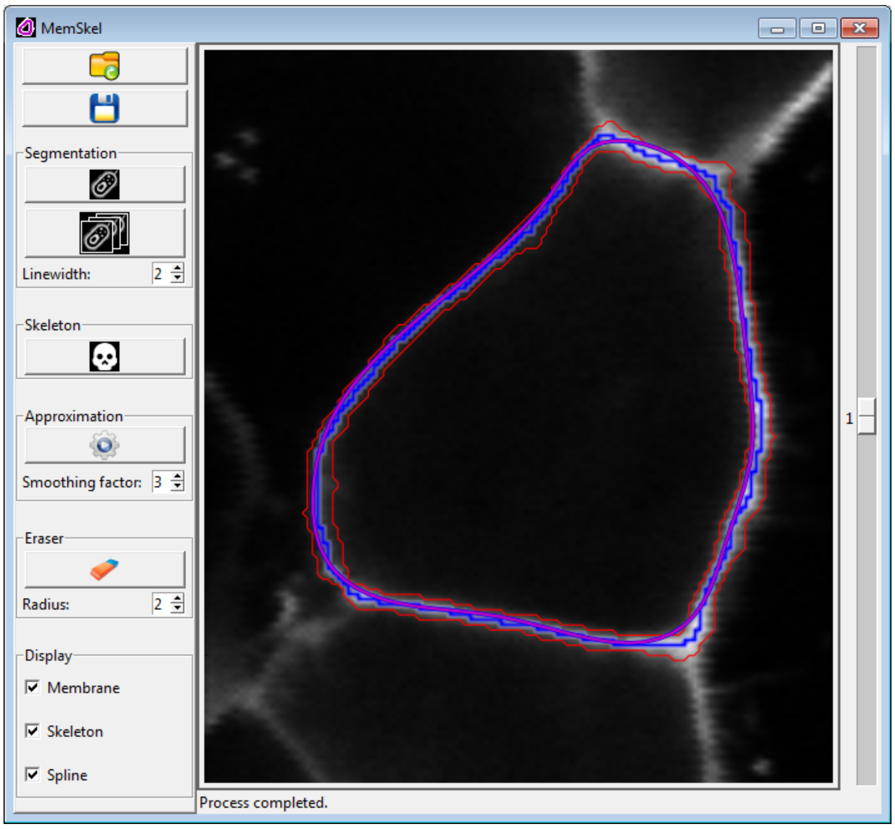
MemSkel - User Guide

Tomas Ryba
tryba@kky.zcu.cz

July 6, 2015

1 Introduction

The application MemSkel (**Mem**brane **Skel**etonization) is a tool for segmentation of a cell membrane. The process is interactive which means that a user guidance is necessary. The user has to mark (with the mouse) at least one point in the image. These marked points are called seeds. After the segmentation is done it is possible to edit obtained result manually with Eraser tool. When the user is satisfied with the segmentation the skeletonization can be run. This process finds the medial axis of the segmentation mask. For this to work, the segmentation mask must be closed otherwise the skeletonization will fail. The output of the skeletonization (the skelet) can be further approximated using b-spline with variable smoothing parameter.



Process completed.

Figure 1: The main window of the application.

2 Graphical User Interface

The application consists of the main parts: tool window, visualisation. In this section individual components of the GUI are described. The components are numbered as follows: 2.

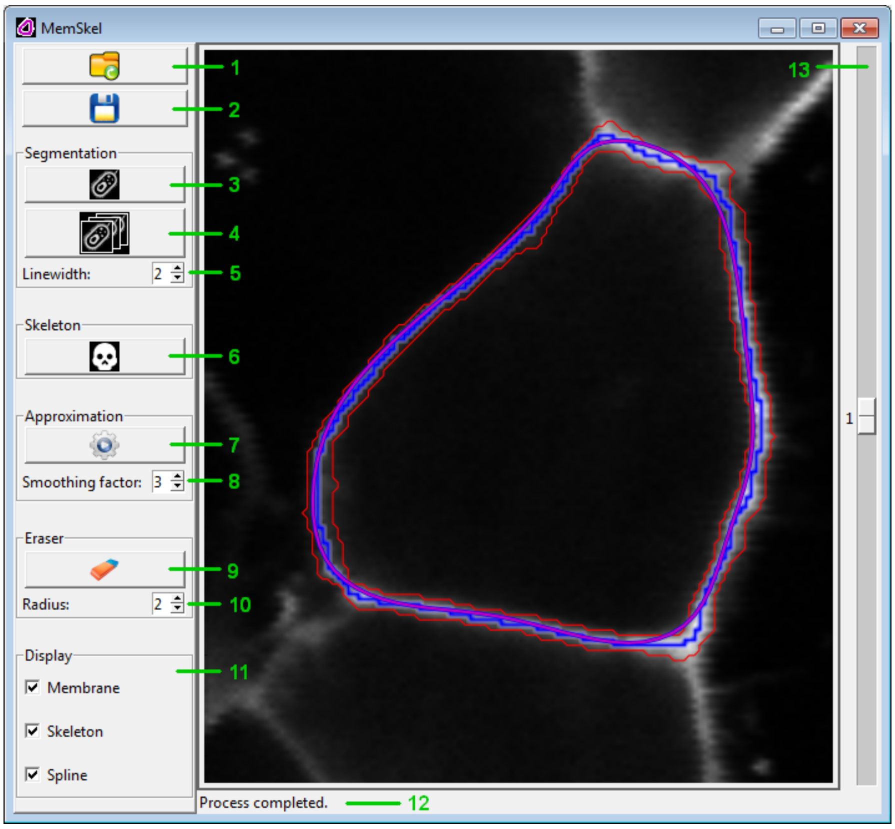


Figure 2: The main window with numbered components.

The numbered main components from figure 2 are described bellow:

1. Open file . . . opens a dialogue window for loading data from a file.
2. Save . . . opens a dialogue window for saving the results.
3. Segmentation of a frame . . . segments actual frame.
4. Segmentation of the stack . . . segments all frames (pages) in data.
5. Linewidth . . . spinbox for changing the width of line, which is used for marking the membrane.

6. Skeletonization . . . runs the skeletonization procedure to get the medial axis of the membrane mask.
7. Approximation . . . runs the approximation of the medial axis with the smoothing parameter given by the spinbox bellow.
8. Smoothing factor . . . spinbox for changing the smoothing factor that is used in approximation.
9. Eraser . . . runs process for deleting points from the membrane mask.
10. Radius . . . spinbox for changing the Eraser radius.
11. Display . . . checkboxes for enabling/disabling visualisation of appropriate parts.
12. Status bar . . . displays information about the state of the application.
13. Slider . . . allows the user to change data frame.

3 Use of the Application

This section refers to individual components with numbers that corresponds to the figure 2 and the list above. The references are hold in circles with the number inside it, e.g. the reference of the button for opening a file has the form ①.

3.1 Running and Closing

To start the application the file *memSkel_main.exe* needs to be runned. Traditionally, the application terminates by clicking the red cross button in upper right corner of the main window.

3.2 Loading the Data

By clicking the button ① the file dialogue for opening a file pops up. At this moment, only TIFF format for the input data is supported. The data format should be complete, i.e. it should contain metainformation about frame size. If you're not able to open data, re-saving it another image viewer/editor should help. This way the missing data should be filled in. One such an editor is *IrfanView*. Another approach that could help is to save the data in 8-bit format.

When the data is loaded, they are shown in the editor.

3.3 Segmentation of Cell Membrane

As already mentioned, the segmentation procedure needs to know at least one point (seed) that belongs to the membrane. The mode for marking the seed points is ran by clicking the button ③ or ④. After that the user can mark points by moving the mouse when the left mouse button is held down. After hitting the Enter key, the algorithm starts segmenting the rest of the membrane. After the algorithm terminates the user is able to supply more seeds and rerun the segmentation process (again by the Enter key).

To end the segmentation process, you must wait until the algorithm terminates. After that just press Enter without providing any other seed points. Then it is possible to use the Eraser to delete areas outside the membrane.

During the editation mode it is possible to change the line width with the spinbox ⑤ or by the mouse wheel. This way you can provide the algorithm with more points or achieve higher precision by thinning the line.

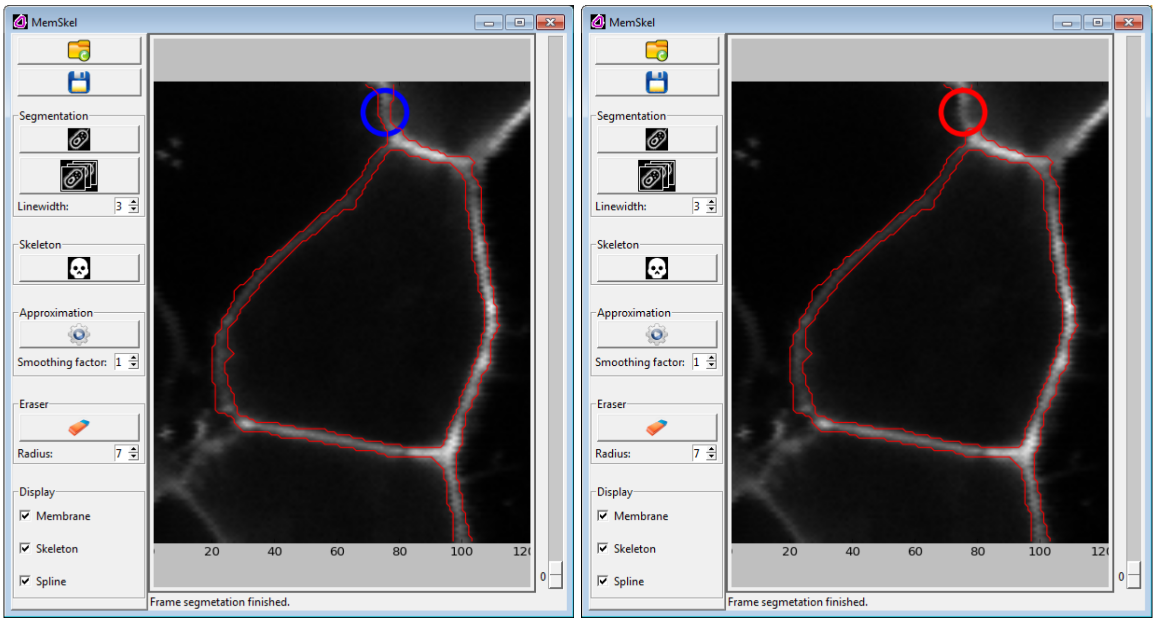
Clicking the button ④ segments the whole stack, i.e. all frames or pages. In this case, segmentation of the first frame is the same as described above. After segmenting the actual frame, the segmentation starts to grow to other frames. To do this, the algorithm needs to know the skeleton, therefore its computation is ran automatically. The editing of the segmented mask is possible after all frames are segmented.

3.4 Eraser Tool

Using the Eraser tool it is possible to edit the binary mask of the membrane. If you press the key ⑨ then a blue circle around the mouse cursor will appear. The size of the circle represents area that is affected by the Eraser and could be changed by the spinbox ⑩ or by the mouse wheel. By holding the left mouse button the erasing mode starts, which is indicated by change of the circle color from blue to red, see figure 3. When erasing, all pixels of the membrane that are located inside the circle will be erased from the mask, i.e. marked as pixels that do not belong to the membrane. The user can hold down the left mouse button and move it to perform ”erasing on the move”.

3.5 Skeletonization

The next step after the membrane is found is to define its medial axis, i.e. its skeleton. This process is triggered by hitting the button ⑥. The membrane needs to be closed otherwise the skeletonization yields a one-point skeleton, see fig. 4.



Frame segmetation finished.

Frame segmetation finished.

(a) (b)

Figure 3: The use of the Eraser tool. The figure. 3a shows the erasing mode, figure 3b shows situation when the user holds down the left mouse button.

3.6 Approximation

The skeleton can be further approximated by a b-spline by clicking on the button ⑦. The user is also able to define the smoothness factor by the spinbox ⑧. An example showing the use of approximation is shown in fig. 5.

3.7 Saving the results

To save the results the user needs to click the button ②. After that the software creates two files:

1. binary image defining the segmented membrane,
2. text file describing the parameters of the approximating spline.

4 Miscellaneous

4.1 Status bar

The status bar (12) shows main information about running processes in the editor, e.g. information about successful/unsuccessful data loading and saving, process flow etc. When segmenting multiframe data, the right part of the status bar shows a progress bar. It informs the user what part of data was already processed and what part remains.

4.2 Log file

If the software ends with a critical error, a window pops up that informs the user about logging the error in log file. This file is named *memskel_main.exe.log* and provide more information about the error. This file is extremely important for the developers to debug the software.

5 Running the software from source

This section describes how to install and run the software.

5.1 Required packages

The software uses several packages that are listed here. Used package version is also provided.

- Python 2.7.5
- numpy-MKL-1.7.1
- scipy-0.12.0
- matplotlib-1.1.1 (newer version doesn't work), before installation you need to install following packages:
  - python-dateutil-1.5
  - pytz-2013b
  - pyparsing-1.5.7
- pylibtiff-0.3.0.dev82
- Pillov-2.0.0

5.2 Order of installing packages

An order of installing the packages needs to be fulfilled, otherwise the software might not work.

1. python-2.7.5.msi . . . installs the Python itself; it does not matter if you install 32 or 64 bit version as long as you installed the same bit version of all following packages,
2. numpy-MKL-1.7.1.win32-py2.7.exe . . . working with n-dimensional arrays,
3. scipy-0.12.0.win32-py2.7.exe . . . for approximation of the skeleton,
4. python-dateutil-1.5.win32-py2.7.exe . . . prerequisite of the matplotlib package,
5. pytz-2013b.win32-py2.7.exe . . . prerequisite of the matplotlib package,
6. pyparsing-1.5.7.win32-py2.7.exe . . . prerequisite of the matplotlib package,
7. matplotlib-1.1.1.win32-py2.7.exe . . . visualization, more recent version might not work,
8. pylibtiff-0.3.0.dev82.win32-py2.7.exe . . . working with TIFF images,
9. Pillow-2.0.0.win32-py2.7.exe . . . image processing,
10. setuptools-0.7.4.win32-py2.7.exe . . . instalation of packages from source codes,
11. pymorph-0.96 . . . morphological operations; installation is described bellow.

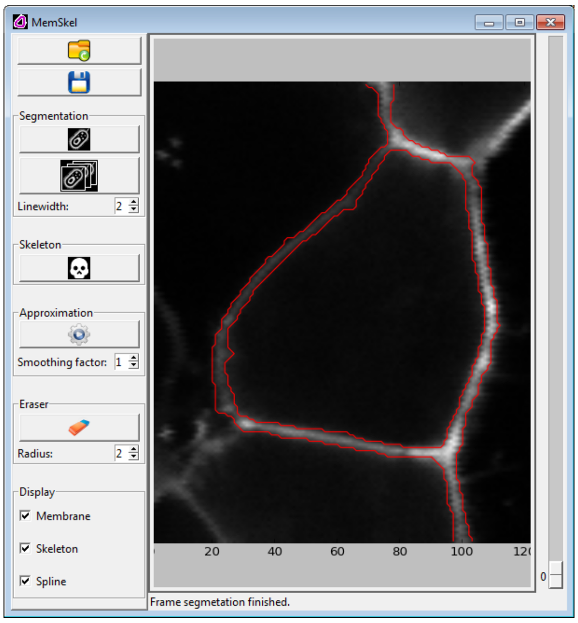
For installation of the *pymorph* package it is necessary to add the Python do system variable PATH. Some package may to do that automatically during its installation - in that case this step is not necessary. To verify if the Python is in the system PATH you can do the following:

1. run command line, e.g. with *Win+R* and writing *cmd* as in fig. 6,
2. write *python –help* in the command line - some text about python command should be displayed as in fig 7a.
3. if some text is shown as in fig. 7a, the Python is in your system PATH,
4. if a text sayng that Python is unknown command is shown (7b), Python is not in you system Path and you need to put it there, e.g. with command *setx path "%path%;c:\Python27"*; information about successful operation should be shown (8); after restarting the command line an repeating the step 2 should provide you with successful message.

When the Python is in the system PATH you can install the package *pymorhp* with the *setuptools*. It is necessary in command line to get inside a directory *pymorph-0.96*. Then you can run the installation with the command *python setup.py install*. When the installation is successful, the command line should inform you with a successful message(9).

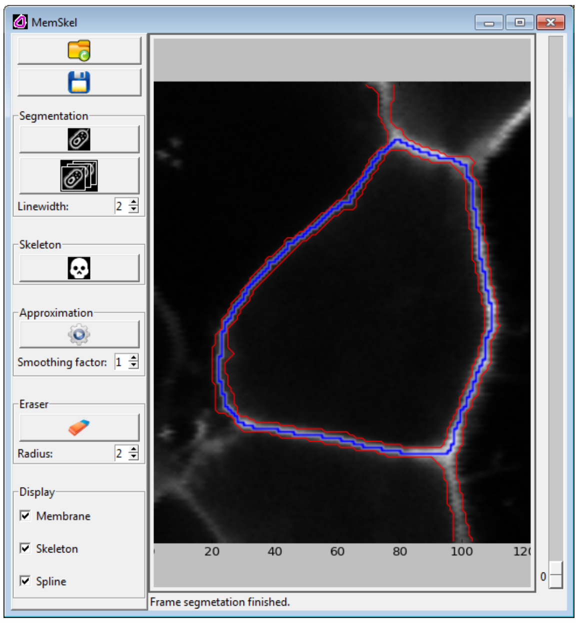
5.3 Running the software

If the Python is in the system PATH (it should be), it is possible to run the MemSkel with file *memSkel_main.py*. Eventually it is possible to run it from the command line with command *python memSkel_main.py*, but you need to be inside the directory which contains the file *memSkel_main.py*, see fig. 10.



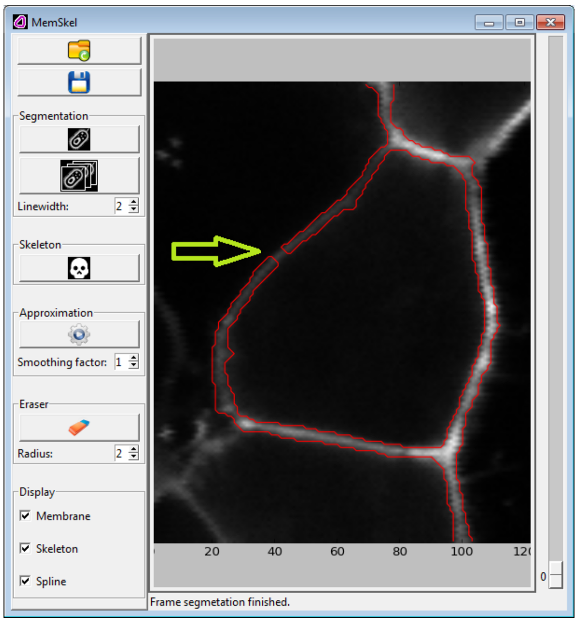
Linewidth: 2
Smoothing factor: 1
Radius: 2
☑ Membrane
☑ Skeleton
☑ Spline
Frame segmentation finished.

(a)



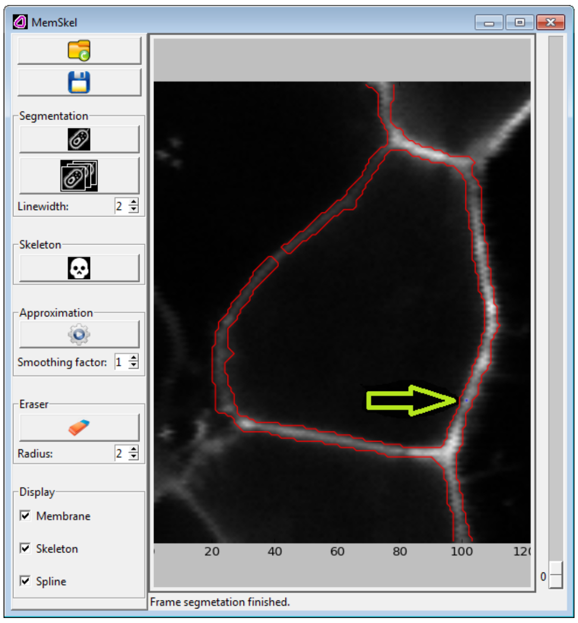
Linewidth: 2
Smoothing factor: 1
Radius: 2
☑ Membrane
☑ Skeleton
☑ Spline
Frame segmentation finished.

(b)



Linewidth: 2
Smoothing factor: 1
Radius: 2
☑ Membrane
☑ Skeleton
☑ Spline
Frame segmentation finished.

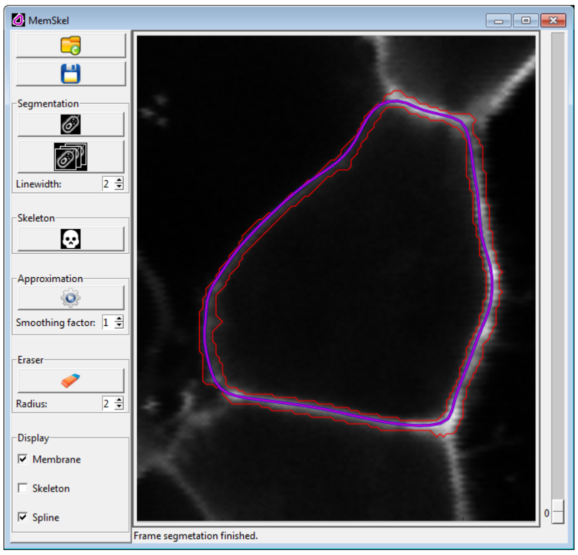
(c)



Linewidth: 2
Smoothing factor: 1
Radius: 2
☑ Membrane
☑ Skeleton
☑ Spline
Frame segmentation finished.

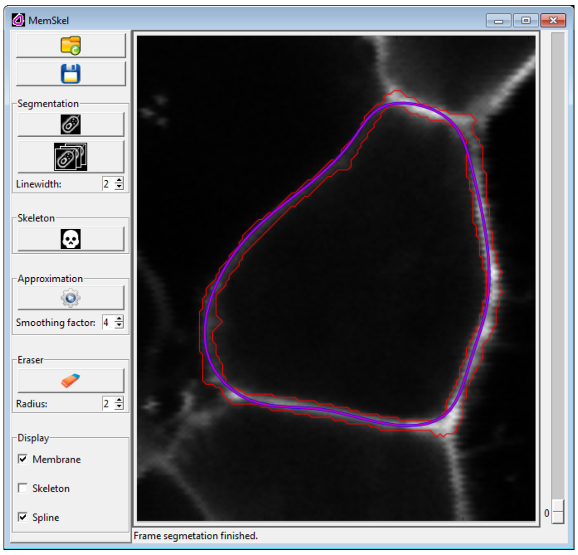
(d)

Figure 4: An example of the skeletonization. Fig. 4a shows right segmented membrane and its skeleton is shown in 4b. If the membrane is not closed, see fig. 4c, the skeleton will end like an point as shown in fig. 4d.



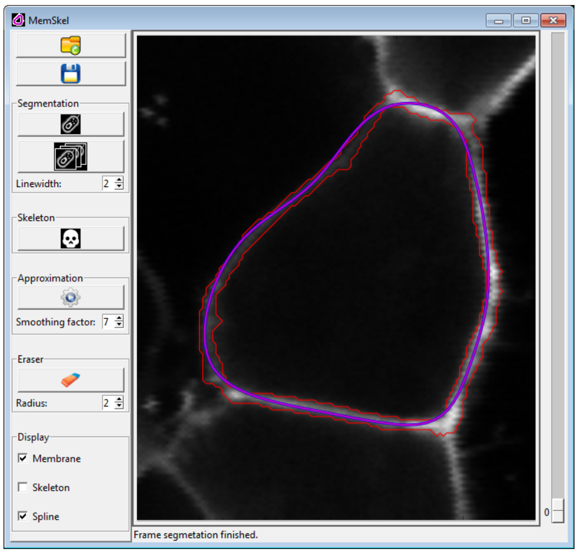
Linewidth: 2
Smoothing factor: 1
Radius: 2
☑ Membrane
☐ Skeleton
☑ Spline
Frame segmetation finished.

(a)



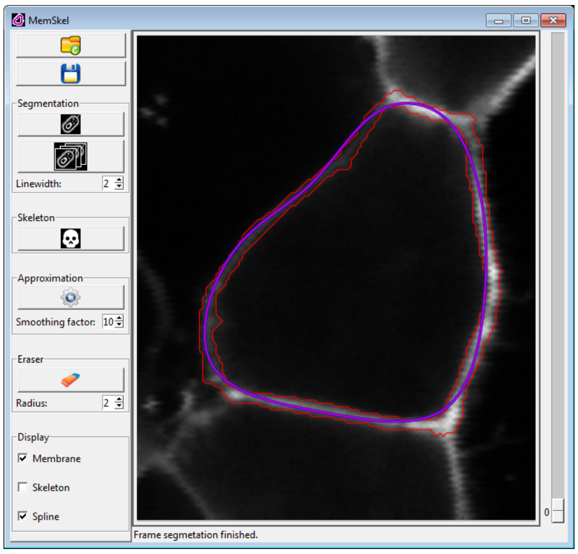
Linewidth: 2
Smoothing factor: 4
Radius: 2
☑ Membrane
☐ Skeleton
☑ Spline
Frame segmetation finished.

(b)



Linewidth: 2
Smoothing factor: 7
Radius: 2
☑ Membrane
☐ Skeleton
☑ Spline
Frame segmetation finished.

(c)



Linewidth: 2
Smoothing factor: 10
Radius: 2
☑ Membrane
☐ Skeleton
☑ Spline
Frame segmetation finished.

(d)

Figure 5: An example of the approximation of the skeleton with different smoothing factor: factor = 1 ... 5a, factor = 4 ... 5b, factor = 7 ... 5c a factor = 10 ... 5d.

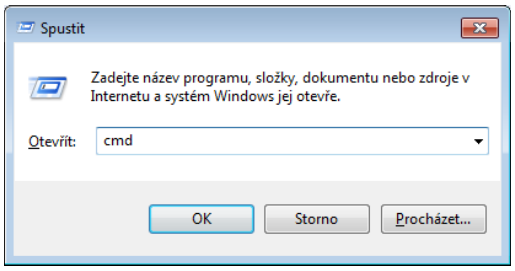
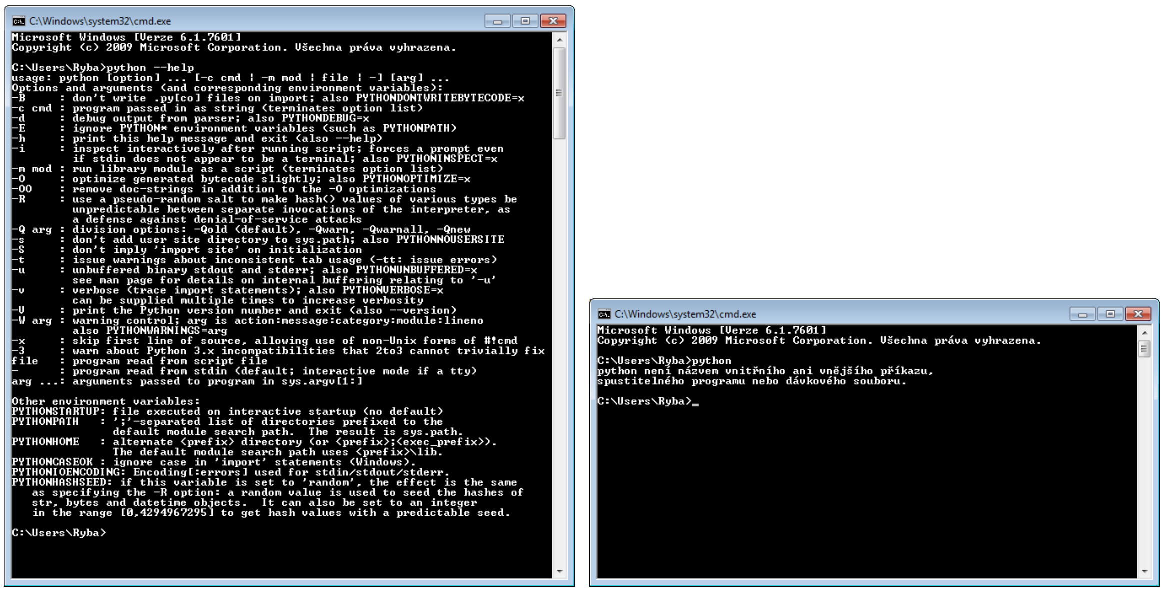

Spustit
Zadejte název programu, složky, dokumentu nebo zdroje v Internetu a systém Windows jej otevře.
Otevřít:
cmd
OK
Storno
Procházet...

Figure 6: Running the command line.



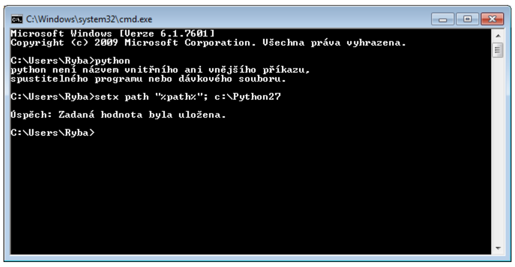
```
Microsoft Windows [Verze 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. Všechna práva vyhrazena.

C:\Users\Ryba>python --help
usage: python [option] ... [-c cmd | -m mod | file | -] [arg] ...
Options and arguments (and corresponding environment variables):
-B     : don't write .py[co] files on import; also PYTHONDONTWRITEBYTECODE=x
-c cmd : program passed in as string (terminates option list)
-d     : debug output from parser; also PYTHONDEBUG=x
-E     : ignore PYTHON* environment variables (such as PYTHONPATH)
-h     : print this help message and exit (also --help)
-i     : inspect interactively after running script; forces a prompt even
         if stdin does not appear to be a terminal; also PYTHONINSPECT=x
-m mod : run library module as a script (terminates option list)
-O     : optimize generated bytecode slightly; also PYTHONOPTIMIZE=x
-OO    : remove doc-strings in addition to the -O optimizations
-R     : use a pseudo-random salt to make hash() values of various types be
         unpredictable between separate invocations of the interpreter, as
         a defense against denial-of-service attacks
-Q arg : division options: -Qold (default), -Qwarn, -Qwarnall, -Qnew
-s     : don't add user site directory to sys.path; also PYTHONNOUSERSITE
-S     : don't imply 'import site' on initialization
-t     : issue warnings about inconsistent tab usage (-tt: issue errors)
-u     : unbuffered binary stdout and stderr; also PYTHONUNBUFFERED=x
         see man page for details on internal buffering relating to '-u'
-v     : verbose (trace import statements); also PYTHONVERBOSE=x
         can be supplied multiple times to increase verbosity
-V     : print the Python version number and exit (also --version)
-W arg : warning control; arg is action:message:category:module:lineno
         also PYTHONWARNINGS=arg
-x     : skip first line of source, allowing use of non-Unix forms of #!cmd
-3     : warn about Python 3.x incompatibilities that 2to3 cannot trivially fix
file   : program read from script file
-      : program read from stdin (default; interactive mode if a tty)
arg ...: arguments passed to program in sys.argv[1:]

Other environment variables:
PYTHONSTARTUP: file executed on interactive startup (no default)
PYTHONPATH   : ';'-separated list of directories prefixed to the
               default module search path.  The result is sys.path.
PYTHONHOME   : alternate <prefix> directory (or <prefix>;<exec_prefix>).
               The default module search path uses <prefix>\lib.
PYTHONCASEOK : ignore case in 'import' statements (Windows).
PYTHONIOENCODING: Encoding[:errors] used for stdin/stdout/stderr.
PYTHONHASHSEED: if this variable is set to 'random', the effect is the same
   as specifying the -R option: a random value is used to seed the hashes of
   str, bytes and datetime objects.  It can also be set to an integer
   in the range [0,4294967295] to get hash values with a predictable seed.

C:\Users\Ryba>
```

(a) Python is in system PATH

```
Microsoft Windows [Verze 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. Všechna práva vyhrazena.

C:\Users\Ryba>python
python není názvem vnitřního ani vnějšího příkazu,
spustitelného programu nebo dávkového souboru.

C:\Users\Ryba>_
```

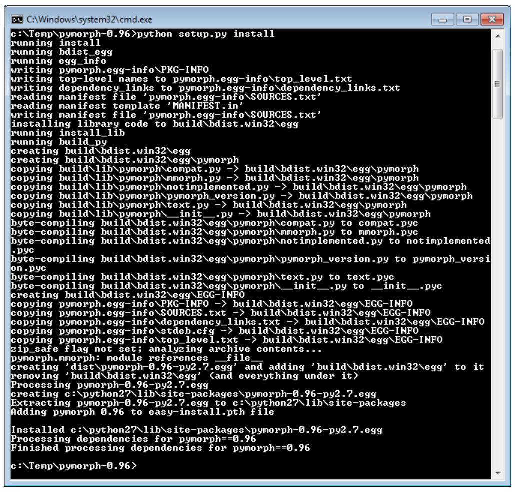
(b) Python is not in system PATH

Figure 7: Checking whether the Python is in your system PATH variable.



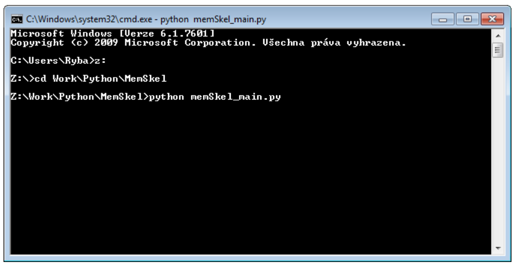
```
Microsoft Windows [Verze 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. Všechna práva vyhrazena.

C:\Users\Ryba>python
python není názvem vnitřního ani vnějšího příkazu,
spustitelného programu nebo dávkového souboru.

C:\Users\Ryba>setx path "%path%"; c:\Python27

Úspěch: Zadaná hodnota byla uložena.

C:\Users\Ryba>
```

Figure 8: Adding Python to the system PATH.



```
c:\Temp\pymorph-0.96>python setup.py install
running install
running bdist_egg
running egg_info
writing pymorph.egg-info\PKG-INFO
writing top-level names to pymorph.egg-info\top_level.txt
writing dependency_links to pymorph.egg-info\dependency_links.txt
reading manifest file 'pymorph.egg-info\SOURCES.txt'
reading manifest template 'MANIFEST.in'
writing manifest file 'pymorph.egg-info\SOURCES.txt'
installing library code to build\bdist.win32\egg
running install_lib
running build_py
creating build\bdist.win32\egg
creating build\bdist.win32\egg\pymorph
copying build\lib\pymorph\compat.py -> build\bdist.win32\egg\pymorph
copying build\lib\pymorph\mmorph.py -> build\bdist.win32\egg\pymorph
copying build\lib\pymorph\notimplemented.py -> build\bdist.win32\egg\pymorph
copying build\lib\pymorph\pymorph_version.py -> build\bdist.win32\egg\pymorph
copying build\lib\pymorph\text.py -> build\bdist.win32\egg\pymorph
copying build\lib\pymorph\__init__.py -> build\bdist.win32\egg\pymorph
byte-compiling build\bdist.win32\egg\pymorph\compat.py to compat.pyc
byte-compiling build\bdist.win32\egg\pymorph\mmorph.py to mmorph.pyc
byte-compiling build\bdist.win32\egg\pymorph\notimplemented.py to notimplemented
.pyc
byte-compiling build\bdist.win32\egg\pymorph\pymorph_version.py to pymorph_versi
on.pyc
byte-compiling build\bdist.win32\egg\pymorph\text.py to text.pyc
byte-compiling build\bdist.win32\egg\pymorph\__init__.py to __init__.pyc
creating build\bdist.win32\egg\EGG-INFO
copying pymorph.egg-info\PKG-INFO -> build\bdist.win32\egg\EGG-INFO
copying pymorph.egg-info\SOURCES.txt -> build\bdist.win32\egg\EGG-INFO
copying pymorph.egg-info\dependency_links.txt -> build\bdist.win32\egg\EGG-INFO
copying pymorph.egg-info\stdeb.cfg -> build\bdist.win32\egg\EGG-INFO
copying pymorph.egg-info\top_level.txt -> build\bdist.win32\egg\EGG-INFO
zip_safe flag not set; analyzing archive contents...
pymorph.mmorph: module references __file__
creating 'dist\pymorph-0.96-py2.7.egg' and adding 'build\bdist.win32\egg' to it
removing 'build\bdist.win32\egg' (and everything under it)
Processing pymorph-0.96-py2.7.egg
creating c:\python27\lib\site-packages\pymorph-0.96-py2.7.egg
Extracting pymorph-0.96-py2.7.egg to c:\python27\lib\site-packages
Adding pymorph 0.96 to easy-install.pth file

Installed c:\python27\lib\site-packages\pymorph-0.96-py2.7.egg
Processing dependencies for pymorph==0.96
Finished processing dependencies for pymorph==0.96

c:\Temp\pymorph-0.96>
```

Figure 9: Instalation of the package *pymorph*.



```
Microsoft Windows [Verze 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. Všechna práva vyhrazena.

C:\Users\Ryba>z:

Z:\>cd Work\Python\MemSkel

Z:\Work\Python\MemSkel>python memSkel_main.py
```

Figure 10: Running the software MemSkel from the command line.