Image Processing

Report on Laboratory Session #5

Giacomo Deodato
Alessandro Patti

November 13, 2017

Part #1

Processing of stereo images. Stereo matching

For the first part of the exercise we completed the *Matching()* function so that it can correctly compute the Sum of Absolute intensity Differences and then fill the disparity map with the shift leading to the minimum SAD.

```
% compute Cost from Sum absolute Diffrences(SAD)
for c=1:1:P
  SAD=SAD+abs(imL(i+k,j+l,c)-imR(i+k,j+l-d,c));
end
```

```
%to change disp to d value  having the least dissimilarity
if (SAD<error)
  error = SAD;
  disp(i,j) = d;
end
```

Then we read the source images and we computed the disparity map using the matching function with given parameters (dmax = 45, and block size of 3 and 9).

Figure 1: Disparity map with block size of 3.

It can be noticed that increasing the block size produces less noisy results because the matching function has better precision when comparing bigger blocks of pixels but this comes with an increase in the computation cost (for

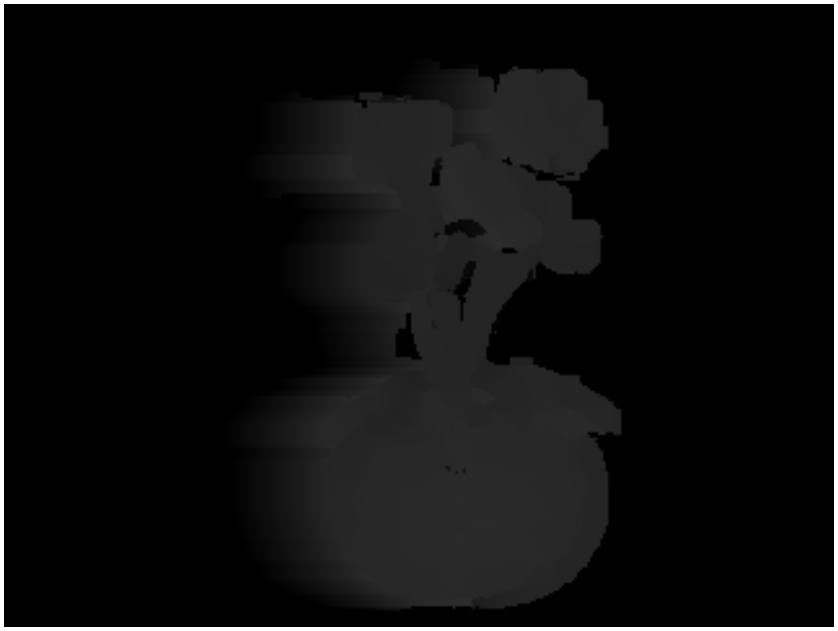

Figure 2: Disparity map with block size of 9.

the same reason we already scaled the image by a factor of 0.4).

To reduce the noise in the first disparity map an average filter with kernel size of 3 has been used.

Finally we applied the matching process also to the other couples of images with block size of 1, from the results we can see that as the baseline increases we obtain separated figures inside the same map, also due to the distance values and the block size.

Part #2

Processing of stereo images. Depth computation and segmentation

For this exercise we started by completing the code of the *DepthCompute()* function respecting the following equation:

$$Z(x,y) = f \cdot \frac{B}{disp(x,y)}, \quad for\ disp(x,y) \neq 0$$

$$Z(x,y) = 255, \quad for\ disp(x,y) = 0$$

```
function [ DepthMap] = DepthCompute (disp, B, f )
  [N, M] = size(disp);
  DepthMap = zeros(N,M);
  max = 255;

  for i = 1:1:N
    for j = 1:1:M
      if (disp(i,j) == 0)
        DepthMap(i, j) = max;
      else
        DepthMap(i,j) = f * B / disp(i, j);
      end
    end
  end
```

Then we created the ex2 file, we read the source disparity map and converted it to depth map by means of *DepthCompute()*.

To obtain a smoother result we applied median filtering with kernel size of 19 to the resulting image.

Finally we read the left image of the stereo pair, warped it on the depth data, and showed the results.

```
figure;
warp(-depth_map_filtered, imgL);
title("Warp of the image on the stereo data. Hit ENTER to
    change view.");
rotate3d on
view([0,90]);
pause;
view([0,0]);
pause;
view([60,60]);
pause;
```

The second part of the exercise required to separate the background and the different planes in the foreground.
In order to do that we displayed the histogram of the distribution of grey values in the depth map and we took 1 user input to separate background and foreground and 2 user inputs to separate three layers on the foreground. To obtain the final RGB image we created a 3D matrix with 3 layers to separate the Red, Green and Blue channels, then we assigned the correct values to the pixel indexed by the *find()* function.

```
% Show distribution of the values.
figure;
imhist(uint8(depth_map_filtered));
title('Distribution of grey values in the depth map')

% Subtract the background
x = ginput(1);
background = find(depth_map_filtered > x(1));
depth_map_filtered(background) = 255;
depth_map_filtered = 255-depth_map_filtered;
figure;
imshow(uint8(depth_map_filtered),[]);
title("Background subtraction");

% Separate intermediate layers and foreground
figure;
imhist(uint8(depth_map_filtered));
title('Identify to thesholds for intermediate planes');
x = ginput(2);
if x(1) > x(2)
  tmp = x(1);
  x(1) = x(2);
  x(2) = tmp;
end
[N, M] = size(depth_map_filtered);
rgb_map = zeros([N M 3]);

[row, col] = find(depth_map_filtered < x(1) &
   depth_map_filtered > 0);
for i = 1:size(row)
  rgb_map(row(i), col(i), 1) = 255;
end
[row, col] = find(depth_map_filtered < x(2) &
   depth_map_filtered > x(1));
for i = 1:size(row)
  rgb_map(row(i), col(i), 2) = 255;
end
[row, col] = find(depth_map_filtered > x(2));
for i = 1:size(row)
  rgb_map(row(i), col(i), 3) = 255;
end
imshow(rgb_map);
```

Part #3

3D displays using colour filter glasses. Anaglyph producing and perceiving in 3D

For the last exercise we just uploaded the two source images, created a new image with same dimensions and copied the red channel of the left image and the green and blue channels of the right one.
Finally we showed the result of this exercise and the "Lion Statue anaglyph" image to look at them with colour filter glasses.

```
% read source images
img0 = imread('Pair_anaglyph/x=0.0.jpg');
img1 = imread('Pair_anaglyph/x=0.1.jpg');

% copy the red color channel of the left image
img(:,:,1)=img0(:,:,1);

% copy the green and blue channels of the right image
img(:,:,2)=img1(:,:,2);
img(:,:,3)=img1(:,:,3);

% show the result
figure;
imshow(img);
title('Flower Anaglyph');

% show the lion statue anaglyph
figure;
imshow(imread('Lion_anaglyph/Lion_Statue_anaglyph.jpg'));
title('Lion statue anaglyph');
```