Hough変換を用いた移動天体の検出

森井幹雄
統計数理研究所
代理発表　浦川聖太郎
日本スペースガード協会

2019.07.10、「木曽シュミットシンポジウム」

内容

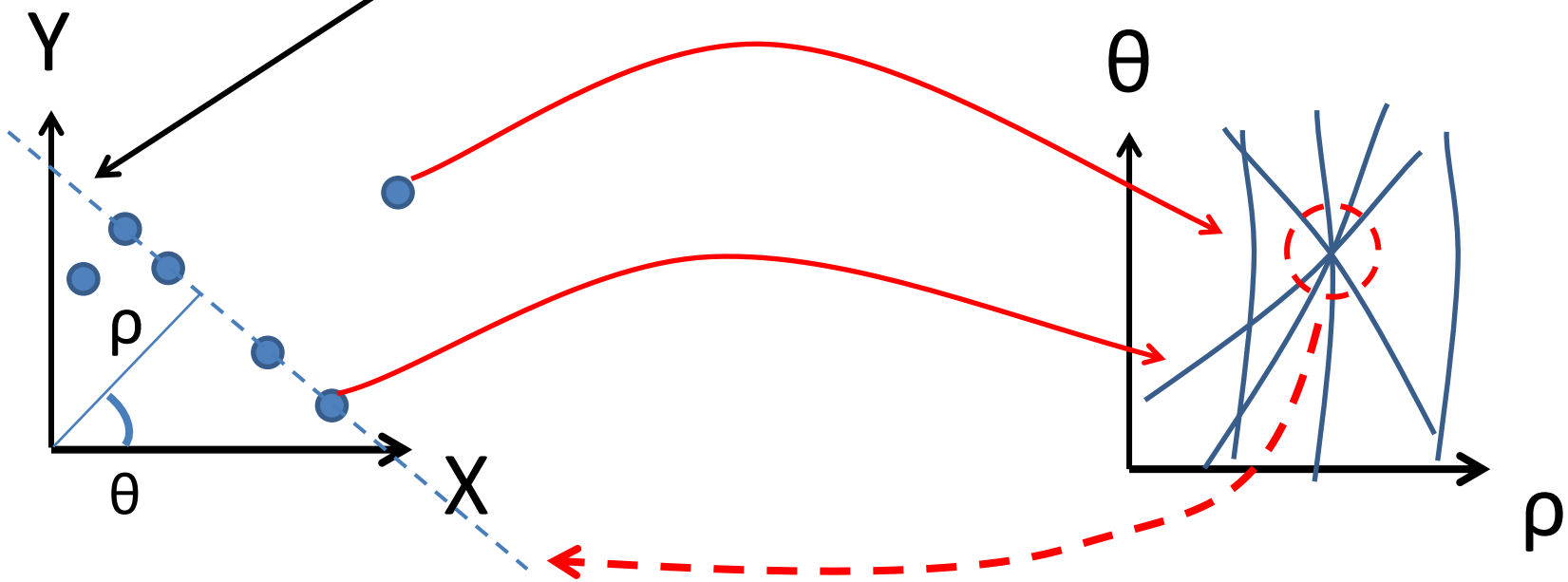
- Hough変換（２次元、３次元）
- デモ
  - Tomo-e Gozen の1 chip のデータ内に対して、Hough変換を用いた移動天体の検出を行った。
  - 前処理
  - Hough変換
  - 直線に沿った変動のチェック
- 実行時間

Hough変換

- Hough変換は画像の中から、直線や円を検出する方法。
- 2次元の中の直線を検出する方法は、よく知られている。
- 移動天体は、画像(平面)×時刻の3次元空間中の直線に沿っている。
- 3次元空間中の直線を検出するために、Hough変換による直線検出法を拡張した。
- 実装は簡単。

Hough変換による直線検出（2次元の場合）

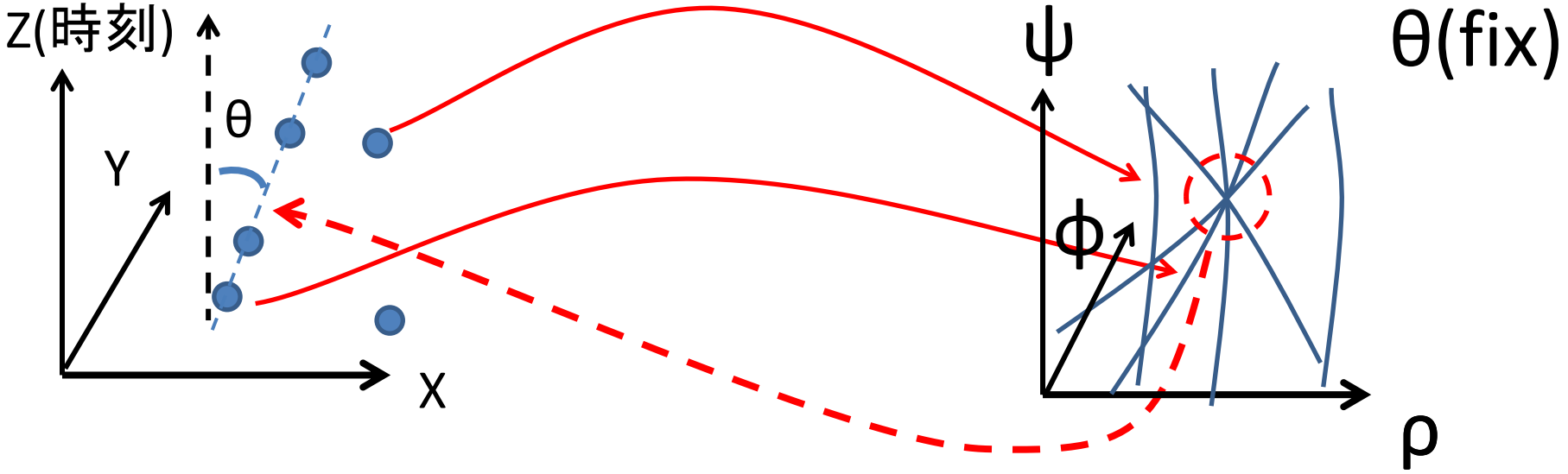
$$\rho = x\cos\theta + y\sin\theta$$



(x, y) 空間上の１点が、(ρ, θ)空間上の１本の曲線に対応する。
(ρ, θ) の空間上で、最も多くの曲線が交わる点に対応する直線が、求めたい(x, y)空間内の直線となる。

Hough変換による直線検出（3次元の場合）

$$\begin{aligned} \rho\cos\phi &= x\cos\psi + y\sin\psi \\ \rho\sin\phi &= -x\sin\psi\cos\theta + y\cos\psi\cos\theta + z\sin\theta \end{aligned}$$



(x, y, z) 空間上の１点が、(ρ, ϕ, ψ, θ)空間上の１枚の曲面に対応する。
Θは、移動天体の速度に相当するパラメータ。Θを fix した場合は、(x, y, z)空間上の１点が、(ρ, ϕ, ψ) 空間上の１本の曲線に対応する。
(ρ, ϕ, ψ) の空間上で、最も多くの曲線が交わる点に対応する直線が、求めたい(x, y, z)空間内の直線となる。この探索を、天体の移動速度θ毎に行う。

Hough変換による移動天体の検出

- 使用方法は2通り。
  - (1) 事前に、各フレームに対して天体検出をし、天体位置（x, y)と時刻(t) のリストを作成する。このリストを入力としてHough 変換を行う。
  - (2) 動画データ(pixel データ)に対して直接使用する（本発表）
- (1) は、各フレームで数σ以上の有意度で検出できているものだけから探索する場合。（複数チップにまたがって広域探索する場合にも使えるかも。）
- (2) 重ね合わせないと検出できないような暗い天体でも可能かも。

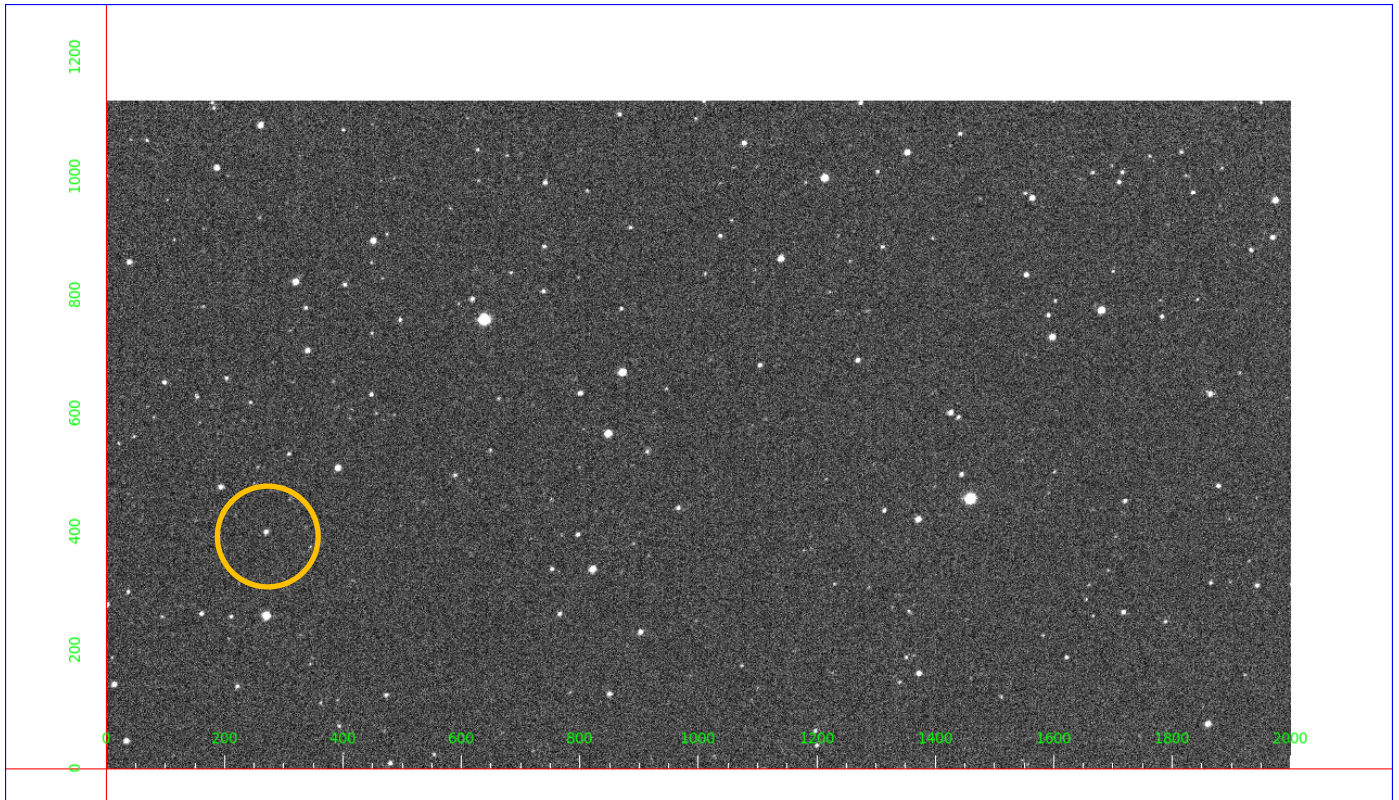
デモ

- Tomoe-Gozen のデータ（浦川さん提供）

  （Tomoe_2010wc9） (2018.07.12のメール)

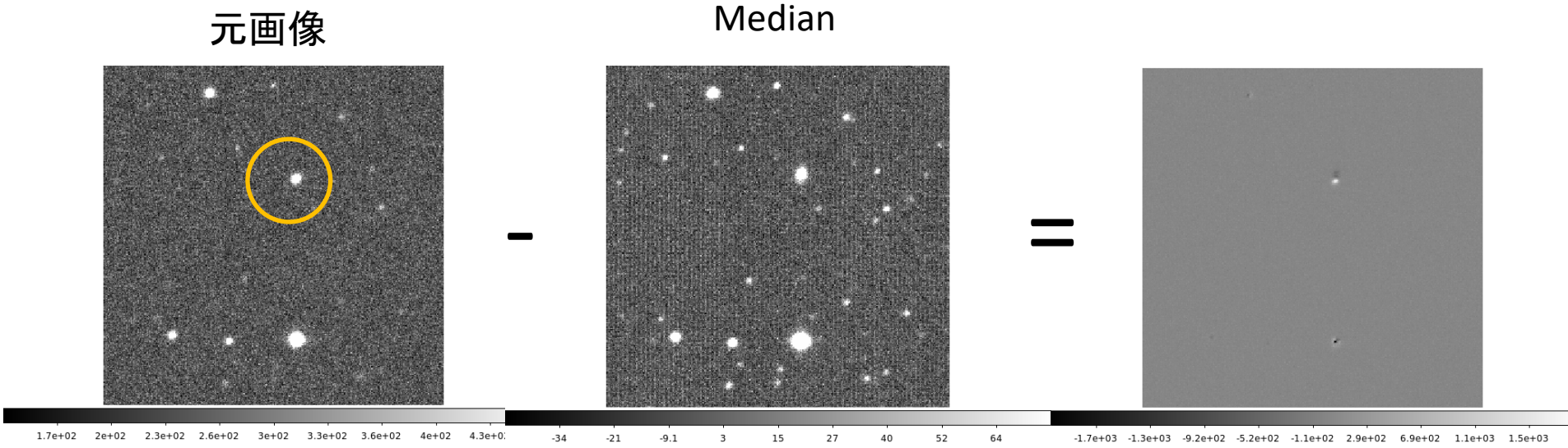
  – 12 frameから成る。2Hz のデータ。

  – 2000 x 1128 (pixel^2)



前処理

- Medianフレームの引き算
  - 背景天体は、Medianフレームを引き算すると消えることが期待されるので、Medianフレームを作成する。
  - 各フレームから Medianフレームを引き算する。

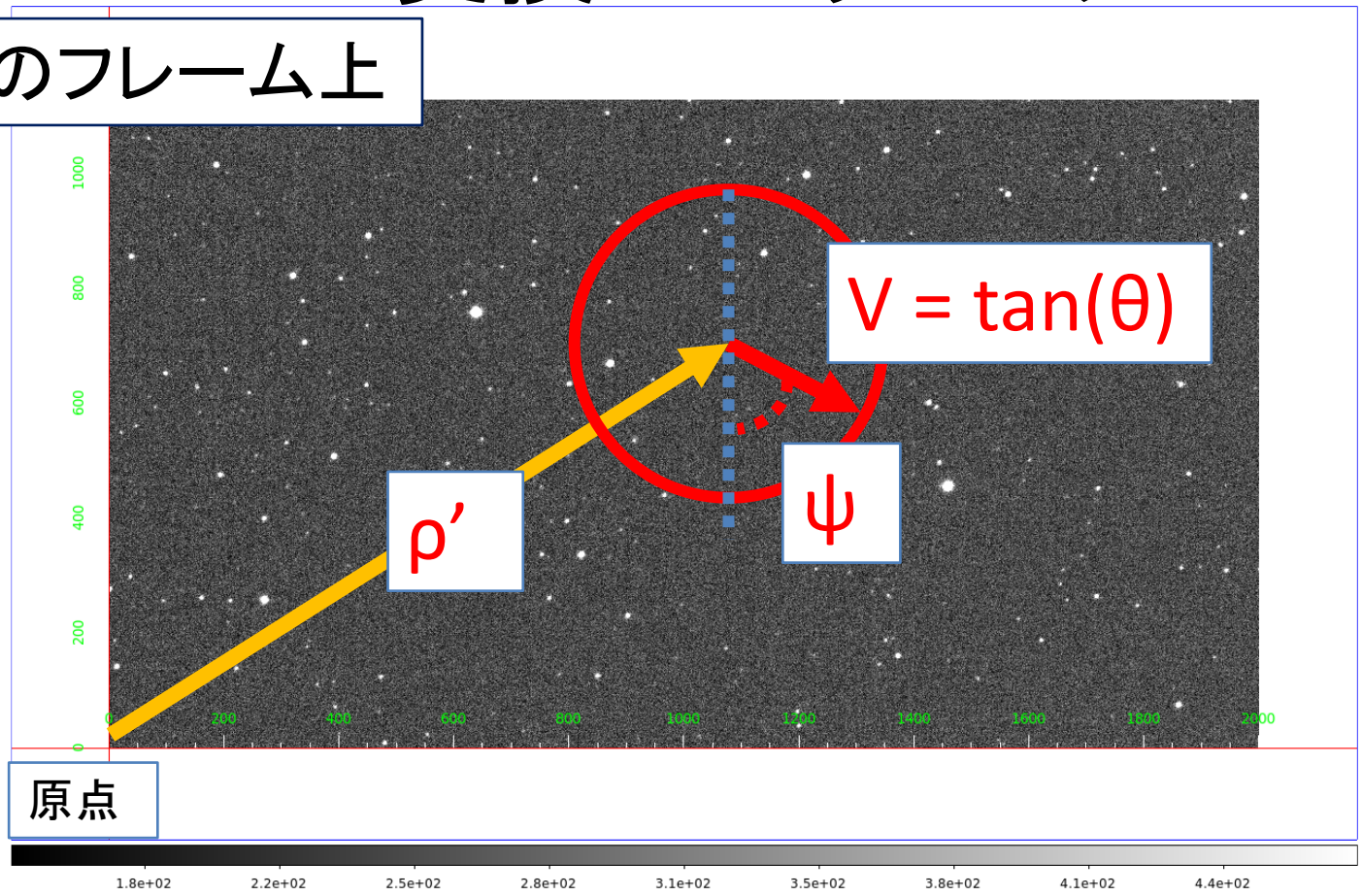


移動天体の検出

- Hough変換を用いる。
- 探索範囲
  - Velocity (pixel/sec) = tan(θ): 0.75 – 3.75, 6 分割
  - (ρ, ϕ, ψ) = (1000 bin, 200 bin, 200 bin)
  - ρ : Sqrt(2000^2 + 1128^2) / 1000 = 2.3 pixelで、psf の幅は大体10 – 20 pixel なので、rhoのbin数は妥当。
- 移動天体は、３次元cube データ中の斜めの直線になる。一方、背景の天体は時刻軸に沿った直線になる。
- Hough 変換の空間上で大きい値に相当する直線を検出する。
- 検出したもののうち、直線に沿った値の変動が大きいものは除く。

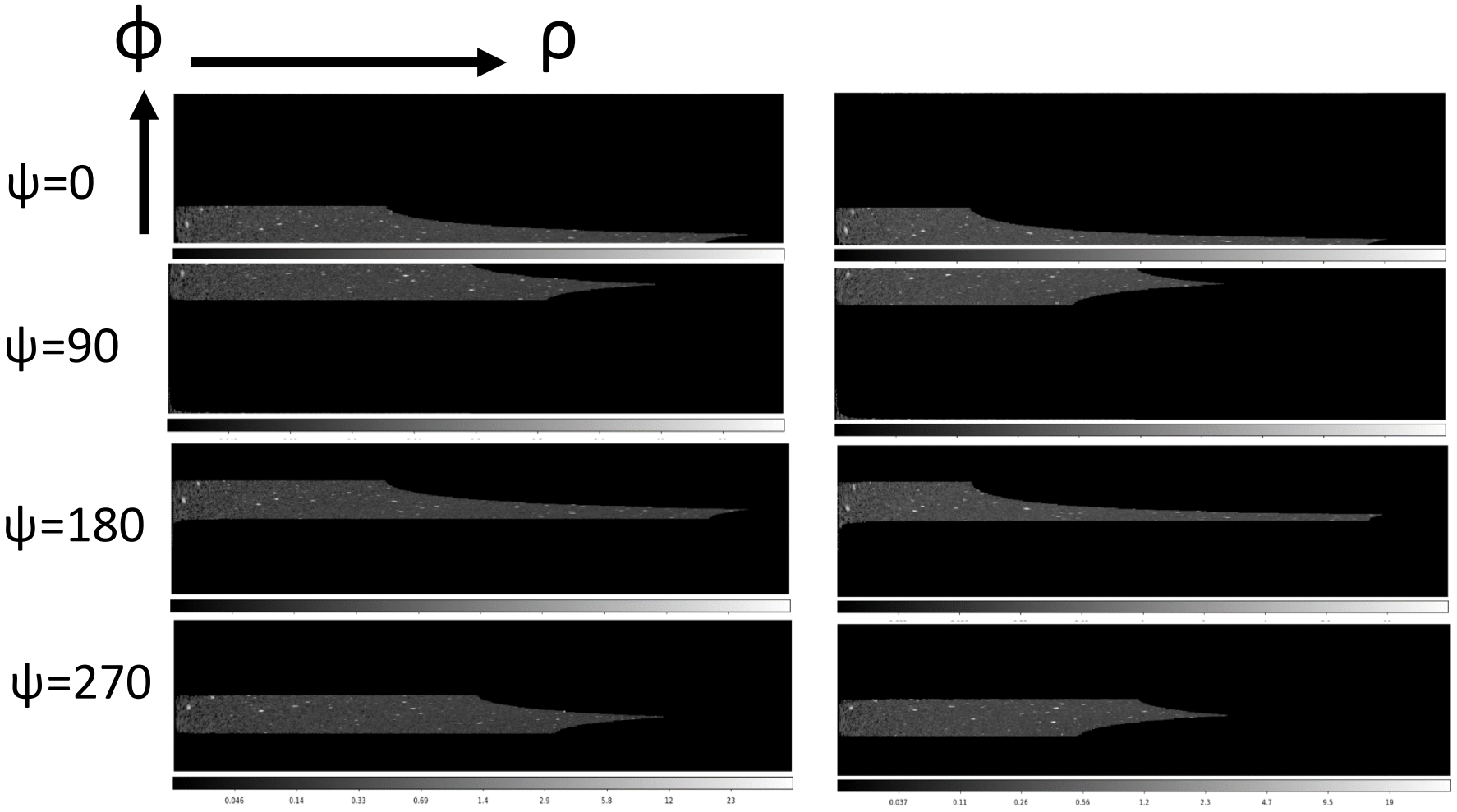
ハフ変換のパラメータ

0枚目のフレーム上



$$\rho' = \mathrm{sqrt}\{\ \rho^2\ (\ \cos^2\phi + \sin^2\phi\ /\ \cos^2\theta)\ \}$$

Hough変換したデータ



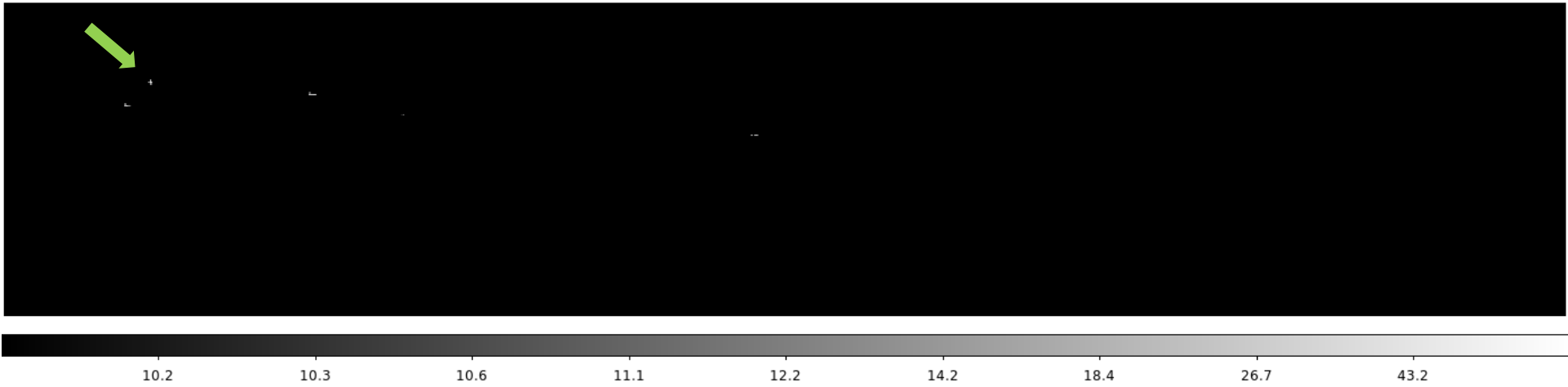
Velocity = 1 (pixel/sec)

Velocity = 2 (pixel/sec)

Hough変換のパラメータ(Velocity, ρ, ϕ, ψ) の4次元空間内で、明るい点が移動天体に対応する。

移動天体に対応する点

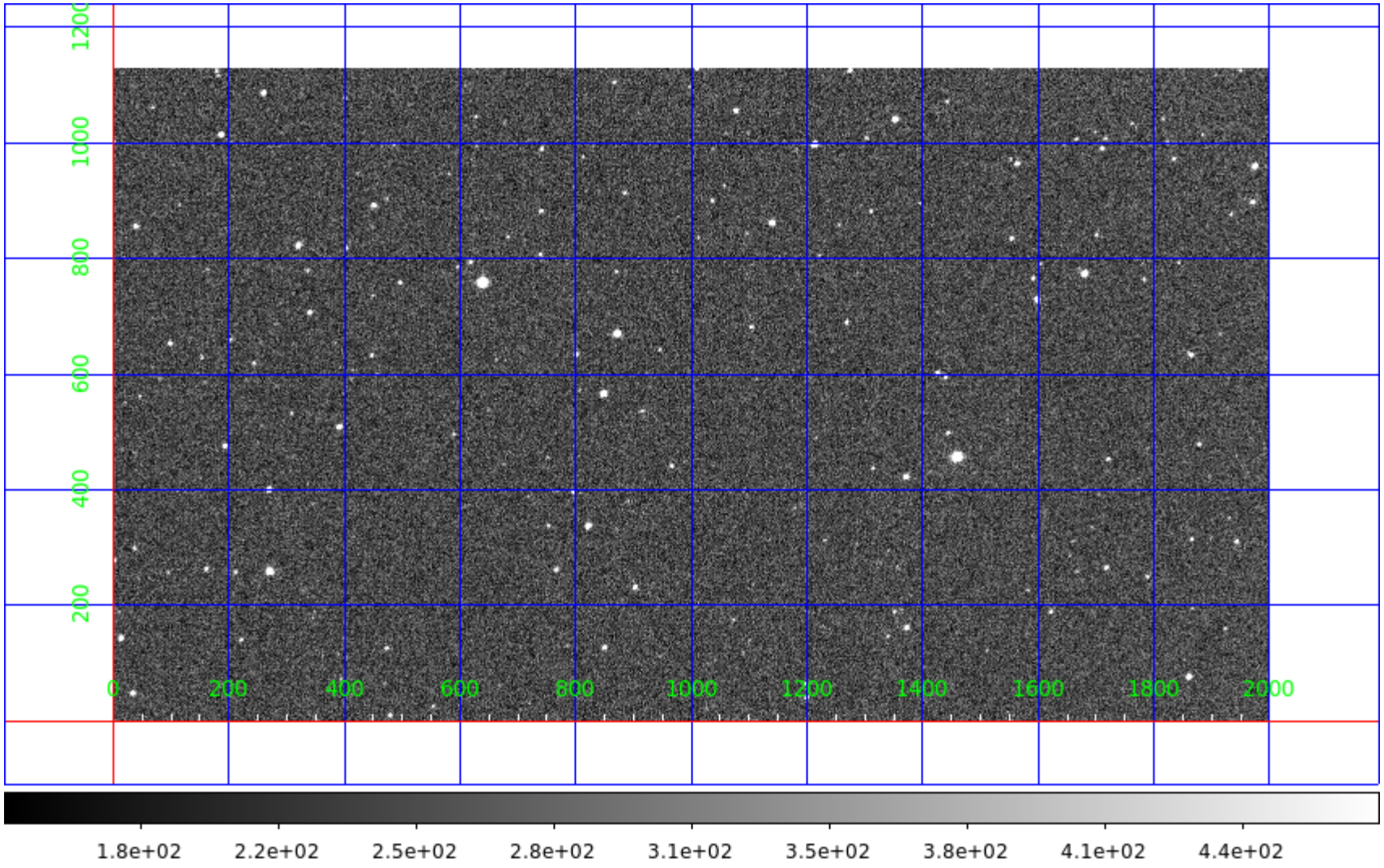
Velocity = 2 (pixel /sec) , ψ = 149 度



他の明るい点は別の背景天体に対応する。移動天体は、軌跡の直線に沿った変動が小さいが、背景天体の場合は変動が大きくなる(PSFの裾から裾を結んだ直線に相当するため)。その違いを用いると、背景天体は除くことができる。

