Edge Dectection, Template Matching and Blob Coloring

Homework 3

Jinaykumar Patel
1001937580

1. Implement Edge Detection Using Prewit Templates

$$\text{vert.}: \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \qquad \text{min. diag.}: \begin{bmatrix} -1 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}$$

$$\text{hor.}: \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} \qquad \text{maj. diag.}: \begin{bmatrix} 0 & -1 & -1 \\ 1 & 0 & -1 \\ 1 & 1 & 0 \end{bmatrix}$$

Edge detection is implemented using convolution with the four Prewit templates shown above. Each template is implemented separtely and results for each templates are saved in PNG files as ***p1_chess***_{*template_name*} and ***p1_nedderman***_{*template_name*}. The results obtained from edge detection implementation in Chess and Nedderman images are shown in Fig. 1 and Fig. 2, respectively.

2. Implement Template Matching Using Normalized Convolution

Here a specific region is chosen and is used as the template i.e. finding all instances of similar objects in the image. Template matching is implemented using this template with the use of normalized cross correlation (convolution which adjusts for the image and template means as well as for the image and template variances - i.e. contrast). The result of the convolution is written back into the result image, normalizing the values to be between 0 and 255.

Two templates are chosen in the nedderman images:

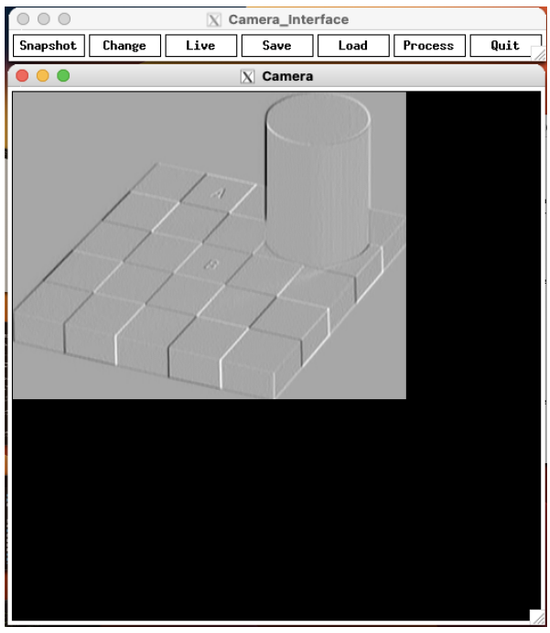
1. **Small windows** in the nedderman buildings are selected as template and used for template matching. The results is shown in Fig. 3. We can see the bright regions in the Fig. 3b where the windows are in the original image.

2. **Tree** in the middle of the image is selected as a template. The result is shown in Fig. 4. We can see the bright spot in the place of the tree in the middle (Fig. 4b).

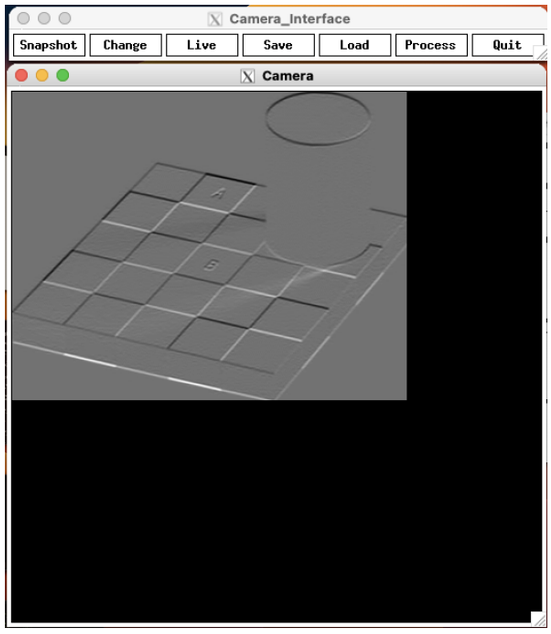
Now, chess image is used. A corner between white and black spots is chosen as the template and template matching is implemented. The results are shown in Fig. 5. We can see that some white corners and some black corners in Fig. 5b. Black corner are the ones which are exactly opposite to the region selected as the template.

