Final Project Report

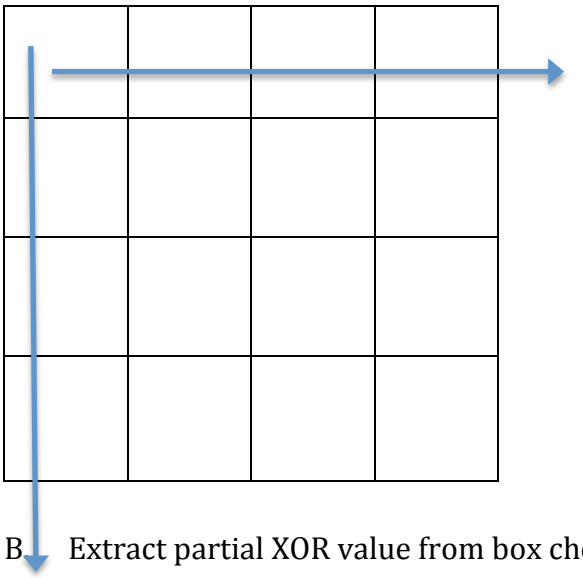
- Renfei Wang
- Shaowei Su

1. Description of algorithm

The solution is based on the fact that after the random swap between rows and columns, all elements of certain row/column keeps the same, which means that the XOR result of one specific row/column is fixed. Except this, we can split XOR result from one box checksum (2x2, 4x4 or 8x8) into several distinct pieces corresponds to different rows/columns. By these separate XOR values extracted from box checksum, we will be allowed to calculate the XOR values to every row and column for the original pictures. And then after simple comparison with the XOR values calculated through distorted pictures, we can swap between rows and columns to get the original picture.

Now I will demonstrate the entire unscrambled process step by step.

A. Calculate XOR values for each row and column of distorted pictures



B. Extract partial XOR value from box checksum CSV files

For example, if the box size is 8x8 then the corresponding XOR will be 64-bit.
(1010498490797080327 is the column checksum to the first box in large.png_8x8.csv)

1010498490797080327 →

```
0000 1110 0000 0110 0000 0011 0000 0110
63                  47                  32
0000 0110 0000 0111 0011 1111 0000 0111
31                  15                  0
```

Then we can split the 64-bit value into 8 pieces with each piece corresponds to XOR value of one row inside the box. Finally, we get all the partial XOR results saved to array result_xor[].

C. Combine partial XOR results into row/column XOR value

Literally, we are going to find the XOR value of entire row/column through the partial values we found

in B.

D. Compare XOR values in A and C

Through comparison we can locate one specific row/column to its original position and then swap them. After all the swaps, we finally get the original pictures.

2. CUDA kernel design

The CPU version of unscrambling strictly follows our algorithm described above and the results are quite satisfying: every possible combination of PNG file and CSV file input can be done within 1 second. But there is still a chance to accelerate through GPU parallelization.

Since the XOR value can be calculated independently within boxes/rows/columns, all of the computation part with multiple for loop inside can be conducted in parallel inside kernel. Specifically, we are going to launch Number of boxes in one row * M kernels.

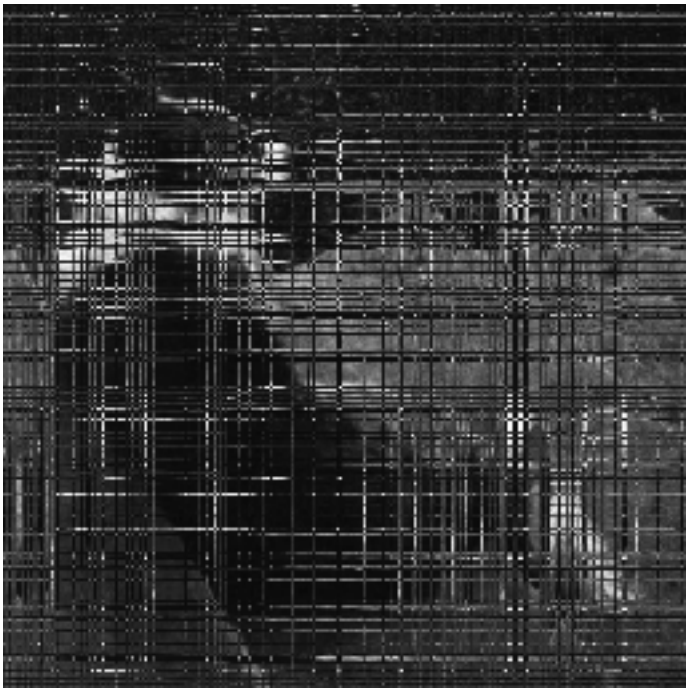
3. Results

3.1 Execution time

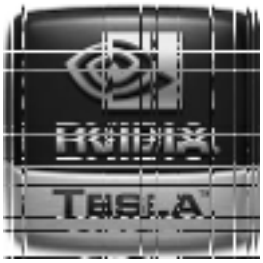
cpu version				
		Inputed csv file		
	Exe time(ms)			
Inputed png file		2x2	4x4	8x8
	large	48.45	1.61	1.38
	medium	0.48	0.25	0.22
	small	0.08	0.05	0.05
gpu version				
		Inputed csv file		
	Exe time(ms)			
Inputed png file		2x2	4x4	8x8
	large	0.36	0.32	0.32
	medium	0.16	0.13	0.14
	small	0.08	0.09	0.09

3.2 Original pictures

Large:



Medium:



Small: