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 ψ satisfied following property. $\langle \psi, \psi \rangle = 1$

And the family of the Hilbert Basis wavelets are as follow.

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

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

$$\vec{F} = m\vec{a}$$

$$\vec{a} = \frac{d\vec{v}}{dt} = \frac{d^2\vec{x}}{dt^2}$$

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

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

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 Haar 1D sequence Decomposition

For a 2^N points sequence, the mother wavelet is

$$\psi(n) = \begin{cases} 0, & \text{if } n < 0 \\ \frac{1}{\sqrt{2^N}}, & \text{if } 0 \leq n < 2^{N-1} \\ \frac{-1}{\sqrt{2^N}}, & \text{if } 2^{N-1} \leq n < 2^N \\ 0, & \text{if } n >= 2^N \end{cases}$$

We have

$$\langle \psi, \psi \rangle = \sum_{k=0}^{2^N-1} \frac{1}{2^N} = 1$$

For other derived basis we use this discretized formula

$$\psi_{j,k}(n) = \sqrt{2^j}\psi(2^j(n - 2^{N-j}k)), 0 \leq j \leq N-1, k <= 2^j - 1$$

$$\begin{bmatrix} \psi_{0,0} & 0 & \cdots & 0 \\ \psi_{1,0} & \psi_{1,1} & \cdots & 0 \\ \psi_{2,0} & \cdots & \psi_{2,3} & 0 \\ \vdots & \vdots & \ddots & 0 \\ \psi_{N-1,0} & \psi_{N-1,1} & \cdots & \psi_{N-1,2^{N-1}-1} \end{bmatrix}$$

And a average basis as following

$$\phi(n) = \begin{cases} 0, & \text{if } n < 0 \\ \frac{1}{\sqrt{2^N}}, & \text{if } 0 \leq n < 2^{N-1} \\ \frac{1}{\sqrt{2^N}}, & \text{if } 2^{N-1} \leq n < 2^N \\ 0, & \text{if } n >= 2^N \end{cases}$$

there are

$$1 + \sum_{k=0}^{N-1} 2^k = 2^N$$

basis. which will be 2^N coefficients then every 2^N sequence $f(n)$ can be written as

$$f(n) = a\phi(n) + \sum_{j=0}^{N-1} \sum_{k=0}^{2^j-1} c_{j,k}\psi_{j,k}$$

Then each coefficients can be obtain by inner product.

$$a = \langle f(n), \phi(n) \rangle$$

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

The inner product of two sequence are define as follow.

$$\langle f(n), g(n)\rangle = \sum_{k=0}^{2^N-1} f(k)g(k)$$

the transformed sequence $\hat{f}(n)$ satisfies

$$c_{j,k} = \hat{f}(2^j + k)$$

$$a = \hat{f}(0)$$

3 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

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

For any function satisfied the following property

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

We can decomposed it in to the form

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

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

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

To decomposed.

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

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

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

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

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 $[v_0, v_1]$

3.1 a simple prompt

To be more clear of the notations we use, we have $i \in \{1, 2, ..., 10\} = N$ 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.

$$\vec{f_i} = m_i\Big(\sum_{\forall k \neq i, \|\vec{d_i} - \vec{d_k}\| \leq R} \Big(\frac{\vec{d_i} - \vec{d_k}}{\|\vec{d_i} - \vec{d_k}\|^3}\Big) + \Big(\frac{\vec{d_{t(i)}} - \vec{d_i}}{\|\vec{d_{t(i)}} - \vec{d_i}\|} - v_i\Big)\Big)$$

This model is easy to explain, the first term is just a inverse square propell force, the second term is make the velocity quickly approch 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.

