Image Zooming In and Out

Name: Kevin McGinley
ID: 14372681
Email: k.mcginley4@nuigalway.ie

Description:

Goal:
Zoom grey-scale PBM images in and out.

How:
To zoom an image in, it is required to transform the image to a scaled up version. This is achieved by mapping pixels of original image to each pixel of transformed image.

To zoom an image out, it is required to transform the image to a scaled down version. This is achieved by discarding the required number of rows and columns of the original image. Pattern of rows and columns to be discarded can be determined for a scaling factor int or floating.
Although the algorithm zoom In(floating) below could take integer values, the resultant dimensions are very inaccurate for certain odd integers, such as 3,5 and 9.

Zoom Out:

Goal: Scale matrix up by factor by using nearest neighbour interpolation
How: Loops through each entry of new transformMatrix N*M. Left,right,top and bottom defined by transforming from transformMatrix to original image. For example: Left is transformMatrix j index transformed to original image (divided by factor), rounded down. Top is transformMatrix i index transformed to original image (divided by factor), rounded up.
The average of the 4, not necessarily unique, points is taken as new entry value for transformMatrix at i,j.
Efficiency: Linear with respect to pixels transformed to.

Zoom In(floating):

Goal: Scale matrix down by floating point factor by discarding rows and columns.
How: Finds rows and cols to keep. Nth Row or column is discarded if N equal to a multiple of factor plus or minus (factor - 1)/2, otherwise that row or column is stored into an array. A matrix of the transformed row number and original column number is constructed by setting entries from original matrix and discarding rows not in our row array. Then a matrix of transformed row and column number is constructed by setting entries from matrix of transformed row number and discarding columns not in our column array. The final matrix is the desired transformed image
Efficiency: Linear with respect to pixels transformed to. Similar efficiency to Zoom In(integer).

Zoom In(integer):

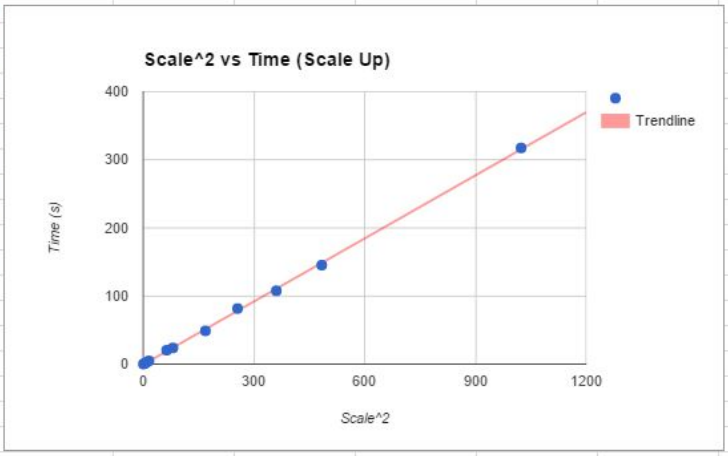
Goal: Scale matrix down by an integer factor by discarding rows and columns.
How: Finds rows and cols to keep. Nth row or column is stored into an array, if N modulo factor = 0. Transformed matrix is constructed similarily to Zoom In(float).
Efficiency: Linear with respect to pixels transformed to. Similar efficiency to Zoom In(floating).

Efficiency:

Scaled image of 960*720 by various scale factors to test the efficiency of the three algorithm described above. All algorithms are of linear complexity as shown below. Integer and floating scale down algorithms were of similar efficiency.
Scale Down Integer's slope: 0.155
Scale Down Floating's slope: 0.143

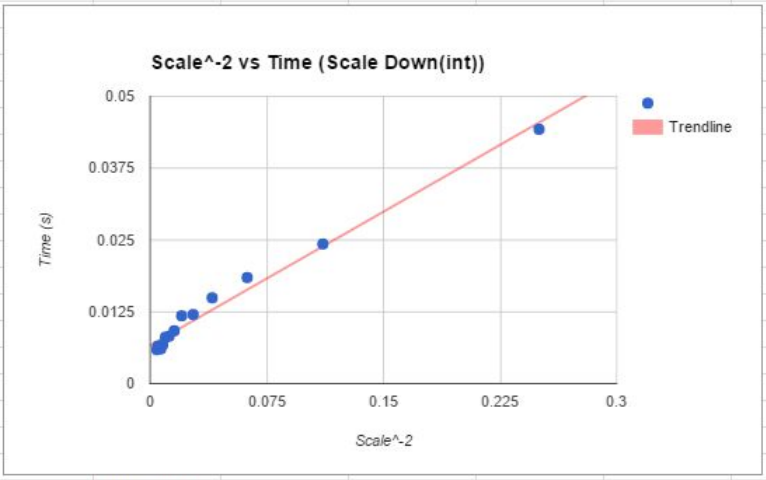
Scale Up

	A	B	C
1	Scale Up	Scale^2	Time(seconds)
2	1	1	0.30294
3	2	4	1.19257
4	4	16	5.0259
5	8	64	20.6693
6	16	256	81.793
7	32	1024	317.386
8	3	9	2.5795
9	9	81	24.04
10	13	169	49.057
11	19	361	107.851
12	22	484	145.414



Scale Down Integer

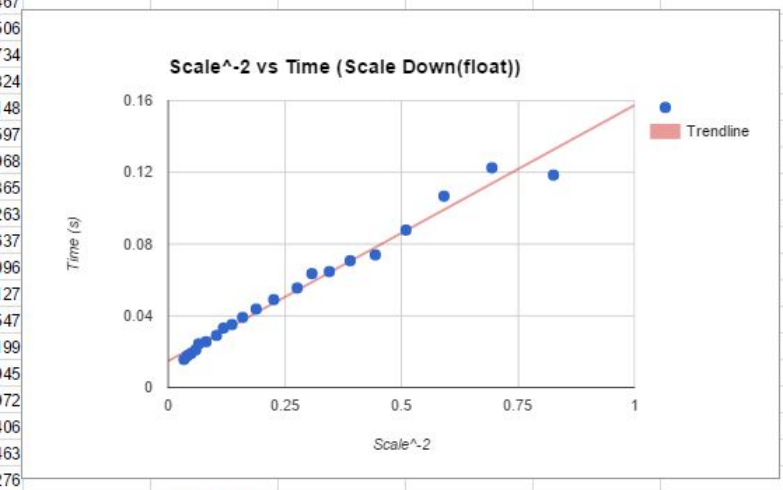
Scale Down	Scale^-2	Time(seconds)
2	0.25	0.0442522
3	0.1111111111	0.0243062
4	0.0625	0.0184751
5	0.04	0.0149477
6	0.02777777778	0.0120355
7	0.02040816327	0.0118209
8	0.015625	0.00921797
9	0.01234567901	0.00827036
10	0.01	0.00810656
11	0.00826446281	0.00679253
12	0.006944444444	0.00606305
13	0.005917159763	0.0064485
14	0.005102040816	0.00653874
15	0.004444444444	0.00593385



slope = 0.155

Scale Down Floating

Scale Down	Scale^-2	Time(seconds)
1.1	0.826446281	0.118467
1.2	0.6944444444	0.122506
1.3	0.5917159763	0.106734
1.4	0.5102040816	0.087824
1.5	0.4444444444	0.0740148
1.6	0.390625	0.0707597
1.7	0.3460207612	0.0646968
1.8	0.3086419753	0.0635365
1.9	0.2770083102	0.0555263
2.1	0.2267573696	0.0490637
2.3	0.1890359168	0.0438096
2.5	0.16	0.039127
2.7	0.1371742112	0.0351547
2.9	0.1189060642	0.033199
3.1	0.1040582726	0.0290945
3.5	0.08163265306	0.0255972
3.9	0.06574621959	0.024406
4.1	0.05948839976	0.0210463
4.5	0.04938271605	0.0190276
4.9	0.04164931279	0.0177587
5.4	0.03429355281	0.0157256



Slope = 0.143

Scale Down and Up:

Scaled image 960*720 image by factor 3/10 and then scaled by 10/3 to original image dimensions. Resultant image clearly has much less information than original image of same dimensions, as expected.

Original:

Scaled Down:

Scaled Back Up: