Final Project

111318093 黃俊傑

111C52005 吳翊睿

111C52016 吳佩霖

2022/12/19

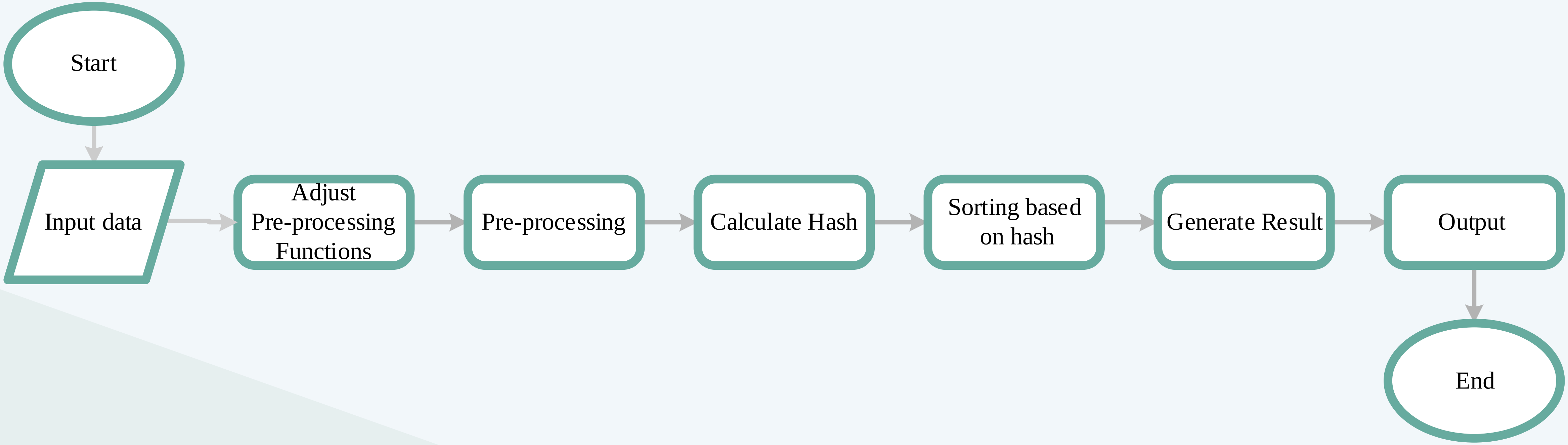
Content

1. Image Pre-processing
2. Hash Function
3. Sorting
4. Parallel Computing
5. Evaluation
6. Results

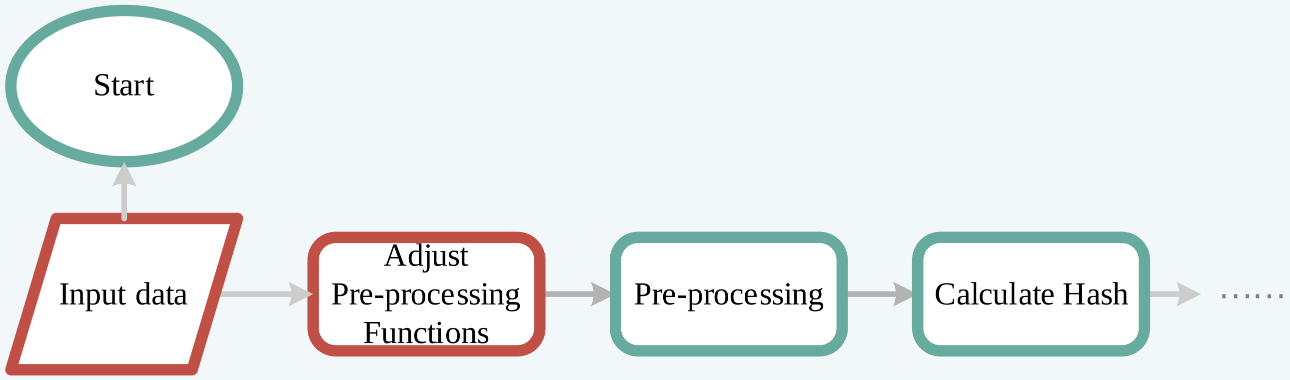
1

Image Pre-processing

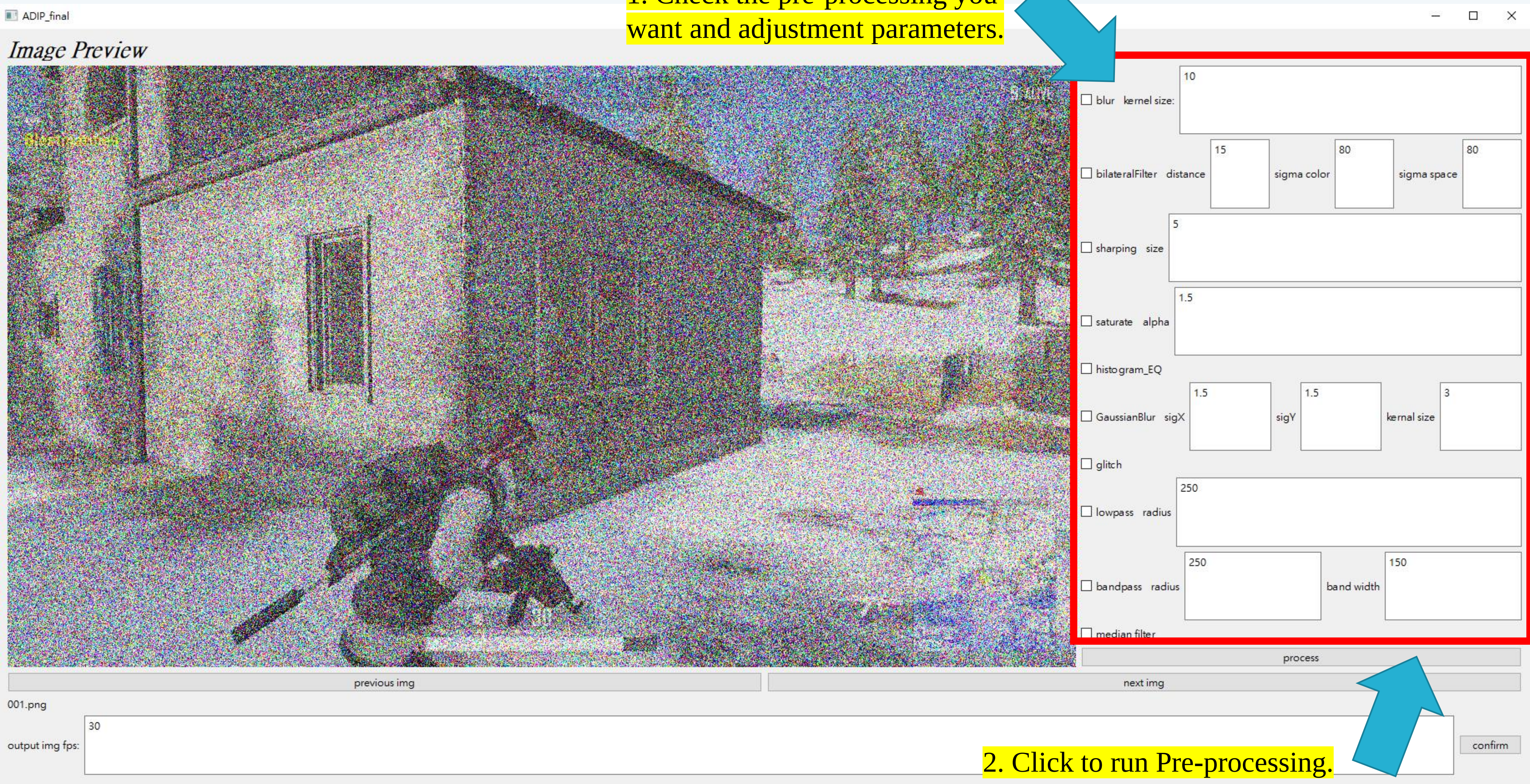
Flow Chart


Start
Input data
Adjust
Pre-processing
Functions
Pre-processing
Calculate Hash
Sorting based
on hash
Generate Result
Output
End

