Benchmark Smil vs Scikit-Image Mathematical Morphology features

José Marcio Martins da Cruz
Samy Blusseau

November 15, 2021

https://smil.cmm.minesparis.psl.eu
jose-marcio.martins@minesparis.psl.eu
samy.blusseau@minesparis.psl.eu

Plan

- Introduction
- Benchmarking - what and how
- Speed-UP results (per image and structuring element size)
- Resources usage (CPU and memory)
- Typical elapsed times
- Conclusion

Useful links to follow the presentation :

- This presentation at : https://tinyurl.com/smil-gdmm

Part I

Introduction

Who ...

- **scikit-image** - widely known general purpose image processing library
  `https://scikit-image.org`
- **Smil** - mathematical morphology based image processing library
  `https://smil.cmm.minesparis.psl.eu`

Benchmark

- compare execution times and resources usage;
- use a small, but representative, subset of morphological library functions and images;
- full results and discussion can be found at :
  `https://smil.cmm.minesparis.psl.eu/smil-vs-skimage`
- this document at :
  `https://tinyurl.com/smil-gdmm`

Smil vs scikit-image

Comparing libraries... (1)

Some differences

- **What's the goal of these libraries ?**
  - skimage : general purpose image handling;
  - Smil : mathematical morphology image handling.
- **Language**
  - skimage : Python and C (Cython)
  - Smil : C++, SWIG based Python interface
- **Morphological features** - Smil has a much richer set of morphological functions already included : granulometry, geodesy, line morphology, ... and continuously including results from our research and other state of the art algorithms.

Smil vs scikit-image

Comparing libraries... (2)

Some differences

- **Data types** :
  - skimage : Numpy based data types
  - Smil : internally usual C++ data types
- **Image data types**
  - skimage : boolean, signed and unsigned integer and float ([-1,1])
  - Smil : UINT8, UINT16 and UINT32.
    Binary images are represented with only two values in these data types
- **Image grids**
  - skimage : square grids
  - Smil : square and hexagonal grids
- **Structuring elements** - this point is important as it impacts performance
  - skimage : a Numpy array
  - Smil : a list of points

More details at :

- `https://smil.cmm.minesparis.psl.eu/smil-vs-skimage/images.html`

Part II

Benchmarking - what and how

Evaluate

- Elapsed times and SpeedUp

$$SpeedUp = \frac{Elapsed\ time\ on\ skImage}{Elapsed\ time\ on\ Smil}$$

- SpeedUp dependency on sizes :
  - image size ranging from 256x256 to 8192x8192 (multiplying by 2 at each step);
  - structuring element from 1 to 8.
- Resources : memory and CPU usage

More details at :

- `https://smil.cmm.minesparis.psl.eu/smil-vs-skimage/protocol.html`

Smil vs scikit-image

Which functions

Binary image functions

	Smil	skImage
erosion	`erode()`	`erosion()`
opening	`open()`	`opening()`
area Threshold	`areaThreshold()`	`label(); remove_small_objects()`
euclidean distance	`distanceEuclidean()`	`distance_transform_edt()`
labeling	`label()`	`label()`
thinning	`fullThin()`	`thin()`
segmentation (application)	`watershed() + ...`	`watershed() + ...`