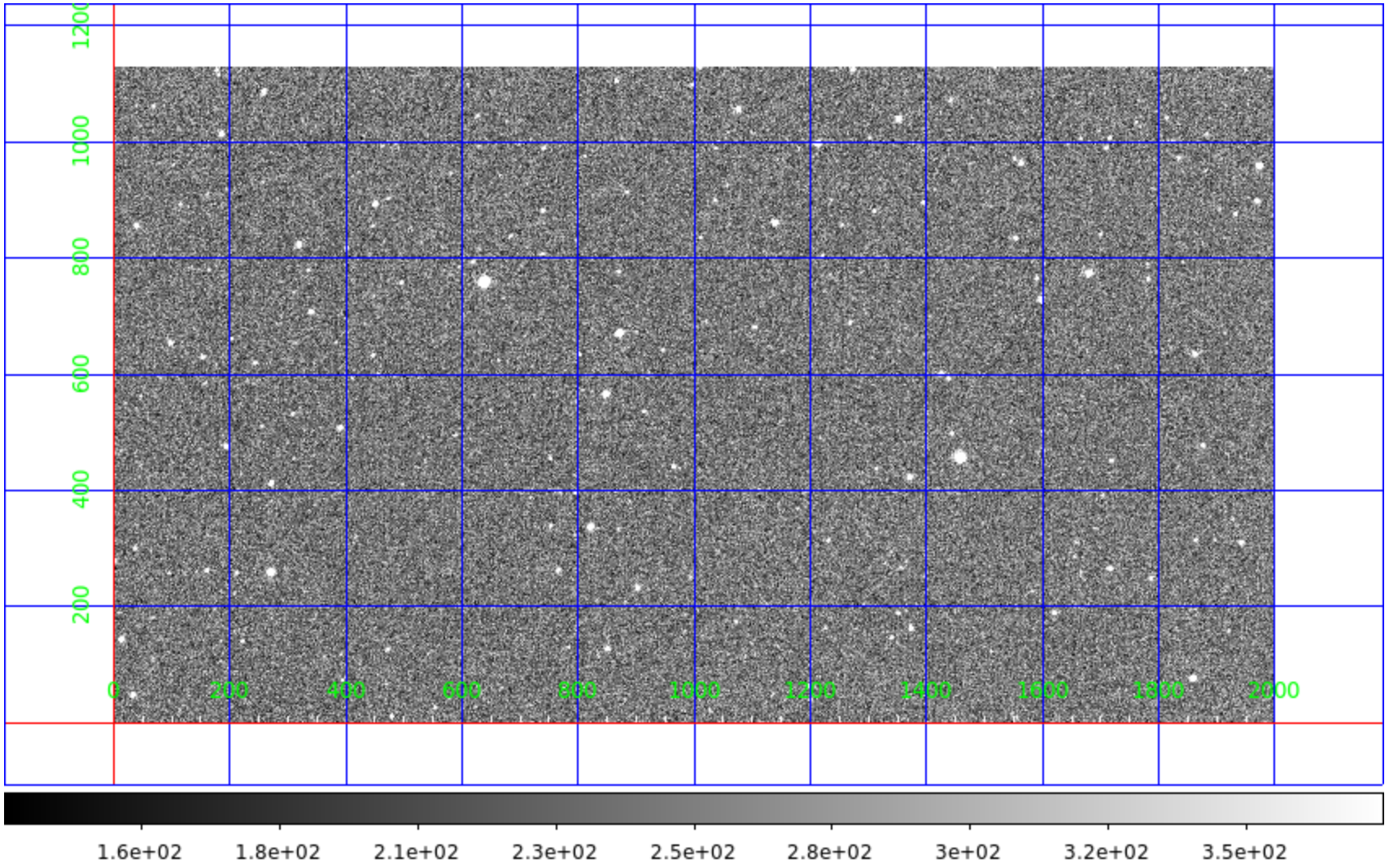
Tomoe_2010wc9 の1 枚目

元画像



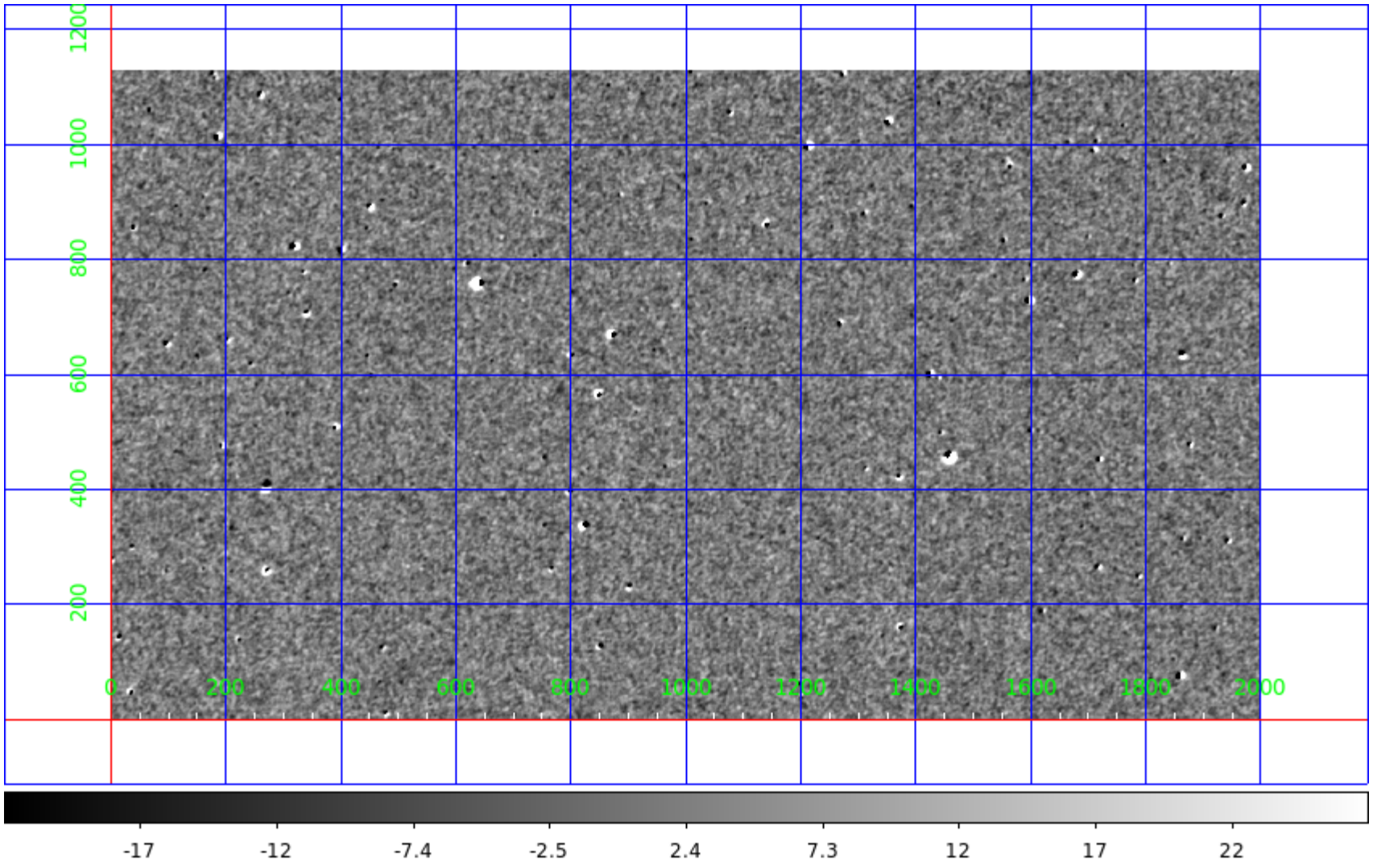
Tomoe_2010wc9 の12 枚目

元画像



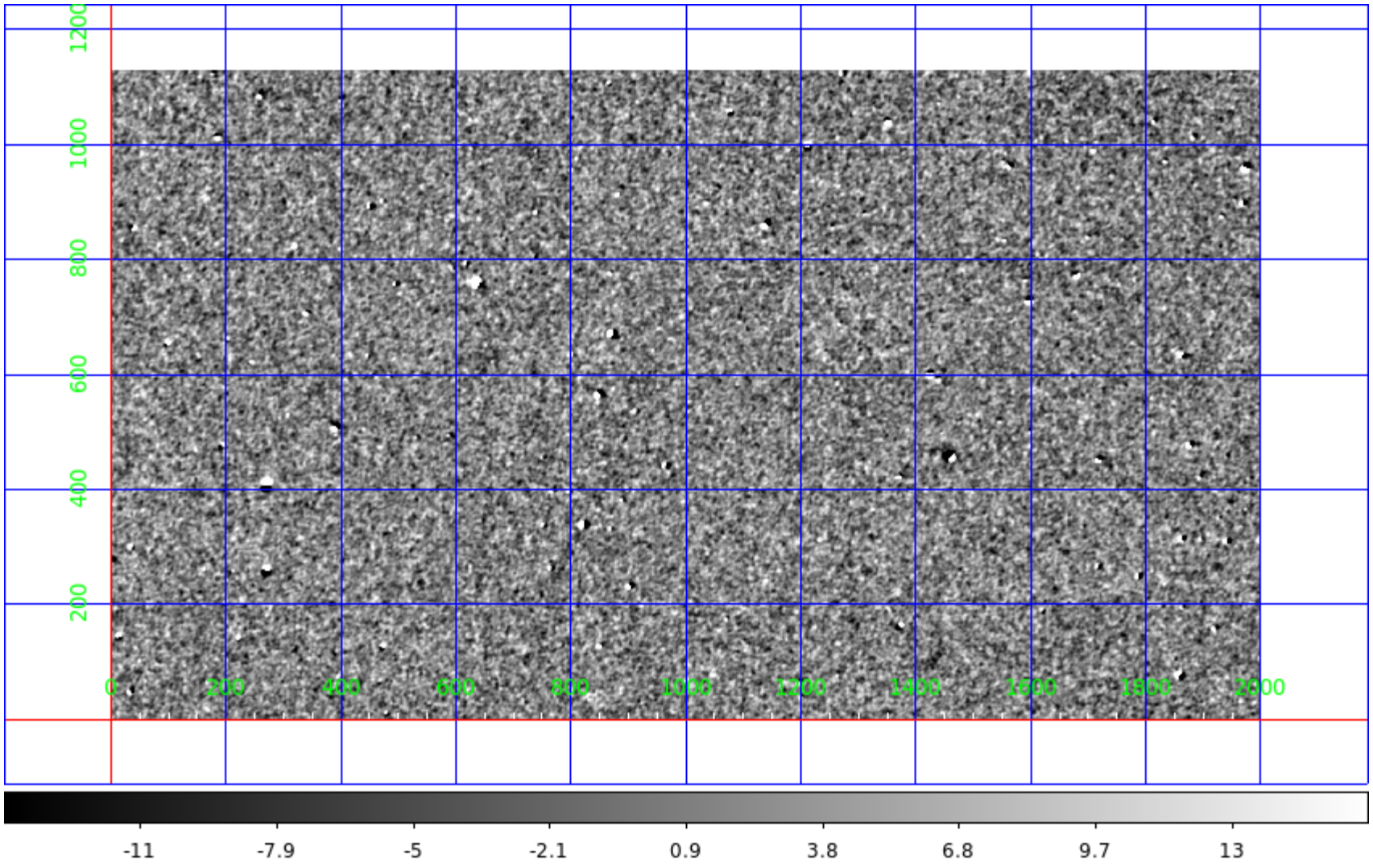
Tomoe_2010wc9 の1 枚目

前処理後



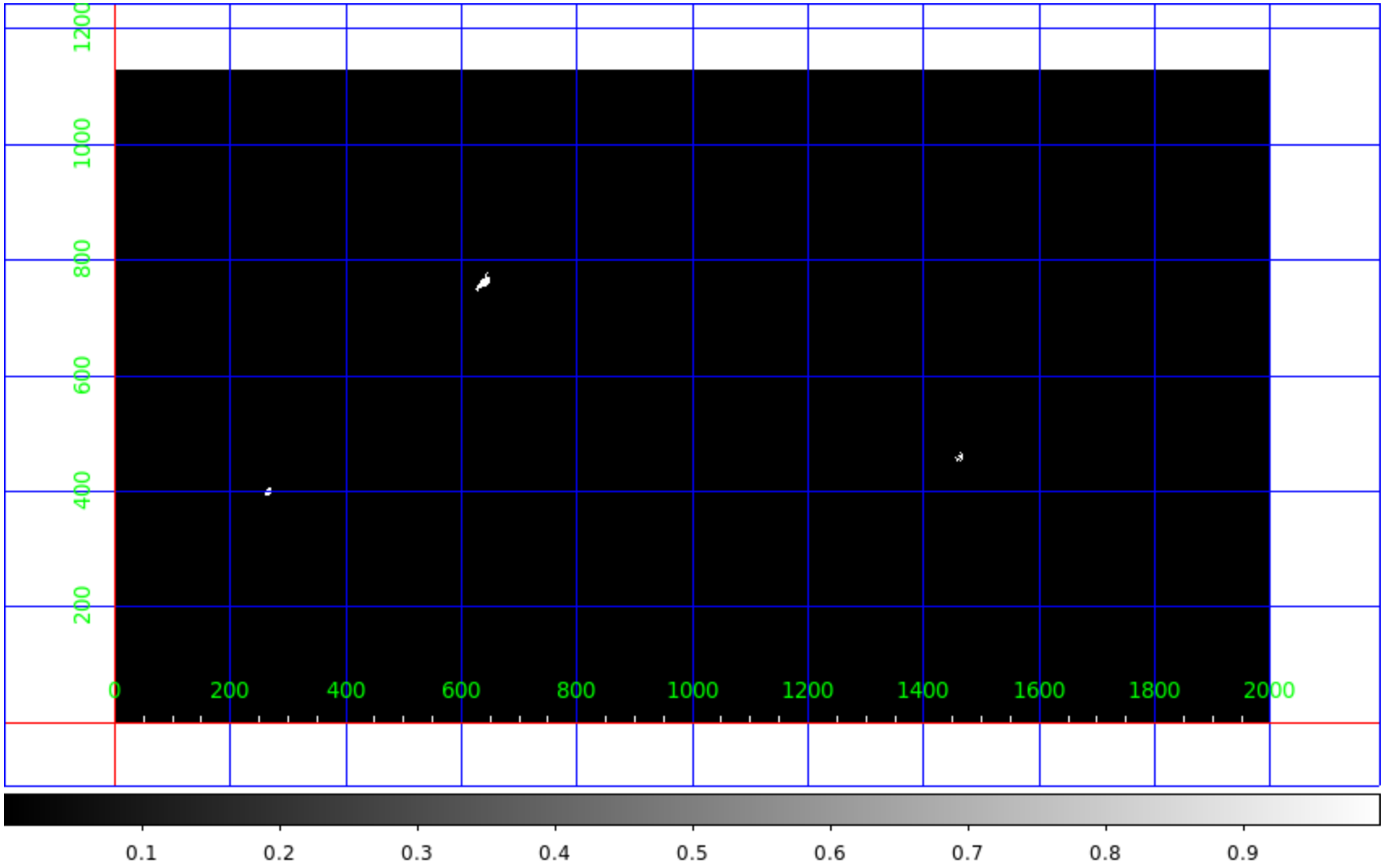
