MSeg Documentation

Release 1.2.0

Christian R. G. Dreher

Apr 04, 2018

Guides

MSeg is a framework for the ArmarX robot development environment, built to run and evaluate motion segmentation algorithms.

It provides the core functionality to easily integrate motion segmentation algorithms written in various programming languages. To integrate a motion segmentation algorithm, a supplied *Programming Language Interface* (PLI) can be used.

For more information about the structure of MSeg, please refer to the *architectural overview*:

CHAPTER 1

Quickstart

This guide will show how to install MSeg on a 64 bit Ubuntu machine. This includes the MSeg Core Module, all PLIs, and all available tools to use with MSeg.

Note: **THIS IS A WORK IN PROGRESS DOCUMENTATION - THIS WAY OF INSTALLING MSEG IS NOT YET SUPPORTED**. Please refer to the installation from source

Note: It is *strongly recommended* to use MSeg with Ubuntu 14.04 x64, as it eases the installation process drastically. If this should be a problem, consider using a virtual machine like VirtualBox and install Ubuntu 14.04 there.

1.1 Preconditions

- Freshly installed Ubuntu 14.04.5 LTS x64 trusty [64-bit PC (AMD64) desktop image]
- `sudo` privileges
- Approx. hard disk size required: 1 GiB (Make sure to have enough space if you want to install additional software like IDEs or MATLAB)

1.2 Installation

Open a terminal (`Ctrl` + `Shift` + `T`) and add the H2T package server and the MSeg package server. The H2T package server is needed as it bundles required dependencies like ArmarX.

```
# Add H2T signing key and package server
$ curl https://packages.humanoids.kit.edu/h2t-key.pub | sudo apt-key add -
$ echo -e "deb http://packages.humanoids.kit.edu/trusty/main trusty main\ndeb http://
↪packages.humanoids.kit.edu/trusty/testing trusty testing" | sudo tee /etc/apt/
↪sources.list.d/armarx.list
```

```
# Add MSeg signing key and package server
$ sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-keys 04FF7B61
$ sudo add-apt-repository "deb https://repo.christian-dreher.name/ trusty main"
# Update lists
$ sudo apt update
```

Now MSeg can be installed. This command will download roughly 300 MiB, resulting in about 1 GiB after installation.

```
# Download and install MSeg and all its dependencies
$ sudo apt install mseg
```

1.3 Initialisation

After the installation, both ArmarX and MSeg need to be initialised.

```
# Run armarx for the first time. You will be prompted to add a line to
# .bashrc for autocompletion. Accept with y
$ armarx
# Source .bashrc as indicated
$ source ~/.bashrc
# Download the MSeg datasets of labelled whole-body motion recordings
$ msegdata fetch
```

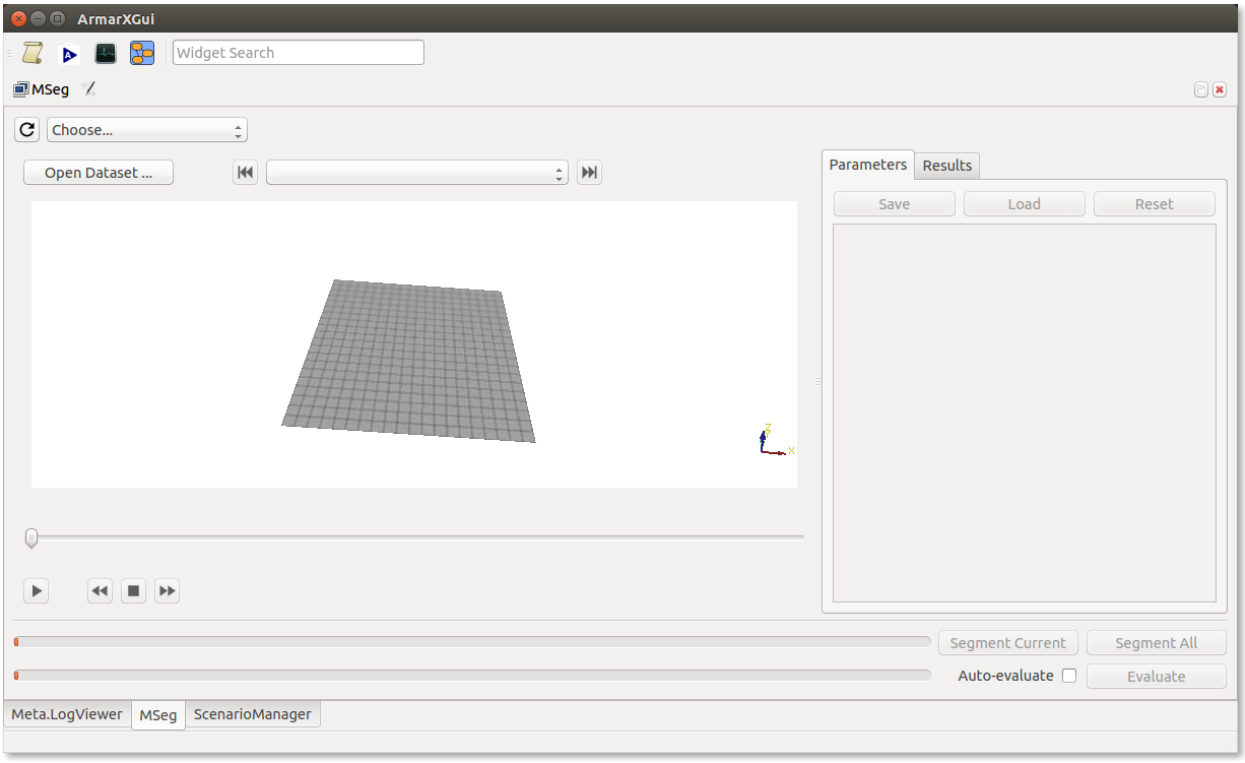
Now we're ready for the first run.

```
# First, we need to start ArmarX
$ armarx start
# Now we need to start the MSeg core module
$ msegcm start
# Eventually, launch the ArmarX GUI
$ armarx gui
```

The GUI will now load and eventualy, a popup should appear.

- You may check the checkbox `Do not show again`.
- Push the `Open empty GUI` button.
- You can now use the MSeg GUI by clicking on `Add Widget | MSeg`.

The MSeg GUI should now show up like this:



1.4 How to Proceed

With MSeg being set up, you can proceed with integrating your own motion segmentation algorithm in *C++*, *Java*, *Python* or *MATLAB*.

CHAPTER 2

Integrate a C++ Algorithm

This guide will show you how to setup your workspace to integrate your own C++ motion segmentation algorithm using the C++ PLI. It is assumed that you either followed the *quickstart guide* or you compiled the *MSeg Core Module* and the *C++ PLI* from source.

2.1 Install the Skeleton Project

We will use the `mseggen` terminal tool to install a skeleton project.

```
# Define the base directory where the project directory should be located
$ export $PROJECT_BASE_DIR=~/projects
# Run the generation tool to create the skeleton.
# If your algorithm requires training, add the --train flag
$ mseggen -b $PROJECT_BASE_DIR --cpp ExampleAlgorithm
```

2.2 Compiling the Algorithm

Your project should now look similar to this (If you changed the algorithm name the paths will be slightly altered):

```
$PROJECT_BASE_DIR/
`--- examplealgorithm
    |--- build/
    |--- source/
    |    |--- examplealgorithm.cpp
    |    |--- examplealgorithm.h
    |    `--- main.cpp
    `--- CMakeLists.txt
```

To compile the skeleton, we change into the build folder and run the corresponding commands like this:

```
# Change into the build folder
$ cd $PROJECT_BASE_DIR/examplealgorithm/build
# Run CMake
$ cmake ..
# Run make
$ make
```

You may get warnings about unused parameters, but eventually, the project should compile.

2.3 Run the Algorithm

To run the algorithm, both ArmarX and the MSeg core module must be running. You can then start the algorithm like this:

```
# Change into the build folder
$ cd $PROJECT_BASE_DIR/examplealgorithm/build
# Run the algorithm
$ ./examplealgorithm
```

You should now see an output like this:

```
user@machine:~/projects/examplealgorithm/build$ ./examplealgorithm
Connecting to MSeg core module...
Connection established
ExampleAlgorithm ready
```

You can stop the algorithm with `Ctrl` + `C` at any time.

2.4 How to Proceed

You can now start implementing your algorithm. You may also want to get more familiar with the *SegmentationAlgorithm API reference*, the *MSeg GUI* and the *msegcm, msegdata, and mseggen terminal tools*.

CHAPTER 3

Integrate a Java Algorithm

This guide will show you how to setup your workspace to integrate your own Java motion segmentation algorithm using the Java PLI. It is assumed that you either followed the *quickstart guide* or you compiled the *MSeg Core Module* and the *Java PLI* from source.

3.1 Install the Skeleton Project

We will use the `mseggen` terminal tool to install a skeleton project.

```
# Define the base directory where the project directory should be located
$ export $PROJECT_BASE_DIR=~/projects
# Run the generation tool to create the skeleton.
# If your algorithm requires training, add the --train flag
$ mseggen -b $PROJECT_BASE_DIR --java ExampleAlgorithm
```

3.2 Compiling the Algorithm

Your project should now look similar to this (If you changed the algorithm name the paths will be slightly altered):

```
$PROJECT_BASE_DIR/
`--- examplealgorithm
    |--- build/
    |--- lib/
    |--- src/
    |    `--- ExampleAlgorithm.java
    `--- build.xml
```

To compile the skeleton, we change into the project root folder and run *ant*:

```
# Change into the build folder
$ cd $PROJECT_BASE_DIR/examplealgorithm
# Run ant
$ ant
```

3.3 Run the Algorithm

To run the algorithm, both ArmarX and the MSeg core module must be running. You can then start the algorithm like this:

```
# Change into the build folder
$ cd $PROJECT_BASE_DIR/examplealgorithm/build
# Run the algorithm
$ java -jar examplealgorithm.jar
```

You should now see an output like this:

```
user@machine:~/projects/examplealgorithm/build$ java -jar examplealgorithm.jar
Connecting to MSeg core module...
Connection established
ExampleAlgorithm ready
```

You can stop the algorithm with `Ctrl` + `C` at any time.

3.4 How to Proceed

You can now start implementing your algorithm. You may also want to get more familiar with the *SegmentationAlgorithm API reference*, the *MSeg GUI* and the *msegcm, msegdata, and mseggen terminal tools*.

CHAPTER 4

Integrate a Python Algorithm

This guide will show you how to setup your workspace to integrate your own Python motion segmentation algorithm using the Python PLI. It is assumed that you either followed the *quickstart guide* or you compiled the *MSeg Core Module* from source and set up the *Python PLI*.

4.1 Install the Skeleton Project

We will use the `mseggen` terminal tool to install a skeleton project.

```
# Define the base directory where the project directory should be located
$ export $PROJECT_BASE_DIR=~/projects
# Run the generation tool to create the skeleton.
# If your algorithm requires training, add the --train flag
$ mseggen -b $PROJECT_BASE_DIR --python ExampleAlgorithm
```

4.2 Run the Algorithm

To run the algorithm, both ArmarX and the MSeg core module must be running. You can then start the algorithm like this:

```
# Change into the project folder
$ cd $PROJECT_BASE_DIR/examplealgorithm
# Run the algorithm
$ python examplealgorithm.py
```

You should now see an output like this:

```
user@machine:~/projects/examplealgorithm$ python examplealgorithm.py
Connecting to MSeg core module...
Connection established
ExampleAlgorithm ready
```

You can stop the algorithm with `Ctrl` + `C` at any time.

4.3 How to Proceed

You can now start implementing your algorithm. You may also want to get more familiar with the *SegmentationAlgorithm API reference*, the *MSeg GUI* and the *msegcm, msegdata, and mseggen terminal tools*.

CHAPTER 5

Integrate a MATLAB Algorithm

This guide will show you how to setup your workspace to integrate your own MATLAB motion segmentation algorithm using the MATLAB PLI. It is assumed that you either followed the *quickstart guide* or you compiled the *MSeg Core Module* from source and set up the *MATLAB PLI*.

5.1 Install the Skeleton Project

We will use the `mseggen` terminal tool to install a skeleton project.

```
# Define the base directory where the project directory should be located
$ export $PROJECT_BASE_DIR=~/projects
# Run the generation tool to create the skeleton.
# If your algorithm requires training, add the --train flag
$ mseggen -b $PROJECT_BASE_DIR --matlab ExampleAlgorithm
```

5.2 Run the Algorithm

To run the algorithm, both ArmarX and the MSeg core module must be running. You can then start the algorithm like this:

```
# Change into the project folder
$ cd $PROJECT_BASE_DIR/examplealgorithm
# Run the algorithm
$ ./main
```

Note: The `main` executable *cannot* be started from within MATLAB.

Running the algorithm from the terminal, you should now see an output like this:

```
user@machine:~/projects/examplealgorithm$ ./main
Initialising MATLAB
Connecting to MSeg core module...
Connection established
ExampleAlgorithm ready
```

You can stop the algorithm with `Ctrl` + `C` at any time.

5.3 How to Proceed

You can now start implementing your algorithm. You may also want to get more familiar with the *SegmentationAlgorithm API reference*, the *MSeg GUI* and the *msegcm, msegdata, and mseggen terminal tools*.

CHAPTER 6

Compile the MSeg Core Module from Source

This guide will show how to compile the MSeg core module from source on a 64 bit Ubuntu machine.

Note: It is *strongly recommended* to use the MSeg core module with Ubuntu 14.04, as it eases the installation process drastically. If this should be a problem, consider using a virtual machine like VirtualBox and install Ubuntu 14.04 there.

6.1 Preconditions

- Freshly installed Ubuntu 14.04.5 LTS x64 "trusty" [64-bit PC (AMD64) desktop image]
- `sudo` privileges
- Approx. hard disk size required: 1 GiB (Make sure to have enough space if you want to install additional software like IDEs or MATLAB)

6.2 Setup Environment Variables and Installation Directories

Open a terminal (`Ctrl` + `Shift` + `T`) and set the needed environment variables and create the installation directory.

```
# Add ARMARX_INSTALL_DIR to .bashrc
$ echo "export ARMARX_INSTALL_DIR=$HOME/armarx" >> ~/.bashrc
# Source .bashrc
$ source ~/.bashrc
# Add MSEG_INSTALL_DIR to .bashrc
$ echo "export MSEG_INSTALL_DIR=$ARMARX_INSTALL_DIR/MSeg" >> ~/.bashrc
# Source .bashrc
$ source ~/.bashrc
# Create ArmarX install directory
$ mkdir -p $ARMARX_INSTALL_DIR
```

6.3 Dependencies

In this step, all needed dependencies will be installed. Where repositories are available, they will be used. Otherwise, they will be compiled from source.

6.3.1 Standard Ubuntu Packages

Install the following packages:

```
$ sudo apt install astyle cmake cmake-qt-gui cppcheck curl doxygen \
freeglut3-dev g++ git gsl-bin ivy lcov libalglib-dev libboost-all-dev \
libcoin80-dev libcv-dev libcvaux-dev libdb5.1-dev libdc1394-22-dev \
libeigen3-dev libgraphviz-dev libgsl0-dev libgstreamer-plugins-base0.10-dev \
libhighgui-dev libjsoncpp-dev libnlopt-dev libopencv-dev libopencv-gpu-dev \
libopencv-photo-dev libopencv-stitching-dev libopencv-superres-dev \
libopencv-ts-dev libopencv-videostab-dev libpcre3-dev libqwt-dev \
libsoqt4-dev libsqlite3-dev libssl-dev libtinyxml-dev libv4l-dev mcpp \
mongodb openjdk-7-jdk python-argcomplete python-docutils python-psutil \
python-setuptools zeroc-ice35
```

6.3.2 Packages from H2T Package Server

Add the H2T key and install additional packages.

```
# Add H2T key and add repositories
$ curl https://packages.humanoids.kit.edu/h2t-key.pub | sudo apt-key add -
$ echo -e "deb http://packages.humanoids.kit.edu/trusty/main trusty main\ndeb http://
↪packages.humanoids.kit.edu/trusty/testing trusty testing" | sudo tee /etc/apt/
↪sources.list.d/armarx.list
# Update lists
$ sudo apt update
# Install additional packages
$ sudo apt install ivt ivtrecognition mmmcore simox
```

