# 3x3 Kernel Comparison Algorithm

This documentation explained the theories behind the 3x3 kernel algorithm of ‘comp\_pixel’ here:

https://github.com/khanh93vn/CogAlg/commit/f46e02c6c68d0ad25ed7ce36a68d82a1da67427f

## Equation Derivation

Based on Figure 1, given a pixel (in red) and there are 8 pixels surrounding the pixel (central pixel),

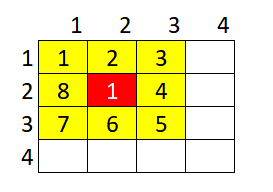


Figure 1. Central pixel with 8 surrounding pixels

The direction derivative is summarized in Equation 1:

|  |  |
| --- | --- |
|  | (1) |

Where:

is the direction derivative,

is the total surrounding pixels and in this case = 8,

is the th coefficient,

is the th pixel differences.

From Equation 1, it could be expanded to Equation 2:

|  |  |
| --- | --- |
|  | (2) |

Where:

is the th pixel,

is the central pixel.

Then, Equation 2 is separated into 2 sequences of equal length as shown in Equation 3:

|  |  |
| --- | --- |
|  | (3) |

The sequence in the left and right should have similar length given is even ( = 8 in this case).

From Equation 3, expand again the in right sequence and form Equation 4:

|  |  |
| --- | --- |
|  | (4) |

By using Equation 4, with n = 8, the series would form Equation 5：

|  |  |
| --- | --- |
|  | (5) |

From the sequence in Equation 5, we can conclude that the series would form Equation 6:

|  |  |
| --- | --- |
|  | (6) |

It is proven that diametrically opposed pixels are have the same corresponding coefficients with opposing sign, hence Equation 7 is formed from Equation 6:

|  |  |
| --- | --- |
|  | (7) |

Where:

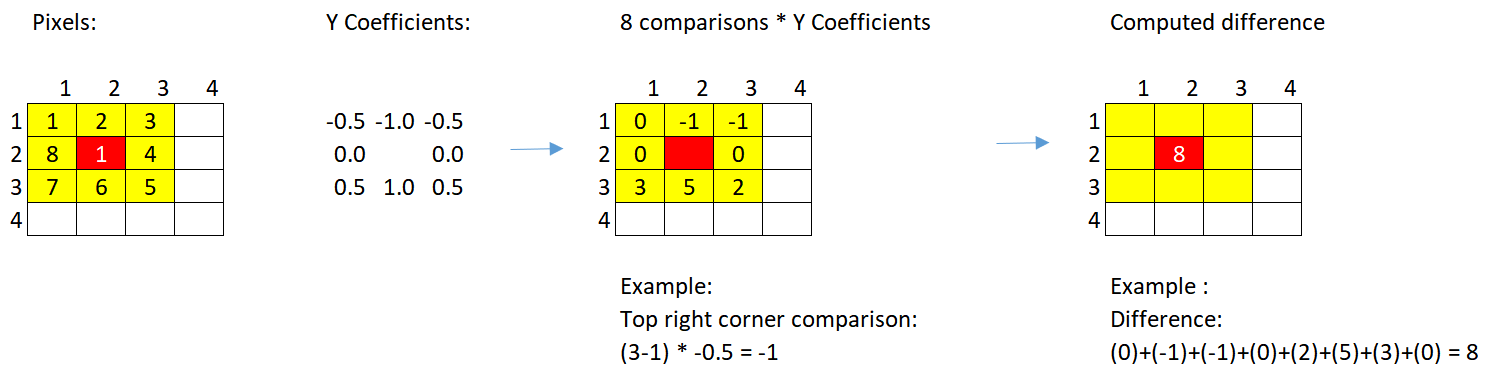
From Equation 7, it would form Equation 8 which is the method used in the update:

|  |  |
| --- | --- |
|  | (8) |

The similar concept is applied on direction derivative as well.

## Visualization

Original 3x3 kernel comparisons:



Optimized method:

