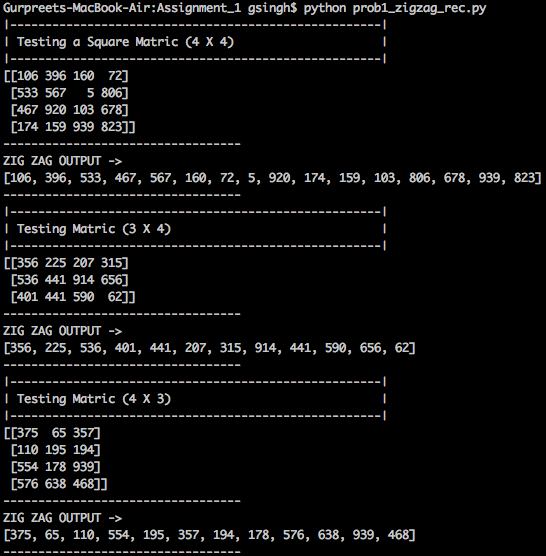
|  |  |
| --- | --- |
| CSc 479 Digital Libraries (Spring 2015) | Student: Gurpreet Singh |
| Professor Jie Wei | Assignment 1 |

1. Implement the zig-zag encoding procedure for a rectangular matrix.

Below is the output of my implementation of the zig - zag encoding procedure.



1. Prove for two symbols a and b, if p(a) >= p(b), then according to Huffman encoding algorithm, the resultant code length L(a) <= L(b).

**Basis Step**

The smallest k, the number of symbols, that we can have is k = 2. Let A be symbol 1 with probability 0.5, let B be symbol 2 with probability 0.5, and then we can assign A with 0 and B with 1.

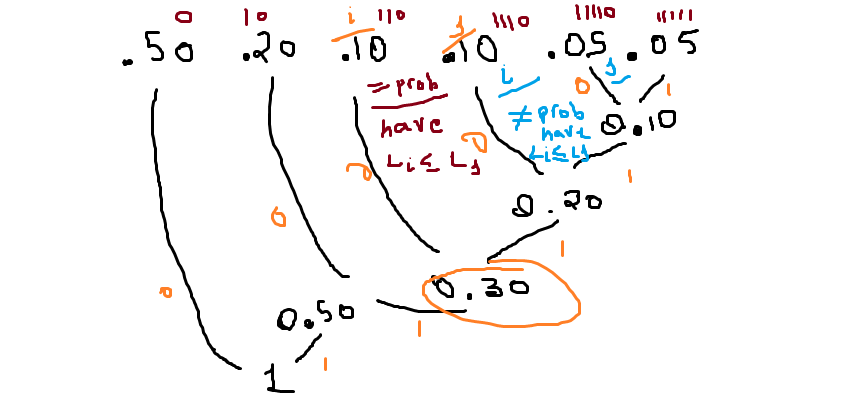
**Induction Hypothesis**

Assume that this works correctly for all lengths smaller than k, where k >= 3, and if p(a) >= p(b), then the resultant code length L(a) <= L(b).

**Induction Step**

We have two cases to consider: X and Y are either the leading symbols, meaning they have the highest probability, or they are not the leading symbols.

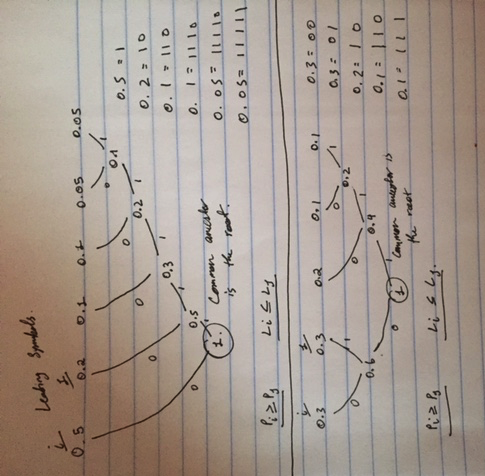
If they are not the leading symbols, consider the common ancestor, CA, of both symbols X and Y. If CA is not the root then the tree of CA has less then k symbols, including X and Y, and by induction we know it works correctly. Here is an example of that:



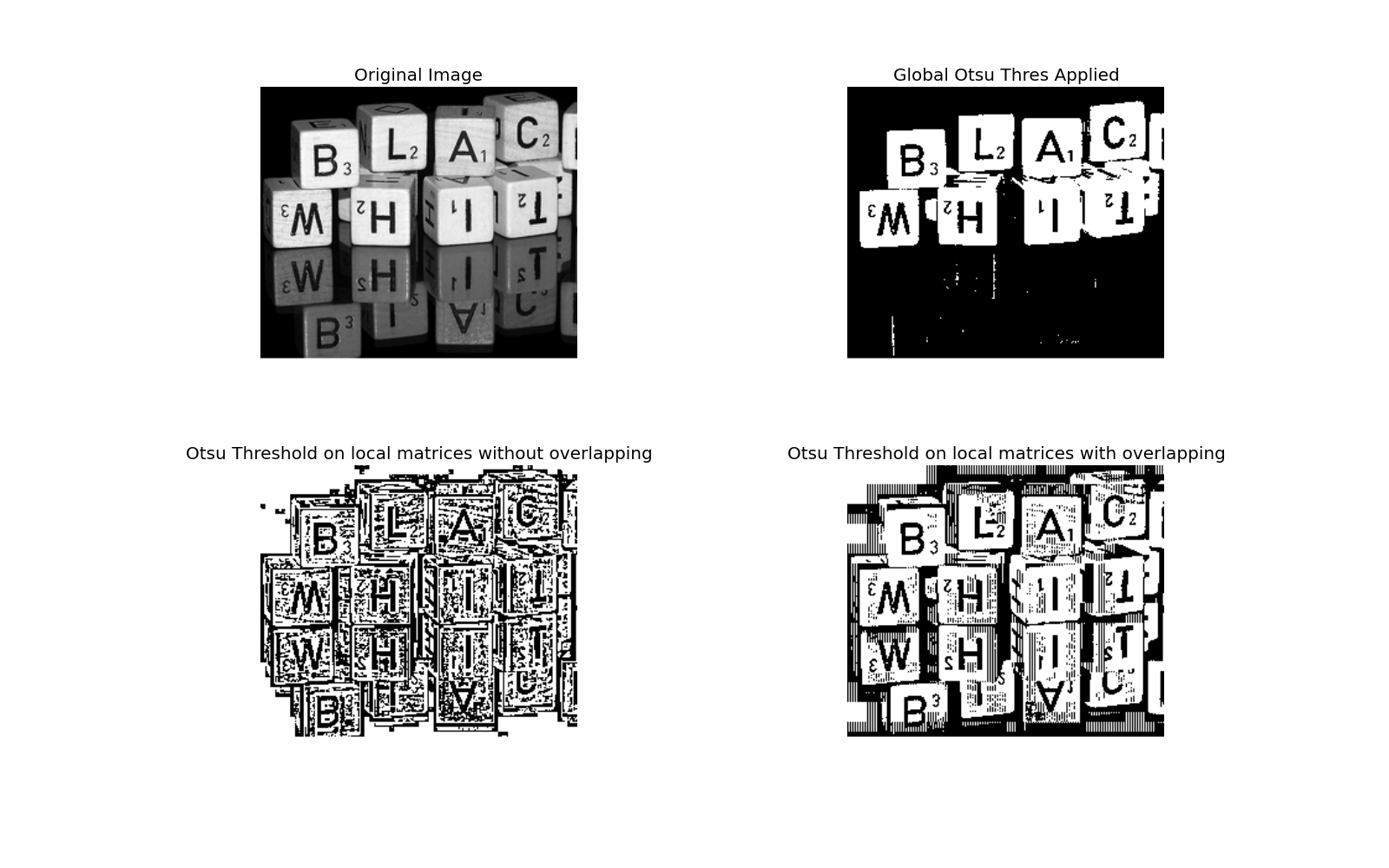
When X and Y are the leading symbols:

