Nanyang Technological University

Computer Vision Assignment02

Stereo Vision

Author:
Xu Yifan

Lecturer:
Lu Shijian

October 14, 2020

Contents

1 Describe the procedure of disparity computing

1. Polar rectifying: The purpose of rectifying is to make polar of two images in a horizontal line. In this assignment, we ignore this procedure because we have gotten rectified images before.

2. Feature matching: In this step, we use NCC to achieve feature matching. After rectifying a pair of images, the depth is calculated by horizontal bias because the depth is proportionate to the bias. After calculating the depth Z, we can get the disparity map which is the same size as original images.

$$Z = \frac{fb}{x_l - x_r}$$

   f is the focus
   b is the distance between centers of two different viewpoints

3. Depth recovering: From the disparity map, we also get the depth map basing on the formula of resembled rectangular.

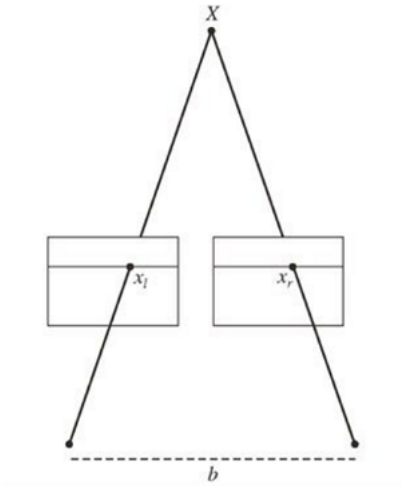

Figure 1: disparity map

2 Write a computer algorithm that computes the disparity

2.1 Code

The Python function mainly achieves the mathematic formula NCC (Normalization cross-correlation) and return best disparity for every pixel.

$$ncc(I_1, I_2) = \frac{\sum_x (I_1(x) - \mu_1)(I_2(x) - \mu_2)}{\sqrt{\sum_x (I_1(x) - \mu_1)^2 \sum_x (I_2(x) - \mu_2)^2}}$$

```
im_l = np.array(Image.open('../pictures/corridorl.jpg').convert('L'
                                    ), 'f')
im_r = np.array(Image.open('../pictures/corridorr.jpg').convert('L'
                                    ), 'f')
# Set steps and starts
steps = 12
start = 4
# Set the length of ncc
wid = 9
res = plane_sweep_ncc(im_l, im_r, start, steps, wid)
scipy.misc.imsave('../pictures/depth_corridor.jpg', res)

# Generate the comparison
plt.rcParams['figure.figsize'] = (16.0, 16.0)
plt.subplot(2, 2, 1)
plt.title("corridorl.jpg")
plt.imshow(Image.open("../pictures/corridorl.jpg"), cmap="gray")

plt.subplot(2, 2, 2)
plt.title("corridorr.jpg")
plt.imshow(Image.open("../pictures/corridorr.jpg"), cmap="gray")

plt.subplot(2, 2, 3)
plt.title("depth_corridor.jpg")
plt.imshow(Image.open("../pictures/depth_corridor.jpg"), cmap="gray
                                    ")
```

2.2 Result

In order to get the comparison, we use the Python function to get the disparity map and then compare it with original images. Use the code below to generate the result of comparison.

```
from PIL import Image
import numpy as np
from scipy.ndimage import filters
import scipy.misc
import warnings
warnings.filterwarnings("ignore")


def plane_sweep_ncc(im_l, im_r, start, steps, wid):
```

```
    """ Use NCC to compute disparity map """
    m, n = im_l.shape
    # Save different sums
    mean_l = np.zeros((m, n))
    mean_r = np.zeros((m, n))

    s = np.zeros((m, n))
    s_l = np.zeros((m, n))
    s_r = np.zeros((m, n))
    # SSave the disparity maps
    dmaps = np.zeros((m, n, steps))
    # Computer the average of images' trunks
    filters.uniform_filter(im_l, wid, mean_l)
    filters.uniform_filter(im_r, wid, mean_r)
    # Normalize the picture
    norm_l = im_l - mean_l
    norm_r = im_r - mean_r
    # Try different disparities
    for displ in range(steps):
        # make left image move to the right and summarize them
        filters.uniform_filter(np.roll(norm_l, -displ - start) *
                                        norm_r, wid, s)
        filters.uniform_filter(np.roll(norm_l, -displ - start) * np
                                        .roll(norm_l, -displ -
                                        start), wid, s_l)
        filters.uniform_filter(norm_r * norm_r, wid, s_r)
        # Save the grade of NCC
        dmaps[:, :, displ] = s / np.sqrt(s_l * s_r)
    # Choose best pixel for every sub-pictures
    return np.argmax(dmaps, axis=2)
```

2.3 Compare disparity maps

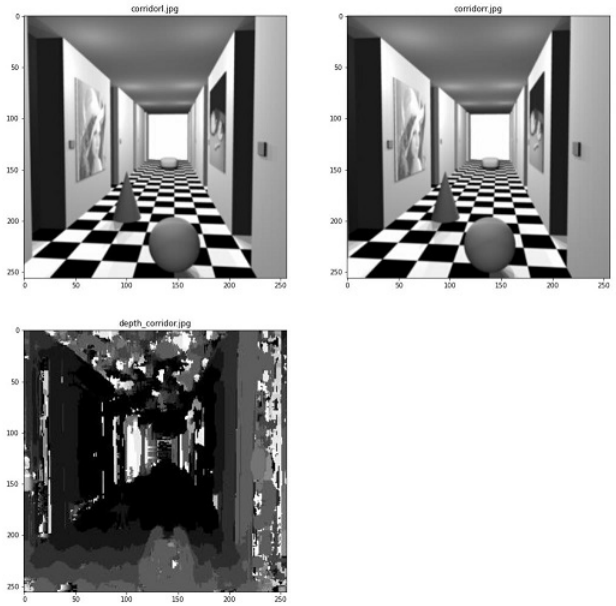

Figure 2: Corridors' comparisons between disparity map and original images

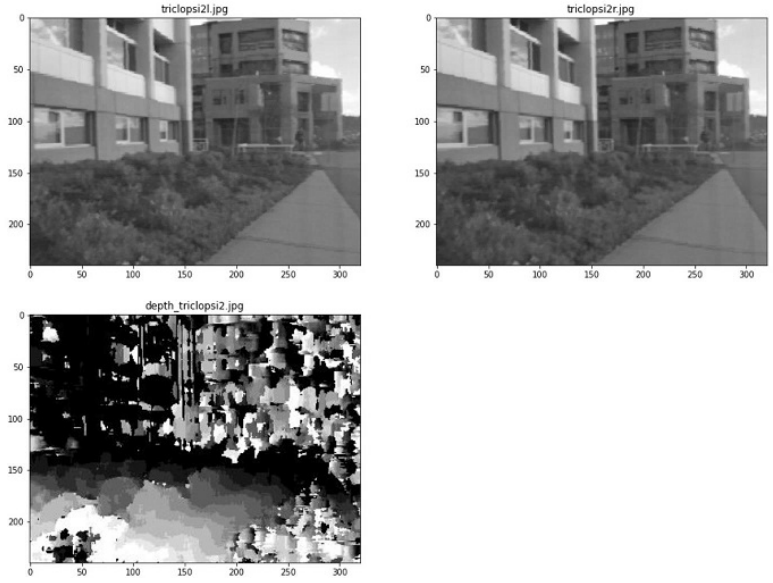

Figure 3: Triclopsi2's comparisons between disparity map and original images

3 Apply developed algorithm and derive the disparity maps

3.1 Gaussian Filter

After reading some papers, I try to use NCC with gaussian filter to improve the disparity map.

```
def plane_sweep_gauss(im_l, im_r, start, steps, wid):
    """ Use cross-correlation with Gaussian filter to compute
                                        disparity map """
    m, n = im_l.shape
    # Save different sums
    mean_l = np.zeros((m, n))
    mean_r = np.zeros((m, n))

    s = np.zeros((m, n))
    s_l = np.zeros((m, n))
    s_r = np.zeros((m, n))

    # Save the disparity maps
    dmaps = np.zeros((m, n, steps))
    #  Computer the average
    filters.gaussian_filter(im_l, wid, 0, mean_l)
    filters.gaussian_filter(im_r, wid, 0, mean_r)
    # Normalize the picture
    norm_l = im_l - mean_l
    norm_r = im_r - mean_r
    # Try different disparities
    for displ in range(steps):
        # make left image move to the right and summarize them
        filters.gaussian_filter(np.roll(norm_l, -displ - start) *
                                        norm_r, wid, 0, s)
        filters.gaussian_filter(np.roll(norm_l, -displ - start) *
                                        np.roll(norm_l, -displ -
                                        start), wid, 0, s_l)
        filters.gaussian_filter(norm_r * norm_r, wid, 0, s_r)
        # Save the grade of NCC
        dmaps[:, :, displ] = s / np.sqrt(s_l * s_r)
    # Choose best pixel for every sub-pictures
    return np.argmax(dmaps, axis=2)
```

4 Discuss your observation of the obtained disparity maps

4.1 Observation

From the result based on NCC or Gaussian filter, we can know that:

1. There is more information among the disparity maps based on NCC, but also more noise in these pictures. However, for gaussian filter to calculate the disparity, the image is smooth but too noisy.

2. For a series of the pictures about corridors, the disparity map filtered by NCC is better than another, because it includes more details and will be easier to observe.

3. For a series of the pictures about triclopsi2, the disparity map filtered by Gaussian filter is better than another, because it ignores more useless details and can show a more smooth disparity map to us.

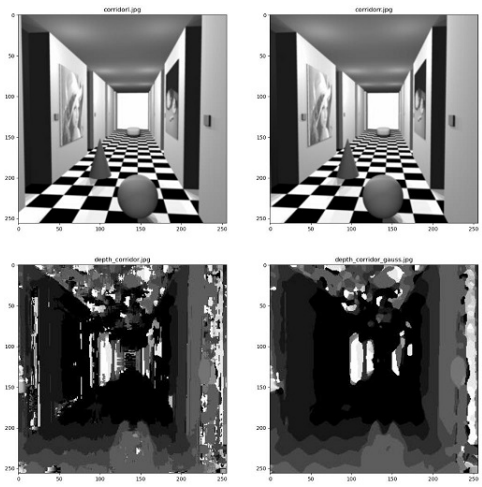

Figure 4: 4 comparisons between disparity map and corridors' images

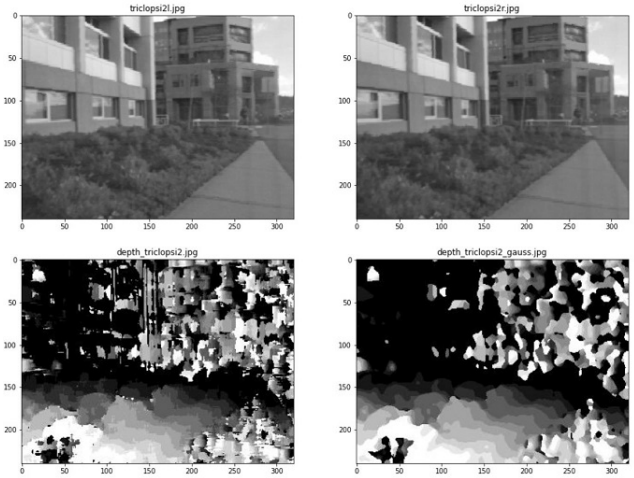

Figure 5: 4 comparisons between disparity map and triclopsi2' images

5 Discuss what affects the disparity map and improvements

5.1 Affects

1. The algorithm: From the 2 algorithms we chose above, the results of disparity map mainly depend on filters. The weights of the NCC filter are same, but the weights of gaussian filter are decreasing from the centre to borders. So that's why we can see more smooth images by the way of gaussian filter.

2. The size of windows: The size of filters influences on the result of filtering and finally influences on disparity map. Seeing the disparity map below, we know that for the way of NCC, the larger the size of filters, the less details are, but the smoother the images are. On the contrast, for the way of gaussian filter, the increase of the size of windows will show much less information to us and it is not suitable to expand the size of windows by this way.

6 Implement and verify your ideas

6.1 Changing the size of windows

Based on the analysis and assumptions of affects and improvements above, we can test different the size of windows to influence the rectified images.

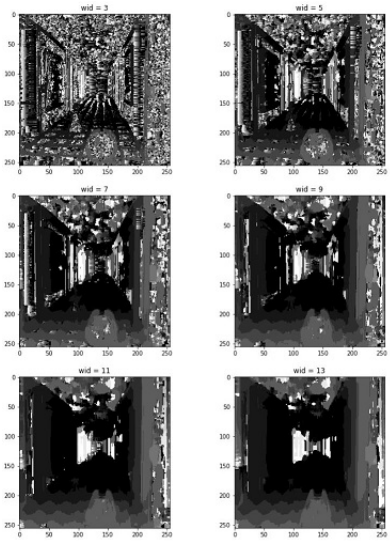

Figure 6: Corridors' ncc of different wids

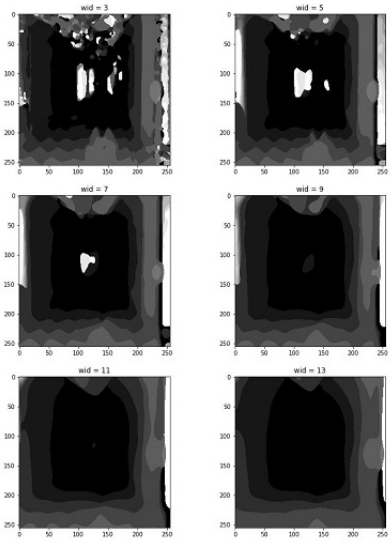

Figure 7: Corridors' ncc of different wids with gaussian filter

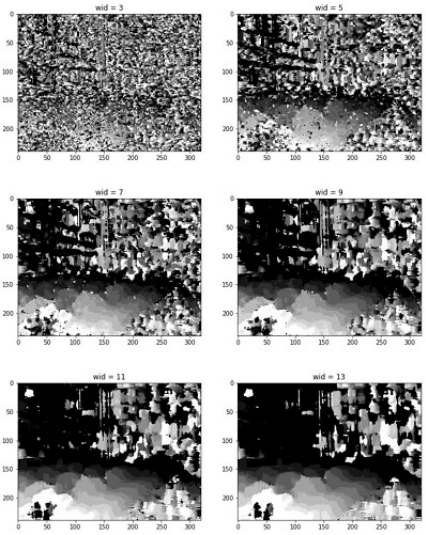

Figure 8: Triclopsi2's ncc of different wids

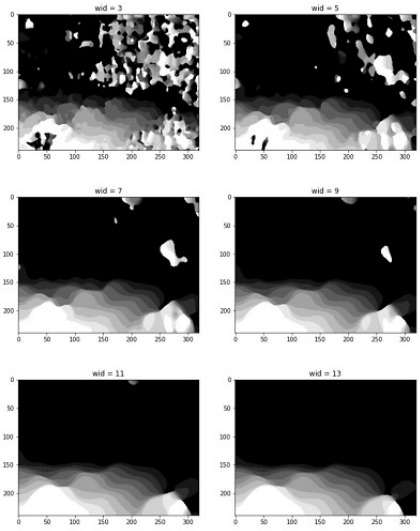

Figure 9: Triclopsi2's ncc of different wids with gaussian filter

Summary Finally, we find that if we choose the size of windows for basic NCC is 9 or 11, not too big or too small, the result is better than others. And if we choose the size of windows for Gaussian filter is 3 or 5, it's enough because too large size will ignore too many details.

References

[1] Daniel Scharstein. A taxonomy and evaluation of dense two-frame stereo correspondence. In *Proceedings of the IEEE Workshop on Stereo and Multi-Baseline Vision, Kauai, HI, Dec, 2001*, 2001.

[2] Daniel Scharstein and Richard Szeliski. High-accuracy stereo depth maps using structured light. In *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.*, volume 1, pages I–I. IEEE, 2003.

[3] Jan Erik Solem. *Programming Computer Vision with Python: Tools and algorithms for analyzing images.* " O'Reilly Media, Inc.", 2012.

[4] Wikipedia contributors. Computer stereo vision — Wikipedia, the free encyclopedia, 2020. [Online; accessed 14-October-2020].