Choose Pre-process



1. Check the pre-processing you want and adjustment parameters.



2. Click to run Pre-processing.

Choose Pre-process

Pre-processing → Calculate Hash → Generate Result → Output

1. Mean Filter
2. Bilateral Filter
3. Sharping
4. Saturate
5. Contrast Limited AHE
6. Gaussian Blur
7. Glitch （Preset）
8. Lowpass Filter
9. Bandpass Filter
10. Median Filter（Preset）

Pre-processing - Preset Process

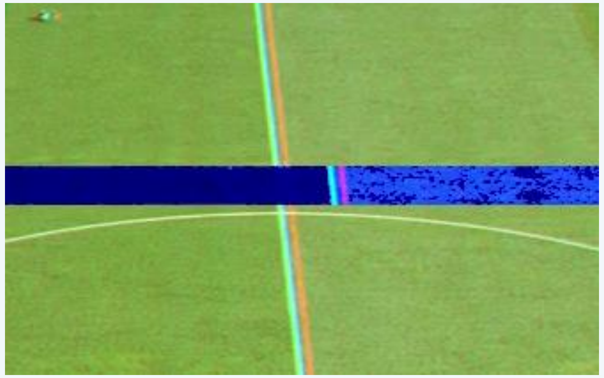
Pre-processing → Calculate Hash → Generate Result → Output

（a）Original image

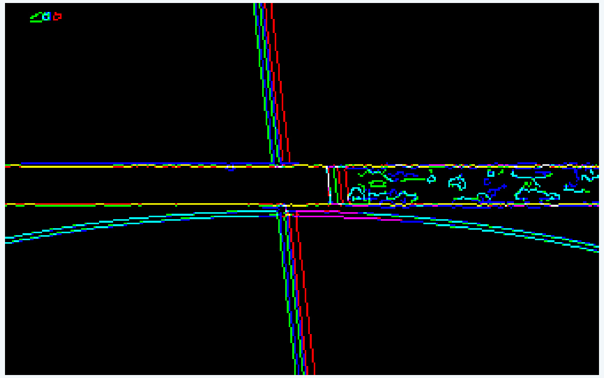
（b）Directly apply median filter

（c）Apply our median filter

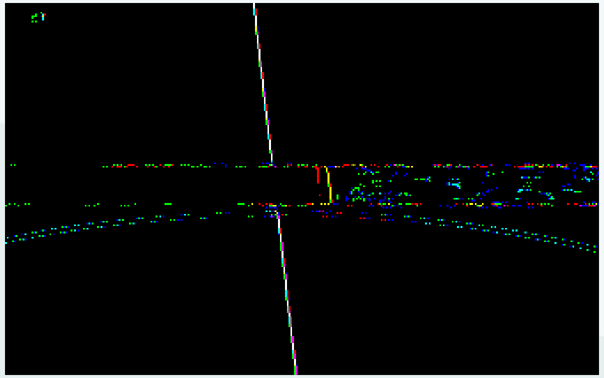
Pre-processing - Glitch (Color Plane Alignment)



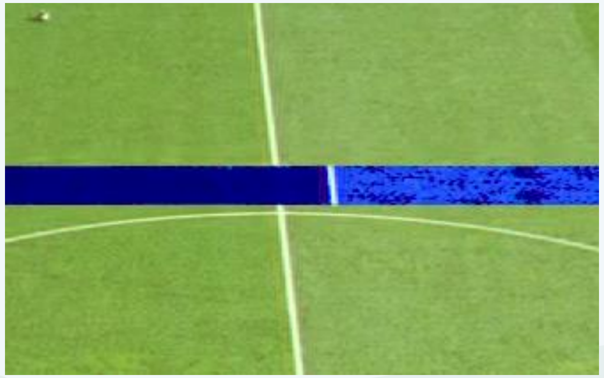
（a）Original image



（b）Edge of each color plate

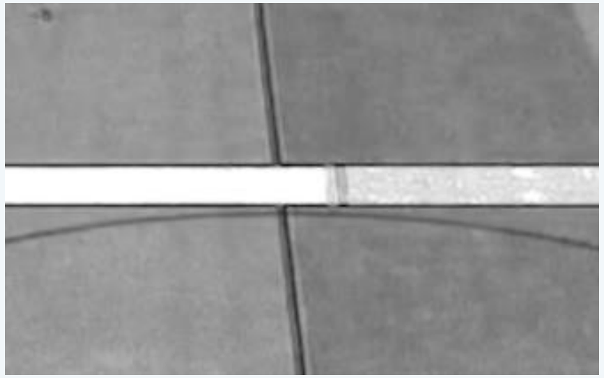


（c）Alignment results for each color plate

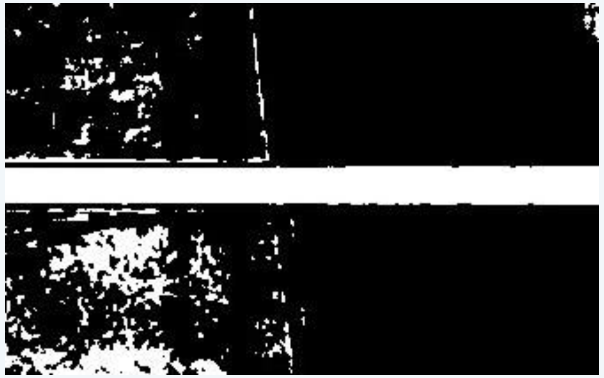


（d）Color plane alignment result

Pre-processing - Glitch

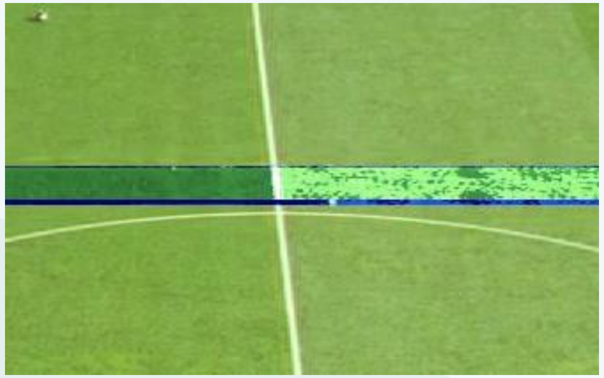


（a） s channel



（b）Binarized image

（c）After morphological processing



（d）Results of Glitch

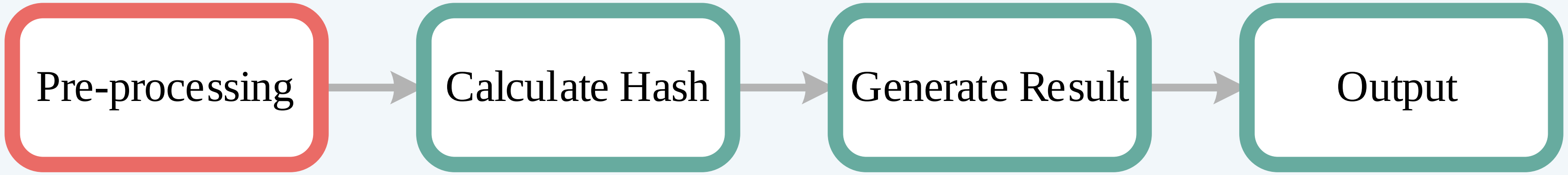
0	0	1	1	1	0	0
0	1	1	1	1	1	0
1	1	1	1	1	1	1
1	1	1	1	1	1	1
1	1	1	1	1	1	1
0	1	1	1	1	1	0
0	0	1	1	1	0	0

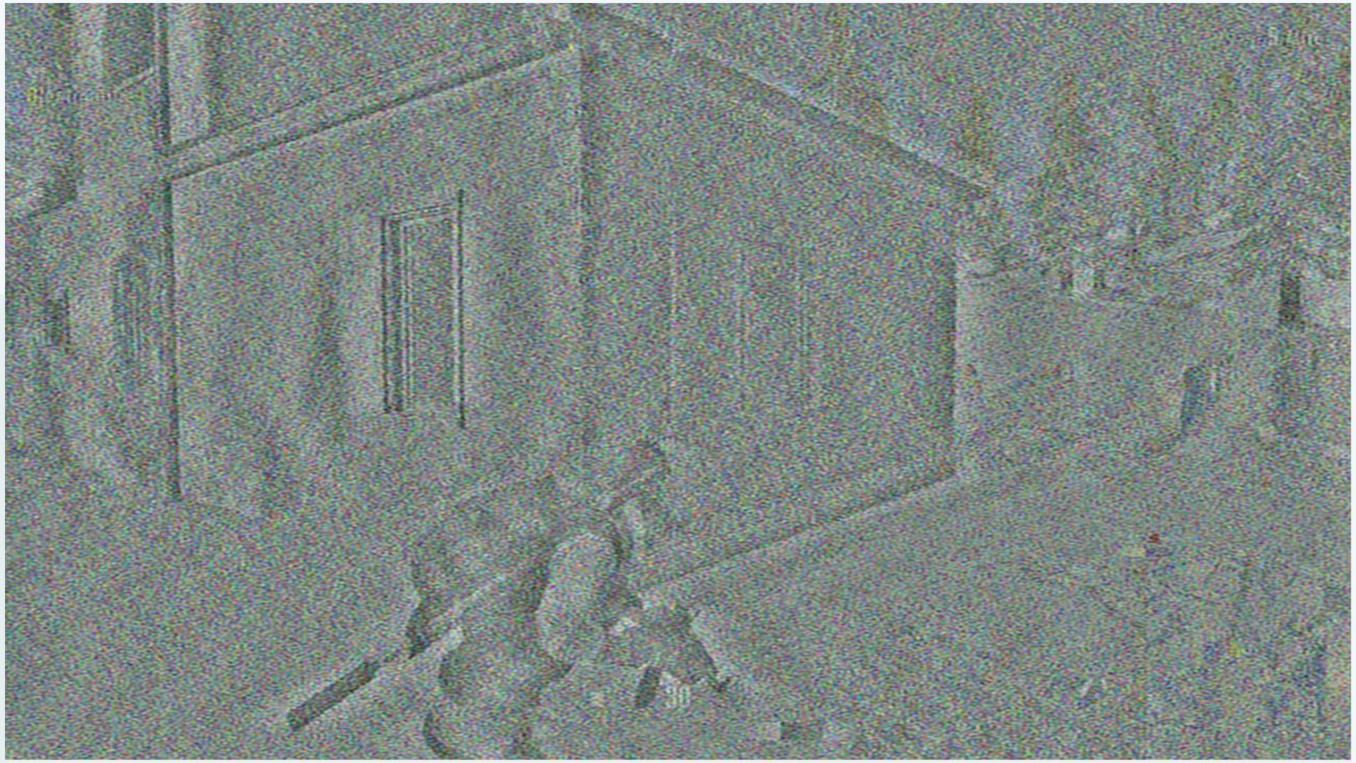
（A）kernel 1

0	1	1	1	0
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
0	1	1	1	0

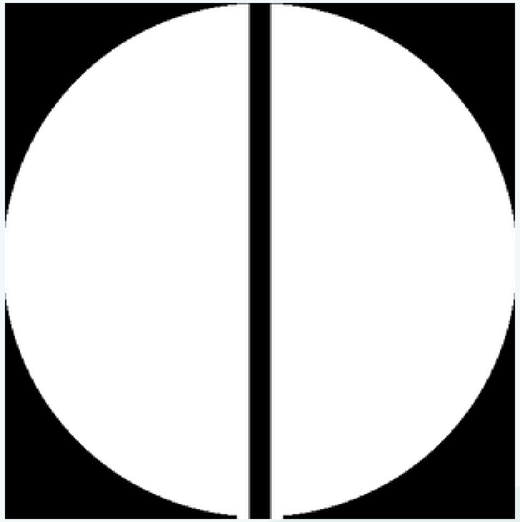
（B）kernel 2

Pre-processing - Lowpass Filter





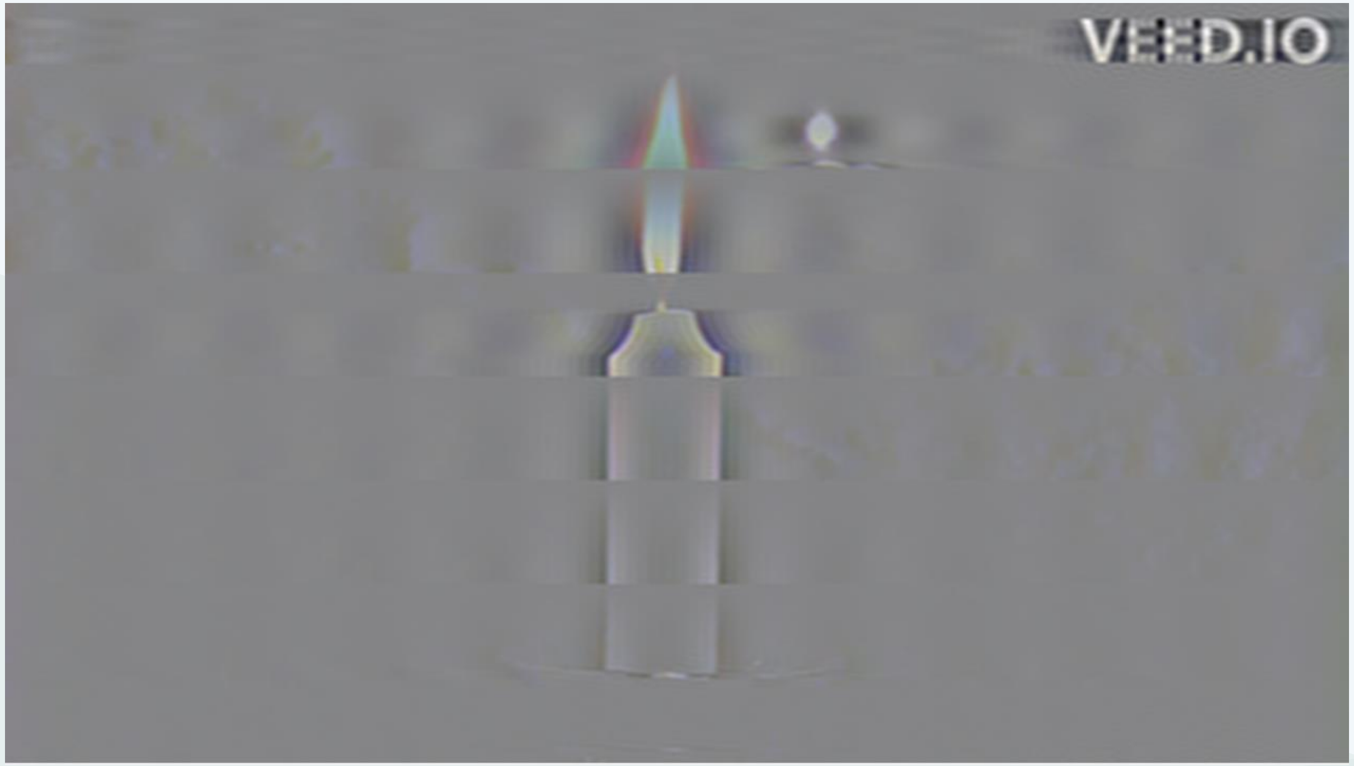
（a）Processed image



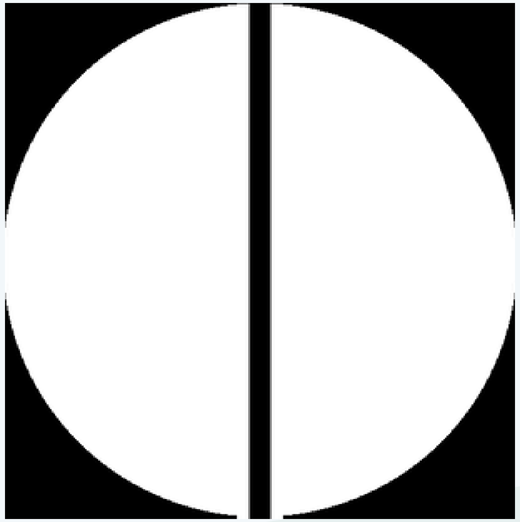
（b）Filter schematic

Pre-processing - Lowpass Filter

Pre-processing → Calculate Hash → Generate Result → Output



（a） Processed image

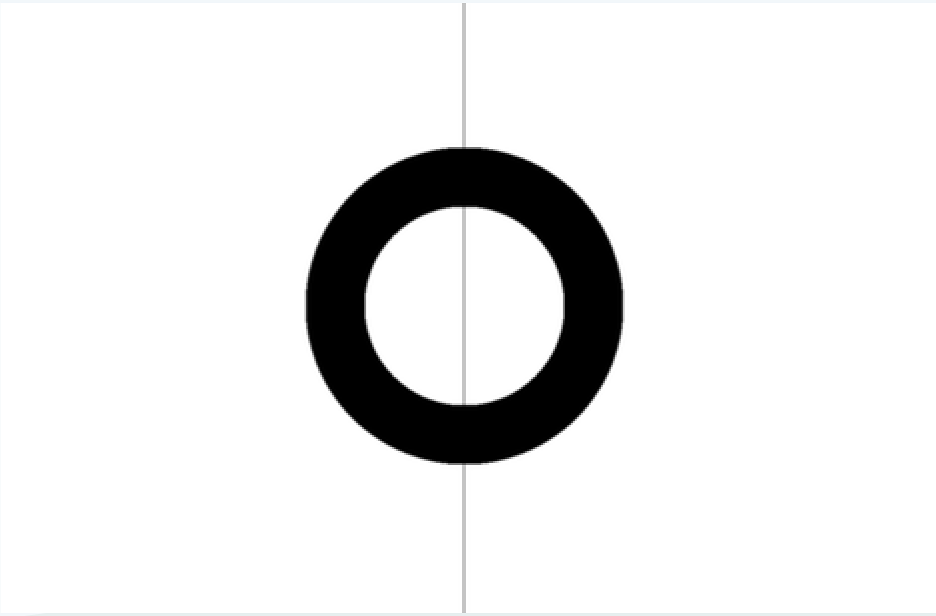


（b） Filter schematic

Pre-processing - Bandpass Filter

Pre-processing → Calculate Hash → Generate Result → Output

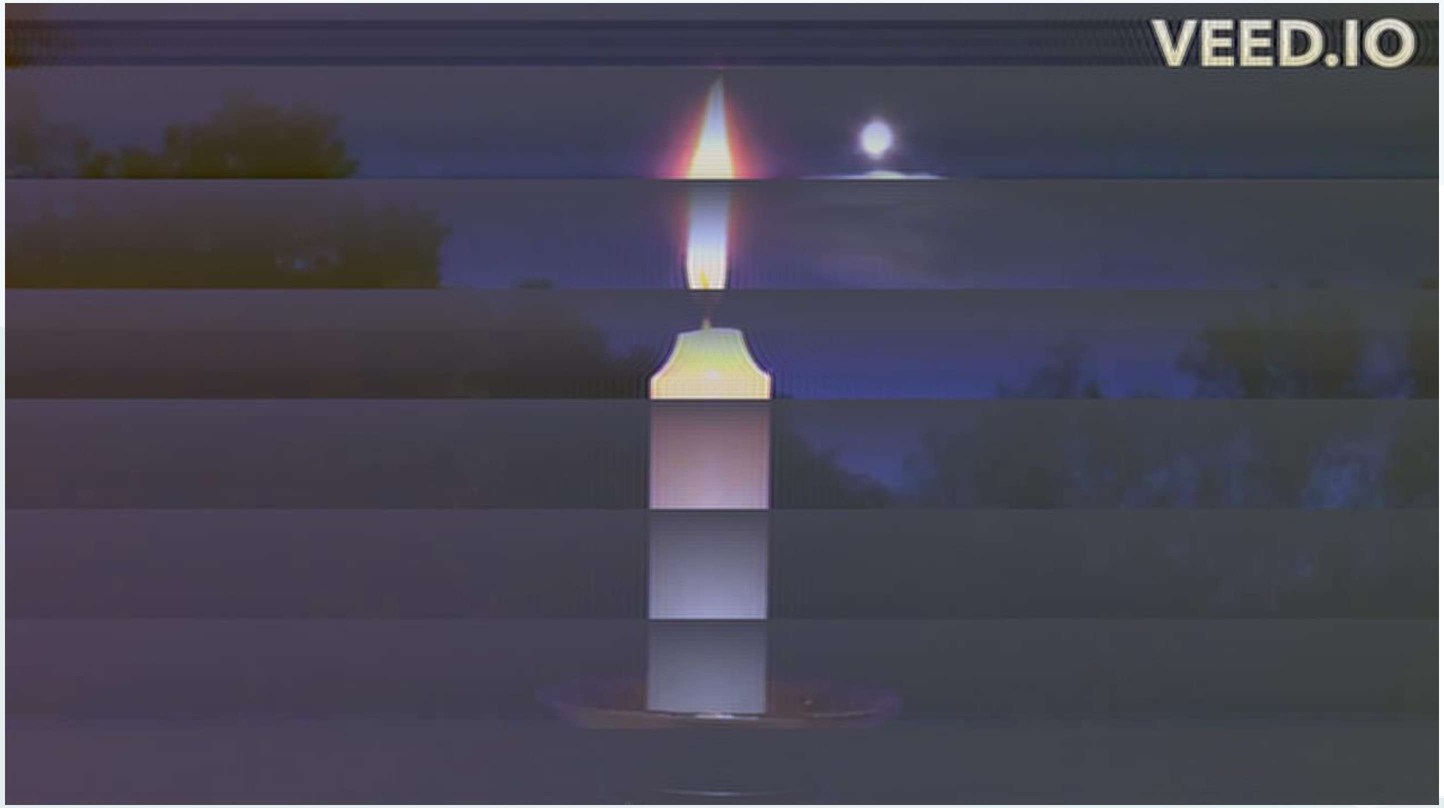
（a） Processed image



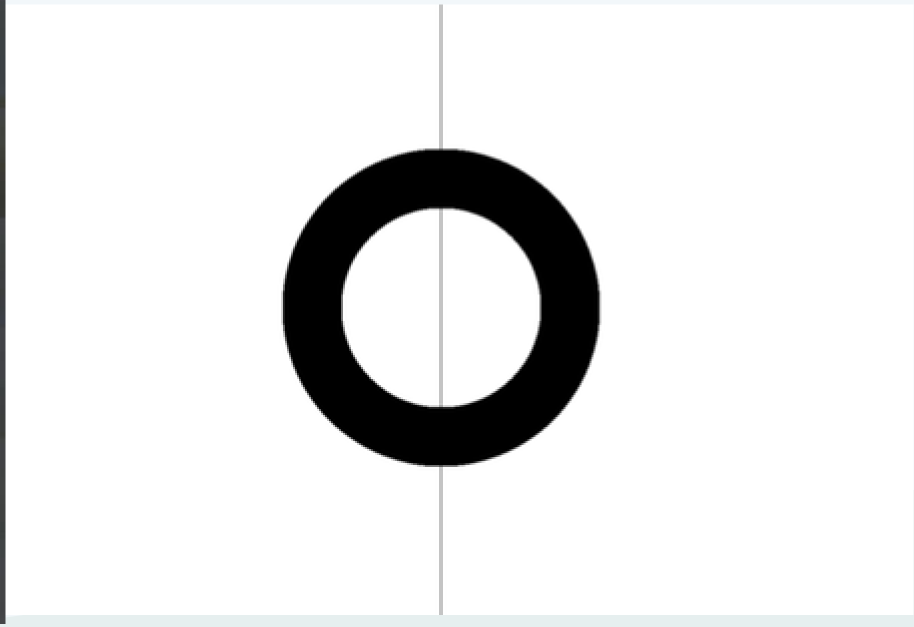
（b） Filter schematic

Pre-processing - Bandpass Filter

Pre-processing → Calculate Hash → Generate Result → Output



（a） Processed image



（b） Filter schematic

2

Hash Function

Marr-Hildreth Hash

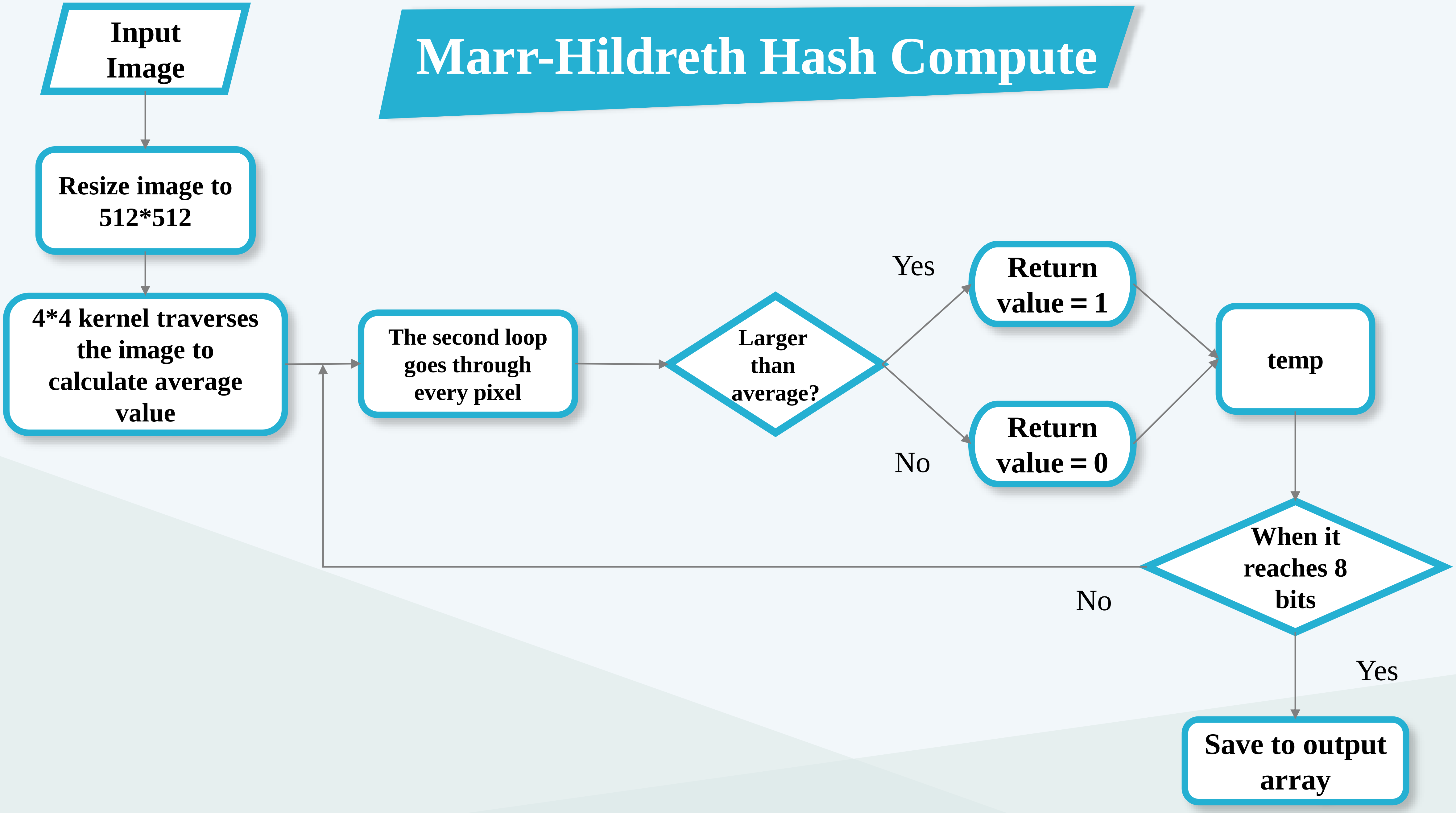
Pre-processing → Calculate Hash → Generate Result → Output

The basic idea of the Marr-Hildreth algorithm is to smooth the image using a Gaussian smoothing filter and then use the Laplace operator to calculate the second-order derivatives of the image. Based on the magnitude and direction of the second-order derivative, the edges of the image are calculated.

