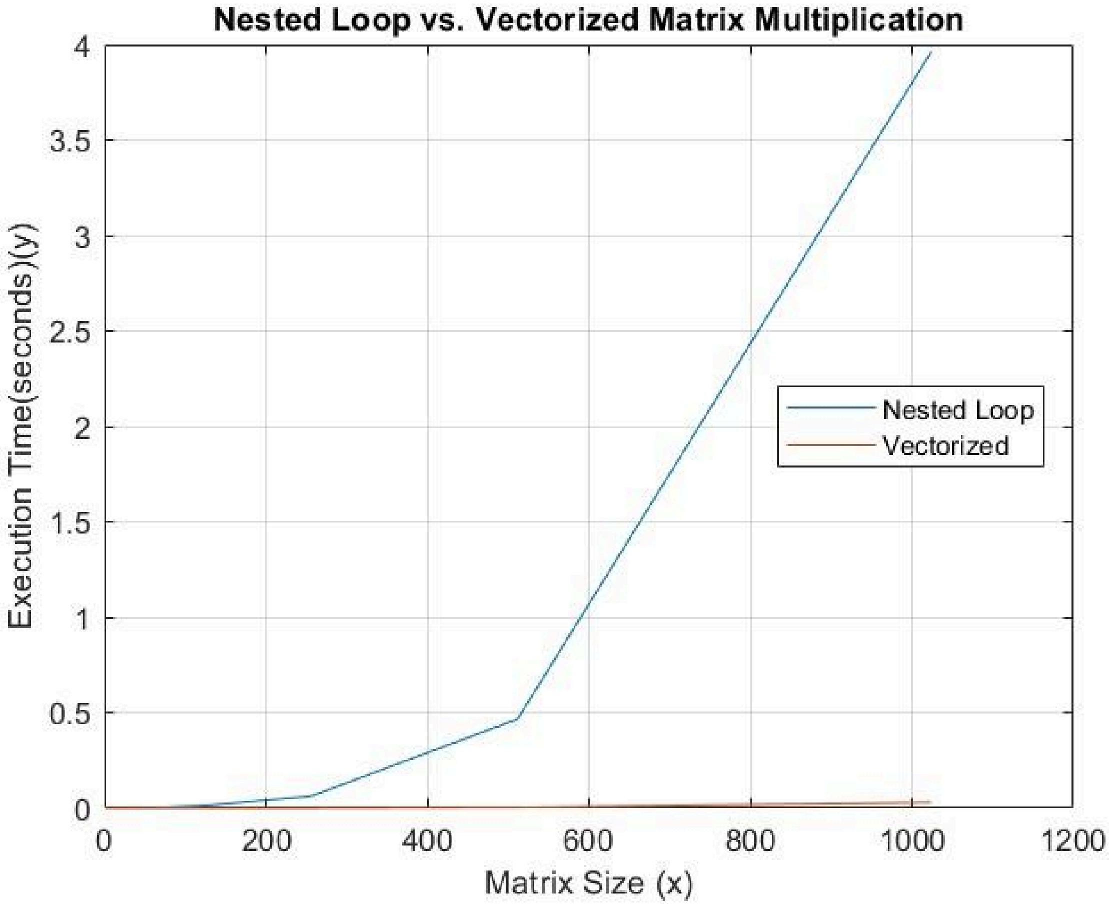
Homework 1

Problem 1:

Algorithm step

```
1. Size[] ← size of dimension.
   loopTime[] ← execute time by using nest loop for matrix multiplication.
   vectorTime[] ← excite time by vectorization method.
2. i ← 1
3. Size[] ← append 2^i
4. Matrix_1 ← random matrix with dimension of 2^i by 2^i
   Matrix_2 ← random matrix with dimension of 2^i by 2^i
5. loopTime[] ← append the execute time from function loopMatrix(Matrix_1, Matrix_2)
           loopMatrix(M1, M2){
                   For i to rowSize;
                           For j to colSize;
                                   Algebra compute the result
                   end
           }
6. vectorTime[] ← append the execute time of vectorization operation Matrix_1*Matrix_2
7. i++;
8. Repeat step 3 to step 6 until i >10
9. Using data size[] as X-axis control variable, and loopTime[] and vectorTime[] as Y-axis
   variable to construct plot graph.
```

Plot Graph



Result

This experiment is using a square matrix for convenience purposes. Dimensions of the matrix used are 2, 4, 8, 16, 32, 64, 128, 256, 512, and 1024. A higher size was attempted by failure due to hardware and computability issues. Observed from the output data(graph) above, the control variable in this experiment is the size of the matrix. As the size of the matrix increases, the difference of the execution time(Y-axis) between the nested loop method and the vectorized method is getting bigger. A proper algorithm or method is significantly important in image processing, especially processing high resolution images.

Resources that Helped Me

- https://www.mathworks.com/help/matlab/ref/mtimes.html
  openExample('matlab/MultiplyTwoArraysExample')
- https://www.mathworks.com/matlabcentral/answers/1550-execution-time#answer_2300
- https://www.mathworks.com/help/matlab/creating_plots/combine-multiple-plots.html

Problem 2:

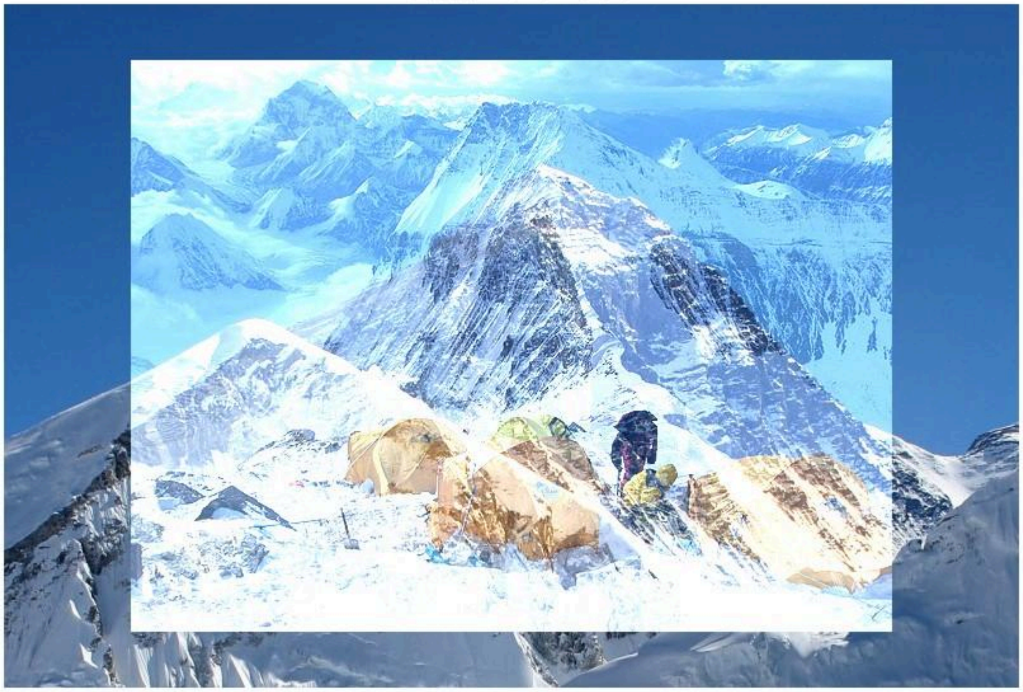
Algorithm step

1. Image1 ← read given image `Everest_expedition.jpg`
   Image2 ← read given image `Everest_kalapatthar.jpg`
2. Call function imagePad(Image1, Image2)
   - imagePad(img1, img2){
     1) max_greyValue ←255;
     2) Size1, size2, maxSize ← get size of img1, img2 and max height and weight from the two image.
     3) Scale_1 ← [(maxHeight - img1 Height), (maxWeight - img1 Weight)]
        Scale_2 ← [(maxHeight - img2 Height), (maxWeight - img2 Weight)]
        compute the difference between max height and weight size with the two images. (for image copy starting point)
     4) padImg_1, padImg_2 ← create max height by max weight new matrix and pad with 0.
     5) Copying img1 and img2 to padImg_1 and padImg_2, also using Scale_1 and Scale_2 to determine the copy process starting position which centers the padded position.
        Ex: padImg_1's startRow ← 1+scale_1(Height) to img1_Height+scale_1(Height)
     6) Output1 ← padImg_1 + padImg_2 (combine two pad images)
     7) Output1 ← min(output1, max_greyValue), (ensure all pixel value under the limit max_greyValue value)
     8) Display output1;
     9) Reinitialize padImg_1 and padImg_2
     10) Copying img1 and img2 to padImg_1 and padImg_2 from top-left corner.
     11) Output2 ← imadd(padImg_1, padImg2) //(immad(img1, img2) == padImg_1 + padImg_2, combine two pad image as step 6 and step 7)
     12) Display output2;
   - } //end imagePad function
3. Call function imageFuse(image1, image2)
   - imageFuse(img1, img2){
     1) Output ← imfuse(img1, img2, 'blend') //build in function that joint two image different than padding(imadd()).
     2) Display output;
   - } //end imageFuse function

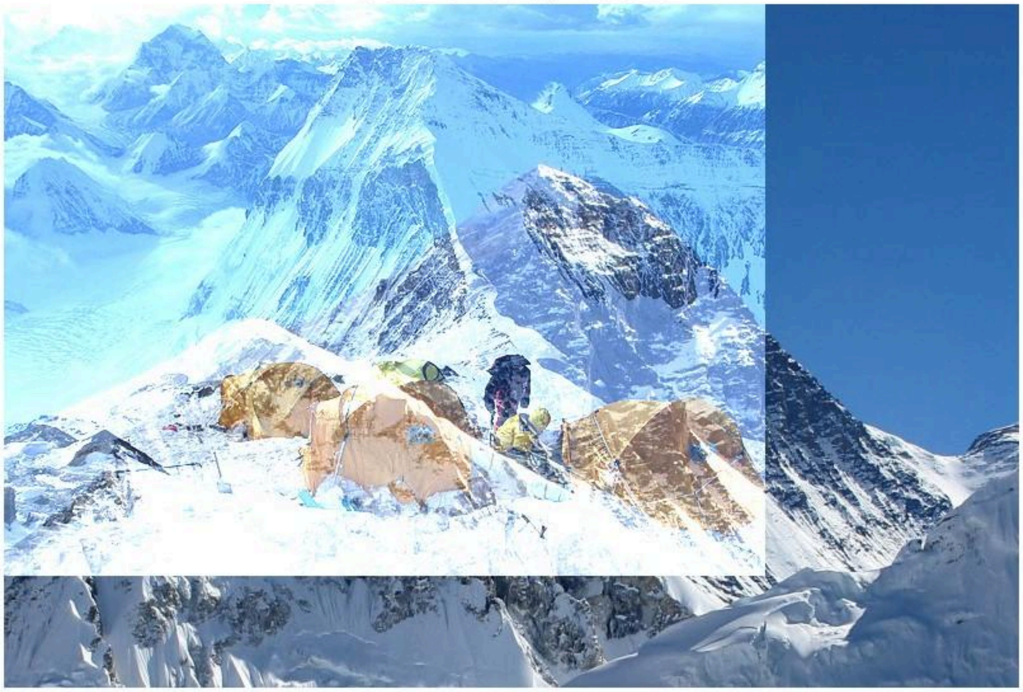
Input Images

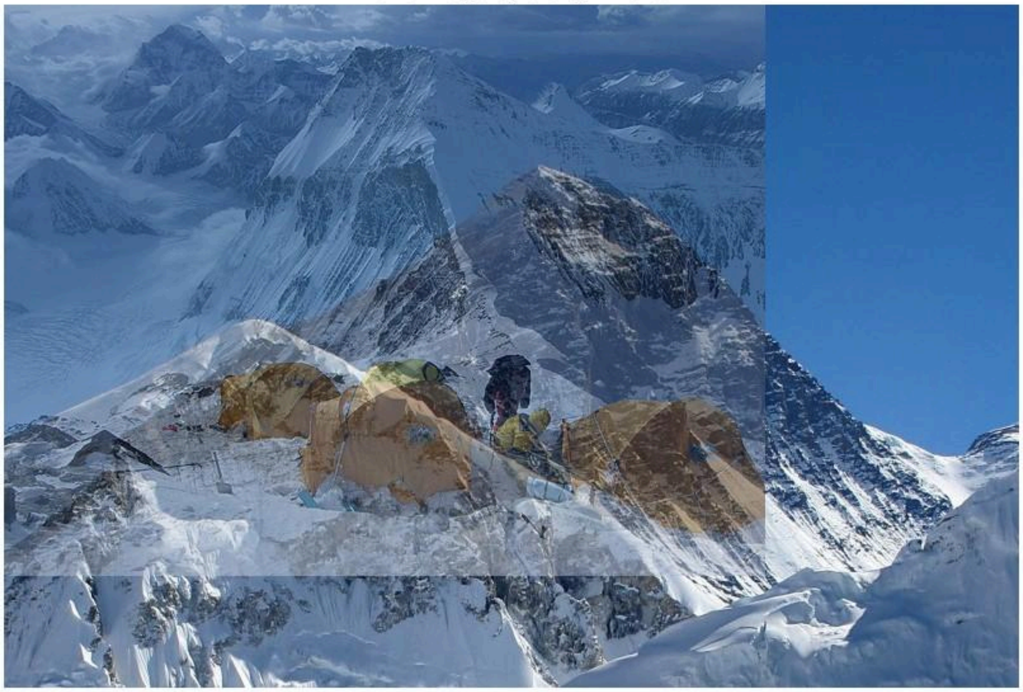
Output Images

Padded in the middle



Padded from the top-left corner





Blend image by using imfuse

Result

At the beginning of the homework, I first found out the function imfuse() to combine two images. However, the outcome is not what I expected for this assignment. The imfuse() method blends the images together, instead of adding the two corresponding images' pixel gray value together. Observing the blend output image above, clearly can distinguish that the combine area does not show any increase in exposure. So, then using an alternative method of padding. And, in my opinion, the outcome is success since the combined area has brighter color. I had padded the images in different positions to check if all combined areas are having higher exposure(more intensive brighter color) than other areas. Controlling the variable max_greyValue in my code can also adjust the overall intensive combined image.

Resources that Helped Me

https://www.mathworks.com/matlabcentral/answers/373174-what-does-imfuse-exactly-do-and-how-does-it-differ-from-imadd#answer_296480
https://www.mathworks.com/help/images/ref/imfuse.html#bta30pd-1-ColorChannels
https://www.mathworks.com/help/images/ref/imadd.html#f4-334800_seealso
https://www.mathworks.com/matlabcentral/answers/469306-zero-pad-to-resize-an-image#answer_381148
https://www.mathworks.com/matlabcentral/answers/80295-how-to-copy-one-image-to-another-blank-pixel-by-pixel