eartrack Documentation

Release 1.0.0

Nicolas Brichet

Oct 04, 2017

Contents

An imaging library to detect and track future position of ear on maize plants.

earTrack is released under a Cecill-C license.

CHAPTER 1

Installation

Installation

Installation with Miniconda

Miniconda installation

Follow official website instruction to install miniconda :

http://conda.pydata.org/miniconda.html

On Linux / Ubuntu / MacOS

Create virtual environment and activate it

```
conda create --name eartrack python
source activate eartrack
```

Dependencies install

```
conda install -c conda-forge numpy matplotlib opencv scikit-image
conda install -c openalea openalea.deploy openalea.core
```

(Optional) Package managing tools :

```
conda install -c conda-forge notebook nose sphinx sphinx_rtd_theme pandoc
```

Eartrack install

```
conda install -c openalea openalea.eartrack
```

On Windows

Create virtual environment and activate it

```
conda create --name eartrack python
activate eartrack
```

Dependencies install

```
conda install -c conda-forge numpy matplotlib scikit-image opencv pywin32
conda install -c openalea openalea.deploy openalea.core
```

(Optional) Package managing tools :

```
conda install -c conda-forge notebook nose sphinx sphinx_rtd_theme pandoc
```

Eartrack install

```
conda install -c openalea openalea.eartrack
```

CHAPTER 2

Notebooks Tutorial

Notebooks Tutorial

- Ear Tracking tutorial

CHAPTER 3

API References

References

Release 1.0.0

Date Oct 04, 2017

An imaging library to detect and track future position of ear on maize plants.

API Reference

The exact API of all functions and classes, as given by the docstrings. The API documents expected types and allowed features for all functions, and all parameters available for the algorithms.

binarisation

dilate(binary_image[, kshape, ksize, iterations])	Dilate an image
open(binary_image[, kshape, ksize, iterations])	Open an image
close(binary_image[, kshape, ksize, iterations])	Close an image
erode_dilate(binary_image[, kernel_shape, ...])	Applied a morphology (erode & dilate) on binary_image on mask ROI.
threshold_hsv(image, hsv_min, hsv_max[, mask])	Binarize HSV image with hsv_min and hsv_max parameters.
threshold_meanshift(image, mean_image[, ...])	Threshold pixels in numpy array such as:
mean_shift_hsv(image, mean_img[, threshold, ...])	Segmentation using mean shift method
mean_image(images)	Compute the mean of a image list.
color_tree(bgr[, cabin, mask_pot, ...])	Segmentation using decision tree and mask
decision_tree_threshold_phenoarch_1(bgr)	Implementation of a decision tree
decision_tree_threshold_phenoarch_2(bgr)	Implementation of a decision tree

openalea.eartrack.binarisation.dilate

openalea.eartrack.binarisation.**dilate**(*binary_image*, *kshape='MORPH_CROSS'*, *ksize=3*, *iterations=1*)

Dilate an image

Dilate an image using opencv dilate method :param binary_image: numpy.ndarray

2-D array

Parameters

- **kshape** – str, opt See opencv documentation
- **ksize** – int, opt See opencv documentation
- **iterations** – int, opt Number of iteration of dilatation

Returns dilated : numpy.ndarray 2-D image

openalea.eartrack.binarisation.open

openalea.eartrack.binarisation.**open**(*binary_image*, *kshape='MORPH_CROSS'*, *ksize=3*, *iterations=1*)

Open an image

Perform morphology opening algorithm on image using opencv method :param binary_image: numpy.ndarray

2-D array

Parameters

- **kshape** – str, opt See opencv documentation
- **ksize** – int, opt See opencv documentation
- **iterations** – int, opt Number of iteration

Returns opened : numpy.ndarray 2-D image

openalea.eartrack.binarisation.close

openalea.eartrack.binarisation.**close**(*binary_image*, *kshape='MORPH_CROSS'*, *ksize=3*, *iterations=1*)

Close an image

Perform morphology closing algorithm on image using opencv method :param binary_image: numpy.ndarray

2-D array

Parameters

- **kshape** – str, opt See opencv documentation
- **ksize** – int, opt See opencv documentation
- **iterations** – int, opt Number of iteration

Returns closed : numpy.ndarray 2-D image

openalea.eartrack.binarisation.erode_dilate

openalea.eartrack.binarisation.**erode_dilate**(*binary_image*, *kernel_shape=(3, 3)*, *iterations=1*, *mask=None*)

Applied a morphology (erode & dilate) on binary_image on mask ROI.

Parameters

- **binary_image** (*numpy.ndarray*) – 2-D array
- **kernel_shape** (*(N, M) of integers, optional*) – kernel shape of (erode & dilate) applied to binary_image
- **iterations** (*int, optional*) – number of successive iteration of (erode & dilate)
- **mask** (*numpy.ndarray, optional*) – Array of same shape as *image*. Only points at which mask == True will be processed.

Returns **out** – Binary Image

Return type numpy.ndarray

openalea.eartrack.binarisation.threshold_hsv

openalea.eartrack.binarisation.**threshold_hsv**(*image*, *hsv_min*, *hsv_max*, *mask=None*)

Binarize HSV image with hsv_min and hsv_max parameters. => cv2.inRange(hsv_image, hsv_min, hsv_max)

If mask is not None : => cv2.bitwise_and(binary_hsv_image, mask)

Parameters

- **image** (*numpy.ndarray of integers*) – 3-D array of image RGB
- **hsv_min** (*tuple of integers*) – HSV value of minimum range
- **hsv_max** (*tuple of integers*) – HSV value of maximum range
- **mask** (*numpy.ndarray, optional*) – Array of same shape as *image*. Only points at which mask == True will be thresholded.

Returns **out** – Thresholded binary image

Return type numpy.ndarray

See also:

threshold_meanshift()

openalea.eartrack.binarisation.threshold_meanshift

openalea.eartrack.binarisation.**threshold_meanshift**(*image*, *mean_image*, *threshold=0.3*, *mask=None*)

Threshold pixels in numpy array such as:

```
image / mean <= (1.0 - threshold)
```

If reverse is True (Inequality is reversed):

```
image / mean <= (1.0 + threshold
```

Parameters

- **image** (*numpy.ndarray of integers*) – 3-D array
- **mean_image** (*numpy.ndarray of the same shape as 'image'*) – 3-D array 'mean_image'
- **threshold** (*float, optional*) – Threshold value. Must between 0.0 and 1.0
- **reverse** (*bool, optional*) – If True reverse inequality
- **mask** (*numpy.ndarray, optional*) – Array of same shape as *image*. Only points at which mask == True will be thresholded.

Returns **out** – Thresholded binary image

Return type numpy.ndarray

See also:

`get_mean_image()`, *threshold_hsv()*

openalea.eartrack.binarisation.mean_shift_hsv

openalea.eartrack.binarisation.**mean_shift_hsv**(*image*, *mean_img*, *threshold=0.3*, *hsv_min=(30, 11, 0)*, *hsv_max=(129, 254, 141)*, *iterations_clean_noise=3*, *iterations=1*, *mask_mean_shift=None*, *mask_hsv=None*, *mask_clean_noise=None*)

Segmentation using mean shift method

Compute segmentation of an object in image using a combination of meanshift method and hsv threshold

Parameters

- **image** – numpy.ndarray of integers 3-D array
- **mean_img** – numpy.ndarray of integers (same shape as 'image') 3-D array
- **threshold** – float, optional Threshold value. Must between 0.0 and 1.0
- **hsv_min** – tuple of 3 int, optional Minimum values to threshold hsv image. Values must be between 0 and 255
- **hsv_max** – tuple of 3 int, optional Maximum values to threshold hsv image. Values must be between 0 and 255
- **iterations_clean_noise** – int, optional Number of iterations to clean noise on binary result image under mask
- **iterations** – int, optional Number of iterations to clean noise on binary result image
- **mask_mean_shift** – numpy.ndarray, optional Array 2-D of same shape as *image*. Only points at which mask == True will be calculated in meanshift method.
- **mask_hsv** – numpy.ndarray, optional Array 2-D of same shape as *image*. Only points at which mask == True will be calculated with hsv method.
- **mask_clean_noise** – numpy.ndarray, optional Array 2-D of same shape as *image*. Only points at which mask == True will be cleaned

Returns

result: numpy.ndarray 2-D of same shape as *image* Binary image representing plant segmentation of 'image'

openalea.eartrack.binarisation.mean_image

openalea.eartrack.binarisation.**mean_image**(*images*)

Compute the mean of a image list.

Parameters **images** (*[numpy.ndarray of integers]*) – list of 3-D array

Returns **out** – Mean of the list image

Return type numpy.ndarray

See also:

threshold_meanshift()

openalea.eartrack.binarisation.color_tree

openalea.eartrack.binarisation.**color_tree**(*bgr*, *cabin=None*, *mask_pot=None*, *mask_rails=None*, *empty_img=None*)

Segmentation using decision tree and mask

Platform specific method, masks and decision trees depend on imagery cabin :param bgr: numpy.ndarray of integers

3-D array

Parameters

- **cabin** – string, 2 possible values : cabin-1 or cabin-2
- **mask_pot** – mask_mean_shift: numpy.ndarray, optional Array 2-D of same shape as *bgr* representing pot position on image
- **mask_rails** – mask_mean_shift: numpy.ndarray, optional Array 2-D of same shape as *bgr* representing rails position
- **empty_img** – numpy.ndarray of integers 3-D array of empty cabin (without plant)

Returns

result [numpy.ndarray 2-D of same shape as *bgr*] Binary image representing plant segmentation of 'bgr'

openalea.eartrack.binarisation.decision_tree_threshold_phenoarch_1

openalea.eartrack.binarisation.**decision_tree_threshold_phenoarch_1**(*bgr*)

Implementation of a decision tree

Platform specific method, for top image in cabin 1 of Phenoarch :param bgr: numpy.ndarray of integers

3-D array

Returns

result [numpy.ndarray 2-D of same shape as *bgr*] Binary image representing True or False value of each pixel threw decision tree

openalea.eartrack.binarisation.decision_tree_threshold_phenoarch_2

openalea.eartrack.binarisation.**decision_tree_threshold_phenoarch_2**(*bgr*)

Implementation of a decision tree

Platform specific method, for top image in cabin 1 of Phenoarch :param bgr: numpy.ndarray of integers

3-D array

Returns

result [numpy.ndarray 2-D of same shape as *bgr*] Binary image representing True or False value of each pixel threw decision tree

eartrack

top_analysis(top_binary_img, ...)	Top image analysis
side_analysis(binary_img, color_img, angle, ...)	Side image analysis for ear tracking
get_skeleton(binary_image)	Perform skeleton on image
distance_transform(binary_image[, ...])	Perform distance transform on image
binary_biggest_region(binary_image)	Look for the biggest object on a binary image
get_endpoints(skeleton, center, height)	Look for stem extremities
skeleton_cleaning(skeleton, begin)	Clean the skeleton
find_route(skeleton, begin, end)	Perform shortest path algorithm on skeleton image
find_cross_route(skeleton, begin)	Perform shortest path algorithm on skeleton image unknowing upper node
find_cross_route(skeleton, begin)	Perform shortest path algorithm on skeleton image unknowing upper node
get_distances(route, distance_transform_img)	Get the distances transform values along a route
derivate(route)	Perform discrete derivative on a curve
differential_cleaning(diff, x, y, max_space, ...)	Clean derivatives values
differential_separate(x, y, indices)	Deep analysis of derivatives values
majors_axes_regression_ww(pixels)	Performs a major axis regression on 2D distributed dots
majors_axes_regression_line(binary_img)	Performs a major axis regression on binary image
robust_majors_axes_regression_ww(pixels)	Performs a robust major axis regression on 2D distributed dots
get_view_angles(binary_img, mask)	Extract interesting view angles from top image
robust_mean(values, images[, std_error])	Look for most representative position in a small set of positions
ear_detection(distances)	Look for ear in a stem width curve

openalea.eartrack.eartrack.top_analysis

openalea.eartrack.eartrack.**top_analysis**(*top_binary_img*, *existing_angles*, *center_mask*)

Top image analysis

Analyse top binary image to determine best side view images allowing to see the stem and find ear :param top_binary_img: (numpy array of uint8) representing binary image :param existing_angles: (list of int) list of existing angle for this snapshot :param center_mask: (numpy array of uint8) mask representing the center of image to know if a leave can be considered as obstructing :return:

(list of int) informative angles of view to analyse (numpy array of uint8) result image for log (string) log to write

openalea.eartrack.eartrack.side_analysis

openalea.eartrack.eartrack.**side_analysis**(*binary_img*, *color_img*, *angle*, *pot_height*, *pot_center*)

Side image analysis for ear tracking

Perform the analysis of side view maize plant's image to extract ear position :param binary_img: (numpy array of uint8) binary image :param color_img: (numpy array of uint8) color image in BGR matrix :param angle: (int) view angle of the image :param pot_height: (int) height position of the top of the pot :param pot_center: (int) width position of the center of the pot :return: positions: (np array of uint numpy array) Kept position(s) as

probable(s) ear(s), each position as [x, y, angle] useful_images: (np array of str) ids of images corresponding to

each position log: (string) log to write img_debug: (list of numpy array) list of output images from

different stages of calculation

openalea.eartrack.eartrack.get_skeleton

openalea.eartrack.eartrack.**get_skeleton**(*binary_image*)

Perform skeleton on image

Use skimage medial axis to perform skeleton on binary image :param binary_image: (numpy 2D array of binary uint8) binary image to perform skeleton :return: (numpy 2D array of binary uint8) binary image of skeleton

openalea.eartrack.eartrack.distance_transform

openalea.eartrack.eartrack.**distance_transform**(*binary_image*, *distance_type=1*, *mask_size=5*)

Perform distance transform on image

Perform opencv distance transform on binary image :param binary_image: (numpy 2D array of binary uint8) binary image to perform distance transorm :param distance_type: see cv::DistanceTypes :param mask_size: see cv::DistanceTransformMasks :return: (numpy 2D array of uint8) binary image transformed in distances

openalea.eartrack.eartrack.binary_biggest_region

openalea.eartrack.eartrack.**binary_biggest_region**(*binary_image*)

Look for the biggest object on a binary image

Parameters **binary_image** – (numpy 2D array of binary uint8) binary image to

analyse :return: (numpy 2D array of binary uint8) binary image containing only the biggest object

openalea.eartrack.eartrack.get_endpoints

openalea.eartrack.eartrack.**get_endpoints**(*skeleton*, *center*, *height*)