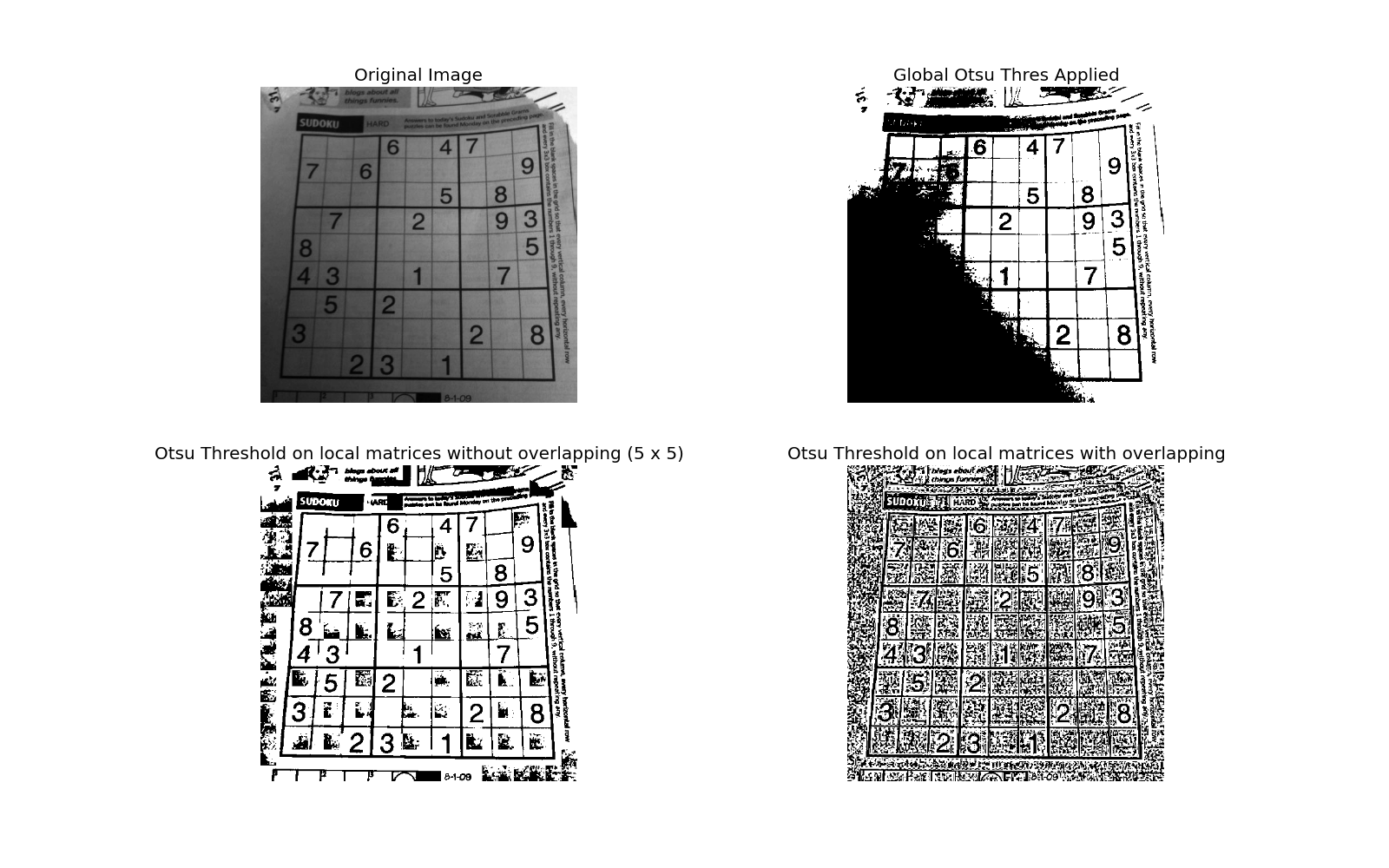
Suppose we have two symbols X and Y, with P(X) > P(Y). We have k >= 3. Tracing the parent nodes of X and Y: The huffman coding algorithm takes the two smallest probabilities and creates a new tree from those probabilities until we have only one tree. If the parent node of X is the root this means the length of X is less then all other symbols. The depth of the huffman tree is the length of the encoding of symbols.

