Depix——a tool for recovering passwords from pixelized screenshots

Github Page: http://github.com/beurtschipper/Depix

Reporter: Jiawei Shan

Institute of Statistics & Big Data
Renmin University of China

December 30, 2020

Introduction

1. What is Pixelization?
2. How to implement pixelization?
3. How to recover from pixelization?

Introduction

What is Pixelization?

Introduction

What is Pixelization?

Introduction

What is Pixelization?

Introduction

- Pixelization describes the process of partially lowering the resolution of an image to censor information.
- The **linear box filter** method is commonly used to implement pixelization, which is simple and works fast.
- A linear box filter takes a box of pixels, and overwrites the pixels with the average value of all pixels in the box.

Here is an illustation of the linear box filter:

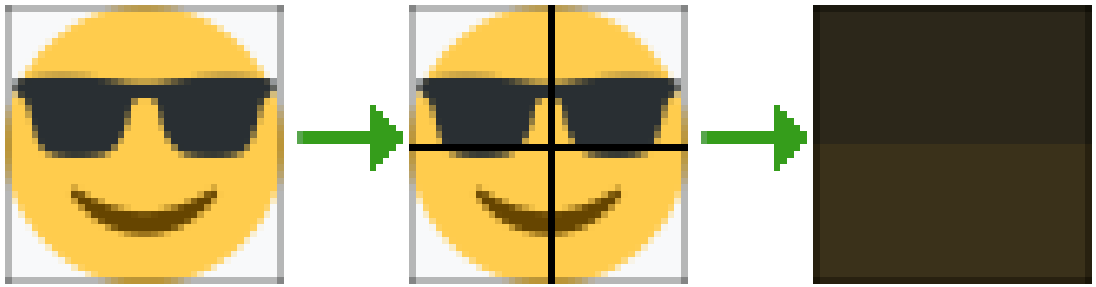

Figure: Illustration of the linear box filter.

Introduction

Depix is a tool for recovering passwords from pixelized screenshots, and the image below shows one of the test results.

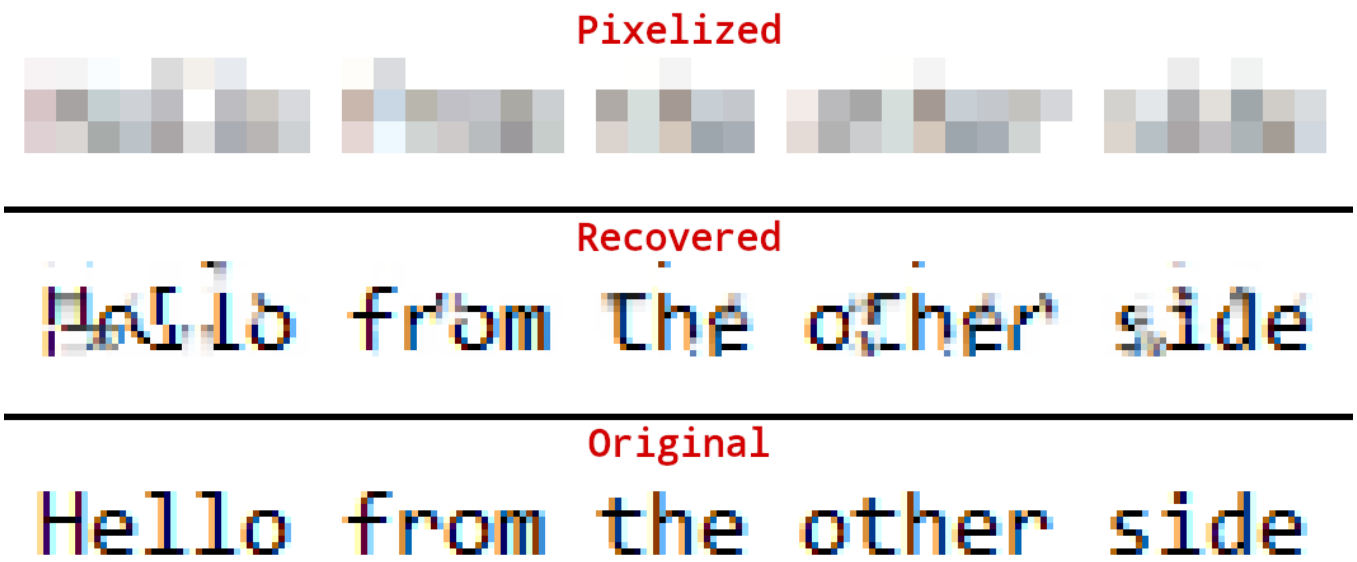

Figure: Test result of Depix.

Algorithm description

- Since the linear box filter is a deterministic algorithm, pixelizing the same values will always result in the same pixelated block.
- Pixelizing the same text - using the same locations of blocks - will result in the same block values.
- We can try to pixelate text to find matching patterns.
- The key to find matching is that the exact block of the same configuration of pixels to exist in the search image.
- This solution is quite simple: take a **De Bruijn sequence** of expected characters, paste it in the same editor, and make a screenshot of that. That screenshot is used as a lookup image for similar blocks.

Algorithm description

De Bruijn sequence

- In combinatorial mathematics, a **De Bruijn sequence** of order n on a size-k alphabet A, denoted by $B(k, n)$ is a cyclic sequence in which every possible length-n string on A occurs exactly once as a substring (i.e., as a contiguous subsequence).
- For example, taking $A = \{0, 1\}$, there are two distinct $B(2, 3)$: 00010111 and 11101000, one being the reverse or negation of the other.

Algorithm description

- Here is an example of De Bruijn sequence with $n = 2$ and $A = \{0 - 9\} \bigcup \{a - z\} \bigcup \{A - Z\}$.
- It's important that 2-character combinations are used, because some blocks can overlap two characters.

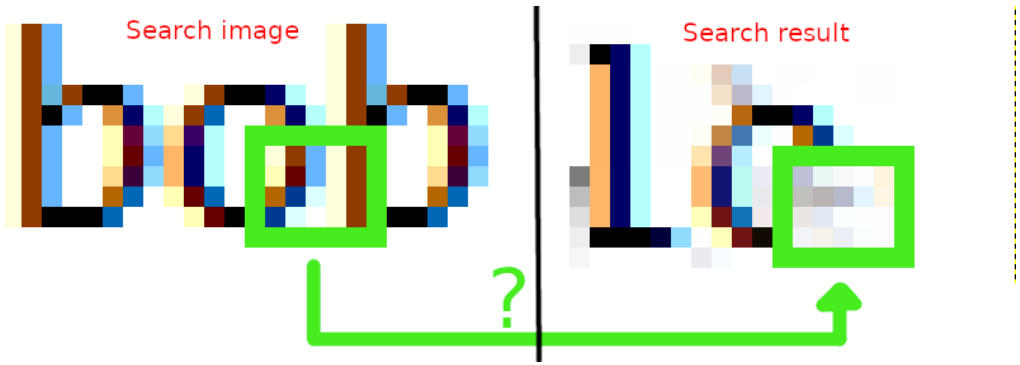

Figure: Search using De Bruijn sequence.

Conclusion

Limitations:

- The expected characters which is possible to exist in the image should be given in advance with the same font settings (text size, font, color, hsl) and in the same editor.
- This algorithm is helpless to recover general images from pixelization.
- An image with both spaced and close letters takes longer to search.

Finally, we show a test result in jupyter notebook[1].

[1] ssh-connection