can built your full application.

You can now start adding elements using the built in designer in JBuilder, by selecting the *Design*-tab.

When first starting with JBuilder and GUI's, it is advisable to follow one of the tutorials available within JBuilder. That is, if you have installed the documentation.

Go to the tutorials by selecting 'JBuilder Tutorials' from the 'Help'-menu. If you have installed the documentation and the samples, you will see a list of tutorials on the right hand of the screen. Click on 'user interface tutorials'. You will now have three tutorials to choose from:

- "Building a Java text editor"
- "Creating a UI with nested layouts"
- "GridBagLayout tutorial"

The best to begin with is *Building a Java Text Editor* so I'd suggest to start of with that one. It will provide a basic knowledge of gui's, necessary to work with JBuilder.

5.2 MikTex 2.3

To write this book, two possibilities were offered. The first was to type the book using a standard word processor like *Microsoft Word*, and the second was to work using LaTeX.

This book was written using the latter, and this because of several reasons:

- Free
- Positive experience in the past
- Widely accepted
- Available for many platforms
- Faster
- Known difficulty of Word with large documents

The template used to create this book can be found at the back of this book, and may be used freely.

The Not So Short Introduction to LaTeX2e is an 87 page introduction LaTeX2e by Tobias Oetiker et al (DVI file). Updated Apr, 1999. Worth a read.

5.2.1 What is LaTeX?

LaTeX is a document preparation system for high-quality typesetting. It is most often used for medium-to-large technical or scientific documents, but it can be used for almost any form of publishing.

LaTeX is not a word processor! Instead, LaTeX encourages authors not to worry too much about the appearance of their documents, but to concentrate on getting the right content.

LaTeX contains features for:

- Typesetting journal articles, technical reports, books, and slide presentations.
- Control over large documents containing sectioning, cross-references, tables and figures.
- Typesetting of complex mathematical formulae.
- Advanced typesetting of mathematics with AMS-LaTeX.
- Automatic generation of bibliographies and indexes.
- Multi-lingual typesetting.
- Inclusion of artwork, and process or spot color.
- Using PostScript or Metafont fonts

LaTeX is based on Donald E. Knuth's TeX typesetting language. LaTeX was first developed in 1985 by Leslie Lamport, and is now being maintained and developed by the LaTeX3 Project[1]. LaTeX is available for free by anonymous ftp[2].

5.2.2 What is MiKTeX?

MiKTeX (pronounced mik-tech) is an up-to-date implementation of TEXand related programs for Windows (all current variants) on x86 systems.

MiKTeXs main features include:

- easy to install
- complete: 887 packages (fonts, macros, ...) are included
- living: the package repository is updated periodically
- easy package management: a wizard helps you to keep your installation up-to-date
- the DVI viewer Yap allows for an optimized edit-compile-view cycle
- MiKTeX is open source

The MiKTeX distribution consists of the following components:

- TeX: the classic TeX compiler
- pdfTeX, e-TeX, pdf-e-TeX, Omega, NTS: various TeX variants
- Dvipdfm: converts TeX output into PDF documents
- MetaPost: converts picture specifications into PostScript commands
- a complete set of macro packages and fonts
- Yap: a viewer for TeX output
- TeXify: a TeX compiler driver
- MiKTeX Options: assists in configuring MiKTeX
- Lots of utilities: tools for the creation of bibliographies & indexes, PostScript utilities, and more.

The current version of MiKTeX is version 2.3. The homepage of MiKTex can be found at `http://www.miktex.org`.

Thus, in plain words, MiKTeX provides a compiler in order to process LATEX-documents under the windows environment.

[1] `http://www.latex-project.org/latex3.html`
[2] `http://www.latex-project.org/ftp.html`

5.2.3 Installation of MiKTeX

The installation screenshots are taken from the MiKTeX website, since the installation was already performed on this pc, and no screenshots were taken then.

Four steps are necessary to install MiKTeX from the Internet:

1. Check to see if the prerequisites are met.
2. Choose a package set.
3. Download MiKTeX.
4. Install MiKTeX.

5.2.3.1 Prerequisites

These are system requirements imposed by MiKTeX. These imply the availability of certain files on the system, yet since Windows 98 these are automatically available in any windows version.

File Name	Required Version	Distribution Platform
comctl32.dll	5.80.2614.3600	Common Control Library Patch (50comupd.exe)[3]
wininet.dll	4.70.0.1300	Internet Explorer 4.0 or better

Table 5.1: Required Windows components

5.2.3.2 Choosing a package set

You can choose between three package sets: "Small MiKTeX", "Large MiKTeX" and "Total MiKTeX".

Small MiKTeX This is a basic MiKTeX system with TeX, pdfTeX, recommended LaTeX packages and Type 1 fonts. This uses about 100 MB disk space.

Large MiKTeX This adds O(mega), ConTeXt, e-TeX and more. This uses about 200 MB disk space.

Total MiKTeX This includes all available packages. This uses about 500 MB disk space.

You can start with the small package and add packages later using the Package manager, or immediately select a larger package.

5.2.3.3 Downloading MiKTeX

It is advisable to install MiKTeX using the installation wizard. This installation wizard can be downloaded from numerous places. The latest version can always be found at `http://prdownloads.sourceforge.net/miktex/setup.exe?download`.

1. Start the wizard (setup.exe). The welcome page will be presented:

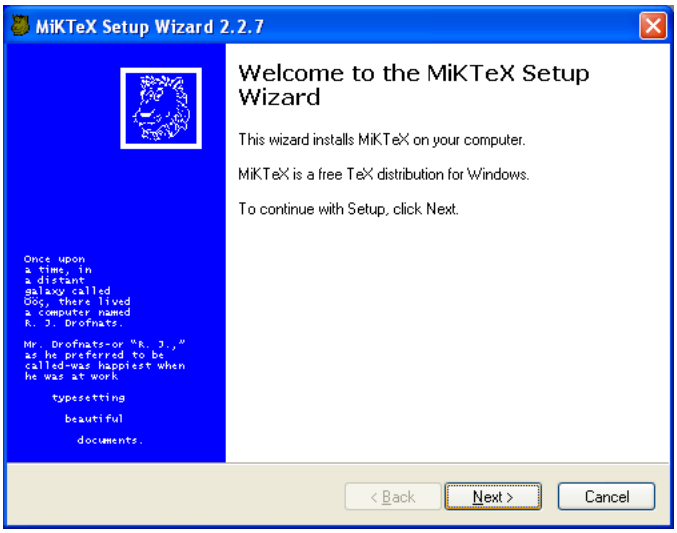


Figure 5.3: MiKTeX Welcome screen

   Click Next ¿ to advance to the next page.

2. Choose "Download only" as the primary setup task, and press Next to advance

3. Choose the package set you wish to download:

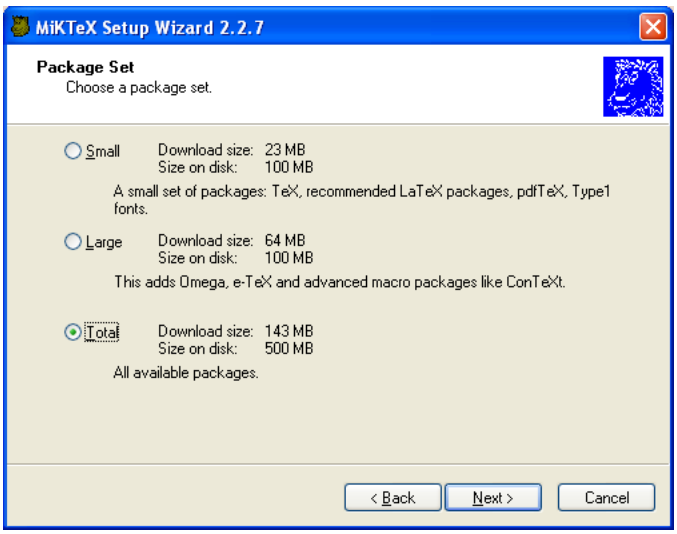


Figure 5.4: Select package

4. It will now provide with a list of mirrors where you can download the packages. It is advisable to select a mirror near your location.

5. The next step is to select a folder where the packages will be downloaded to. This is not necessarily the same folder as the one where MiKTeX will be installed to later on. It is even advised to put this in a different folder, in order to easily be able to reinstall, if needed, without downloading again.

6. The download will start, and you will be able to see how much time is left in order for the download to complete.

7. When the download is complete, you can proceed to the following screen:

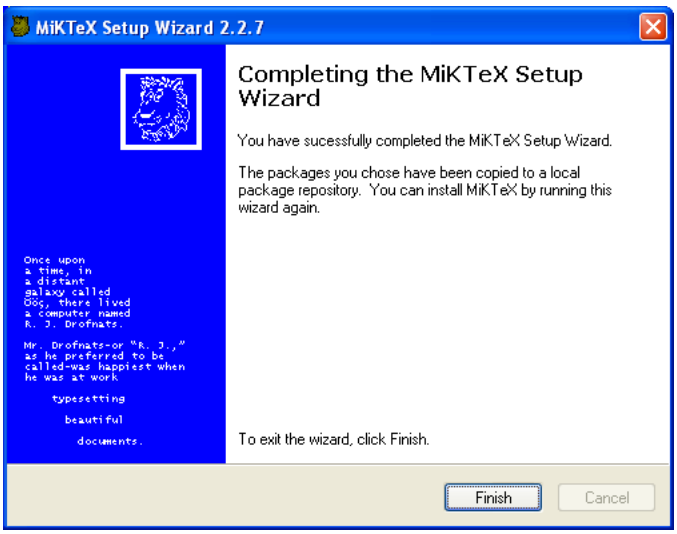


Figure 5.5: Download Complete

5.2.3.4 Installation

1. Start the wizard again (setup.exe)
2. Now select the install-option:

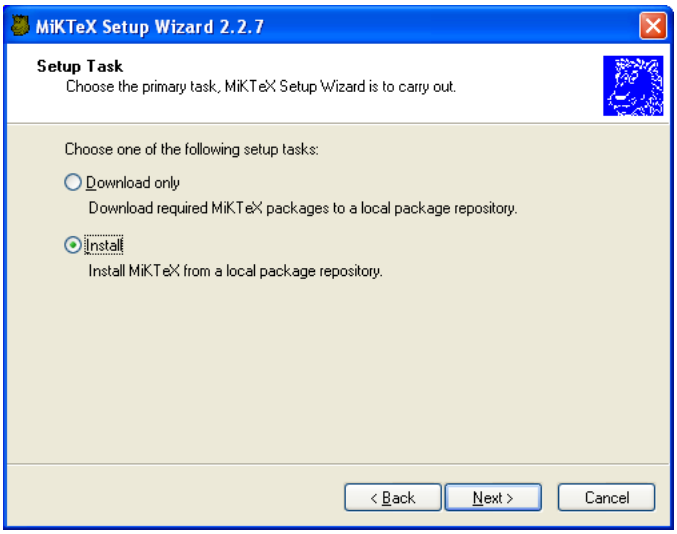


Figure 5.6: Install Option

3. Select the package you wish to install, preferably this is the same as the downloaded one, but you can also select a smaller package.

4. Select whether to install a shared or private environment. The shared environment lets other users of the computer also use the miktex environment.

5. Select the folder where you have downloaded the packages.

6. Next, select the folder where you would like to install MiKTeX, remember that depending on the packages you would like to install, up till 500MB of disk space can be needed.

7. The next option lets you select whether you would like to incorporate other texmf-directories. These are directories that contain additional packages in the Network Neighbourhood. You can safely defer this decision, because you can subsequently add additional TEXMF directories with the help of MiKTeX Options.

