

# Wavelet Transform for Image Processing

Zhang Jinrui\*  
jerryzhang40@gmail.com

20250728

## Abstract

In this article, I tried to use Wavelet Transform to compress images.

## 1 Decomposition & Mother Wavelet

A mother wavelet  $\psi$  satisfied following property.  $\langle \psi, \psi \rangle = 1$   
And the family of the Hilbert Basis wavelets are as follow.

$$\psi_{j,k} = \sqrt{2^j} \psi(2^j(x - 2^{-j}k)) \quad j, k \in \mathbb{Z}$$

A function  $f$  can be decomposed use these basis as follow.

$$f = c_{j,k} \psi_{j,k}$$

There are 10 drones and fly on the sky obeys Newton's second law of motion.  
which is

$$\begin{aligned} \vec{F} &= m\vec{a} \\ \vec{a} &= \frac{d\vec{v}}{dt} = \frac{d^2\vec{x}}{dt^2} \end{aligned}$$

And I mean the policy by, we need a function of force depending on some communication between drones to decide the  $\vec{F}$

$$\vec{F} = f(\text{thecurrentinformation})$$

And then we want the following dynamic system

$$\begin{bmatrix} \frac{d\vec{d}}{dt} \\ \frac{d\vec{v}}{dt} \end{bmatrix} = \begin{bmatrix} \vec{v} \\ \vec{a} = f/m \end{bmatrix}$$

has some Self-organized emergent phenomena, to automatically emergent a circle rounding pattern.

question: framelet multiresolution(sacle function and funciont)

---

\*alternative email:zhangjr1022@mails.jlu.edu.cn

## 2 Fractal Approach

For a mother wavelet  $\psi(x)$  satisfied following property.

$$\langle \psi, \psi \rangle = 1$$

$$\int_{-\infty}^{\infty} \psi(x) dx = 0$$

And the family

$$\psi_{j,k} = \sqrt{2^j} \psi(2^j(x - 2^{-j}k)) \quad j, k \in \mathbb{Z}$$

For any function satisfied the following property

$$\int_{-\infty}^{\infty} f(x) dx = 0$$

We can decomposed it in to the form

$$f = \sum_{j,k} c_{j,k} \psi_{j,k}$$

And the coefficients can obtained by the following inner product.

$$\langle f, \psi_{j,k} \rangle = c_{j,k}$$

For a function restricted on  $[0, 1]$  we can only use

$$\psi_{j,k} = \sqrt{2^j} \psi(2^j(x - 2^{-j}k)) \quad k < 2^j, k \in \mathbb{N}$$

To decomposed.

And even more for a discrete function restricted on  $[0, 1]$ , with  $2^n$  sample points we can only use

$$\psi_{j,k} = \sqrt{2^j} \psi(2^j(x - 2^{-j}k)) \quad k < 2^j \leq 2^n, k \in \mathbb{N}$$

To decomposed.

The inverse process to generate all  $\psi_{j,k}$  for the discrete case I want to propose a different approach.

Say I can have a function

$$F : \mathbb{R}^2 \rightarrow \mathbb{R}^2$$

and a base vector

$$[v_0, v_1]$$

then the next basis would obtain by the following

$$F([v_0, v_1]) = [w_0, w_2]$$

the next basis is

$$[w_0, w_1, w_2, w_3]$$

To keep every iteration to be orthogonal We need to restrict  $F$  as following.

$$w_1 = -\frac{v_0 w_0}{v_1}$$

$$w_3 = -\frac{v_0 w_2}{v_1}$$

Then every basis need to be nomalized to fit the property

$$\langle \psi, \psi \rangle = 1$$

Then using

$$[v_0, v_1]$$

and  $F$  we can generate all the basis.

Do the nomal decomposition we will get coefficients  $c_{j,k}$  We know  $c_{j,k}c_{j,k} = C$  If we want all the information constrain in one coefficients we need to make  $c_{j,k}c_{j,k}c_{j,k}c_{j,k}$  as large as possible.

