天文图像常用算法的Python实现

* 图像的重新投影
* 三维数据的光谱获取、速度场计算
* 图像的平滑、插值、重建
* 天体认证，团块结构提取
* 傅里叶变换
* 光谱拟合

* 数据下载: https://pan.baidu.com/s/14MP_wNwGc08g1RD3y4XYTA　密码: fauv

* 主题：使用Python实现(天文)图像处理和分析
* 安装Anaconda Python 工作环境：
  https://www.anaconda.com/what-is-anaconda/
  (如果不成功，尝试以前版本：https://repo.anaconda.com/archive/)
* 设置python3.5/3.6环境（3.7 太新，很多软件有冲突。）

```
> conda create --name py35 python=3.5
> conda activate py35
 py35> conda install ipython
py35> conda install notebook ipykernel
py35> conda install jupyter
# ( quite py35 environment : conda deactivate )
```

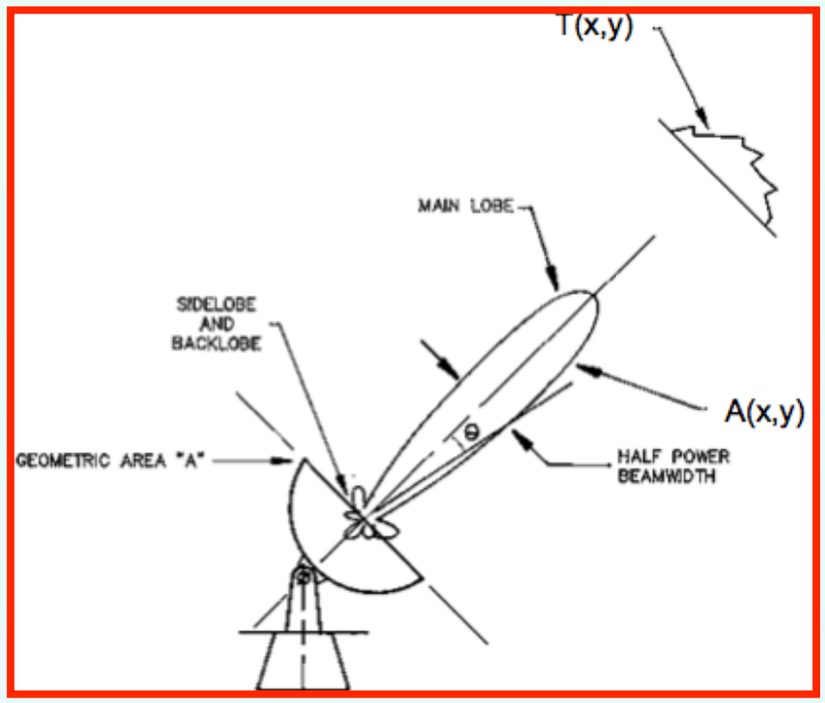
* basic packages and modules:

```
py35> conda install matplotlib astropy scikit-image
py35> pip install scipy
py35> pip install numpy==1.17.2  # will cover the conda version, compatible with mlfit
py35> pip install fil_finder
py35> pip install seaborn
py35> pip install pycupid
py35> pip install spectral-cube
py35> pip install lmfit
py35> pip install reproject
py35> # 3d plot tools:
py35> pip install PyQt5
py35> pip install mayavi
py35> pip install --upgrade scikit-image
```

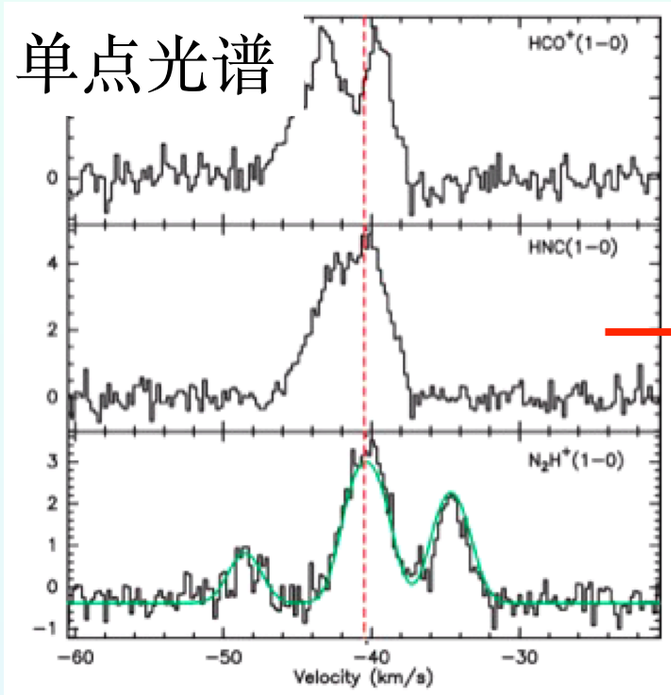
* 链接: https://pan.baidu.com/s/14MP_wNwGc08g1RD3y4XYTA 密码: fauv

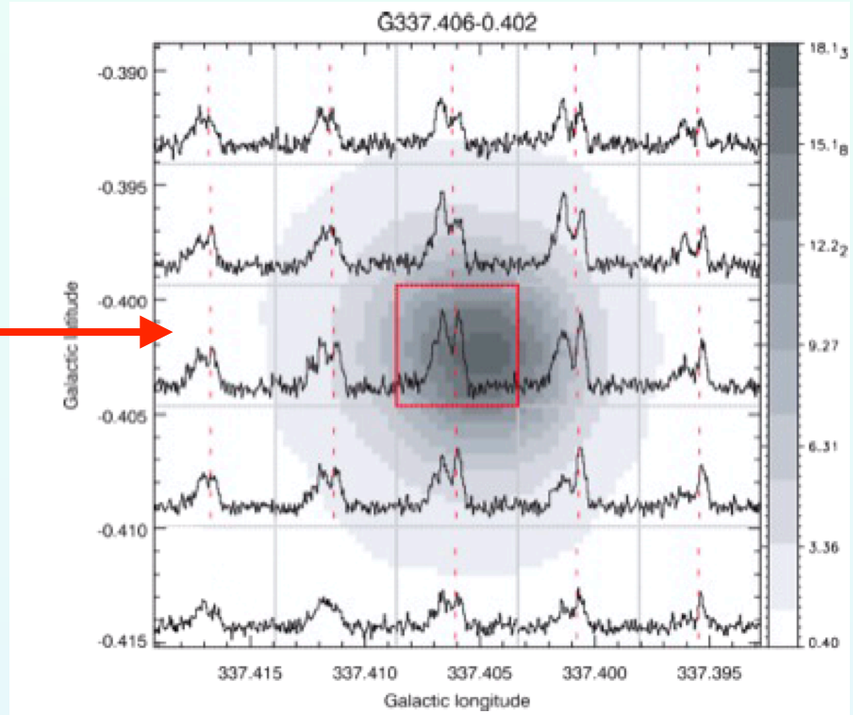
多波段天文：三维数据（data cube）

采集数据的方法: 天空平面点阵采样+插值成图



单点光谱



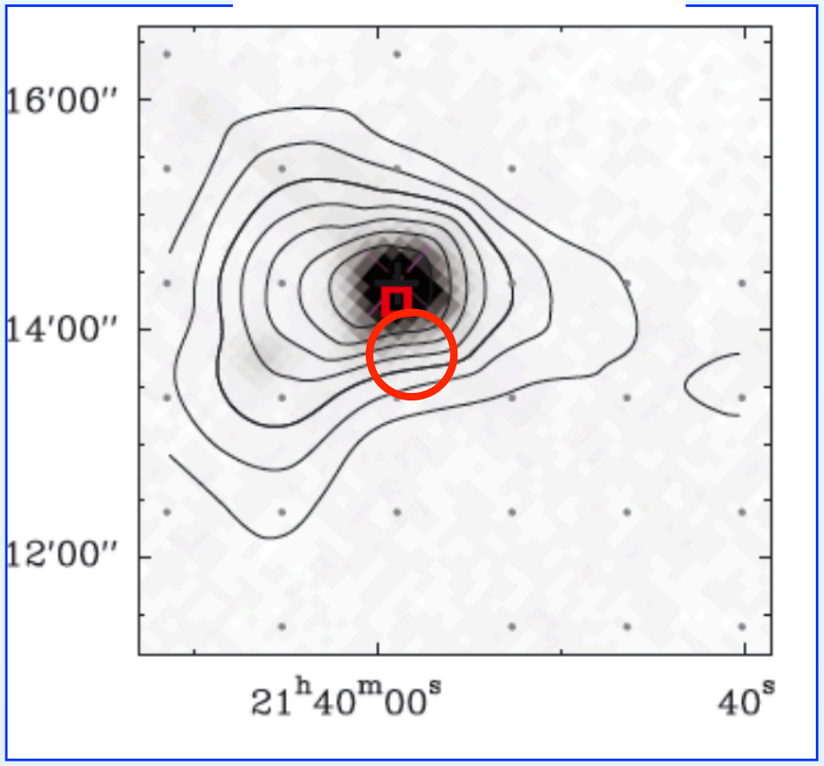


$$\mathrm{BWFN} = 2.439\,\frac{\lambda}{D}\,\mathrm{rad} \simeq 139.8^{\circ}\,\frac{\lambda}{D}$$

λ:观测波长； D:天线直径

涉及到的其他仪器参量:
主波束宽度、波束效率、
系统温度

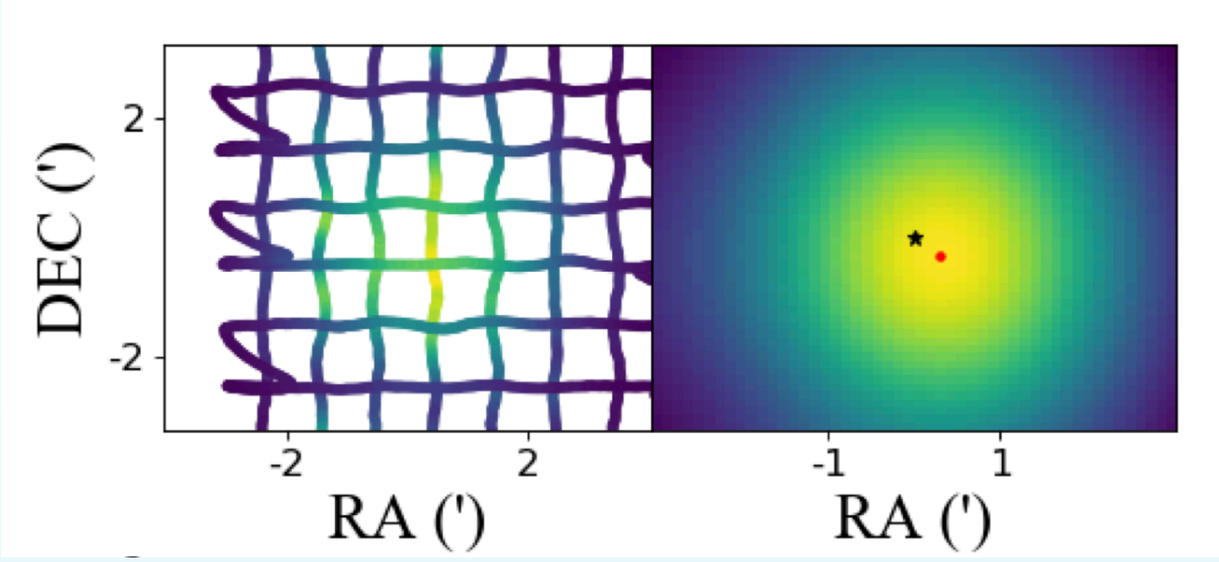
点阵采样-成图



基本观测模式:
采样点阵在天空平面上均匀分布
间距 = 主波束(分辨率)*0.5
(Nyquist Sampling)

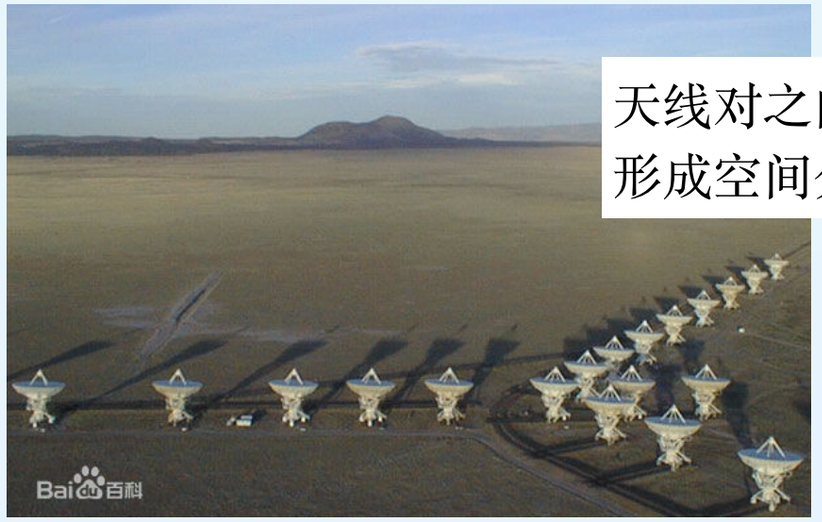
采集数据的方法

2. 连续扫描观测(On the Fly)

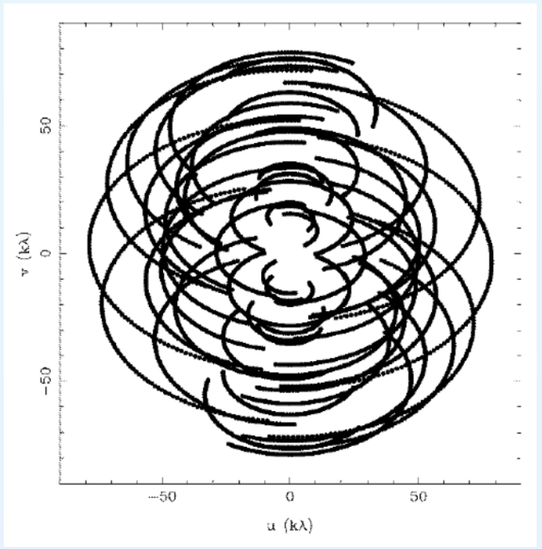


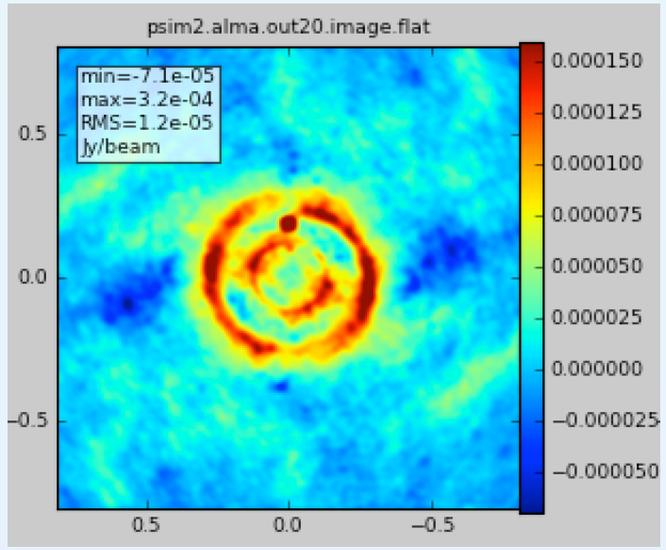
3C273, FAST 0-1.5 GHz emission

2.干涉仪直接成图观测：不改变指向，在不同频率上直接成图



天线对之间的干涉效应
形成空间分辨率





傅里叶变换：一维

$$F(u)=\int_{-\infty}^{\infty} f(x)e^{-j2\pi ux}dx$$

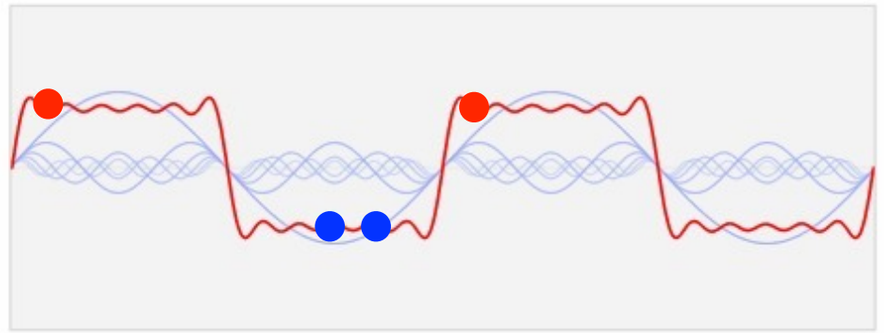
$$f(x)=\int_{-\infty}^{\infty} F(u)e^{j2\pi ux}du$$

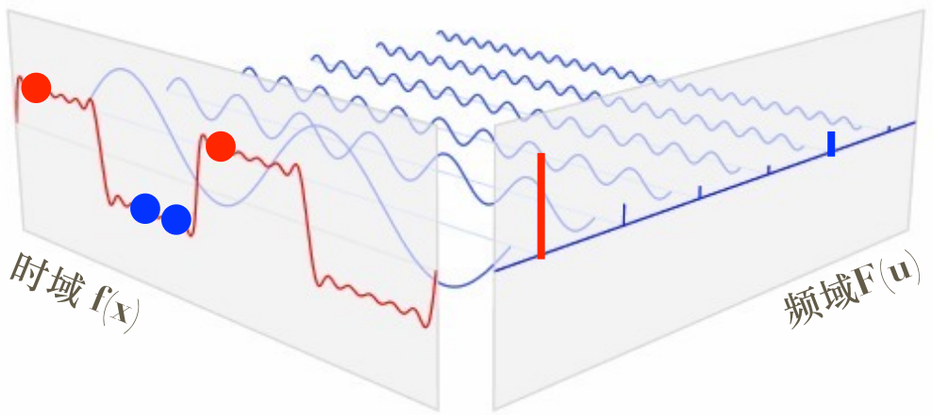
F(u): 频率为u的单频成分的振幅

时域上随时间变化的起伏

|

频域上各种单频成分的组合





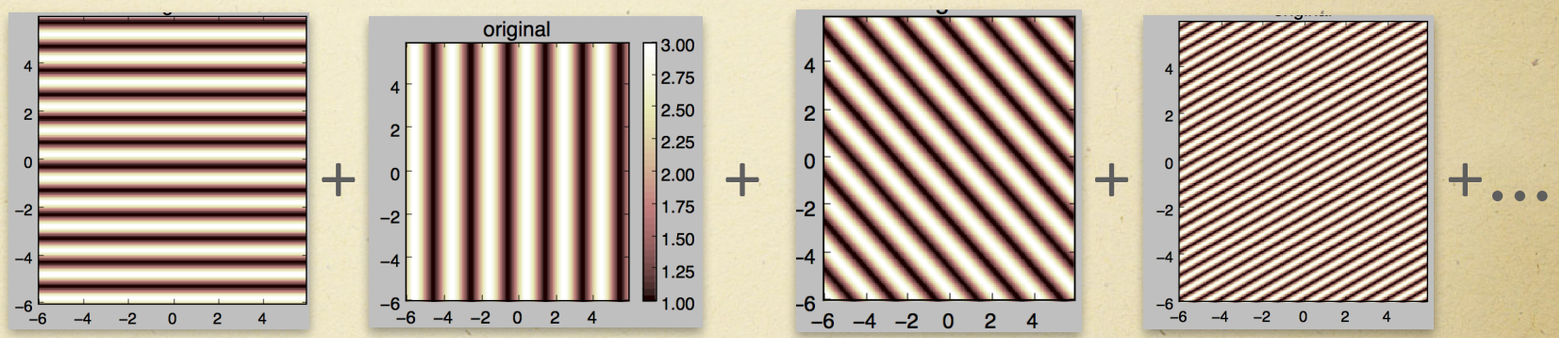
傅里叶变换：二维

$$F(u,v)=\int_{-\infty}^{\infty} f(x,y)e^{-j2\pi(ux+vy)}dxdy$$

$$f(x,y)=\int_{-\infty}^{\infty} F(u,v)e^{j2\pi(ux+vy)}dudv$$

f(x,y)

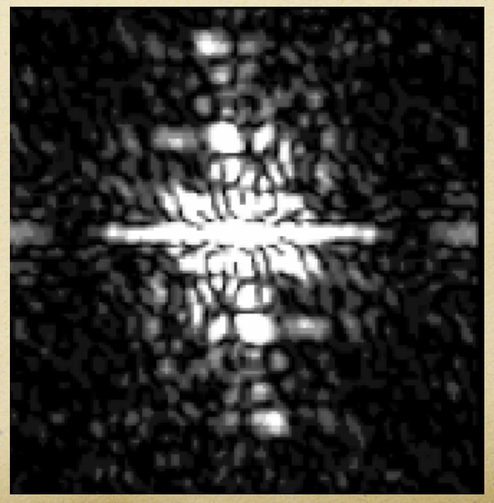
FFT



=

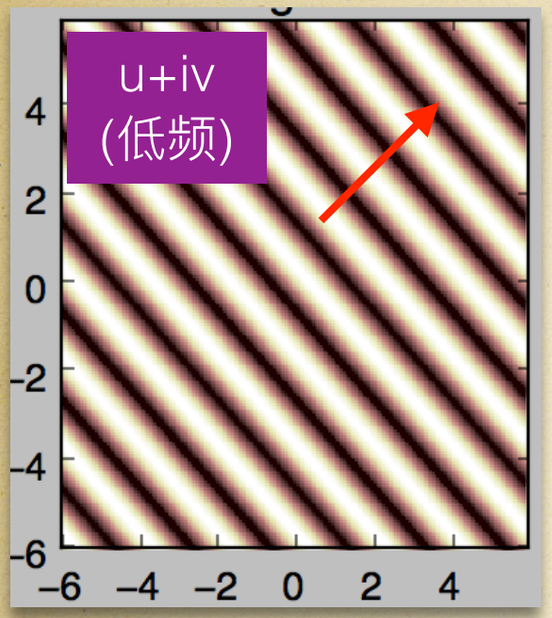
图像上随空间变化的起伏

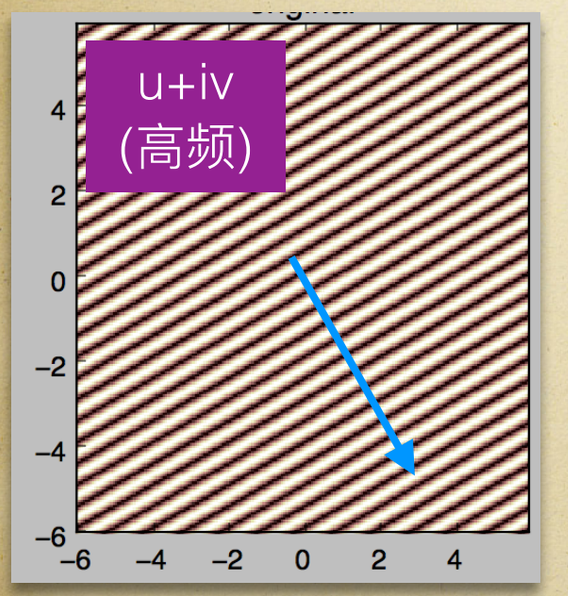
空间频域上各种单频成分的组合



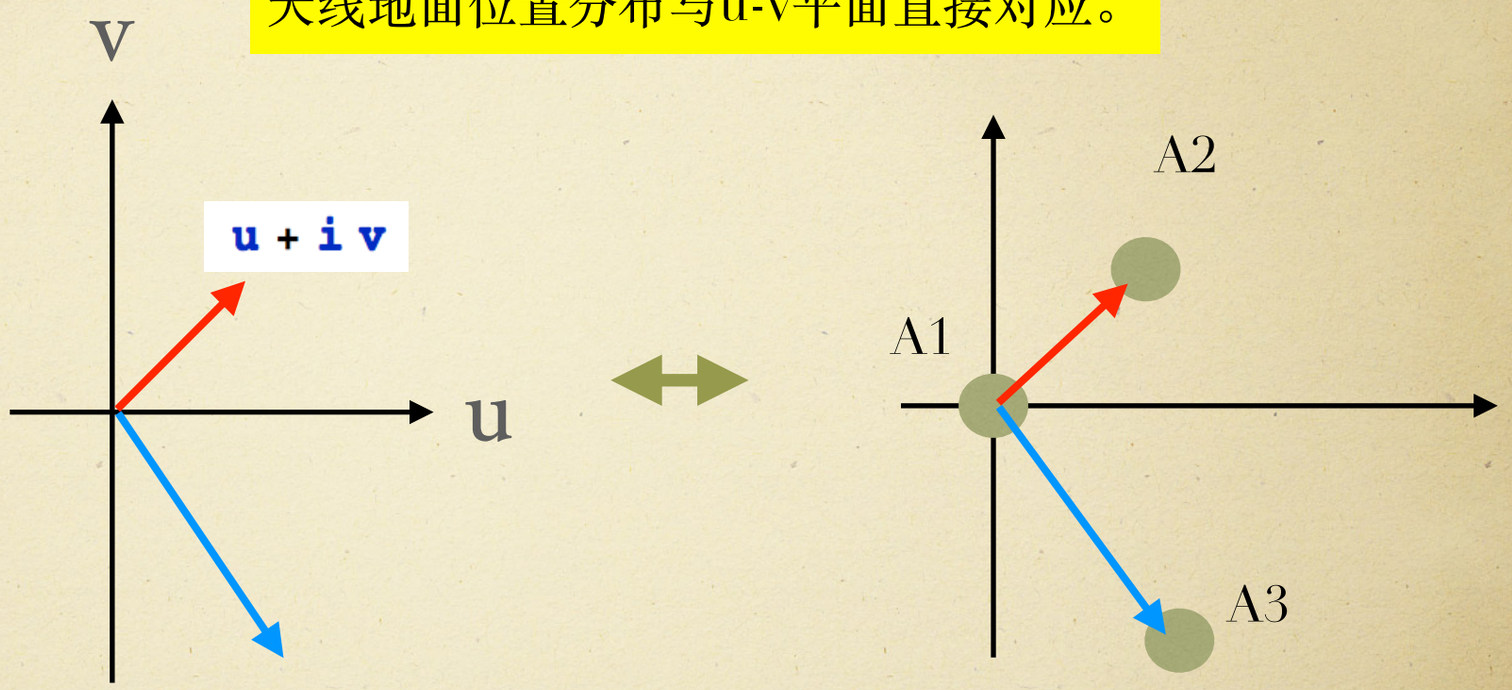
F(u,v)

干涉仪工作原理：复可见度函数F(u,v)





天线地面位置分布与u-v平面直接对应。



- 特征尺度/分辨率：

$$\theta = \frac{1}{\sqrt{u^2 + v^2}} \text{ (arcsec)}$$

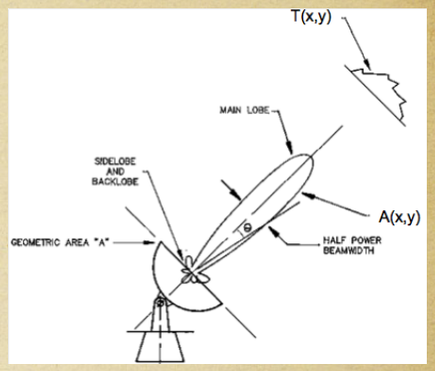
$$\sqrt{u^2 + v^2} = \frac{D_{antenna}}{206\,265\,\lambda}$$

$$\frac{206\,265\,\lambda}{D} = 20 \text{ arcsec} \left(\frac{\lambda}{1 \text{ cm}}\right)\left(\frac{D}{100 \text{ meter}}\right)^{-1}$$

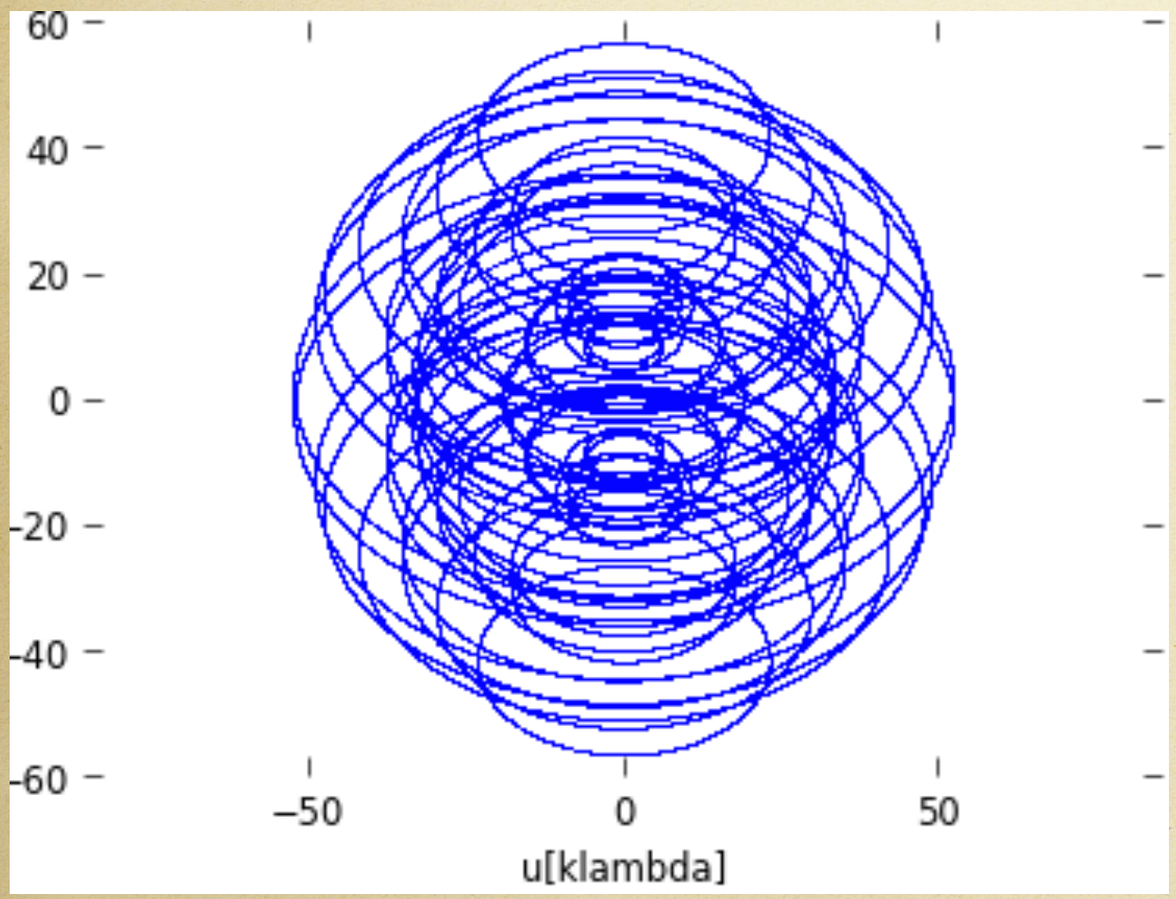
- A1–A2相距100米，在波长1cm下所能测到的特征尺度为20 arcsec
- A1–A3相距200米，在波长1cm下所能测到的特征尺度为10 arcsec
- 相比之下：太阳直径张角=0.5度=1800 arcsec

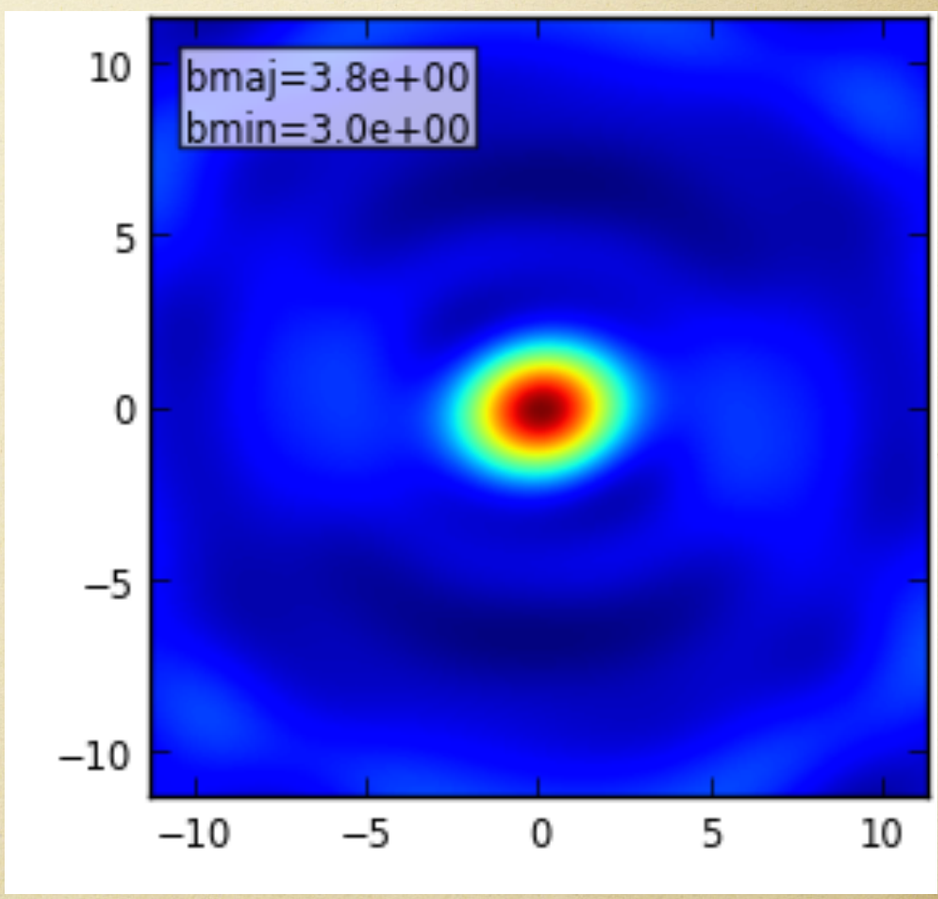
干涉仪工作原理：UV覆盖与脏束（dirty beam）

单天线的beam



UV 覆盖 $\xrightarrow{FFT^{-1}}$ 脏束



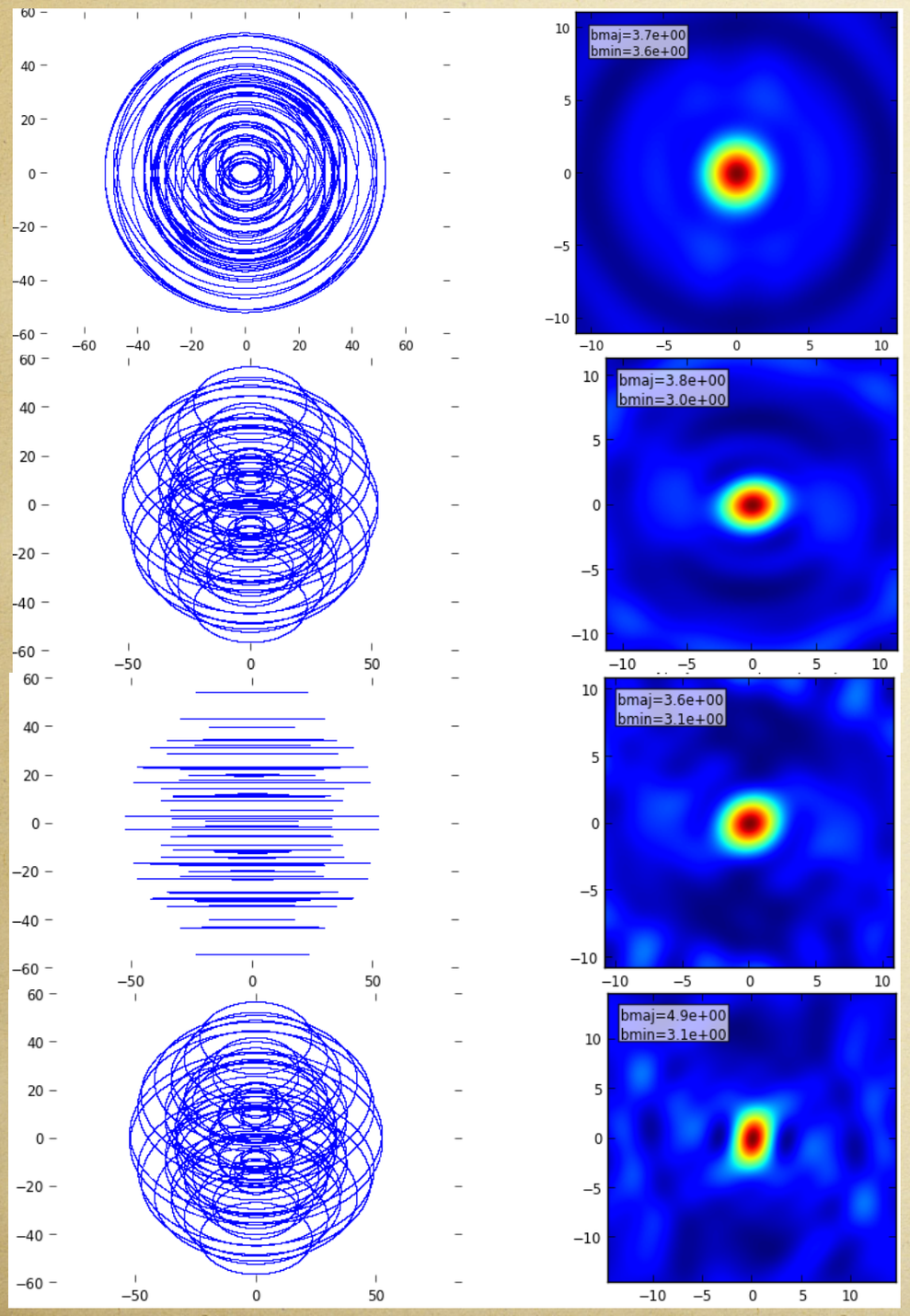


干涉仪工作原理：UV覆盖与脏束（dirty beam）

total time=24hr

SMA compact

total time=2hr

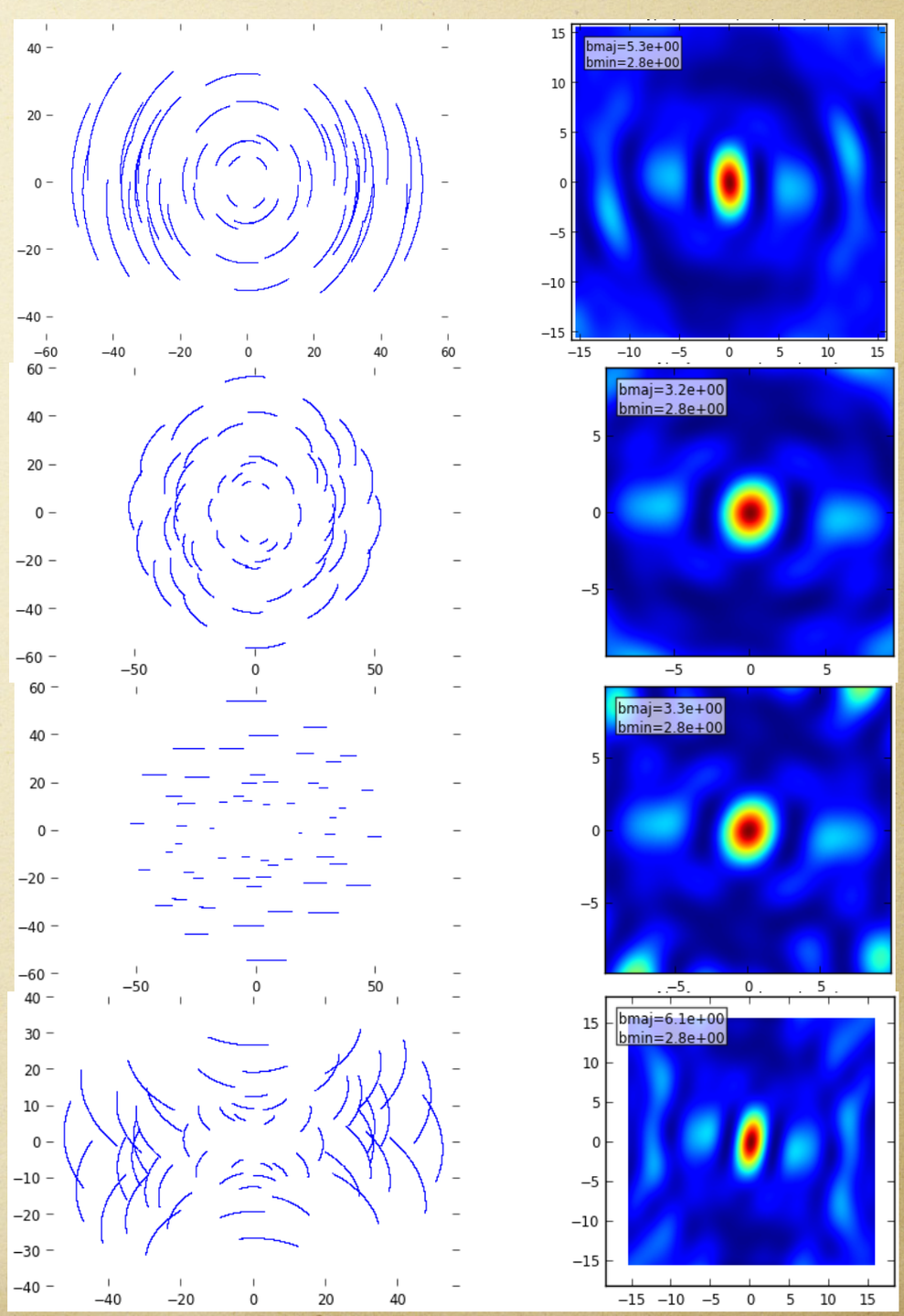


Dec=80deg

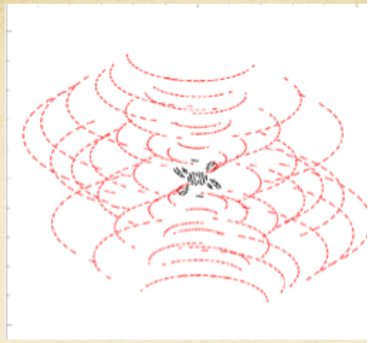
Dec=40deg

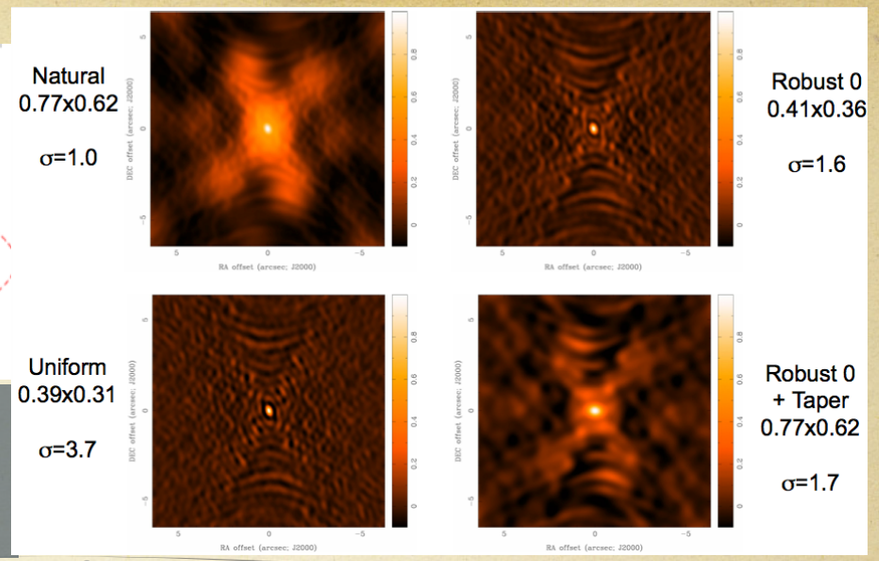
Dec=0deg

Dec=-40deg



Clean (to image data)



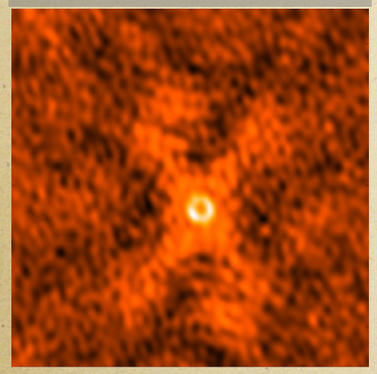


UV-data

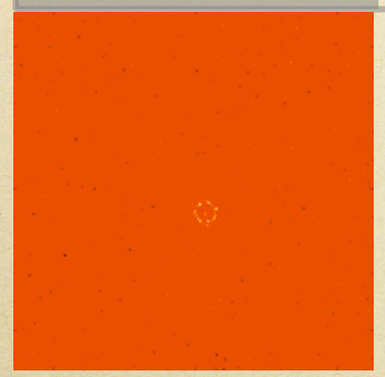
UV-coverage

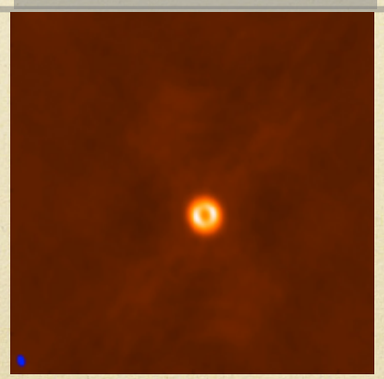
loops

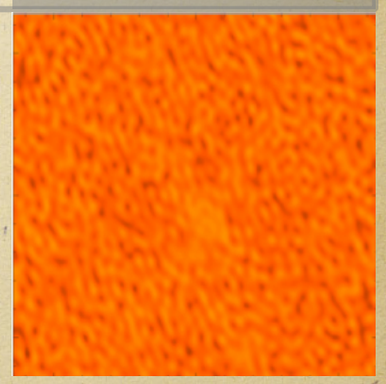
Dirty Image − **Delta (peak)** ⊗ **Dirty Beam** × **Gain (<1)** = **Residual**



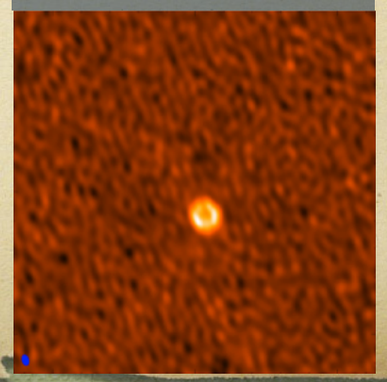
Clean Model ⊗ **Clean Beam** + **Residual Image**



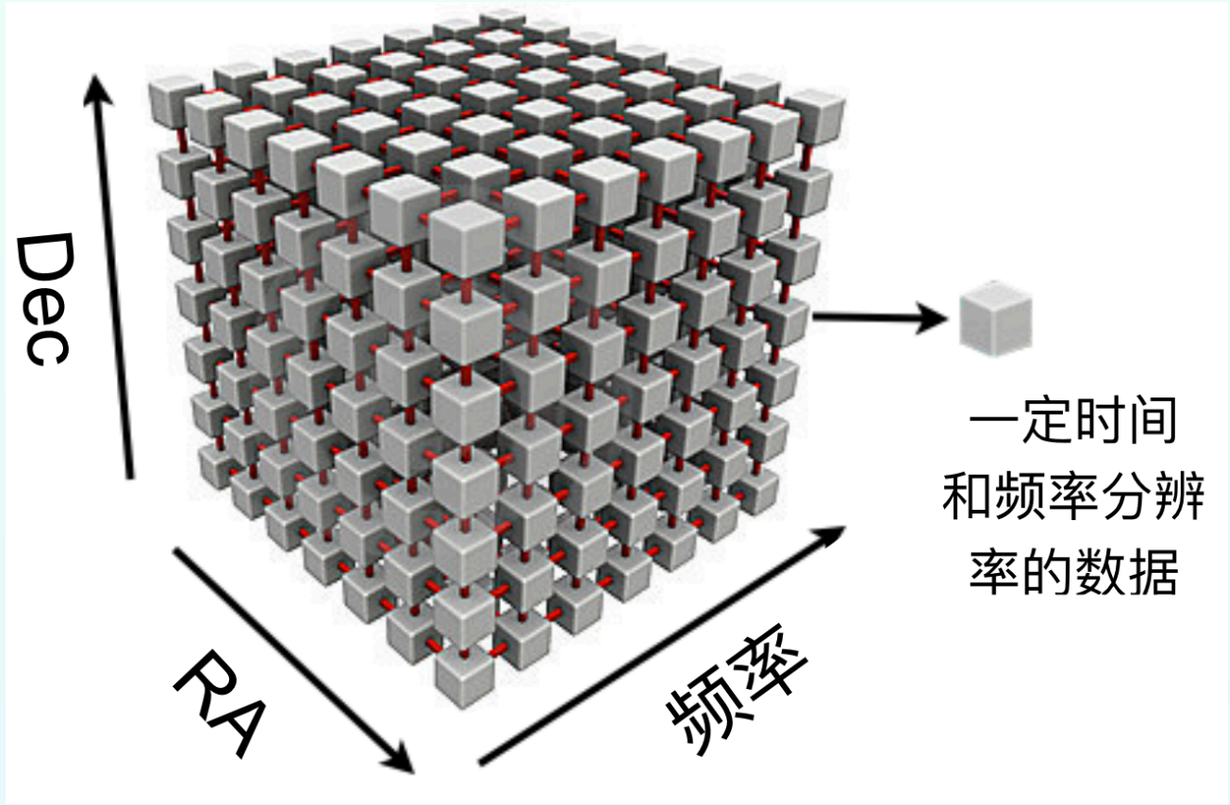


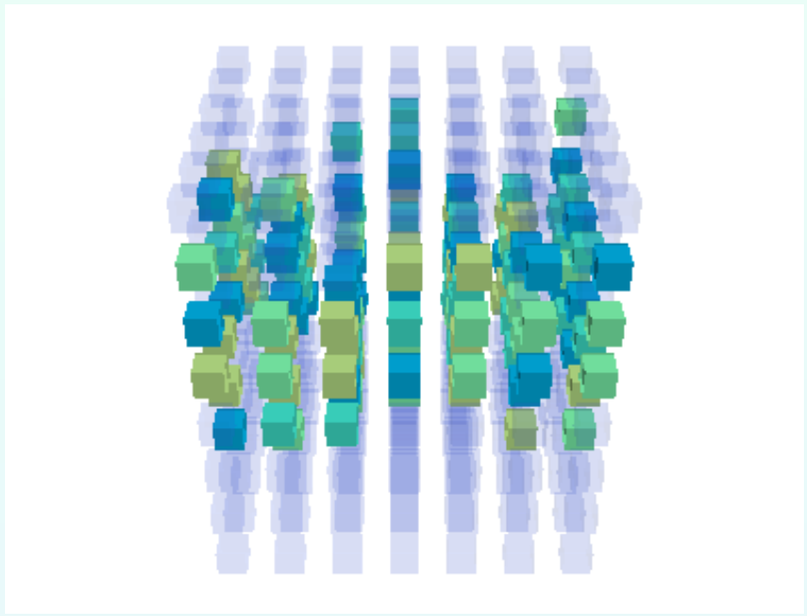


Restored Image

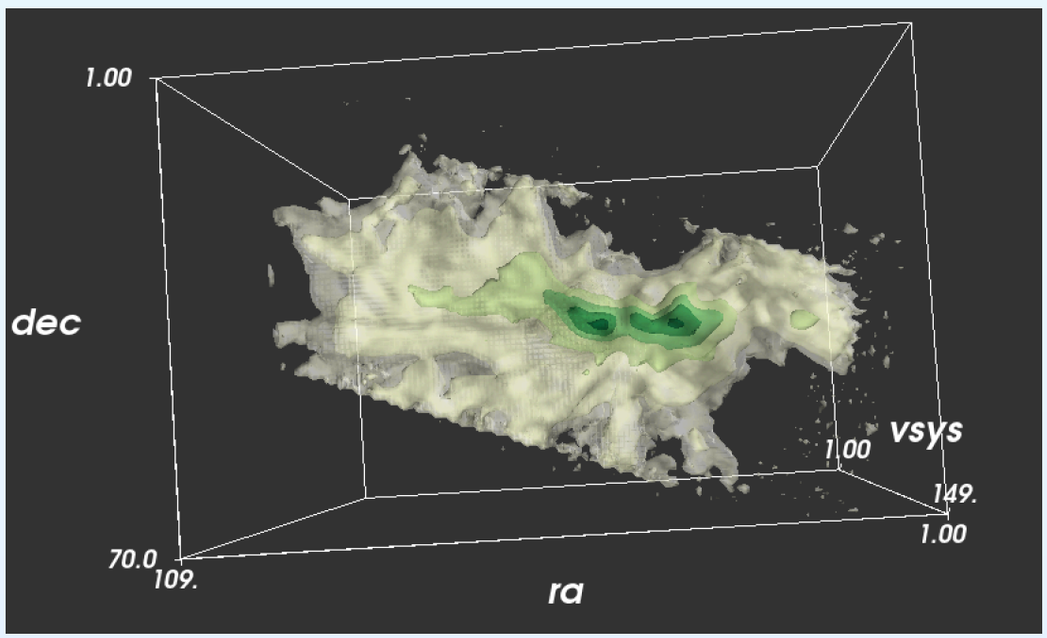


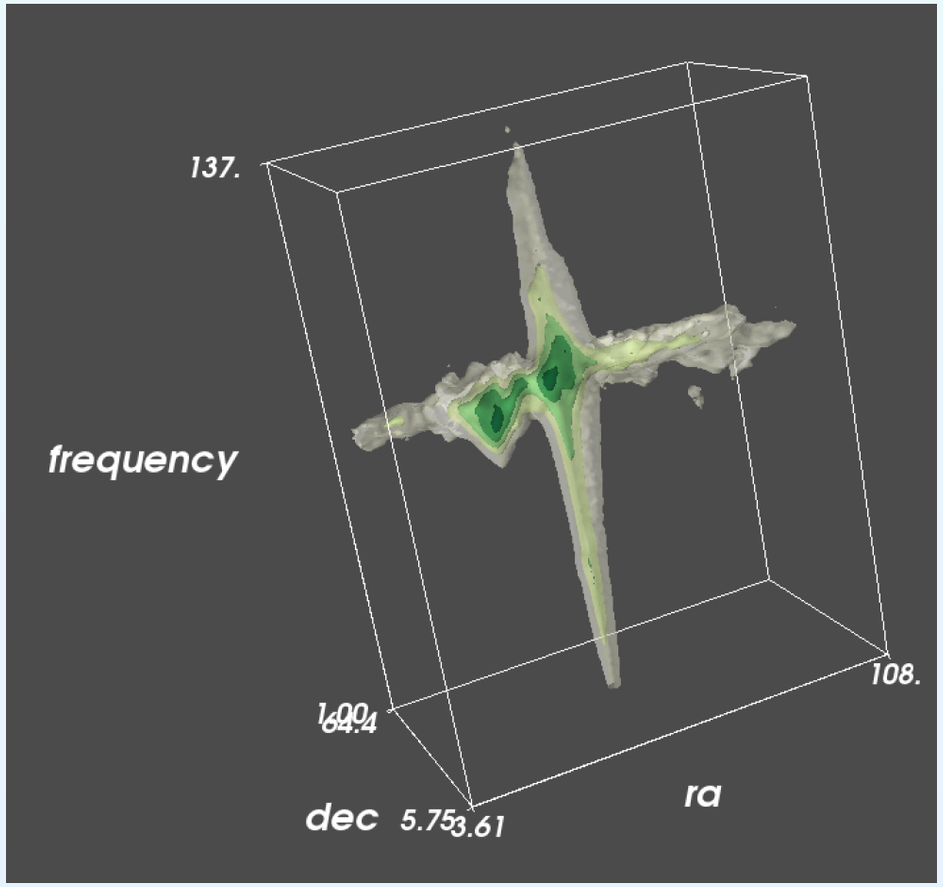
多波段天文：三维数据（data cube）



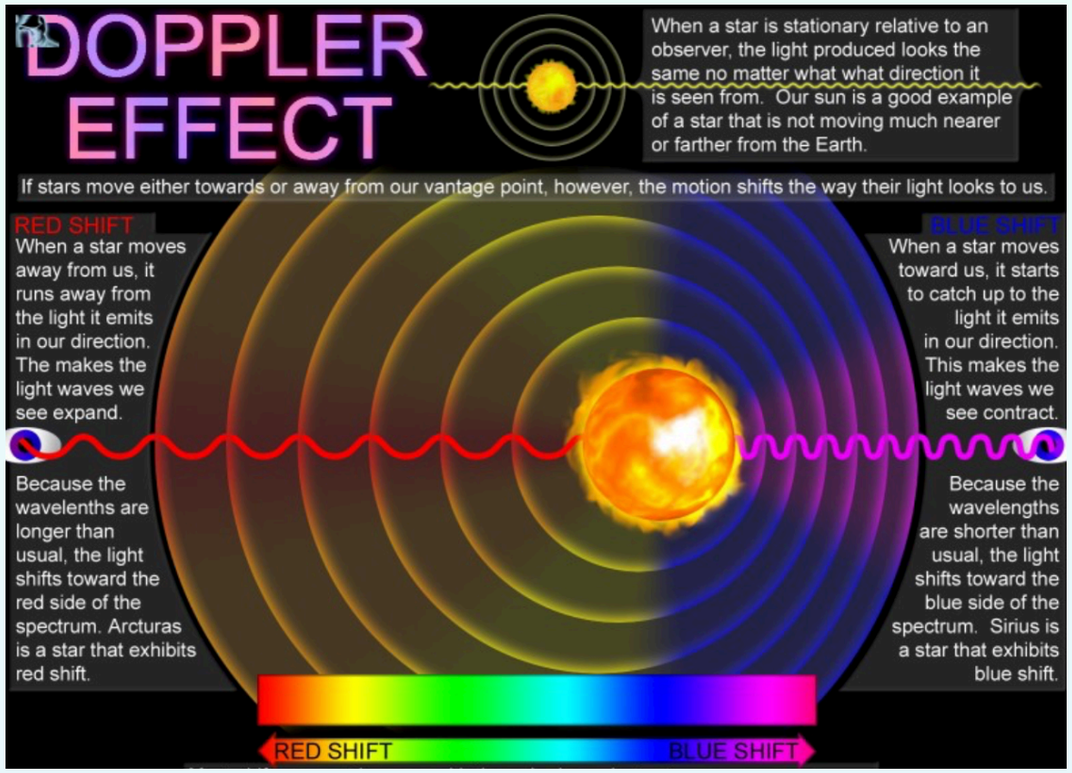


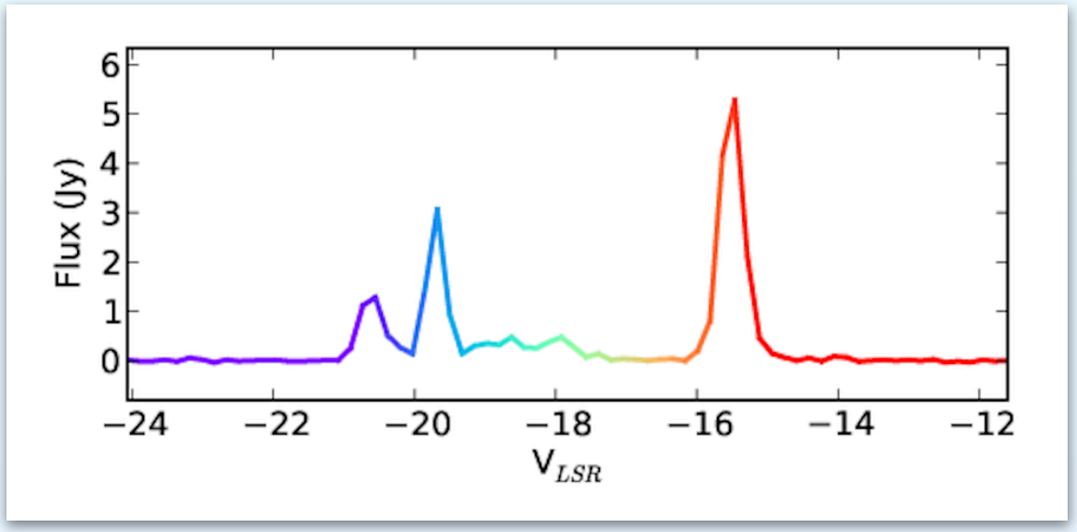
不同方向的透视图





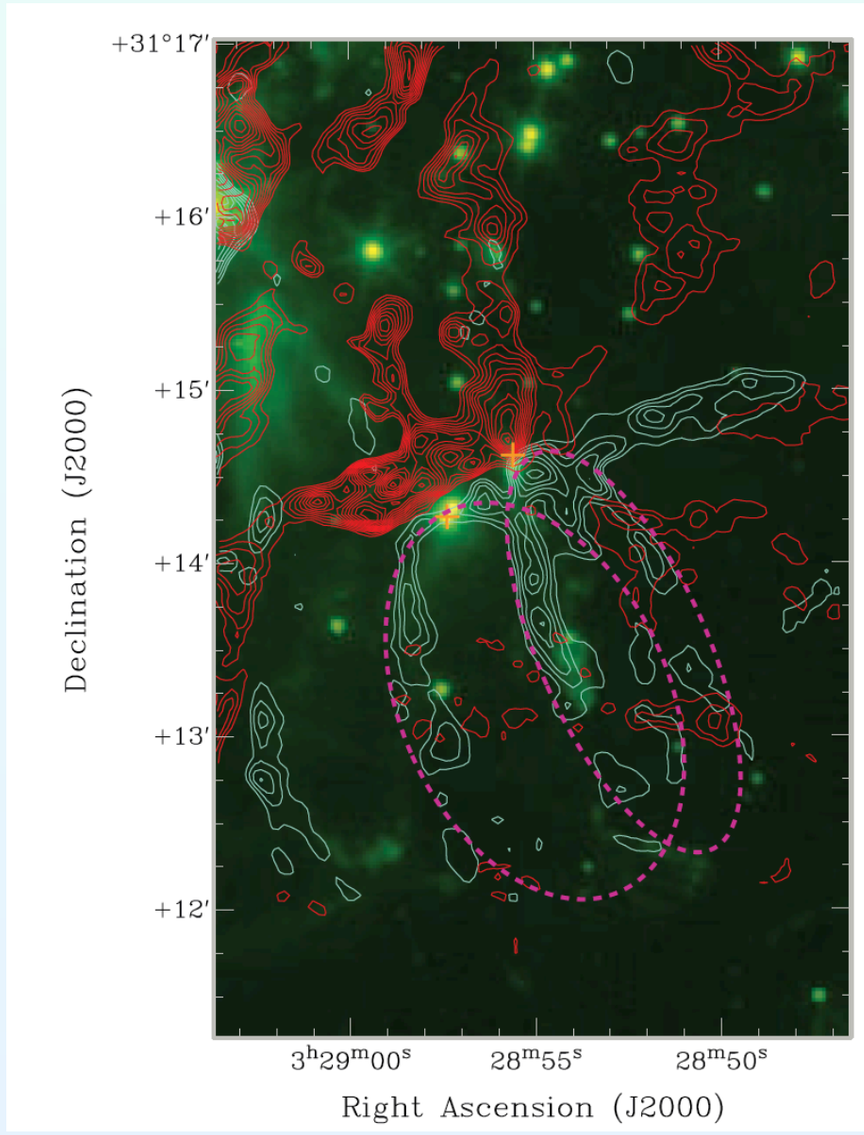
多普勒展宽： $\Delta V/c = \Delta f/f_0$





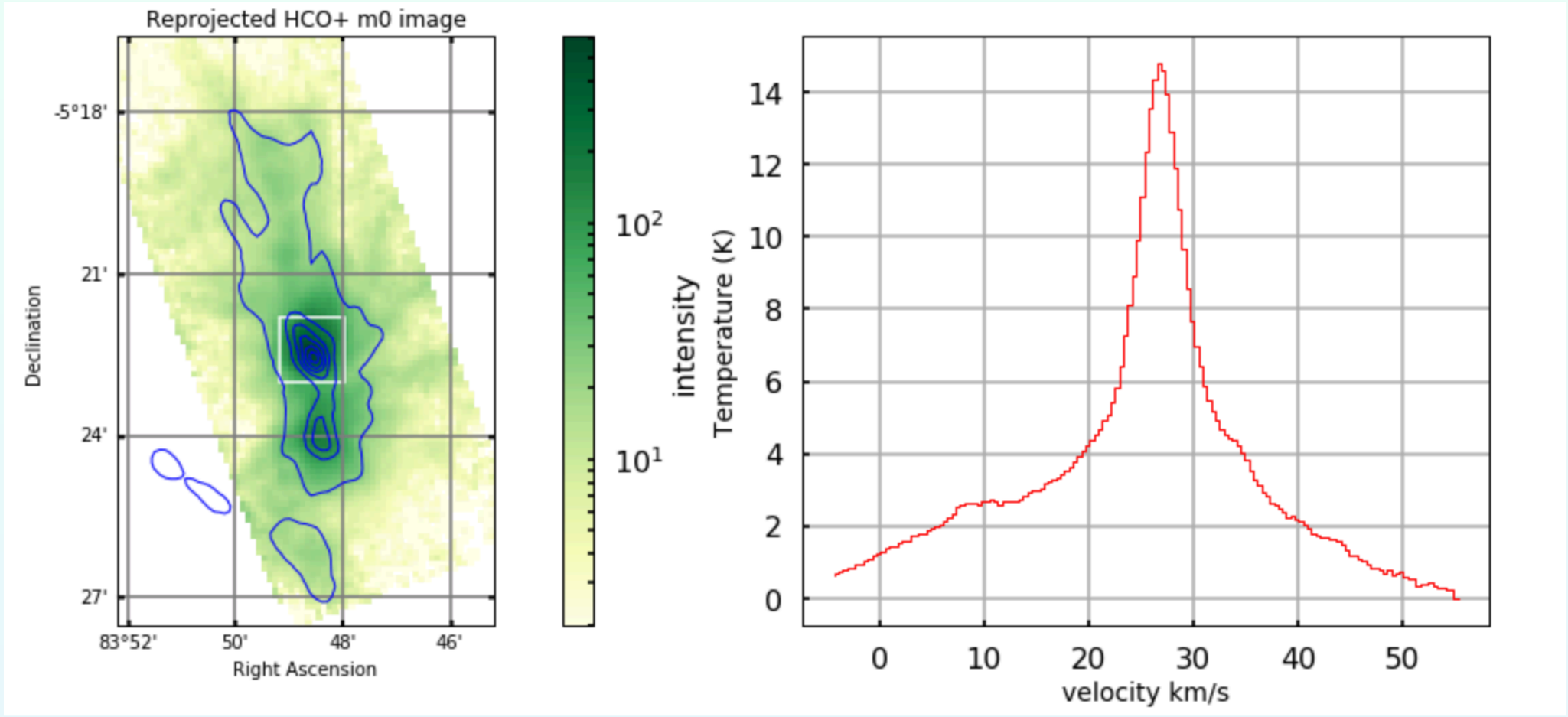
V<5 km/s 气体团块内的热运动-湍动

V>5 km/s 气体外流或内流

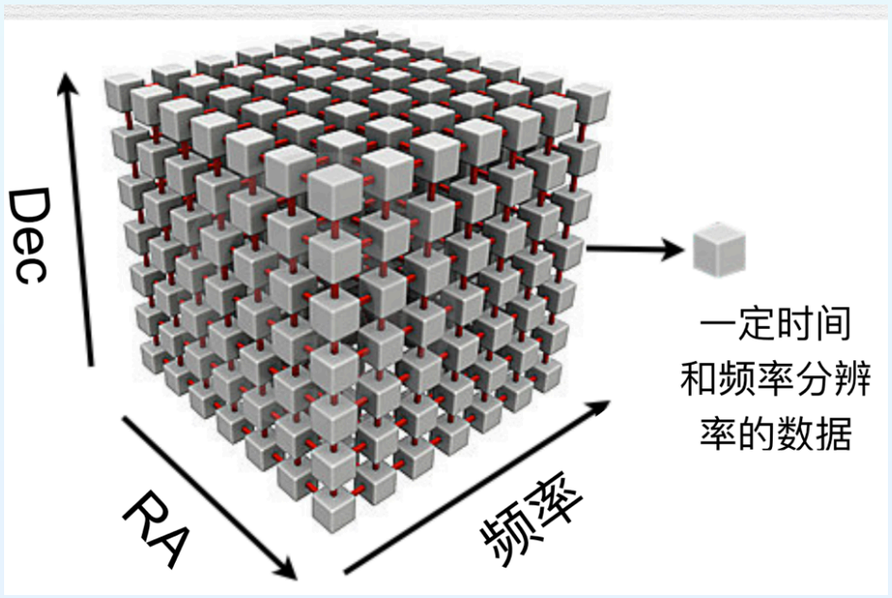


Plunkett 2013, NGC 1333 outflow

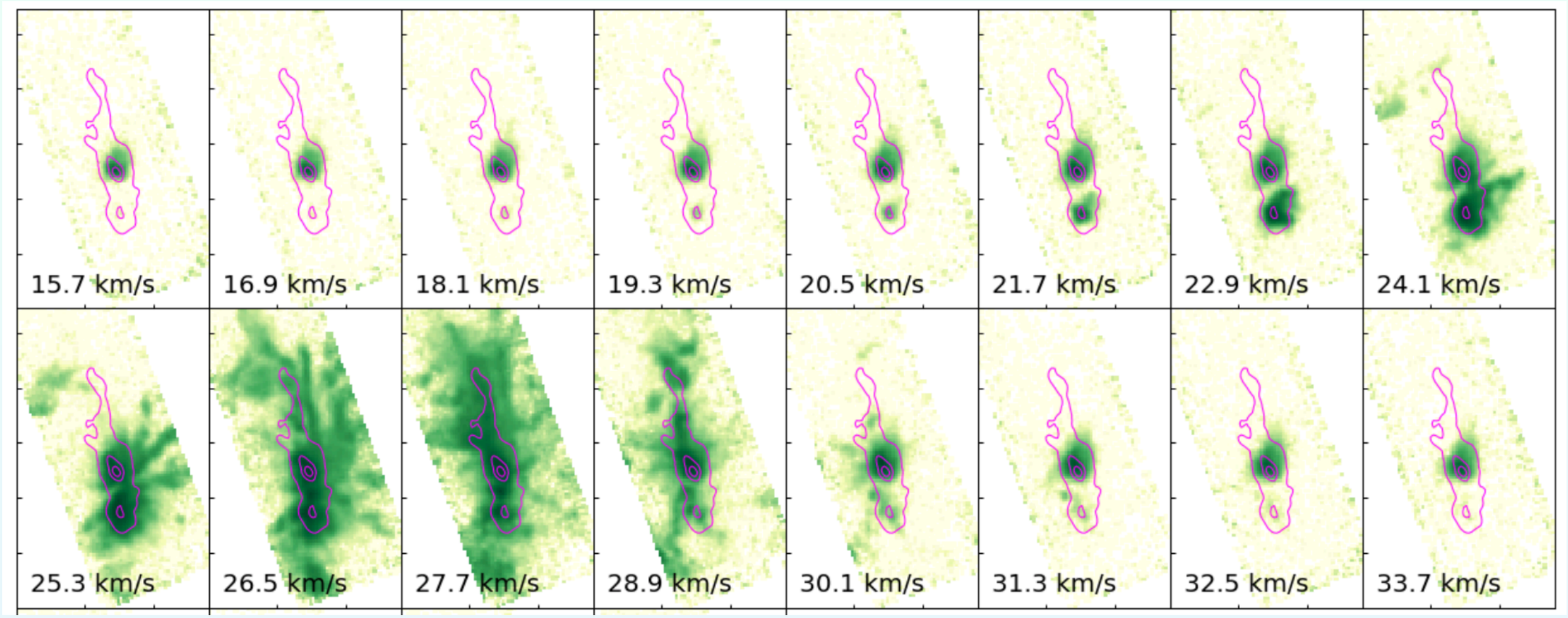
三维数据——光谱的提取：




```
# -------------spectrum display---------------
ax1 = fg1.add_subplot(1,2,2)
sp_v = scu1.spectral_axis.value
Tb_v = scu1_data[:, y0:(y0+yw), x0:(x0+xw)].mean(axis=(1,2))
ax1.plot(sp_v, Tb_v,c='r',drawstyle='steps',lw=1.0)
plt.grid(True)
ax1.set_xlabel('velocity km/s',fontsize=14)
ax1.set_ylabel('Temperature (K)',fontsize=14)
```

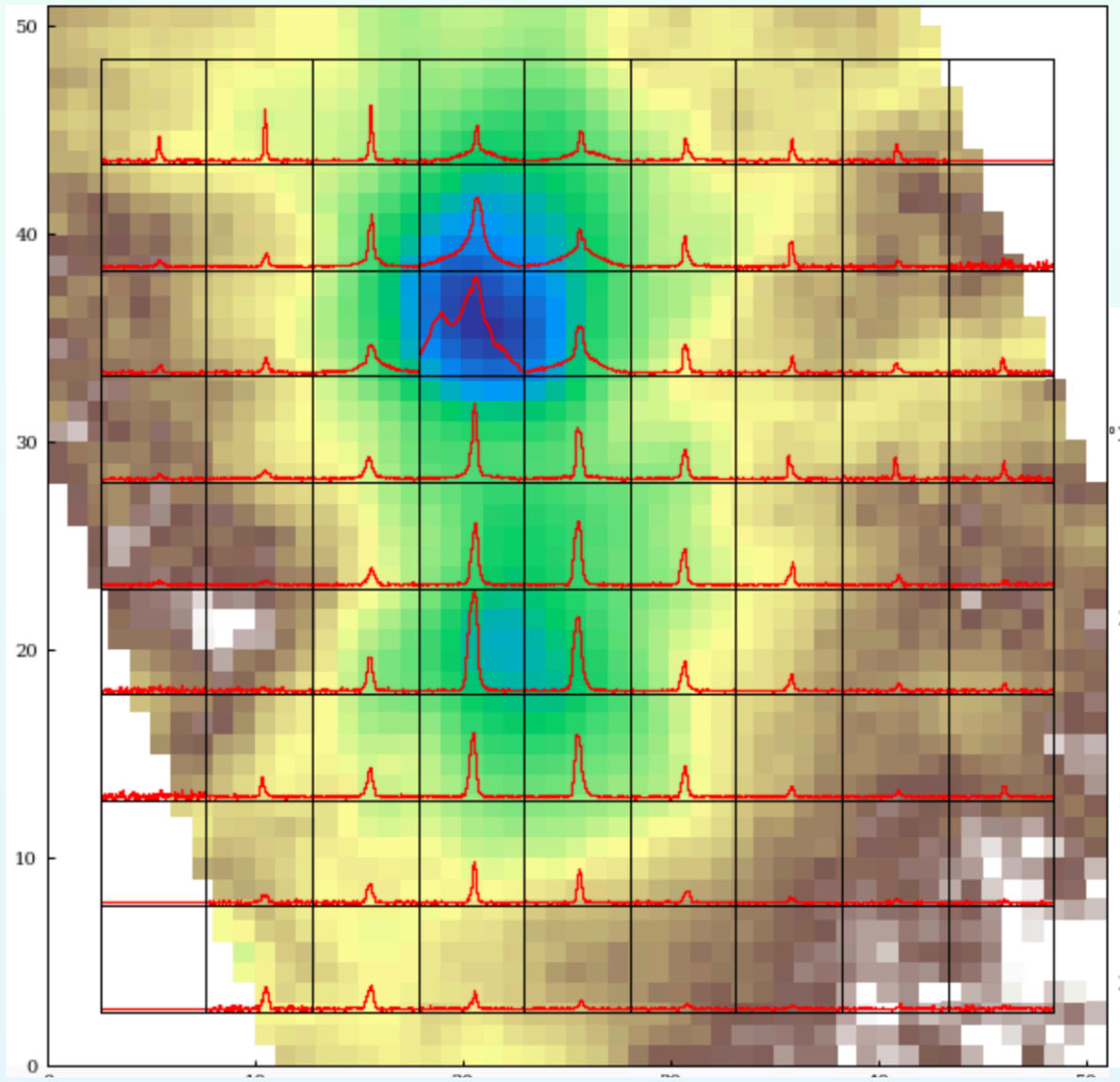


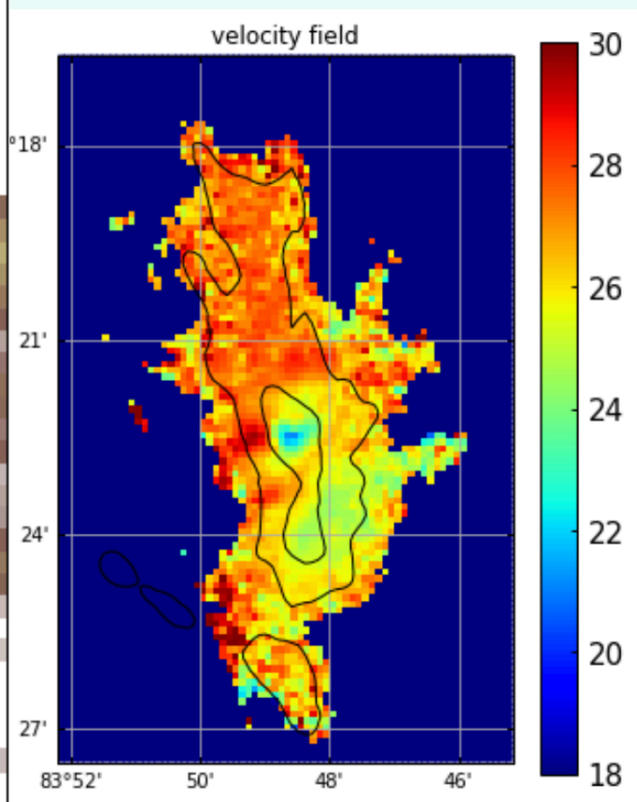
绘制速度通道图

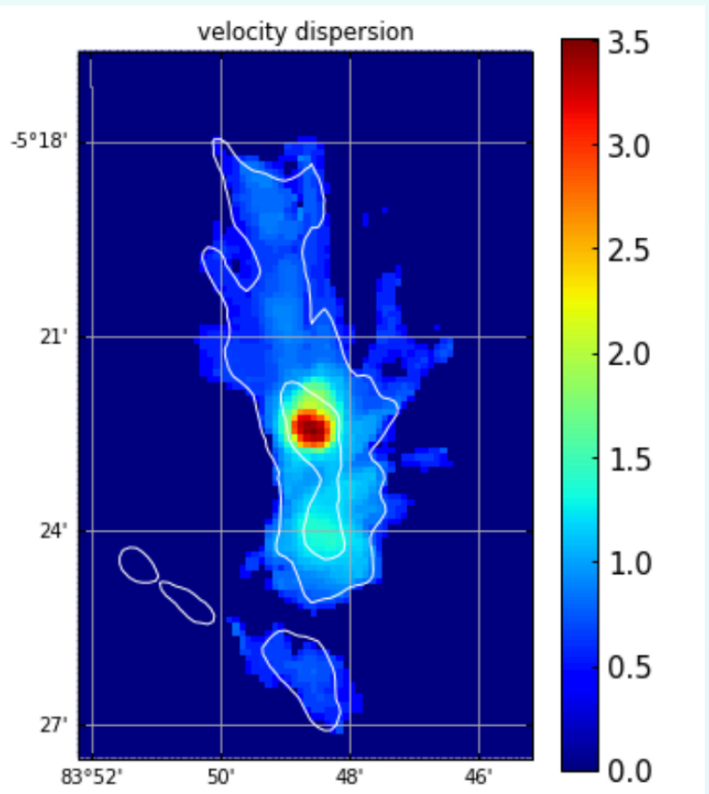



```
for i in np.arange(0,dn):
    ax1 = fig.add_subplot(ny,nx,i+1)
    Tb_chan = Tb_m[i*nstep+n1,:,:]
    ax1.pcolor(Tb_chan, cmap=plt.cm.YlGn, \
                norm=colors.LogNorm(vmin=0.1, vmax=max0))
    ax1.contour(hdu_sp250.data, levels=lvs, colors='magenta',linewidths=1.0)
    ax1.axes.set_aspect('equal')
```

绘制速度场
谱线速度和线宽的空间分布



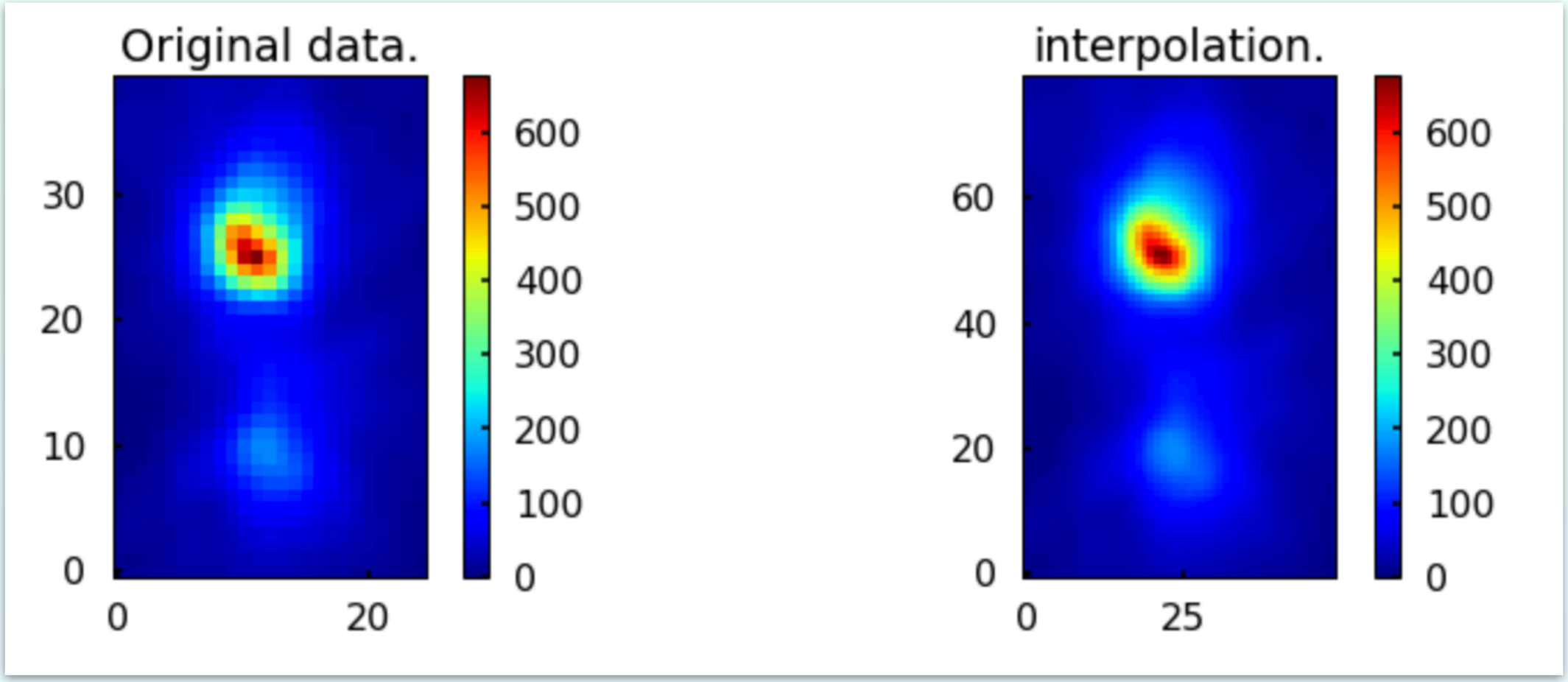





```
3  scu2b_rg2 = sc.read('./scu2b_rg2.fits',hdu=0)
```

```
7  scu2_mi = scu2b_rg2.moment(order=0)
8  # moment0: integrated intensity
9  # moment1: average velocity
10 # moment2: velocity dispersion (also : scu2b_rg2b.linewidth_sigma())
```

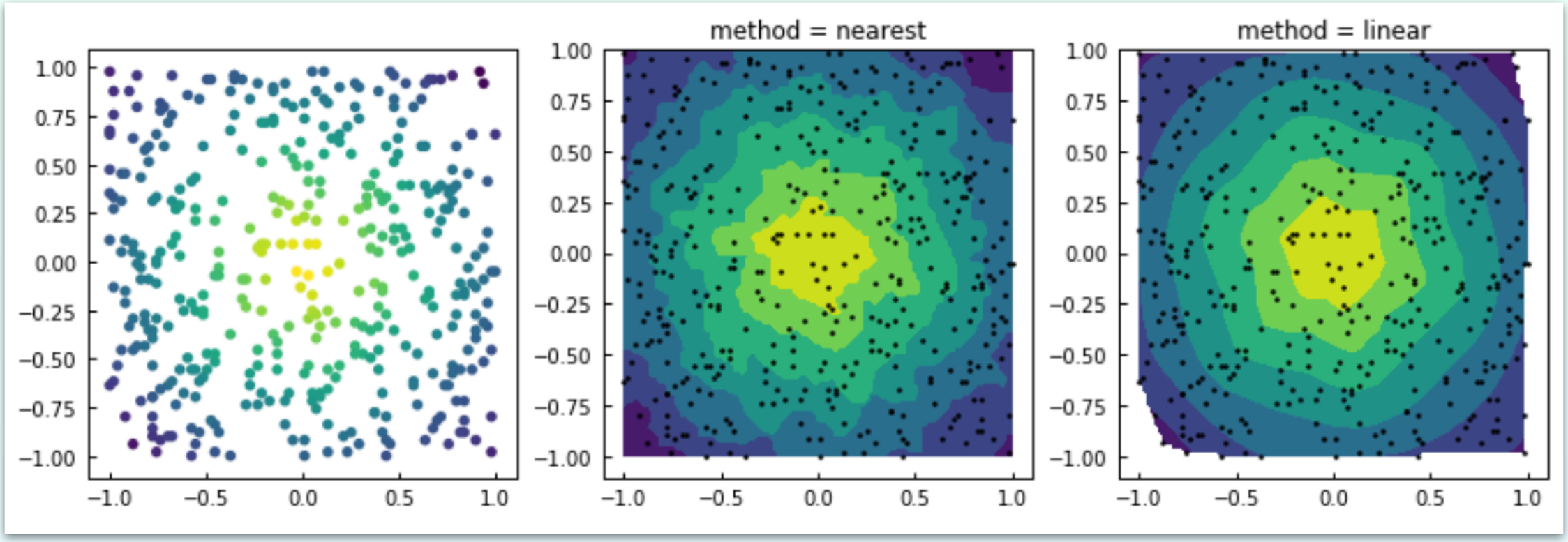
图像的平滑和插值




```python
# ------------------------ interpolation ----------------------------
from scipy import interpolate
scu2_mi = scu2b_rg2.moment(order=0)  # moment0: integrated intensity
scu2_mi = scu2_mi.hdu.data
map0_area = scu2_mi[25:65,25:50]

nx, ny = np.shape(map0_area)
# xm, ym = np.mgrid[-2:2:nx*1j, -2:2:ny*1j]
xm, ym = np.linspace(-2,2,nx), np.linspace(-2,2,ny)
nt = 2
xm2, ym2 = np.linspace(-2,2,nx*nt), np.linspace(-2,2,ny*nt)
tck = interpolate.RectBivariateSpline(xm,ym,map0_area, s=0.3, kx=3, ky=3)
map0_area2 = tck(xm2, ym2)
#map0_area2 = np.transpose(map0_area2)
```

不均匀数据点的插值：

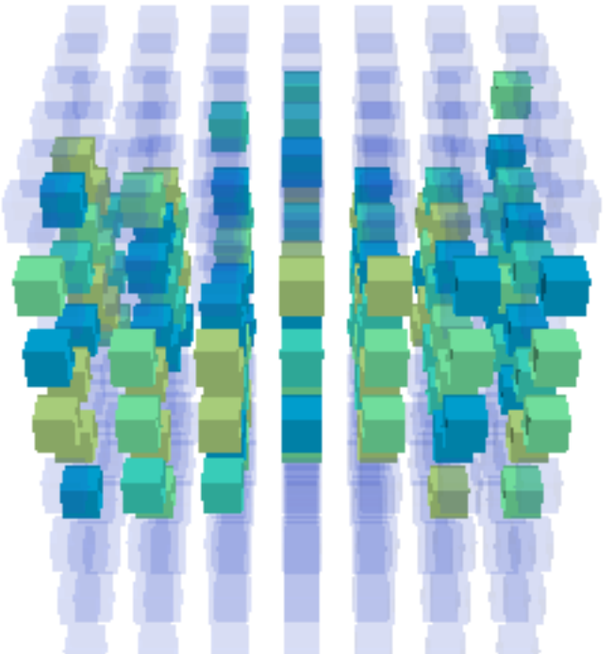



```
x = np.linspace(-1,1,100)
y =  np.linspace(-1,1,100)
X, Y = np.meshgrid(x,y)
# for i, method in enumerate(('nearest', 'linear', 'cubic')):
Ti = griddata((px, py), data(px,py), (X, Y), method=method)
```

cubic：假设光滑曲面

三维数据的处理：整体可视化展现

```
 5 import numpy as np
 6 #from enthought.mayavi import mlab
 7 from mayavi import mlab
 8 import numpy as np
 9 import mayavi.mlab as mlab
10 import moviepy.editor as mpy
11
12 x, y, z = np.mgrid[:6,:7,:8]
13 c = np.zeros((6, 7, 8), dtype=np.int)
14 c.fill(1)
15 k = np.random.randint(2, 5, size=(6, 7))
16
17 idx_i, idx_j, _ = np.ogrid[:6, :7, :8]
18 idx_k = k[:,:, np.newaxis] + np.arange(3)
19 c[idx_i, idx_j, idx_k] = np.random.randint(2,6, size=(6,7,3))
20
21 # clmp="viridis"
22 # clmp="YlGn"
23 clmp="rainbow"
```



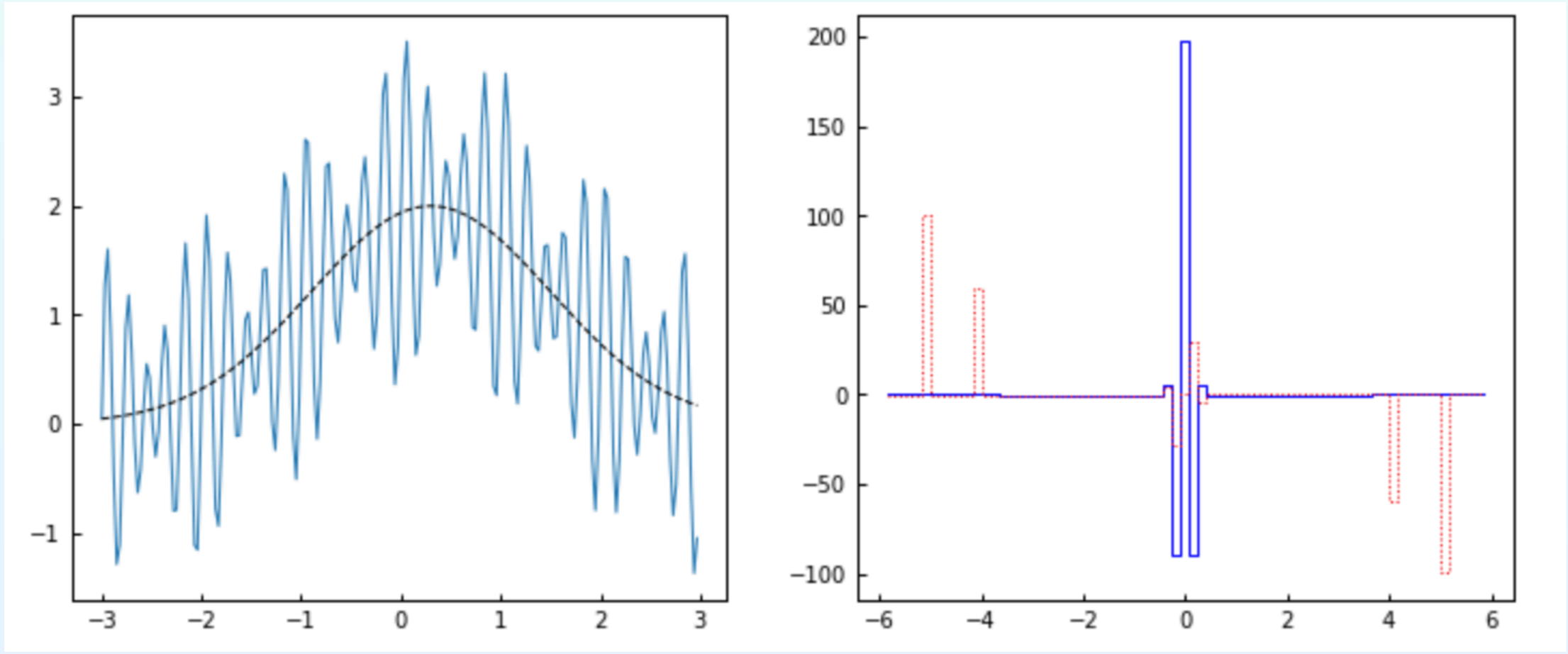
```
28 mlab.points3d(x[c>1], y[c>1], z[c>1], c[c>1], mode="cube", scale_factor=0.6,-
29     scale_mode="none", transparent=False, vmin=0, vmax=8, colormap=clmp, opacity = 1.0)
30
31 mlab.points3d(x[c==1], y[c==1], z[c==1], c[c==1], mode="cube", scale_factor=0.6,
32     scale_mode="none", transparent=True, vmin=0, vmax=8, colormap=clmp, opacity = 0.2)
33 mlab.gcf().scene.background = (1,1,1)
34 mlab.show()
```

傅里叶变换和滤波

$$\mathscr{F}[f(t)] = F(\omega) = \int_{-\infty}^{+\infty} f(t)e^{-i\omega t}\,dt$$

$$\mathscr{F}^{-1}[F(\omega)] = f(t) = \frac{1}{2\pi}\int_{-\infty}^{+\infty} F(\omega)e^{i\omega t}\,d\omega$$

```
7  y_ft = np.fft.fft(signal_arr)
8  y_fts = np.fft.fftshift(y_ft)
```



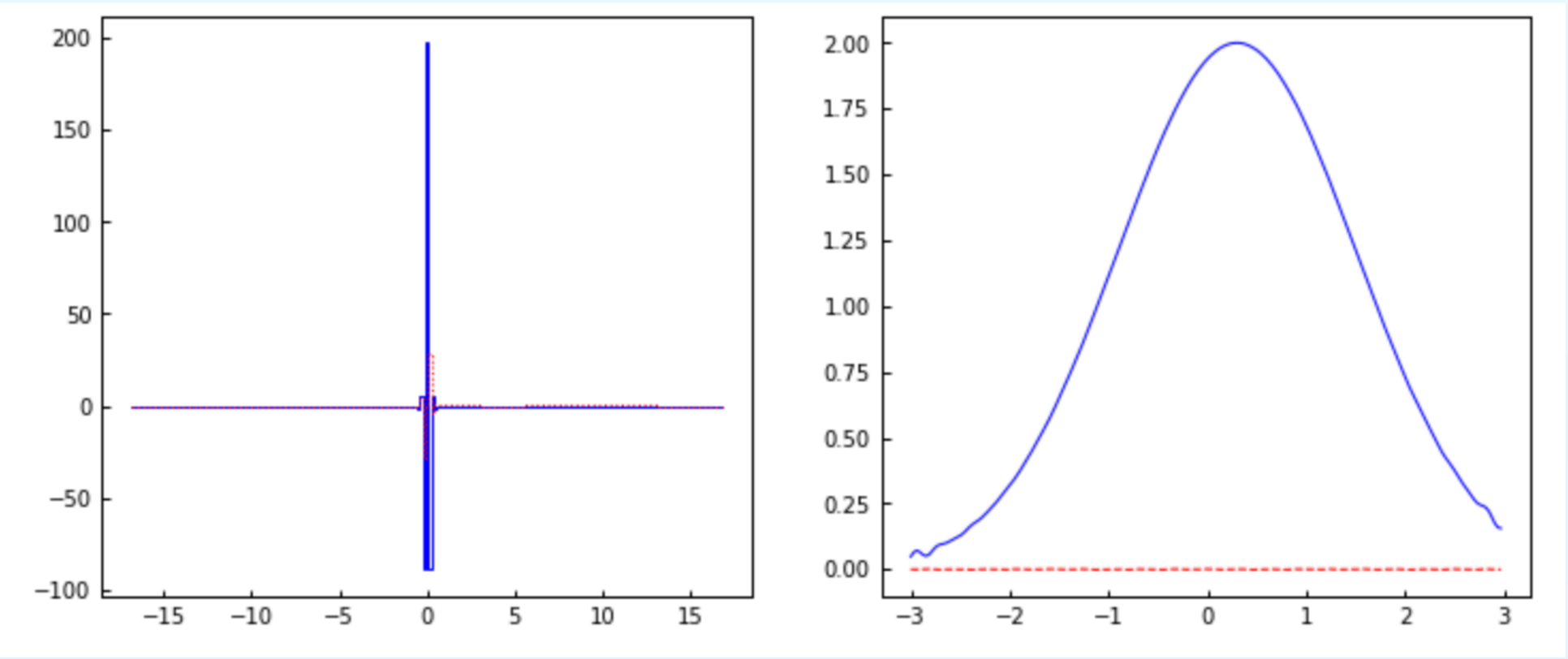
傅里叶变换和滤波

```
1  # ++++ filter width affect the output
2  def filter_fc(freq_arr, y_ft, f_c, df):
3      ix_arr = np.where(np.abs(np.abs(freq_arr)-f_c)<=df)[0]
4      y_flt = np.fft.fftshift(y_ft)
5      y_flt[ix_arr] = (y_ft[ix_arr[0]-1] + y_ft[ix_arr[-1]+1])/2
6      # y_flt[ix_arr] = 0
7      return np.fft.fftshift(y_flt)
8
9  y_flt0 = filter_fc(freq_arr, y_ft, 4, 0.8)
10 y_flt1 = filter_fc(freq_arr, y_flt0, 5, 0.8)
11 # y_flt1 = y_flt0
12
13 y_ifft = np.fft.ifft(y_flt1, n=len(t_arr))
```

filter function

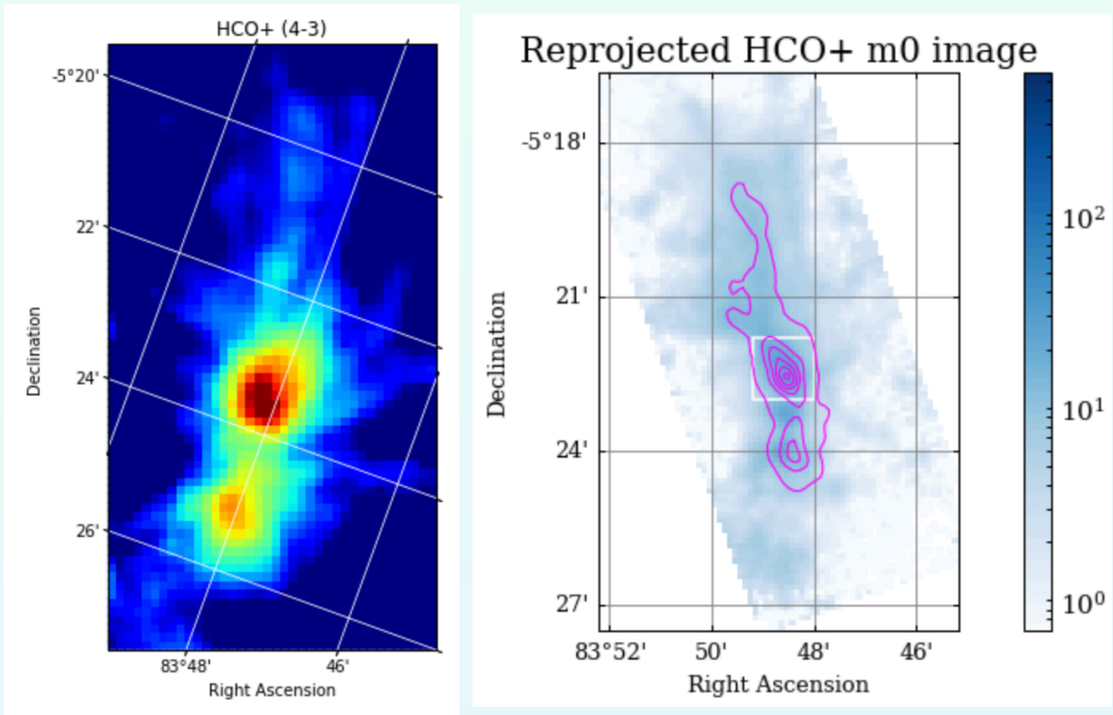
filter apply

inverse FFT transform

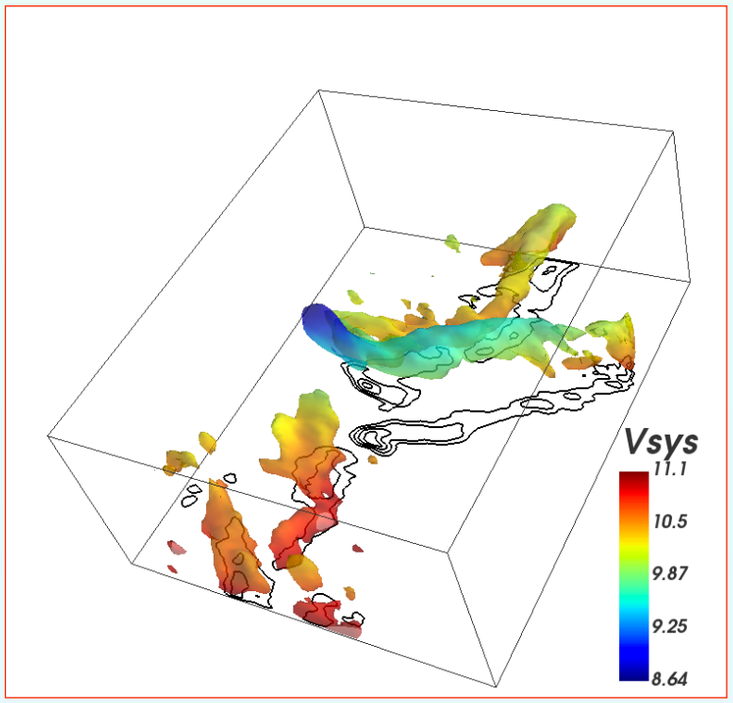


1. 基本数据-图像处理相关天文数据包

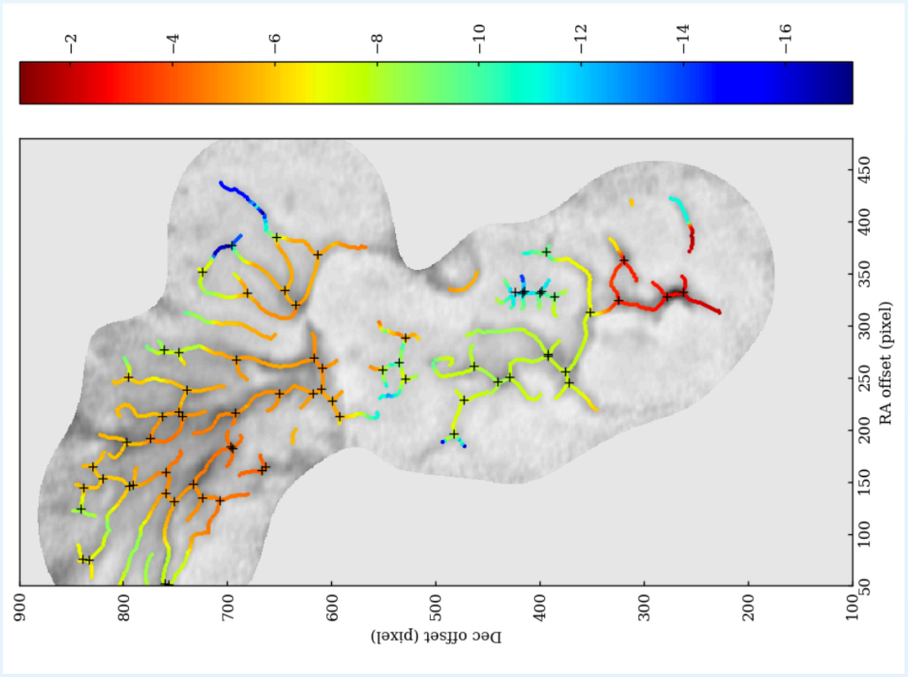
reproject: 不同坐标系投影和转换



spectral-cube: 三维数据转换处理



fil_finder: 线性结构提取



pycupid: 团块结构提取