



# *Getting started with ViSP 2.4.0*

Fabien Spindler, Anthony Saunier

Lagadic

IRISA / INRIA Rennes

<http://www.irisa.fr/lagadic>

February 13, 2007



# *Outline*

1. What is ViSP
2. Improvements in ViSP 2.4.0
3. Prerequisites
4. Getting ViSP
5. ViSP build chain
6. ViSP configuration
7. ViSP building
8. ViSP testing
9. ViSP packaging
10. ViSP as a third party library
11. ViSP road map



# 1. *What is ViSP*

- <http://www.irisa.fr/lagadic/visp>
- Multi platform library dedicated to visual servoing applications
- Open source software: QPL license from TrollTech
- Public project on Inria's forge: <http://gforge.inria.fr/projects/visp>
- Capabilities:
  - Image framegrabbing, display
  - Image processing
    - Dot, moving edges tracking
    - Homographies
  - Pose handling
  - Visual servoing
    - Features: point, line, ellipse, circle, sphere, cylinder
    - Eye in hand, eye to hand control laws
  - Robot control
  - Mathematical skills
  - Simulation



## *2. Improvements in ViSP-2.4.0*

- ❑ Windows compatible
- ❑ Better coding standard respect
- ❑ New functionalities (vp1tfig8Grabber, vp1394TwoGrabber)
- ❑ Memory leaks free
- ❑ Web and doxygen documentation
- ❑ Tests and examples
- ❑ Testing and daily builds with dashboards
- ❑ Packaging



### 3. Prerequisites

- CMake (<http://www.cmake.org>) for ViSP configuration
- Depending on your attendees, third party libraries

Framegrabber	Linux	OSX	Win	Class
ICcomp	intern			vpIcCompGrabber
Video 4 Linux 2				vpV4I2Grabber
Itifg-8.x				vpItifg8Grabber
libdc1394-1.x				vp1394Grabber
libdc1394-2.x				vp1394TwoGrabber
CFOX				vpOSXcfoxGrabber
DirectShow				vpDirectShowGrabber



Supported by ViSP



Lagadic internal usage



Potentially supported (not tested yet)



Display	Linux	OSX	Win	Class
X				vpDisplayX
GTK				vpDisplayGTK
GDI				vpDisplayGDI
Direct3D				vpDisplayD3D

Robots	Linux	OSX	Win	Class
Afma4	intern			vpRobotAfma4
Afma6	intern			vpRobotAfma6
Ptu-46 head	intern			vpRobotPtu46
Biclops head				vpRobotBiclops

Simulator	Linux	OSX	Win	Class
Coin3D, SoQt, Qt				vpSimulator

Matrix computation	Linux	OSX	Win	Class
Gnu Scientific Library				vpMatrix



## 4. Getting ViSP

- <http://www.irisa.fr/lagadic/visp>
- Latest downloadable release : [ViSP-2.4.0.tar.gz](#) (February 6, 2007)
- Latest work in progress version
  - For registered members on the forge (recommended for Lagadic members)

```
setenv CVS_RSH ssh (or export CVS_RSH=ssh)  
cvs -d :ext:username@scm.gforge.inria.fr:/cvsroot/visp checkout ViSP
```

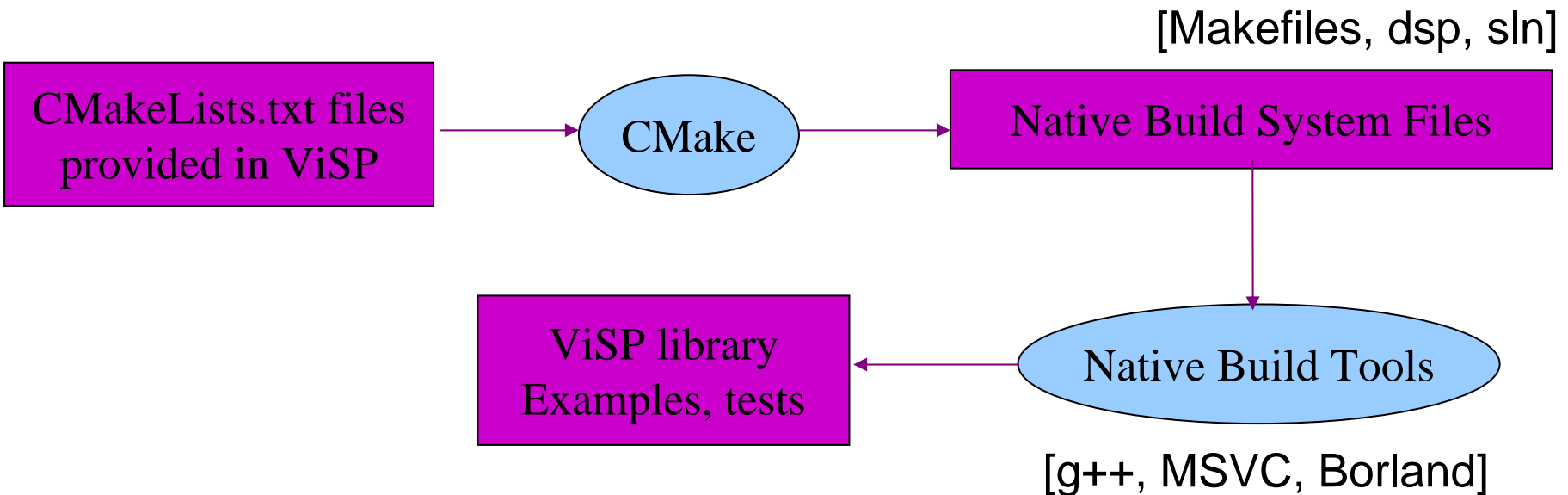
- By anonymous access on the forge

```
cvs -d :pserver:anonymous@scm.gforge.inria.fr:/cvsroot/visp login  
(just press enter for the passwd)  
cvs -d :pserver:anonymous@scm.gforge.inria.fr:/cvsroot/visp checkout ViSP
```



## 5. ViSP build chain

- First, ViSP configuration with CMake
- Then, ViSP compilation with a native build tool: g++, MSVC, ...
- If ViSP is to use as a third party library in your development, don't forget to install ViSP headers and library

