GE and SR for astrometric centering of Hubble Space Telescope images

GI@ICSE 2024

Sarmiento, R.; de la Cruz, M.; Ortega A.;
Girard, T.M.; Casetti-Dinescu, D.I.; Cervantes, A.; Baena-Galle, R.

GE

Informal description:

- GE is a genetic-inspired automatic programming model independent on the programming language that mixes genetic algorithm and formal models (grammars) of the programming language.

Automatic programming and “genetic programming”:

- GE is not the first attempt to generate programs by means of genetic algorithms.
- Koza’s works introduce an automatic programming model in LISP for solving problems using genetic algorithms (“genetic programming”).

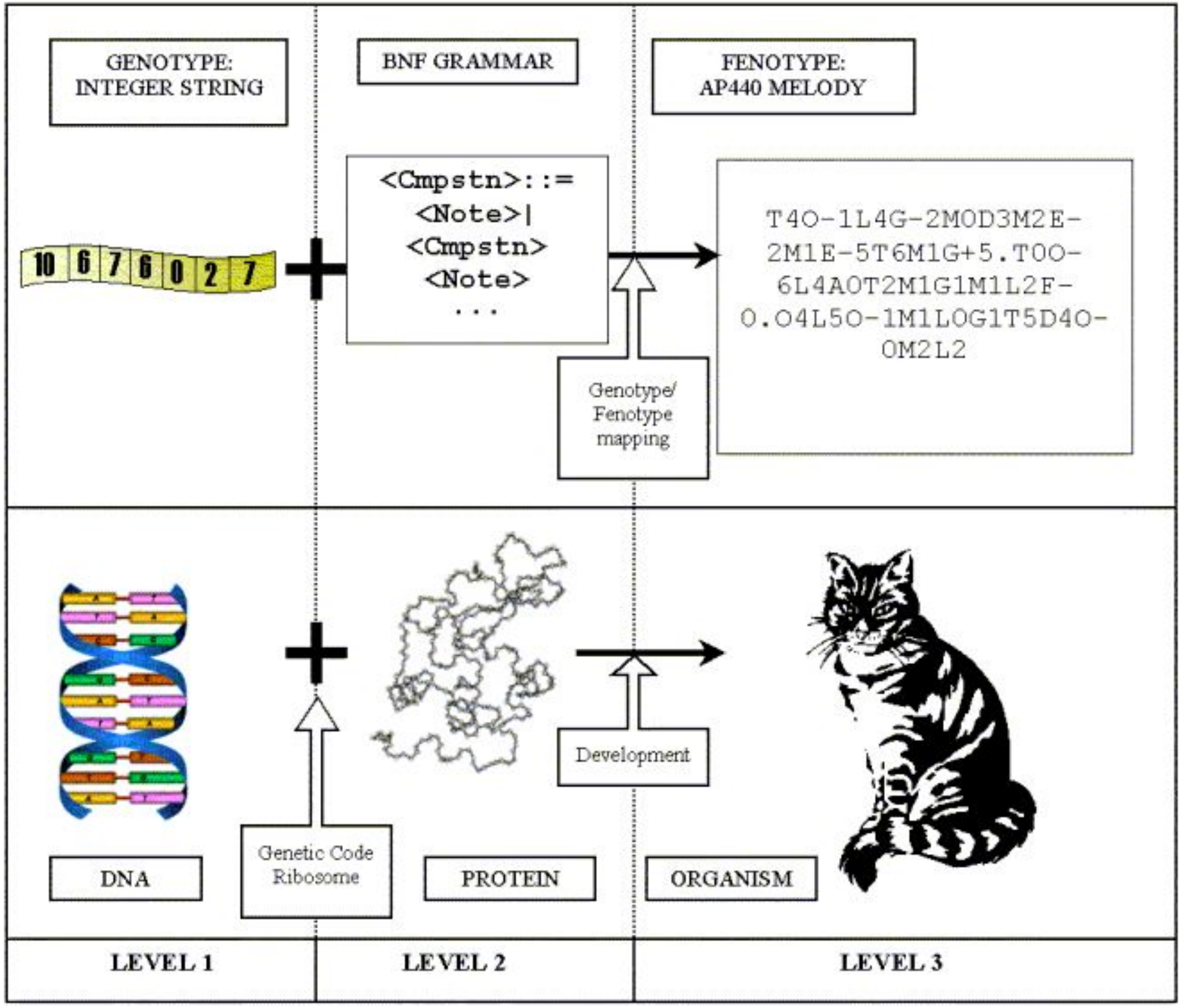
GE

Informal description:

- GE uses a formal grammar in the mapping process from genotype to phenotype to model the DNA contribution to this process in Nature.
- A key contribution of GE is a mapping mechanism that does not depend on the problem of the individuals format (genotype: integer chains called codons) and the translation mechanism is independent on the problem.
- The use of different grammars allows to finally obtain different individuals (phenotypes). The following figure shows this.

GE

Biological metaphor



GE

Genotype-phenotype mapping

- It starts from the grammar axiom and the genotype (vector of integers named codons)
- It finishes exhausting the genotype or generating a valid phenotype (that is, when the "current genotype" has become a valid phenotype because it contains only terminal symbols).
- Each codon in the genotype is used to choose the derivation applied to the left most non terminal by some simple method for example by means of the remainder module the number of rules applicable to the non-terminal).
- Next figure depicts the process for generating arithmetic expressions.

220	240	220	203	101	53	202	203	102	55	220	202	243	130	37	202	203	140	39	202	203	102

Fin proceso: expresión

1.0- sin(x)*sin(x) - sin(x)*sin(x)

```
<expr>
<expr><op><expr>
<expr><op><expr><op><expr>
<expr><op><expr><op><expr><op><expr>
1.0<op><expr><op><expr><op><expr>
1.0-<expr><op><expr><op><expr>
1.0- sin(<expr>)<op><expr><op><expr>
1.0- sin(<var>)<op><expr><op><expr>
1.0- sin(x)<op><expr><op><expr>
1.0- sin(x)*<expr><op><expr>
1.0- sin(x)*<expr><op><expr><op><expr>
1.0- sin(x)*<pre-op>(<expr>)<op><expr><op><expr>
1.0- sin(x)*sin(<expr>)<op><expr><op><expr>
1.0- sin(x)*sin(<var>)<op><expr><op><expr>
1.0- sin(x)*sin(x)<op><expr><op><expr>
1.0- sin(x)*sin(x) -<expr><op><expr>
1.0- sin(x)*sin(x) - <pre-op>(<expr>)<op><expr>
1.0- sin(x)*sin(x) - sin(<expr>)<op><expr>
1.0- sin(x)*sin(x) - sin(<var>)<op><expr>
1.0- sin(x)*sin(x) - sin(x)<op><expr>
1.0- sin(x)*sin(x) - sin(x)*<expr>
1.0- sin(x)*sin(x) - sin(x)*<pre-op>(<expr>)
1.0- sin(x)*sin(x) - sin(x)*sin(<expr>)
1.0- sin(x)*sin(x) - sin(x)*sin(<var>)
1.0- sin(x)*sin(x) - sin(x)*sin(x)
```

```
<expr>::=<expr><op><expr>       (0)
        |(<expr><op><expr>)     (1)
        |<pre-op>(<expr>)       (2)
        |<var>                  (3)
<op>::=+          (0)
     | -          (1)
     | /          (2)
     | *          (3)
<pre-op>::= sin
<var>::= x        (0)
       | 1.0      (1)
```

GE

Genotype-phenotype mapping: remarks

- What about exhausting the genotype without getting a valid phenotype?
- Usually the worst fitness value is assigned to punish them
- Which are the values used as codons.
- They have to be large enough to be able to choose all the possible right hand sides, that is, their value should be greater than the maximum number of options for all the non-terminals
- Usually larger values are used and more than one different codon chose the same right hand side mimicking in this way the biological mechanism that preserves diversity. A typical maximum value is 256
- This mapping is deterministic

GE

Practical considerations

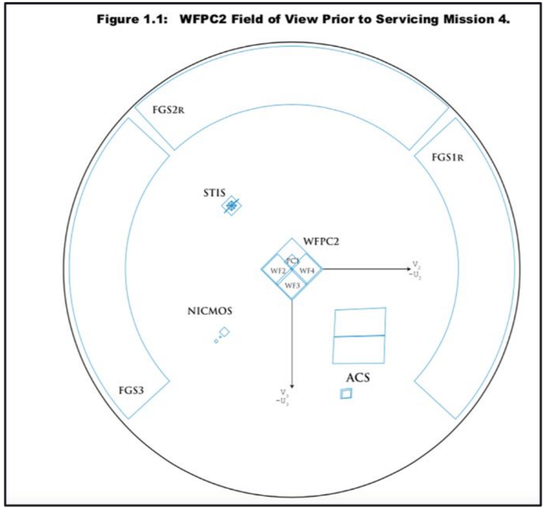
- GE is computational expensive mainly because of genotype-phenotype mapping:
    - It is itself expensive and
    - It is executed as many times as the fitness function.
- But GE is very expressive
    - Able to generate programs no matter the language
    - Compatible with semantic extensions (Attribute and Christiansen GE)
- To apply GE makes sense when
    - The problem is quite hard
    - The search space is huge

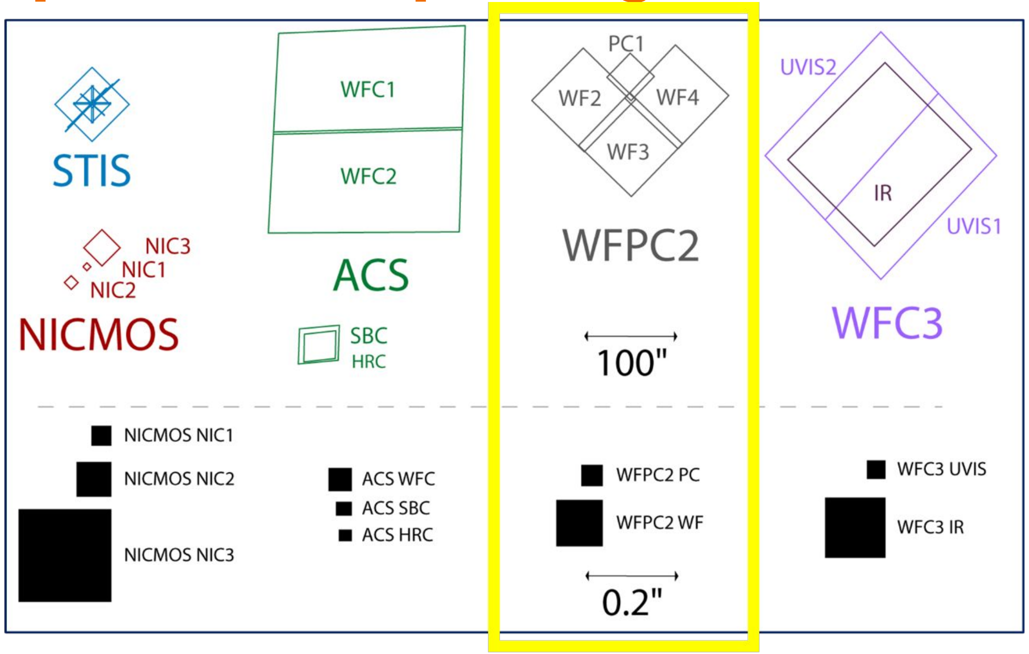
Astrometry and SR: once again XAI

Astronomy (among other Physics domains)

- Proposes mathematical models
- Should be comprehensible for researchers and other related domains
- Deep Learning (as an usually opaque ML model) lacks of explanations
- SR is more and more interesting both
  - From evolutionary approaches
  - From conventional approaches

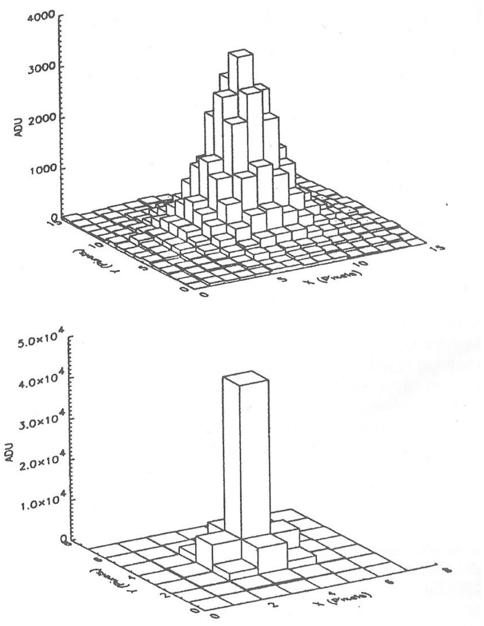
Centering of Hubble Space Telescope images

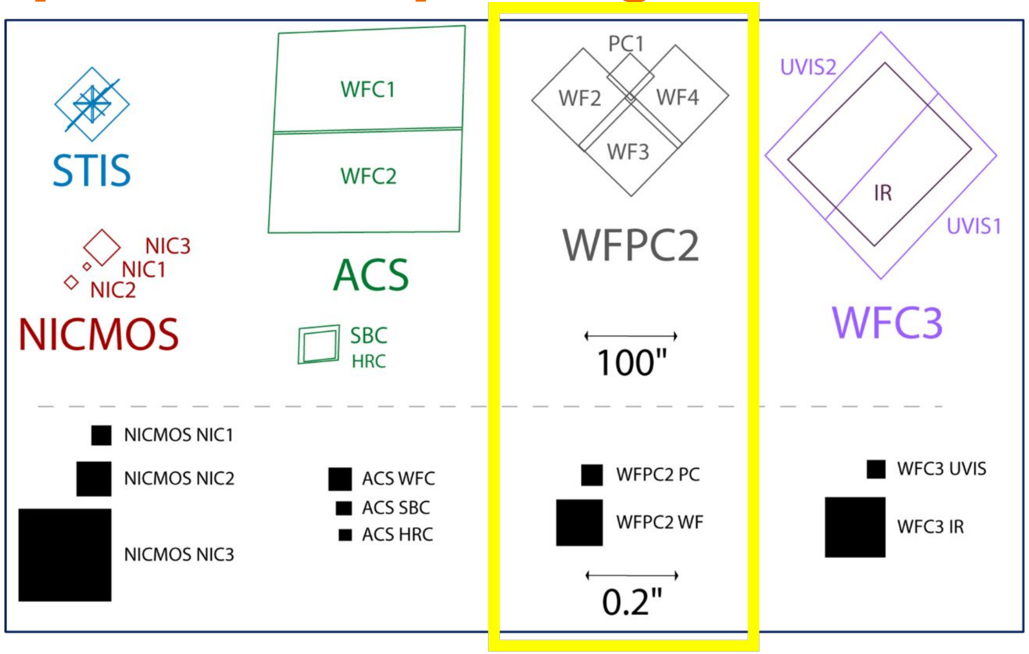




PC pix = 0.046"
WF pix = 0.1"

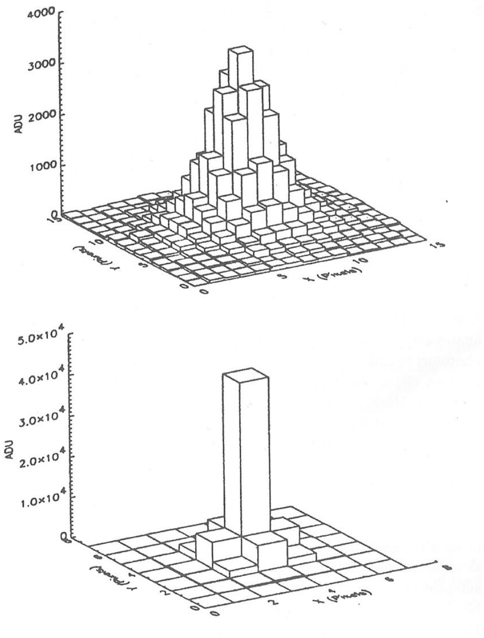
Centering of Hubble Space Telescope images





PC pix = 0.046"
WF pix = 0.1"

Centering of Hubble Space Telescope images



But, actually, simulated (synthetic) labeled datasets are used (for supervised ML purposes)

- Models of the PSF (or ***p***oint-***s***pread ***f***unction around a known center point) compatible con real images are used
- To induce the intra-pixel position of the center of the star
- Because the center itself is mainly unknown

State of the art

Classid methods that differ in the software and the *PSF* used for **centering**

1. **2D Gaussian :** elyptical *bidimensional Gaussian* profiles **+ centering** routines from **Lee & van Altena (**1983).

2. **Empirical PSFs :** effective *PSFs obtained from real observations* **+** hst1pass code from **Anderson & King** (2000), which has its own detection algorithm and provides **object centers**, magnitude, etc.

3. **TinyTim :** these *PSFs are optical models* obtained from HST TinyTim *simulator* (Krist et a. 2011) for each camera, chip and filter **+ DOLPHOT v2.0 package** (Dolphin 2000), which is used to determine the astrometric precision.

4. **PSFex :** developed by Bertin & Arnouts (2010), this code builds *empirical model for shift-variant PSFs* using *real observations* from globular cluster 47 Tucanae (NGC 104) **+ SExtractor**, which is used to measured object **centers**.

5. **Ideal PSF :** an *ideal PSF model* at the telescope diffraction limit is built, this is used to *deconvolve the image* in the Fourier domain, to *avoid noise amplification* an LP Butterworth filter is applied **+ centering** routines from **Lee & van Altena** (1983).

