平方與倒數交叉代變 → 負責萃取所有離散時間數據
矩陣{X{t},X{Y}{t}, X{t}′,X{Y}{t}′, X{t}′′,X{Y}{t}′′}交叉做時間平方差分與倒數差分{X,X{Y}, X′,X{Y}, X′′,X{Y}′′}{t-Δt} = 1/({X{t},X{Y}{t}, X{t}′,X{Y}{t}′, X{t}′′,X{Y}{t}′′} - {X,X{Y}, X′,X{Y}, X′′,X{Y}′′}{t-Δt}^2)^2
切交線時不變 → 負責萃取離散時間數據裡的不變特徵
{T = {cosθ, sinθ}, {cosθ, sinθ}′, {cosθ, sinθ}′′ | (tan (θ - μ{X{t},X{Y}{t}, X{t}′, X{Y}{t}′, X{t}′′ , X{Y}{t}′′}{1}}) σ{X{t},X{Y}{t}, X{t}′, X{Y}{t}′, X{t}′′ , X{Y}{t}′′}{1}}) * (cot (θ + μ{X{t},X{Y}{t}, X{t}′, X{Y}{t}′, X{t}′′ , X{Y}{t}′′}{1}}) σ{X{t},X{Y}{t}, X{t}′, X{Y}{t}′, X{t}′′ , X{Y}{t}′′}{1}}) = A{{X{t},X{Y}{t}, X{t}′, X{Y}{t}′, X{t}′′ , X{Y}{t}′′}{1}}}

T•{X,X{Y}, X′,X{Y}, X′′,X{Y}′′}{Δt} = ((1/T)•(1/{X,X{Y}, X′,X{Y}, X′′,X{Y}′′}{(t-Δt)′}) - T•{X,X{Y}, X′,X{Y}, X′′,X{Y}′′}{Δt})^2

導入一份本地視頻{X{t},X{Y}{t}}, 並且代換進{X{t},X{Y}{t}, X{t}′,X{Y}{t}′, X{t}′′,X{Y}{t}′′}, (t-Δt)′表達時間軸倒轉，Δt 表達瞬時率，X{t}′,X{Y}{t}′, X{t}′′,X{Y}{t}′′表達一階變化率、二階變化率。3D 播放萃取結果：T•{X,X{Y}, X′,X{Y}, X′′,X{Y}′′}{Δt} = ((1/T)•(1/{X,X{Y}, X′,X{Y}, X′′,X{Y}′′}{(t-Δt)′}) - T•{X,X{Y}, X′,X{Y}, X′′,X{Y}′′}{Δt})^2

html canvas code.

每個特徵值對應的 Z 軸高度都會獲得相當於它自己的指數增幅。
這個版本顯得更差了，看來是我的思路錯了，應該直接高通 Z 軸高度，顯高不顯低。

我們可以繼續求時間再反轉，(t-Δt)′′二階級項：

矩陣 T•{X,X{Y}, X′,X{Y}, X′′,X{Y}′′}{Δt}交叉做時間平方差分與倒數差分{X,X{Y}, X′,X{Y}, X′′,X{Y}′′}{t-Δt}′′ = 1/(T•{X,X{Y}, X′,X{Y}, X′′,X{Y}′′}{Δt} - T•{X,X{Y}, X′,X{Y}, X′′,X{Y}′′}{t-Δt}^2)^2

萃取離散時間數據裡的不變特徵
1/((1/T)^2)•{X,X{Y}, X′,X{Y}, X′′,X{Y}′′}{(t-Δt)′′} = (((1/T)^2)•(1/{X,X{Y}, X′,X{Y}, X′′,X{Y}′′}{(Δt)′}) - (1/T)•{X,X{Y}, X′,X{Y}, X′′,X{Y}′′}{t-Δt})^2