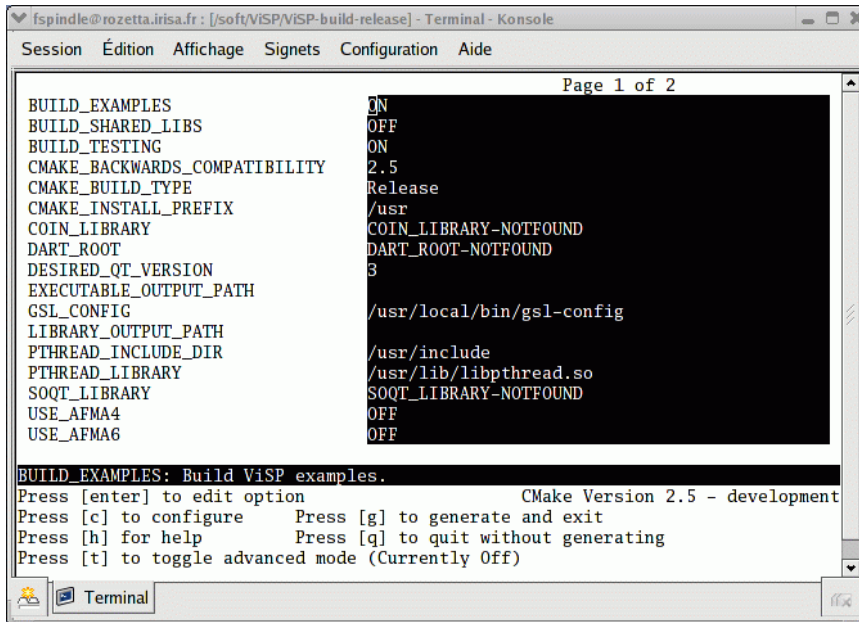




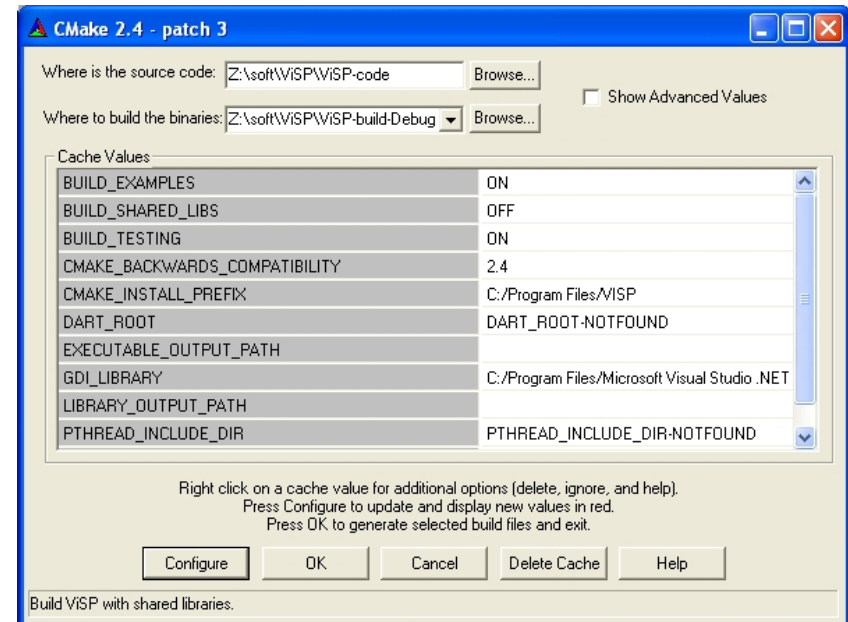
# 6. ViSP configuration

- By CMake GUI usage
- Edit cache entries to configure the build
- Use configure button after a change
- Use OK (generate) button when finished

## ccmake (unix)



## CMake (Windows)





## *Important cache entries*

- Build type:

```
CMAKE_BUILD_TYPE=[Debug, Release]
```

- Where to install ViSP headers and library:

```
CMAKE_INSTALL_PREFIX=[/usr/local, C:\Program Files\ViSP]
```

- How to see compilation command (only when Makefiles are in use)

```
CMAKE_VERBOSE_MAKEFILE=ON
```

- How to build only ViSP library (not the examples and tests)

```
BUILD_EXAMPLES=OFF
```

```
BUILD_TESTING=OFF
```

- How to build ViSP library as shared (not static)

```
BUILD_SHARED_LIBS=ON
```



# *In-source or out-of-source build*

## □ *In-source build*

- Code, library and binaries in the same directory
- Suggested for end-users

```
cd ViSP-code  
ccmake .  
make
```

— ViSP-code – [code, binaries]

## □ *Out-of-source build*

- Code separated from library and binaries
- Possible to handle several versions
- Suggested for developers

```
mkdir ViSP-build  
cd ViSP-build-debug  
ccmake ../ViSP-code  
make
```

— ViSP-code – [code]  
— ViSP-build-debug – [binaries]  
— ViSP-build-g++4.x – [binaries]  
— ViSP-build-g++3.x – [binaries]



## 7. ViSP build

- Use of native build system files produced by CMake: Makefile, Microsoft Visual Studio project files .dsl, .sln, ...
- ViSP library building and optionally examples and tests

```
make
```

```
MSVC projet "ALL_BUILD"
```

- ViSP headers and library installation (needed if ViSP has to be used as a third party project)

```
make install
```

```
MSVC projet "INSTALL"
```

- Online documentation creation with doxygen

```
make html-doc
```

```
MSVC projet "html-doc"
```



