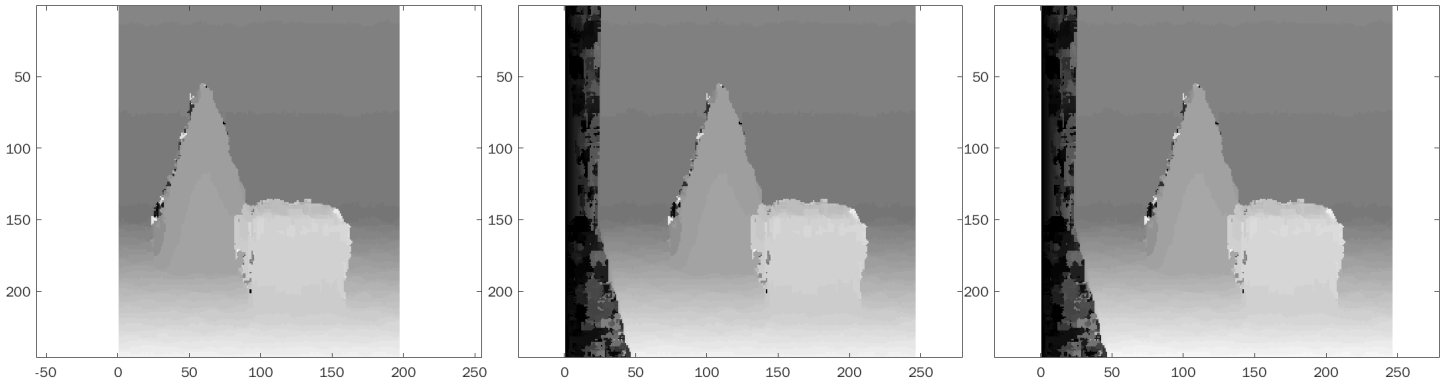
1) Compute a disparity map for the images left.png and right.png (having parallel optical axes) using the basic stereo matching algorithm. Use the NCC function to perform the template matching for each patch in the left image searching in the right image (search only leftward from – and including! – the starting point along each row!), and use a window size of 11x11 pixels. To make things run a bit faster for the grader, when searching leftward, only move up to 50 pixels to the left (instead of going all the way to the edge of the image). Use the following Matlab code (or Python equivalent) to display the disparity map D with a gray colormap and clip the disparity values at 50 pixels, making sure to display the full range of remaining values (e.g., using Matlab's imagesc function): [5 pts]



The first image is a clipped image that shows only the X-axis between 50 and 246 (11 pixels less than the original image due to the boundary). The range for each pixel is set to [0 50]. Therefore, if there is a value greater than 50, it will be set to 50. If there is a value less than 0, it will be set to 0.The second image is the full image, and the range of each pixel is set to [0 50].The third image is the full image, with every pixel full.And we can see, we have a nice parallax diagram, we can see the parallax clearly.I use NCC and only search left. Left-only searches help reduce the time complexity of the program.

2) As usual, turn in and upload your material

```
OriIm = double(imread('right.png'));
T = double(imread('left.png'));
```

```
[y0,x0] = size(OriIm);
[yT,xT] = size(T);
n = 11*11 -1;

for r = 1: (yT-(11-1))

    for c = 1:(xT - (11-1))
         Window(1:11,1:11) = T(r:r+(11-1),c:c+(11 -1));
         result_location = zeros(256,256);
         result_l = [];
         i = 1;
         if c >= 50
             for c2 = (c-49):c
                 P(1:11,1:11) = OriIm(r:r+(11-1),c2:c2+(11-1));
                 sum = 0;

                 meanP = mean(P,'all');
                 meanW = mean(Window,'all');
                 denominator = std(P, 0,'all') * std(Window, 0,'all');
                 if denominator == 0
                     denominator = 1;
                 end
                 for y = 1: 11
                     for x = 1: 11
                            %numerator = (P(y, x, z) - mean(P(:,:,z),'all'))
* (T(y, x, z) - mean(T(:,:,z),'all'));
                            numerator = (P(y, x) - meanP) * (Window(y, x) -
meanW);
                           % denominator = std(P(:,:,z), 1,'all') *
std(T(:,:,z), 1,'all');
                            sum = sum + (numerator/denominator);

                     end
                 end
                 result_location(r,c2) = sum/n;
                 result_l(i) = sum/n;
                 i = i+1;
             end

         else
             for c2 = 1:c
                 P(1:11,1:11) = OriIm(r:r+(11-1),c2:c2+(11-1));
                 sum = 0;

                 meanP = mean(P,'all');
                 meanW = mean(Window,'all');
                 denominator = std(P, 0,'all') * std(Window, 0,'all');
                 if denominator == 0
                     denominator = 1;
                 end
                 for y = 1: 11
                     for x = 1: 11
                            %numerator = (P(y, x, z) - mean(P(:,:,z),'all'))
* (T(y, x, z) - mean(T(:,:,z),'all'));
                            numerator = (P(y, x) - meanP) * (Window(y, x) -
meanW);
                           % denominator = std(P(:,:,z), 1,'all') *
std(T(:,:,z), 1,'all');
```

```
                        sum = sum + (numerator/denominator);

                    end
                end
                result_location(r,c2) = sum/n;
                result_l(i) = sum/n;
                i = i+1;
            end

        end
        sorted = sort(result_l, 'descend');
        [M,I] = max(result_l, [], 'all', 'linear');
        fr = 0;
        fc = 0;
        [fr,fc] = find(result_location == sorted(1));
        resule_c(r,c) = sorted(1);
        D(r,c) = abs(c -fc(1));



    end
end
imagesc(D, [0 50]);
axis equal;
colormap gray;
%pause;

imagesc(D(:,50:end), [0 50]);
axis equal;
colormap gray;
%pause;

imagesc(D);
axis equal;
colormap gray;
pause;
```