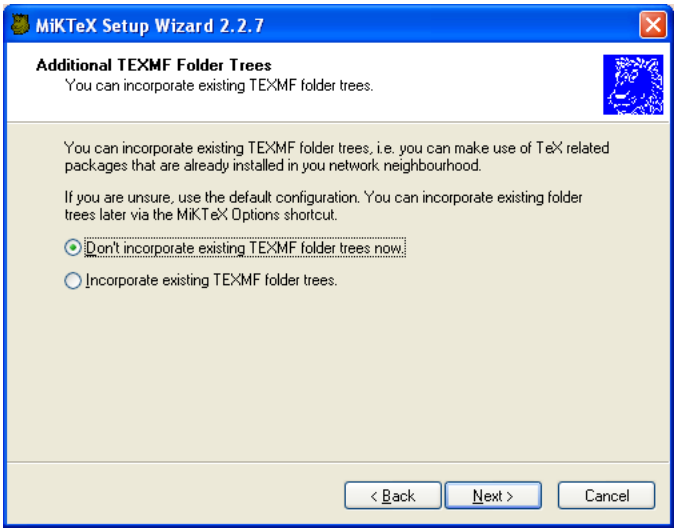


Figure 5.7: Remote TexMF Folders

8. After a review of the chosen options, the installation can begin.

9. After a successful installation, the following screen is shown:

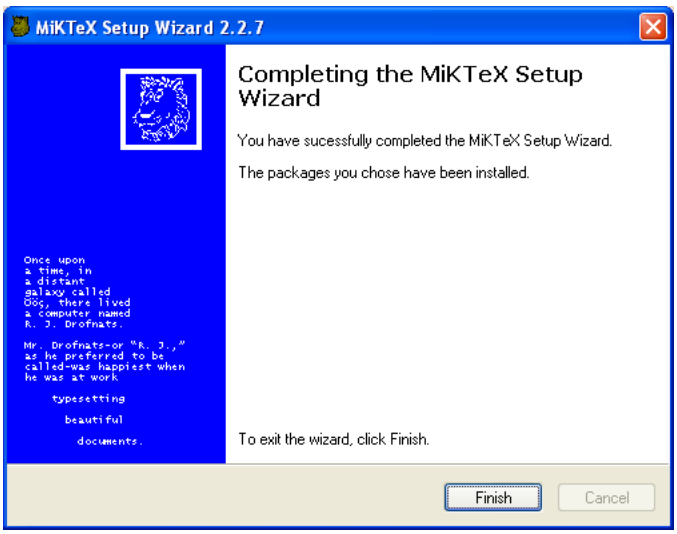


Figure 5.8: Install Finished

5.2.3.5 Using MiKTeX

In order to avoid needing to know the command line based compilation of MiKTeX, we will use an integrated developing environment, called TeXnicCenter to develop our LaTeX-files. See more about this in the following section.

5.3 TeXnicCenter 1b6.0

TeXnicCenter is an integrated development environment (IDE) for developing LaTeX-documents on Microsoft Windows (Windows 9x/ME, NT/2000).

IDE means, that TeXnicCenter is an application, that integrates all the tools, needed to develop documents with LaTeX, in just one application. You have the editor to write your LaTeX files with, you can start the building process just by choosing a menu item and the output of the LaTeX compiler is written to a window of TeXnicCenter and analyzed, so that you can simply jump from one error, warning or bad box to another one.

Also viewing the generated output is easy with TeXnicCenter. Just choose a menu item and the correct viewer application will be started and if supported by the viewer, the output will be displayed at the position belonging to the current source position in TeXnicCenter.

TeXnicCenter's aim is to support the LaTeX-newbie by providing him the most important LaTeX constructs via menu and by abstracting the use of the LaTeX compiler and other tools like MakeIndex and BibTeX and even support the LaTeX-pro by providing a powerfull, fully customizable

and integrated environment.

The TeXnicCenter's main web page can be found at the following address: `http://www.toolscenter.org/products/texniccenter/`.

5.3.1 Download and Installation

The current latest version of TeXnicCenter is version 1 beta 6.01. This version can be downloaded from the main website, or directly from this address: `http://prdownloads.sourceforge.net/texniccenter/TXCSetup_1Beta6_01.exe?download`.

The installation of TeXnicCenter is easy, and should offer no problems.

5.3.2 Configuration

When first running TeXnicCenter, tell the configuration wizard to configure for use with MikTeX. This will automatically set all paths correctly to use LaTeXcorrectly.

There is only one setting to be changed, this is when the configuring wizard asks to fill in a path for the different viewer. In the textbox for the *DVI-viewer*, you should select the *Yap*-viewer that is installed together with MiKTeX. This viewer can be found in the following folder: `c:\texmf\miktex\bin\yap.exe` if you accepted the default settings of the MiKTeX-installation.

5.3.3 How to Work with TeXnicCenter

The main screen of TeXnicCenter is quite similar to that known in other editors like Word. The two lower toolbars allow you to easily access LaTeX-commands such as *Emphasize*, **Bold**, or allow easy creation of enumerating-environments or likewise, without having to know the exact syntax of the LaTeX-commands.

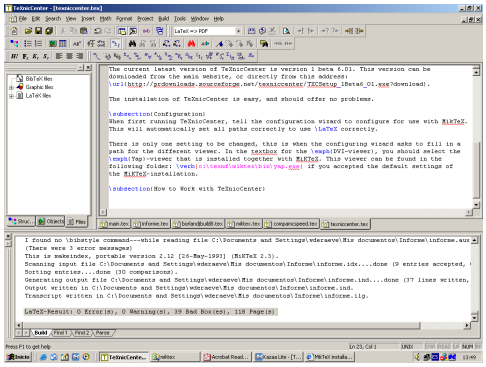


Figure 5.9: TeXnicCenter main screen

Since we are working in Windows, and would like to generate a book available on any platform, Acrobat's PDF-format is most appropriate [4]. Thus in the dropdown button we must select *"LaTex=¿PDF"*.

When working with TeXnicCenter, you have two possibilities:

- Work in a project
- Work in single files

The latter is only useful when working with small documents. When writing a book of substantial size, it is better to create a new project. This allows for easy managing of the different files of the project.

5.3.3.1 Creating a new project

A new project can be started by selecting 'File'-'New Project' in the menu bar. You then get the following screen:

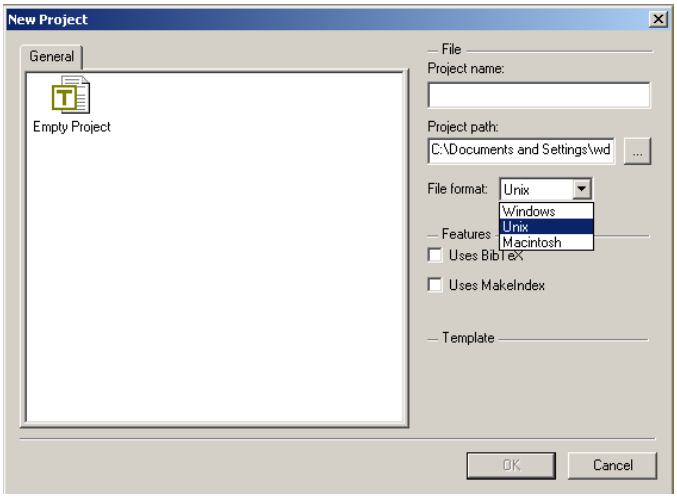


Figure 5.10: New TeXnic Project

Choose a name for the project, and the default empty project template. For compatibility issues, it is advisable to select the UNIX-file format. The other options may be left to default.

You will now have a clean project to start with, and the main file of your project opened. For this main file, you can use the template provided with this book, use others, or start from scratch.

[4] `http://www.adobe.com/acrobat`

5.3.3.2 Compiling the project

After writing your LaTeX-code, using the IDE, or entering your commands manually. The project can be compiled using the buttons known from any other integrated developing environment.

These buttons are located right to the dropdown selection box:

Figure 5.11: TeXnic Build Buttons

The main button is the first, which compiles the entire project. In the box below, the result of the build will be displayed. If no errors occured, the output can be viewed by clicking the preview button, which is located in the same bar.

The interface of TeXnicCenter should be quite intuitive in order to work on more elaborate projects.

5.4 TextPad Text Editor

TextPad is a powerful, general purpose editor for plain text files. During this project, it was several times used to quickly review files, algorithms, source code,... Or to write new test files, test programs, etc.

It is a powerful replacement for any other text editor, offering following features:

- English, French, German, Italian, Polish, Portuguese and Spanish user interfaces.
- A spelling checker with dictionaries in 10 languages.
- Unlimited undo/redo capability. The undo buffer can be optionally cleared when a file is saved, or by using the Mark Clean command.
- Viewer for binary files using a hexadecimal display format.
- Syntax highlighting depending on input file
- User-adjustable highlighting
- Plug-ins for external languages
- External command execution
- Java Compilation/Execution on the fly

- ...

The main webpage of TextPad can be found at `http://www.textpad.com`, where also other plug-ins can be downloaded. These add-ons provide support for syntax-highlighting, command-shortcuts,... for several file types.

A few examples of these add-ons that were particularly useful in the project:

- Syntax definitions for LaTex
- Syntax highlighting for Matlab Release 13 functions and operators. Also Matlab function for generating Matlab keywords.
- Syntax definitions for Java 2 JDK 1.4.0
- Syntax definitions for C and C++ which includes definitions for ATL, WTL, and STL.

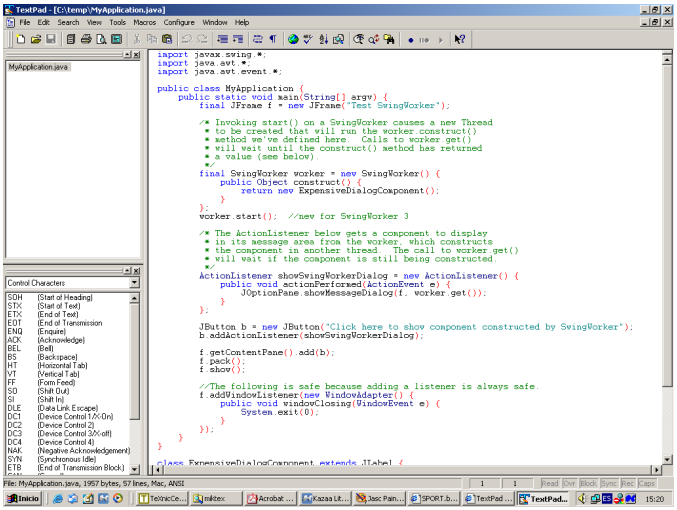


Figure 5.12: TextPad

5.5 Jasc Paint Shop Pro 7

Paint Shop Pro, developed by the Jasc Software company[5], is one of the most used photo and graphics editors, mainly because of the following reasons:

- Easy-to-use

[5] `http://www.jasc.com`

- Compatible with plug-ins of Adobe Photoshop
- Fast

All the screenshots in this book were created and edited using Paint Shop Pro.

The version used in this project was version 7.04. Currently, version 8 is available, of which a 30-day trial version can be downloaded from here: `http://www.jasc.com/download_4.asp?`.

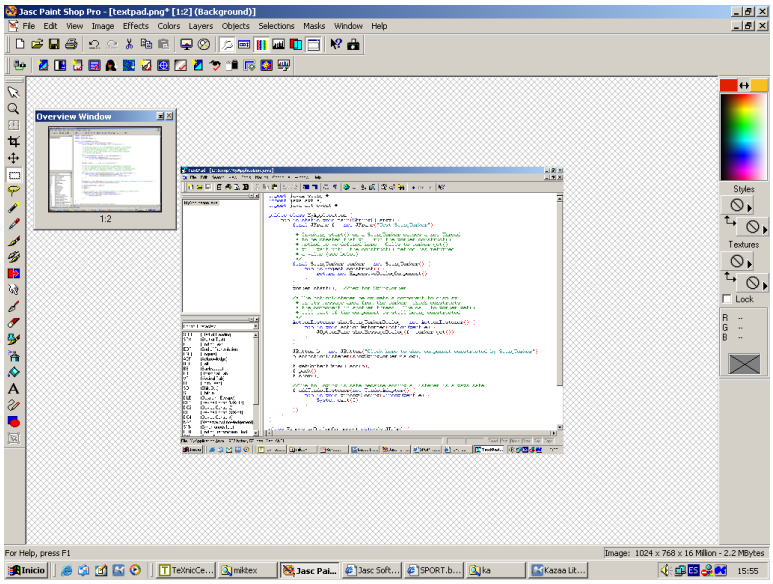


Figure 5.13: Paint Shop Pro

5.6 Jaws PDF Editor 1.1

Jaws PDF Editor, produced by Global Graphics software [6], allows basic manipulation of PDF-documents, which would otherwise be impossible.

The main reason for using the software was to be able to select certain parts of a text out of a PDF-document in order to be able to reuse it in this book.

Other than that, Jaws PDF-editor offers the following features:

View and print PDF files Open, navigate and print any PDF file using Jaws PDF Editor. Navigation

[6] `http://www.globalgraphics.com/`

Manipulate PDF files Jaws PDF Editor allows you to manipulate PDF pages and documents by rotating, deleting, inserting, extracting and re-ordering pages and then saving the changes to the file.

Annotate with comments Jaws PDF Editor includes the ability to create,add and save comments directly in the PDF file.Add "sticky note" comments or highlight, strikeout and underline text.All comment types have a pop-up comment window into which you can type additional information.

Security Settings Jaws PDF Editor allows users to set passwords or change security rights on the PDF file.

Jaws PDF Editor is available as a 30-Day Trial version from their website: `http://www.jawspdf.com/pdf_editor/index.html`

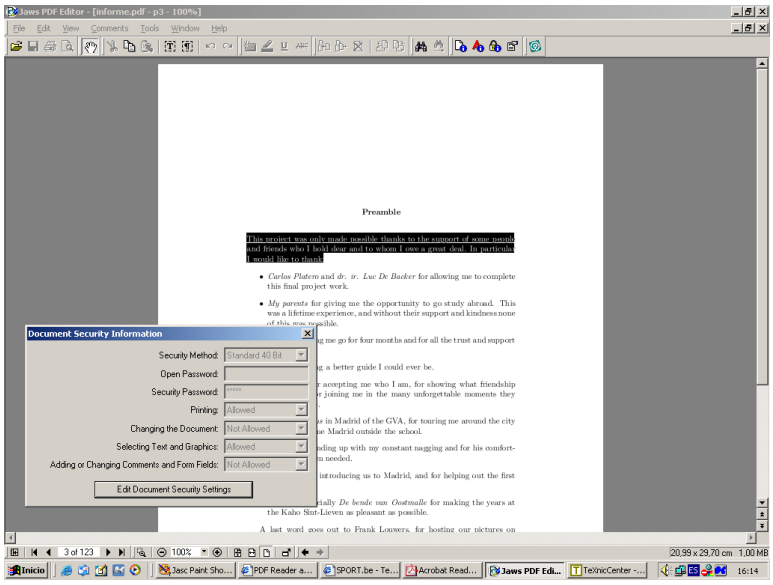


Figure 5.14: Jaws PDF Editor

Chapter 6

Conclusion

6.1 Achievement of goals

The following goals were achieved during this project:

- Conversion of algorithms to stand-alone application in Windows environment using Matlab Compiler
- Investigation difficulties Matlab Compiler
- Comprehensive reference/manual on Matlab Compiler
- Integration Matlab Compiler/Microsoft Visual Studio
- Comparison runtime environments
- Introduction to Vigra Image Library
- Evaluation of Vigra Image Library
- Development of graphical user interface for automatic processing

Thus we can say the main objective is succeeded. Since the main objective was to convert the algorithms to c/c++ code, we could say this part is succeeded. Yet, the generated code from the Compiler is not platform independent, so the same procedure is necessary on any other platform too. Yet the Compiler-related problems should already be solved, since these problems are independent of the operating system. The solution to these problems has been described thoroughly throughout the book, so they should impose nor difficulties, nor time-consumption.

Also a comparison between the two runtime environments, being the Matlab Environment and the Windows Stand-Alone environment was given, making clear the advantages and disadvantages both carry.

Secondly, an introduction and a thorough installation procedure to the Vigra Imaging Library was given. This offers a solid base for any further

development in the future, if necessary. The development using this library was stopped during this project, because of reasons that are described in the corresponding section.

Finally, the project was ended by developing a graphical user interface using the platform independent language of Java. This graphical user interface was developed to facilitate the usage of the converted algorithm, as well as to provide an easy way to process larger quantities of images.

6.2 Future possibilities

This project is the first in trying to convert the algorithms written in Matlab code to stand-alone, platform independent applications. In this project mainly the usage of the built-in compiler is tested. This turned out to be a fast, easy and fluent way of converting the algorithms. That is, in the Windows environment, mainly thanks to the availability of a plug-in for the Microsoft Visual C/C++ integrated developing environment. In the other operating system, being Linux, where such a plug-in is non-existing, this conversion using the Compiler is basically the same, yet more complex. This because every step of the process needs to be taken manually. This being the configuring of the Compiler, the configuring of the libraries, path settings, C/C++-compiler configuration, etc.

Also a small side-step to other possibilities of converting the algorithms was made, in casu the Vigra Imaging Libraries. Although this way is a lot more inconvenient, difficult and time-consuming, this might be a path to take in the future. Also the possibilities of other image processing libraries might be tested in the future, since they are widely spread, but, due to the complexity of the C/C++-programming language, may turn out to be equally difficult.

Also the user interface might be ported to other languages. If both the algorithms and the GUI would be ported to the C/C++-language, a speed increase would be possible, since both could then be more easily integrated. Also the porting to different platforms would be a more fluent task.

Thus, concluding, the following improvements or extensions could be made in the future:

- Converting algorithms using Compiler in Linux
- Porting algorithms to C/C++ using external libraries
- Porting graphical user interface to C/C++

Appendix A

Example Algorithm

A.1 Matlab-original algorithm

```
function [ContCell,CodErr] = ContCellAzulSinV1(NomFich_n,NomFich_m)

%Inicializar codigo de error
CodErr = 0;

%Leer fichero
ImgEnt = imread(NomFich_n);
ImgMar = imread(NomFich_m);

%Detectar Bordes


ImgBorde=edge(ImgEnt(:,:,2),'canny',.4,2); %Emplea el verde

%Ampliar el borde
se= strel('square',5);
ImgBorde = imdilate(ImgBorde,se);


%
%Obtener nucleos
%
%ImgObj = logical((ImgEnt(:,:,2)>100)-ImgBorde);
%Determinar Umbral nucleos
Hist = imhist(ImgEnt(:,:,2));
Umbral = BIN_Iterativo(Hist);
%Determinar nucleos
ImgObj = (ImgEnt(:,:,2)>Umbral) & (~ImgBorde);
```

```
se = strel('disk',10);
ImgObj = imopen(ImgObj,se);
%Contabilidad
ImgEtiq=bwlabel(ImgObj);
ContCell = max(max(ImgEtiq));

%
%Obtener nucleos seleccionados
%
%Determinar Umbral etiquetados
ImgResta = imsubtract(ImgMar(:,:,2),ImgMar(:,:,1));
Hist = imhist(ImgResta);
Umbral = BIN_Momentos(Hist,size(ImgMar,1)*size(ImgMar,2));
ImgEtiqM = (ImgEtiq .* (ImgResta>Umbral));
%ImgEtiqM = imopen(ImgEtiqM,se);
stat = imfeature (ImgEtiqM,'Area');
ContCellMar = 0;
ObjBorrar = [];
for i=1:size(stat,1)
    if(stat(i).Area > 150)
        ContCellMar = ContCellMar+1;
    else
        ObjBorrar = [ObjBorrar;i];
    end
end
ContCell = [ContCell,ContCellMar];


%Eliminar Objetos no etiquetados
%for i=1:size(ObjBorrar,1)
%    ImgEtiq( ImgEtiq == ObjBorrar(i) ) = 0;
%end


%Resultado
%ImgEnt = [ImgEnt ImagResMarcado(ImgEnt,ImgObj)];
%ImgSal = [ImgMar ImagResMarcado(ImgMar,ImgEtiq>0)];
```

A.2 Matlab-algorithm prepared for conversion

```
function ContCellAzulSinV1()
%As can be seen, no inputs nor outputs
```

```
%Inicializar codigo de error
CodErr = 0;

%initialize timer
%(in order to be able to view execution time)
tic

%Read image filenames (replaces input parametres)
fid=fopen('files.txt','r');
FileNameInBlue=fgetl(fid);
FileNameInSel=fgetl(fid);
FileNameOutBlue=fgetl(fid);
FileNameOutSel=fgetl(fid);
fclose(fid);

%Leer fichero
%Read images
ImgInBlue=imread(FileNameInBlue);
ImgInSel=imread(FileNameInSel);

%Leer fichero (Replaced by lines above)
%ImgEnt = imread(NomFich_n);
%ImgMar = imread(NomFich_m);

%Detectar Bordes


ImgBorde=edge(ImgEnt(:,:,2),'canny',.4,2); %Emplea el verde
%this causes runtime error
%see notes on edge in book for solution

%Ampliar el borde
se= strel('square',5);
%causes runtime error, see notes on strel
ImgBorde = imdilate(ImgBorde,se);


%
%Obtener nucleos
%
%ImgObj = logical((ImgEnt(:,:,2)>100)-ImgBorde);
%Determinar Umbral nucleos
Hist = imhist(ImgEnt(:,:,2));
Umbral = BIN_Iterativo(Hist);
```

```
%Determinar nucleos
ImgObj = (ImgEnt(:,:,2)>Umbral) & (~ImgBorde);
se = strel('disk',10);
ImgObj = imopen(ImgObj,se);
%Contabilidad
ImgEtiq=bwlabel(ImgObj);
ContCell = max(max(ImgEtiq));

%
%Obtener nucleos seleccionados
%
%Determinar Umbral etiquetados
ImgResta = imsubtract(ImgMar(:,:,2),ImgMar(:,:,1));
%imsubract works only in C, not C++
Hist = imhist(ImgResta);
Umbral = BIN_Momentos(Hist,size(ImgMar,1)*size(ImgMar,2));
ImgEtiqM = (ImgEtiq .* (ImgResta>Umbral));
%ImgEtiqM = imopen(ImgEtiqM,se);
%stat = imfeature (ImgEtiqM,'Area');
%imfeature does not work with Compiler
%use regionprops instead
stat = regionprops(ImgEtiqM,'Area');
ContCellMar = 0;
ObjBorrar = [];
for i=1:size(stat,1)
    if(stat(i).Area > 150)
        ContCellMar = ContCellMar+1;
    else
        ObjBorrar = [ObjBorrar;i];
    end
end
ContCell = [ContCell,ContCellMar];


%Eliminar Objetos no etiquetados
%for i=1:size(ObjBorrar,1)
%    ImgEtiq( ImgEtiq == ObjBorrar(i) ) = 0;
%end


%Resultado (replaced by procedure below)
%ImgEnt = [ImgEnt ImagResMarcado(ImgEnt,ImgObj)];
%ImgSal = [ImgMar ImagResMarcado(ImgMar,ImgEtiq>0)];
```

```
%Visualize results
ImgOutBlue = ImagResMarcado(ImgInBlue,ImgObj);
ImgOutSel = ImagResMarcado(ImgInSel,ImgEtiq>0);


%Write results
imwrite(ImgOutBlue,FileNameOutBlue);
imwrite(ImgOutSel,FileNameOutSel);

%Write countings to file
fid=fopen('results.txt','w');
fprintf(fid,'%d\n',CountBlue);
fprintf(fid,'%d\n',CountSel);
fclose(fid);

%end timer
toc
```

Appendix B

LaTeX Template

```
\documentclass[11pt,a4paper]{report}

\usepackage{url}
%\usepackage[T1]{fontenc}
\usepackage[latin1]{inputenc}

\usepackage{amssymb}
\usepackage{longtable}
\usepackage[section]{placeins}

\usepackage{float}

\usepackage{makeidx}
\makeindex

\usepackage[pdftex]{graphicx}

\usepackage{fancyhdr}
\pagestyle{fancy}
\addtolength{\headheight}{1.6pt}
\lhead{}
\chead{}
\rhead{\rightmark}
\lfoot{GVA-ELAI-UPM\circledR PFC0059-2001}
\cfoot{}
\rfoot{\thepage}
\renewcommand{\headrulewidth}{0.4pt}
\renewcommand{\footrulewidth}{0.4pt}

%\usepackage{hyperref}
```

```
\newenvironment{preamble}
{
\titlepage
\null
\vfil
\begin{center}
\textbf{Preamble}\\
\vspace{10pt}
\end{center}
}
{
\par\vfil\null\endtitlepage
}

\usepackage{listings}
\usepackage{color}


\begin{document}
\title{Title here}
\author{Name here}
\date {\today}
\maketitle

\begin{abstract} \index{Abstract}

\end{abstract}


\begin{preamble} \index{Preamble}

\end{preamble}

\tableofcontents
\addcontentsline{toc}{chapter}{Abstract}
\addcontentsline{toc}{chapter}{Preamble}
\newpage
\chapter{here chapters}
 \section{Section}
\input{inputfile.tex}
\appendix
\chapter{here appendices}
```

```
\listoffigures
\listoftables

\begin{thebibliography}
here bibliography
\end{thebibliography}

\end{document}
```

Appendix C

InspectImage Header

C.0.1 Include-listing

C.0.1.1 Old code

```
#ifndef VIGRA_INSPECTIMAGE_HXX
#define VIGRA_INSPECTIMAGE_HXX

#include <vector>
#include <algorithm>
#include "vigra/utilities.hxx"
#include "vigra/numerictraits.hxx"
#include "vigra/iteratortraits.hxx"
#include "vigra/rgbvalue.hxx"
```

C.0.1.2 New code

```
#ifndef VIGRA_INSPECTIMAGE_HXX
#define VIGRA_INSPECTIMAGE_HXX

#include <vector>
#include <algorithm>
#include "vigra/utilities.hxx"
#include "vigra/numerictraits.hxx"
#include "vigra/iteratortraits.hxx"
#include "vigra/rgbvalue.hxx"
#include "vigra/minmax.h"
```

C.0.2 Min/Max-references

C.0.2.1 Old code

```
void operator()(argument_type const & coord)
    {
        if(!valid)
        {
            upperLeft = coord;
            lowerRight = coord + Diff2D(1,1);
            valid = true;
        }
        else
        {
            upperLeft.x = std::min(upperLeft.x, coord.x);
            upperLeft.y = std::min(upperLeft.y, coord.y);
            lowerRight.x = std::max(lowerRight.x, coord.x + 1);
            lowerRight.y = std::max(lowerRight.y, coord.y + 1);
        }
    }

        /** update rectangle by merging it with another rectangle
        */
    void operator()(FindBoundingRectangle const & otherRegion)
    {
```

```
        if(!valid)
        {
            upperLeft = otherRegion.upperLeft;
            lowerRight = otherRegion.lowerRight;
            valid = otherRegion.valid;
        }
        else if(otherRegion.valid)
        {
            upperLeft.x = std::min(upperLeft.x, otherRegion.upperLeft.x);
            upperLeft.y = std::min(upperLeft.y, otherRegion.upperLeft.y);
            lowerRight.x = std::max(lowerRight.x, otherRegion.lowerRight.x);
            lowerRight.y = std::max(lowerRight.y, otherRegion.lowerRight.y);
        }
    }
```

C.0.2.2 New code

```
void operator()(argument_type const & coord)
    {
        if(!valid)
        {
            upperLeft = coord;
            lowerRight = coord + Diff2D(1,1);
            valid = true;
        }
        else
        {
            upperLeft.x = min(upperLeft.x, coord.x);
            upperLeft.y = min(upperLeft.y, coord.y);
            lowerRight.x = max(lowerRight.x, coord.x + 1);
            lowerRight.y = max(lowerRight.y, coord.y + 1);
        }
    }

        /** update rectangle by merging it with another rectangle
        */
    void operator()(FindBoundingRectangle const & otherRegion)
    {
        if(!valid)
        {
            upperLeft = otherRegion.upperLeft;
            lowerRight = otherRegion.lowerRight;
            valid = otherRegion.valid;
        }
        else if(otherRegion.valid)
        {
            upperLeft.x = min(upperLeft.x, otherRegion.upperLeft.x);
            upperLeft.y = min(upperLeft.y, otherRegion.upperLeft.y);
            lowerRight.x = max(lowerRight.x, otherRegion.lowerRight.x);
            lowerRight.y = max(lowerRight.y, otherRegion.lowerRight.y);
        }
    }
```

Appendix D

Vigra License

VIGRA is subject to a license which is modeled after the Perl Artistic License and thus more liberal than the GPL.

D.0.3 The VIGRA Artistic License

(Modeled after the Perl Artistic License)

D.0.3.1 Preamble

The intent of this document is to state the conditions under which VIGRA may be copied, such that the author maintains some semblance of artistic control over the development of the library, while giving the users of the library the right to use and distribute VIGRA in a more-or-less customary fashion, plus the right to make reasonable modifications.

D.0.3.2 Definitions

"Copyright Holder" of the VIGRA library is Ullrich Koethe, Cognitive Systems Group, University of Hamburg, Germany.

"Library" refers to the collection of files distributed by the Copyright Holder under the name "VIGRA" (including this LICENSE file and all accompanying documentation), and derivatives of that collection of files created through textual modification.

"Standard Version" refers to the Library if it has not been modified, or has been modified in accordance with the wishes of the Copyright Holder as specified below.

"You" is you, if you're thinking about using, copying, modifying or distributing this Library.

"Freely Available" means that no fee is charged for the item. It also means that recipients of the item may redistribute it under the same conditions they received it.

"Reasonable copying fee" is whatever you can justify on the basis of media cost, duplication charges, time of people involved, and so on. (You will not be required to justify it to the Copyright Holder, but only to the computing community at large as a market that must bear the fee.)

D.0.3.3 License terms

1. You may make and give away verbatim copies of the Standard Version of this Library without restriction, provided that you duplicate all of the original copyright notices, this license, and associated disclaimers.

2. The Standard Version of the Library may be distributed as part of a collection of software, provided no more than a reasonable copying fee is charged for the software collection.

3. You may apply bug fixes and portability fixes derived from the Public Domain or from the Copyright Holder. A Library modified in such a way shall still be considered the Standard Version.

4. You may otherwise modify your copy of this Library in any way, provided that you insert a prominent notice in each changed file stating how and when you changed that file, and provided that you do at least ONE of the following:

   (a) place your modifications in the Public Domain or otherwise make them Freely Available, for example by allowing the Copyright Holder to include your modifications in the Standard Version of the Library.