If the parent node of X is not the root, we can continuly follow the grand parents of both X and Y and we will reach the root. It will be X to reach the root first because it had the highest probabily and accordinng to the huffman coding algorithm it will be the last to create a subtree. See example below:

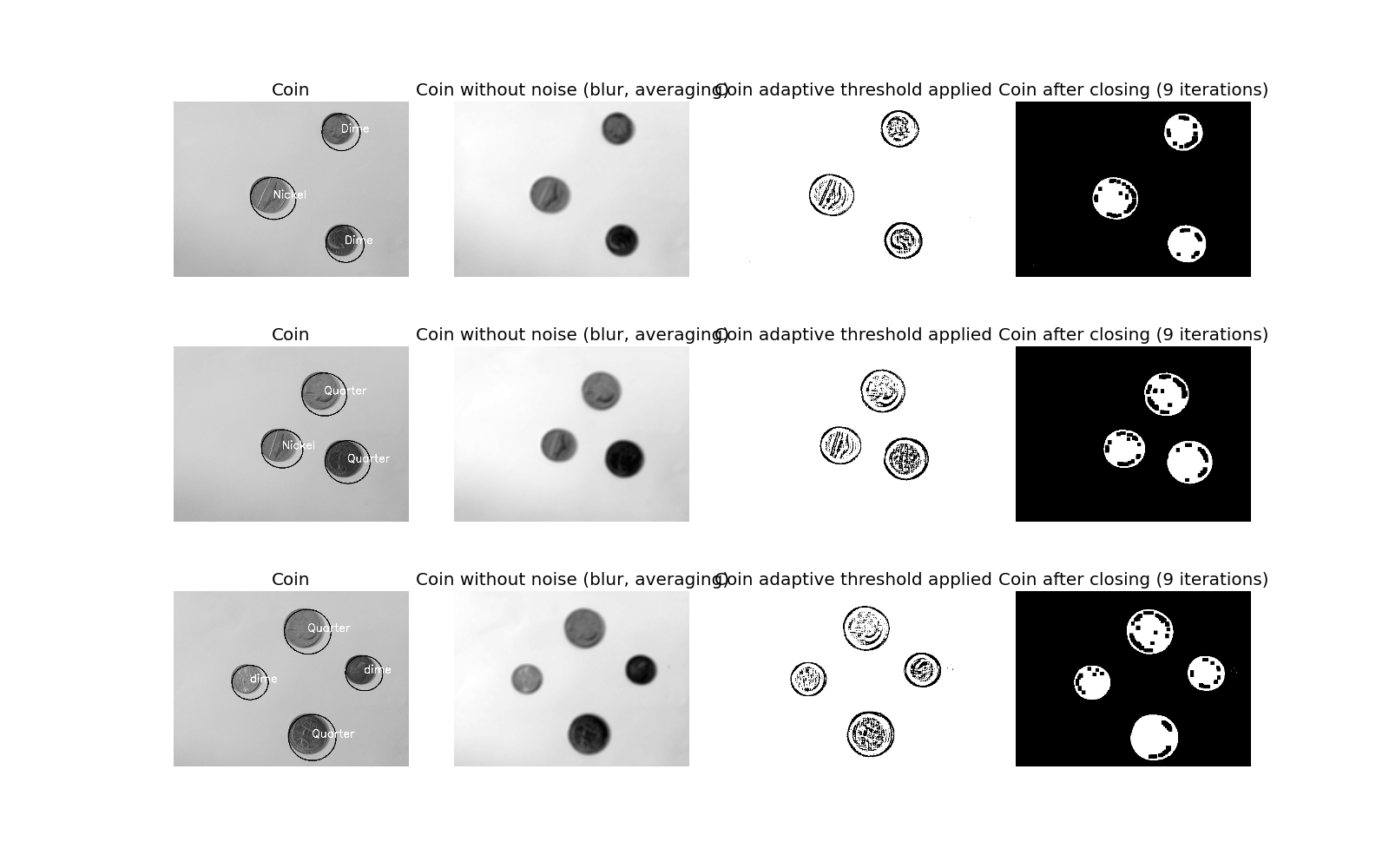


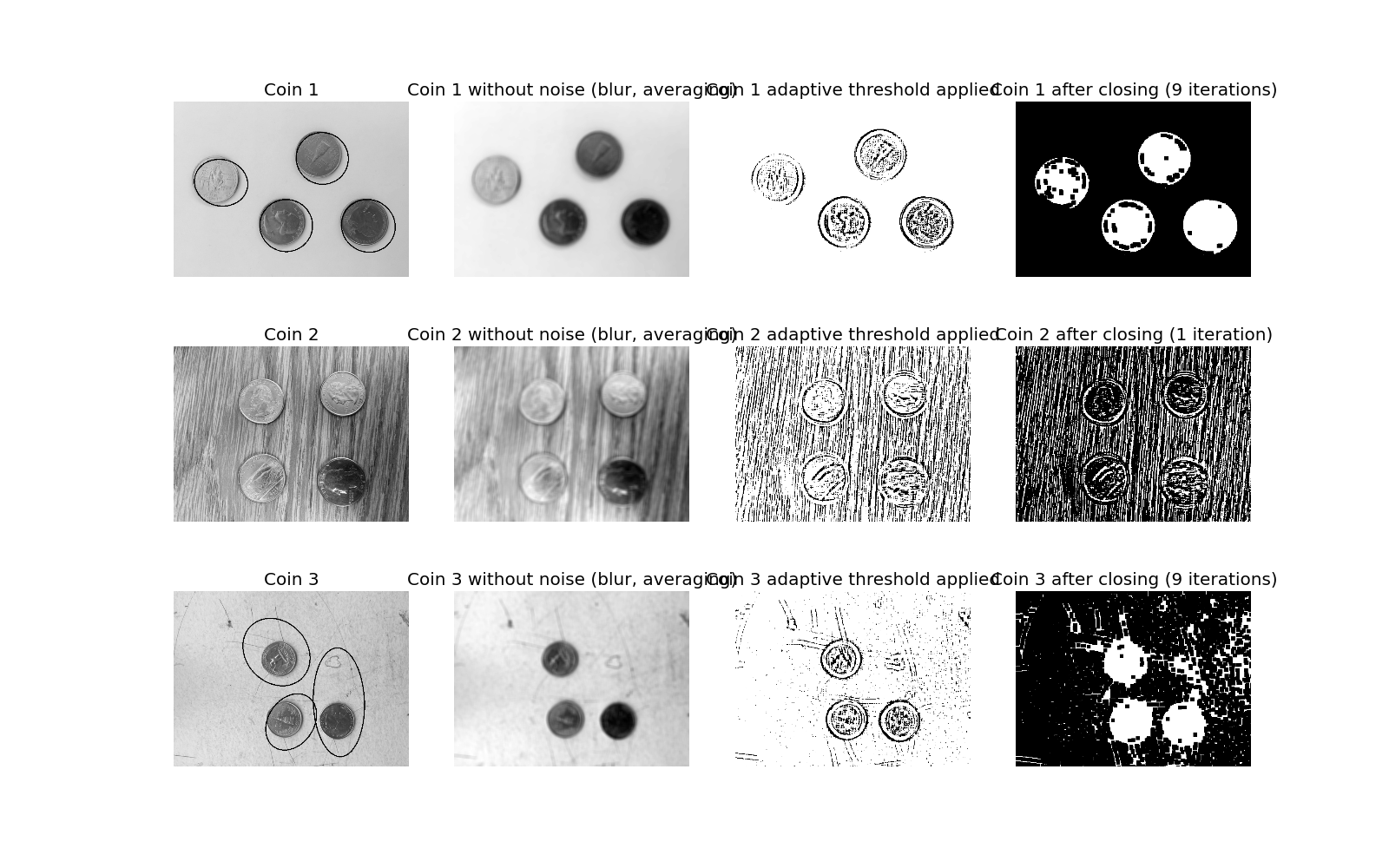
1. Implement two different versions of the adaptive Otsu’s thresholding algorithm, i.e., with or without overlapping local matrices, then compare the performances of these two algorithms.





1. Take some pictures of at least two coins from different background, and come up with a program to identify these coins in a robust manner.





Note:

This is an individual assignment, you should be the sole author of all the codes; use of any code from the Internet is viewed as cheating.

Deadline is Mar. 30, 11:59pm by email.

Submit source code, screen shot of sample running results of all four programs.