$$\vec{a_i} = \sum_{\forall k \neq i, \|\vec{d_i} - \vec{d_k}\| \leq R} \left(\frac{\vec{d_i} - \vec{d_k}}{\|\vec{d_i} - \vec{d_k}\|^3}\right) + \left(\frac{\vec{d_{t(i)}} - \vec{d_i}}{\|\vec{d_{t(i)}} - \vec{d_i}\|} - v_i\right)$$

separately view this is combined by two independent force

$$(\vec{a_i})_{target} = \left(\frac{\vec{d_{t(i)}} - \vec{d_i}}{\|\vec{d_{t(i)}} - \vec{d_i}\|} - v_i\right)$$

$$(\vec{a_i})_{propell} = \sum_{\forall k \neq i, \|\vec{d_i} - \vec{d_k}\| \leq R} \left(\frac{\vec{d_i} - \vec{d_k}}{\|\vec{d_i} - \vec{d_k}\|^3}\right)$$

3.2 a simple prompt:simulation

3.2.1 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

$$\begin{bmatrix} \vec{d_{n+1}}_i \\ \vec{v_{n+1}}_i \end{bmatrix} = \begin{bmatrix} \vec{d_n}_i + \vec{v_n}_i dt \\ \vec{v_n}_i + (\sum_{\forall k \neq i, \|\vec{d_n}_i - \vec{d_n}_k\| \leq R} (\frac{\vec{d_n}_i - \vec{d_n}_k}{\|\vec{d_n}_i - \vec{d_n}_k\|^3}) + (\frac{\vec{d_n}_{t(i)} - \vec{d_n}_i}{\|\vec{d_n}_{t(i)} - \vec{d_n}_i\|} - v_{n\,i}))dt \end{bmatrix}$$

simple Euler method.

3.2.2 Four drone case:code & result

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

3.2.3 Ten drone case:derivation

undergoing

3.2.4 Ten drone case:result

undergoing

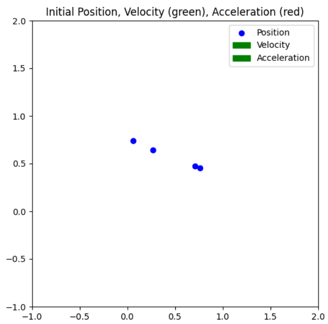

Figure 1: sample1 randomly initial position

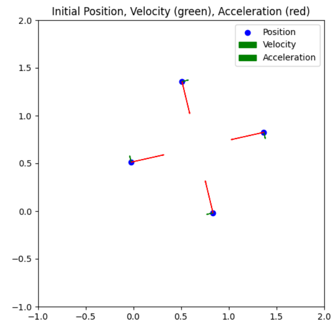

Figure 2: sample1 after a period of time

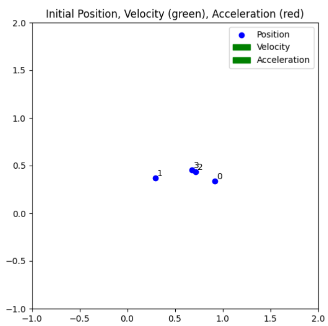

Figure 3: sample1 randomly initial position

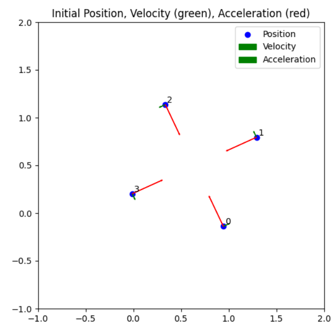

Figure 4: sample2 after a period of time

3.3 some analysis why it will have a stability property

3.3.1 the terminate radius R

undergoing

3.3.2 the terminate center O

undergoing

3.3.3 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.

3.4 target distance method

undergoing

3.5 target distance method:simulation

undergoing

4 Zinan Su's approach

4.1 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

$$\begin{bmatrix} \frac{d\vec{p_i}}{dt} \\ \frac{d\vec{v_i}}{dt} \end{bmatrix} = \begin{bmatrix} \vec{v_i} \\ \vec{a_i} \end{bmatrix}$$

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. 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. 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. Accessed: 2025-07-20.