1. **Zoom:**

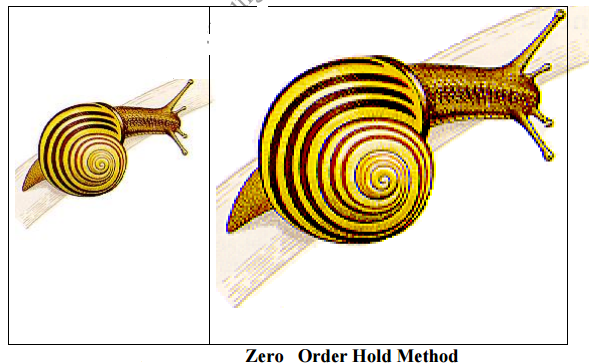
We use one of the following method, to enlarge an image:

1. Zero-Order Hold.

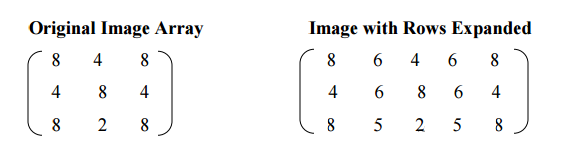
2. First \_Order Hold.

3. Convolution.

1. **Zero-Order hold:** is performed by repeating previous pixel values, thus creating a blocky effect as in the following figure:

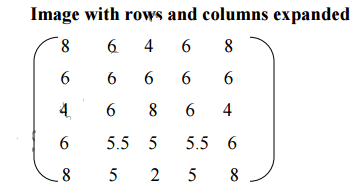


**2. First\_Order Hold:** is performed by finding linear interpolation between a adjacent pixels, i.e., finding the average value between two pixels and use that as the pixel value between those two, we can do this for the rows first as follows:



The first two pixels in the first row are averaged (8+4)/2=6, and this number is inserted between those two pixels. This is done for every pixel pair in each row.

Next, take result and expanded the columns in the same way as follows:

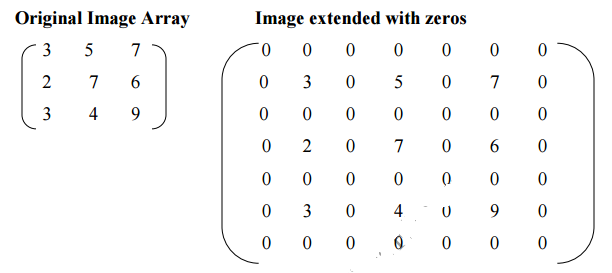


This method allows us to enlarge an N×M sized image to a size of (2N-1) × (2M-1) and be repeated as desired.

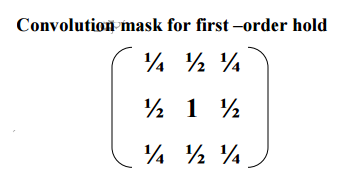
**3- Convolution**: this process requires a mathematical process to enlarge an image. This method required two steps:

1. Extend the image by adding rows and columns of zeros between the existing rows and columns.
2. Perform the convolution.

The image is extended as follows:



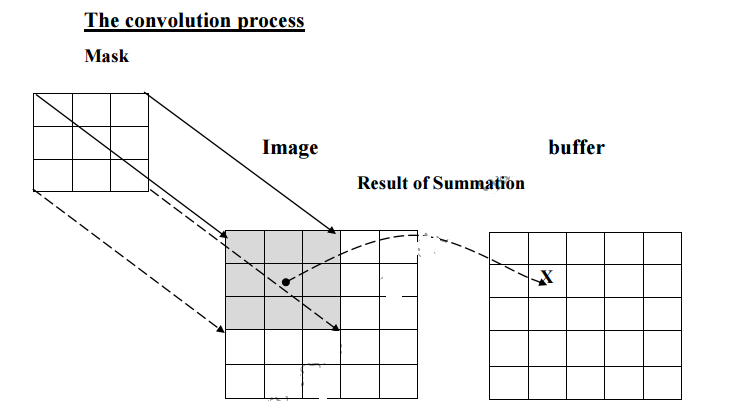
Next, we use convolution mask, which is slide a cross the extended image, and perform simple arithmetic operation at each pixel location