6.3.3 Install MMMTools from Source

MMMTools is used for the visualisation of motion data. Unfortunately, it is not available as binary and has to be compiled from source.

```
# Get source code
$ git clone https://gitlab.com/mastermotormap/mmmtools.git $ARMARX_INSTALL_DIR/
↪mmmtools
# Go into the source folder of MMMTools
$ cd $ARMARX_INSTALL_DIR/mmmtools
# Create build directory and go into it
$ mkdir build && cd build
# Run CMake
$ cmake ..
# Compile. Use 'make -j4' to use 4 cores
$ make
```

6.4 Setup ArmarXCore

In this step, ArmarXCore will be compiled from source and setup. We start with fetching the source code and compiling it.

```
# Get source code
$ git clone https://gitlab.com/ArmarX/ArmarXCore $ARMARX_INSTALL_DIR/ArmarXCore
# Go to build directory
$ cd $ARMARX_INSTALL_DIR/ArmarXCore/build
# Run CMake
$ cmake ..
# Compile. Use 'make -j4' to use 4 cores
$ make
```

ArmarXCore should be compiled now. Now add the binary directory to `$PATH` to ease working with it.

```
$ echo "export PATH=\"$ARMARX_INSTALL_DIR/ArmarXCore/build/bin:\$PATH\"" >> ~/.bashrc
$ source ~/.bashrc
```

We also need to install the Python packages.

```
# Go into python scripts directory
$ cd $ARMARX_INSTALL_DIR/ArmarXCore/etc/python
# Run setup
$ python setup.py develop --user
```

With everything being setup, we can now start ArmarX. Config files needed later will be created then.

```
# Run armarx for the first time. You will be prompted to add a line to
# .bashrc for autocompletion. Accept with y
$ armarx
# Source .bashrc as indicated
$ source ~/.bashrc
# Run armarx again since the previous step just modified the .bashrc file
$ armarx start
```

The config file now got created. Run the following command and apply the changes below.

```
# Open config file
$ gedit ~/.armarx/default.cfg
```

Changes:

```
++ Ice.MessageSizeMax=10240
   Ice.Default.Locator=IceGrid/Locator:tcp -p 12454 -h localhost
   IceGrid.Registry.Client.Endpoints=tcp -p 12454

   ArmarX.MongoHost=localhost
   ArmarX.MongoPort=12455

   #Put your custom ArmarX Packages in this list, e.g. so that the gui can find their␣
↪plugins.
-- ArmarX.AdditionalPackages=
++ ArmarX.AdditionalPackages=MSeg
```

Save and close the config file. Finally, restart ArmarX.

```
# Restart ArmarX
$ armarx reset
```

6.5 Setup ArmarXGui

We will now proceed to install the ArmarXGui. Again we start with fetching the source code and compiling it.

```
# Get source code
$ git clone https://gitlab.com/ArmarX/ArmarXGui $ARMARX_INSTALL_DIR/ArmarXGui
# Go to build directory
$ cd $ARMARX_INSTALL_DIR/ArmarXGui/build
# Run CMake
$ cmake ..
# Compile. Use 'make -j4' to use 4 cores
$ make
```

You should now be able start the GUI.

```
# Start ArmarX GUI
$ armarx gui
```

The GUI will now load and eventualy, a popup should appear.

- You may check the checkbox `Do not show again`.
- Push the `Open empty GUI` button.

Close the GUI for now using the × button.

6.6 Setup the MSeg Core Module

In this step the MSeg core module will be compiled and setup.

6.6.1 Compile the MSeg Core Module

Once more, fetch the source code and compile.

```
# Get source code
$ git clone https://gitlab.com/h2t/kit-mseg/core.git $MSEG_INSTALL_DIR
# Go into the MSeg installation directory
$ cd $MSEG_INSTALL_DIR
# Get the most recent stable version
$ git checkout 1.1-stable
# Create the build folder and go into it
$ mkdir build && cd build
# Run cmake
$ cmake ..
# Compile. Use 'make -j4' to use 4 cores
$ make
```

Now that the project is compiled, we proceed to add the `mseg*` terminaltool to `$PATH`:

```
$ echo "export PATH=\"$MSEG_INSTALL_DIR/etc/scripts/:\$PATH\"" >> ~/.bashrc
$ source ~/.bashrc
```

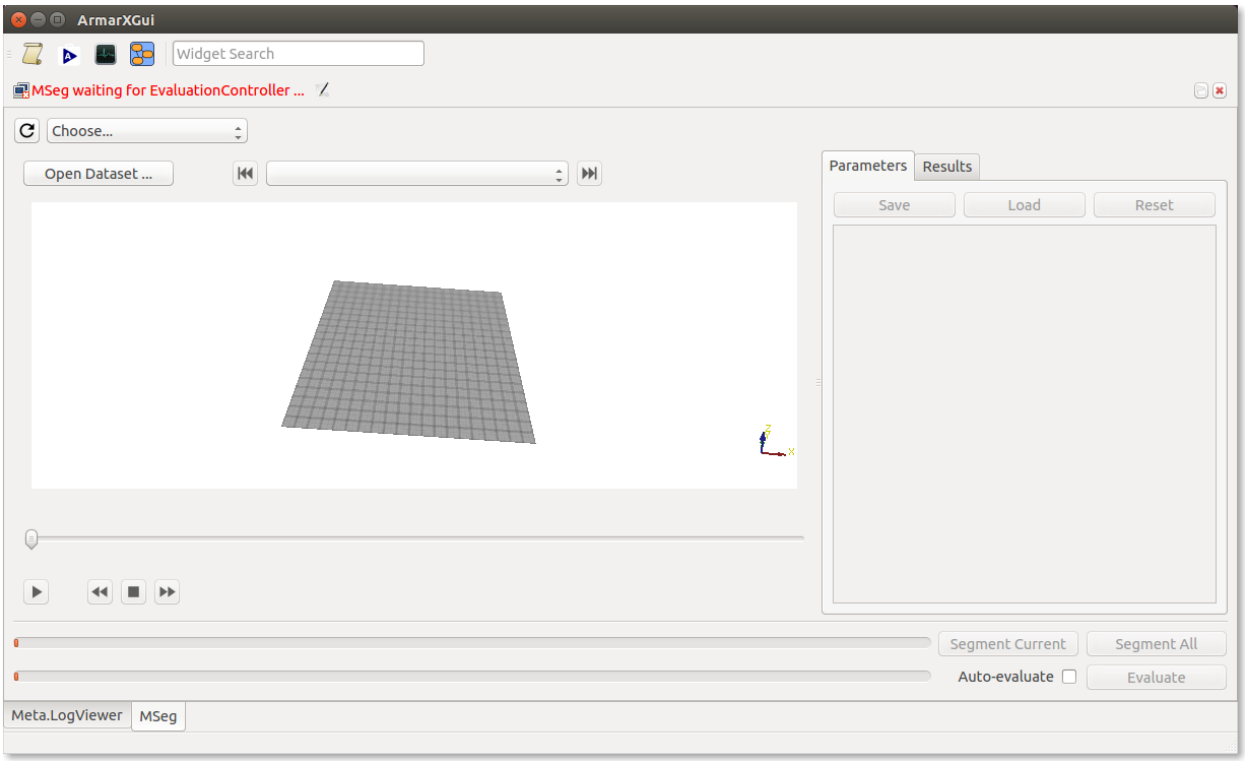
We can now start the ArmarX GUI and MSeg. Firstly, make sure that ArmarX is running: `armarx status`. If the output is similar to the output below, start ArmarX as indicated using `armarx start`.

```
# An output similar to this is shown when 'armarx status' is called
# and ArmarX is not running:
Exception while initializing Ice:
Ice.ConnectionRefusedException:
Connection refused
IceGrid is not running
Try starting it with 'armarx start'
```

Launch the GUI: `armarx gui`. You can now use the MSeg GUI by clicking on `Add Widget | MSeg`.

6.6.2 Start the Core Module

When you open the MSeg GUI you should see a window similar to this:



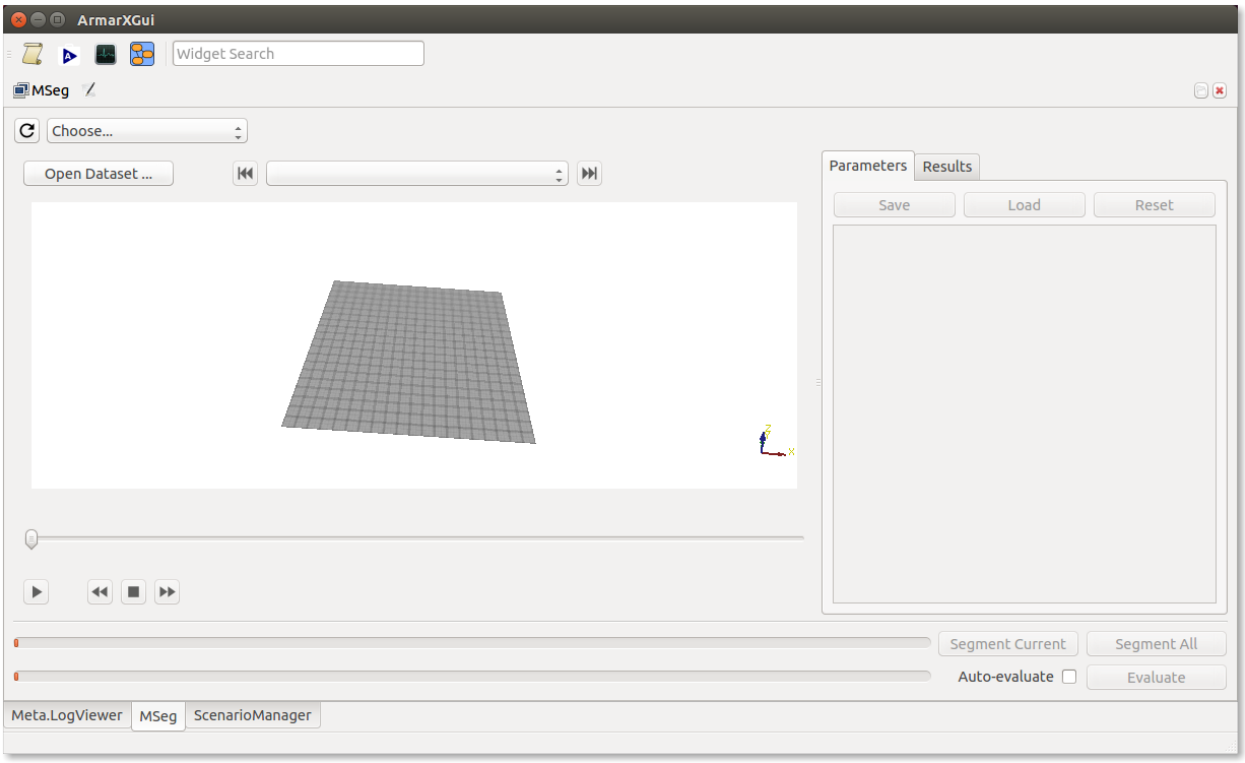
Note the red text which says:

> MSeg waiting for EvaluationController and SegmentationController

Both the `EvaluationController` and the `SegmentationController`, as well as the `DataExchange`, which is not listed here, are components that form the core module. In order for the GUI to work, these components have to be started.

```
# Start the core module
$ msegcm start
```

The components of the core module are now running. If you now head back to the MSeg GUI, you should see that the warning showed before disappeared.



Remember to start these components every time you reboot your computer. Also, when a component crashes, it may be required to restart them using `msegcm restart`. You will be notified if components need to be restarted by the red text as shown in the first graphic.

6.7 How to Proceed

With the core module being set up, you can now proceed to *compile a PLI* which is needed to offer an API to communicate with the core module in the programming language the motion segmentation algorithm is implemented in. Supported programming languages are C++, Python, Java and MATLAB.

CHAPTER 7

Compile a PLI from Source

This guide will show you how to compile a PLI from source for C++, Python, Java or MATLAB. It is assumed that the MSeg core module was installed as described in the *guide about compiling the MSeg core module from source*.

7.1 Programming Language Specific Prerequisites

Some PLIs require certain prerequisites which will be elaborated in the following.

7.1.1 C++, Python and Java

Nothing specific to consider.

7.1.2 MATLAB

We need to install the MATLAB Python libraries so the PLI can access the MATLAB engine.

```
# Change to the MATLAB installation's script folder
$ cd $MATLAB_HOME/extern/engines/python
# Run the install script
$ python setup.py install --user
```

7.2 Setup PLIs

Run one of these `make`-commands, depending on which PLI you want to use.

```
# Change into the MSeg build directory
$ cd $MSEG_INSTALL_DIR/build
# Use this target to build the C++ PLI
```

```
$ make plicpp
# Use this target to build the Python PLI
$ make plipython
# Use this target to build the Java PLI
$ make plijava
# Use this target to build the MATLAB PLI
$ make plimatlab
# Use this target to build all PLIs
$ make plis
```

7.3 Setup Bundled Algorithms

You can try the motion segmentation algorithms bundled with MSeg to see how they use the PLI APIs. The source codes are located in `$MSEG_INSTALL_DIR/source/algorithms`.

You can compile them with:

```
# Change into the MSeg build directory
$ cd $MSEG_INSTALL_DIR/build
# Use this target to build all algorithms
# (Requires that all PLIs were built)
$ make algorithms
# Use this target to build the ZVC algorithm
# (Requires that the C++ PLI was built)
$ make algorithmzvc
# Use this target to build the SSAV algorithm
# (Requires that the Python PLI was built)
$ make algorithmssav
# Use this target to build the PCA algorithm
# (Requires that the Java PLI was built)
$ make algorithmpca
```

The compiled algorithms can then be found in `$MSEG_INSTALL_DIR/build/bin` (the executables or scripts start with "algorithm"). Before launching the algorithm, please make sure that both ArmarX and the MSeg core module are running, otherwise the algorithm will crash. Run `armarx status` to see if ArmarX is running, and `armarx start` to start it. Similary, use `msegcm status` to see the status of the components of MSeg core module, and `msegcm start` to start them.

7.4 How to Proceed

Now that both the MSeg core module and the PLI is setup, you can proceed with integrating your own motion segmentation algorithm in *C++*, *Java*, *Python* or *MATLAB*.

CHAPTER 8

Architectural Overview

Coming soon

CHAPTER 9

The `mseg*` Terminal Tools

MSeg ships with three terminal tools, namely:

- `msegcm` Control the MSeg core module
- `msegdata` Manage the datasets
- `mseggen` Help setting up new motion segmentation algorithm projects

Note: You can run `<command> --help` at any time to get a quick overview on the usage and synopsis. If you need more detail for a given subcommand, you can run `<command> <subcommand> --help` instead. *The usage of the tools may change with future versions of MSeg*. If the information on this page is contradicting the information given from the `--help`, *always rely on that help*.

9.1 `msegcm` Tool

With `msegcm`, the MSeg core module can be controlled.

9.1.1 Available subcommands

- `status`: Prints the status of the MSeg core module processes
- `start`: Start the MSeg core module processes
- `stop`: Stop the MSeg core module processes
- `restart`: Stop and start the MSeg core module processes

If an unexpected error leads to the crash of one or more processes, you will need to restart the MSeg core module using `msegcm restart`. A crash can be diagnosed if `msegcm status` indicates that the status of the MSeg core module is either "mixed" or "stopped".

Note: The MSeg core module cannot start properly if ArmarX is not running. You can query the status of ArmarX by running `armarx status`, and you can start it by running `armarx start`.

9.2 `msegdata` Tool

With `msegdata`, the datasets available with MSeg can be managed. The datasets are not included by default because newer datasets may still be useful for older versions of MSeg. The datasets are versioned separately for that reason.

9.2.1 Available subcommands

- `fetch`: Fetches the datasets and stores them locally into `~/.mseg/datasets` (This is a Git repository). This subcommand **must** be ran once to initialise the datasets (see *quickstart guide*)
- `update`: Updates the local datasets to the newest versions. Old versions will remain untouched

Note: If you are experienced with Git, then please note that `msegdata` is merely a façade for the corresponding Git repository (`~/.mseg/datasets`) of the datasets. If you prefer, you can use Git in first place instead of `msegdata`. You are *still* advised to at least run `msegdata fetch` to clone the Git repository into the correct location.

9.3 `mseggen` Tool

The `mseggen` command is a helper tool to kick-start a new motion segmentation algorithm project. It creates all source files (and build files, if appropriate) needed for a given programming language with well-commented stubs and dummy-implementations.

9.3.1 Synopsis

```
mseggen [--base-path BASE_PATH] [--train] (--cpp | --python | --java |
--matlab) <project-name>
```

- The option `--base-path` can be used to set the base path in which the project should be created. It defaults to the current working directory
- The flag `--train` is optional. If set, the code will be included to use the train API. If not set, those parts will be commented out
- The flags `--cpp`, `--python`, `--java` and `--matlab` are mutually exclusive. Exactly one of them must be set
- `<project-name>` is the name of the project, respectively the motion segmentation algorithm. It must me alphanumeric and start with a character

9.3.2 Examples

- To implement the PCA approach by Barbič in Java the usage would be: `mseggen --java PCABarbic`
- To implement a machine learning algorithm in Python, located in `~/projects`: `mseggen --base-path ~/projects --train --java MLApproach`

CHAPTER 10

The MSeg Graphical User Interface

Coming soon

CHAPTER 11

`SegmentationAlgorithm` Interface

All PLIs offer the class `SegmentationAlgorithm` which can be derived in order to communicate with the core module.

11.1 Properties

Properties of `SegmentationAlgorithm`.

11.1.1 `data`

Type: *DataExchangeProxy*

Description Proxy to exchange data between the PLI and the core module. This property is initialised by the corresponding PLI and must not be overwritten.

11.1.2 `name`

Type: String

Description: The name of the algorithm. The name must not be empty and must be alphanumeric and start with a character. Set this property in the constructor.

11.1.3 `requiresTraining`

Type: bool

Default: false

Description: Flag to indicate whether the algorithm should be trained. Set this property in the constructor.

11.1.4 trainingGranularity

Type: Granularity

Default: Granularity.Medium

Description: Used to filter the training ground truth data for the set granularity. Possible values are `Granularity.Fine`, `Granularity.Medium`, and `Granularity.Rough`. Set this property in the constructor. This property is only considered if `requiresTraining` is set to true.

11.2 Methods

Methods of `SegmentationAlgorithm`.

11.2.1 SegmentationAlgorithm() or __init__() or constructor.m

Signature: `SegmentationAlgorithm` ()

Description: Constructor. Set properties like `name`, `requiresTraining` or `trainingGranularity` here. Set algorithm parameters here.

11.2.2 train() or train.m

Signature: void `train` ()

Return: void

Description: This method will be called for each motion recording in the training dataset. You can fetch the current motion recording here using the `data` property and use it to train your variables.

11.2.3 resetTraining() or reset_training.m

Signature: void `resetTraining` ()

Return: void

Description: This method will be called after one part of the cross validation was executed. Make sure to reset all trained variables to not falsify the evaluation.

11.2.4 segment() or segment.m

Signature: void `segment` ()

Return: void

Description: This method will be called for each motion recording in the testing dataset.

11.2.5 registerBoolParameter()

Signature: void registerBoolParameter (String name, String description, bool defaultValue)

Return: void

Description: Registers a boolean parameter name with a default value of defaultValue to tweak in the GUI.

11.2.6 registerIntParameter()

Signature: void registerIntParameter (String name, String description, int defaultValue, int minimumValue = INT_MIN, int maximumValue = INT_MAX)

Return: void

Description: Registers an int parameter name with a default value of defaultValue to tweak in the GUI. The allowed range of values can be controlled with minimumValue and maximumValue.

11.2.7 registerFloatParameter()

Signature: void registerFloatParameter (String name, String description, float defaultValue, int decimals = 2, float minimumValue = -FLOAT_MAX, float maximumValue = FLOAT_MAX)

Return: void

Description: Registers a float parameter name with a default value of defaultValue to tweak in the GUI. The allowed range of values can be controlled with minimumValue and maximumValue. The precision (number of decimals) can be controlled with decimals.

11.2.8 registerStringParameter()

Signature: void registerStringParameter (String name, String description, String defaultValue)

Return: void

Description: Registers a String parameter name with a default value of defaultValue to tweak in the GUI.

11.2.9 registerJsonParameter()

Signature: void registerJsonParameter (String name, String description, JSON defaultValue)

Return: void

Description: Registers a JSON parameter name with a default value of defaultValue to tweak in the GUI.

11.2.10 `getBoolParameter()`

Signature: bool `getBoolParameter` (String `name`)

Return: bool

Description: Get a registered bool parameter by its `name`.

11.2.11 `getIntParameter()`

Signature: int `getIntParameter` (String `name`)

Return: int

Description: Get a registered int parameter by its `name`.

11.2.12 `getFloatParameter()`

Signature: float `getFloatParameter` (String `name`)

Return: float

Description: Get a registered float parameter by its `name`.

11.2.13 `getStringParameter()`

Signature: String `getStringParameter` (String `name`)

Return: String

Description: Get a registered String parameter by its `name`.

