Assignment 05: Stereo Vision

1 Part 01: For Layercake images

- Main Octave Script: (`main_script.m`)
  First run this script.

```
clear
pkg load image

% Call the layercake function. It will generate a pair of stereo images
% (L: left-eye and R: Right-eye).
% 512 by 512 images (L and R) having 5 layers above the background.
[L,R] = layercake(512,512,5);

% Calling the stereo function. Here smin = 0, smax = 5, sigma = 3
D = stereo(L,R,0,5,3);

% Scaling pixel values of D to [0 to 255] range
min_value = min(D(:));
D = D - min_value;
max_value = max(D(:));
D = uint8((D./max_value) .* 255);

colormap(gray(256));
image(D+1); axis image;
%imwrite(D,'out_layercake.jpg');
```

- Octave Script for Stereo: (`stereo.m`)

```
function D = stereo(L,R,smin,smax,sig)
    % This function return the disparity map (D) of a pair of
    % stereo images (L and R). Here L is the left eye image,
    % R is the right eye image, smin and smax are the range
    % of disparity, and sig is the width of gaussian smoothing kernal.

    % for all possible disparities between smin to smax
    for s = smin:smax
        % M is the measure of similarity between L and shifted ...
            (columnwise) R.
        % Applying Gaussian smoothing kernal (Deriche implementation)
        M = deriche(abs(L.-shift(R,s,2)),sig);

        if s == smin % If first iteration
            D = ones(size(L)) .* smin;
            M_min = M;
        else
```

```
17                  pos = find(M < M_min);
18                  D(pos) = s; % update the disparity map (D)
19                  M_min(pos) = M(pos); % update the M_min
20              end
21          end
22      end
```

- Other Octave Functions:
  - `layercake.m` (Given with the assignment)
  - `deriche.m` (Given with the assignment)
- Input Images:

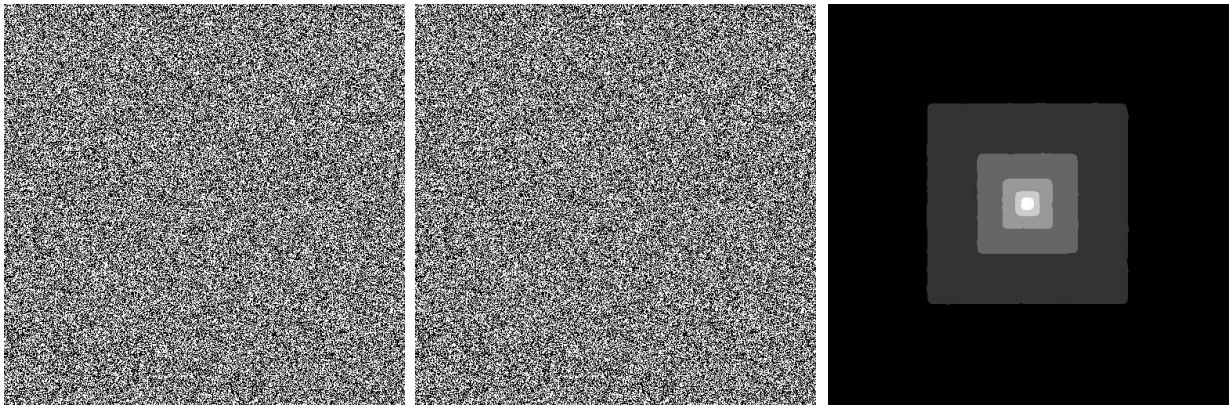

Figure 1: The input and the output images of layercake (having five layers). First column is the left eye image, second column in the right eye image and the third column is the output image(disparity map).

2 Part 02: For movie images

- Main Octave Script: (`main_script.m`)
  First run this script.

```
clear
pkg load image

input_img = imread('dino.pgm');
[h,w] = size(input_img);
L = input_img(:,1:floor(w/2)); % Cropping left image.
R = input_img(:,floor(w/2)+1:w); % Cropping right image.

% Calling the stereo function. Here smin = -5, smax = 5 and sigma = 3
D = stereo(double(L),double(R),-5,5,3);

% Scaling pixel values of D to [0 to 255] rnage
min_value = min(D(:));
D = D - min_value;
max_value = max(D(:));
D = uint8((D./max_value) .* 255);

colormap(gray(256));
image(D+1); axis image;
%imwrite(D,'out_dino.jpg');
```

- Other Octave Functions:
  - `deriche.m` (Given with the assignment)
  - `stereo.m` (Given with the assignment)

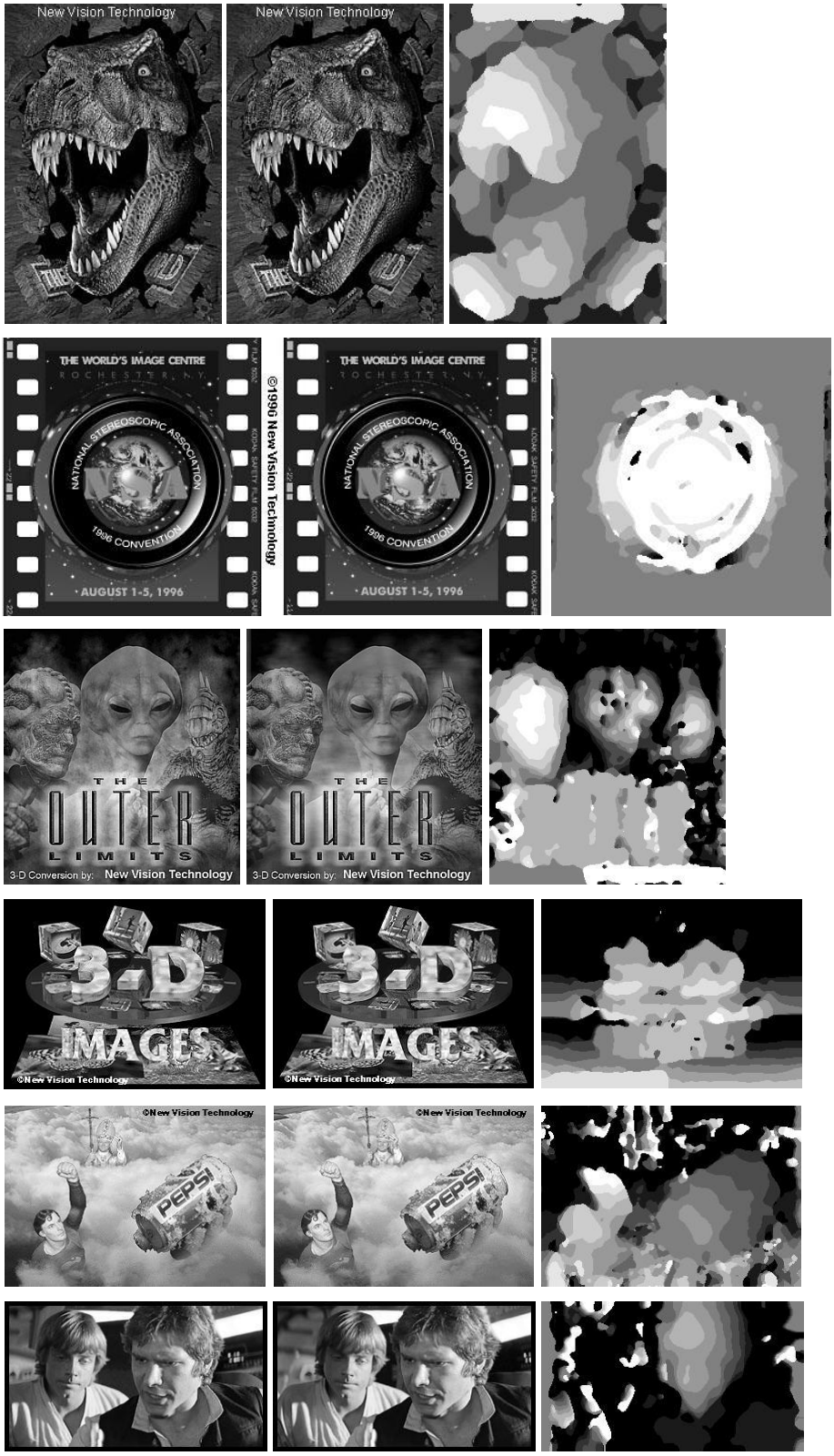

Figure 2: The input and the output images (a set of movie images). First column is the left eye image, second column in the right eye image and the third column is the output image(disparity map).

3 Part 03(Option 1): Stereo vision having pre-process with LOG filter

- Main Octave Script: (`main_script.m`)
  First run this script.

```
clear
pkg load image

input_img = imread('dino.pgm');
[h,w] = size(input_img);
L = input_img(:,1:floor(w/2)); % Cropping left image.
R = input_img(:,floor(w/2)+1:w); % Cropping right image.

% Pre-process the L and R images with LOG filter
LOG = fspecial('log',[3 3],3); % 3 by 3 window, sigma = 3
L = conv2(double(L),LOG,'same');
R = conv2(double(R),LOG,'same');

% Calling the stereo function. Here smin = -5, smax = 5 and sigma = 3
D = stereo(L,R,-5,5,3);

% Scaling pixel values of D to [0 to 255] range
min_value = min(D(:));
D = D - min_value;
max_value = max(D(:));
D = uint8((D./max_value) .* 255);

colormap(gray(256));
image(D+1); axis image;
%imwrite(D,'log_dino.jpg');
```

- Other Octave Functions:
  - `deriche.m` (Given with the assignment)
  - `stereo.m` (Given with the assignment)

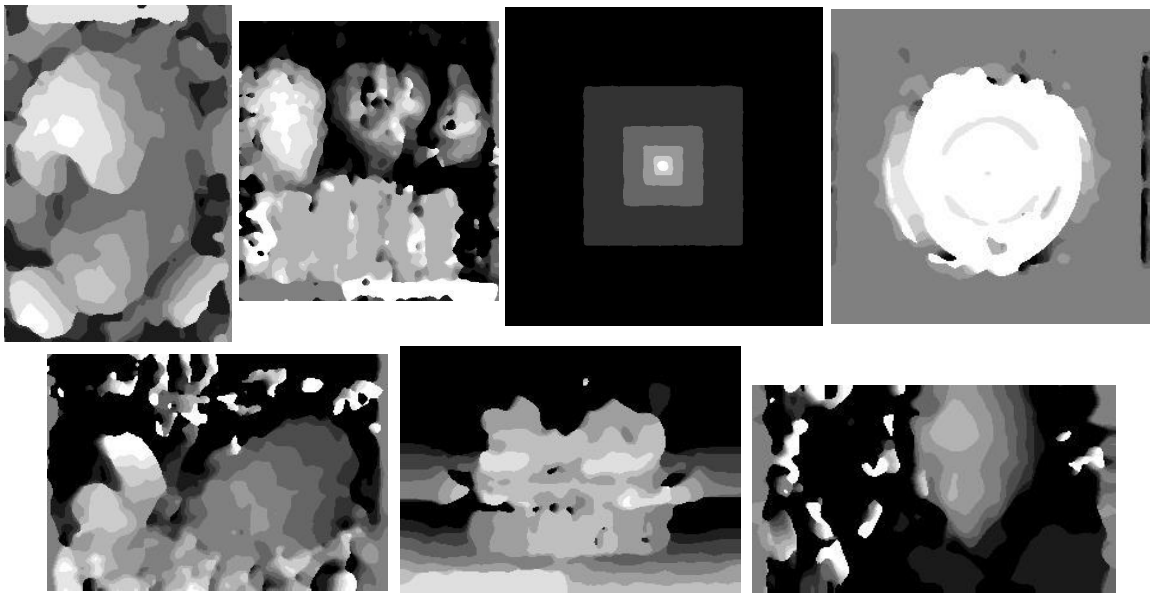

Figure 3: The output disparity images after applying LOG filter (Applied on the same set of input images given in Figure 1 and 2)

4 Part 04(Option 2): Verify Stereo vision (using forward and reverse disparity image)

- Main Octave Script: (`main_script.m`)
  First run this script.

```
1   clear
2   pkg load image
3
4   input_img = imread('dino.pgm');
5   [h,w] = size(input_img);
6   half_of_width = floor(w/2);
7   L = input_img(:,1:half_of_width); % Cropping left image.
8   R = input_img(:,half_of_width+1:w); % Cropping right image.
9
10  % Calling the stereo function. Here smin = -5, smax = 5 and sigm = 5
11  D_LR = stereo(double(L),double(R),-5,5,5); % Forward (left to right)
12  D_RL = stereo(double(R),double(L),-5,5,5); % Reverse (right to left)
13
14
15  % Verify using forward and reverse match.
16  % Generate a binary image where match is found.
17
18  % Creating row grid and column grid
19  row_grid = repmat([1:h]',1, half_of_width);
20  col_grid = repmat(1:half_of_width,h,1);
21
22  % Adding forward disparity with 'col_grid'
23  new_col_grid = col_grid .+ D_LR;
24
25  % Remove row,column values where column values are out of boundary
26  pos = find(new_col_grid < 1 | new_col_grid > half_of_width);
```

```
27      col_grid(pos) = [];
28      new_col_grid(pos) = [];
29      row_grid(pos) = [];
30
31      % Generate linear index
32      location_LR  = sub2ind([h half_of_width],row_grid,col_grid);
33      location_RL  = sub2ind([h half_of_width],row_grid,new_col_grid);
34
35      % Binary image
36      bin = false(h,half_of_width);
37      bin(location_LR) = D_LR(location_LR) == -1*D_RL(location_RL); % If ...
            match is valid
38
39      bin = uint8(bin) .* 255;
40      colormap(gray(256));
41      image(bin+1); axis image;
42      %imwrite(bin,'bin_dino.jpg');
```

- Other Octave Functions:
  - `deriche.m` (Given with the assignment)
  - `stereo.m` (Given with the assignment)

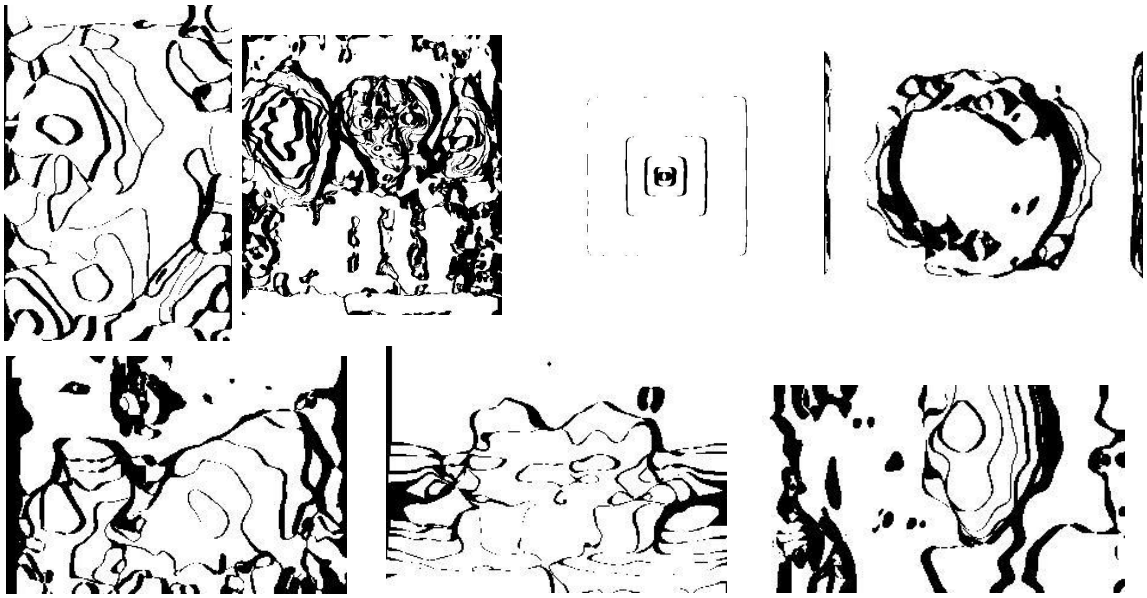

Figure 4: The output of verification image (binary image). Applied on the same set of input images given in Figure 1 and 2)

5 My Salient Observations:

- Observation 1: My observation is that disparity image gives approximate depth information of the objects in an image. The disparity is the inversely proportional to the depth of the objects. So the object nearer to the camera have high disparity (towards white) and objects further away from the camera have less disparity (towards black).
- Observation 2: In the layercake image of Figure 1, if I use sigma = 10, then the stereo algorithm is unable to show all five layers and also make smooth (round shape) the corners of each layer. The fifth (top) layer is missing in that case. Figure 5 is showing the case when sigma is 10. So high value of sigma is not a good choice.

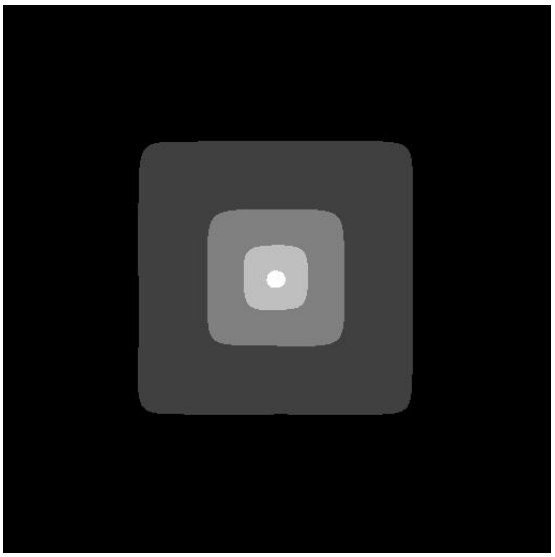

Figure 5: The disparity image for layercake image when sigma = 10. The output image have 4 visible layers (topmost is missing).

- Observation 3: In the layercake image of Figure 1, if the value of smax = 4, then the stereo algorithm is unable to show all five layers.The fifth (top) layer is missing in that case. Figure 6 is showing the case when smax is 4. This is due to the fact that the topmost layer have disparity of 5 pixels (according to function of layercake), so during stereo algorithm no match is found because of maximum shift amount is 4. So minimum value for smax should be 5 for the layercake image.
- Observation 4: In the verification process using forward and reverse disparity, the black pixels are showing the area where mismatch is happen. In Figure 7 we can see that there are some vertical black edges where mismatch happens. This is because in left eye's and right eye's view (or image), there should be some occluded area which must create mismatch during stereo algorithm.
- Observation 5: In Figure 7, now we can see that in the layercake image there is 01 pixel horizontal disparity in the first layer, 02 pixels horizontal disparity in the second layer, 03 pixels horizontal disparity in the third layer and so on.

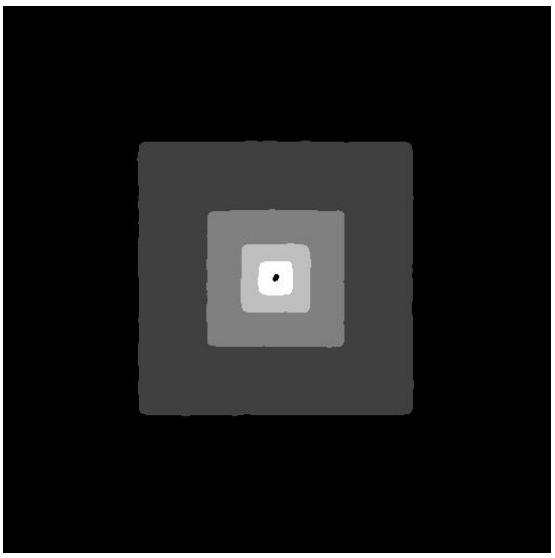

Figure 6: The disparity image for layercake image when smax = 4. The output image have 4 visible layers (topmost is missing).

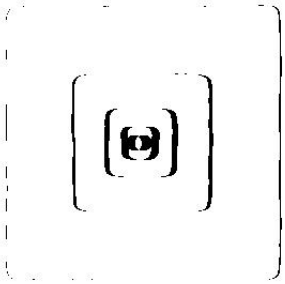

Figure 7: The binary image (for verification) for layercake image. Black areas are showing mismatch regions.