COMPASS benchmark results

Contents

The goal of this benchmark is to test and compare several features included on COMPASS. More precisely, entire end-to-end simulations are performed and performances in terms of Strehl ratio (final long exposure after 2,000 iterations, short exposures, RMS value) and in terms of execution time for each function that composed the AO loop.

Cases

In this benchmark, several method of centroiding have been tested with various reconstruction method on different sizes of telescope and Shack-Hartmann dimensions.

Centroiders
Cog
Thresholded cog
Brightest pixels cog
Weighted cog
Correlation
Geometric

Controllers
Least Square
LS + modal optimization
Minimum Variance
Geometric

The study cases follows :

Star type	Telescope diameter	Number of subapertures	Number of pixel per subap
NGS	8 m	16x16	8x8
			16x16
		40x40	8x8
			16x16
	32 m	64x64	8x8
			16x16
	40 m	80x80	8x8
			16x16
LGS	8m	16x16	8x8
		40x40	10x10

Additionally, some noisy cases have been run for NGS 8x8 pix cases: “noisy” means the number of photons per sub-apertures is limited to ~125, instead of >1,000 for normal cases.

Centroiders parameters

Here comes the parameters used for each centroiding method for those benchmarks:

Thresholded cog	Thresh = 9.0
Brightest pixels cog	Nmax = 4 (hippo4) or 16
Weighted cog	type_fct = “gauss” width = 2.0
Correlation	type_fct = “gauss” width = 2.0

Controllers parameters

LS / LS+modopti / MV / GEO	Maxcond = 1500. delay = 1 gain = 0.4
LS+modopti	Nrec = 2048 gmin = 0.001 gmax = 0.5 ngain = 500

The parameter nmodes specific to the LS+modal optimization controllers is set for each cases as follow:

Number of subapertures	y_controllers.nmodes
16x16	216
40x40	1286
64x64	3259
80x80	5064

Other parameters

All the other input parameters of the simulations are constant for all the cases: the parameter file template used for those benchmarks is joined to this document as annex.

Environment

Hardware & software

This benchmark have been run on 3 different systems whose configurations are summarized in this table:

Name	OS	CPU		GPU	CUDA Version	MAGMA Version
		Cores (HT)	Type			
Hippo4	Linux	2x6 (24)	Intel Xeon X5690@3,47GHz	Tesla M2090	5.5	1.4
Yogi	Linux	1x4 (8)	Core i7-870@2.93GHz	GeForce GTX 680	6.0	1.5
Idris	Linux	2x8 (32)	Sandy Bridge E5-2670@2.60GHz	Tesla K20	6.5	1.6

Benchmark script

The benchmark script used is available on the repository as *benchmark_script.i* . It takes as inputs a parfile where the case dimensions are specified, the centroider and the controller to use. All the parfiles used are available on the repository in *yoga_ao/data/par/par4bench/*. After initializing the specific parameters for the input centroider and controller, it runs a classical AO loop.

The timers used to profile the simulation are CUDA timers interfaced with Yorick to be called from the script. To be sure to obtain the most precise time possible, each function is surrounded with a call to *ThreadSync* that ensure the kernel function is over.

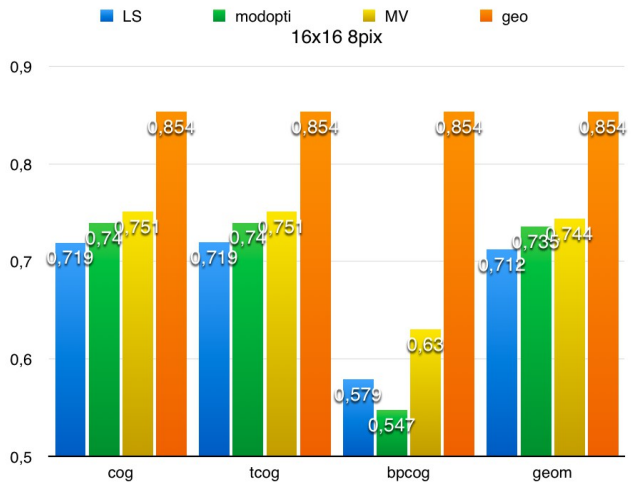
Outputs are all the function timers, the final long exposure Strehl ratio, short exposure Strehl ratio at each iteration and RMS value of short exposure Strehl ratio, stored in 2 CSV files.

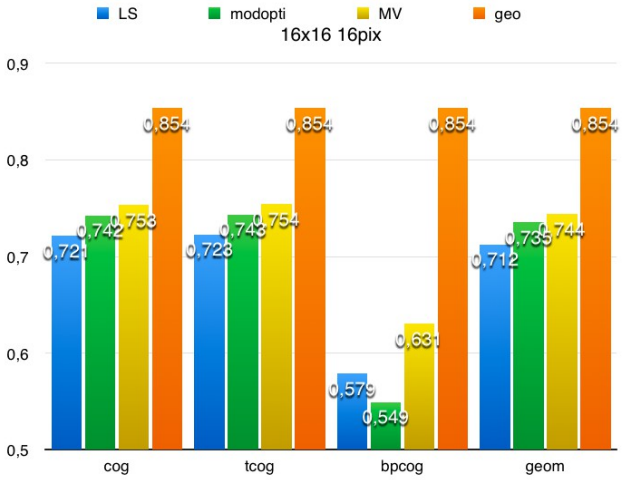
This script is called in batch mode from a shell script *benchmark.sh* where the parameter files, centroider and controller to use are specified.

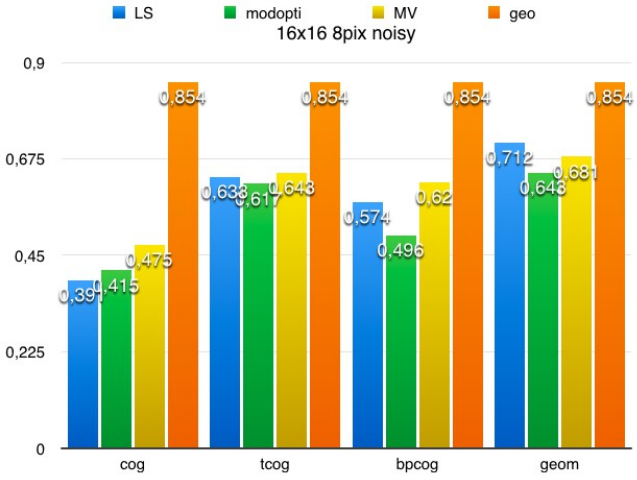
Results

Strehl performances

8 m, NGS 16x16 subap



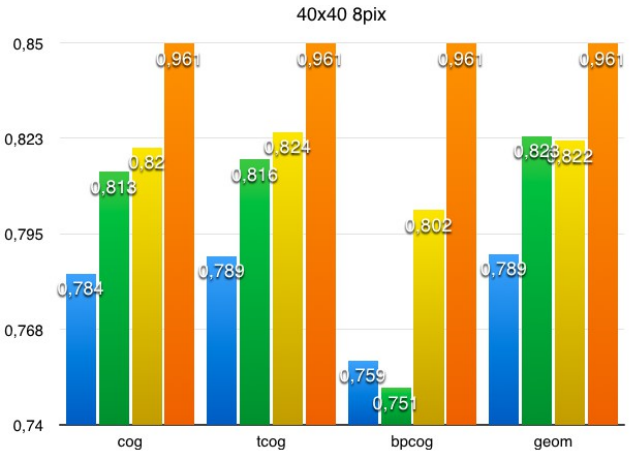


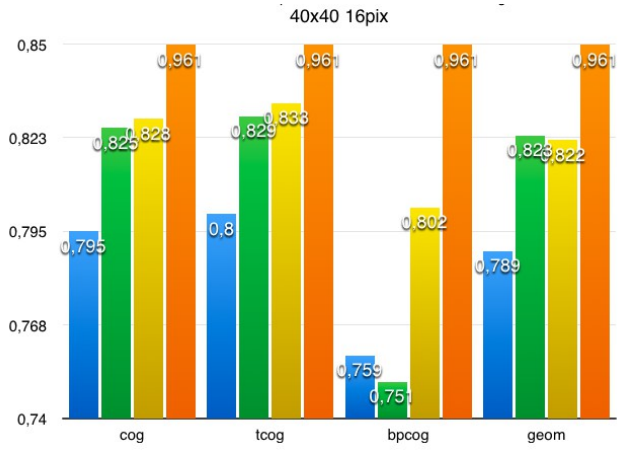


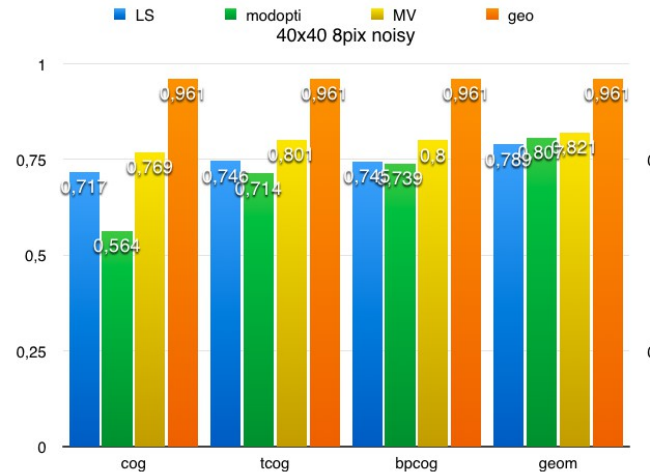
Final SR on Hippo4 with:

- imat_geom
- 4 brightest pixels

8m, NGS 40x40



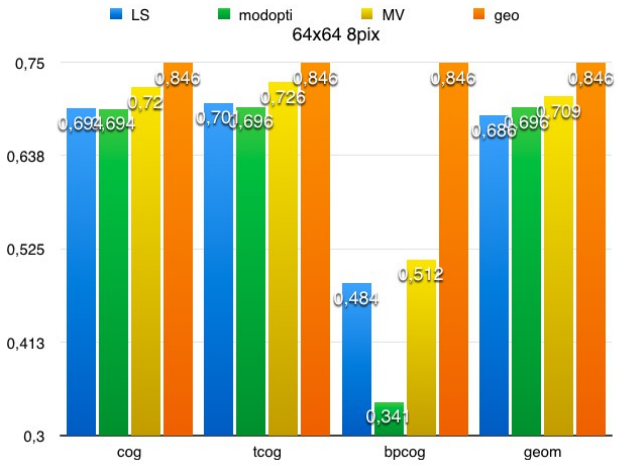


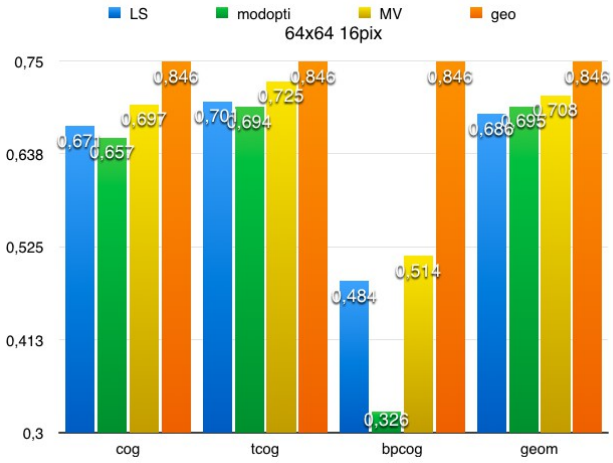


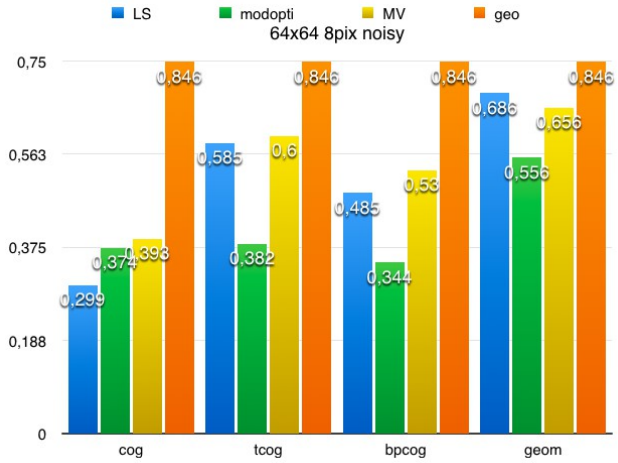
Final SR on Hippo4 with:

- imat_geom
- 4 brightest pixels

32m, NGS 64x64



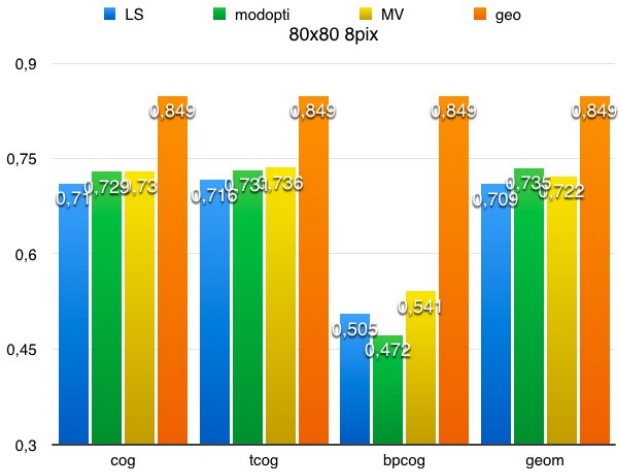


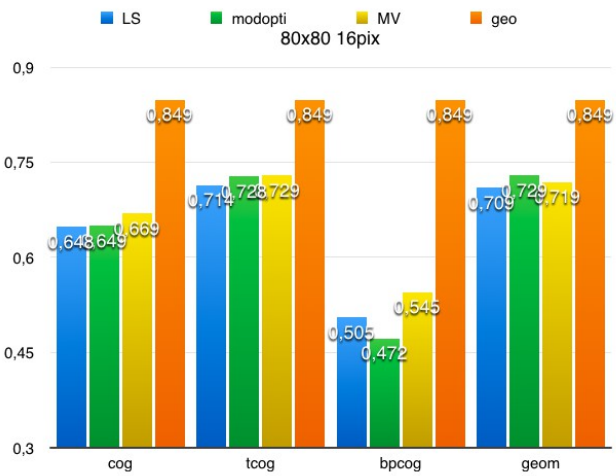


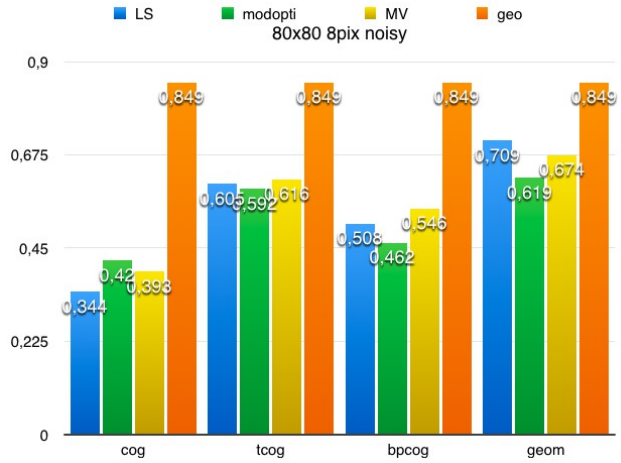
Final SR on Hippo4 with:

- imat_geom
- 4 brightest pixels

40m, NGS 80x80



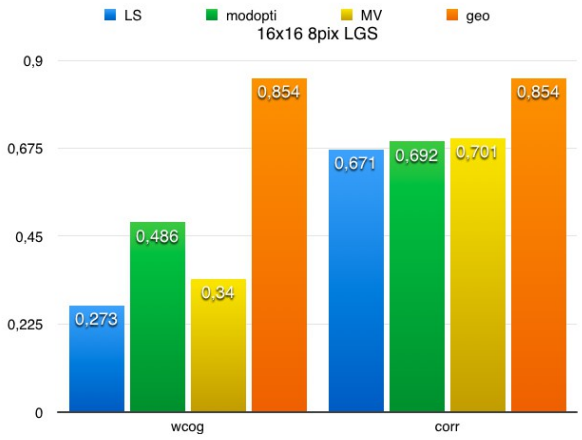


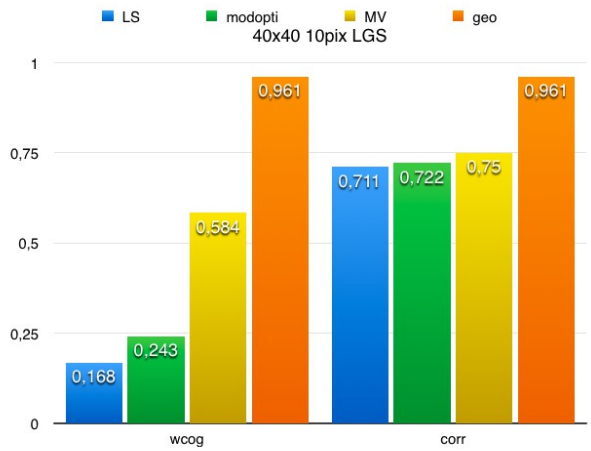


Final SR on Hippo4 with:

- imat_geom
- 4 brightest pixels

LGS cases

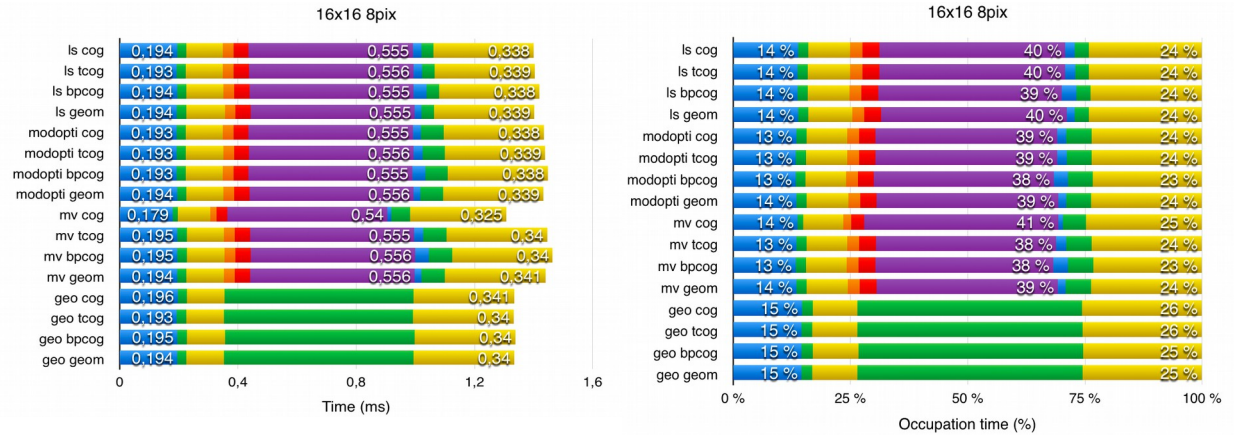

LS
modopti
MV
geo
16x16 8pix LGS
0,9
0,675
0,45
0,225
0
0,273
0,486
0,34
0,854
0,671
0,692
0,701
0,854
wcog
corr


LS
modopti
MV
geo
40x40 10pix LGS
1
0,75
0,5
0,25
0
0,168
0,243
0,584
0,961
0,711
0,722
0,75
0,961
wcog
corr