11.2.14 `getJsonParameter()`

Signature: JSON `getJsonParameter` (String `name`)

Return: JSON

Description: Get a registered JSON parameter by its `name`.

CHAPTER 12

`DataExchangeProxy` Interface

An instance of the `DataExchangeProxy` class is accessible via the `data` property of the derived *SegmentationAlgorithm* instance. It is used to communicate with the core module to exchange data.

12.1 Methods

Methods of `DataExchangeProxy`.

Method *getFrameCount()*

12.1.1 `getFrameCount()`

Signature: int `getFrameCount` ()

Return: int

Description: Returns the total number of frames for the current motion recording.

12.1.2 `getMMMFile()`

Signature: String `getMMMFile` ()

Return: String

Description: Returns the MMM file of the current motion recording as String. MMM is an XML format file which can be parsed with various XML readers or the already existing MotionReaderXML for C++.

12.1.3 `getGroundTruth()`

Signature: List<int> `getGroundTruth` ()

Return: List<int>

Description: Returns the ground truth for the current motion recording with the granularity determined by the `trainingGranularity` property of the derived SegmentationAlgorithm instance.

12.1.4 `getJointAnglesForFrame()`

Signature: List<float> `getJointAnglesForFrame` (int `frameNumber`)

Return: List<float>

Description: Returns the joint angles of the frame with the number `frameNumber` for the current motion recording.

12.1.5 `reportKeyFrame()`

Signature: void `reportKeyFrame` (int `frameNumber`)

Return: void

Description: Reports that the frame with the number `frameNumber` is a key frame where a segmentation occurs.

12.1.6 `reportKeyFrames()`

Signature: void `reportKeyFrames` (List<int> `frameNumbers`)

Return: void

Description: Reports that the list of frame numbers `frameNumbers` are key frames where segmentations occur.

CHAPTER 13

Data Type Mappings

Data types used in this documentation map to different data structures or constructs depending on the used programming language. The following table shows the corresponding mappings.

CHAPTER 14

Changelog

14.1 Version 1.1.1

Bugfixes

- Fix a CMake dependency problem (#81)

14.2 Version 1.1.0

Features

- Implement parameter save/load (#4)
- Allow exporting evaluation results as XML (#70)

Bugfixes

- Load motions in the background (#65)
- Disable segment current button (not yet implemented) (#67)
- Fix a bug where motions are loaded twice (#68)
- Disable segment and evaluate buttons while a new dataset ist being opened (#71)
- Fix an issue where thousands separators could cause crashes (#75)
- Fix issues with a more recent version of MMMTools (#80)

14.3 Version 1.0.2

Bugfixes

- Fix an issue where thousands separators could cause crashes (#75)

14.4 Version 1.0.1

Bugfixes

- Disable segment current button (not yet implemented) (#67)
- Fix a bug where motions are loaded twice (#68)

14.5 Version 1.0.0

Bugfixes

- Fix MATLAB dependency issues (#56)
- Fix stacktrace output for MATLAB exceptions (#59)
- Made everything locale independent (#33, #61)
- Fix various UI issues (#44, #60, #64)

14.6 Version 1.0.0-rc5

Features

- Improved error handling (#47, #49, #50)
- Refactor algorithm name handling (#48)
- Unify PLI output (#45)

14.7 Version 1.0.0-rc4

Features

- Add commands to `mseg` to start/stop/restart the core module (#37)
- Add subcommand to `mseg` to generate a Java skeleton project (#40)
- Include MATLAB PLI (#23)
- Add subcommand to `mseg` to generate a MATLAB skeleton project (#42)

Bugfixes

- Fix a bug where shutting down a Java algorithm would raise an exception

14.8 Version 1.0.0-rc3

Features

- Add `mseg` tool
- Add subcommand to `mseg` to generate a C++ skeleton project (#35)
- Add subcommand to `mseg` to generate a Python skeleton project (#36)

14.9 Version 1.0.0-rc2

Features

- Include ZVC algorithm (#25)
- Allow algorithms to require training data of a certain granularity (#27)

Bugfixes

- Fix an unimplemented method preventing Java PLI from compiling (#30, #31)

14.10 Version 1.0.0-rc1

- *Initial release candidate*

CHAPTER 15

License

15.1 GNU GENERAL PUBLIC LICENSE

Version 2, June 1991

> Copyright (C) 1989, 1991 Free Software Foundation, Inc. 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA
>
> Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

15.1.1 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 Lesser 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.

15.2 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.

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.

15.3 END OF TERMS AND CONDITIONS

15.4 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.

```
one line to give the program's name and an idea of what it does.
Copyright (C) yyyy name of author

This program is free software; you can redistribute it and/or
modify it under the terms of the GNU General Public License
as published by the Free Software Foundation; either version 2
of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program; if not, write to the Free Software
Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA  02110-1301, USA.
```

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

```
Gnomovision version 69, Copyright (C) year name of author
Gnomovision comes with ABSOLUTELY NO WARRANTY; for details
type `show w'.  This is free software, and you are welcome
to redistribute it under certain conditions; type `show c'
for details.
```

The hypothetical commands *show w* and *show c* should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than *show w* and *show c*; they could even be mouse-clicks or menu items–whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names:

```
Yoyodyne, Inc., hereby disclaims all copyright
interest in the program `Gnomovision'
(which makes passes at compilers) written
by James Hacker.

signature of Ty Coon, 1 April 1989
Ty Coon, President of Vice
```

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 Lesser General Public License instead of this License.