   (b) use the modified Library only within your corporation or organization.

   (c) make other distribution arrangements with the Copyright Holder.

5. You may distribute programs which use this Library in object code or executable form without restriction.

6. Any object code generated as a result of using this Library does not fall under the copyright of this Library, but belongs to whomever generated it, and may be sold commercially.

7. The name of the Copyright Holder or the Library may not be used to endorse or promote products derived from this software without specific prior written permission.

8. THIS LIBRARY IS PROVIDED AS IS AND WITHOUT ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

   IN NO EVENT SHALL THE COPYRIGHT HOLDER BE LIABLE FOR ANY SPECIAL, INCIDENTAL, INDIRECT OR CONSEQUENTIAL DAMAGES OF ANY KIND OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER OR NOT ADVISED OF THE POSSIBILITY OF SUCH DAMAGES, OR ON ANY THEORY OF LIABILITY ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS LIBRARY.

Appendix E

IJG JPEG License

The authors make NO WARRANTY or representation, either express or implied, with respect to this software, its quality, accuracy, merchantability, or fitness for a particular purpose. This software is provided "AS IS", and you, its user, assume the entire risk as to its quality and accuracy.

This software is copyright (C) 1991-1998, Thomas G. Lane.
All Rights Reserved except as specified below.

Permission is hereby granted to use, copy, modify, and distribute this software (or portions thereof) for any purpose, without fee, subject to these conditions:

1. If any part of the source code for this software is distributed, then this README file must be included, with this copyright and no-warranty notice unaltered; and any additions, deletions, or changes to the original files must be clearly indicated in accompanying documentation.

2. If only executable code is distributed, then the accompanying documentation must state that "this software is based in part on the work of the Independent JPEG Group".

3. Permission for use of this software is granted only if the user accepts full responsibility for any undesirable consequences; the authors accept NO LIABILITY for damages of any kind.

These conditions apply to any software derived from or based on the IJG code, not just to the unmodified library. If you use our work, you ought to acknowledge us.

Permission is NOT granted for the use of any IJG author's name or company name in advertising or publicity relating to this software or products derived from it. This software may be referred to only as "the Independent JPEG Group's software".

We specifically permit and encourage the use of this software as the basis of commercial products, provided that all warranty or liability claims are assumed by the product vendor.

ansi2knr.c is included in this distribution by permission of L. Peter Deutsch, sole proprietor of its copyright holder, Aladdin Enterprises of Menlo Park, CA.
ansi2knr.c is NOT covered by the above copyright and conditions, but instead by the usual distribution terms of the Free Software Foundation; principally, that you must include source code if you redistribute it. (See the file ansi2knr.c for full details.) However, since ansi2knr.c is not needed as part of any program generated from the IJG code, this does not limit you more than the foregoing paragraphs do.

The Unix configuration script "configure" was produced with GNU Autoconf. It is copyright by the Free Software Foundation but is freely distributable. The same holds for its supporting scripts (config.guess, config.sub, ltconfig, ltmain.sh). Another support script, install-sh, is copyright by M.I.T. but is also freely distributable.

It appears that the arithmetic coding option of the JPEG spec is covered by patents owned by IBM, AT&T, and Mitsubishi. Hence arithmetic coding cannot legally be used without obtaining one or more licenses. For this reason, support for arithmetic coding has been removed from the free JPEG software. (Since arithmetic coding provides only a marginal gain over the unpatented Huffman mode, it is unlikely that very many implementations will support it.)

So far as we are aware, there are no patent restrictions on the remaining code.

The IJG distribution formerly included code to read and write GIF files. To avoid entanglement with the Unisys LZW patent, GIF reading support has been removed altogether, and the GIF writer has been simplified to produce "uncompressed GIFs". This technique does not use the LZW algorithm; the resulting GIF files are larger than usual, but are readable by all standard GIF decoders.

We are required to state that "The Graphics Interchange Format(c) is the Copyright property of CompuServe Incorporated. GIF(sm) is a Service Mark property ofCompuServe Incorporated."

Appendix F

LIBTIFF Copyright

Appendix G

ZLIB License

zlib.h – interface of the 'zlib' general purpose compression library version 1.1.4, March 11th, 2002

Jean-loup Gailly jloup@gzip.org Mark Adler madler@alumni.caltech.edu

Appendix H

GNU License

H.1 GNU GENERAL PUBLIC LICENSE

Version 2, June 1991

H.1.1 Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software–to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

H.1.2 TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law:that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

H.1.3 Warranty

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

H.1.4 End of Terms and Conditions

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.

List of Figures

List of Tables

Bibliography

[Matlab] Mathworks' Matlab's main webpage
http://www.mathworks.com

[Matlab Compiler] Mathworks' Matlab Compiler 3.0 webpage
http://www.mathworks.com/products/compiler

[Matlab Compiler] Mathworks' Matlab Compiler 3.0 documentation
http://www.mathworks.com/access/helpdesk/help/pdf_doc/compiler/compiler3.pdf

[Vigra Homepage] Vigra Computer Vision Library webpage
http://kogs-www.informatik.uni-hamburg.de/~koethe/vigra/

[Microsoft Visual Studio] Microsoft Visual Studio webpage
http://msdn.microsoft.com/vstudio/

[Microsoft Visual Compiler C++ Problems] "Known problems in using the Microsoft Visual C++ compiler"
http://www.acceleratedcpp.com/details/msbugs.html

[TIFF Libraries] http://www.libtiff.org/

[JPEG Libraries] http://www.ijg.org/

[ZLIB Libraries] http://www.gzip.org/zlib/

[FFTW Libraries] http://www.fftw.org/

[GNU Win32] http://gnuwin32.sourceforge.net/

[Java] Sun's Java webpage
http://java.sun.com/

[Java's Runtime Executable] "When Runtime.exec() won't"
http://www.javaworld.com/javaworld/jw-12-2000/jw-1229-traps.html

[Java 2 in 21 dagen] by Laura Lemay & Rogers Cadenhead - SAMS ISBN 90-430-0395-6

[The Not So Short Introduction to LaTeX 2_ε] by Tobias Oetiker, Hubert Partl , Irene Hyna and Elisabeth Schlegl