Gray level image functions

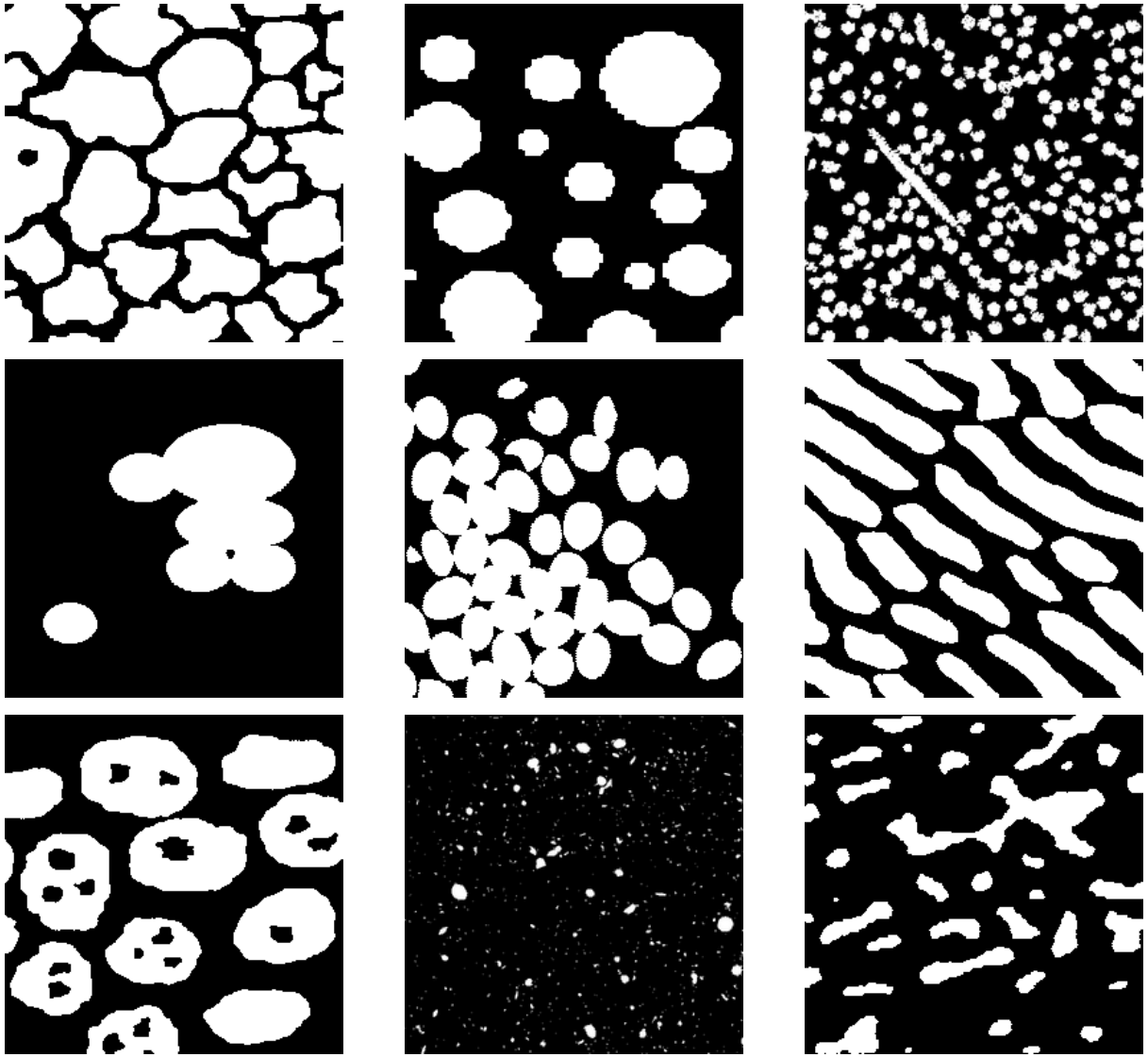
	Smil	skImage
erosion	`erode()`	`erosion()`
opening	`open()`	`opening()`
top hat	`topHat()`	`white_tophat()`
hMinima	`hMinima`	`h_minima()`
areaOpening	`areaOpen()`	`area_opening()`
segmentation (application)	`watershed() + ...`	`watershed() + ...`

More details at :

- `https://smil.cmm.minesparis.psl.eu/smil-vs-skimage/functions.html`

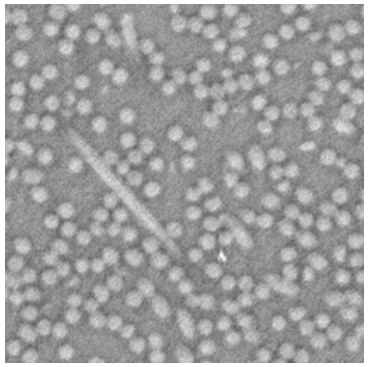
Smil vs scikit-image

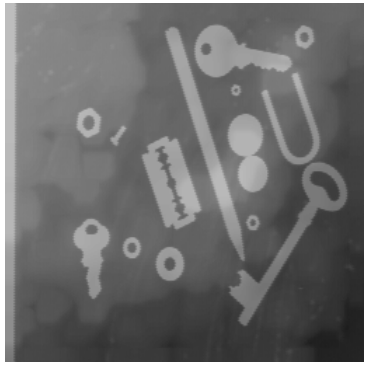
Binary test images



Smil vs scikit-image

Gray level test images

More details at :

- `https://smil.cmm.minesparis.psl.eu/smil-vs-skimage/images.html`

Smil vs scikit-image

Test platforms

- **taurus** - server
  - Dell Poweredge R540
  - CPU : 2 X Intel(R) Xeon(R) Gold 5118 CPU @ 2.30GHz
  - 12 cores - 2 threads - Total 48 VCPUs
  - Memory : 384 GiB
  - Linux Ubuntu 18.04
  - Smil Python version : 1.0.0-dev
  - scikit-image version : 0.18.1
  - Python 3.8.11 (conda 4.10.3)
- **nestor** - desktop computer
  - Dell OptiPlex 7010
  - 1 x Intel(R) Core(TM) i7-3770 CPU @ 3.40GHz
  - 4 cores - 2 threads - Total 8 VCPUs
  - Memory : 16 GiB
  - Linux Ubuntu 20.04
  - Python 3.8.10
  - Smil Python version : 1.0.0-dev
  - scikit-image version : 0.18.2

Part III

Speedup results

Smil vs scikit-image

Image size SpeedUp : some details...

SpeedUp dependency on image size

- image size ranging from 256x256 to 8192x8192 (multiplying by 2 at each step);
- structuring element : 4-connexity of fixed size `1` : **`CrossSE(1)`** for Smil and **`diamond(1)`** for skimage.
- images are scaled up (or down) to obtain different image sizes (multiplying by 2 at each step);
- Python **`timeit`** module : one round to determine the length of each round followed by 7 measure rounds.

More details at :

- `https://smil.cmm.minesparis.psl.eu/smil-vs-skimage/protocol.html`

Smil vs scikit-image

Image size Speedup results : nestor (desktop computer)

Binary Images

	alumine		balls		bubbles_bin	
Function	**512x512**	**8192x8192**	**512x512**	**8192x8192**	**512x512**	**8192x8192**
erode	97.50	83.30	97.50	83.50	106.00	83.70
open	108.00	101.00	108.00	101.00	115.00	101.00
distance	2.73	2.97	3.13	3.30	3.04	3.31
segmentation	2.93	0.90	0.88	0.86	7.62	1.39
label	0.41	0.41	0.51	0.57	0.55	0.69
fastLabel	0.54	1.15	0.43	0.87	0.29	0.78
areaThreshold	0.53	0.64	0.67	0.84	0.63	0.95
thinning	5.48	5.56	5.38	5.54	5.37	5.99

Gray Images

	astronaut		bubbles_gray		hubble_EDF_gray	
Function	**512x512**	**8192x8192**	**512x512**	**8192x8192**	**512x512**	**8192x8192**
erode	153.00	83.40	165.00	83.20	164.00	83.40
open	154.00	100.00	167.00	101.00	164.00	101.00
tophat	90.30	55.10	100.00	55.90	96.00	55.40
gradient	336.00	290.00	320.00	289.00	339.00	294.00
hMinima	1.70	3.05	1.84	3.03	1.67	3.96
watershed	3.11	6.48	4.15	7.08	3.48	5.89
segmentation	2.46	3.19	2.56	3.29	2.06	2.83
areaOpen	9.72	48.80	12.40	140.00	12.50	347.00

Full results at :

- https://smil.cmm.minesparis.psl.eu/smil-vs-skimage/nestor-summary.html
- https://smil.cmm.minesparis.psl.eu/smil-vs-skimage/nestor-bin.html
- https://smil.cmm.minesparis.psl.eu/smil-vs-skimage/nestor-gray.html

1911022

Smil vs scikit-image

Image size Speedup results : taurus (server)

Binary Images

	alumine		balls		bubbles_bin	
Function	**512x512**	**8192x8192**	**512x512**	**8192x8192**	**512x512**	**8192x8192**
erode	53.10	612.00	51.30	562.00	50.80	530.00
open	54.50	694.00	61.30	707.00	53.60	719.00
distance	2.25	3.11	2.53	3.38	2.45	3.14
segmentation	1.79	1.14	0.64	1.23	4.25	1.28
label	0.28	0.47	0.36	0.69	0.36	0.79
fastLabel	0.45	1.29	0.36	1.05	0.21	0.93
areaThreshold	0.39	0.55	0.45	0.75	0.43	0.90
thinning	4.29	10.10	4.20	9.81	4.95	10.80

Gray Images

	astronaut		bubbles_gray		hubble_EDF_gray	
Function	**512x512**	**8192x8192**	**512x512**	**8192x8192**	**512x512**	**8192x8192**
erode	48.40	607.00	49.90	622.00	49.90	604.00
open	55.60	707.00	63.40	719.00	54.30	564.00
tophat	30.30	177.00	32.60	179.00	32.10	176.00
gradient	195.00	742.00	181.00	722.00	235.00	788.00
hMinima	1.22	2.49	1.42	2.46	1.23	3.25
watershed	2.10	5.13	2.68	5.84	2.12	5.00
segmentation	1.54	3.09	1.56	3.13	1.27	2.77
areaOpen	5.68	44.30	8.09	115.00	8.83	340.00

Full results at :

- https://smil.cmm.minesparis.psl.eu/smil-vs-skimage/taurus-summary.html
- https://smil.cmm.minesparis.psl.eu/smil-vs-skimage/taurus-bin.html
- https://smil.cmm.minesparis.psl.eu/smil-vs-skimage/taurus-gray.html

Smil vs scikit-image

Explaining...

Why so high SpeedUps ?

- At processor level - Intel Intrinsics - *SIMD* - same instruction over 128 bits wide registers (16 *8-bits* pixels);
- At system level - *OpenMP* threads - processing distributed over all available cores;
- At library level - structuring elements representation - Numpy arrays (square complexity) vs list of neighbors (linear complexity);
- ... + programming ??

Why Smil labelling is slower ?

- Smil uses a generic structuring element;
- skimage uses a **`connectivity`** parameter defined as the number of unit size ortogonal jumps. This parameter allows use of an optimized labeling algorithm but it's incompatible with generic SE and hexagonal grids.
- Kensheng Wu, Ekow Otoo and Arie Shoshani, *Optimizing connected component labeling algorithms*, Lawrence Berkeley National Laboratory, http://repositories.cdlib.org/lbnl/LBNL-56864

SpeedUp dependency on structuring element size

- image with fixed size : 2048x2048;
- structuring element : 4-connexity (**`CrossSE()`** for Smil and **`diamond()`** for skimage), with size ranging in the interval **`[1,8]`**
- Python **`timeit`** module : one round to determine the length of each round followed by 7 measure rounds.

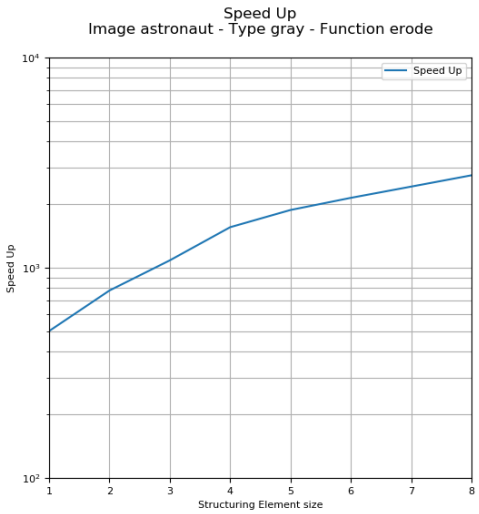
More details at :

- `https://smil.cmm.minesparis.psl.eu/smil-vs-skimage/protocol.html`

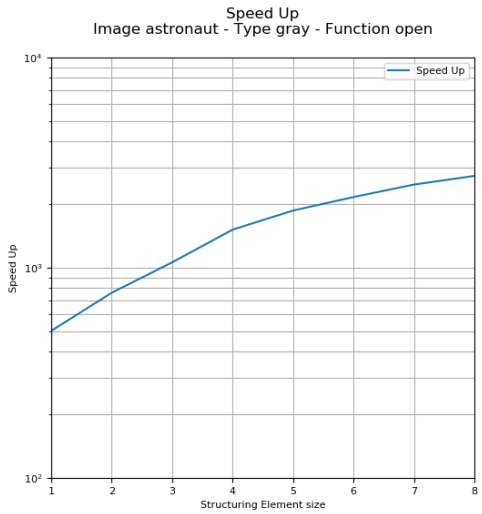
Smil vs scikit-image

Structuring element size results : taurus (server)

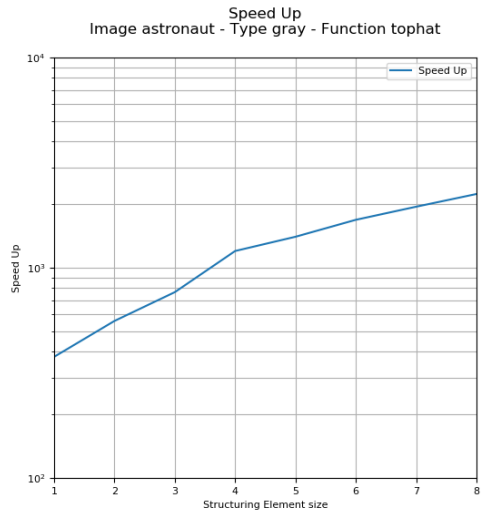
Image : astronaut.png - Computer : taurus



erode()



open()



tophat()