State of the art

Classic methods: performance

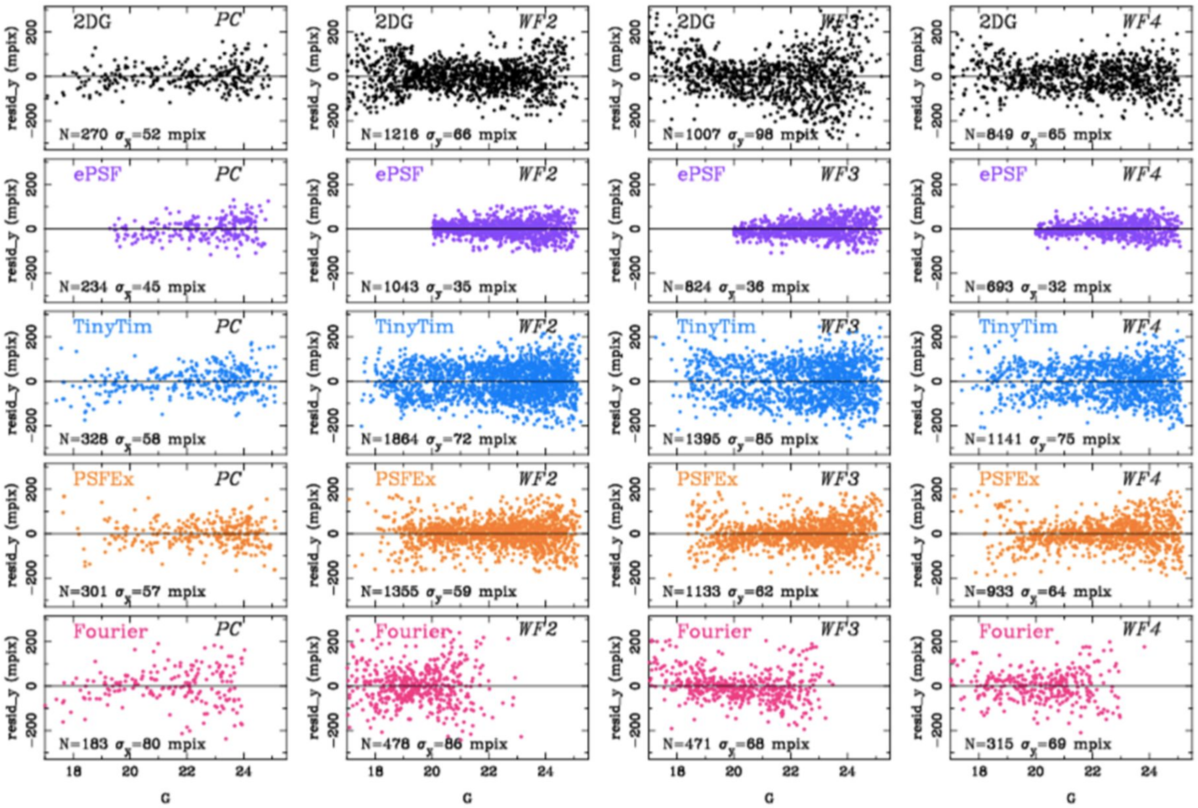


Figure 2. Residuals of the transformation of one F555W, 1400-sec exposure (PID 6114) into the 47 Tuc standard catalog as a function of G mangitude. Each centering algorithm and chip are specified. The number of stars participating in the solution and the standard error of the solution are also indicated for each case. The contribution of the catalog position errors to the error budget is ~ 31 mpix for the PC and 14 mpix for the WF.

State of the art

Deep Learning is applied by our authors

DL Roberto **Baena-Gallé, Terrence M. Girard, Dana Casetti-Dinescu**, Max Martone, Astrometric Centering of WFPC2/HST images with Deep Learning. **Deep learning** to learn the position **from datasets**

State of the art

Deep Learning performance

PC	2DG		hst1pass		VGG 34K		VGG 214K	
	x	y	x	y	x	y	x	y
simu_wfpc2 + ePSF	31.9	27.9	9.5	8.6	7.8	6.9	6.8	6.8
simu_wfpc2 + PSFex	15.0	12.0	47.7	43.7	6.4	6.4	5.4	5.4
skymaker + ePSF	36.3	32.1	8.7	8.5	9.3	8.1	7.5	6.4
skymaker + PSFex	17.3	13.6	48.5	41.7	8.5	7.5	6.8	6.3

WF2	2DG		hst1pass		VGG 34K		VGG 214K	
	x	y	x	y	x	y	x	y
simu_wfpc2 + ePSF	34.1	35.1	8.3	8.0	7.8	7.4	6.8	6.5
simu_wfpc2 + PSFex	12.3	13.7	13.6	14.3	6.7	6.0	5.0	5.0
skymaker + ePSF	41.1	44.3	9.1	9.0	8.7	8.8	7.4	7.2
skymaker + PSFex	13.4	15.8	20.8	16.2	8.3	7.4	5.9	5.9

Our proposal: first steps of GE

Where is the improvement in the state-of-the-art software?

- This proposal is a proof of concept: is at least GE as applicable as others?
    - Hypothesis: “yes it is”
    - Test:
        - Just to check the machine runs
        - To Evolve as few elements as possible
    - But
        - GE’s expressive power is not fully exploited but its computational bill is paid in full
- After this successful first step
    - GE is an up to date tool.
    - That will overcome the state of the art of SR in the domain
        - Because it does in general similar (hard/huge search spaces) domains

Our proposal: first steps of GE

Where is the improvement in the state-of-the-art software?

- But, right now, even with a limited expressive power, **we get from GE the complete PSF along with the center *to explain how we get it***

Our proposal: first steps of GE

Technology

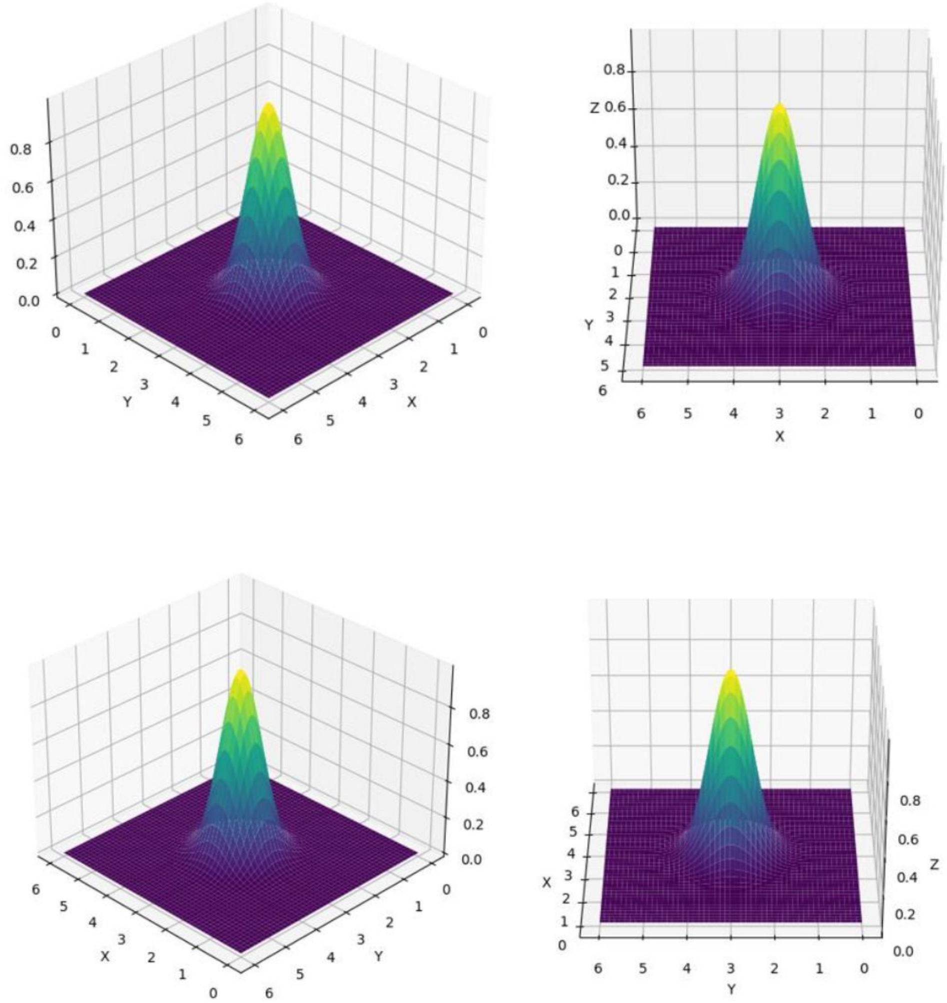
- PonyGE2 (https://github.com/PonyGE/PonyGE2)
  - Because it is open source
    - Fully configurable and adaptable
    - Extensible (see further research lines)
  - Python supported
    - Able to communicate with tons of libs
    - Compatible with cloud computing

Our proposal: PSF model

Circular Gaussian

$$G_{circ}(x,y) = S_0 + S_1 \exp\left\{-|w|\left[(x-x_0)^2 + (y-y_0)^2\right]\right\}$$

$S_0 = 0$
$S_1 = 1$
$x_0 = 3$
$y_0 = 3$
$w = 2$



$$G_{circ}(x,y) = S_0 + S_1 \exp\left\{-|w|\left[(x-x_0)^2 + (y-y_0)^2\right]\right\}$$

PSF model

Circular Gaussian grammar

```
<program> ::=
def PSF_estimada(S0,S1,x0,y0,w):
    return lambda x,y: S0 + S1*np.exp(-np.abs(w)*((x-(x0-1))**2+(y-(y0-1))**2))

S0 = 0.0<num><num><num>
S1 = 0.9<num><num><num>
x0 = 3.<num><num><num><num>
y0 = 3.<num><num><num><num>

<v_param5>

if (S1==0):
    S1 = 0.0001

mat = PSF_estimada(S0,S1,x0,y0,w)(*np.indices(data.shape))

<v_param5> ::= w = 1.<num_2_4><num><num><num> |
               w = 1.<num_5_7><num><num><num> |
               w = 1.<num_8_9><num><num><num> |
               w = 2.<num_0_2><num><num><num>

<num> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

<num_0_2> ::= 0 | 1 | 2

<num_2_4> ::= 2 | 3 | 4

<num_5_7> ::= 5 | 6 | 7

<num_8_9> ::= 8 | 9
```

Our proposal: PSF model

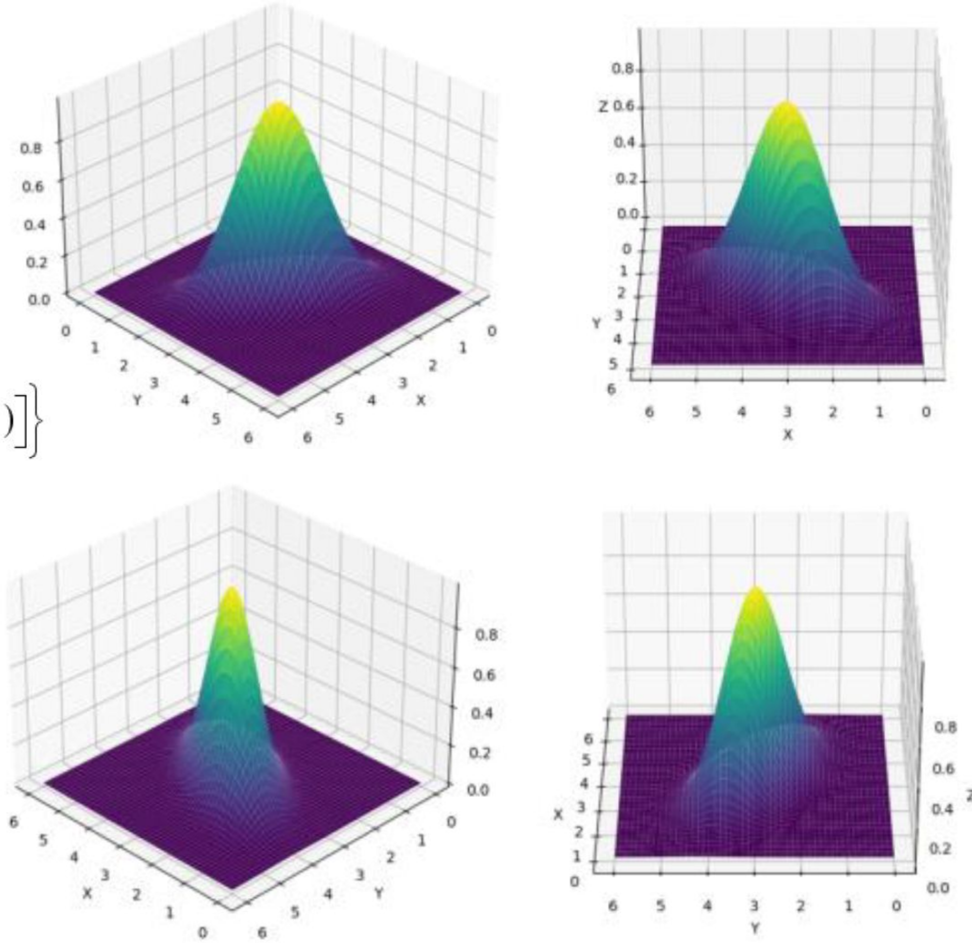
Elliptical Gaussian

$$G_{elliptical}(x,y) = S_0 + S_1 \exp\left\{-\frac{1}{2}\left[A(x-x_0)^2 + B(y-y_0)^2 + C(x-x_0)(y-y_0)\right]\right\}$$

$$A = \left(\frac{\cos\varphi}{\sigma_1}\right)^2 + \left(\frac{\sin\varphi}{\sigma_2}\right)^2 \qquad B = \left(\frac{\sin\varphi}{\sigma_1}\right)^2 + \left(\frac{\cos\varphi}{\sigma_2}\right)^2$$

$$C = 2\sin\varphi\cos\varphi\left(\frac{1}{\sigma_1^2} - \frac{1}{\sigma_2^2}\right)$$

$S_0 = 0$
$S_1 = 1$
$x_0 = 3$
$y_0 = 3$
phi = 45
$sigma_1 = 0.5$
$sigma_2 = 1$



$$G_{elliptical}(x,y) = S_0 + S_1 \exp\left\{-\frac{1}{2}\left[A(x-x_0)^2 + B(y-y_0)^2 + C(x-x_0)(y-y_0)\right]\right\}$$

Our proposal: PSF model

$$A = \left(\frac{\cos\varphi}{\sigma_1}\right)^2 + \left(\frac{\sin\varphi}{\sigma_2}\right)^2 \qquad B = \left(\frac{\sin\varphi}{\sigma_1}\right)^2 + \left(\frac{\cos\varphi}{\sigma_2}\right)^2$$

Elliptical Gaussian grammar

$$C = 2\sin\varphi\cos\varphi\left(\frac{1}{\sigma_1^2} - \frac{1}{\sigma_2^2}\right)$$

```
<program> ::=
def PSF_estimada(S0,S1,x0,y0,sig_1,sig_2,phi):
    A = (np.cos(phi)/sig_1)**2. + (np.sin(phi)/sig_2)**2.
    B = (np.sin(phi)/sig_1)**2. + (np.cos(phi)/sig_2)**2.
    C = 2.0*np.sin(phi)*np.cos(phi)*(1./(sig_1**2.)-1./(sig_2**2.))
    return lambda x,y: S0 + S1*np.exp(-0.5*(A*((x-(x0-1))**2)+B*((y-(y0-1))**2)+C*(x-(x0-1))*(y-(y0-1))))

S0 = 0.0<num><num><num>
S1 = 0.9<num><num><num>
x0 = 3.<num><num><num><num>
y0 = 3.<num><num><num><num>

<v_param5>

<v_param6>

<v_param7>
```

PSF model

Elliptical Gaussian grammar

$$G_{elliptical}(x,y) = S_0 + S_1 \exp\left\{-\frac{1}{2}\left[A(x-x_0)^2 + B(y-y_0)^2 + C(x-x_0)(y-y_0)\right]\right\}$$

$$A = \left(\frac{\cos\varphi}{\sigma_1}\right)^2 + \left(\frac{\sin\varphi}{\sigma_2}\right)^2 \qquad B = \left(\frac{\sin\varphi}{\sigma_1}\right)^2 + \left(\frac{\cos\varphi}{\sigma_2}\right)^2$$

$$C = 2\sin\varphi\cos\varphi\left(\frac{1}{\sigma_1^2} - \frac{1}{\sigma_2^2}\right)$$

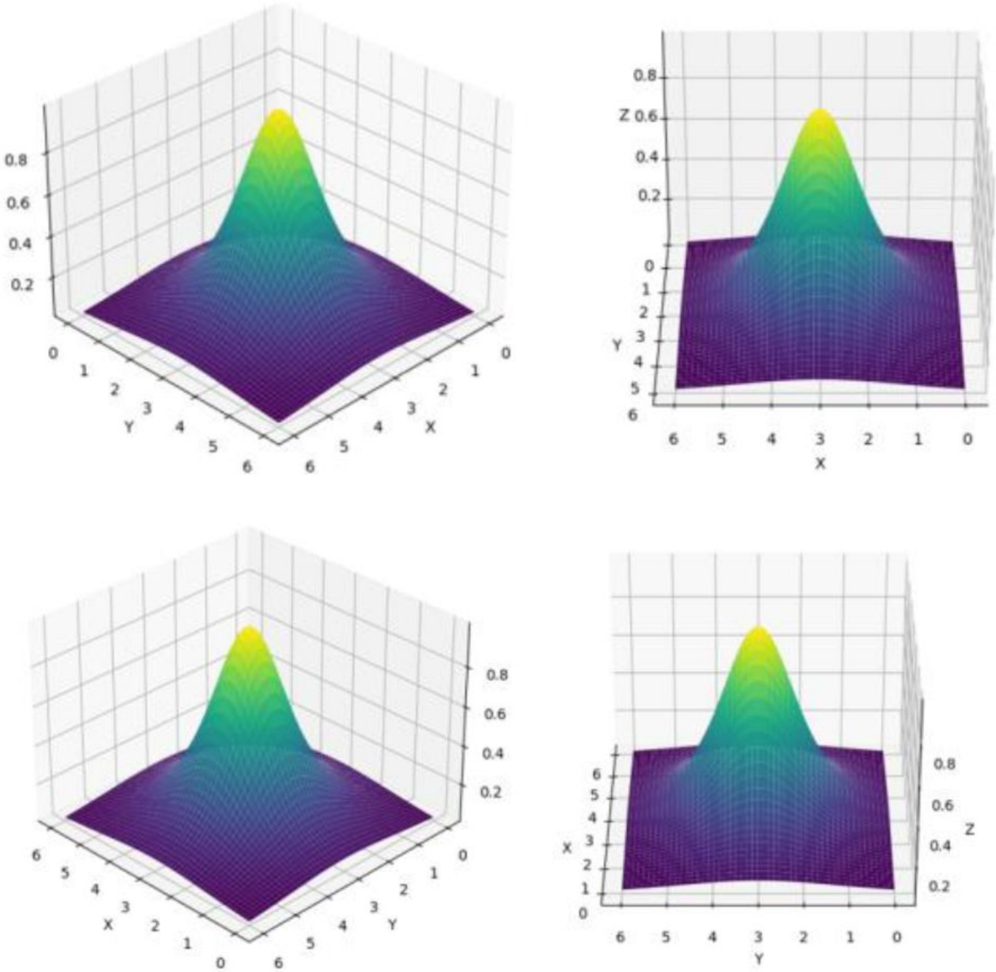
```
if (S1==0):
    S1 = 0.0001
if (phi "<" -90):
    phi += 180
if (phi ">" 90):
    phi -= 180

mat = PSF_estimada(S0,S1,x0,y0,sig_1,sig_2,phi)(*np.indices(data.shape))

<v_param5> ::= sig_1 = 0.4<num_6_9><num><num> |
               sig_1 = 0.5<num_0_4><num><num> |
               sig_1 = 0.5<num_5_9><num><num> |
               sig_1 = 0.6<num_0_3><num><num>

<v_param6> ::= sig_2 = 0.4<num_6_9><num><num> |
               sig_2 = 0.5<num_0_4><num><num> |
               sig_2 = 0.5<num_5_9><num><num> |
               sig_2 = 0.6<num_0_3><num><num>

<v_param7> ::= phi = -<num><num>.<num><num><num><num> |
               phi = <num><num>.<num><num><num><num>

<num> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
<num_0_3> ::= 0 | 1 | 2 | 3
<num_0_4> ::= 0 | 1 | 2 | 3 | 4
<num_5_9> ::= 5 | 6 | 7 | 8 | 9
<num_6_9> ::= 6 | 7 | 8 | 9
```

Our proposal: PSF model

Circular Moffat

$$M_{circular}(x,y) = S_0 + \frac{S_1}{\left[1 + \frac{(x-x_0)^2 + (y-y_0)^2}{\alpha^2}\right]^{\beta}}$$

$S_0 = 0$
$S_1 = 1$
$x_0 = 3$
$y_0 = 3$
alpha = 1.5
Beta = 1.6



$$M_{circular}(x,y) = S_0 + \frac{S_1}{\left[1 + \frac{(x-x_0)^2 + (y-y_0)^2}{\alpha^2}\right]^{\beta}}$$

Our proposal: PSF model

Circular Moffat

grammar

```
<program> ::=
def PSF_estimada(S0,S1,x0,y0,alpha,beta):
    return lambda x,y: S0 + S1/((1.+(((x-(x0-1))**2+(y-(y0-1))**2)/alpha**2.))**beta)

S0 = 0.0<num><num><num>
S1 = 0.9<num><num><num>
x0 = 3.<num><num><num><num>
y0 = 3.<num><num><num><num>

<v_param5>

beta = <num><num>.<num><num><num><num>

if (S1==0):
    S1 = 0.0001

mat = PSF_estimada(S0,S1,x0,y0,alpha,beta)(*np.indices(data.shape))

<v_param5> ::= alpha = 0.<num_5_9><num><num><num> |
               alpha = 1.<num_0_4><num><num><num> |
               alpha = 1.<num_5_9><num><num><num>

<num> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

<num_0_4> ::= 0 | 1 | 2 | 3 | 4

<num_5_9> ::= 5 | 6 | 7 | 8 | 9
```

Our proposal: PSF model

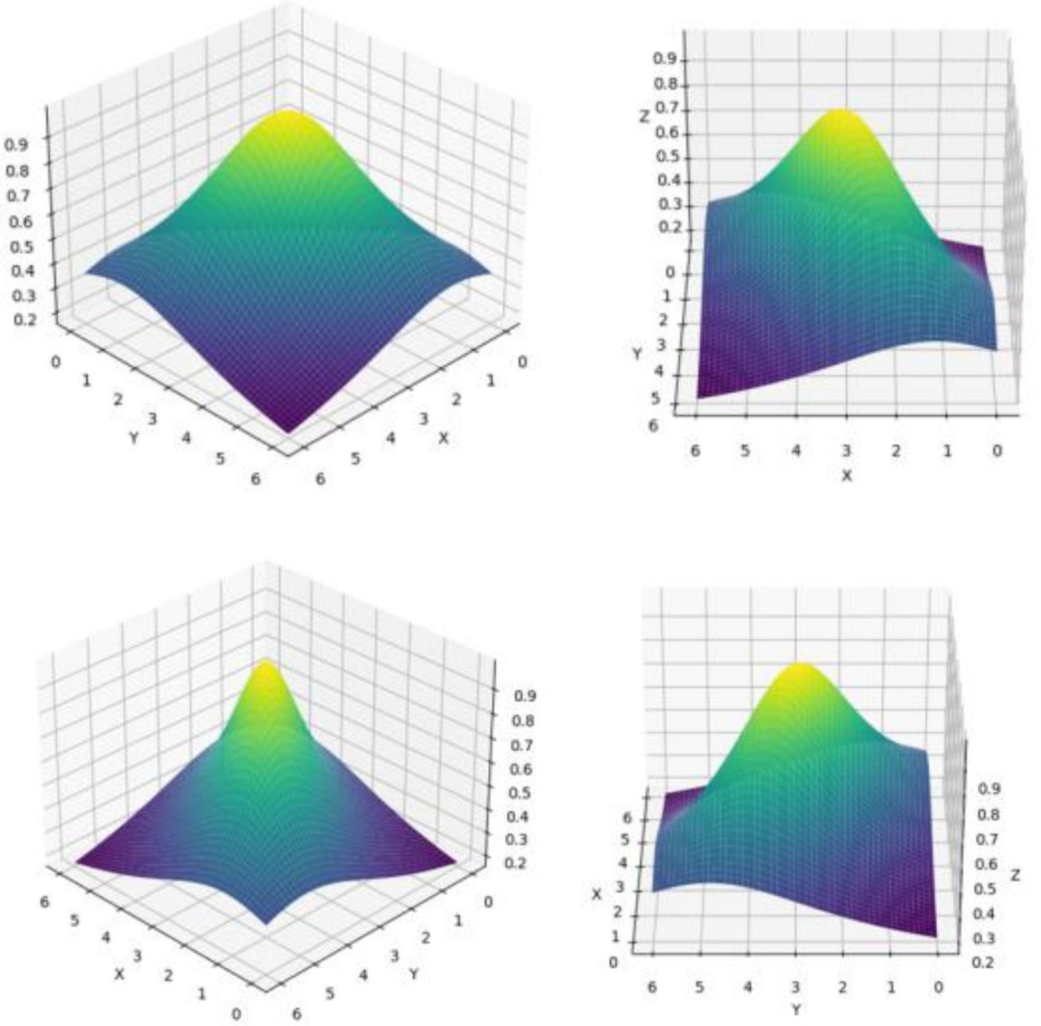
Elliptical Moffat

$$M_{elliptical}(x,y) = S_0 + \frac{S_1}{\left[1 + A(x-x_0)^2 + B(y-y_0)^2 + C(x-x_0)(y-y_0)\right]^{\beta}}$$

$$A = \left(\frac{\cos\varphi}{\alpha_1}\right)^2 + \left(\frac{\sin\varphi}{\alpha_2}\right)^2 \qquad B = \left(\frac{\sin\varphi}{\alpha_1}\right)^2 + \left(\frac{\cos\varphi}{\alpha_2}\right)^2$$

$$C = 2\sin\varphi\cos\varphi\left(\frac{1}{\alpha_1^2} - \frac{1}{\alpha_2^2}\right)$$

$S_0 = 0$
$S_1 = 1$
$x_0 = 3$
$y_0 = 3$
$alpha_1 = 1$
$alpha_2 = 2$
phi = 45
beta = 0.6



$$M_{elliptical}(x,y) = S_0 + \frac{S_1}{\left[1 + A(x-x_0)^2 + B(y-y_0)^2 + C(x-x_0)(y-y_0)\right]^{\beta}}$$

Our proposal: PSF model

$$A = \left(\frac{\cos\varphi}{\alpha_1}\right)^2 + \left(\frac{\sin\varphi}{\alpha_2}\right)^2 \qquad B = \left(\frac{\sin\varphi}{\alpha_1}\right)^2 + \left(\frac{\cos\varphi}{\alpha_2}\right)^2$$

Elliptical Moffat grammar

$$C = 2\sin\varphi\cos\varphi\left(\frac{1}{\alpha_1^2} - \frac{1}{\alpha_2^2}\right)$$

```
<program> ::=
def PSF_estimada(S0,S1,x0,y0,alpha_1,alpha_2,phi,beta):
    phi = phi*np.pi/180
    A = (np.cos(phi)/alpha_1)**2. + (np.sin(phi)/alpha_2)**2.
    B = (np.sin(phi)/alpha_1)**2. + (np.cos(phi)/alpha_2)**2.
    C = 2.0*np.sin(phi)*np.cos(phi)*(1./alpha_1**2. - 1./alpha_2**2.)
    return lambda x,y: S0 + S1/((1.+ A*((x-(x0-1))**2) + B*((y-(y0-1))**2) + C*(x-(x0-1))*(y-(y0-1)))**beta)

S0 = 0.0<num><num><num>
S1 = 0.9<num><num><num>
x0 = 3.<num><num><num><num>
y0 = 3.<num><num><num><num>
alpha_1 = <num>.<num><num><num><num>
alpha_2 = <num>.<num><num><num><num>

```

$$M_{elliptical}(x,y) = S_0 + \frac{S_1}{\left[1 + A(x-x_0)^2 + B(y-y_0)^2 + C(x-x_0)(y-y_0)\right]^{\beta}}$$

$$A = \left(\frac{\cos\varphi}{\alpha_1}\right)^2 + \left(\frac{\sin\varphi}{\alpha_2}\right)^2 \qquad B = \left(\frac{\sin\varphi}{\alpha_1}\right)^2 + \left(\frac{\cos\varphi}{\alpha_2}\right)^2$$

$$C = 2\sin\varphi\cos\varphi\left(\frac{1}{\alpha_1^2} - \frac{1}{\alpha_2^2}\right)$$