This is a parameter optimization problem

$$\max_{F, [v_0, v_1]} c_{j,k}c_{j,k}c_{j,k}c_{j,k}$$

Then we can just transfer  $F$  and  $[v_0, v_1]$

## 2.1 a simple prompt

To be more clear of the notations we use, we have  $i \in \{1, 2, \dots, 10\} = N$  And the drones are ignored of its flying height, which the position vector can be a 2d vector note it as  $\vec{d}_i$  And so the velocity and acceleration we denote as  $\vec{v}_i = \frac{d\vec{d}_i}{dt}$  and  $\vec{a}_i = \frac{d\vec{v}_i}{dt}$  I want to prompt a  $f$  so that it can form a cirle.

$$\vec{f}_i = m_i \left( \sum_{\forall k \neq i, \|\vec{d}_i - \vec{d}_k\| \leq R} \left( \frac{\vec{d}_i - \vec{d}_k}{\|\vec{d}_i - \vec{d}_k\|^3} \right) + \left( \frac{d_{t(i)} - \vec{d}_i}{\|d_{t(i)} - \vec{d}_i\|} - v_i \right) \right)$$

This model is easy to explain, the first term is just a inverse square propell force, the second term is make the velocity quickly approach a set direction the  $t(i)$  is just a randomly choosed target drone other than  $i$  that is  $t(i) \in N, t(i) \neq i$

This formula can be rewrite without physical term as follow.

$$\vec{a}_i = \sum_{\forall k \neq i, \|\vec{d}_i - \vec{d}_k\| \leq R} \left( \frac{\vec{d}_i - \vec{d}_k}{\|\vec{d}_i - \vec{d}_k\|^3} \right) + \left( \frac{d_{t(i)} - \vec{d}_i}{\|d_{t(i)} - \vec{d}_i\|} - v_i \right)$$

separately view this is combined by two independent force

$$(\vec{a}_i)_{target} = \left( \frac{d_{t(i)} - \vec{d}_i}{\|d_{t(i)} - \vec{d}_i\|} - v_i \right)$$

$$(\vec{a}_i)_{propell} = \sum_{\forall k \neq i, \|\vec{d}_i - \vec{d}_k\| \leq R} \left( \frac{\vec{d}_i - \vec{d}_k}{\|\vec{d}_i - \vec{d}_k\|^3} \right)$$

## 2.2 a simple prompt:simulation

### 2.2.1 Four drone case:derivation

This case just choose  $N = \{1, 2, 3, 4\}$  and don't allow  $t(t(i)) = i$  which definitely form a three element loop and a dangling drone.

We have a  $(4,2)$ -tensor  $\vec{d}_i$  and two other  $(4,2)$ -tensor  $\vec{v}_i$  and  $\vec{a}_i$ . The initial points are randomly choosed in Uniformly  $[0, 1] \times [0, 1]$

choose a time increment  $dt$  and the simulation update formula is simple to write

just as follow

$$\begin{bmatrix} \vec{d}_{n+1_i} \\ \vec{v}_{n+1_i} \end{bmatrix} = \begin{bmatrix} \vec{d}_{n_i} + \vec{v}_{n_i} dt \\ \vec{v}_{n_i} + (\sum_{\forall k \neq i, \|\vec{d}_{n_i} - \vec{d}_{n_k}\| \leq R} (\frac{\vec{d}_{n_i} - \vec{d}_{n_k}}{\|\vec{d}_{n_i} - \vec{d}_{n_k}\|^3}) + (\frac{\vec{d}_{n_{t(i)}} - \vec{d}_{n_i}}{\|\vec{d}_{n_{t(i)}} - \vec{d}_{n_i}\|} - \vec{v}_{n_i})) dt \end{bmatrix}$$

simple Euler method.

### 2.2.2 Four drone case:code & result

the computational code are `[1, FourDroneCase]`. the results are shown by the following pictures which generated by the code. The video are `[2, sample1-video]` and `[3, sample2-video]` and more other in the same folder on github.

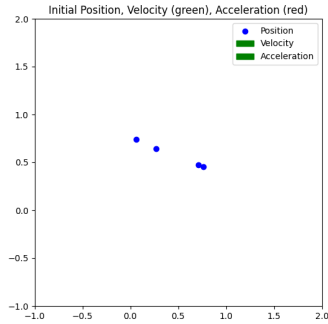


Figure 1: sample1 randomly initial position

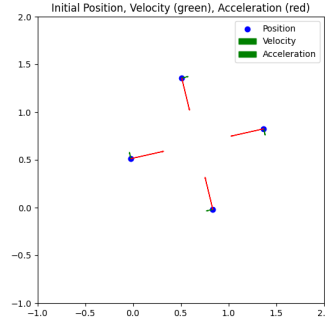


Figure 2: sample1 after a period of time

### 2.2.3 Ten drone case:derivation

undergoing

### 2.2.4 Ten drone case:result

undergoing

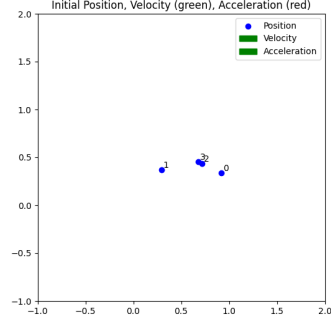


Figure 3: sample1 randomly initial position

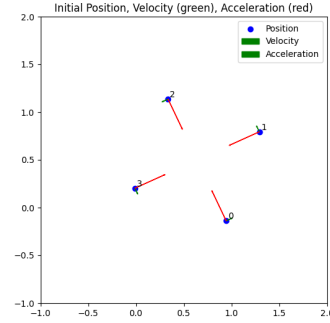


Figure 4: sample2 after a period of time

## 2.3 some analysis why it will have a stability property

### 2.3.1 the terminate radius R

undergoing

### 2.3.2 the terminate center O

undergoing

### 2.3.3 graph theory part

The  $t(i)$  forms a graph which have  $n$  points and  $n$  oriented edges, this forms a tree with a extra edges, and this case It will obviously form a Unicyclic Graph.

Which is a tree if we treat all the point on the loop as the same point.

## 2.4 target distance method

undergoing

## 2.5 target distance method:simulation

undergoing

# 3 Zinan Su's approach

## 3.1 notations & equations

safe collide radius is  $d_s$

$$\sigma = 2d_s$$

$NUM$  is the total number of the drones. And then we want the following dynamic system

$$\begin{bmatrix} \frac{dp_i}{dt} \\ \frac{dv_i}{dt} \end{bmatrix} = \begin{bmatrix} \vec{v}_i \\ \vec{a}_i \end{bmatrix}$$

circle origin is a function

$$c = \frac{1}{NUM} \sum_{k=1}^{NUM} p_k$$

Four constants.

$$k_p =$$

$$k_d =$$

$$k_v =$$

$$k_r =$$

$$R^* = \frac{1}{NUM} \sum_{k=1}^{NUM} p_k(0) - c(0)$$

$$v_d = \frac{1}{NUM} \sum_{k=1}^{NUM} \|v_k(0)\|$$

and

$$r_i = p_i - c$$

$$d_i = \|r_i\|$$

$$\hat{r}_i = \frac{r_i}{d_i}$$

$$\hat{\theta}_i = \mathbb{M}_\theta r_i$$

$$\mathbb{M}_\theta = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

$$v_{i\parallel} = \hat{r}_i \cdot v_i$$

$$v_{i\perp} = \hat{\theta}_i \cdot v_i$$

$$U(r) = k_r e^{-\frac{r}{2\sigma^2}}$$

$$U_{ij} = U(\|p_i - p_j\|)$$

$$\vec{u}_{i1} = [-k_p(d_i - R^*) - k_d v_{i\parallel}] \hat{r}_i$$

$$\vec{u}_{i2} = [-k_v(v_{i\perp} - v_d)] \hat{\theta}_i$$

$$\vec{u}_{i3} = \sum_{\forall k \neq i} (-\nabla_{p_i} U_{ij})$$

$$\vec{u}_i = \vec{u}_{i1} + \vec{u}_{i2} + \vec{u}_{i3}$$

## References

- [1] Zhang Jinrui. Fourdronecase.py. [https://github.com/jerzha40/2025\\_exchange\\_at\\_universityofalberta/blob/main/DroneInCircleEXP/FourDroneCase.py](https://github.com/jerzha40/2025_exchange_at_universityofalberta/blob/main/DroneInCircleEXP/FourDroneCase.py). Accessed: 2025-07-20.
- [2] Zhang Jinrui. sample1-video. [https://github.com/jerzha40/2025\\_exchange\\_at\\_universityofalberta/blob/main/DroneInCircle/fig/sample1/0001-18255.mp4](https://github.com/jerzha40/2025_exchange_at_universityofalberta/blob/main/DroneInCircle/fig/sample1/0001-18255.mp4). Accessed: 2025-07-20.
- [3] Zhang Jinrui. sample2-video. [https://github.com/jerzha40/2025\\_exchange\\_at\\_universityofalberta/blob/main/DroneInCircle/fig/sample2/0001-18994.mp4](https://github.com/jerzha40/2025_exchange_at_universityofalberta/blob/main/DroneInCircle/fig/sample2/0001-18994.mp4). Accessed: 2025-07-20.