CMSC 420
Project 2

Your second programming project involves building a multiresolution stereo program for a pair of calibrated stereo images. You will assume that the images are in "standard" configuration so that the rows of the images form the epipolar lines of the stereo pair. The steps involved in your project are as follows:

1. Create an image pyramid
2. For each level of the pyramid
   - Compute the disparity map for that level
   - Expand it to form the input disparity map for the next level
3. Compare your disparity map with the ground truth

1. Creation of the image pyramid.

The original image pair will be 256 x 256 gray level images. Create the next two levels of the pyramid, creating 128 x 128 and 64 x 64 images by replacing each non-overlapping block of 2 x 2 pixels with its average gray level.

2. Disparity map creation

Creating disparity maps is done in two steps for each level of the pyramid:

1. Creation of the Match-score matrix (MSM)
2. Creation of the output disparity map (D_{out}) through the application of a dynamic programming algorithm to the MSM.

The output disparity maps are created for each row of the stereo pair independently, so in the discussion that follows we will consider what is involved in matching a row of the left image against the corresponding row of the right image. Also, we will let $D_{in}(j)$ be the input estimate of disparity for the pixel in the j'th column of the row of the left image we are considering. The pyramid has 3 levels, and we will refer to the 64 x 64 level as level 0, the 128 x 128 level as level 1 and the 256 x 256 level as level 2. For level 0, $D_{in}(j) = 0$ for all j. Let NC_{level} be the number of columns in a row for the image at pyramid level "level."

The MSM is array of integers that has one column for each pixel in the left image row, and as many rows (+1 – see below) as possible disparities it computes. Generally, for each pixel in the left image, we will compute match scores for 2k+1 pixels in the right image, for some value of k. So, if k = 3, we will compute 7 match scores for each pixel in the left image, and for level 0 the MSM matrix would be a 7 x 64 matrix. The value of k need not be the same for each level of the pyramid, so in the algorithm to follow we will let k_{le} be the value of k used when processing level le of the pyramid. The match score we will use is the sum of absolute differences computed over a fixed size r/2 x r/2 neighborhood. This is computed as follows:

```
Integer function SAD(Left, Right, row, j, j', r);
\* Left and Right are the 2D left and right images; row is the index of the row we are
currently comparing, j the index of a pixel in the left image row and j' the index of a pixel
in the right image row we want to compare to j; r is the radius of the neighborhood used
to compute the SAD. The algorithm given here does NOT check boundary conditions,
which your implementation, of course, must. *\

SAD := 0
for $\Delta i$ = -r, r
        for $\Delta j$ = -r, r
        SAD := SAD + |Left(row- $\Delta i$, j - $\Delta j$) – Right(row-$\Delta i$, j'-$\Delta j$)|
        endfor
endfor
Return SAD
```

The SAD function is used to compute the match matrix, MSM. Basically, for each pixel, j, in the left image, we compare its rxr neighborhood to the neighborhoods of pixels in an interval in the right image centered at j+D_{in}(j). If any of the pixels in that interval would result in negative disparity, then we set the MSM value to maxint. The algorithm is as follows:

```
Compute-MSM(Left, Right, r, $D_{in}$, MSM)
Initial MSM to 0.
For j = r to $NC_{level}$ – r   \* These are all the pixels whose 2r+1x2r+1 neighborhoods are
                                      contained in the image *\
        For $\Delta j$ = -k, k
        j' :=  j + $D_{in}$[j] + $\Delta j$   \* One point in the matching interval in the right image *\
        if j' < j then MSM[$\Delta j$,j] := Maxint else MSM[$\Delta j$, j] := SAD(Left, Right, row, j, j',
r)
endfor
```

Now, when processing level 0 our initial estimate of disparity (array D_{in}) is uniformly 0. In this case, in order not to have to create a special case for level 0, we can let k = maxdis/4, where maxdis is the maximum disparity we expect in the original stereo pair. This results in some wasted space in the MSM array, since we know the first maxdis/4 rows of each column will receive the Maxint value, since they will all correspond to prohibited negative disparity values. However, the advantage is that you can use the same code for level 0 as for the other levels.

Now, the MSM array is the input to the dynamic programming algorithm. The dynamic programming algorithm constructs a function, D_{out}, mapping pixels in the left row to the set {0, 1, …, maxdis/level, maxdis/level + 1}, where 0, 1, … maxdis/level are the possible disparity values for pixels at the current level, and maxdis/level + 1 indicates that the dynamic programming algorithm chooses to assign **no match** to a given pixel (and we pre fill this row of the MSM matrix with a penalty score associated with the no-match condition). We associate the following total cost with any such function D_p:

$$Cost(D_p) = \sum_{j=r}^{NC_{level}-r} MCM(D_p(j) - D_{in}(j), j)$$

Your dynamic programming algorithm solves for the D_{out} that has minimal cost over ALL allowable D_p. D_p is allowable if j < j' implies $j + D_p(j) < j' + D_p(j')$. This, essentially, enforces the left to right ordering constraint of the stereo correspondence algorithm.

In order to use D_{out} as the input disparity map for the next level of the pyramid we must

a) fill in the gaps for the no-match points, and
b) double its resolution so that we have one disparity estimate per pixel.

This can be simply done by first copying the elements of D_{out} into an array twice its length:

```
For j = 1, $NC_{level}$
$D_{in}$[2j] := $D_{out}${[j]
```

This will introduce lots of 1 pixel gaps into D_{in} as well as double the sizes of the gaps the dynamic programming algorithm left in $D_{out.}$ To fill in those gaps, simply linearly interpolate between the end points of the gap. That is, if we see the gap:

10 ? ? 20

We split the 10 disparity difference and obtain

10 13 17 20

after rounding.

3. Comparison to Ground Truth

After you run all three stages of the matching algorithm you will have a disparity map for all pixels in the original high resolution left image (except for pixels in the first and last few rows and columns). The TA will be providing "ground truth" disparity maps for the stereo pairs that you will be testing your algorithm on, and you need to decide how you are going to perform this comparison. So, let D_{out}[i,j] be the disparity values that your program has computed, and D_{truth}[i,j] be the true values. The simplest measure of accuracy is to look at some measure of average error, such as

$$Error = (1/NC * NR)\sum_{i=1}^{NR}\sum_{j=1}^{NC}(D_{out}(i,j) - D_{truth}(i,j))^2$$

However, this will be misleading, since your stereo program will do VERY poorly near disparity discontinuities – i.e., edges in the image between nearby objects and the more distant background – and these sparse but very large errors will tend to make the average error very large. Instead, you should construct a histogram of **relative** disparity errors. That is, create the image

$$\left|\frac{D_{out}(i,j) - D_{truth}(i,j)}{D_{truth}(i,j)}\right|$$

and then display the histogram of this image. From this histogram you could determine, for example, what percentage of the time your estimate was within 10% of the true disparity. There are many other possibilities – e.g., creating a two dimensional histogram of relative error versus local disparity gradient. As part of your report, you must develop a method of comparing your output disparity against the ground truth, and explain why you think your method is good!