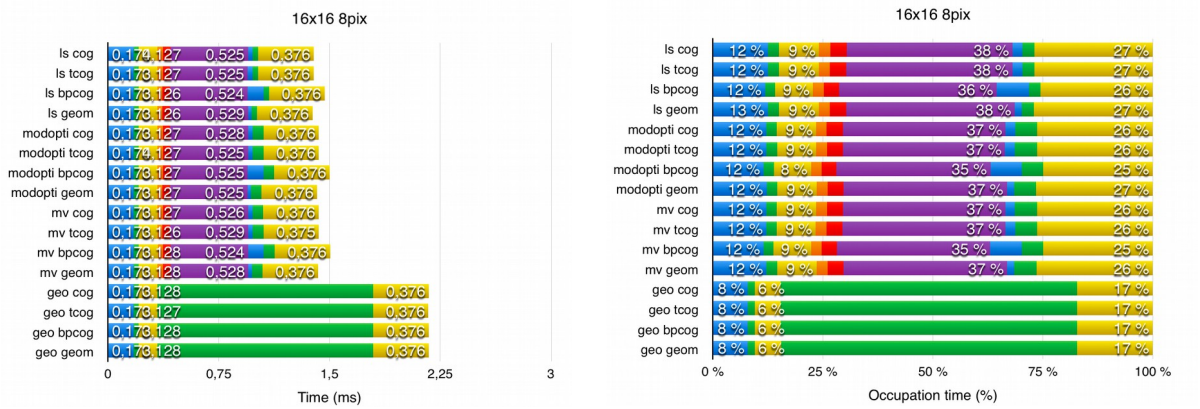
Computation performances

8m, 16x16, 8x8 pixels

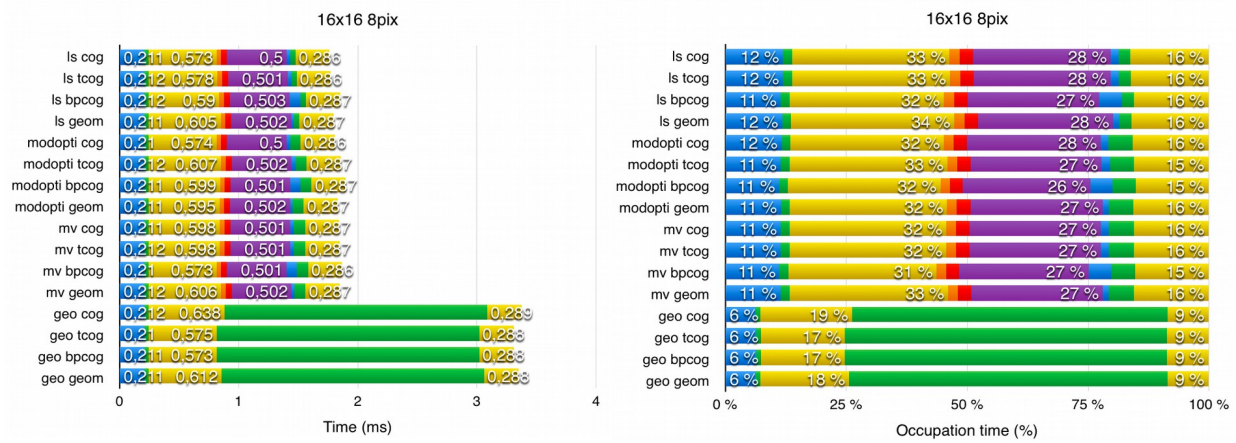
Hippo



Yogi

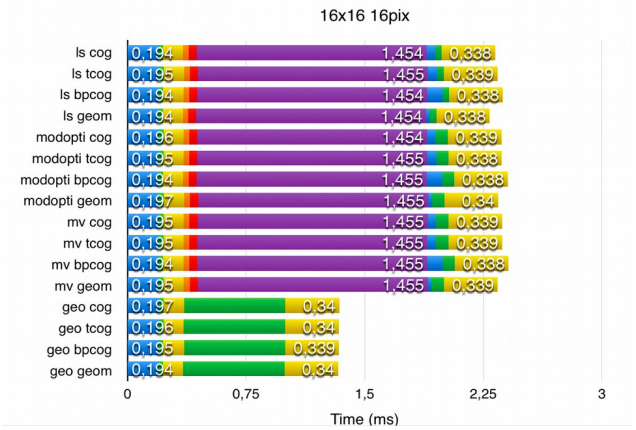


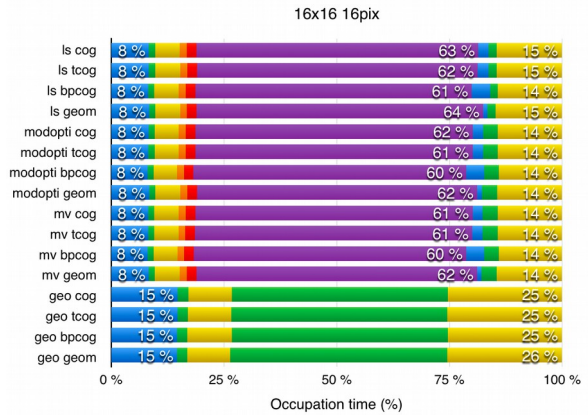
Idris



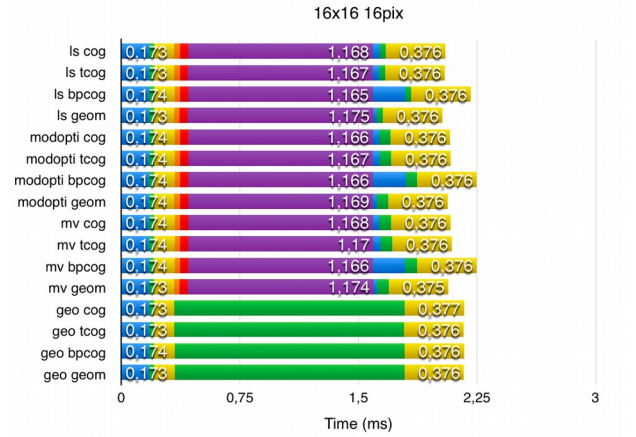
8m, 16x16, 16x16 pixels

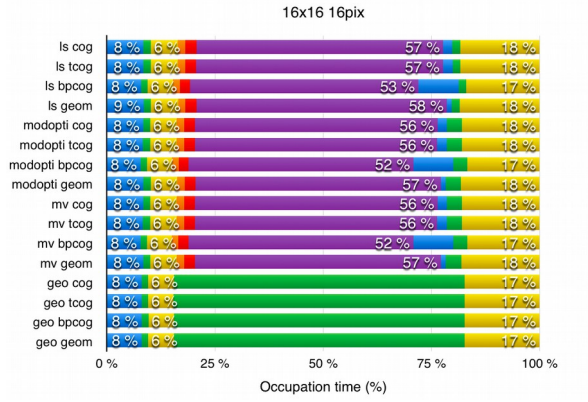
Hippo



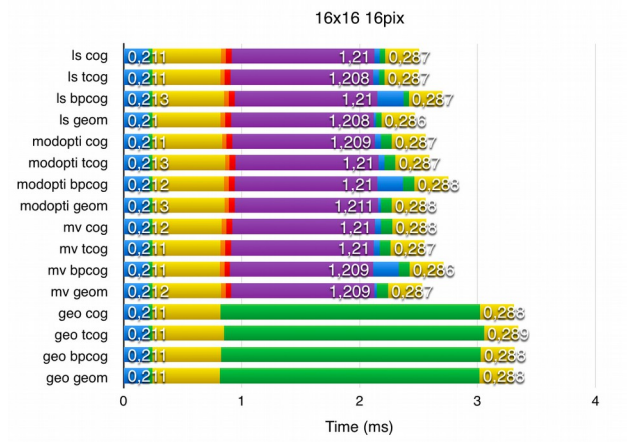


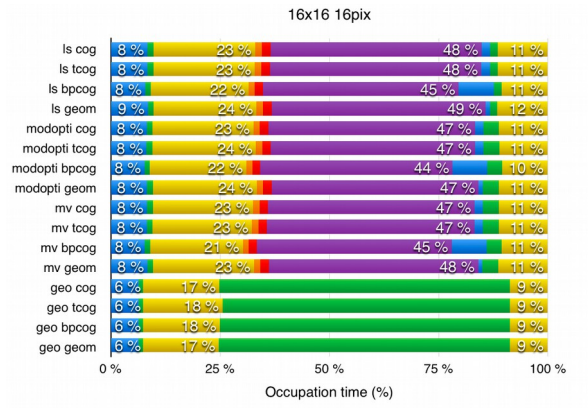
Yogi





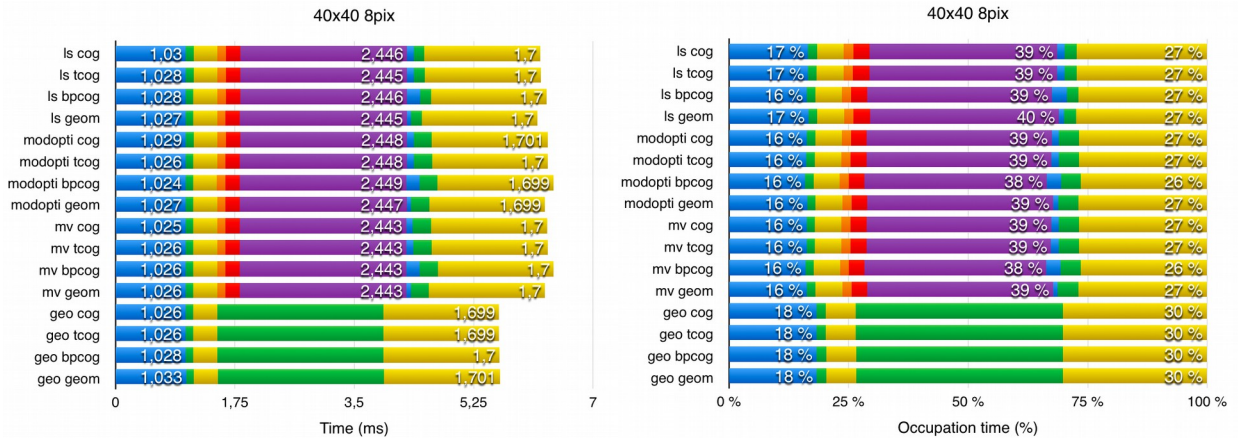
Idris



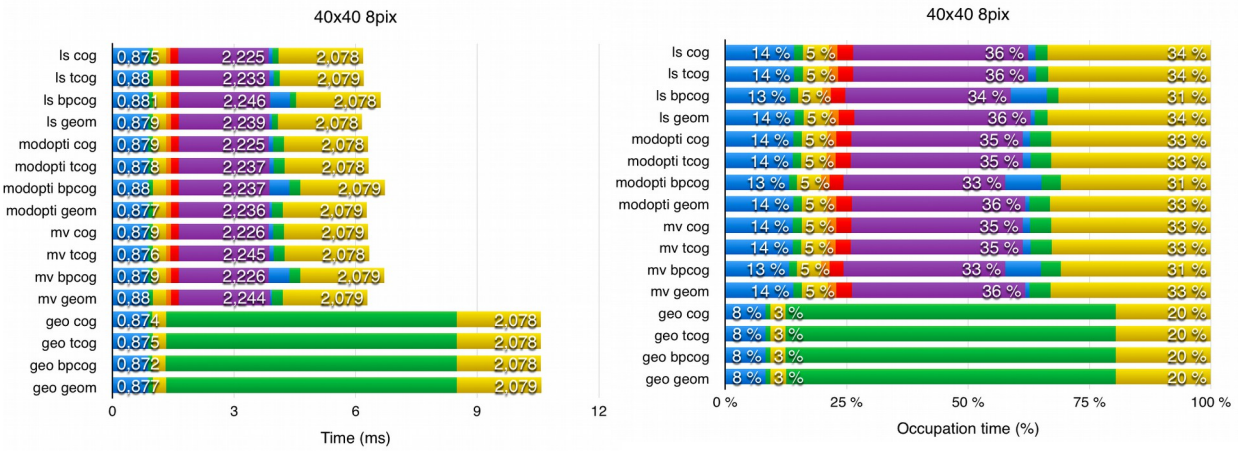


8m, 40x40, 8x8 pixels

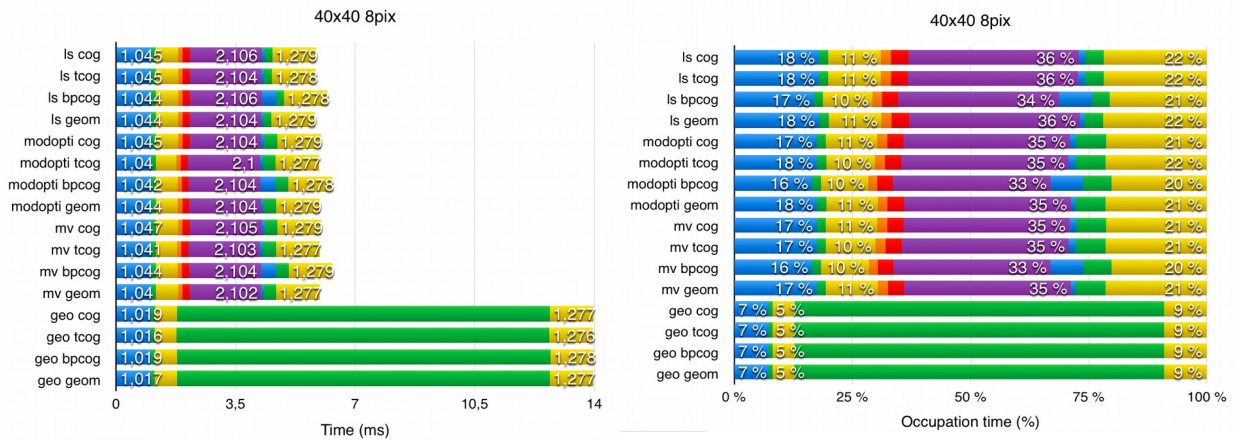
Hippo



Yogi

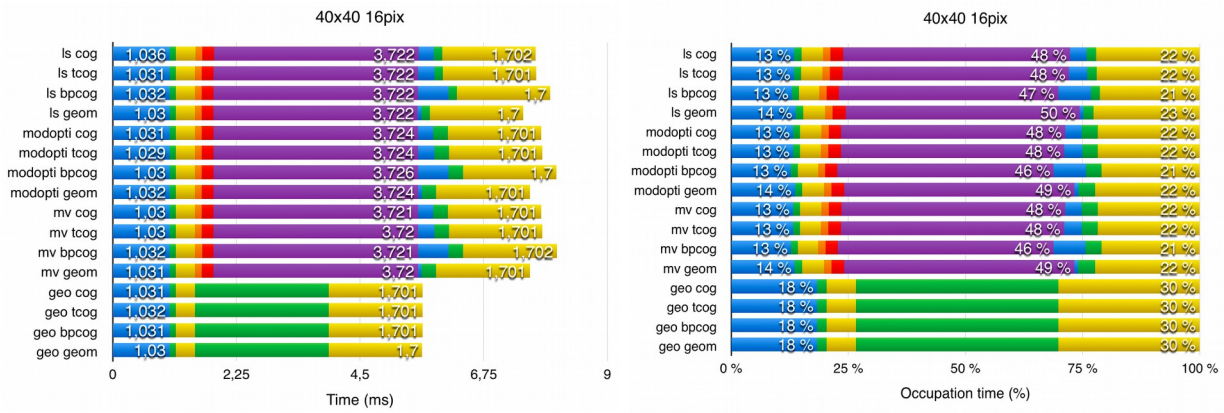


Idris

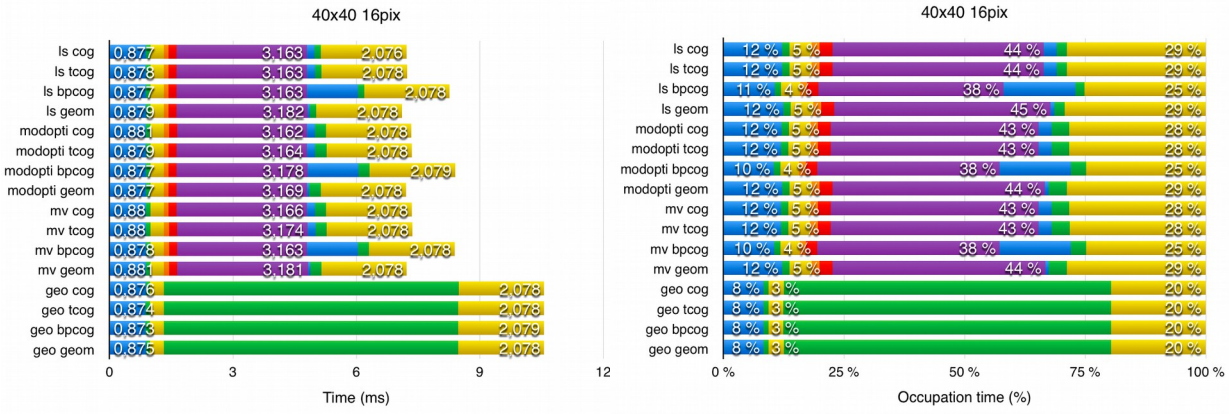


8m, 40x40, 16x16 pixels

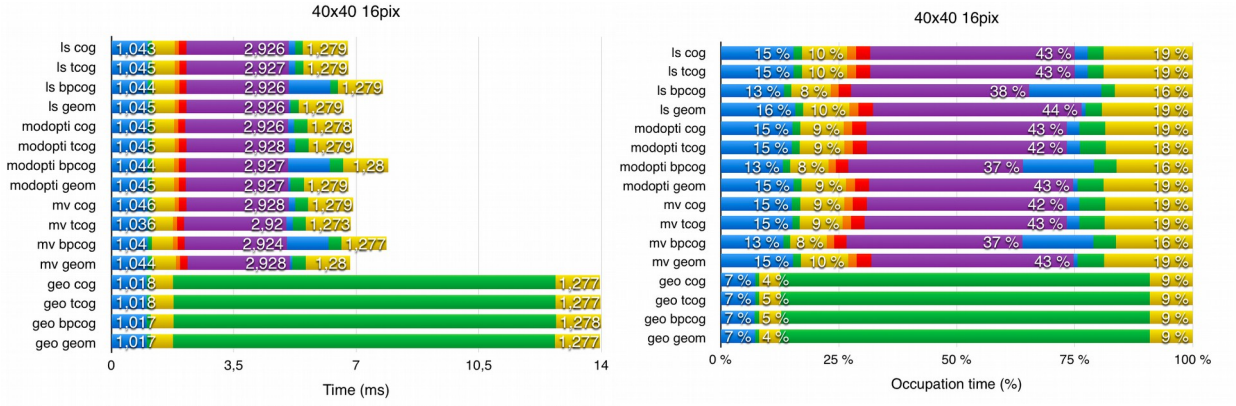
Hippo



Yogi

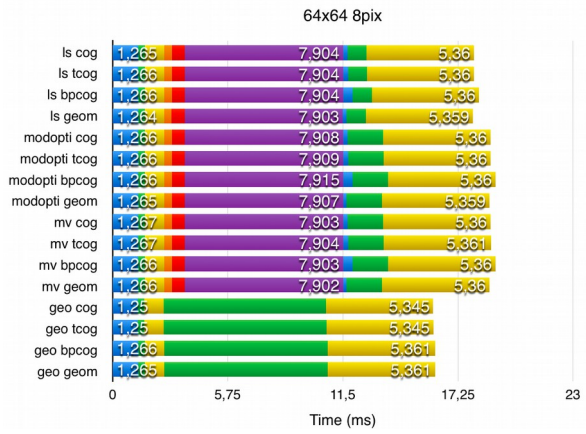


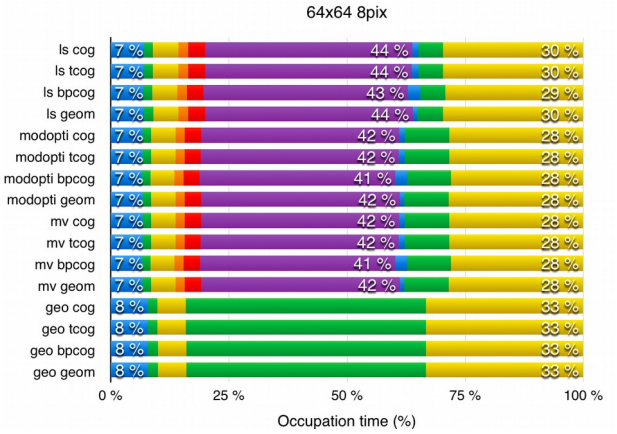
Idris



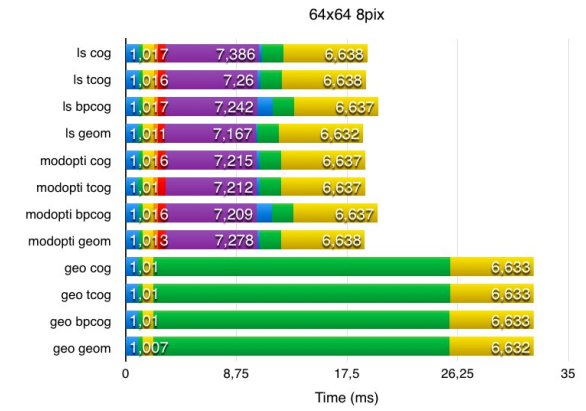
32m, 64x64, 8x8 pixels

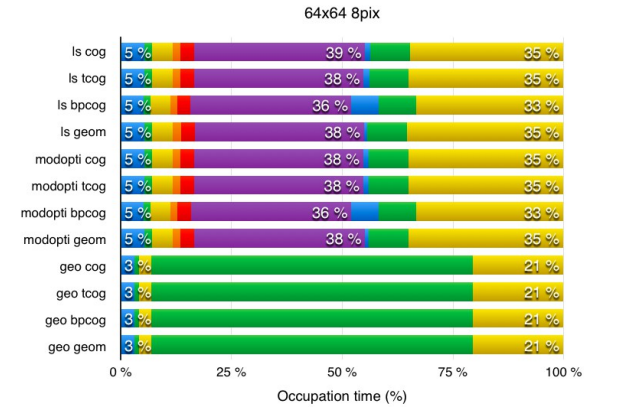
Hippo



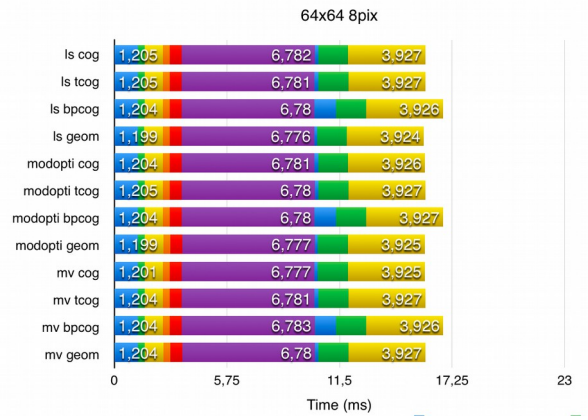


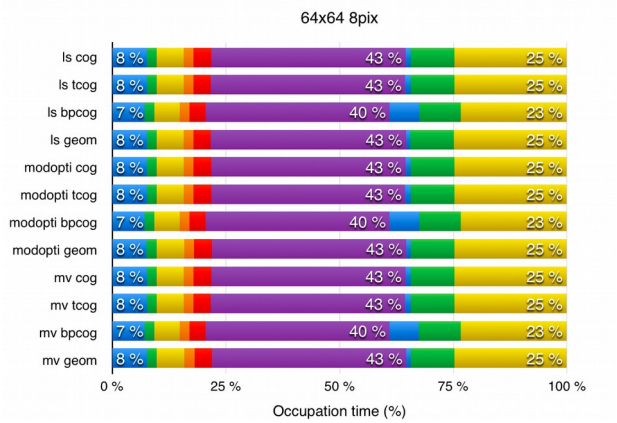
Yogi





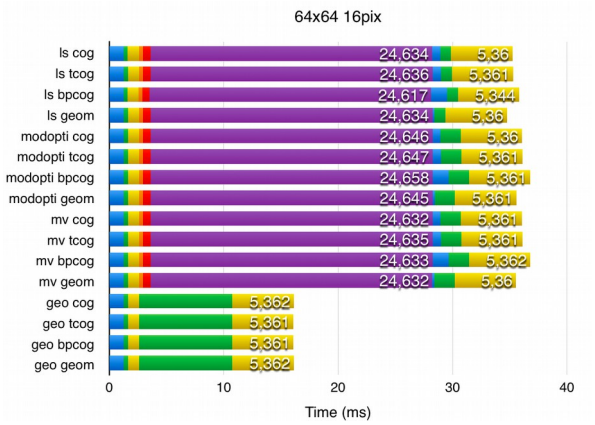
Idris

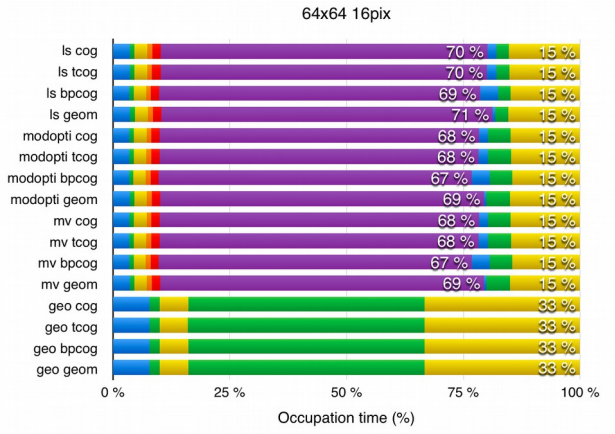




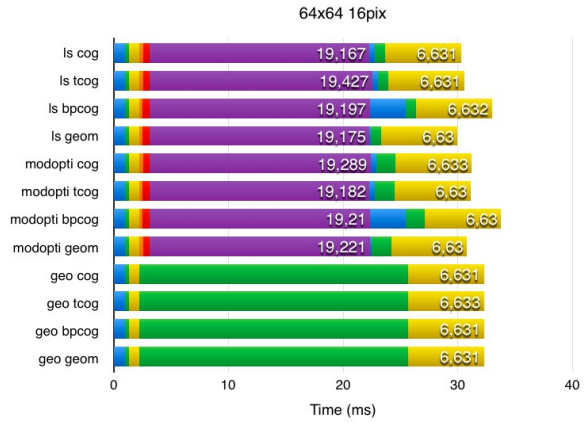
32m, 64x64, 16x16 pixels

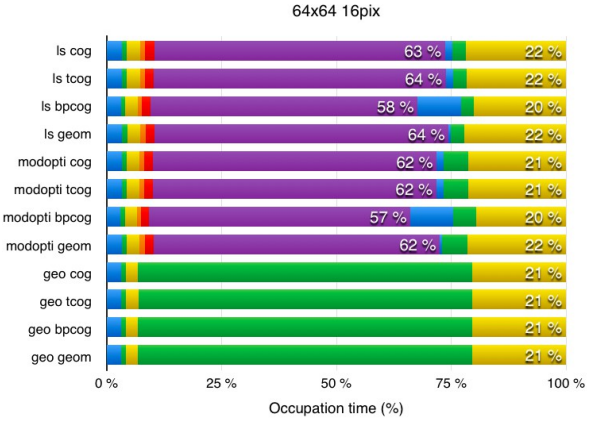
Hippo



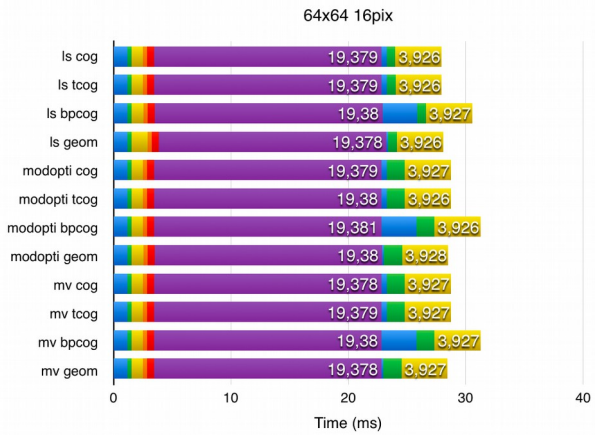


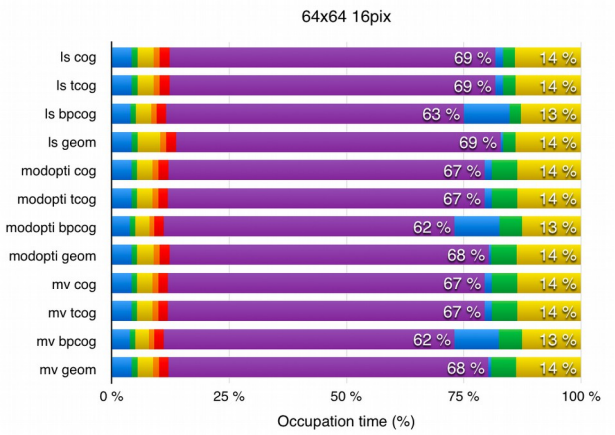
Yogi





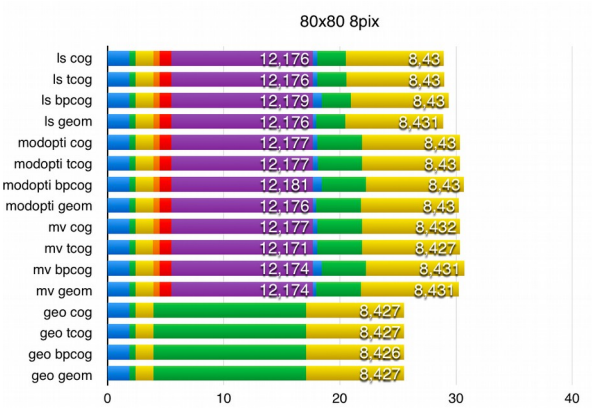
Idris

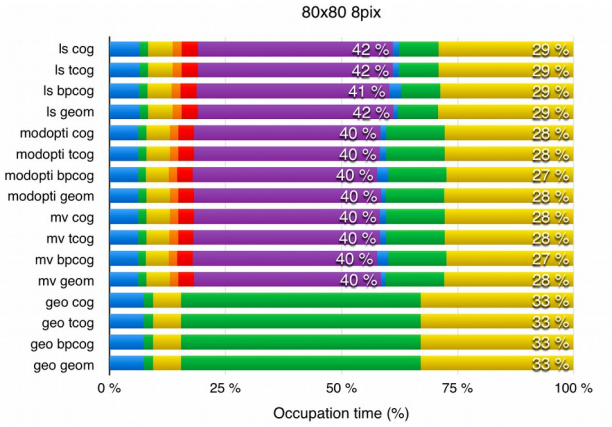




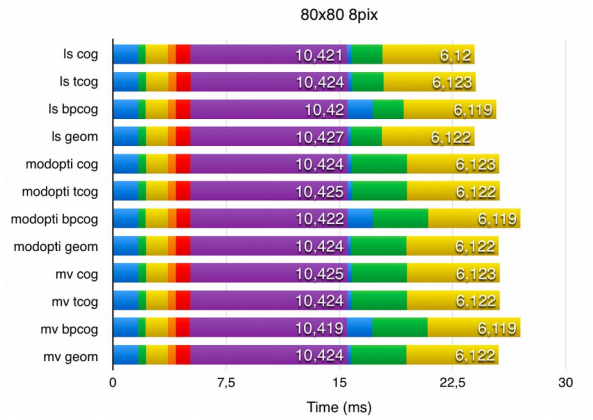
40m, 80x80, 8x8 pixels

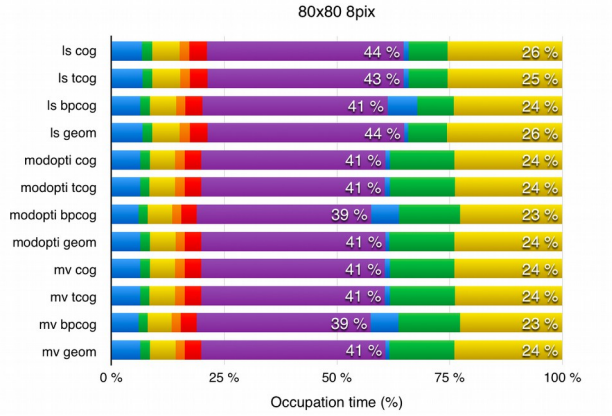
Hippo





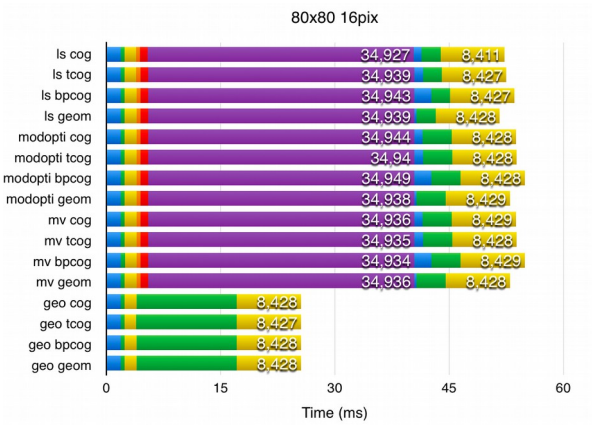
Idris

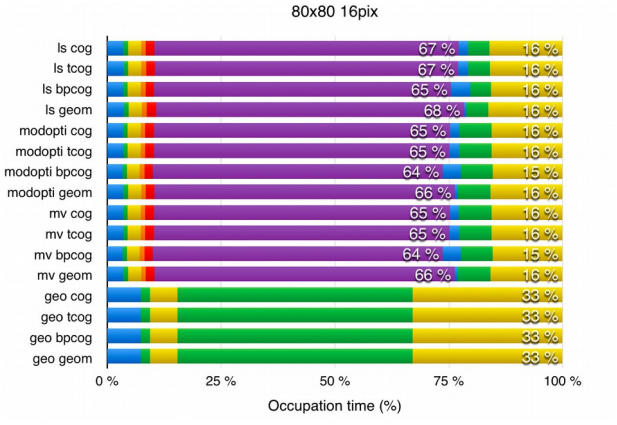




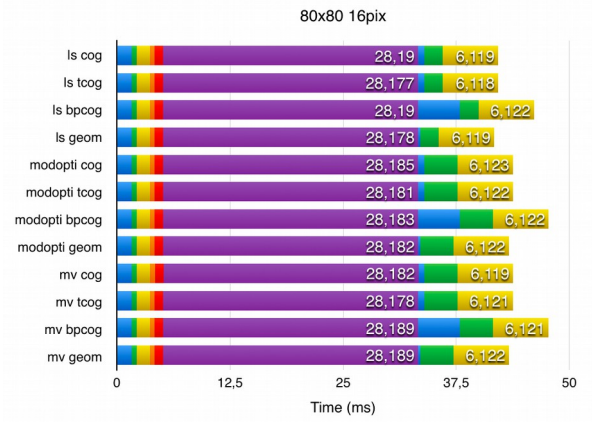
40m, 80x80, 16x16 pixels

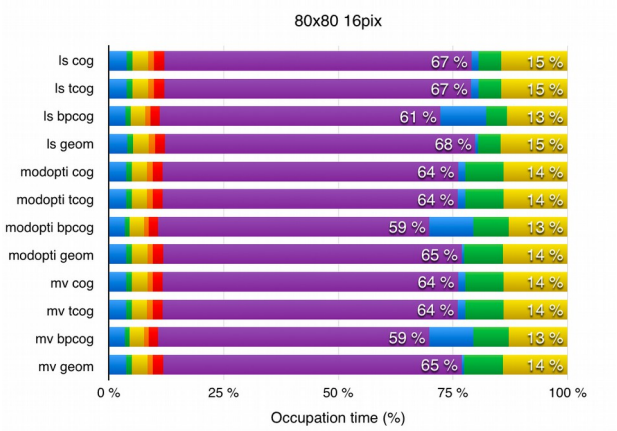
Hippo





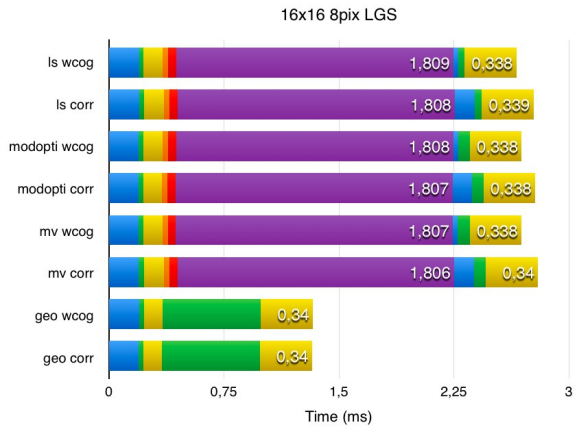
Idris

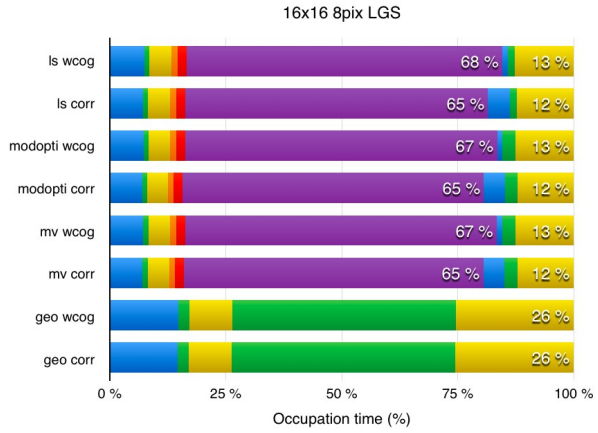




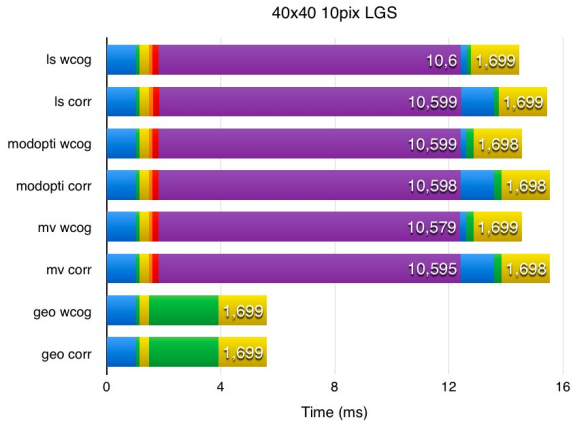
LGS cases (Hippo)

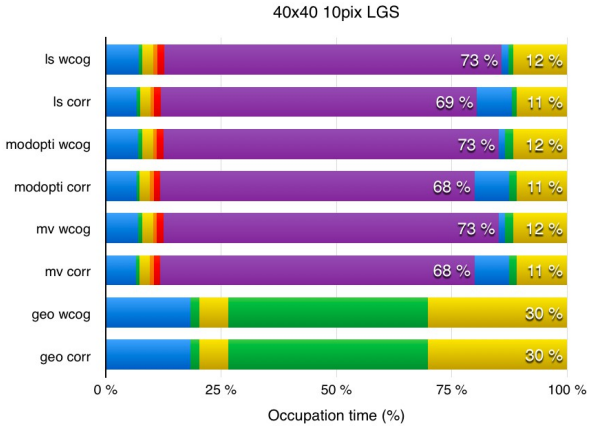
16x16



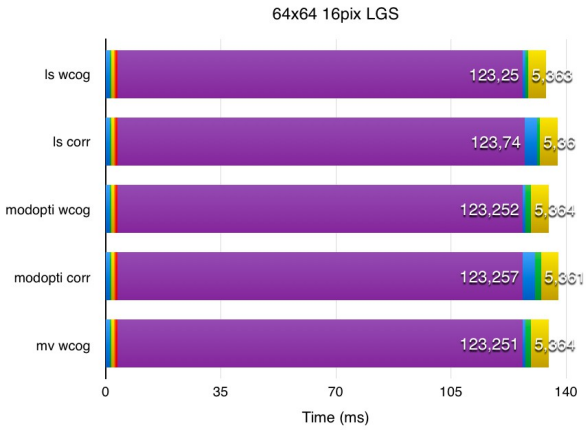


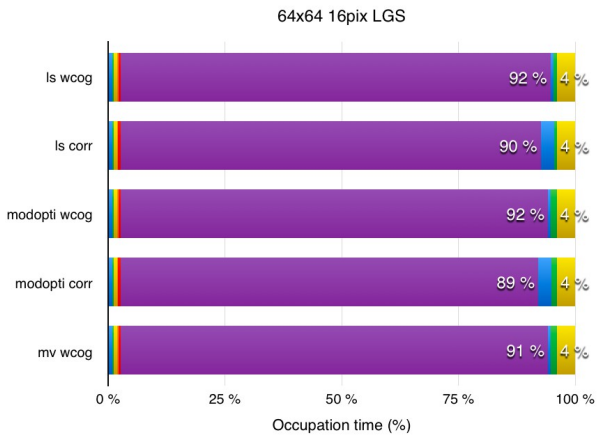
40x40





64x64





Remarks

Note that the minimum variance in the 32x32 case and ELT cases cannot be run on Yogi because the available memory is not enough.

On Idris, the geometric controller computation failed for cases >= 32x32 because of an unknown bug with the CUSPARSE Library.

The image computation of the WFS is the major part of the profile with the computation of the DM shape.

For the DM shape, other implementations have been tried : one using the CUSPARSE Library with a matrix-vector multiplication, the second one using a convolution process between actuator position and influence function. Those two implementations are about 2 times slower than the current one.

Annex

Parameter file template

In blue, the parameters that changes with the cases. All the other never change.

```
/*
geometry inits
*/
y_geom.zenithangle = 0.;

/*
telescope inits
*/
y_tel.diam = 40.f;
y_tel.cobs = 0.12f;

/*
atmosphere inits
*/
y_atmos.r0          = 0.16;
y_atmos.nscreens    = 1;
y_atmos.frac        = &([1.]);
y_atmos.alt         = &([0.]);
y_atmos.windspeed   = &([20.]);
y_atmos.winddir     = &([45]);
y_atmos.L0          = &([1.e5f]);

/*
target inits
*/
y_target.ntargets = 1;
y_target.xpos     = &([0.0f]);
y_target.ypos     = &([0.0f]);
y_target.lambda   = &([1.65]);
y_target.mag      = &([10.]);

/*
loop inits
*/
y_loop.niter  = 5000;
y_loop.ittime = 1/500.0f;

/*
wfs inits
*/
y_wfs               = array(wfs_struct(atmos_seen=1),1); // clean start
y_wfs(1).type       = "sh";
y_wfs(1).nxsub      = 80;
y_wfs(1).npix       = 8;
y_wfs(1).pixsize    = 0.3;
y_wfs(1).fracsub    = 0.8;
```

```
y_wfs(1).xpos         = 0.0;
y_wfs(1).ypos         = 0.0;
y_wfs(1).lambda       = 0.5;
y_wfs(1).gsmag        = 3.;
y_wfs(1).optthroughput = 0.5;
y_wfs(1).zerop        = 1.e11;
y_wfs(1).noise        = 3;

/*
dm inits
*/
ndm = 2;
y_dm                = array(dm_struct(),ndm); // clean start

y_dm(1).type         = "pzt";
y_dm(1).nact         = y_wfs(1).nxsub + 1;
y_dm(1).alt          = 0.;
y_dm(1).thresh       = 0.3;
y_dm(1).coupling     = 0.2;
y_dm(1).unitpervolt  = 0.01;
y_dm(1).push4imat    = 100.0f;

y_dm(2).type         = "tt";
y_dm(2).alt          = 0.;
y_dm(2).unitpervolt  = 0.0005;
y_dm(2).push4imat    = 10.0f;


/*
centroiders inits
*/
ncentroiders = 1;
y_centroiders         = array(centroider_struct(),ncentroiders);

y_centroiders.nwfs    = 1;
y_centroiders(1).type  = "cog";

/*
controllers inits
*/
ncontrollers = 1;
y_controllers         = array(controller_struct(),ncontrollers);

y_controllers(1).type   = "ls";
y_controllers(1).nwfs   = &[1];
y_controllers(1).ndm    = &[1,2];
y_controllers(1).maxcond = 1500.;
y_controllers(1).delay  = 1;
y_controllers(1).gain   = 0.4;

y_controllers(1).modopti = 0;
y_controllers(1).nrec   = 2048;
y_controllers(1).nmodes  = 5064;
y_controllers(1).gmin   = 0.001;
```

```
y_controllers(1).gmax    = 0.5;
y_controllers(1).ngain   = 500;

/*
rtc inits
*/
y_rtc.nwfs =1;
y_rtc.centroiders = &(y_centroiders);
y_rtc.controllers  = &(y_controllers);
```