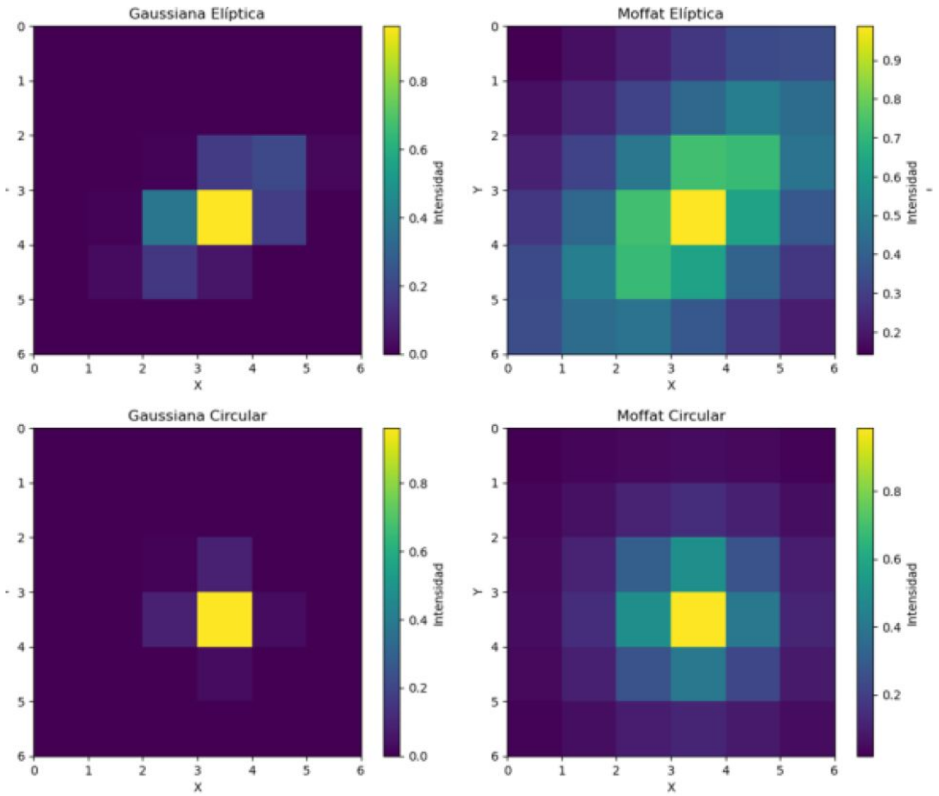
Our proposal: PSF model

Elliptical Moffat grammar

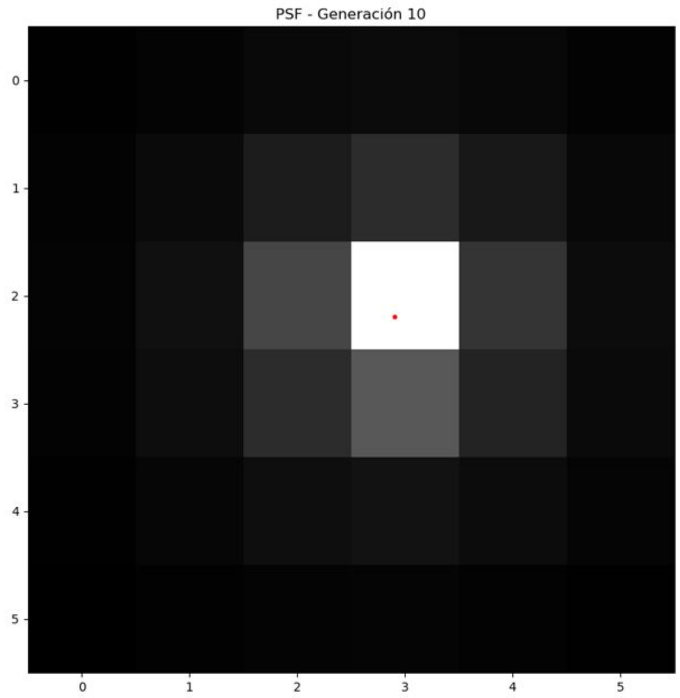
```
<v_param7>

beta = <num><num>.<num><num><num><num>

if (phi "<" -90):
    phi += 180
if (phi ">" 90):
    phi -= 180
if (beta==0):
    beta = 0.0001

mat = PSF_estimada(S0,S1,x0,y0,alpha_1,alpha_2,phi,beta)(*np.indices(data.shape))

<v_param7> ::= phi = -<num><num>.<num><num><num><num> |
               phi = <num><num>.<num><num><num><num>

<num> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
```

Our proposal: PSF model

Comparing all of them



Evolution example



$$0-M_{Circ}(x,y)=0.0137+\frac{0.97}{[1+\frac{(x-3.2441)^2+(y-3.8083)^2}{1.0324}]^{3.0511}}$$

$$1-M_{Circ}(x,y)=0.0643+\frac{0.9359}{[1+\frac{(x-3.1969)^2+(y-3.8316)^2}{1.9516}]^{5.1896}}$$

$$2-M_{Circ}(x,y)=0.0434+\frac{0.9523}{[1+\frac{(x-3.2122)^2+(y-3.8567)^2}{1.6448}]^{4.6784}}$$

$$3-M_{Circ}(x,y)=0.0137+\frac{0.9767}{[1+\frac{(x-3.1447)^2+(y-3.9583)^2}{1.0623}]^{2.0511}}$$

$$4-M_{Circ}(x,y)=0.0137+\frac{0.9767}{[1+\frac{(x-3.1447)^2+(y-3.9583)^2}{1.0623}]^{2.0511}}$$

$$5-M_{Circ}(x,y)=0.013+\frac{0.9757}{[1+\frac{(x-3.1852)^2+(y-3.9583)^2}{1.0623}]^{2.0511}}$$

$$6-M_{Circ}(x,y)=0.0187+\frac{0.905}{[1+\frac{(x-3.1464)^2+(y-3.9583)^2}{1.0623}]^{2.0517}}$$

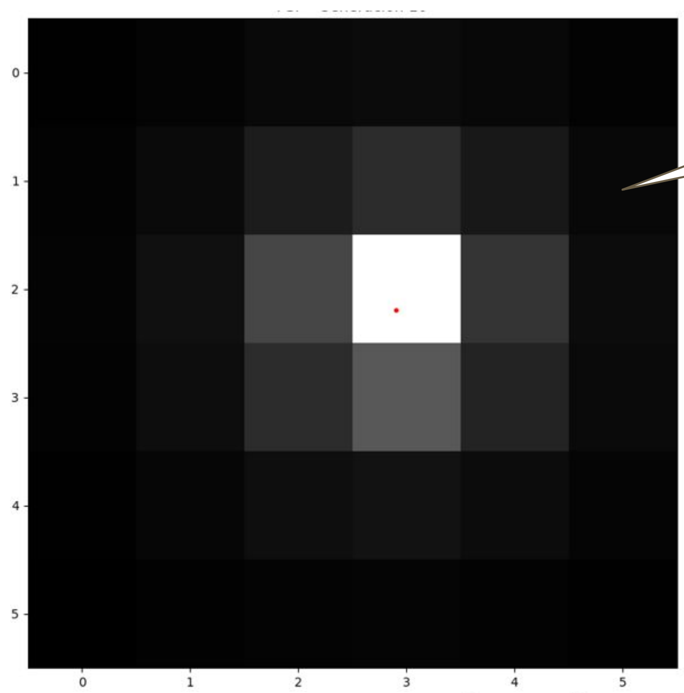
$$7-M_{Circ}(x,y)=0.014+\frac{0.9767}{[1+\frac{(x-3.1847)^2+(y-3.9282)^2}{0.5633}]^{1.0511}}$$

$$8-M_{Circ}(x,y)=0.0117+\frac{0.9787}{[1+\frac{(x-3.1878)^2+(y-3.9083)^2}{0.5825}]^{1.0555}}$$

$$9-M_{Circ}(x,y)=0.0037+\frac{0.9717}{[1+\frac{(x-3.1974)^2+(y-3.9085)^2}{0.5628}]^{1.0515}}$$

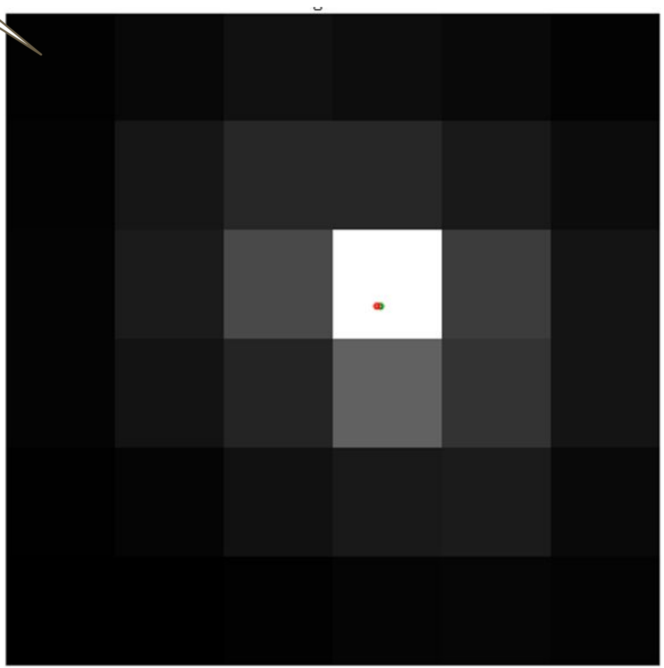
$$10-M_{Circ}(x,y)=0.0035+\frac{0.9715}{[1+\frac{(x-3.1974)^2+(y-3.9086)^2}{0.5628}]^{1.0518}}$$