Look for stem extremities

Try to find the bottom and upper node of the stem in a maize plant :param skeleton: (numpy 2D array of binary uint8) representing the skeleton of side view image of a maize plant :param center: (int) pixel in the width center of the pot (depending on the plateform and the calibration) :param height: (int) pixel in the height top of the pot (depending on the plateform and the calibration) :return: (list of 2 int) pixel of the bottom of the stem

(list of 2 int) pixel of the top of the stem

openalea.eartrack.eartrack.skeleton_cleaning

openalea.eartrack.eartrack.**skeleton_cleaning**(*skeleton*, *begin*)

Clean the skeleton

Parameters **skeleton** – (numpy 2D array of binary uint8) representing the skeleton

of side view image of maize plant :param begin: bottm of stem :return: (numpy 2D array of binary uint8) representing cleaned skeleton

openalea.eartrack.eartrack.find_route

openalea.eartrack.eartrack.**find_route**(*skeleton*, *begin*, *end*)

Perform shortest path algorithm on skeleton image

Find the shortest route on a skeleton between 2 pixels using graph shortest path algorithm :param skeleton: (numpy 2D array of binary uint8) representing the skeleton of side view image of a maize plant :param begin: (list of 2 int) pixel of the bottom of the stem :param end: (list of 2 int) pixel of the top of the stem :return: (list of list of 2 int) list of all the pixels to follow to get the shortest path between begin and end

openalea.eartrack.eartrack.find_cross_route

openalea.eartrack.eartrack.**find_cross_route**(*skeleton*, *begin*)

Perform shortest path algorithm on skeleton image unknowing upper node

Find the shortest route on a skeleton between a beginning pixel and the upper cross on the skeleton using graph shortest path algorithm :param skeleton: (numpy 2D array of binary uint8) representing the skeleton of side view image of a maize plant :param begin: (list of 2 int) pixel of the bottom of the stem :return: (list of list of 2 int) list of all the pixels to follow to get the shortest path between begin and upper cross

openalea.eartrack.eartrack.get_distances

openalea.eartrack.eartrack.**get_distances**(*route*, *distance_transform_img*)

Get the distances transform values along a route

'route' are coordinates in the 'distance_transform_img' shape. :param route: (list of list of 2 int) list of all the pixels to follow a route on image :param distance_transform_img: (numpy 2D array of uint8) binary image transformed in distances :return: (list of int) representing the distances values all along the route

openalea.eartrack.eartrack.derivate

openalea.eartrack.eartrack.**derivate**(*route*)

Perform discrete derivative on a curve

Perform discrete derivative on a route in order to analyse variation of directions :param route: (list of list of 2 int) list of all the pixels to follow a route on image :return: diff: (list of int) values in [-1, 0, 1] representing the variation of the route

x: (list of int) x original position of each diff value y: (list of int) y original position of each diff value

openalea.eartrack.eartrack.differential_cleaning

openalea.eartrack.eartrack.**differential_cleaning**(*diff*, *x*, *y*, *max_space*, *min_length*, *min_height*)

Clean derivatives values

Analyse derivatives values to keep only the significant variations :param diff: (list of int) values in [-1, 0, 1] representing the variation of a route :param x: (list of int) x original position of each diff value :param y: (list of int) y original position of each diff value :param max_space: (int) max length (in pixels) of diff null to reckon that

the increase or decrease is no longer the same variation

Parameters **min_length** – (int) minimum length of variation to reckon that the

variation is significant :param min_height: minimum height of variation to reckon that the variation is significant :return: (list of 3 int list) describing the diff values by parts of same variation [[begin, end, variation]]

openalea.eartrack.eartrack.differential_separate

openalea.eartrack.eartrack.**differential_separate**(*x*, *y*, *indices*)

Deep analysis of derivatives values

Go deeper in derivatives values analyse to find different fast of increase and decrease in order to detect increases and decreases even on inclined stem :param x: (list of int) x original position of each diff value :param y: (list of int) y original position of each diff value :param indices: (list of 3 int list) describing the differentials values by

parts of same variation [[begin, end, variation]]

Returns new_indexes : (list of 3 int list) describing new variations total_means : (list of float) slope of each part of 'new_indexes'

openalea.eartrack.eartrack.majors_axes_regression_ww

openalea.eartrack.eartrack.**majors_axes_regression_ww**(*pixels*)

Performs a major axis regression on 2D distributed dots

Parameters **pixels** – (np array of 2 np array of int) distributed dots to perform

regression :return: a: (float) slope of regression line

b: (float) intercept of regression line mean_error: (float) mean error of dots to regression line

openalea.eartrack.eartrack.majors_axes_regression_line

openalea.eartrack.eartrack.**majors_axes_regression_line**(*binary_img*)

Performs a major axis regression on binary image

True pixels of image are used as distributed dots :param binary_img: (numpy 2D binary uint8 array) binary image to perform regression :return: result: (numpy 3D uint8 array) color image with regression line draws on it

a: (float) slope of regression line b: (float) intercept of regression line mean_error: (float) mean error of pixels to regression line alpha: angle of regression line (in degrees)

openalea.eartrack.eartrack.robust_majors_axes_regression_ww

openalea.eartrack.eartrack.**robust_majors_axes_regression_ww**(*pixels*)

Performs a robust major axis regression on 2D distributed dots

Robustness come from 'hinich et al.' algorithm :param pixels: (np array of 2 np array of int) distributed dots to perform regression :return: a: (float) slope of robust regression line

b: (float) intercept of robust regression line useful_pixels: (np array of 2 np array of int) dots kept by robust

regression useless_pixels: (np array of 2 np array of int) dots ousted by

robust regression

openalea.eartrack.eartrack.get_view_angles

openalea.eartrack.eartrack.**get_view_angles**(*binary_img*, *mask*)

Extract interesting view angles from top image

Parameters

- **binary_img** – (numpy array of uint8) representing binary image
- **mask** – (numpy array of uint8) mask representing the center of

image to know if a leave can be considered as obstructing :return:

(list of int) informative angles of view to analyse (numpy array of uint8) result image for log (string) log to write

openalea.eartrack.eartrack.robust_mean

openalea.eartrack.eartrack.**robust_mean**(*values*, *images*, *std_error=20*)

Look for most representative position in a small set of positions

This function perform a 'vote' between few values to extract the most representative(s) and the corresponding images :param values: (2 dimensional numpy float array) the vote will be perform on first value of each 2 values array :param images: (numpy array of string) id of image corresponding to each value :param std_error: (int) maximum standard error to reckon that 2 values are in the same group :return: means: (2 values numpy array) mean value of kept 2 values array

((-1, -1) if standard error remains more than std_error param) values: (2 dimensional numpy float array) kept values as most

representatives images: (numpy array of string) id of image corresponding to each

kept value

openalea.eartrack.eartrack.ear_detection

openalea.eartrack.eartrack.**ear_detection**(*distances*)

Look for ear in a stem width curve

Parameters **distances** – (list of int) representing distance transform values all

along the stem :return: (list of list of 2 int) first value of each 2 int list is a

probable solution, second value is its weight

(list of (list of (2 int and one list))) representing parts of distances interpreted as stem (begin, end, [values]) (list of (list of (2 int and one list))) representing parts of distances interpreted as leaves (begin, end, [values]) (list of 2 int), width of stem under ear and upper ear

CHAPTER 4

Authors

Authors

- Nicolas Brichet <brichet@supagro.inra.fr>
- Nicolas Brichet <brichet@supagro.inra.fr>

CHAPTER 5

License

License

earTrack is released under a Cecill-C license.

CeCILL-C FREE SOFTWARE LICENSE AGREEMENT

Notice

This Agreement is a Free Software license agreement that is the result of discussions between its authors in order to ensure compliance with the two main principles guiding its drafting:

- firstly, compliance with the principles governing the distribution of Free Software: access to source code, broad rights granted to users,
- secondly, the election of a governing law, French law, with which it is conformant, both as regards the law of torts and intellectual property law, and the protection that it offers to both authors and holders of the economic rights over software.

The authors of the CeCILL-C (for Ce[a] C[nrs] I[nria] L[ogiciel] L[ibre]) license are:

Commissariat à l'Energie Atomique - CEA, a public scientific, technical and industrial research establishment, having its principal place of business at 25 rue Leblanc, immeuble Le Ponant D, 75015 Paris, France.

Centre National de la Recherche Scientifique - CNRS, a public scientific and technological establishment, having its principal place of business at 3 rue Michel-Ange, 75794 Paris cedex 16, France.

Institut National de Recherche en Informatique et en Automatique - INRIA, a public scientific and technological establishment, having its principal place of business at Domaine de Voluceau, Rocquencourt, BP 105, 78153 Le Chesnay cedex, France.

Preamble

The purpose of this Free Software license agreement is to grant users the right to modify and re-use the software governed by this license.

The exercising of this right is conditional upon the obligation to make available to the community the modifications made to the source code of the software so as to contribute to its evolution.

In consideration of access to the source code and the rights to copy, modify and redistribute granted by the license, users are provided only with a limited warranty and the software's author, the holder of the economic rights, and the successive licensors only have limited liability.

In this respect, the risks associated with loading, using, modifying and/or developing or reproducing the software by the user are brought to the user's attention, given its Free Software status, which may make it complicated to use, with the result that its use is reserved for developers and experienced professionals having in-depth computer knowledge. Users are therefore encouraged to load and test the suitability of the software as regards their requirements in conditions enabling the security of their systems and/or data to be ensured and, more generally, to use and operate it in the same conditions of security. This Agreement may be freely reproduced and published, provided it is not altered, and that no provisions are either added or removed herefrom.

This Agreement may apply to any or all software for which the holder of the economic rights decides to submit the use thereof to its provisions.

Article 1 - DEFINITIONS

For the purpose of this Agreement, when the following expressions commence with a capital letter, they shall have the following meaning:

Agreement: means this license agreement, and its possible subsequent versions and annexes.

Software: means the software in its Object Code and/or Source Code form and, where applicable, its documentation, "as is" when the Licensee accepts the Agreement.

Initial Software: means the Software in its Source Code and possibly its Object Code form and, where applicable, its documentation, "as is" when it is first distributed under the terms and conditions of the Agreement.

Modified Software: means the Software modified by at least one Integrated Contribution.

Source Code: means all the Software's instructions and program lines to which access is required so as to modify the Software.

Object Code: means the binary files originating from the compilation of the Source Code.

Holder: means the holder(s) of the economic rights over the Initial Software.

Licensee: means the Software user(s) having accepted the Agreement.

Contributor: means a Licensee having made at least one Integrated Contribution.

Licensor: means the Holder, or any other individual or legal entity, who distributes the Software under the Agreement.

Integrated Contribution: means any or all modifications, corrections, translations, adaptations and/or new functions integrated into the Source Code by any or all Contributors.

Related Module: means a set of sources files including their documentation that, without modification to the Source Code, enables supplementary functions or services in addition to those offered by the Software.

Derivative Software: means any combination of the Software, modified or not, and of a Related Module.

Parties: mean both the Licensee and the Licensor.

These expressions may be used both in singular and plural form.

Article 2 - PURPOSE

The purpose of the Agreement is the grant by the Licensor to the Licensee of a non-exclusive, transferable and world-wide license for the Software as set forth in Article 5 hereinafter for the whole term of the protection granted by the rights over said Software.

Article 3 - ACCEPTANCE

3.1 The Licensee shall be deemed as having accepted the terms and conditions of this Agreement upon the occurrence of the first of the following events:

- (i) loading the Software by any or all means, notably, by downloading from a remote server, or by loading from a physical medium;
- (ii) the first time the Licensee exercises any of the rights granted hereunder.

3.2 One copy of the Agreement, containing a notice relating to the characteristics of the Software, to the limited warranty, and to the fact that its use is restricted to experienced users has been provided to the Licensee prior to its acceptance as set forth in Article 3.1 hereinabove, and the Licensee hereby acknowledges that it has read and understood it.

Article 4 - EFFECTIVE DATE AND TERM

4.1 EFFECTIVE DATE

The Agreement shall become effective on the date when it is accepted by the Licensee as set forth in Article 3.1.

4.2 TERM

The Agreement shall remain in force for the entire legal term of protection of the economic rights over the Software.

Article 5 - SCOPE OF RIGHTS GRANTED

The Licensor hereby grants to the Licensee, who accepts, the following rights over the Software for any or all use, and for the term of the Agreement, on the basis of the terms and conditions set forth hereinafter.

Besides, if the Licensor owns or comes to own one or more patents protecting all or part of the functions of the Software or of its components, the Licensor undertakes not to enforce the rights granted by these patents against successive Licensees using, exploiting or modifying the Software. If these patents are transferred, the Licensor undertakes to have the transferees subscribe to the obligations set forth in this paragraph.

5.1 RIGHT OF USE

The Licensee is authorized to use the Software, without any limitation as to its fields of application, with it being hereinafter specified that this comprises:

1. permanent or temporary reproduction of all or part of the Software by any or all means and in any or all form.
2. loading, displaying, running, or storing the Software on any or all medium.
3. entitlement to observe, study or test its operation so as to determine the ideas and principles behind any or all constituent elements of said Software. This shall apply when the Licensee carries out any or all loading, displaying, running, transmission or storage operation as regards the Software, that it is entitled to carry out hereunder.

5.2 RIGHT OF MODIFICATION

The right of modification includes the right to translate, adapt, arrange, or make any or all modifications to the Software, and the right to reproduce the resulting software. It includes, in particular, the right to create a Derivative Software.

The Licensee is authorized to make any or all modification to the Software provided that it includes an explicit notice that it is the author of said modification and indicates the date of the creation thereof.

5.3 RIGHT OF DISTRIBUTION

In particular, the right of distribution includes the right to publish, transmit and communicate the Software to the general public on any or all medium, and by any or all means, and the right to market, either in consideration of a fee, or free of charge, one or more copies of the Software by any means.

The Licensee is further authorized to distribute copies of the modified or unmodified Software to third parties according to the terms and conditions set forth hereinafter.

5.3.1 DISTRIBUTION OF SOFTWARE WITHOUT MODIFICATION

The Licensee is authorized to distribute true copies of the Software in Source Code or Object Code form, provided that said distribution complies with all the provisions of the Agreement and is accompanied by:

1. a copy of the Agreement,
2. a notice relating to the limitation of both the Licensor's warranty and liability as set forth in Articles 8 and 9,

and that, in the event that only the Object Code of the Software is redistributed, the Licensee allows effective access to the full Source Code of the Software at a minimum during the entire period of its distribution of the Software, it being understood that the additional cost of acquiring the Source Code shall not exceed the cost of transferring the data.