Tomoe_2010wc9 の12 枚目

前処理後



Tomoe_2010wc9 の1 枚目

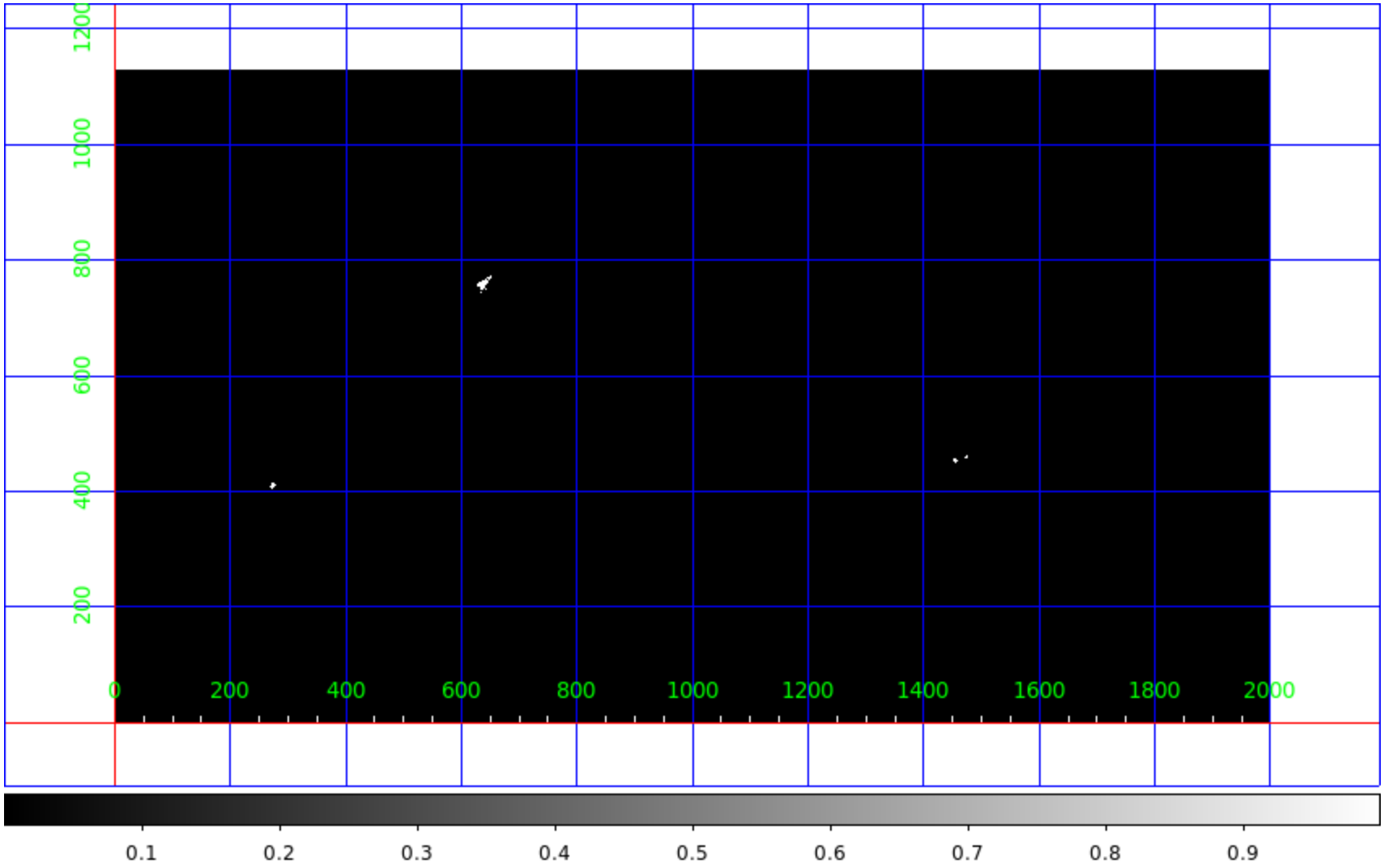
Hough変換を用いて検出した天体の場所



Hough変換の空間(vel, rho, phi, psi) 上で、ピクセル値上位500点に対応する
Cubeデータ上の直線500本をプロット。

Tomoe_2010wc9 の12 枚目

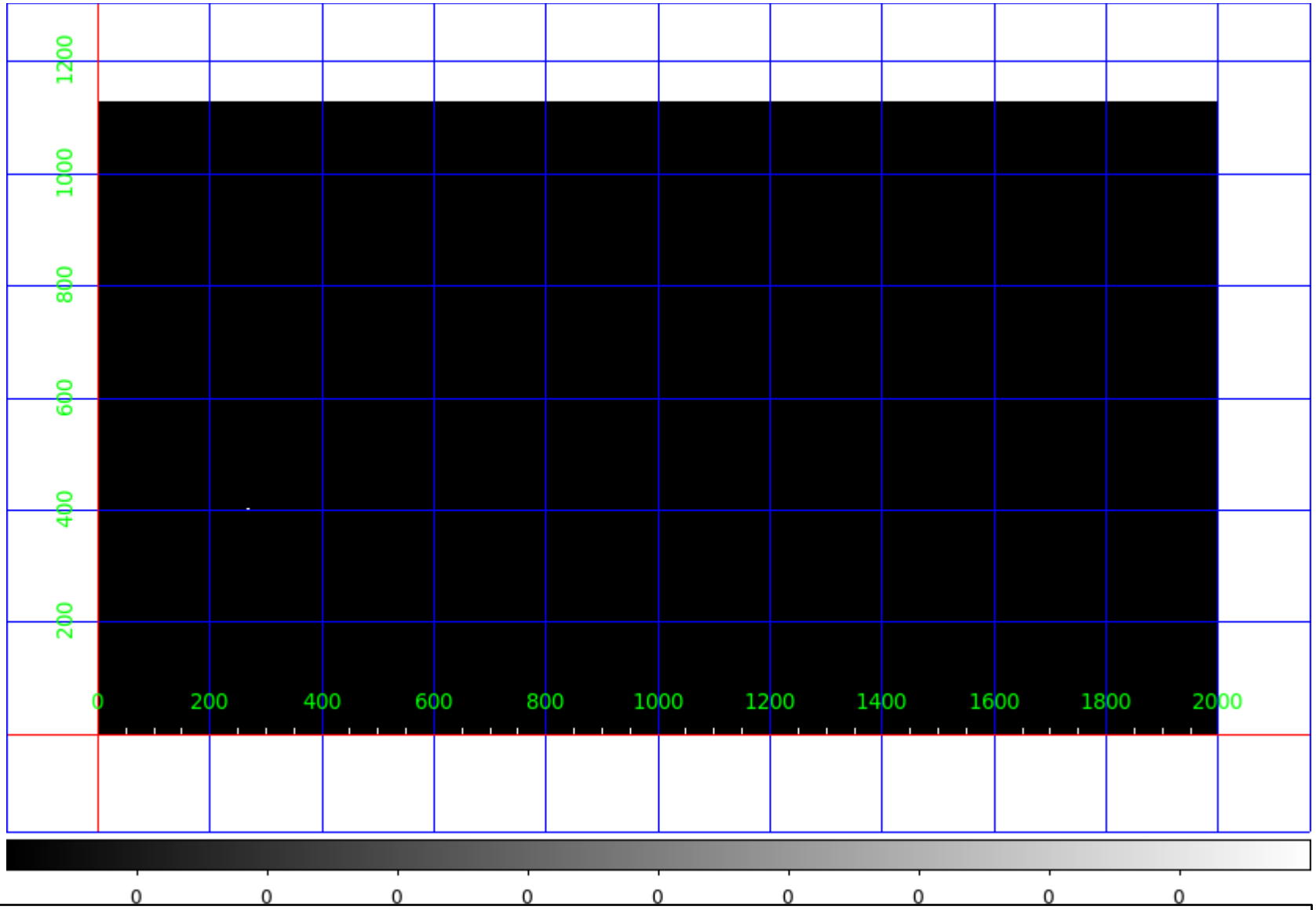
Hough変換を用いて検出した天体の場所



Hough変換の空間(vel, rho, phi, psi) 上で、ピクセル値上位500点に対応する
Cubeデータ上の直線500本をプロット。

Tomoe_2010wc9 の1 枚目

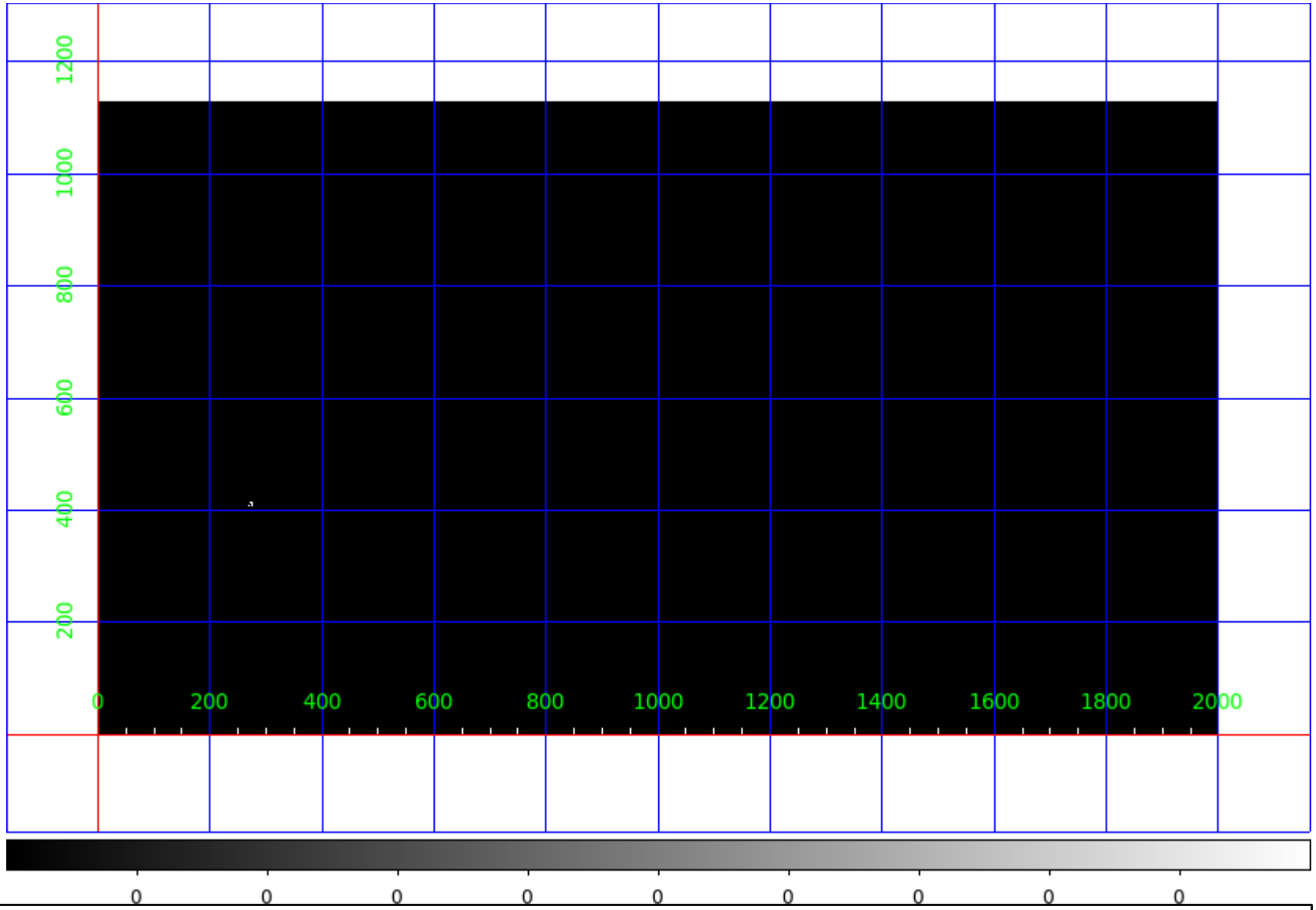
Hough変換 + 直線に沿った変動が50%以下



Hough変換の空間(vel, rho, phi, psi) 上で、ピクセル値上位50点に対応するもの。かつ、直線に沿ったpixel の変動が50%以下のものをプロット。

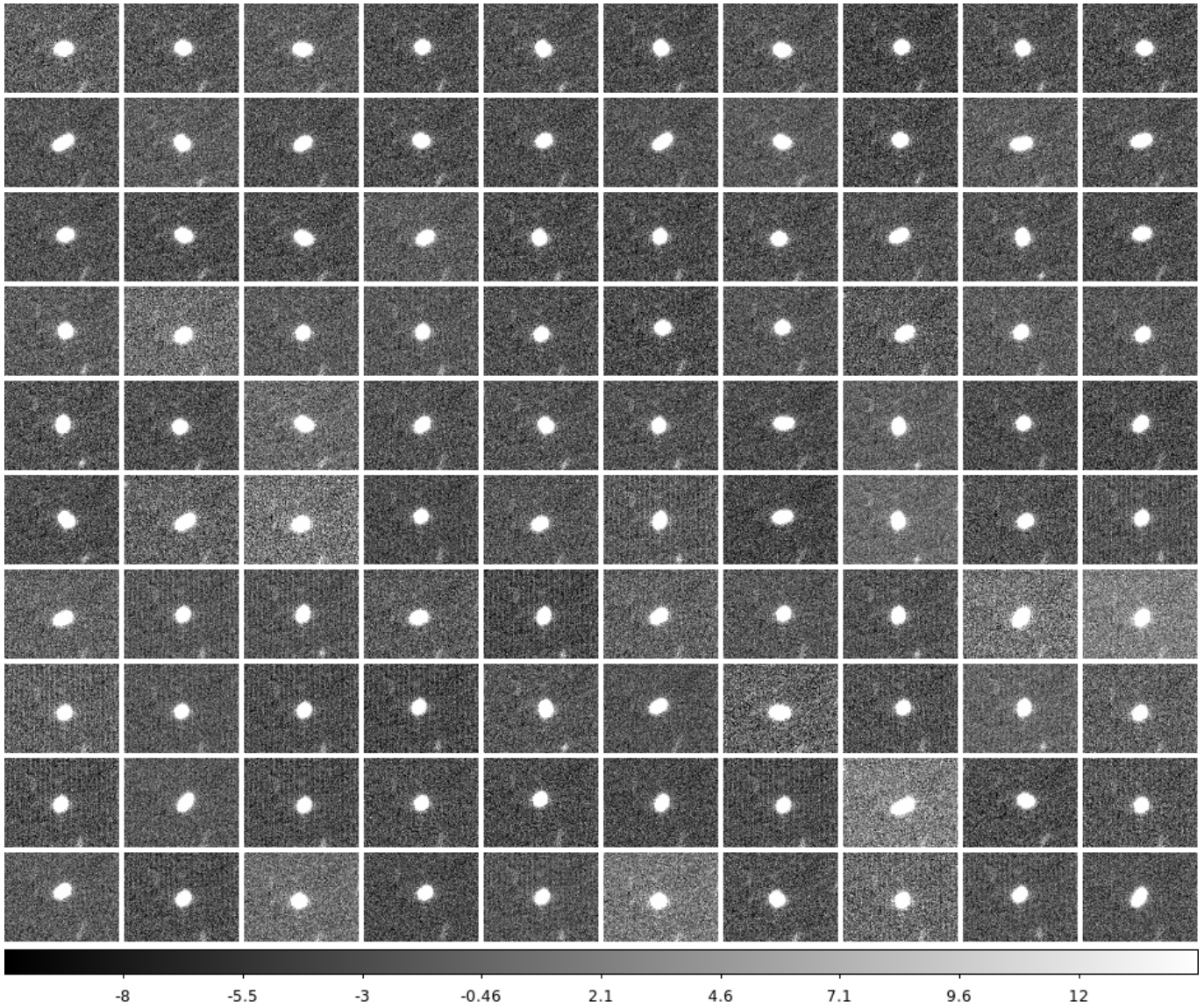
Tomoe_2010wc9 の12 枚目

Hough変換 + 直線に沿った変動が50%以下

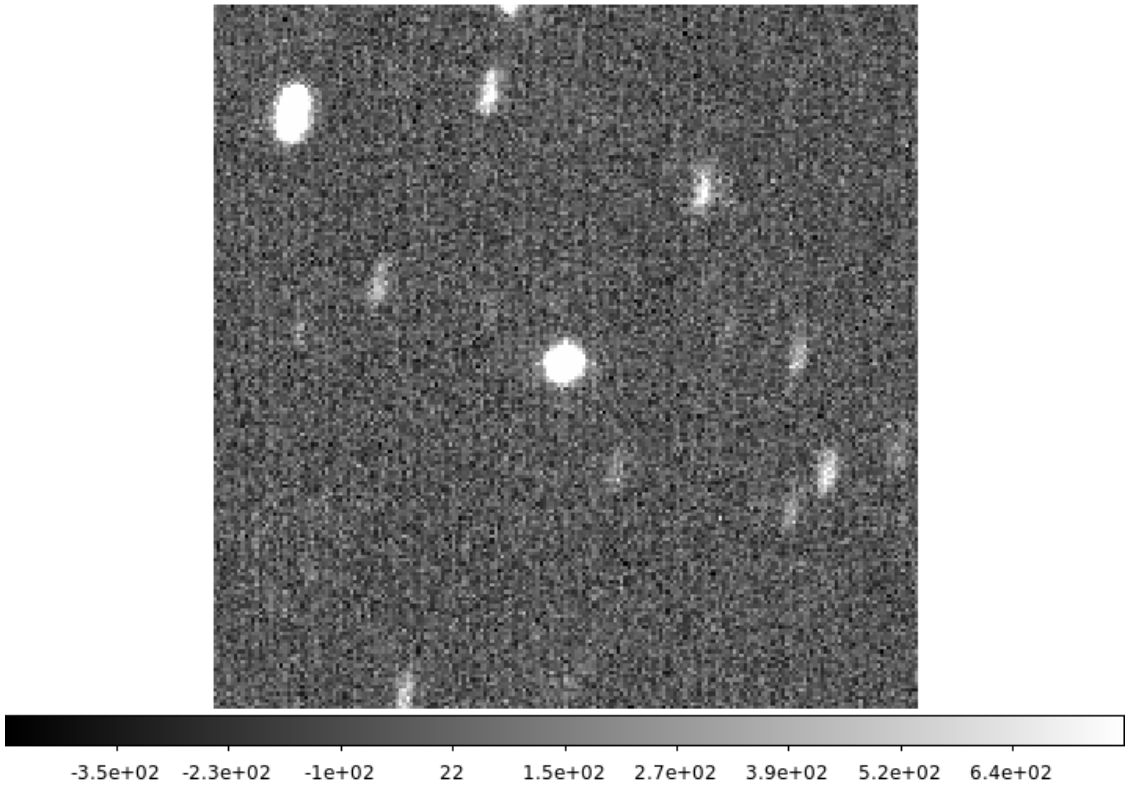


Hough変換の空間(vel, rho, phi, psi) 上で、ピクセル値上位50点に対応するもの。かつ、直線に沿ったpixel の変動が50%以下のものをプロット。

Shift-and-add した結果(上位100個)



- この中から、星像の幅がもっとも小さくなるparameterを選ぶと、



以上の解析で、PSFと比較するようなイメージ解析は行っていない。
単純な方法で移動天体が検出できた。

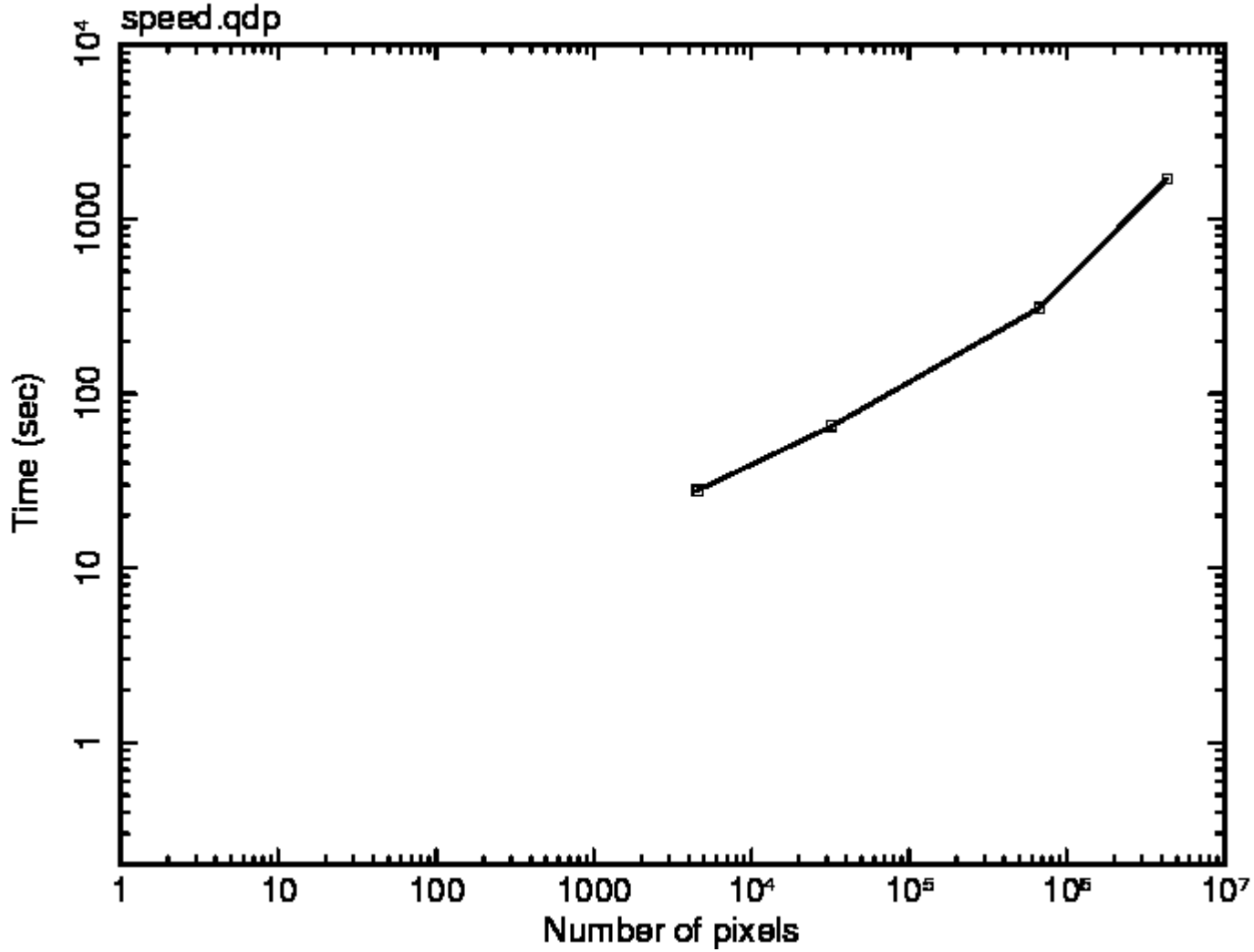
実行時間

- 探索範囲
  - Velocity (pixel/sec) = tan(θ): 0.75 – 3.75, 6 分割
  - (ρ, ϕ, ψ) = (1000 bin, 200 bin, 200 bin)

- 使用CPU: Intel Core-i7 (3.20GHz)
  - 1 core を使用。

- 前処理: 1.1e3 sec
- Detection :
  - 各frame 5 sigma以上のpixel (2.7e7 の0.12%): 65 sec
  - 2 sigma (2.7e7 の2.5%): 3.1e2 sec
  - 1 sigma (2.7e7 の16%): 1.7e3 sec

実行時間



各フレームで天体検出をして、天体の位置と時刻のリストに対して、Hough変換を行う場合は、2000 点くらいに相当するので、実行時間は、10秒程度。

まとめ

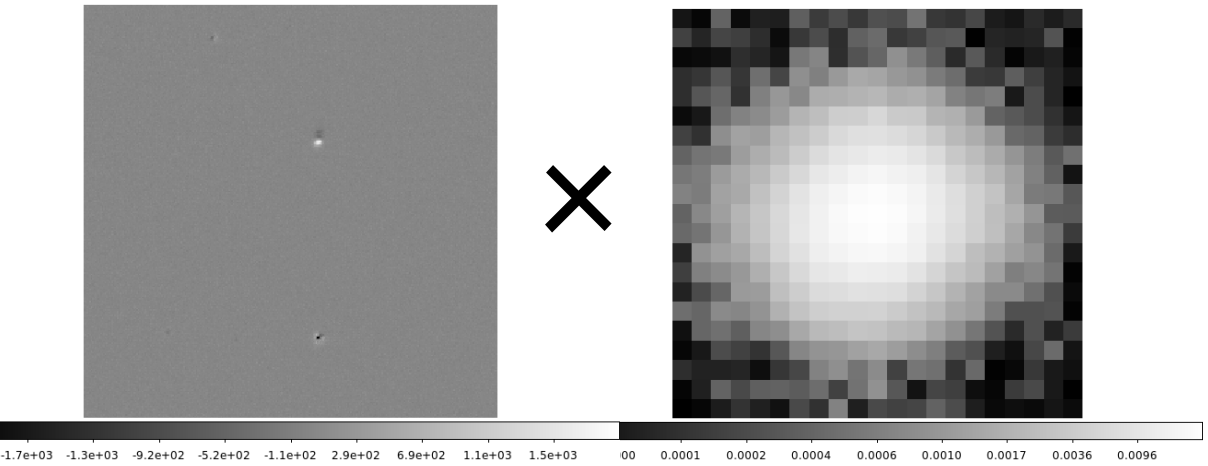
- 3次元中の直線を検出するHough変換の式を導出した。
- Tomo-e Gozen 1chip の領域を使用して移動天体の検出を試みた。
- Hough変換に加えて、直線に沿ったpixel値の変動量を用いることで、移動天体だけを選び出すことができる。
- 各フレームでは検出できないが、重ね合わせると検出できるような暗い移動天体の検出もできるかもしれない。計算時間は、30分～1時間くらいかかる（ (1sigma)の場合を参照）。
- 各フレームでSextractorなどを用いて天体を検出し、天体の位置と時刻のリストを使用する場合であれば、10秒程度で検出できる。
- Hough変換は、撮像画像のうち数枚が恒星と重なるような場合の移動天体検出に有効かもしれない。

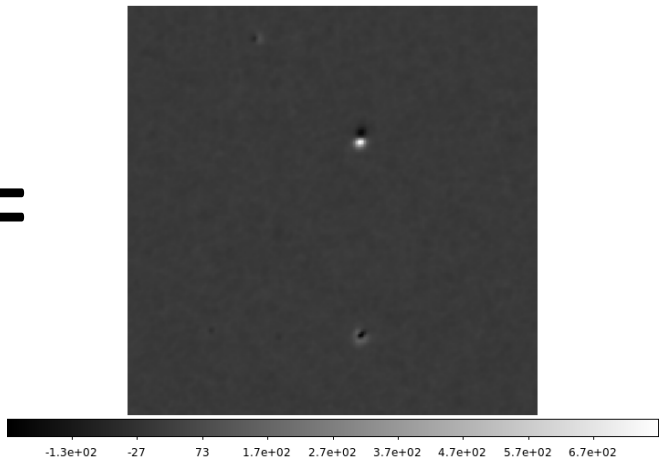
前処理

- Sky backgroundレベルの補正
  - 各フレーム毎に、pixel値の平均と分散を計算し、平均値を引き算して、sky background の値がゼロになるようにする。ただし、明るい天体の影響を除くため、以下の繰り返し計算を行なう。
  - (1) 平均と分散の計算
  - (2) pixel 値が 5σ以上平均値からずれた領域をマスクする。
  - マスク領域のpixel数が増えなくなるまで、(1) (2) の計算を繰り返す。（大抵、５回程度で終了。）

前処理

- PSF の作成
  - Median フレームを引いた画像を見ると分かるように、明るい天体のイメージは変な形状になる（PSFの残差なので）。一方、移動天体の形状は星のPSFに近い。
  - PSFとして、明るい天体の画像をそのまま用いる。
- PSFとの一致度合を表示する。
  - 各ピクセルの周りにPSFのイメージと同じサイズの領域をとり、画像の値とPSFの値との掛け算の総和を求める。画像データと、PSFとの内積をとることに相当する。
  - ここで、PSFの形状を学習させた機械学習判別器を用いて、そのスコア値を使えばよいかもしれない。





Hough変換の式の導出

イメージを $x-y$ 平面とし、時間方向を z 軸とする。$x_1=(\rho,0,0)^T$, $x_2=(\rho,0,1)^T$ $(\rho\geq 0)$ を通る直線 l を原点の周りにオイラー角で回転させた直線 l' の方程式を求めてみる。このように生成した直線 l' は 3 次元空間内の全ての直線を表すことができる。直線 l を、z 軸の周りに $\phi\in[0,2\pi]$、x 軸の周りに $\theta\in[0,\pi/2]$、z 軸の周りに $\psi\in[0,2\pi]$、だけ、この順に 3 回、回転させる。これらの回転行列は、

$$M_z(\phi)=\begin{pmatrix}\cos\phi & -\sin\phi & 0\\ \sin\phi & \cos\phi & 0\\ 0 & 0 & 1\end{pmatrix}$$

$$M_x(\theta)=\begin{pmatrix}1 & 0 & 0\\ 0 & \cos\theta & -\sin\theta\\ 0 & \sin\theta & \cos\theta\end{pmatrix}$$

$$M_z(\psi)=\begin{pmatrix}\cos\psi & -\sin\psi & 0\\ \sin\psi & \cos\psi & 0\\ 0 & 0 & 1\end{pmatrix}$$

$$\begin{pmatrix}x'\\ y'\\ z'\end{pmatrix}=M_z(\psi)M_x(\theta)M_z(\phi)\begin{pmatrix}x\\ y\\ z\end{pmatrix}$$

Hough変換の式の導出

直線 l' の方程式は、

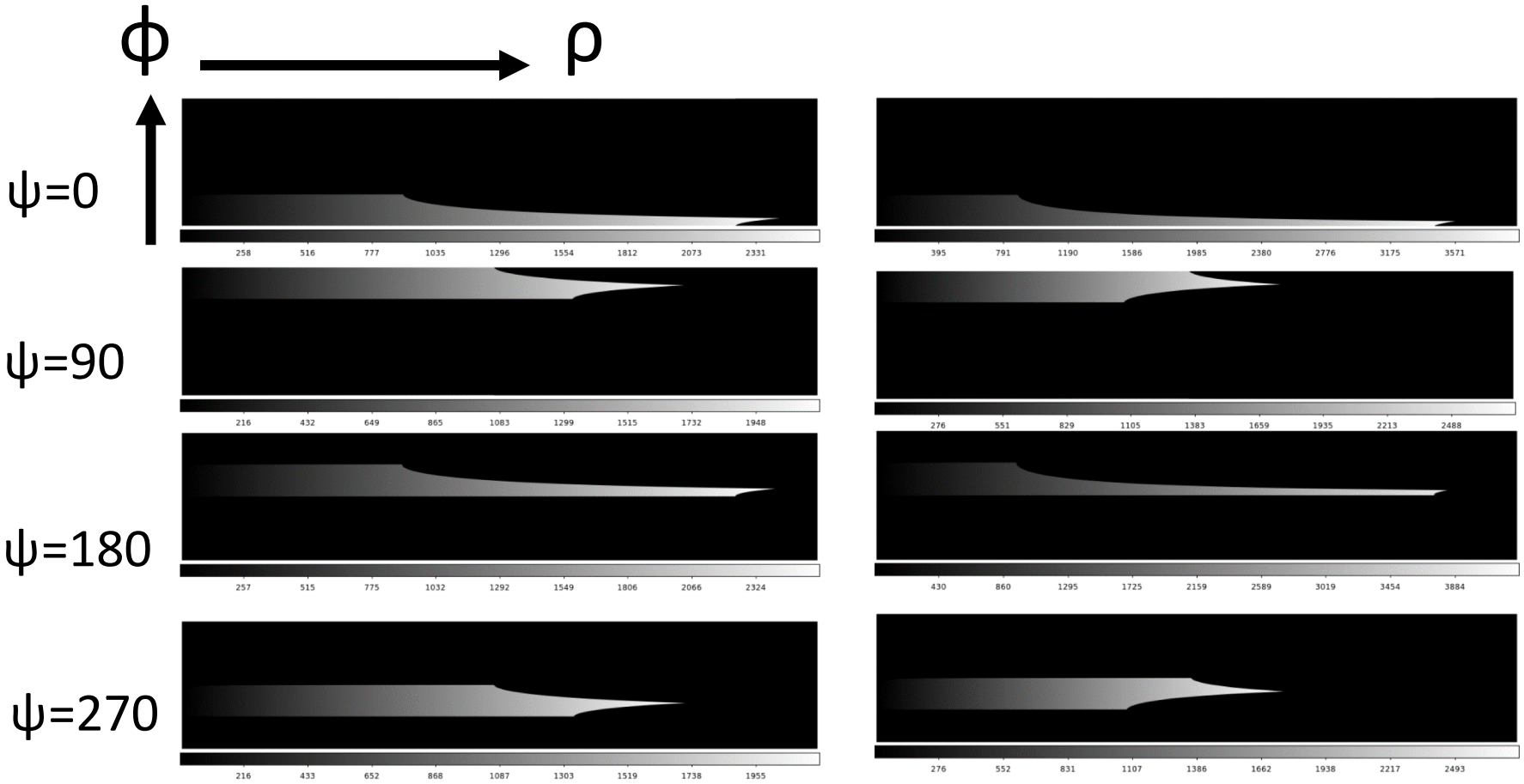
$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = x'_1 + t(x'_2 - x'_1) = \rho \begin{pmatrix} \cos\psi\cos\phi - \sin\psi\cos\theta\sin\phi \\ \sin\psi\cos\phi + \cos\psi\cos\theta\sin\phi \\ \sin\theta\sin\phi \end{pmatrix} + t \begin{pmatrix} \sin\psi\sin\theta \\ -\cos\psi\sin\theta \\ \cos\theta \end{pmatrix}.$$

ただし、$t \in R$.

この式から、t を消去すると、

$$\begin{aligned} \rho\cos\phi &= x\cos\psi + y\sin\psi \\ \rho\sin\phi &= -x\sin\psi\cos\theta + y\cos\psi\cos\theta + z\sin\theta \end{aligned}$$

規格化のデータ



Velocity = 1 (pixel/sec)

Velocity = 2 (pixel/sec)

3d データのピクセルの値をすべて1にセットして、Hough変換を行って作ったイメージ。

メモリサイズの検討

- Tomo-e Gozen 1 chip は、2000 x 1000 pixel
- 原点をchip の真ん中に取る。
- 探索速度(θ)を固定したとして、
  - (ρ, ϕ, ψ)の3次元空間で探索する。
  - 天体のPSFの広がりが10 pixelと仮定し、10 pixel の精度で直線を検出しようとすると、
  - ρ : sqrt(500^2 + 1000^2) / 10 = 110 pixel
  - ϕ: 2 π ρ / 10 = 700 pixel
  - ψ: 2 π ρ / 10 = 700 pixel
  - 110 x 700 x 700 x 8 byte(double型) = 5.4e7 pixel x 8 byte = 4.3e8 byte = 430 MBのメモリが必要