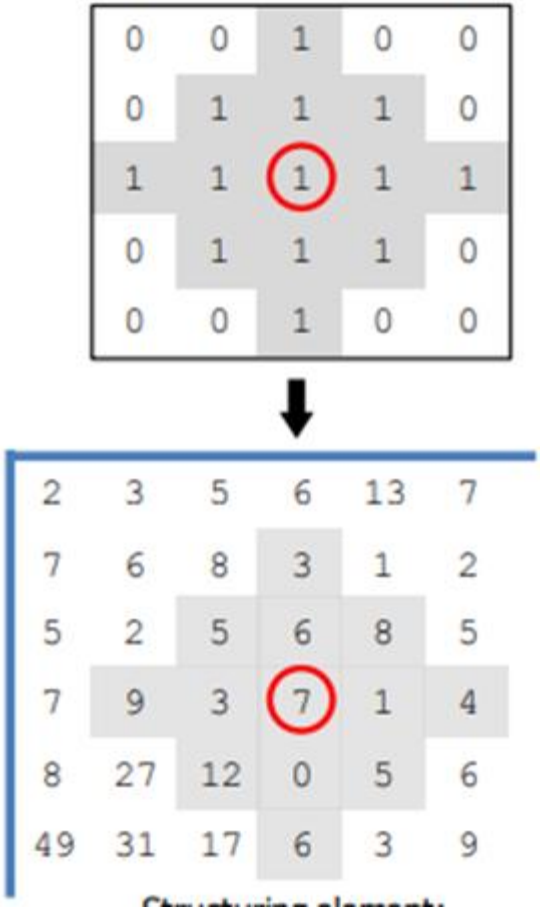
Morphology Operations

Dilation,Erosion, Opening, Closing, Boundary Extraction

ECE 5772, Robert McInerney
Oakland University, Professor Llamocca

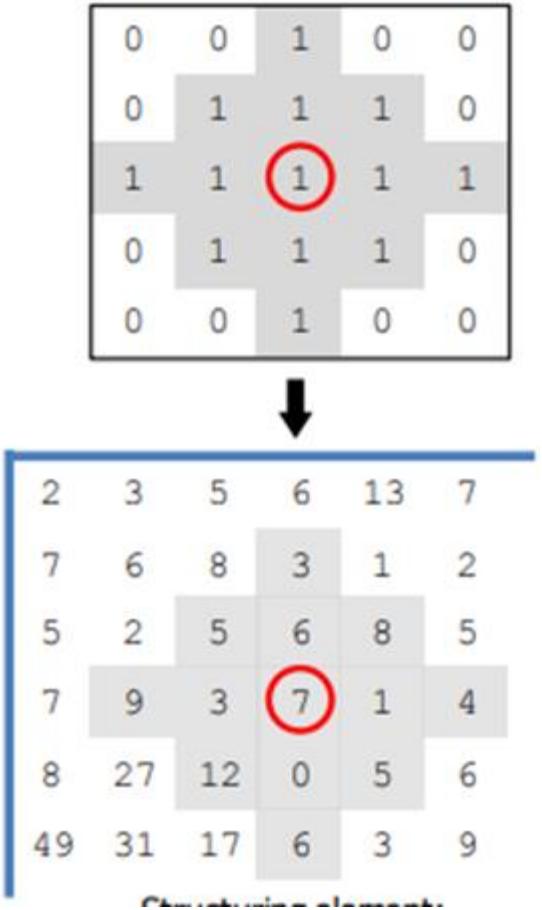
Morphology

- Morphology operations are commonly used in image processing
- They can be used to remove noise and even extract boundaries of images
- Morphology is done using a structuring element on each pixel of a grayscale image
- This project was done using a raster scan approach, afterwards it is checked using matlab



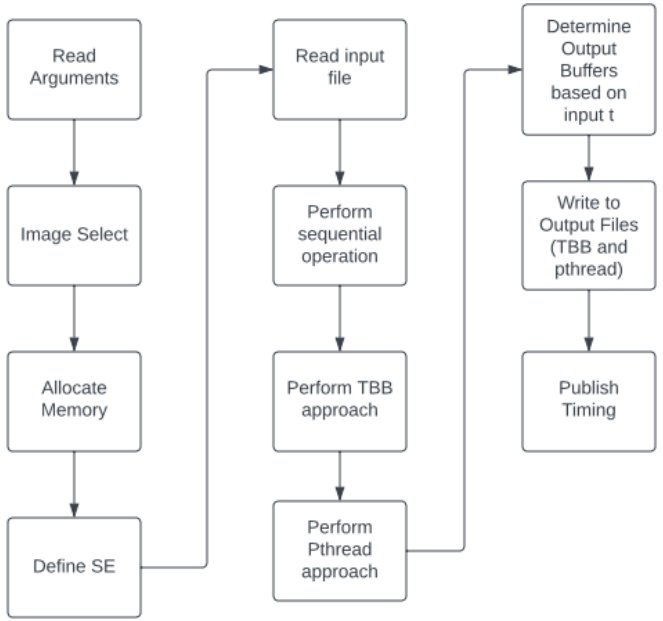
Morphology

- Dilation (max value in SE matrix) (adds pixels from borders)
- Erosion (min value in SE matrix) (removes pixels from borders)
- Opening (erosion followed by dilation) (used for removing small lines)
- Closing (dilation followed by erosion)(used for filling small holes
- Boundary extraction inner (original image - erosion)
- Boundary extraction outer (Dilation – original image)



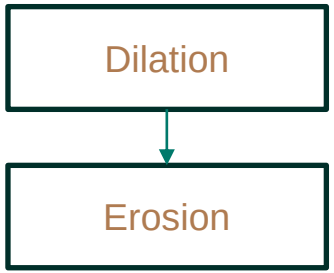
Program Flow

- Main Program Flow
- Input Arguments t, image_sel, SE_size, thread count
  - T = 1 dilation
  - T = 2 Erosion
  - T = 3 Opening
  - T = 4 Closing
  - T = 5 Inner boundary extraction
  - T = 6 Outer Boundary Extraction



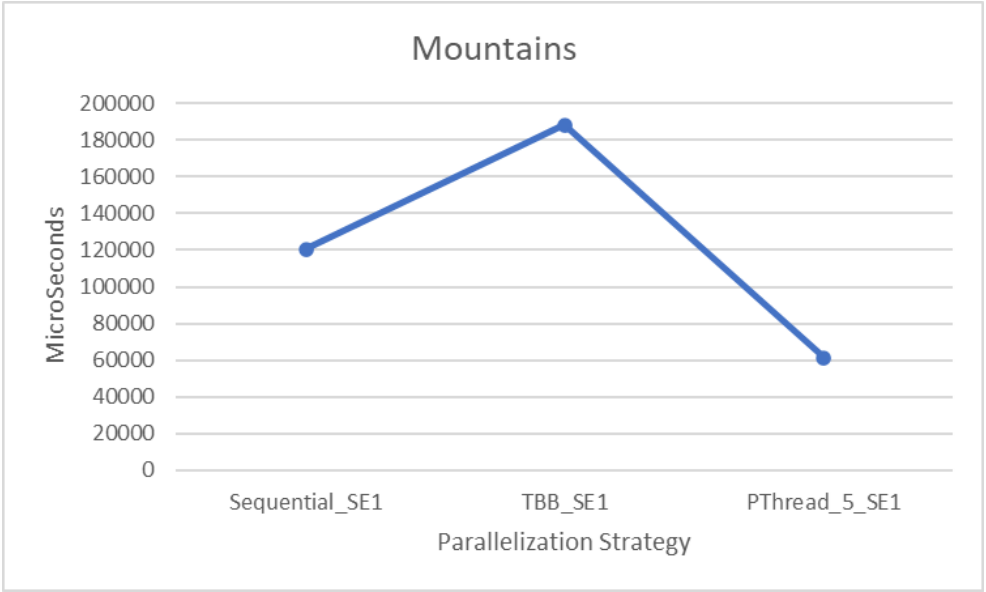
Parallelization Notes

- TBB
  - Nested Parallel_for operations were used for looping through pixels in the x and y direction
- Pthreads
  - Pthreads were launched and then the program waited for all threads to finish
  - Depending on the amount of threads requested each thread would perform x amount of columns of the image
- Some cases required more than one operation (ex. Opening - Dilation followed by erosion)



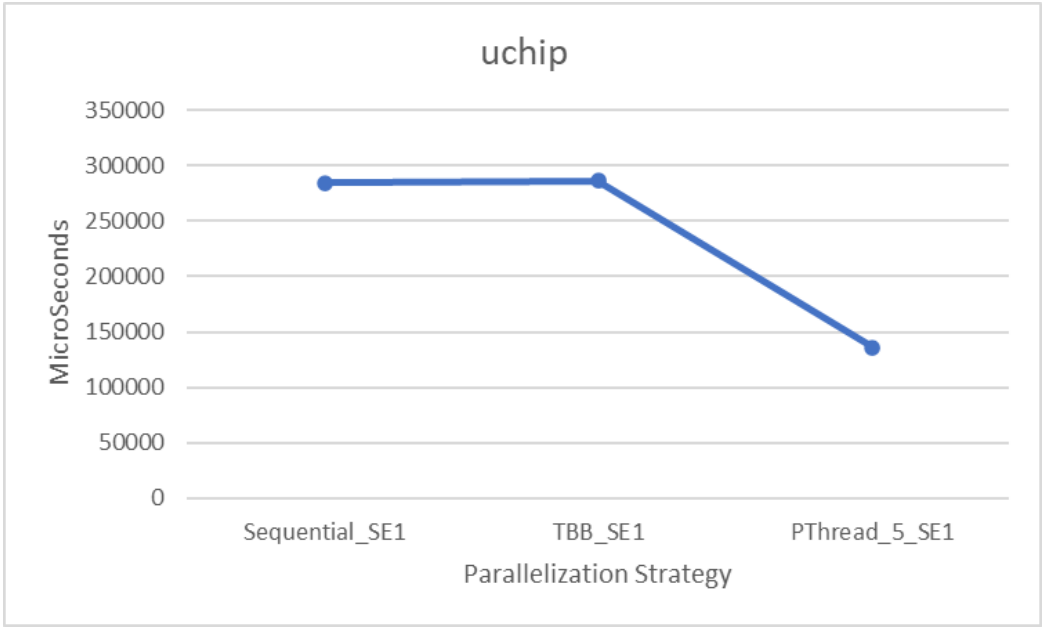
Parallelization Strategies Image 1

- In the case of mountains TBB took longer than performing a sequential operation
- This was the smallest image
- Image 1 - Mountains - 600 x 400
- Thread count of 5 was used for comparison



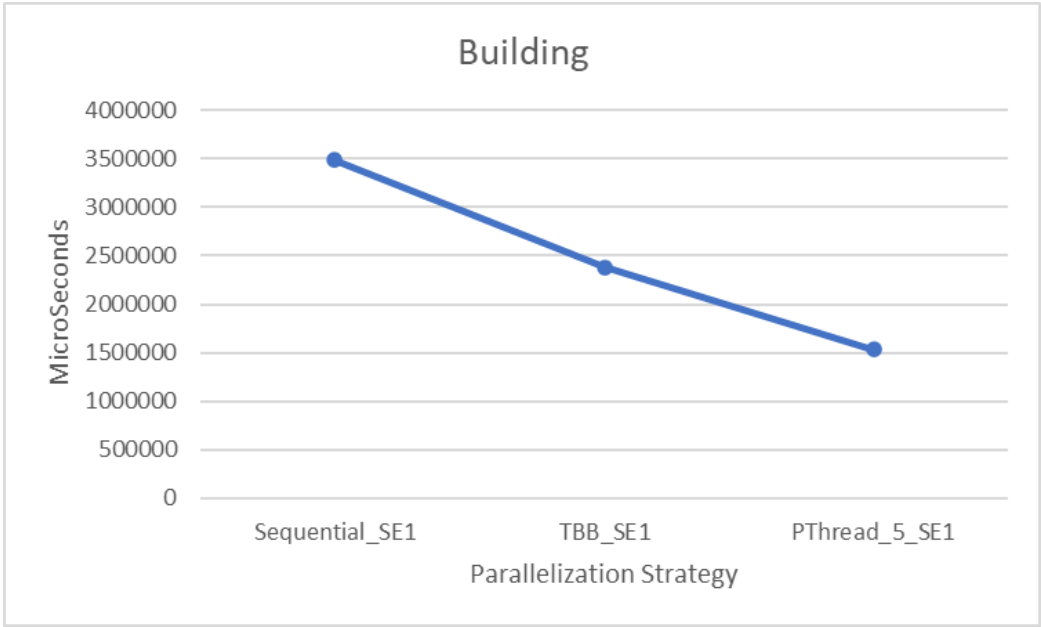
Parallelization Strategies Image 2

- Image 2 was slightly bigger than image 1 and we started to see TBB and sequential become about the same amount of time
- Image 2 - uchip - 940 x 602
- Thread count of 5 was used for comparison



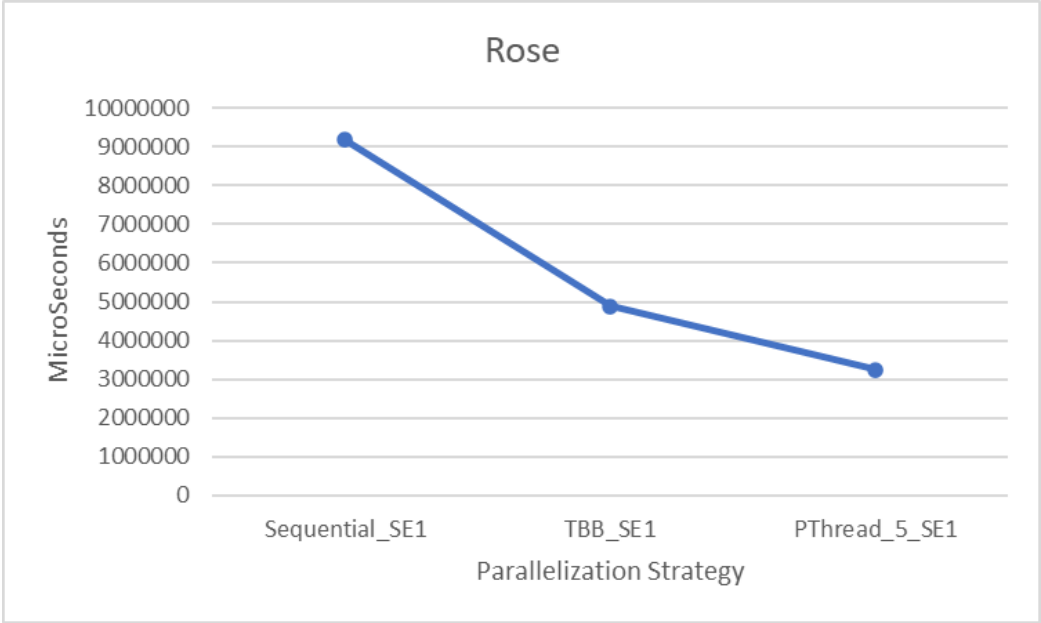
Parallelization Strategies Image 3

- Image 3 was much larger than the other two images and we can start to see much larger gains over the sequential approach
- Image 3 - Buildings - 3472 x 2315
- Thread count of 5 was used for comparison



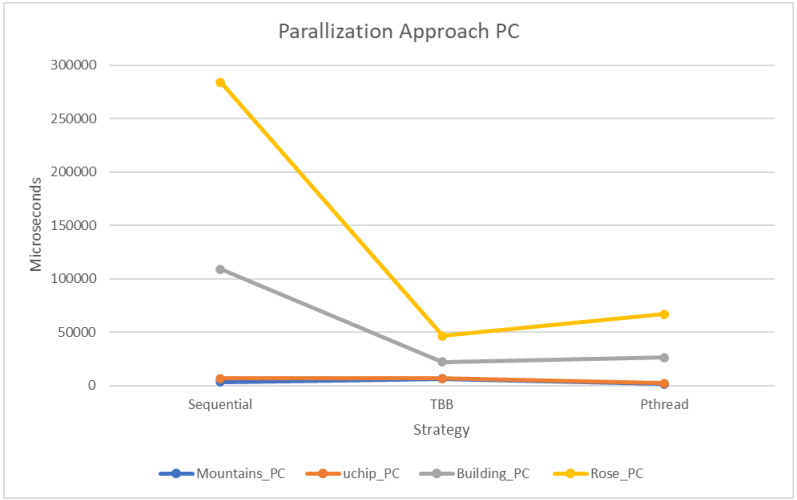
Parallelization Strategies Image 4

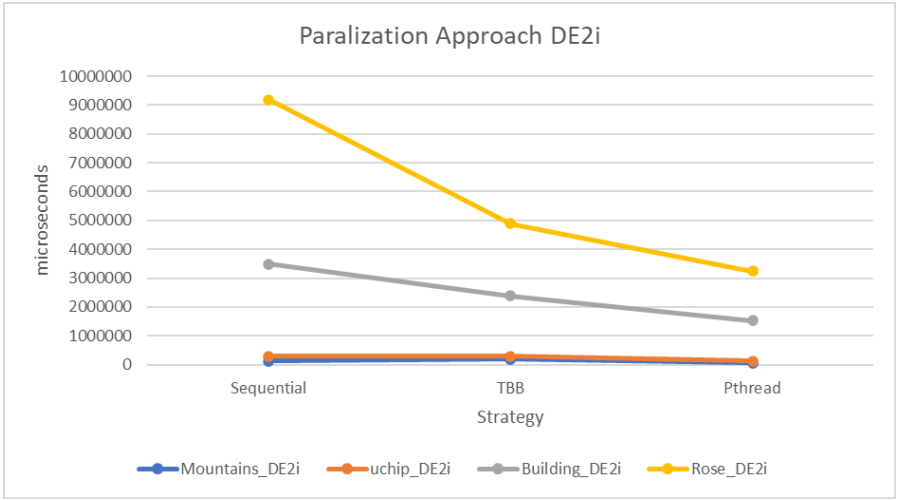
- Image 4 was the largest image and we see similar results of a large gain from TBB
- Image 4 - Rose - 5168 x 4000
- Thread count of 5 was used for comparison



Parallelization Strategies Based on Device

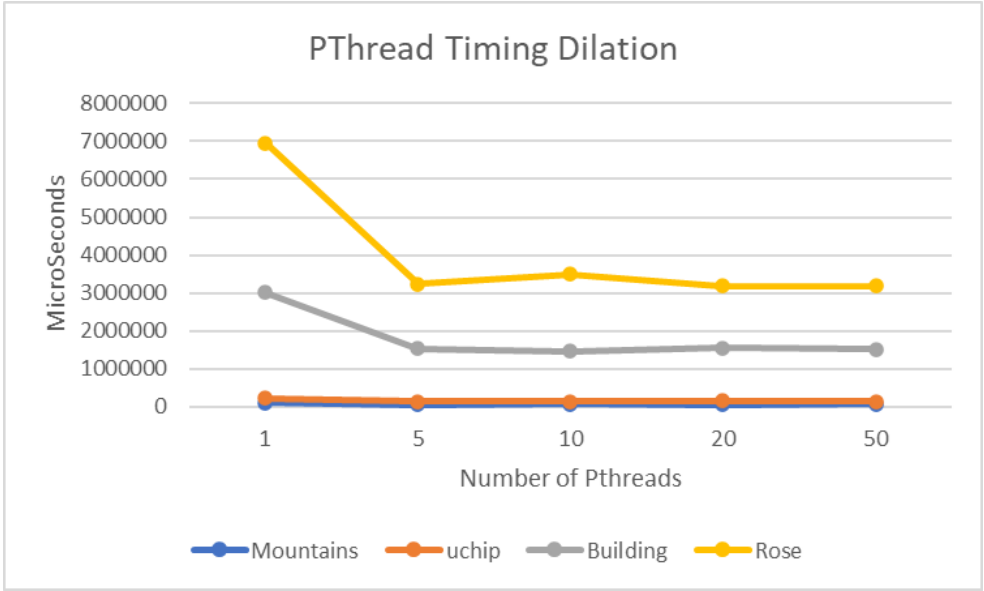
- Device Comparison for various parallelization methods
- DE2i-150 vs AMD Ryzen 7 3700X





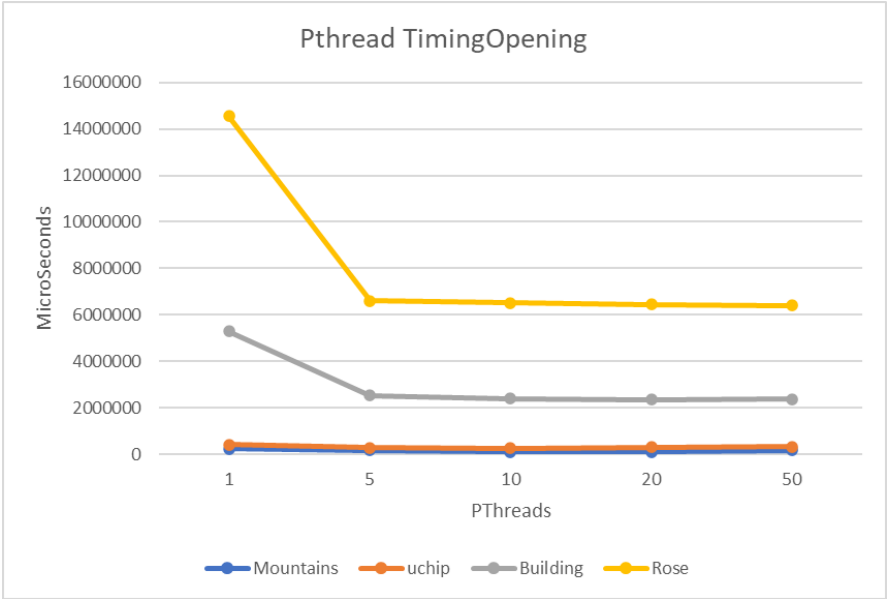
Timing based on Pixel Count

- 4 images were used
  - Image 1 - Mountains - 600 x 400
  - Image 2 - uchip - 940 x 602
  - Image 3 - Buildings - 3472 x 2315
  - Image 4 - Rose - 5168 x 4000



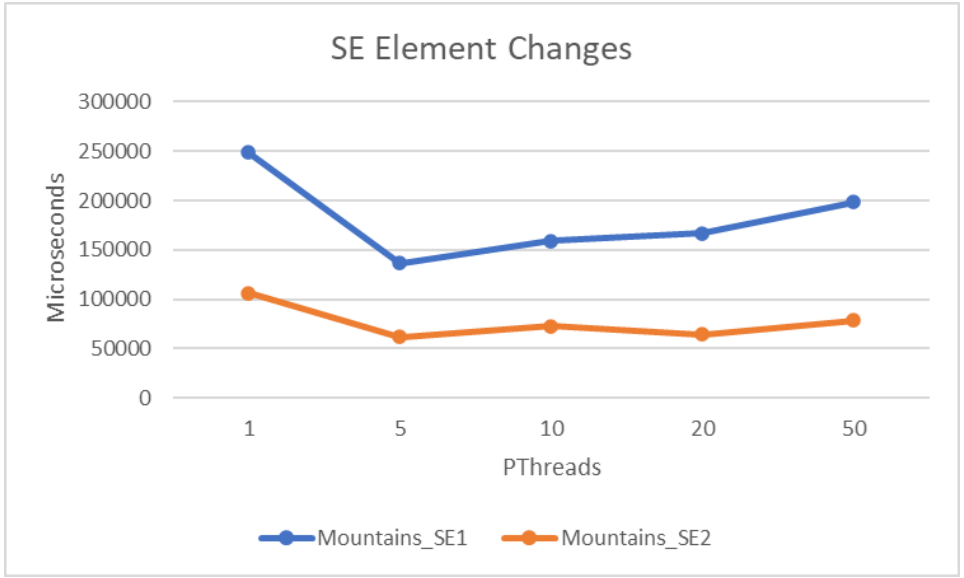
Timing based on Pixel Count

- 4 images were used
  - Image 1 - Mountains - 600 x 400
  - Image 2 - uchip - 940 x 602
  - Image 3 - Buildings - 3472 x 2315
  - Image 4 - Rose - 5168 x 4000



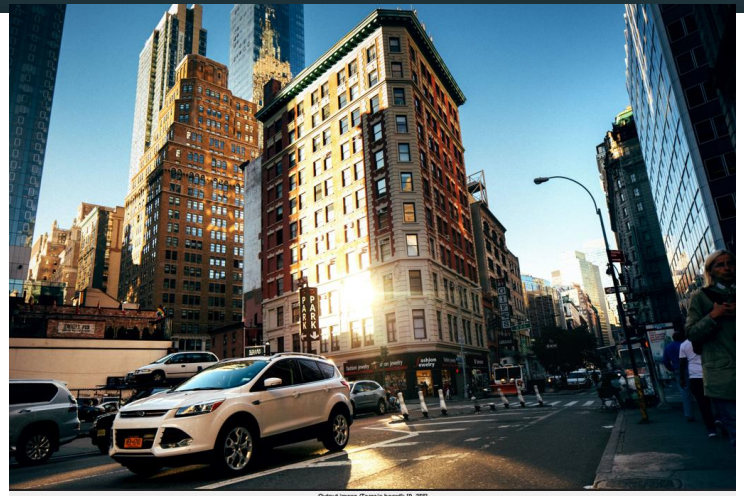
Comparing Structural Element Changes(SE)

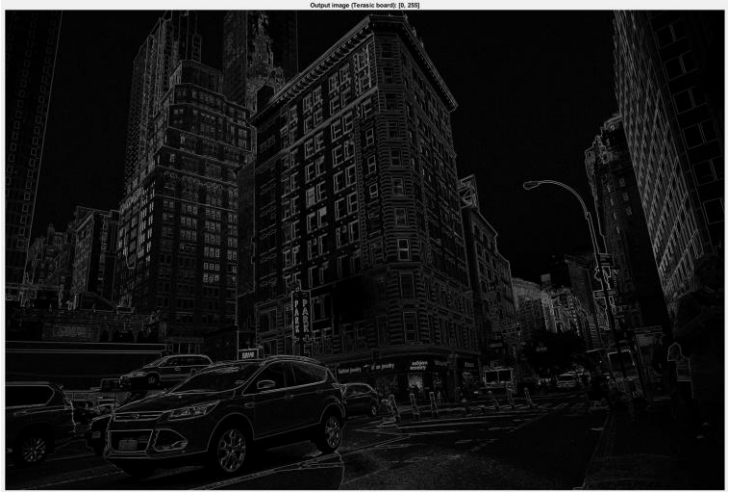
- When changing the structural element to a disk of size 2 to a disk of size 1 we see an increase in execution time.



Matlab Check

- Matlab is used in order to validate the data. A difference compare is used to determine if any pixels are different from the matlab implementation and the code implementation
- Matlab is also used to create the raster scan grayscale image from a jpg image





Demo

```
robert@robert-Cedar-Trail-Client-platform:~$ ./morpho 6 4 2 50
(read_binfile) Input binary file 'uchip.bif': # of elements read = 20672000
(read_binfile) Size of each element: 1 bytes
(read_binfile) Input binary file 'uchip.bif': # of elements read = 20672000
(read_binfile) Size of each element: 1 bytes
(write_binfile) Output binary file 'Outputtbb.bof': # of elements written = 2067
2000
(read_binfile) Size of each element: 1 bytes
(write_binfile) Output binary file 'Outputp.bof': # of elements written = 206720
00
(read_binfile) Size of each element: 1 bytes
Sequential Approach:
        start: 570433 us        end: 111196 us
        Elapsed time (actual computation): 21540763 us
TBB Approach:
        start: 372095 us        end: 536789 us
        Elapsed time (actual computation): 11164694 us
Pthread Approach:
        start: 536790 us        end: 609965 us
        Elapsed time (actual computation): 8073175 us
robert@robert-Cedar-Trail-Client-platform:~$
```

Conclusions

- On the pc where more resources were available I was able to see larger gains from the TBB algorithms
- Larger pixel counts led to higher gains in parallelization
- Increasing the SE to a higher value increased execution time due to needing to do more searching for the min/max pixels. This was not impacted greatly by parallelization strategies as each thread had to loop through additional kernel indexes.