正下方新建一個遙桿阻尼濾波，我們可以控制它來改變畫面的渲染結果。

時間增值算法：

我想到了更激進的方法：
導入一份本地視頻{X{t},X{Y}{t}},
{
m^2 - n^2 = z^2 |
(tan m{x}{t}) * (cot n{x}{t}) = x{t},
(tan m{y}{t}) * (cot n{y}{t}) = y{t},

x{t}*y{t} = 色素值
}
z 為景深
{X{t},X{Y}{t},X{Z}{t}}是萃取目標函數。

觀察力算法：

{
m^2 - n^2 = z^2 + t^2|
(tan m{x}{t}) * (cot n{x}{t}) = x{t} = ((tan z{x}{t}) - (cot t{x}{z}))^2
(tan m{y}{t}) * (cot n{y}{t}) = y{t} = ((tan z{y}{t}) - (cot t{y}{z}))^2
x{t}*y{t} = 色素值
}

{
m′^2 - n′^2 = z′^2 + t′^2 |
((tan z{x}{t}′) - (cot t{x}{z}′))^2 = (tan m{x}{t}′) * (cot n{x}{t}′) ∈ (-x{t}′ , x{t}′),
((tan z{y}{t}′) - (cot t{y}{z}′))^2 = (tan m{y}{t}′) * (cot n{y}{t}′) ∈ (-y{t}′ , y{t}′),
x{t}′*y{t}′ = 一階色素值
}

{
m′′^2 - n′′^2 = z′′ ^2 + t′′^2 |
(tan m{x}{t}′′) * (cot n{x}{t}′′) = 1 = (tan m{x}{t}′′) * (cot n{x}{t}′′)
(tan m{y}{t}′′) * (cot n{y}{t}′′) = 1 = ((tan z{y}{t}′′) - (cot t{y}{z}′′))^2
x{t}′′*y{t}′′ = 二階色素值
}

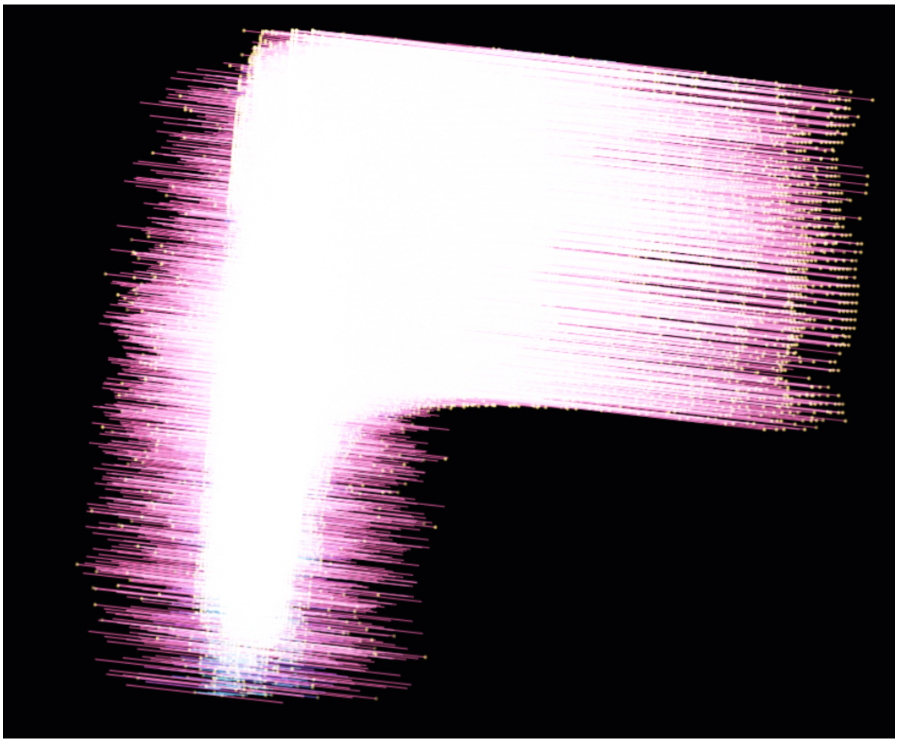
{z,z′,z′′,t,t′,t′′}為景深，z*t 為景深色素。
{X{t},X{Y}{t},X{Z}{t}, X{t}′,X{Y}{t}′,X{Z}{t}′, X{t}′′,X{Y}{t}′′,X{Z}{t}′′}是萃取目標函數。

可以加入暫停鍵，暫停時刻截取渲染結果作為單視圖{X{t}{1},X{Y}{t}{1},X{Z}{t}{1}}
截取兩張
{X{t},X{Y}{t},X{Z}{t}}{1}{3}
後,
{
{X{t},X{Y}{t},X{Z}{t}}{4} = ({X{t},X{Y}{t},X{Z}{t}}{3} - {X{t},X{Y}{t},X{Z}{t}}{1})^2 | (tan {X{t},X{Y}{t},X{Z}{t}}{3}) * (cot {X{t},X{Y}{t},X{Z}{t}}{1}) = {X{t},X{Y}{t},X{Z}{t}}{4}
}
在正下方新建窗口 3D 渲染
{

{M,M{N},M{Z}} = ({X{t},X{Y}{t},X{Z}{t}}{4} - 1/{X{t},X{Y}{t},X{Z}{t}}{4})^2 |
(tan m{x}{t}{4}) * (cot n{x}{t}{4}) = x{t}{4}
(tan m{y}{t}{4}) * (cot n{y}{t}{4}) = y{t}{4}
}

再截取一張{X{t},X{Y}{t},X{Z}{t}}{2}
{
M{Z} |
(tan X{t}{2}) • (cot X{Y}{t}{2}) = X{Z}{t}{2}
(tan m{x}{t}{2}) * (cot n{x}{t}{2}) = x{t}{2}
(tan m{y}{t}{2}) * (cot n{y}{t}{2}) = y{t}{2}
}

不錯。可以把視頻的0～5秒作為相位1，5～10秒作為相位2，10～15秒作為相位3。播放{M,M{N},M{Z}}{t}的渲染結果。
渲染時的點雲不必暫時性， 而是永久保留點雲渲染結果， 連線繼續暫時性， 加一個暫停渲染按鈕。
然後{M,M{N},M{Z}}{t}代換{X{t},X{Y}{t}, X{t}′,X{Y}{t}′, X{t}′′,X{Y}{t}′′}萃取所有離散時間數據、萃取離散時間數據裡的不變特徵。
並且複製一份新的渲染矩陣， 它的Z軸高度為原始渲染矩陣的負數， 並且加上原始渲染矩陣峰值序列裡中值峰值的兩倍2A*， 或1.9A*， 組成包絡列印。
然後用原始渲染序列的渲染點雲連線反轉渲染序列的渲染點雲。
舊連線取消。
然後兩個渲染矩陣都通波濾除小於A*/2的峰值序列不再渲染它們。然後去時間化，把時間序列的所有渲染結果都一次性渲染出來。



總結更激進的方法：

對於時間序列{X{t},X{Y}{t}}
(tan m{x}{t}) * (cot n{x}{t}) = x{t},
(tan m{y}{t}) * (cot n{y}{t}) = y{t},

x{t} * y{t} 表達序列特徵
m^2 - n^2 = z^2 + t^2 表達序列時間深度

(tan m{x}{t}) * (cot n{x}{t}) = x{t},
(tan m{y}{t}) * (cot n{y}{t}) = y{t},
表達的是對於時間序列，總存在 tan * cot 構成，
也就是說，tan * cot 的一階項，是 tan 與 cot 相乘求倒數，當 tan 與 cot 相等時，是平方求倒數。
tan * cot 的二階項，當 tan 與 cot 相等時，是平方求倒數的平方求倒數，交叉地分佈平方與倒數。

所以，沒有必要使用 tan、cot，改成 k{t}, 1/k{t} 也可以成爲切交線濾波。

微積 k{t}, 1/k{t}
...,k{t}^4,k{t}^3,k{t}^2,k{t},1, 1/k{t}, 1/k{t}^2, 1/k{t}^3, 1/k{t}^4,...
加入一個微擾 ε
...,εk{t}^4,εk{t}^3,εk{t}^2,εk{t},ε^2, ε/k{t}, ε/k{t}^2, ε/k{t}^3, ε/k{t}^4,...
再把 ε 也變成時間序列
...,ε{t}^4,ε{t}^3,ε{t}^2,ε{t},1, 1/ε{t}, 1/ε{t}^2, 1/ε{t}^3, 1/ε{t}^4,...
加入一個微擾 k
...,kε{t}^4,kε{t}^3,kε{t}^2,kε{t},k^2, k/ε{t}, k/ε{t}^2, k/ε{t}^3, k/ε{t}^4,...
相干與退相干完全由序列{t}的約束條件決定。
牛頓的幾何學有一種特徵，他能夠用規則與不規則的組合繪製出變換路徑。
我也有自己的幾何學特徵，對於線性，它可以用一群方形互相連結邊長來演繹，而不在乎這些方形的尺寸，只需要邊長與邊長貼合。需要確定變化率時再確定尺寸。
對於非線性，它可以用一群平行四邊形互相連結邊長來演繹，同樣地不在乎這些平行四邊形的尺寸。需要確定變化率時再確定尺寸。
幾乎由濾波來決定幾何尺寸。

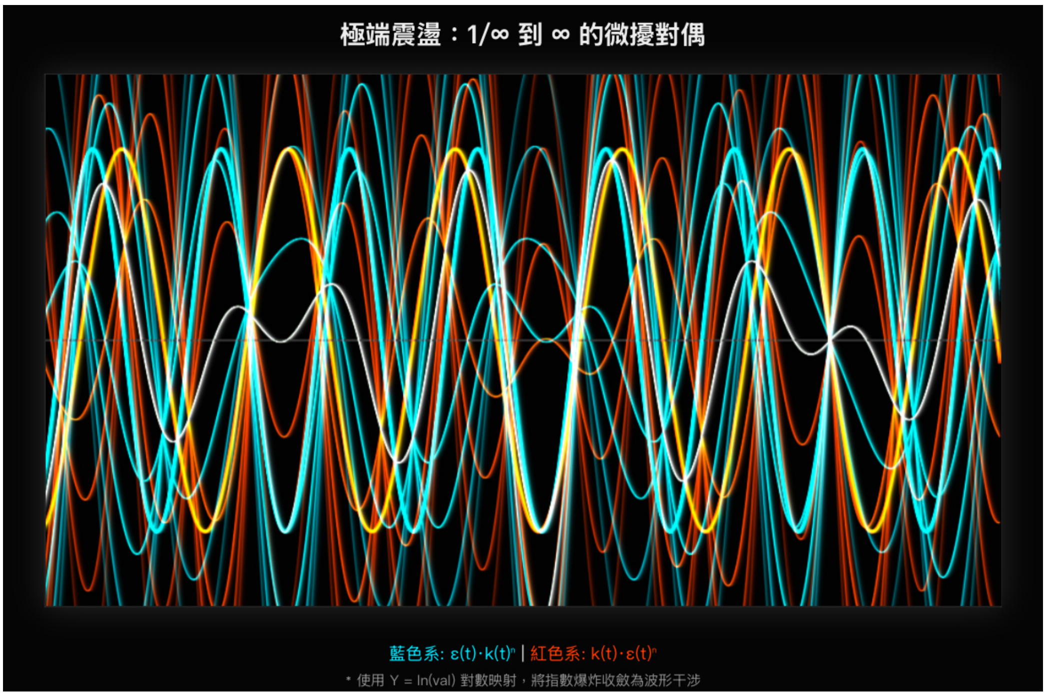
線的形狀可以由面來決定，面的形狀可以由體來決定，體的形狀可以由觀測路徑線來決定，無窮無盡...觀測槓桿的疊加寫成濾波。所以當你在求階時，你應該把常數項分解為這種微擾正交並且求倒數的形式，如果你用泰勒展開，會面臨非線性發散的困境，而被迫增加計算量。微擾正交並且求倒數的微擾正交並且求倒數嵌套。既然如此，我們已經知道計算量會被迫增加，那就主動先把常數分解進映射遞歸，按需增加，讓常數直接變成一套神經網路，按需計算。
直接用手機級的算力作為總部計算量，然後把總部計算量也做微擾正交並且求倒數的嵌套形式。主動加一個限制器。層級如果需要很少那麼先先配置算力^n 給現有層級，配置算力 1/^n 給嵌套擴散。層級如果很多那麼就先配置算力^n 給嵌套擴散，配置算力 1/^n 給現有層級。
如果 t 閾值內迭代完畢，那麼就不必要切換配置策略，如果 t 閾值內沒有迭代完畢，那麼就切換配置策略。