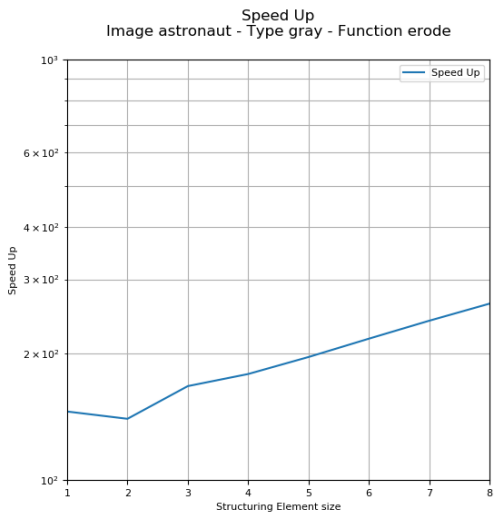
Full results at :

- https://smil.cmm.minesparis.psl.eu/smil-vs-skimage/taurus-summary-se.html
- https://smil.cmm.minesparis.psl.eu/smil-vs-skimage/taurus-bin.html
- https://smil.cmm.minesparis.psl.eu/smil-vs-skimage/taurus-gray.html

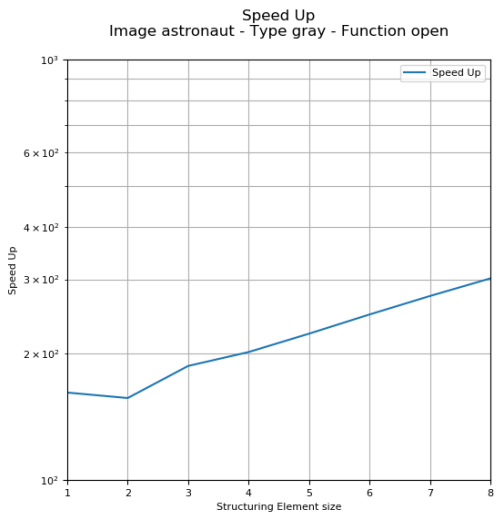
Smil vs scikit-image

Structuring element size results : nestor (desktop computer)

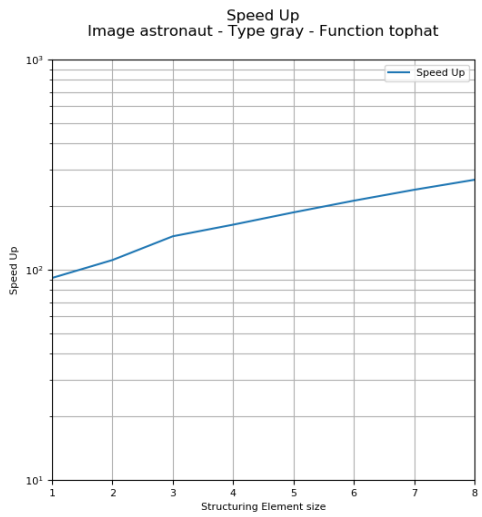
Image : astronaut.png - Computer : nestor



erode()



open()



tophat()

Full results at :

- `https://smil.cmm.minesparis.psl.eu/smil-vs-skimage/nestor-summary-se.html`
- `https://smil.cmm.minesparis.psl.eu/smil-vs-skimage/nestor-bin.html`
- `https://smil.cmm.minesparis.psl.eu/smil-vs-skimage/nestor-gray.html`

Smil vs scikit-image

Explaining Structuring Element SpeedUp...

Computational Complexity (pixel accesses) function of StrElt size

	Data structure	*Steps* (*pixels*)	$Complexity(Size_{SE})$
skImage	Numpy array	$((2 \cdot Size_{SE} + 1)^2 + 1) \cdot Size_{Image}$	$\mathcal{O}(Size_{SE}^2)$
Smil	List of points	$Size_{SE} \cdot (Length_{SE} + 1) \cdot Size_{Image}$	$\mathcal{O}(Size_{SE})$

Part IV

Resources usage

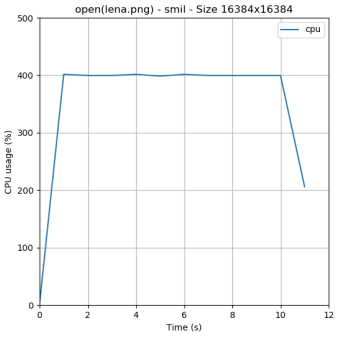
What ?

- 16384x16384 gray image : a mosaic of 64 256x256 images (lena.png)
- functions tested :
  - `open() vs opening()`
  - `hMinima() vs h_minima()`
  - `watershed() vs watershed()`
- Python `timeit` module : one round to determine the length of each round followed by 5 monitored rounds.
- an external process

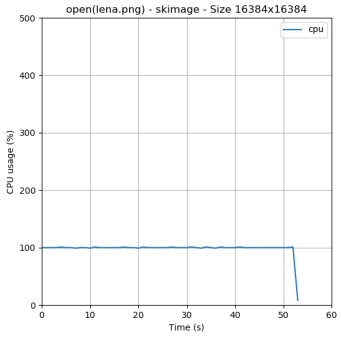
Smil vs scikit-image

Resources usage

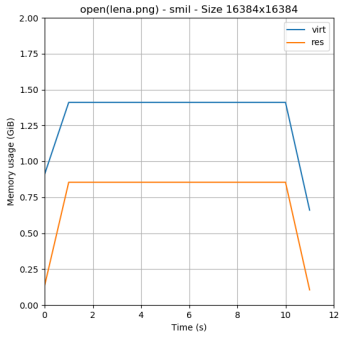
Function : open() - Image : lena.png - Size : 16384x16384
nestor (desktop computer)



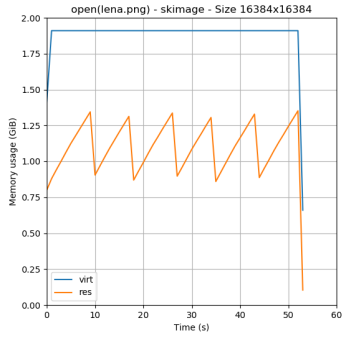
CPU (Smil)



CPU (skImage)



Memory (Smil)



Memory (skImage)

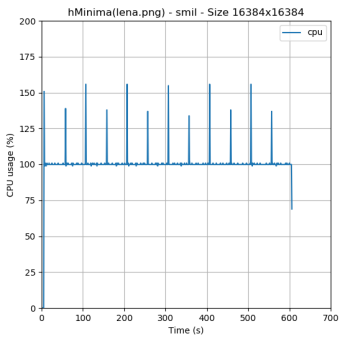
Full results at :

- `https://smil.cmm.minesparis.psl.eu/smil-vs-skimage/nestor-resources.html`
- `https://smil.cmm.minesparis.psl.eu/smil-vs-skimage/taurus-resources.html`

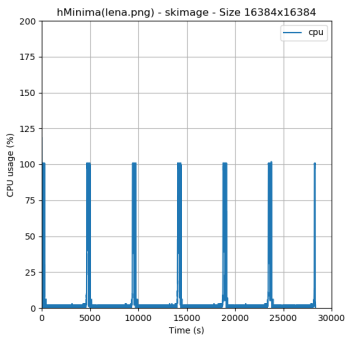
Smil vs scikit-image

Resources usage

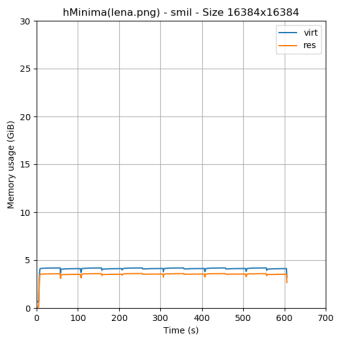
Function : hMinima() - Image : lena.png - Size : 16384x16384
nestor (desktop computer)



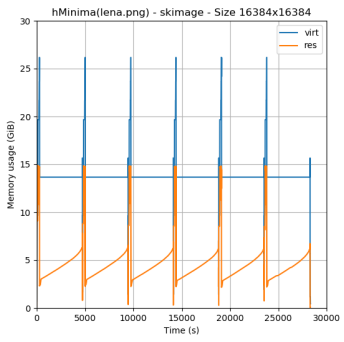
CPU (Smil)



CPU (skImage)



Memory (Smil)



Memory (skImage)

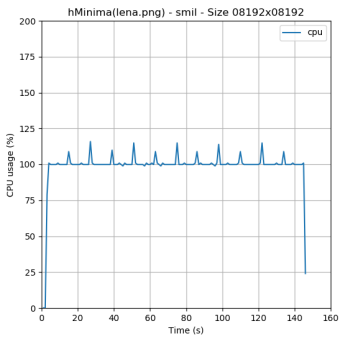
Full results at :

- `https://smil.cmm.minesparis.psl.eu/smil-vs-skimage/nestor-resources.html`
- `https://smil.cmm.minesparis.psl.eu/smil-vs-skimage/taurus-resources.html`

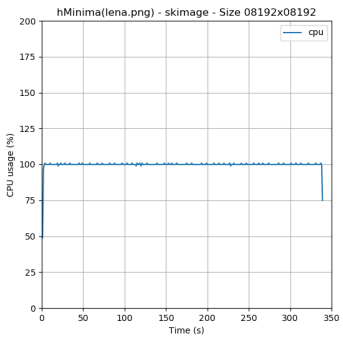
Smil vs scikit-image

Resources usage

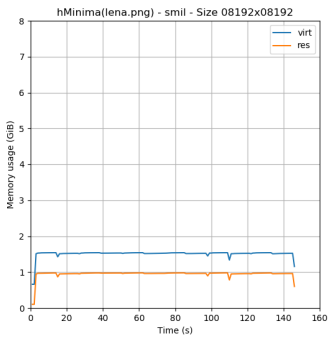
Function : hMinima() - Image : lena.png - Size : 8192x8192
nestor (desktop computer)



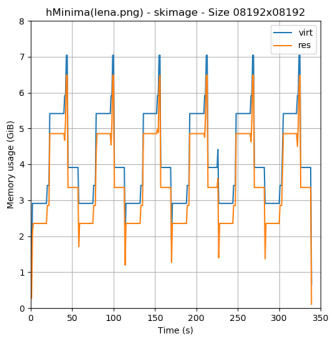
CPU (Smil)



CPU (skImage)



Memory (Smil)



Memory (skImage)

Full results at :

- `https://smil.cmm.minesparis.psl.eu/smil-vs-skimage/nestor-resources.html`
- `https://smil.cmm.minesparis.psl.eu/smil-vs-skimage/taurus-resources.html`

Part V

Elapsed times per function

Smil vs scikit-image

Elapsed times (ms) : binary images

Image size : 2048x2048

Function	nestor				taurus			
	Smil		skImage		Smil		skImage	
	Min	Max	Min	Max	Min	Max	Min	Max
erode	0.37	0.39	53.10	53.30	0.11	0.14	52.30	54.20
open	0.69	0.71	111.00	111.00	0.24	0.25	110.00	117.00
distance	110.00	125.00	444.00	767.00	141.00	153.00	550.00	915.00
label	13.50	120.00	28.90	55.20	18.10	150.00	38.60	63.70
fastLabel	50.40	58.30	29.00	55.30	66.70	86.90	38.50	68.30
segmentation	727.00	888.00	655.00	7310.00	938.00	1130.00	812.00	6220.00
areaThreshold	28.00	165.00	54.50	109.00	48.70	229.00	71.30	145.00
thinning	1330.00	9280.00	7570.00	53200.00	772.00	5400.00	7780.00	54500.00

Image size : 8192x8192

Function	nestor				taurus			
	Smil		skImage		Smil		skImage	
	Min	Max	Min	Max	Min	Max	Min	Max
erode	10.00	10.20	837.00	846.00	1.43	1.66	875.00	881.00
open	17.30	17.40	1740.00	1760.00	2.38	2.71	1770.00	1770.00
distance	3020.00	3080.00	9040.00	10800.00	5310.00	6080.00	17500.00	18700.00
label	214.00	2080.00	438.00	855.00	266.00	2200.00	620.00	1020.00
fastLabel	718.00	775.00	438.00	857.00	759.00	830.00	614.00	1030.00
segmentation	14100.00	20200.00	13100.00	39800.00	18100.00	27700.00	22200.00	44500.00
areaThreshold	419.00	2680.00	838.00	1720.00	717.00	3630.00	1170.00	2000.00
thinning	84200.00	595000.00	504000.00	3590000.00	47800.00	341000.00	514000.00	3670000.00

Full results at :

- `https://smil.cmm.minesparis.psl.eu/smil-vs-skimage/elapsed-times.html`

Smil vs scikit-image

Elapsed times (ms) : gray level images

Image size : 2048x2048

Function	nestor				taurus			
	Smil		skImage		Smil		skImage	
	Min	Max	Min	Max	Min	Max	Min	Max
erode	0.36	0.37	53.10	53.70	0.12	0.13	52.60	54.30
open	0.68	0.76	111.00	111.00	0.20	0.27	109.00	110.00
tophat	1.36	1.58	123.00	124.00	0.34	0.39	124.00	126.00
gradient	2.74	2.77	741.00	768.00	1.50	1.56	807.00	917.00
hMinima	786.00	1080.00	2470.00	3720.00	984.00	1160.00	2250.00	4140.00
watershed	217.00	270.00	1140.00	2160.00	335.00	389.00	1270.00	2320.00
segmentation	1080.00	1240.00	3140.00	4290.00	1420.00	1670.00	3690.00	4890.00
areaOpen	277.00	369.00	7390.00	30000.00	322.00	391.00	7430.00	29400.00

Image size : 8192x8192

Function	nestor				taurus			
	Smil		skImage		Smil		skImage	
	Min	Max	Min	Max	Min	Max	Min	Max
erode	10.00	10.10	837.00	838.00	1.44	1.53	880.00	898.00
open	17.30	17.40	1740.00	1750.00	2.47	3.29	1770.00	1850.00
tophat	34.90	35.10	1940.00	1950.00	11.10	11.30	2000.00	2010.00
gradient	41.50	41.70	12000.00	12200.00	17.80	18.20	13100.00	14300.00
hMinima	15900.00	27300.00	53100.00	82600.00	17800.00	30800.00	48500.00	75800.00
watershed	3430.00	3800.00	14600.00	25500.00	5030.00	5220.00	17200.00	29900.00
segmentation	16800.00	18200.00	47600.00	59700.00	20100.00	22300.00	55800.00	69800.00
areaOpen	4920.00	6780.00	290000.00	2280000.00	5530.00	7910.00	269000.00	2500000.00

Full results at :

- `https://smil.cmm.minesparis.psl.eu/smil-vs-skimage/elapsed-times.html`

Part VI

Conclusion

Conclusion

- Most of the time, except for `label()` and, eventually, other functions making use of it, **Smil** is much faster than **skImage**;
- Whenever possible, **Smil** make use of parallelization. **skImage** doesn't;
- **skImage** is very handy, maybe easier to use than **Smil**
- **Smil** has many advanced morphological features not available under **skImage**
- **Smil** is a good complement to **skImage** in applications needing morphological transformations.

Quick starting with Smil

Some information

- Web site and documentation : `https://smil.cmm.minesparis.psl.eu`
- Download
  - Packages for Linux (Ubuntu, Debian, Centos, Fedora, ...) and Anaconda
  - Packages for Windows : coming soon
  - Get sources from `https://github.com/MinesParis-MorphoMath/smil` and compile it

Installing : as simple as ...

- Linux packages (E.g. Ubuntu)

```
SRV=https://smil.cmm.minesparis.psl.eu
KURI=$SRV/packages/jmartins@cmm.gpg.key
wget -O- $KURI 2>/dev/null | apt-key add -
apt-add-repository \
  "deb [arch=amd64] $SRV/packages/ubuntu focal main"

apt-get update
apt-get install smil-*
```

- Anaconda

```
conda install conda install -c bmarchand smil
```

- See : `https://smil.cmm.minesparis.psl.eu/doc/p230.html`

Quick starting with Smil

Some information

- A Quick Reference of available functions :

  `https://smil.cmm.minesparis.psl.eu/publis/SmilQuickReference.pdf`

- Smil programming under Python

  `https://smil.cmm.minesparis.psl.eu/doc/p600.html`

A very simple example

```
import smilPython as sp
import numpy as np

imIn  = sp.Image("lena.png")
imOut = sp.Image(imIn)
r = sp.erode(imIn, imOut, sp.HexSE(2))
imIn.show("Original image")
imOut.show("Eroded image")

# Smil -> Numpy
im_np = imOut.getNumArray()

# Numpy -> Smil
im = sp.Image(im_np)
```

Thanks !
Questions ?

Or ask :

- jose-marcio.martins@minesparis.psl.eu
- samy.blusseau@minesparis.psl.eu