Evolution example



Predicted

Real



dif: 27 mpix

$$M_{Circ}(x,y) = 0.0035 + \frac{0.9715}{\left[1+\frac{(x-3.1974)^2+(y-3.9086)^2}{0.5628}\right]^{1.0518}}$$

$(x_{est}, y_{est}) = (3.1974, 3.9086)$

$(x_{real}, y_{real}) = (3.1991, 3.9421)$

Our proposal: further technical details

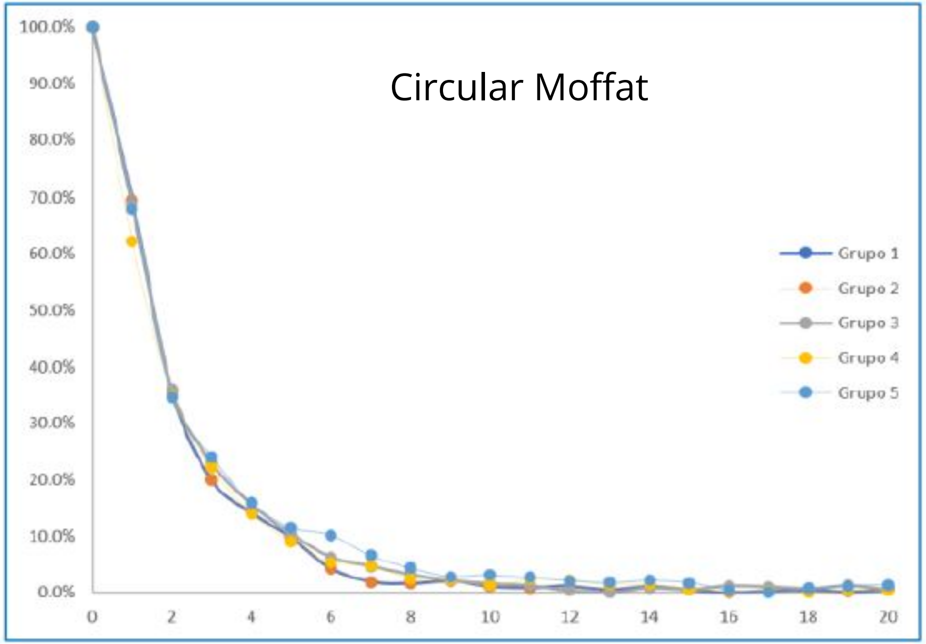
GE (hyper)parameters

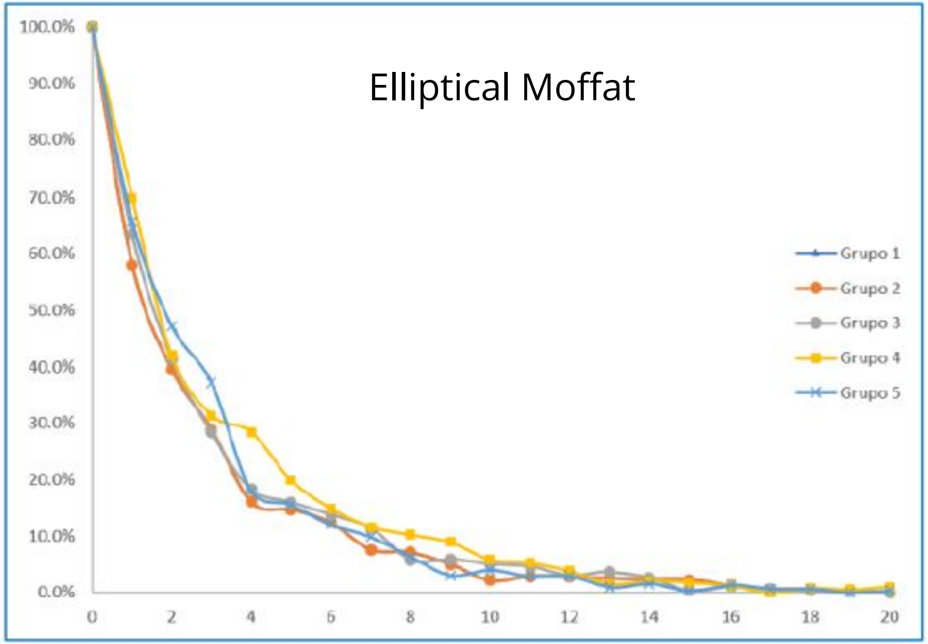
- Groups to be compared

Grupo	initialisation	crossover	mutation
Grupo 1	rvd	fixed_twopoint	int_flip_per_ind
Grupo 2	uniform_genome	fixed_twopoint	int_flip_per_ind
Grupo 3	uniform_tree	subtree	subtree
Grupo 4	PI_grow	subtree	subtree
Grupo 5	rhh	subtree	subtree

Our proposal: further technical details

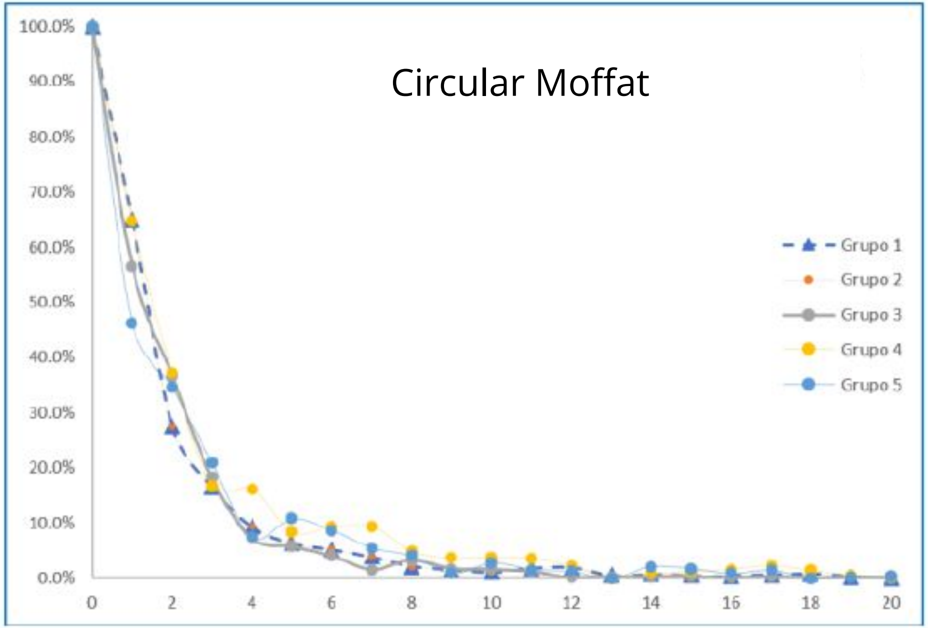
GE (hyper)parameters (Center distance - target/result - vs generation)

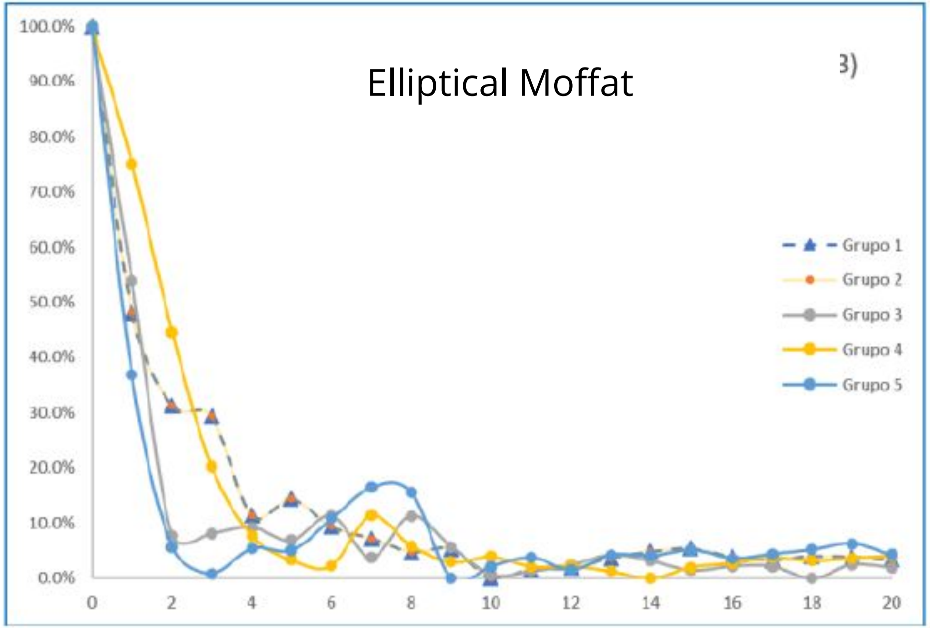




Our proposal: further technical details

GE (hyper)parameters (Center distance - target/result - vs generation in WF)





Our proposal: further technical details

GE (hyper)parameters

	2D Elliptical Gaussian				2D Moffat function			
	PC		WF		PC		WF	
Cost function	**Precision**	**Std dev**	**Precision**	**Std dev**	**Precision**	**Std dev**	**Precision**	**Std dev**
Cosine	52	23	139	34	50	24	123	36
Euclidean	53	23	145	35	52	24	129	41
Kendall	102	88	140	81	91	84	127	75
Manhattan	60	26	165	51	65	34	101	40
Pearson	52	24	139	34	51	21	120	32
Spearman	109	100	161	87	105	94	153	78
Total	**71**	**63**	**148**	**59**	**69**	**60**	**125**	**56**

Table 1: Summary of Experiment 1 using both PSF functions.

Our proposal: further technical details

GE (hyper)parameters

	Scenarios	PC Best	WF Best
Generations	10, 15 and 20	20	20
Population	100, 500, 1000	1000	500
Initialization	RVD / uniform_genome	uniform_genome	RVD
Selection type	Tournament / Truncation	Tournament	Tournament
- Tournament Size	1, 5 or 10	5	10
- Selection Proportion	0.2, 0.5, or 0.8	No trunc.	No trunc.
Crossover type	(fixed or variable) x (onepoint or twopoints)	variable_onepoint	fixed_twopoint
- Crossover probability	0.2, 0.5, and 0.8	0.2	0.2
Mutation type	Per individual / per codon	Per codon	Per individual
- Mutation events	fixed (1, 5, 10) or Per codon prob (0.25, 0.50, or 0.75)	Per codon fixed	Per ind fixed (5)
Elite size	1, 5, 10 and 20	1	10

Table 2: Experiment 2: Scenarios and best parameters found.

Our proposal: performance

Results :

- Elliptical Moffat is in general best
  - PC with Pearson distance
  - WF with Manhattan distance
- Let's see just an example

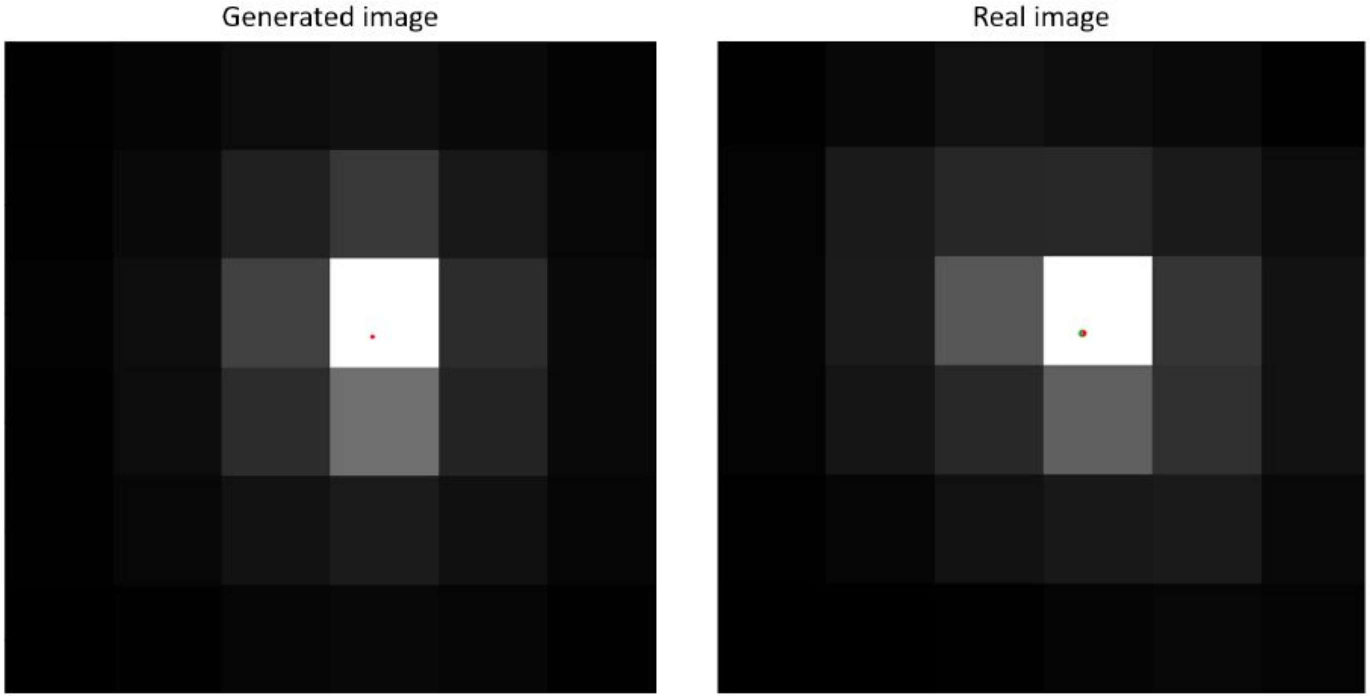


Figure 5: Model generated for a single evolutionary process, with generated image (left) and original image (right). Green dot is the real center $(x_{real}, y_{real}) = (3.2004, 3.8636)$**, red dot is the estimated one** $(x_{est}, y_{est}) = (3.2164, 3.8935)$**. Precision for this specific case is 34 mpix.**

$$M(x,y) = 0.9221 + \frac{0.9452}{[1 + 3.1692(x - 3.2164)^2 + 3.2790(y - 3.8935)^2 + 1.3774(x - 3.2164)(y - 3.8935)]^{1.0146}}$$

Our proposal

Performance:

Acceptable precision

Biases

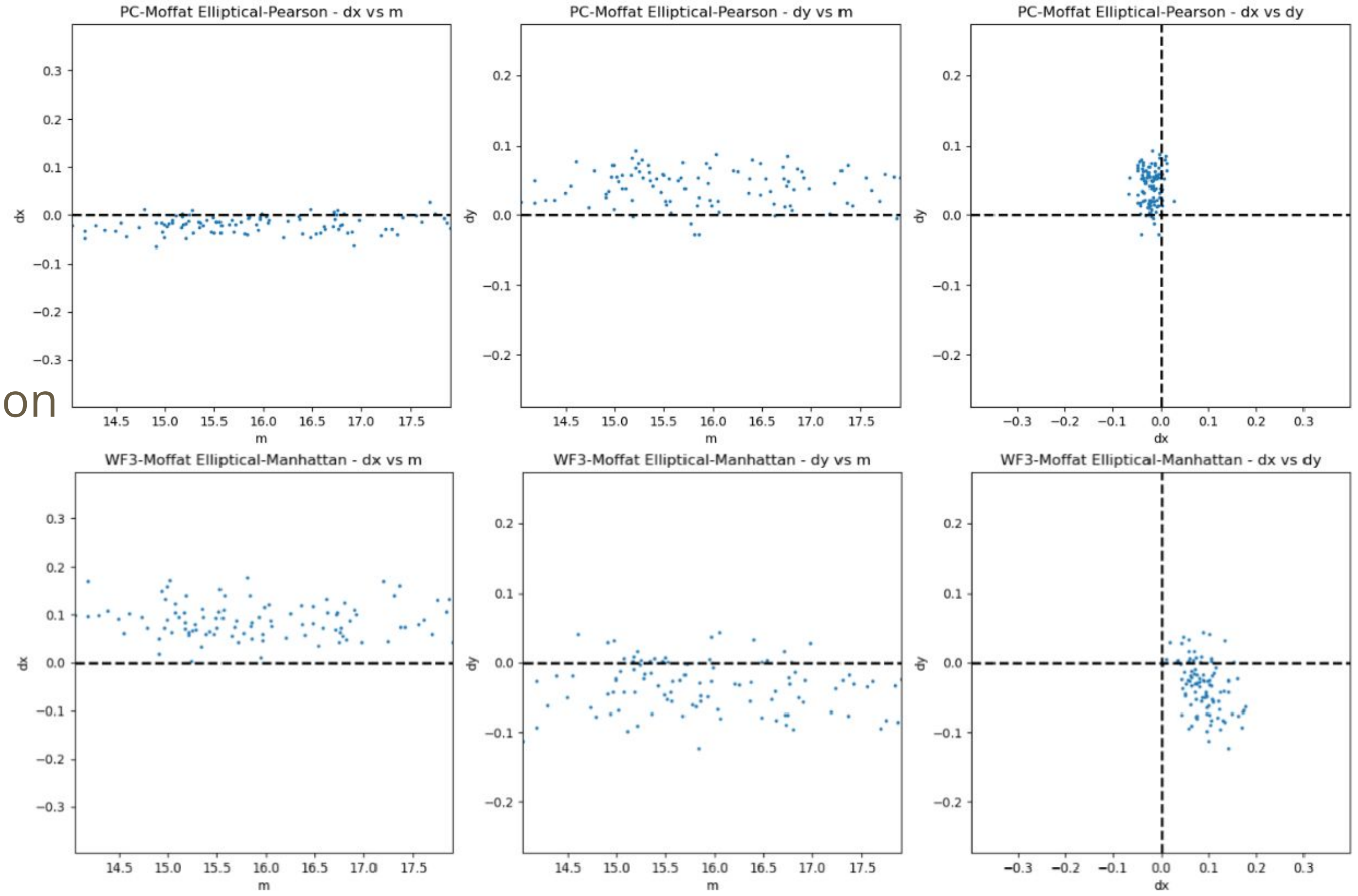


Figure 4: Bias profile for PC using a Pearson fitness function (top), and for WF using Manhattan (bottom). Left and middle columns: residuals in x- and y- w.r.t. the star magnitude. Right panel: residuals distribution w.r.t. (x,y) position.

Our proposal: further research lines

Advanced by the logical next steps:

- More flexible expression (just leting the evolution choose how much Moffat / Gaussian PSF is biases are mitigated and precision is improved)
- Full flexible expressions
  - Will generate possible new knowledge
  - We can programmatically design a PSF that takes flexibly takes into account researchers conditions
    - Periferical regions
    - Combinations of light sources

Thank you very much