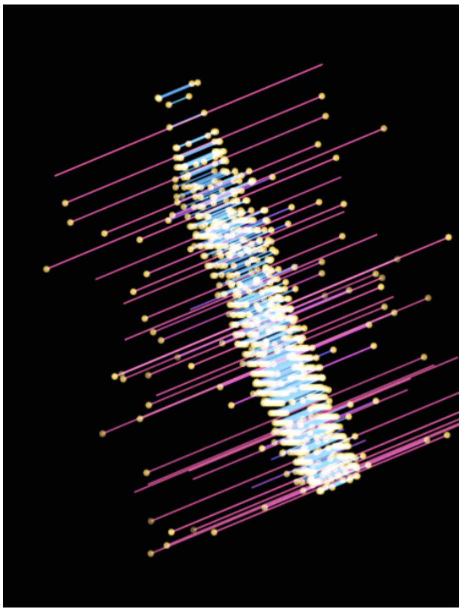
可以設計一條貪吃蛇訓練。

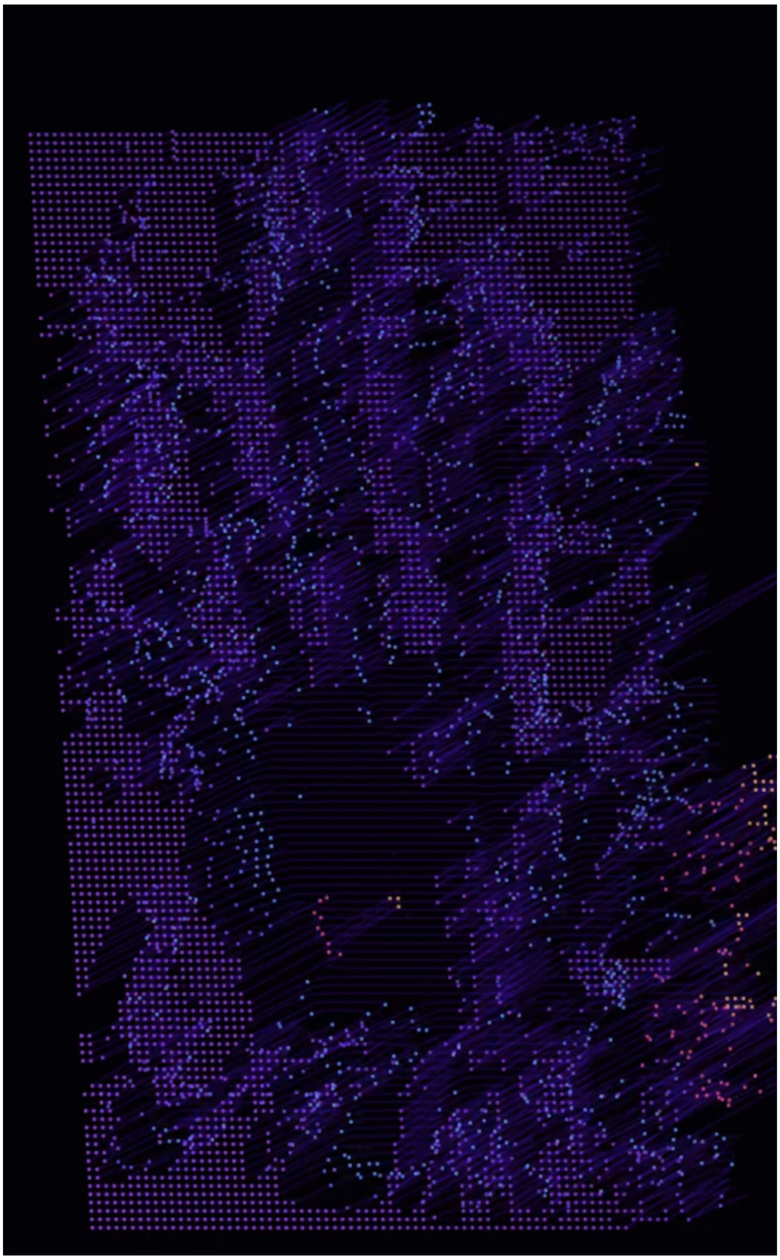
{ε+Δx′′, ε+Δx′, ε+Δx,ε+Δy, ε+Δy′, ε+Δy′′}
是蛇的位置變化。
{εk{t}{x}^3,εk{t}{x}^2,εk{t}{x},ε/k{t}{x}, ε/k{t}{x}^2, ε/k{t}{x}^3},
{εk{t}{y}^3,εk{t}{y}^2,εk{t}{y},ε/k{t}{y}, ε/k{t}{y}^2, ε/k{t}{y}^3},
是對應分解映射。
擴散上限為 Δx^n 與 Δy^n 對應^n 與 1/^n,
隨機設定一個時間與食物， 貪吃蛇需要在時間內吃完所有食物， 貪吃蛇的前進方向也是隨機的， 此時嵌套不擴散， 高階變化截斷， 否則嵌套擴散， 高階變化不截斷。
HTML canvas code.

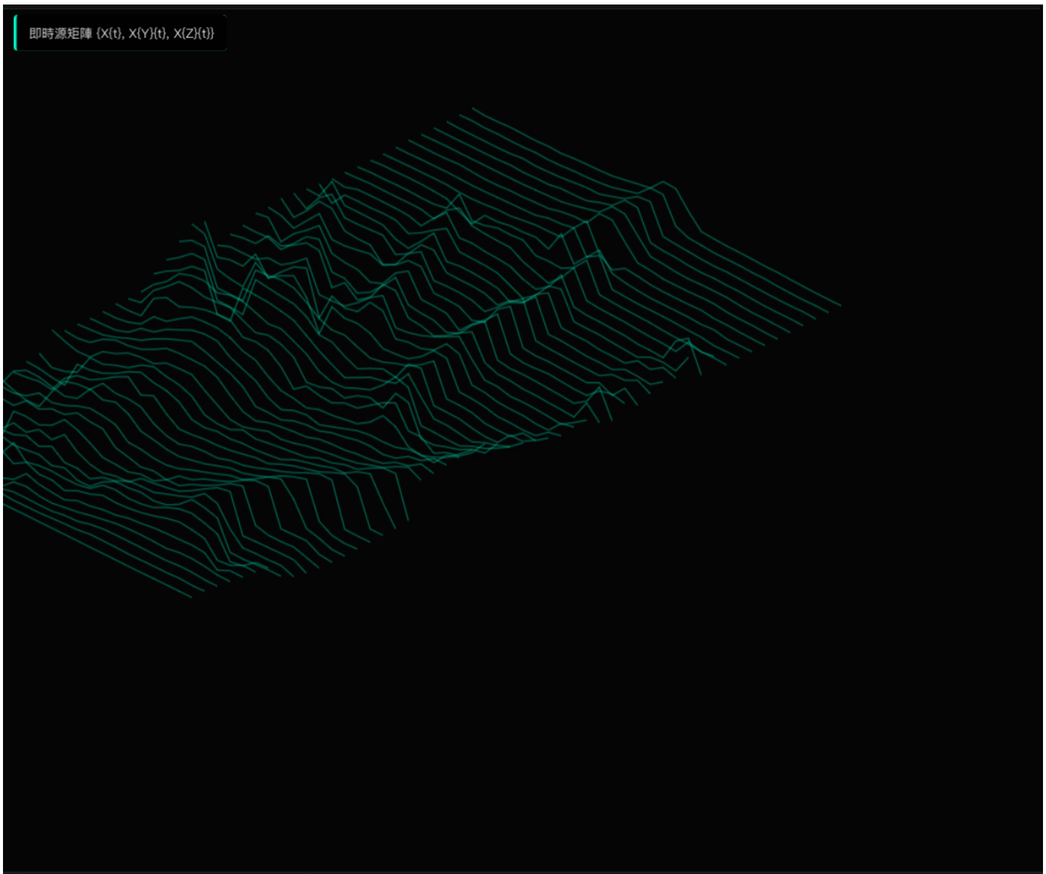
嵌套擴散時， 蛇會隨著層級增加而分裂成多條隨機方向的蛇， 它們會存活， 並且每一條都會繼續分裂， 直至其中有一條蛇吃到食物， 其他蛇就滅絕。並且分裂的次數越多， 微擾 ε 的取值越小。

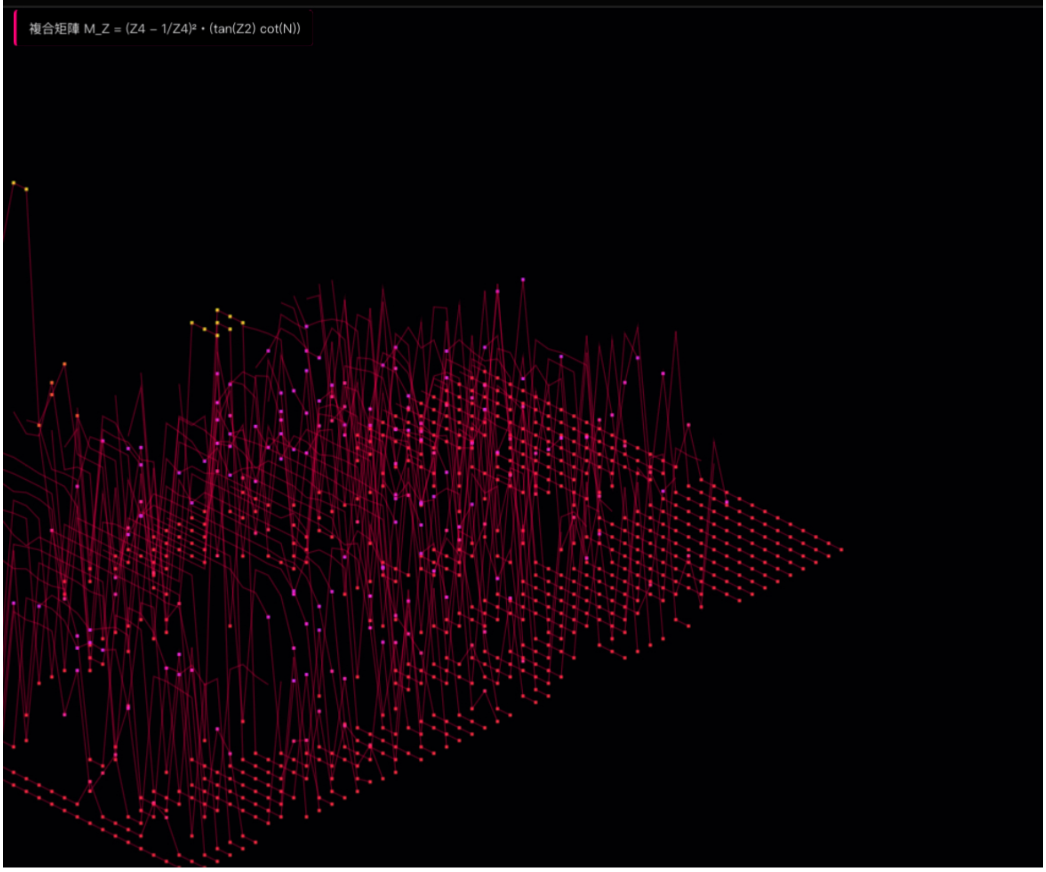
記憶力算法：
Δx{t}^n 與 Δy{t}^n 時間序列離散數據需要被記錄下來， 如果某一個時間序列正好是吃到食物的時間序列， 那麼這個時間對應的位置變化數據會獲得加分， 加分多的位置變化就會成為蛇在嵌套擴散分裂時的初始預設位置變化取值。
如果初始預設位置變化取值沒有馬上吃到食物， 它會不斷減分。 記憶衰退， 衰退掉無用的記憶。 當它變成負分時， 初始預設位置變化取值開始與蛇的前進方向相干。 記憶自相似搜索。 吃到食物後， 得分再次回升， 並且獲得額外加分， 沒吃到， 則繼續減分。 分數越高， 方向與位置變化越退相干， 分數越低， 方向與位置變化越相干。








即時源矩陣 {X{t}, X{Y}{t}, X{Z}{t}}


複合矩陣 M_Z = $(Z4 – 1/Z4)^2$ • (tan(Z2) cot(N))