RBE549 - Homework 5

Keith Chester

Due date: October 6, 2022

Problem 1

In this problem we are looking at how the SURF algorithm integral images quickly. To this end we will be demonstrating how to compute the sum of a rectangular regon of an image using only a 3 operations - $+ and -$. An example image of the region is:

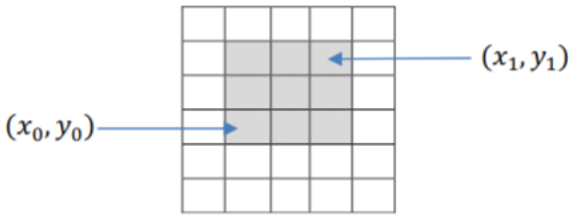

Figure 1: Edges and verticies

Based on the method and direction that the integral image was calculated, we can identify key squares that we can utilize to do our calculation - P, Q, S, and R:

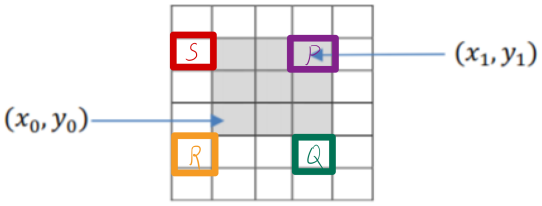

Figure 2: P, Q, S, and R identification

From here we can identify each of P, Q, S, and R via their positions relative to our initial definitions of our region, (x_0, y_0) and (x_1, y_1):

- $P = (x_1, y_1)$
- $Q = (x_1, y_0 - 1)$
- $S = (x_0 - 1, y_1)$
- $R = (x_0 - 1, y_1 - 1)$

With these values you can calculate the sum of the given region by performing the following calculation:

$$I_\Sigma[i, j] = P - Q - S + R \tag{1}$$

B

In this problem we are tasked with with computing a $9x9$ D_{xx} using SURF. To do this we perform the following:

0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

+

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

=

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

(2)

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

+

0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1

=

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

(3)

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

+

0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
-3	-3	-3	-3	-3	-3	0	0	0
-3	-3	-3	-3	-3	-3	0	0	0
-3	-3	-3	-3	-3	-3	0	0	0
-3	-3	-3	-3	-3	-3	0	0	0
-3	-3	-3	-3	-3	-3	0	0	0
-3	-3	-3	-3	-3	-3	0	0	0
-3	-3	-3	-3	-3	-3	0	0	0

=

0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
-2	-2	-2	-2	-2	-2	1	1	1
-2	-2	-2	-2	-2	-2	1	1	1
-2	-2	-2	-2	-2	-2	1	1	1
-2	-2	-2	-2	-2	-2	1	1	1
-2	-2	-2	-2	-2	-2	1	1	1
-3	-3	-3	-3	-3	-3	0	0	0
-3	-3	-3	-3	-3	-3	0	0	0

(4)

0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
-2	-2	-2	-2	-2	-2	1	1	1
-2	-2	-2	-2	-2	-2	1	1	1
-2	-2	-2	-2	-2	-2	1	1	1
-2	-2	-2	-2	-2	-2	1	1	1
-2	-2	-2	-2	-2	-2	1	1	1
-3	-3	-3	-3	-3	-3	0	0	0
-3	-3	-3	-3	-3	-3	0	0	0

+

0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
3	3	3	3	3	3	0	0	0
3	3	3	3	3	3	0	0	0

=

0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
-2	-2	-2	-2	-2	-2	1	1	1
-2	-2	-2	-2	-2	-2	1	1	1
-2	-2	-2	-2	-2	-2	1	1	1
-2	-2	-2	-2	-2	-2	1	1	1
-2	-2	-2	-2	-2	-2	1	1	1
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

(5)

$$\begin{bmatrix} 0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0\\ -2&-2&-2&-2&-2&-2&1&1&1\\ -2&-2&-2&-2&-2&-2&1&1&1\\ -2&-2&-2&-2&-2&-2&1&1&1\\ -2&-2&-2&-2&-2&-2&1&1&1\\ -2&-2&-2&-2&-2&-2&1&1&1\\ 0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0 \end{bmatrix} + \begin{bmatrix} 0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0\\ -3&-3&-3&0&0&0&0&0&0\\ -3&-3&-3&0&0&0&0&0&0 \end{bmatrix} = \begin{bmatrix} 0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0\\ -2&-2&-2&-2&-2&-2&1&1&1\\ -2&-2&-2&-2&-2&-2&1&1&1\\ -2&-2&-2&-2&-2&-2&1&1&1\\ -2&-2&-2&-2&-2&-2&1&1&1\\ -2&-2&-2&-2&-2&-2&1&1&1\\ 0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0 \end{bmatrix} \tag{6}$$

$$\begin{bmatrix} 0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0\\ -2&-2&-2&-2&-2&-2&1&1&1\\ -2&-2&-2&-2&-2&-2&1&1&1\\ -2&-2&-2&-2&-2&-2&1&1&1\\ -2&-2&-2&-2&-2&-2&1&1&1\\ -2&-2&-2&-2&-2&-2&1&1&1\\ 0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0 \end{bmatrix} + \begin{bmatrix} 0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0\\ 3&3&3&0&0&0&0&0&0\\ 3&3&3&0&0&0&0&0&0\\ 3&3&3&0&0&0&0&0&0\\ 3&3&3&0&0&0&0&0&0\\ 3&3&3&0&0&0&0&0&0\\ 3&3&3&0&0&0&0&0&0\\ 3&3&3&0&0&0&0&0&0 \end{bmatrix} = \begin{bmatrix} 0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0\\ 1&1&1&-2&-2&-2&1&1&1\\ 1&1&1&-2&-2&-2&1&1&1\\ 1&1&1&-2&-2&-2&1&1&1\\ 1&1&1&-2&-2&-2&1&1&1\\ 1&1&1&-2&-2&-2&1&1&1\\ 3&3&3&0&0&0&0&0&0\\ 3&3&3&0&0&0&0&0&0 \end{bmatrix} \tag{7}$$

$$\begin{bmatrix} 0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0\\ 1&1&1&-2&-2&-2&1&1&1\\ 1&1&1&-2&-2&-2&1&1&1\\ 1&1&1&-2&-2&-2&1&1&1\\ 1&1&1&-2&-2&-2&1&1&1\\ 1&1&1&-2&-2&-2&1&1&1\\ 3&3&3&0&0&0&0&0&0\\ 3&3&3&0&0&0&0&0&0 \end{bmatrix} + \begin{bmatrix} 0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0\\ -3&-3&-3&0&0&0&0&0&0\\ -3&-3&-3&0&0&0&0&0&0 \end{bmatrix} = \begin{bmatrix} 0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0\\ 1&1&1&-2&-2&-2&1&1&1\\ 1&1&1&-2&-2&-2&1&1&1\\ 1&1&1&-2&-2&-2&1&1&1\\ 1&1&1&-2&-2&-2&1&1&1\\ 1&1&1&-2&-2&-2&1&1&1\\ 0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0 \end{bmatrix} \tag{8}$$

Problem 2

In this section we are tackling problems with quad trees.

A

We are tasked with creating an image from the quad tree output of [[0110]10[0100]]. The image would be presented below:

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

B

Now we're tasked with shifting the image left, with the rightmost column containing all zeroes, and then determining the resultant quad tree encoding. The expected image would be:

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

...this would encode to [1[1001]0[1000]].

C

In this problem, we are asked to describe a recursive algorithm to rotate a quad tree. For instance, given our original quad tree from section A:

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

This had a quad tree encoding of [[0110]10[0100]]. If we were to rotate this image 90° clockwise:

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

This has a quad tree encoding of [[0010][0011]10]. Code for this recursive approach can be found in *problem2.py*.

A description of it loosely: For each node of the quad tree, change the order of the tree to be last, then first through third. Add these nodes to the new resulting tree, but call the same rotation function on each child. If a node is a singular number (meaning all values within the quad tree are the same from that point on) you can just return that value back.

Problem 3

In this problem we aim to prove that rotating and translation an entire point cloud does not affect the distance between any two of the points within, or that $||\vec{x}_1^0 - \vec{x}_0^0|| = ||\vec{x}_1^1 - \vec{x}_0^1||$. We will let $\vec{x}_i^1 = R\vec{x}_i^0 + \vec{T}$.

First, we state our initial equation with the above definitions:

$$||\vec{x}_1^1 - \vec{x}_0^1|| = ||(R\vec{x}_1^0 + \vec{T}) - (R\vec{x}_0^0 + \vec{T})|| \tag{9}$$

...this we can expand and simplify and then get rid of the effect of our translation $\vec{T}$:

$$||\vec{x}_1^1 - \vec{x}_0^1|| = ||R\vec{x}_1^0 + \vec{T} - R\vec{x}_0^0 - \vec{T}|| \tag{10}$$

$$||\vec{x}_1^1 - \vec{x}_0^1|| = ||R\vec{x}_1^0 - R\vec{x}_0^0|| \tag{11}$$

This is equivlanet to the form:

$$||\vec{x}_1^1 - \vec{x}_0^1|| = ||R|| * ||\vec{x}_1^0 - \vec{x}_0^0|| \tag{12}$$

$||R|| = R^T R$, and $R^T R = 1$, so therefore:

$$||\vec{x}_1^1 - \vec{x}_0^1|| = ||\vec{x}_1^0 - \vec{x}_0^0|| \tag{13}$$

This proves that the resulting distance between two moved points would be calculated the same.

Problem 4

In this section, we are presented with the following segmented map with class labels:

1	0	0	0	1
1	1	1	1	1
1	0	1	0	1
1	0	0	0	1
1	2	2	2	1

A

Here we aim to list all regions and their classes, and edges and which regions they separate:

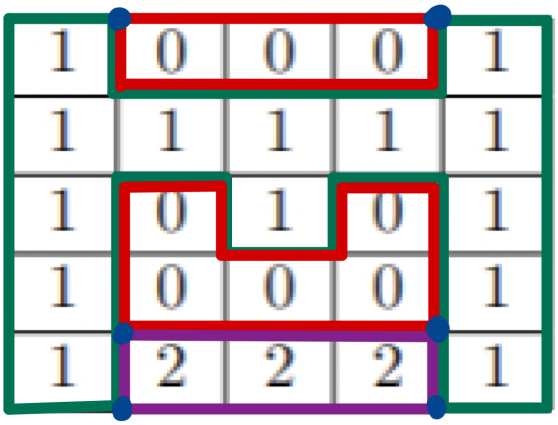

Figure 3: Edges and verticies

From this we can find the following regions and edges:

Region	Class
-1	-1
0	1
1	0
6	0
9	2

Edge	Class	Regions
1	-1	1
2	-1	0
3	0	1
4	1	2
5	0	1
6	0	2
7	1	2
8	-1	2
9	-1	1

B

We are asked here how many verticies are in this diagram. As labeled in the figure above, there are six verticies.

C

We are then asked if this is eight- or four- connected. This work could be either; there are no diagonals that would demonstrate an eight-connected setup, but that doesn't exclude it since eight-connected is a subset of four-connected.