## 8. *ViSP testing*

- ViSP supports CTest, a testing tool
- Perform several operations on the source code
  - Configure, compile library, tests and examples
  - Execute tests and examples binaries
  - Advanced tests such as coverage and memory checking
- Set of testing data and images available: [ViSP-images-1.1.0.tar.gz](http://www.irisa.fr/vi/sp/images-1.1.0.tar.gz)
- Daily builds on several platforms (crontab unix, task scheduler on windows)
  - Nightly builds
  - Experimental builds
  - Results are submitted to a Dart server  
<http://dart.irisa.fr/ViSP2/Dashboard>



# Testing usage

## □ Local testing

- To execute all the tests

```
make test
```

```
MSVC projet "RUN_TESTS"
```

- To execute a specific test

```
ctest -I 2,2
```

## □ Submission of the testing results to the Dart server

```
ctest -D [Experimental|Nightly]
```

```
MSVC projet "Experimental" and "Nightly"
```

## □ Test log are written in **Testing/Temporary**

```
LastTest.log
```

```
LastTestsFailed.log
```

```
Running tests...
Start processing tests
Test project /local/soft/ViSP-build
 1/ 2 Testing testDisplayX   Passed
 2/ 2 Testing testDisplayGTK Passed
100% tests passed, 0 tests failed out of 2
```



## 9. ViSP packaging

- ViSP supports CPack, a packaging tool

Package type	Packaging tool	Linux	OSX	Win
STGZ	Self extracting Tar GZip			
TBZ2	Tar BZip2			
TGZ	Tar GZip			
TZ	Tar Compress			
ZIP	ZIP file format			
PackageMaker	OSX Package Maker			
NSIS	Nullsoft Scriptable Install System			



# Packaging usage

- Packaging tools available :
- Configuration files created by:
  - CPackConfig.cmake
  - CPackSourceConfig.cmake
- Usage

```
cpack --help
```

```
ccmake <source dir>
```

```
make package
```



```
ViSP-2.4.0-Linux.sh
```

```
cpack -G ZIP
```



```
ViSP-2.4.0-Darwin.zip
```

```
MSVC projet "PACKAGE"
```



```
ViSP-2.4.0-win32.exe
```

```
cpack -G TGZ -config CPackSourceConfig.cmake
```

```
make package_source
```



```
ViSP-2.4.0-Source.tar.gz
```

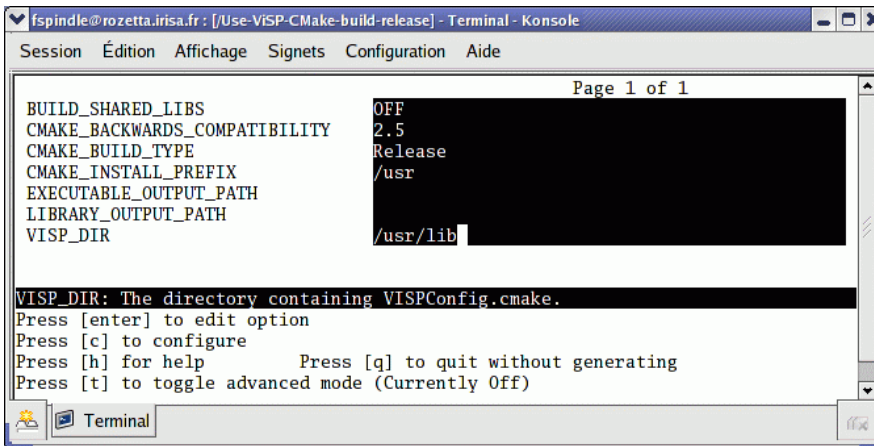


## 10. ViSP as a third party project 1/3

- Use ViSP within another CMake project
- Downloadable example on the forge: [Use-ViSP-CMake-1.0.0.tar.gz](#)
  - Include the following into your top level CMakeList.txt:

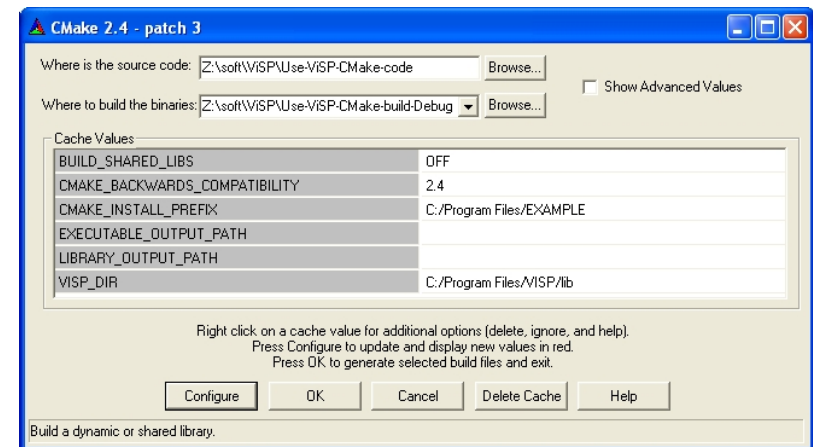
```
FIND_PACKAGE(VISP REQUIRED)
IF(VISP_FOUND)
  INCLUDE(${VISP_USE_FILE})
ENDIF(VISP_FOUND)
```

- ViSP options needed for compilation and linking are automatically added
- Set VISP\_DIR in CMake GUI to the directory where ViSP library is installed



```
Build Shared Libraries: OFF
CMake Backwards Compatibility: 2.5
CMake Build Type: Release
CMake Install Prefix: /usr
Executable Output Path:
Library Output Path:
ViSP Directory: /usr/lib

ViSP_DIR: The directory containing VISPConfig.cmake.
Press [enter] to edit option
Press [c] to configure
Press [h] for help          Press [q] to quit without generating
Press [t] to toggle advanced mode (Currently Off)
```





## *ViSP as a third party project 2/3*

- Use ViSP with configure (autotools)
- Downloadable example on the forge: [Use-ViSP-configure-1.0.0.tar.gz](#)
  - Use “macro/have\_visp.m4” provided with ViSP to check if “visp-config” shell script exists
  - Include the following in your configure.ac:

```
AC_HAVE_VISP_IFELSE(have_visp=yes,have_visp=no)
if test "x$have_visp" = "xyes"; then
  CXXFLAGS="$CXXFLAGS $ac_visp_cflags"
  LIBS="$LIBS $ac_visp_libs"
fi
```

- Options for compilation and linking with ViSP are added in \$(CXXFLAGS) and \$(LIBS) variables
- Classical usage in Makefile.in

```
example: example.cpp
  g++ $(CXXFLAGS) -o example example.cpp $(LIBS)
```



## *ViSP as a third party project 3/3*

### □ Use ViSP with classical Makefile

- Use “<CMAKE\_INSTALL\_PREFIX>/bin/visp-config” shell script

```
visp-config --prefix/--cflags/--libs/--version
```

```
picatchou[15:10]%. /visp-config --libs
-Wl,-rpath,/usr/local/lib -L/usr/local/lib -lvisp-2 -L/usr/lib -L/usr/l
ib/qt-3.1/lib -L/usr/X11R6/lib -lCoin -lqt -lSoQt -lSM -lICE -lSM -lICE -
lX11 -lXext -lX11 -lXext -lm -lpthread -L/usr/local/lib -lgsl -lgslcblas
-lm -lgtk-x11-2.0 -lgdk-x11-2.0 -glib-2.0 -gobject-2.0 -gmodule-2.0 -l
gthread-2.0 -lraw1394 -ldc1394_control
```

- Example of hand made Makefile

```
CXXFLAGS = `/usr/bin/visp-config --cflags`
LIBS      = `/usr/bin/visp-config --libs`
```

```
example: example.cpp
```

```
g++ $(CXXFLAGS) -o example example.cpp $(LIBS)
```



## *11. ViSP road map*

- User documentation, tutorial
- Camera settings positioning with DirectShow
- Stereo grabbing under Windows
- Exceptions
- Simulation capabilities
- opencv support
- MinGW compatible
- New functionalities