5.3.2 DISTRIBUTION OF MODIFIED SOFTWARE

When the Licensee makes an Integrated Contribution to the Software, the terms and conditions for the distribution of the resulting Modified Software become subject to all the provisions of this Agreement.

The Licensee is authorized to distribute the Modified Software, in source code or object code form, provided that said distribution complies with all the provisions of the Agreement and is accompanied by:

1. a copy of the Agreement,
2. a notice relating to the limitation of both the Licensor's warranty and liability as set forth in Articles 8 and 9,

and that, in the event that only the object code of the Modified Software is redistributed, the Licensee allows effective access to the full source code of the Modified Software at a minimum during the entire period of its distribution of the Modified Software, it being understood that the additional cost of acquiring the source code shall not exceed the cost of transferring the data.

5.3.3 DISTRIBUTION OF DERIVATIVE SOFTWARE

When the Licensee creates Derivative Software, this Derivative Software may be distributed under a license agreement other than this Agreement, subject to compliance with the requirement to include a notice concerning the rights over the Software as defined in Article 6.4. In the event the creation of the Derivative Software required modification of the Source Code, the Licensee undertakes that:

1. the resulting Modified Software will be governed by this Agreement,
2. the Integrated Contributions in the resulting Modified Software will be clearly identified and documented,
3. the Licensee will allow effective access to the source code of the Modified Software, at a minimum during the entire period of distribution of the Derivative Software, such that such modifications may be carried over in a subsequent version of the Software; it being understood that the additional cost of purchasing the source code of the Modified Software shall not exceed the cost of transferring the data.

5.3.4 COMPATIBILITY WITH THE CeCILL LICENSE

When a Modified Software contains an Integrated Contribution subject to the CeCILL license agreement, or when a Derivative Software contains a Related Module subject to the CeCILL license agreement, the provisions set forth in the third item of Article 6.4 are optional.

Article 6 - INTELLECTUAL PROPERTY

6.1 OVER THE INITIAL SOFTWARE

The Holder owns the economic rights over the Initial Software. Any or all use of the Initial Software is subject to compliance with the terms and conditions under which the Holder has elected to distribute its work and no one shall be entitled to modify the terms and conditions for the distribution of said Initial Software.

The Holder undertakes that the Initial Software will remain ruled at least by this Agreement, for the duration set forth in Article 4.2.

6.2 OVER THE INTEGRATED CONTRIBUTIONS

The Licensee who develops an Integrated Contribution is the owner of the intellectual property rights over this Contribution as defined by applicable law.

6.3 OVER THE RELATED MODULES

The Licensee who develops a Related Module is the owner of the intellectual property rights over this Related Module as defined by applicable law and is free to choose the type of agreement that shall govern its distribution under the conditions defined in Article 5.3.3.

6.4 NOTICE OF RIGHTS

The Licensee expressly undertakes:

1. not to remove, or modify, in any manner, the intellectual property notices attached to the Software;
2. to reproduce said notices, in an identical manner, in the copies of the Software modified or not;
3. to ensure that use of the Software, its intellectual property notices and the fact that it is governed by the Agreement is indicated in a text that is easily accessible, specifically from the interface of any Derivative Software.

The Licensee undertakes not to directly or indirectly infringe the intellectual property rights of the Holder and/or Contributors on the Software and to take, where applicable, vis-à-vis its staff, any and all measures required to ensure respect of said intellectual property rights of the Holder and/or Contributors.

Article 7 - RELATED SERVICES

7.1 Under no circumstances shall the Agreement oblige the Licensor to provide technical assistance or maintenance services for the Software.

However, the Licensor is entitled to offer this type of services. The terms and conditions of such technical assistance, and/or such maintenance, shall be set forth in a separate instrument. Only the Licensor offering said maintenance and/or technical assistance services shall incur liability therefor.

7.2 Similarly, any Licensor is entitled to offer to its licensees, under its sole responsibility, a warranty, that shall only be binding upon itself, for the redistribution of the Software and/or the Modified Software, under terms and conditions that it is free to decide. Said warranty, and the financial terms and conditions of its application, shall be subject of a separate instrument executed between the Licensor and the Licensee.

Article 8 - LIABILITY

8.1 Subject to the provisions of Article 8.2, the Licensee shall be entitled to claim compensation for any direct loss it may have suffered from the Software as a result of a fault on the part of the relevant Licensor, subject to providing evidence thereof.

8.2 The Licensor's liability is limited to the commitments made under this Agreement and shall not be incurred as a result of in particular: (i) loss due the Licensee's total or partial failure to fulfill its obligations, (ii) direct or consequential loss that is suffered by the Licensee due to the use or performance of the Software, and (iii) more generally, any consequential loss. In particular the Parties expressly agree that any or all pecuniary or business loss (i.e. loss of data, loss of profits, operating loss, loss of customers or orders, opportunity cost, any disturbance to business activities) or any or all legal proceedings instituted against the Licensee by a third party, shall constitute consequential loss and shall not provide entitlement to any or all compensation from the Licensor.

Article 9 - WARRANTY

9.1 The Licensee acknowledges that the scientific and technical state-of-the-art when the Software was distributed did not enable all possible uses to be tested and verified, nor for the presence of possible defects to be detected. In this respect, the Licensee's attention has been drawn to the risks associated with loading, using, modifying and/or developing and reproducing the Software which are reserved for experienced users.

The Licensee shall be responsible for verifying, by any or all means, the suitability of the product for its requirements, its good working order, and for ensuring that it shall not cause damage to either persons or properties.

9.2 The Licensor hereby represents, in good faith, that it is entitled to grant all the rights over the Software (including in particular the rights set forth in Article 5).

9.3 The Licensee acknowledges that the Software is supplied "as is" by the Licensor without any other express or tacit warranty, other than that provided for in Article 9.2 and, in particular, without any warranty as to its commercial value, its secured, safe, innovative or relevant nature.

Specifically, the Licensor does not warrant that the Software is free from any error, that it will operate without interruption, that it will be compatible with the Licensee's own equipment and software configuration, nor that it will meet the Licensee's requirements.

9.4 The Licensor does not either expressly or tacitly warrant that the Software does not infringe any third party intellectual property right relating to a patent, software or any other property right. Therefore, the Licensor disclaims any and all liability towards the Licensee arising out of any or all proceedings for infringement that may be instituted in respect of the use, modification and redistribution of the Software. Nevertheless, should such proceedings be instituted against the Licensee, the Licensor shall provide it with technical and legal assistance for its defense. Such technical and legal assistance shall be decided on a case-by-case basis between the relevant Licensor and the Licensee pursuant to a memorandum of understanding. The Licensor disclaims any and all liability as regards the Licensee's use of the name of the Software. No warranty is given as regards the existence of prior rights over the name of the Software or as regards the existence of a trademark.

Article 10 - TERMINATION

10.1 In the event of a breach by the Licensee of its obligations hereunder, the Licensor may automatically terminate this Agreement thirty (30) days after notice has been sent to the Licensee and has remained ineffective.

10.2 A Licensee whose Agreement is terminated shall no longer be authorized to use, modify or distribute the Software. However, any licenses that it may have granted prior to termination of the Agreement shall remain valid subject to their having been granted in compliance with the terms and conditions hereof.

Article 11 - MISCELLANEOUS

11.1 EXCUSABLE EVENTS

Neither Party shall be liable for any or all delay, or failure to perform the Agreement, that may be attributable to an event of force majeure, an act of God or an outside cause, such as defective functioning or interruptions of the electricity or telecommunications networks, network paralysis following a virus attack, intervention by government authorities, natural disasters, water damage, earthquakes, fire, explosions, strikes and labor unrest, war, etc.

11.2 Any failure by either Party, on one or more occasions, to invoke one or more of the provisions hereof, shall under no circumstances be interpreted as being a waiver by the interested Party of its right to invoke said provision(s) subsequently.

11.3 The Agreement cancels and replaces any or all previous agreements, whether written or oral, between the Parties and having the same purpose, and constitutes the entirety of the agreement between said Parties concerning said purpose. No supplement or modification to the terms and conditions hereof shall be effective as between the Parties unless it is made in writing and signed by their duly authorized representatives.

11.4 In the event that one or more of the provisions hereof were to conflict with a current or future applicable act or legislative text, said act or legislative text shall prevail, and the Parties shall make the necessary amendments so as to comply with said act or legislative text. All other provisions shall remain effective. Similarly, invalidity of a provision of the Agreement, for any reason whatsoever, shall not cause the Agreement as a whole to be invalid.

11.5 LANGUAGE

The Agreement is drafted in both French and English and both versions are deemed authentic.

Article 12 - NEW VERSIONS OF THE AGREEMENT

12.1 Any person is authorized to duplicate and distribute copies of this Agreement.

12.2 So as to ensure coherence, the wording of this Agreement is protected and may only be modified by the authors of the License, who reserve the right to periodically publish updates or new versions of the Agreement, each with a separate number. These subsequent versions may address new issues encountered by Free Software.

12.3 Any Software distributed under a given version of the Agreement may only be subsequently distributed under the same version of the Agreement or a subsequent version.

Article 13 - GOVERNING LAW AND JURISDICTION

13.1 The Agreement is governed by French law. The Parties agree to endeavor to seek an amicable solution to any disagreements or disputes that may arise during the performance of the Agreement.

13.2 Failing an amicable solution within two (2) months as from their occurrence, and unless emergency proceedings are necessary, the disagreements or disputes shall be referred to the Paris Courts having jurisdiction, by the more diligent Party.

Version 1.0 dated 2006-09-05.

CHAPTER 6

Indices and tables

- genindex
- search

Index