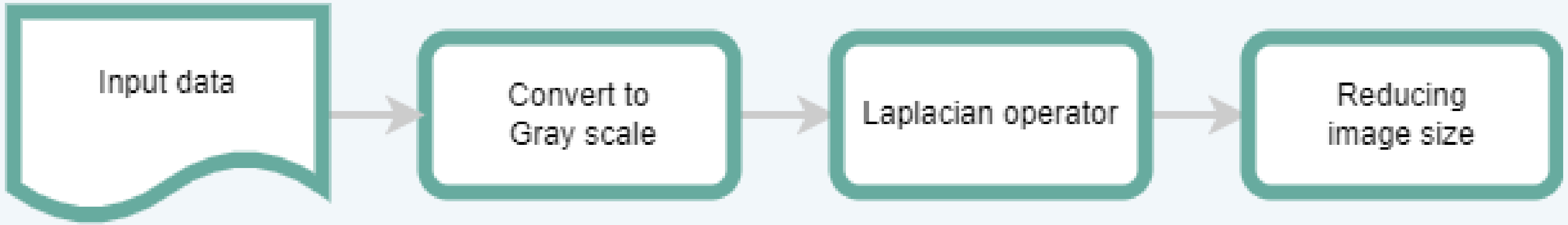
Advantages of the Marr-Hildreth algorithm：

1. Ability to detect fine edges in images.
2. Better performance when handling noisy images.

Marr-Hildreth Hash Compute



Marr-Hildreth Hash Compute



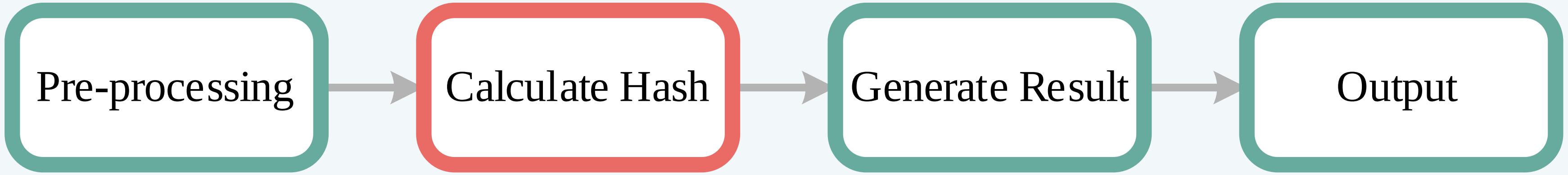
（a）

（b）

Original image size：1280∗720

Hash result size：72∗1

Marr-Hildreth Hash Compare



In image processing, we use the Hamming distance to measure the difference between two images to compare the similarity of two images.

The Hamming distance is calculated as follows.

1. Obtain the pixel values of the binary strings of the two images.
2. Compare the two binary strings by bit and calculate the different digits.
3. Add up the different digits to get the Hamming distance.

The final result of 0 means the same picture, and the larger the number means the more dissimilar the two pictures are.

Sorting

1. Find the image which with the largest hash difference.

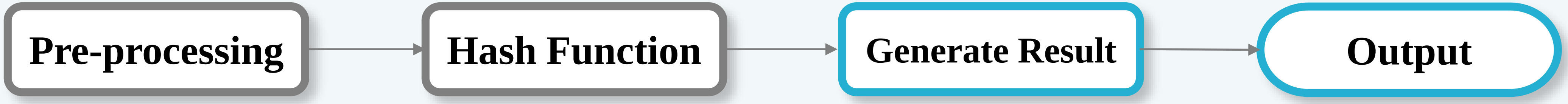
2. Find the image with the smallest hash difference as next frame.

3. When the hash difference is greater than the average of the hash change, the whole sequence is reversed and then continued.

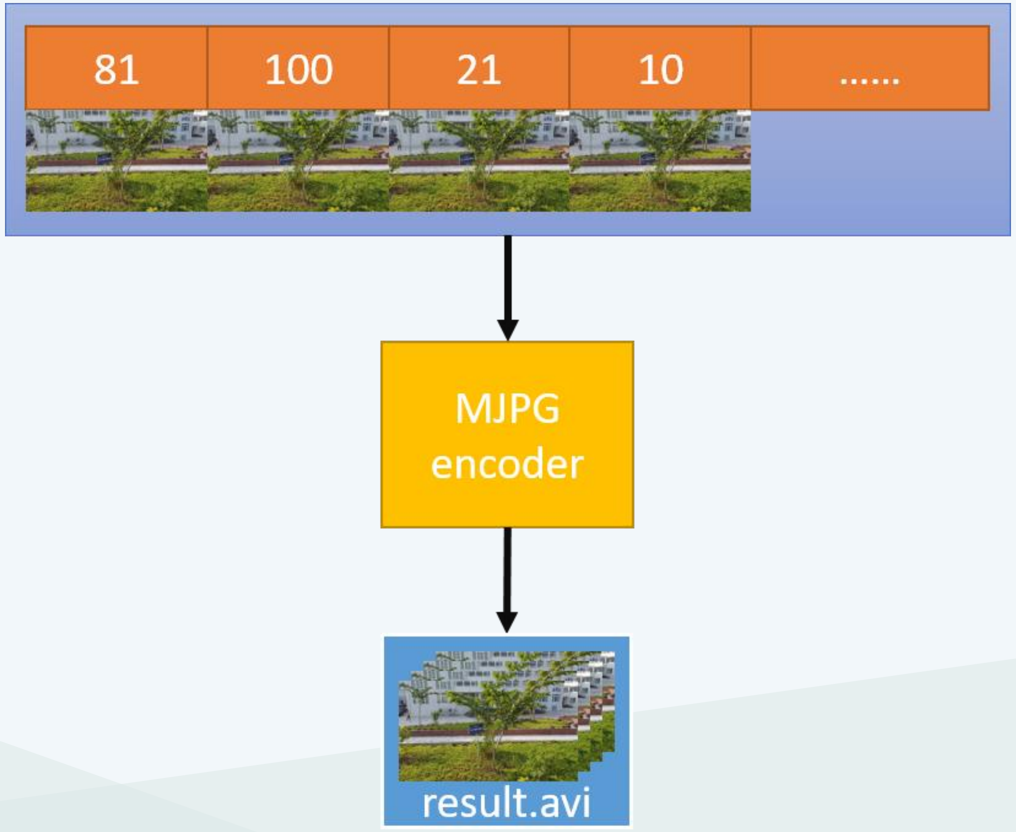
3

Generate Result

Generate Result



The final project uses the VideoWriter function in OpenCV to output avi video files using the MJPG encoder.



4

Parallel Computing

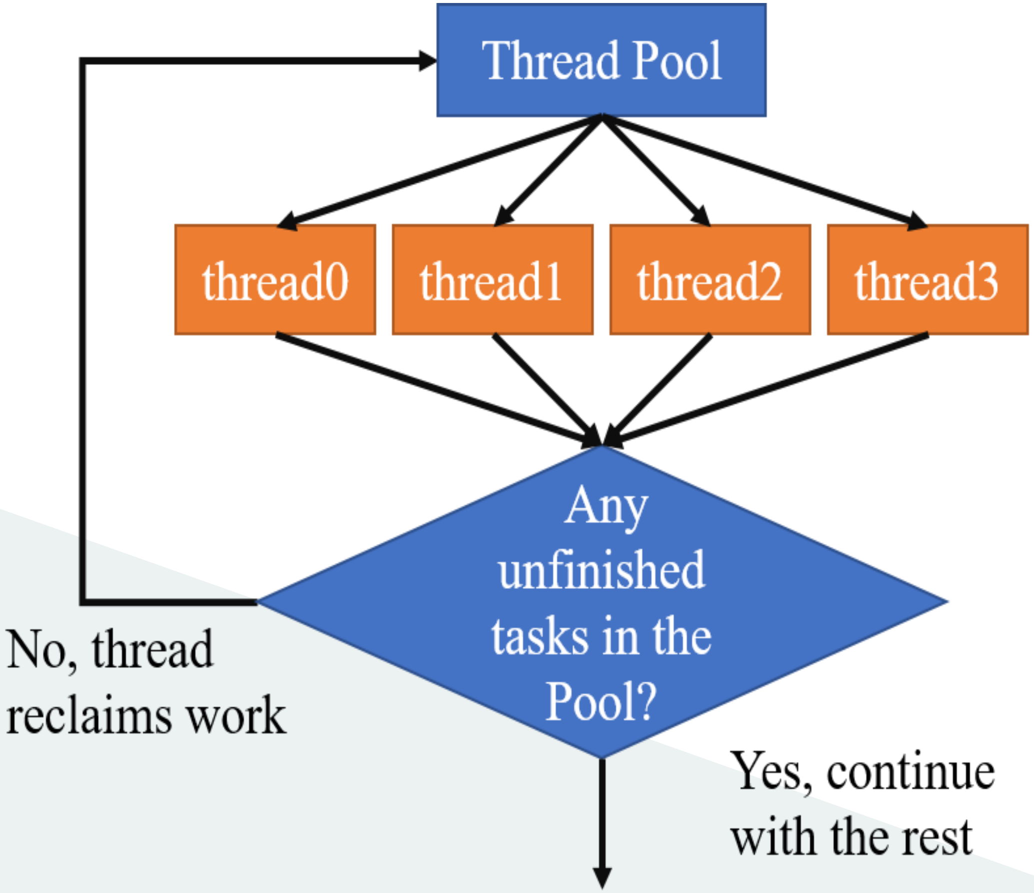
Parallel Computing

Pre-processing → **Hash Function** → **Generate Result** → **Output**

Since image pre-processing and hash calculation takes a lot of time, and due to the nature of C language, only one thread is used by default, the waiting time will be prolonged when multiple images need to be pre-processing and hash calculated, and the computer hardware cannot be used effectively.

However, if we can submit all the tasks to the Thread Pool first, and wait for the Thread Pool to finish all the tasks before running the rest of the programs, we can save a lot of time.

Parallel Computing



Thread Pool： Assigning tasks.

Thread： Process the assigned tasks.

Thread：Complete the task and submit the results.

5

Evaluation

Evaluation

$$\text{SRCC} = \left|1 - \frac{6\sum d_i^2}{n(n^2-1)}\right|$$

$$\text{MSE} = \frac{1}{mn}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}[I(i,j) - K(i,j)]^2$$

MSE for evaluating grayscale images.

$$\text{MSE} = \frac{1}{mnc}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}\sum_{c=0}^{2}[I(i,j,c) - K(i,j,c)]^2$$

MSE for evaluating channel color images.

$$\text{MSE} = \frac{1}{N}\sum_{x=0}^{N-1}\frac{1}{mnc}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}\sum_{c=0}^{2}[I_x(i,j,c) - K_x(i,j,c)]^2$$

MSE for evaluating videos.

i 、 j: location of the pixel
c : image channel
x : video frame
m : image height
n : image width
N : video frame count

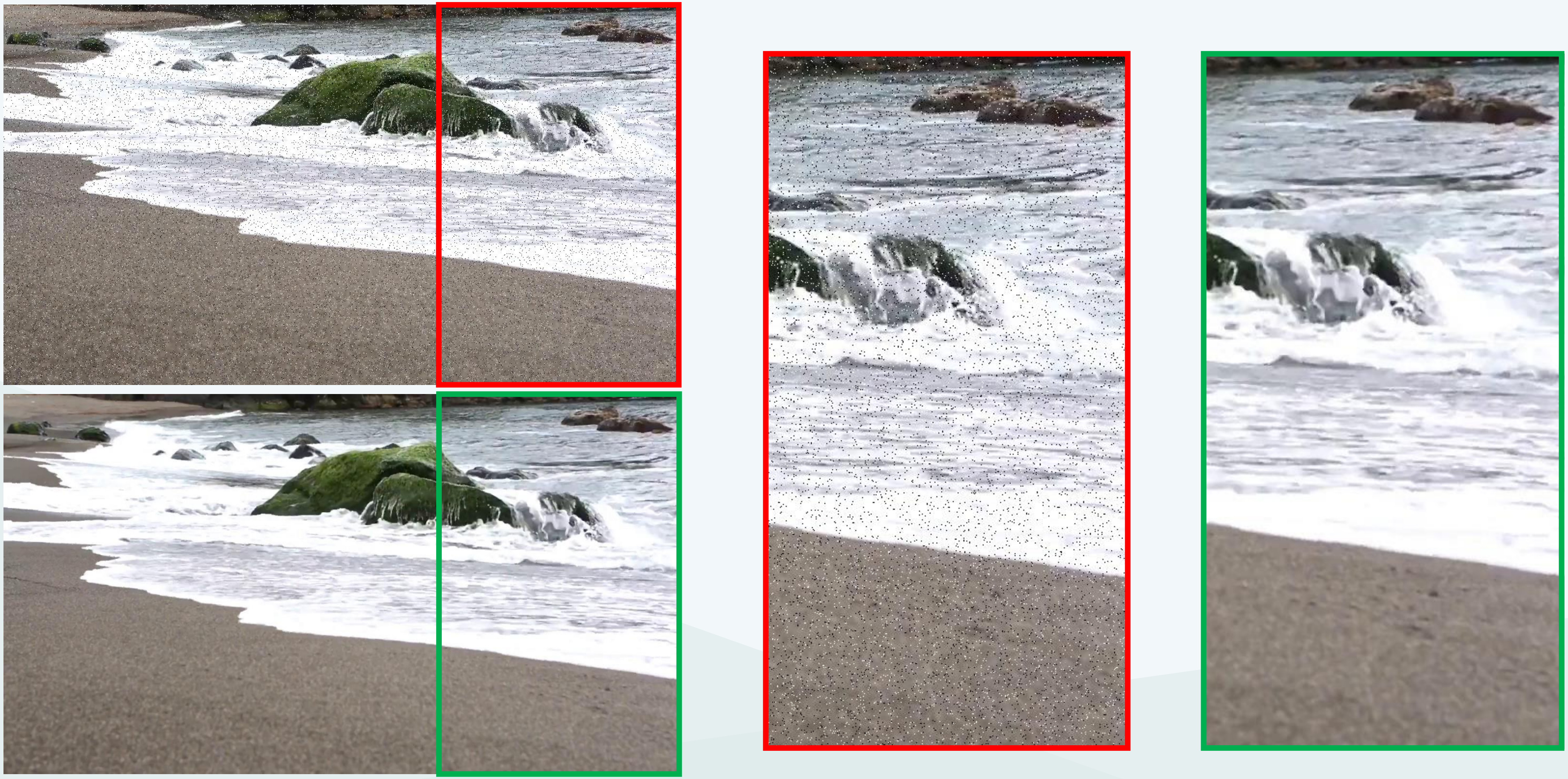
Evaluation

資料集	圖片張數	前處理時間 (sec)	SRCC	MSE	雜湊計算與排序時間 (sec)	處理時長總計(sec)
beach	270	6.718	0.9995	1477.7	0.271	7.522
Boat_style	338	8.063	0.6948	2827.6	0.357	8.980
candle_style	340	11.31	0.0242	7211.3	0.365	12.26
CCTV	162	10.29	0.2582	1553.2	0.303	11.33
coastline	165	4.183	0.4650	15834	0.158	4.681
Desert	142	1.750	0.8145	18181	0.095	1.999
DMC	600	14.38	0.0037	2138.5	0.742	16.36
flyout	450	11.08	0.9860	1102.2	0.493	12.50
helltaker	300	21.46	0.5877	1323.3	0.584	23.16
PUBG	43	1.425	0.0219	47041	0.042	1.573
RushPixar	300	7.498	0.2731	17160	0.287	8.407
School	230	5.942	0.9908	3130.2	0.219	6.637
soccer_style	150	4.537	0.4749	3408.2	0.135	5.000
TKUC	153	3.765	0.5500	15613	0.147	4.220
typing	720	16.78	0.1162	1939.3	0.926	19.070
全體平均	290.866	8.612	0.484	9329.366	0.341	9.579

6

Results

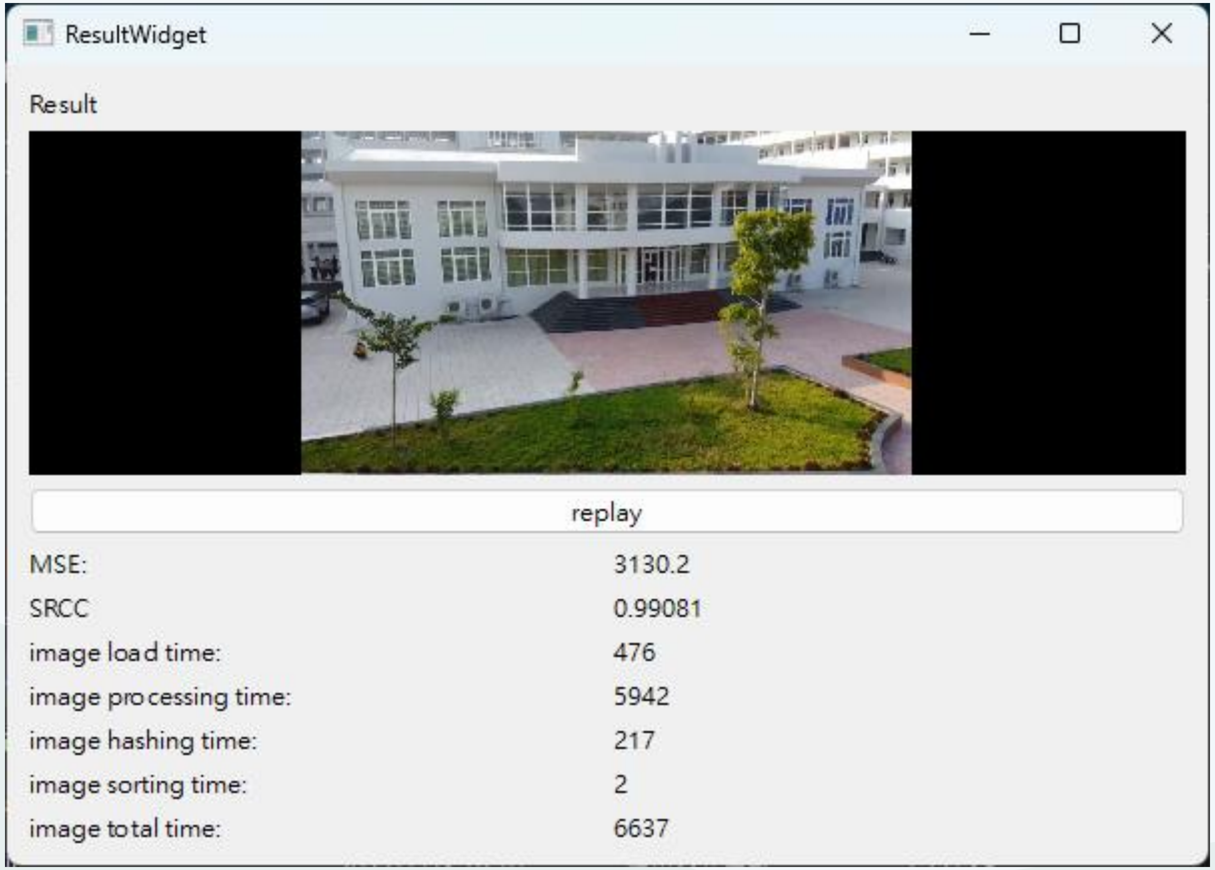
Results



Results

Results

Results


ResultWidget
Result
replay
MSE: 3130.2
SRCC 0.99081
image load time: 476
image processing time: 5942
image hashing time: 217
image sorting time: 2
image total time: 6637

Work Distribution Chart

	前處理 Case 1	前處理 Case 2	前處理 Case 3	雜湊計算排序	評估與輸出	優化執行程式整合	報告書與簡報製作	總計
黃俊傑	✓ 100%			✓ 50%	✓ 100%	✓ 15%	✓ 15%	34%
吳翊睿		✓ 100%				✓ 85%	✓ 15%	34%
吳佩霖			✓ 100%	✓ 50%			✓ 70%	32%

Reference

- [1] https://github.com/opencv/opencv_contrib
- [2] https://github.com/vit-vit/CTPL
- [3] https://ppfocus.com/0/mic6b8c41.html

Thank you for listening