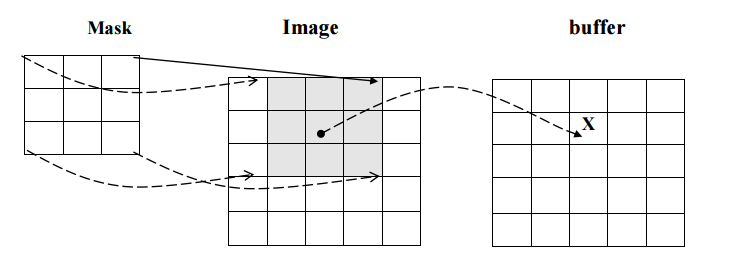
The convolution process requires us to overlay the mask on the image, multiply the coincident (متقابله) values and sum all these results. This is equivalent to finding the vector inner product of the mask with underlying sub image. The vector inner product is found by overlaying mask on sub image. Multiplying coincident terms, and summing the resulting products.

For example, if we put the mask over the upper-left corner of the image, we obtain (from right to left, and top to bottom): 1/4(0) +1/2(0) +1/4(0) +1/2(0) +1(3) +1/2(0) + 1/4(0) +1/2(0) +1/4(0) =3 Note that the existing image values do not change. The next step is to slide the mask over by on pixel and repeat the process, as follows: 1/4(0) +1/2(0) +1/4(0) +1/2(3) +1(0) +1/2(5) + 1/4(0) +1/2(0) +1/4(0) =4 Note this is the average of the two existing neighbors. This process continues until we get to the end of the row, each time placing the result of the operation in the location corresponding to center of the mask. When the end of the row is reached, the mask is moved down one row, and the process is repeated row by row. This procedure has been performed on the entire image, the process of sliding, multiplying and summing is called convolution.

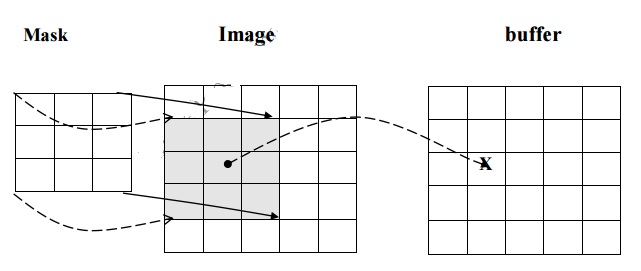
Note that the output image must be put in a separate image array called a buffer, so that the existing values are not overwritten during the convolution process.



**a.** Overlay the convolution mask in the upper-left corner of the image. Multiply coincident terms, sum, and put the result into the image buffer at the location that corresponds to the masks current center, which is (r,c)=(1,1).



b. Move the mask one pixel to the right , multiply coincident terms sum , and place the new results into the buffer at the location that corresponds to the new center location of the convolution mask which is now at (r,c)=(1,2), continue to the end of the row.



c. Move the mask down one row and repeat the process until the mask is convolved with the entire image. Note that we lose the outer row(s) and columns(s).

**Example (1):**

**a. Zero order:**

N\*M=2N\*2M

**b: first order:**

5\*5

N\*M (2N-1)\*(2M-1)

**Example ( 2) use convolution to enlarge the matrix A, with Mask**

**,**

SOLU:-

7\*7

N\*M (2N+1)\*(2M+1)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 3 | 4 | 5 | 6 | 7 | 0 |
| 0 | 2.5 | 4.25 | 6 | 6.25 | 6.5 | 0 |
| 0 | 2 | 4.5 | 7 | 6.5 | 6 | 0 |
| 0 | 2.5 | 4 | 5.5 | 6.5 | 7.5 | 0 |
| 0 | 3 | 3.5 | 4 | 6.5 | 9 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 3 | 4 | 5 | 6 | 7 | 0 |
| 0 | 3 | 4 | 6 | 6 | 7 | 0 |
| 0 | 2 | 5 | 7 | 7 | 6 | 0 |
| 0 | 3 | 4 | 6 | 7 | 8 | 0 |
| 0 | 3 | 4 | 4 | 7 | 9 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |