Adquisición y procesamiento de imágenes de alta cadencia para la generación de curvas de luz en el contexto del proyecto TAOS-II

Tutores: Joel Castro y Mauricio Reyes
Estudiante: Antonio De La Cruz Carrillo

Motivación: TAOS-II

Imágenes de alta cadencia.

Distorsiones debido a la variación atmosférica y la naturaleza del CCD.

Bajo SNR (*signal to noise ratio*).

Necesitamos un procesamiento automático.

Fotografía: propia
Fondo: European Southern Observatory

TAOS-II

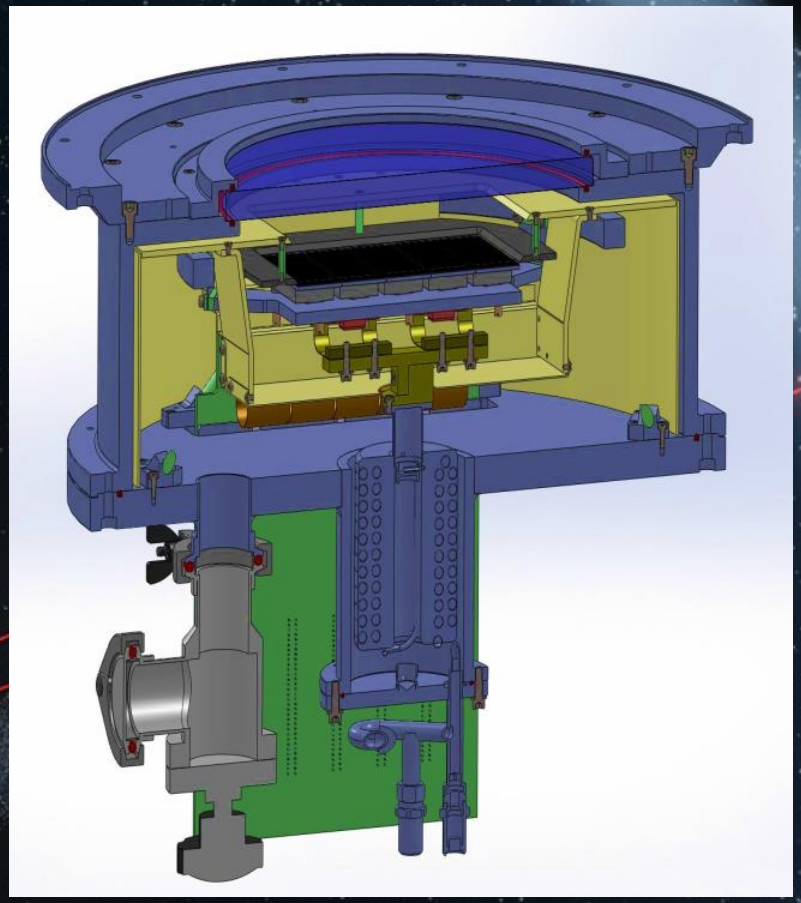
3 telescopios.

Diámetro de 1.3 m.

Campo de 2.3° cuadrados.

10 000 estrellas a 20 Hz.

250 TB por noche.



CfA

YONSEI UNIVERSITY 1885 연세대학교

學大央中立國

Lawrence Livermore National Laboratory

Imagen: Joel Castro

Fondo: European Southern Observatory

Objetivos generales

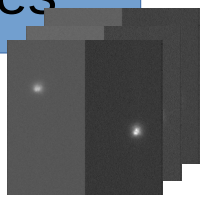
- Mejorar el SNR en las curvas de luz.
- Mejorar la resolución en imagen limitada por escala de placa.

Objetivos personales

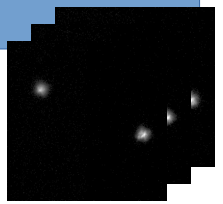
- Conocer el funcionamiento del CCD para el proyecto TAOS-II.
- Crear un software que permita generar una curva de luz a pesar de las aberraciones en las imágenes capturadas por el CCD.

Metodología

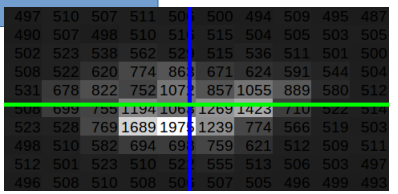
Adquirir las imágenes



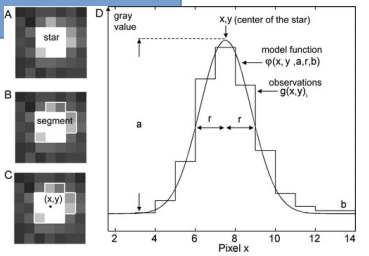
Reducción



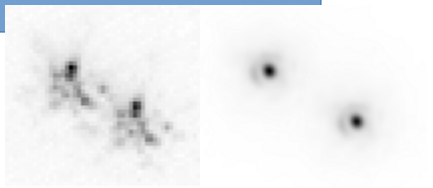
Centroide



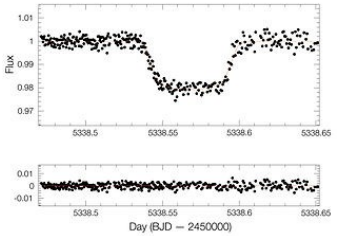
Fotometría PSF



Lucky imaging

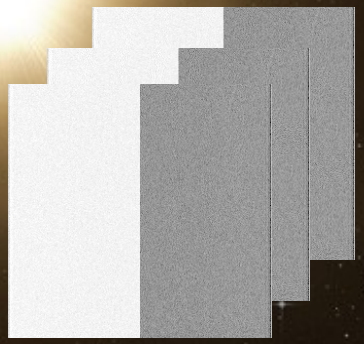


Encontramos su ubicación temporal

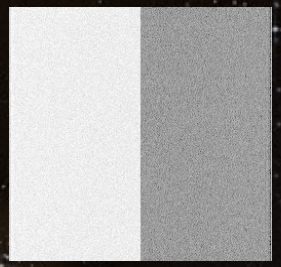


Reducción

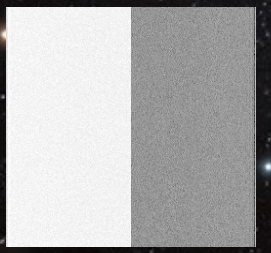
$$Z_{xyt} = \frac{I_{xyt} - B_{M_{xy}} - D_{M_{xy}}}{F_{N_{xy}}}$$



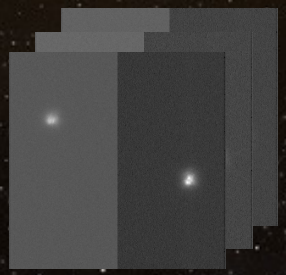
dark

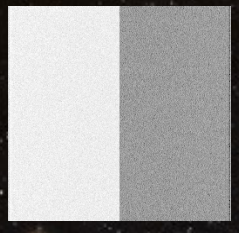


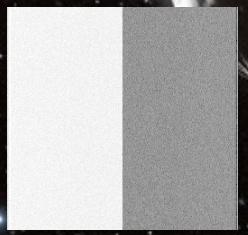
bias
maestro

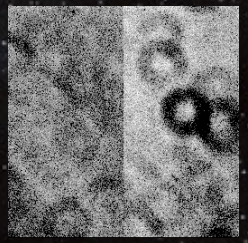


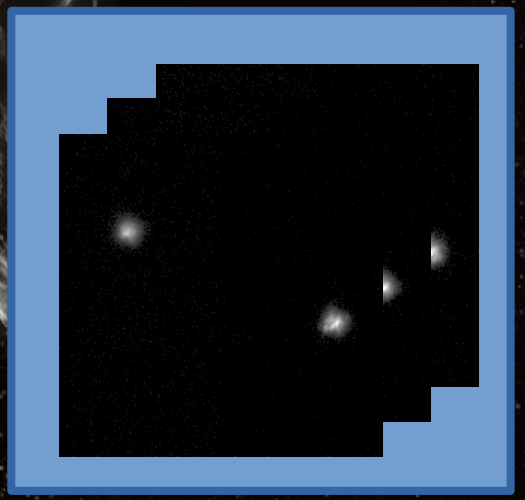
dark
maestro











Imágenes: Joel Castro, Antonio De La Cruz
Fondo: European Southern Observatory

Centroide

- Se toman los valores de una estrella en un área definida.
- Por descomposición geométrica:

$$x_c = \frac{\sum_i^n x_i v_i}{\sum_i^n v_i}, \qquad y_c = \frac{\sum_i^n y_i v_i}{\sum_i^n v_i}$$

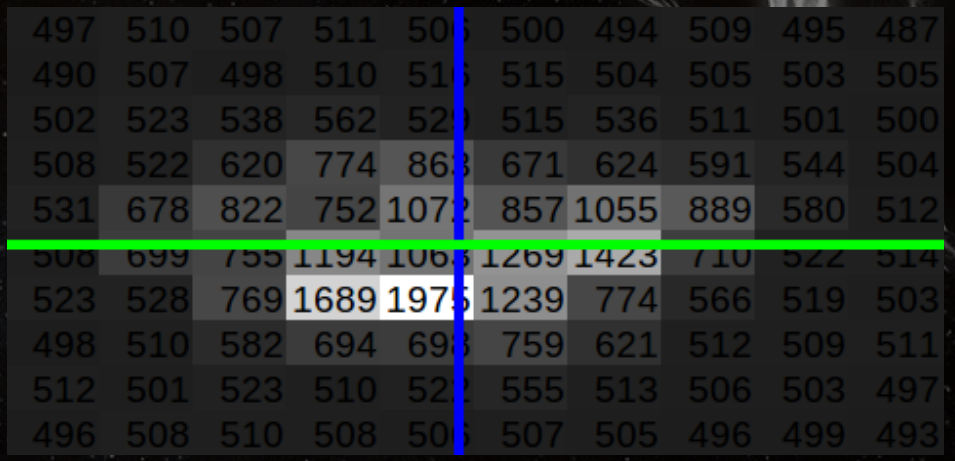


Imagen: Joel Castro, Antonio De La Cruz
Fondo: European Southern Observatory

Fotometría PSF

- *Point spread function*.
- Un criterio posible para *lucky imaging*.
- Se utiliza para valorar el estado de un *frame*.
- Se deben valorar en ambas dimensiones.

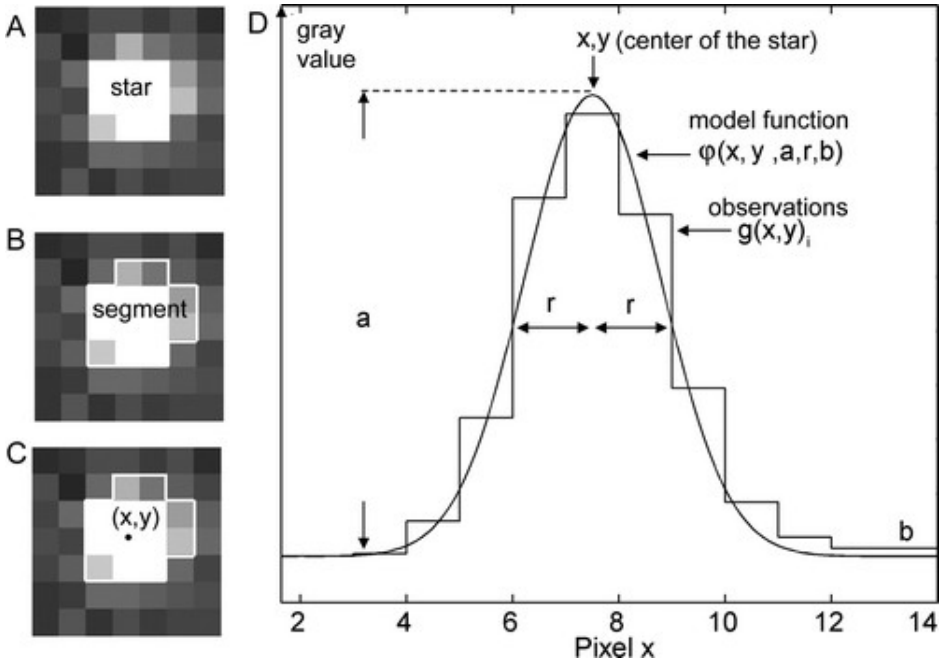
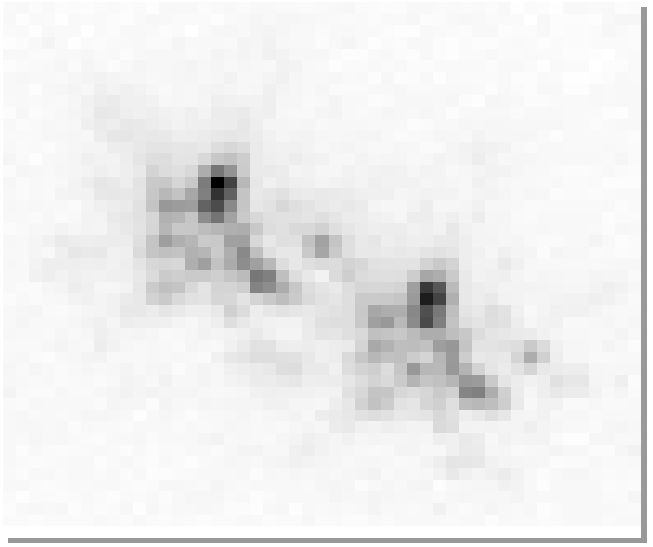


Imagen: American Society of Civil Engineers

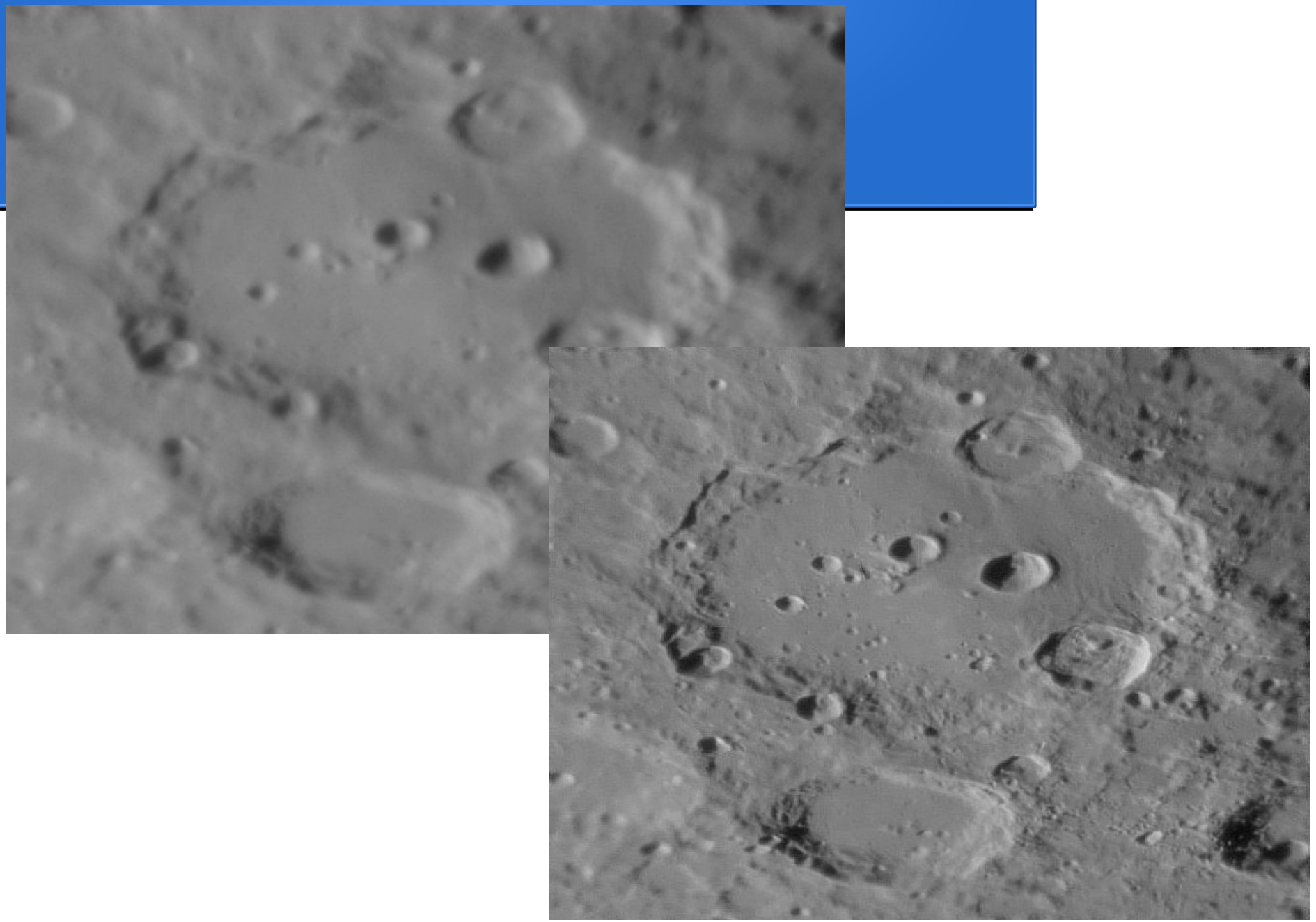
Georgia State University, M. Olmo R. Nave, *HyperPhysics*, 2017.
http://hyperphysics.phy-astr.gsu.edu/hbasees/hframe.html

Lucky imaging

- Seleccionar las mejores imagenes por PSF.
- Obtener los valores de las estrellas.
- Generar una curva de luz.



University of Cambridge, Institute of Astonomy, *Lucky Imaging*, 28 de septiembre de 2016. http://www.ast.cam.ac.uk/research/lucky



University of Cambridge, Institute of Astonomy, *Lucky Imaging*, 28 de septiembre de 2016.
http://www.ast.cam.ac.uk/research/lucky

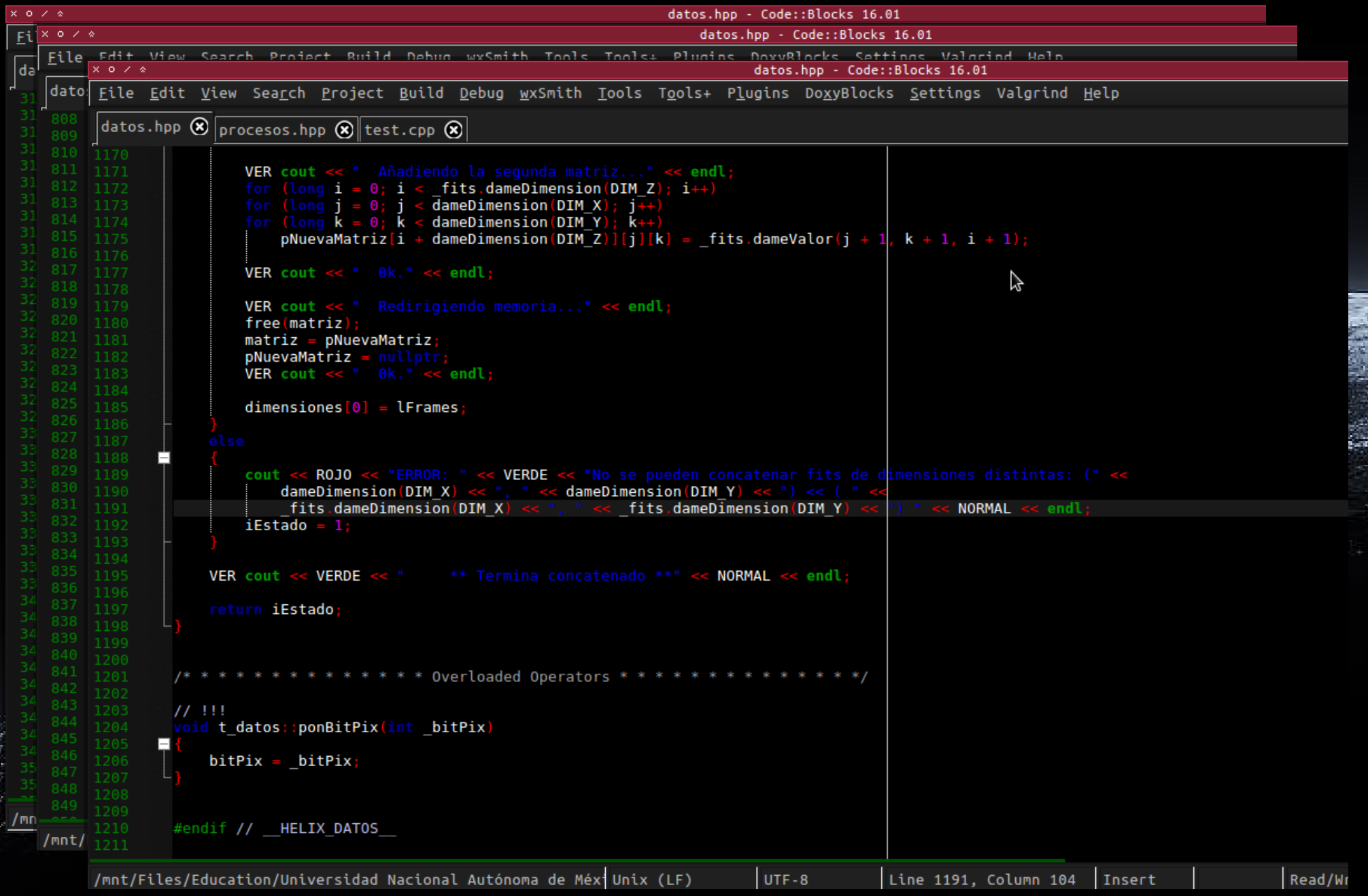
El programa

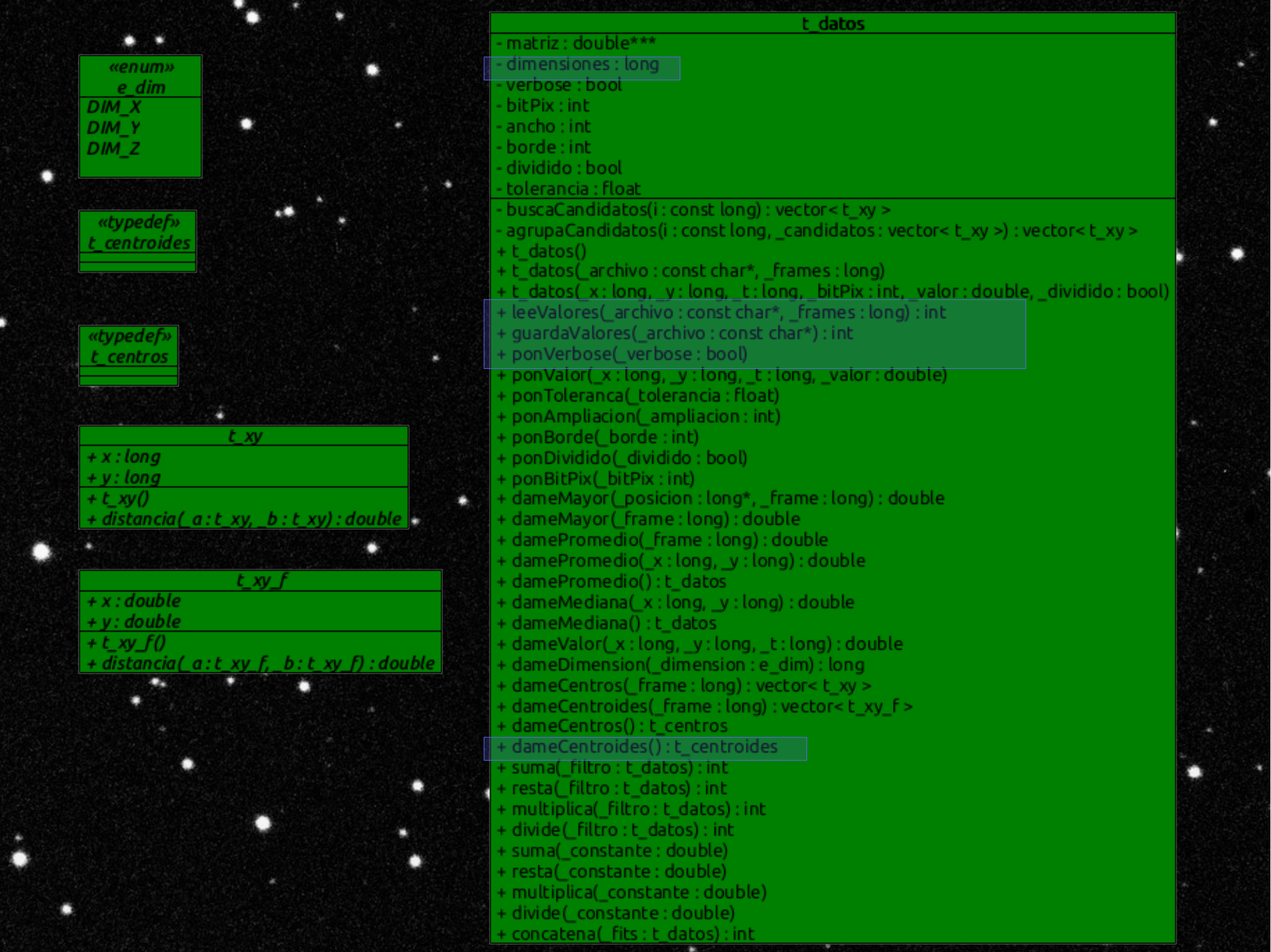
C++ con la biblioteca cfitsio compilado en Linux (Ubuntu 16.04).

Programación modular con clases.

Documentación Doxygen.

```
164
165  /* ******************** Class ******************** */
166  /*! @class t_datos
167   * Guarda todos los valores de un fits en una matriz de tres dimensiones.
168   * SI DESEAS MODIFICAR EL CÓDIGO: las imágenes fits empiezan en la esquina
169   * inferior izquierda con la posición x=1, y=1. Las matrices en C++ comienzan en
170   * la posición [0][0]. Esto debe tomarse en cuenta si para manipular el código
171   * de esta clase. Pero las peticiones de datos se devuelven en posiciones de
172   * fits. Ejemplo: si el valor máximo está en el pixel (100, 200) del frame 10
173   * del fits, el valor se almacenará en la posición [99][199][9] de la matriz,
174   * pero eso es de manera privada, para obtener este valor se trabajará como si
175   * del fits se tratara, debes llamar método getValor(100, 200, 10) y para
176   * cambiarlo debes llamar al método setValor(valor, 100, 200, 10).
177   */
178  class t_datos
179  {
180  private:
181      double ***matriz;        //! Matriz con todos los datos
182      long dimensiones[3];     //! Dimensiones del "cubo" (frame, fila, pixel)
183      bool verbose;            //! Modo verbose, diagnóstico de errores
184      int bitPix,              //! Tipo de dato que guarda el archivo fits
185          ancho,               //! Ampliación al encontrar una posible estrella para calcular su centroide
186          borde;               //! El borde segurido de los frames
187      bool dividido;           //! Pantalla dividida a la mitad izquierda y derecha
188      float tolerancia;        //! Tolerancia para el brillo de una estrella.
189
190      vector<t_xy> buscaCandidatos(const long i);
191      vector<t_xy> agrupaCandidatos(const long i, vector<t_xy> _candidatos);
192  protected:
193  public:
194      // Constructores
195      t_datos();
196      t_datos(const char *_archivo, long _frames);
197      t_datos(long _x, long _y, long _t, int _bitPix, double _valor, bool _dividido);
198
```


datos.hpp - Code::Blocks 16.01
File Edit View Search Project Build Debug wxSmith Tools Tools+ Plugins DoxyBlocks Settings Valgrind Help
datos.hpp
procesos.hpp
test.cpp
VER cout << " Añadiendo la segunda matriz..." << endl;
for (long i = 0; i < _fits.dameDimension(DIM_Z); i++)
for (long j = 0; j < dameDimension(DIM_X); j++)
for (long k = 0; k < dameDimension(DIM_Y); k++)
pNuevaMatriz[i + dameDimension(DIM_Z)][j][k] = _fits.dameValor(j + 1, k + 1, i + 1);
VER cout << " 0k." << endl;
VER cout << " Redirigiendo memoria..." << endl;
free(matriz);
matriz = pNuevaMatriz;
pNuevaMatriz = nullptr;
VER cout << " 0k." << endl;
dimensiones[0] = lFrames;
}
else
{
cout << ROJO << "ERROR: " << VERDE << "No se pueden concatenar fits de dimensiones distintas: (" <<
dameDimension(DIM_X) << ", " << dameDimension(DIM_Y) << ") << (" <<
_fits.dameDimension(DIM_X) << ", " << _fits.dameDimension(DIM_Y) << ") " << NORMAL << endl;
iEstado = 1;
}
VER cout << VERDE << " ** Termina concatenado **" << NORMAL << endl;
return iEstado;
}
/* * * * * * * * * * * * * * * Overloaded Operators * * * * * * * * * * * * * * */
// !!!
void t_datos::ponBitPix(int _bitPix)
{
bitPix = _bitPix;
}
#endif // __HELIX_DATOS__
/mnt/Files/Education/Universidad Nacional Autónoma de Méx
Unix (LF)
UTF-8
Line 1191, Column 104
Insert
Read/Wr


«enum»
e_dim
DIM_X
DIM_Y
DIM_Z
«typedef»
t_centroides
«typedef»
t_centros
t_xy
+ x : long
+ y : long
+ t_xy()
+ distancia(_a : t_xy, _b : t_xy) : double
t_xy_f
+ x : double
+ y : double
+ t_xy_f()
+ distancia(_a : t_xy_f, _b : t_xy_f) : double
t_datos
- matriz : double***
- dimensiones : long
- verbose : bool
- bitPix : int
- ancho : int
- borde : int
- dividido : bool
- tolerancia : float
- buscaCandidatos(i : const long) : vector< t_xy >
- agrupaCandidatos(i : const long, _candidatos : vector< t_xy >) : vector< t_xy >
+ t_datos()
+ t_datos(_archivo : const char*, _frames : long)
+ t_datos(_x : long, _y : long, _t : long, _bitPix : int, _valor : double, _dividido : bool)
+ leeValores(_archivo : const char*, _frames : long) : int
+ guardaValores(_archivo : const char*) : int
+ ponVerbose(_verbose : bool)
+ ponValor(_x : long, _y : long, _t : long, _valor : double)
+ ponToleranca(_tolerancia : float)
+ ponAmpliacion(_ampliacion : int)
+ ponBorde(_borde : int)
+ ponDividido(_dividido : bool)
+ ponBitPix(_bitPix : int)
+ dameMayor(_posicion : long*, _frame : long) : double
+ dameMayor(_frame : long) : double
+ damePromedio(_frame : long) : double
+ damePromedio(_x : long, _y : long) : double
+ damePromedio() : t_datos
+ dameMediana(_x : long, _y : long) : double
+ dameMediana() : t_datos
+ dameValor(_x : long, _y : long, _t : long) : double
+ dameDimension(_dimension : e_dim) : long
+ dameCentros(_frame : long) : vector< t_xy >
+ dameCentroides(_frame : long) : vector< t_xy_f >
+ dameCentros() : t_centros
+ dameCentroides() : t_centroides
+ suma(_filtro : t_datos) : int
+ resta(_filtro : t_datos) : int
+ multiplica(_filtro : t_datos) : int
+ divide(_filtro : t_datos) : int
+ suma(_constante : double)
+ resta(_constante : double)
+ multiplica(_constante : double)
+ divide(_constante : double)
+ concatena(_fits : t_datos) : int

Uso futuro: óptica adaptativa

Video: European Southern Observatory

Conclusiones

- Se creó un sistema automatizado para la reducción de los archivos fits.
- Se identifican los centroides de las estrellas para poder realizar un *luky imaging* en base al PSF.
- Posteriormente se puede utilizar las funcionalidades para óptica adaptativa.

Preguntas

Gracias