3. Implement Segmentation Using Blob Coloring

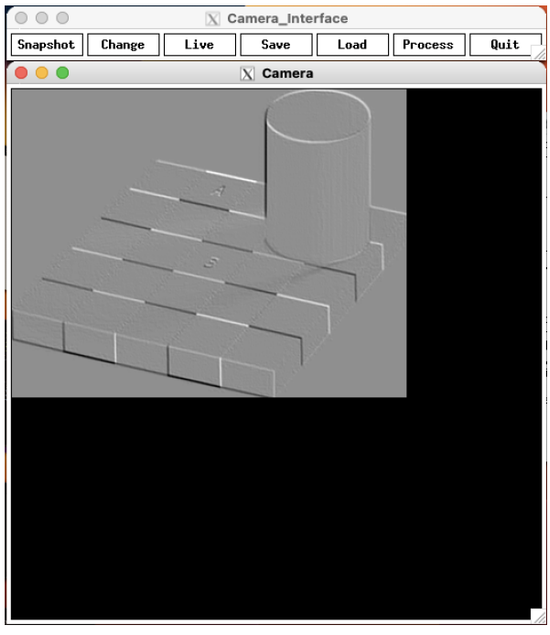
Blob coloring is implemented to identify regions with a common intensity in the image. Here the threshold between the pixels is chosen to be 10. This value can be changed in the threshold function. Fig. 6 shows the result obtain when blob coloring algorithm is implemented. There are some small dotted regions appearing in the resulted image which I am not aware of.



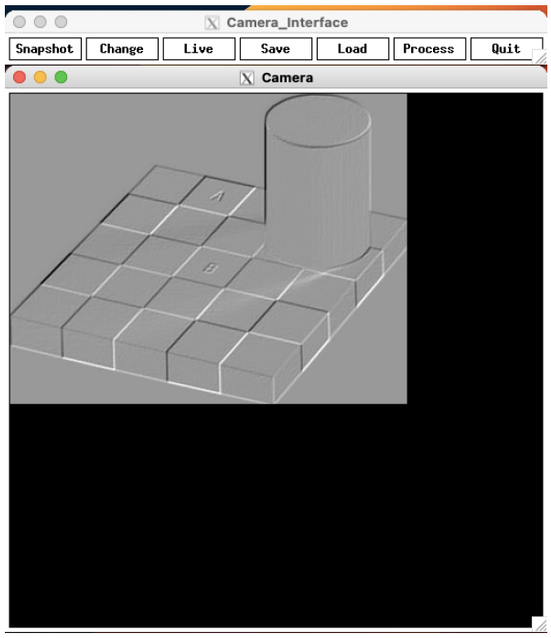
(a) Horizontal Template



(b) Vertical Template

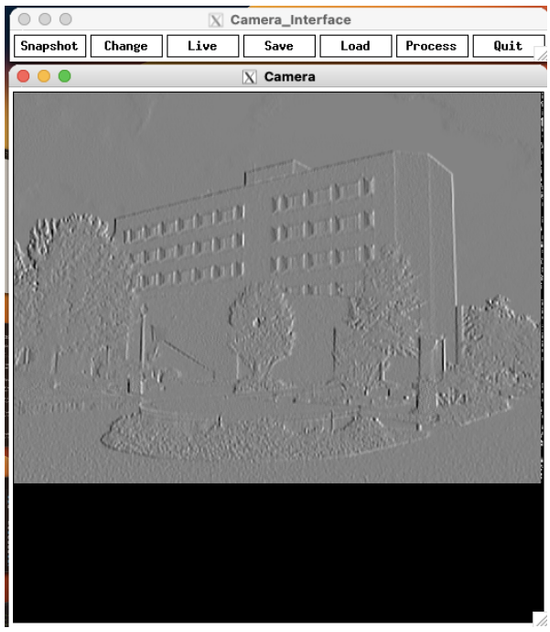


(c) Maj. diag Template

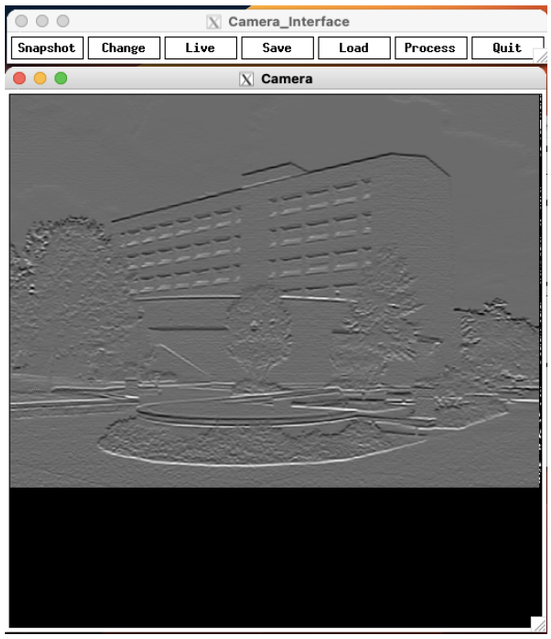


(d) Min. diag Template

Figure 1: Edge Detection (Chess) using Prewit Templates

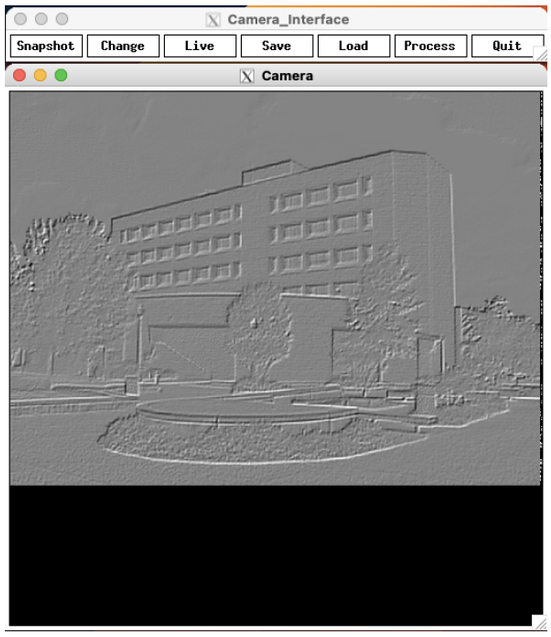


(a) Horizontal Template



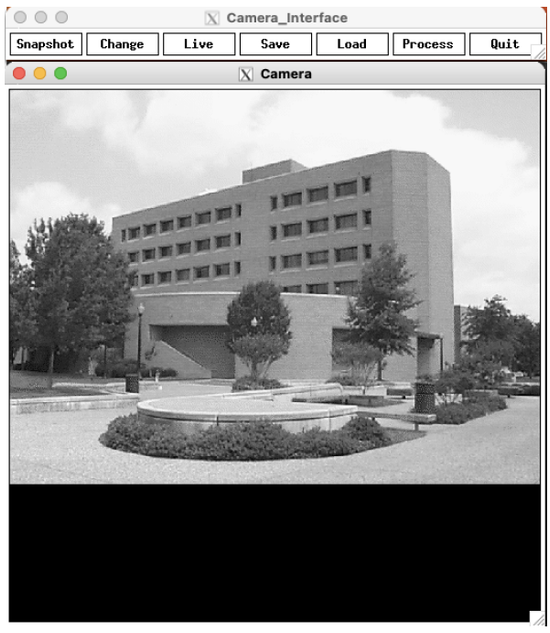
(b) Vertical Template

(c) Maj. diag Template

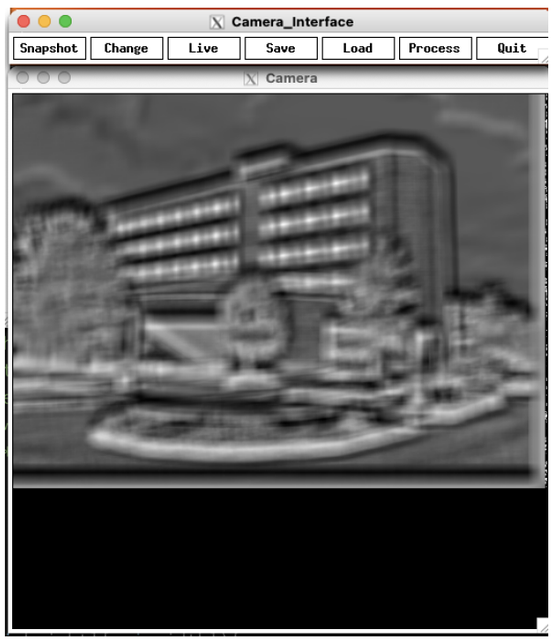


(d) Min. diag Template

Figure 2: Edge Detection (Nedderman) using Prewit Templates

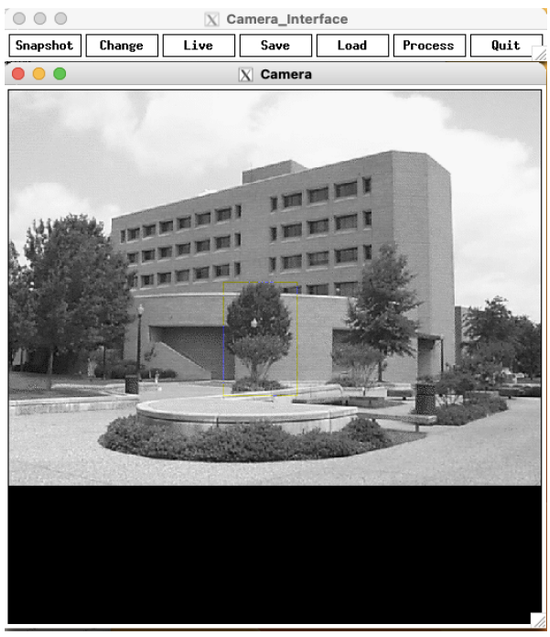


(a) Nedderman Image

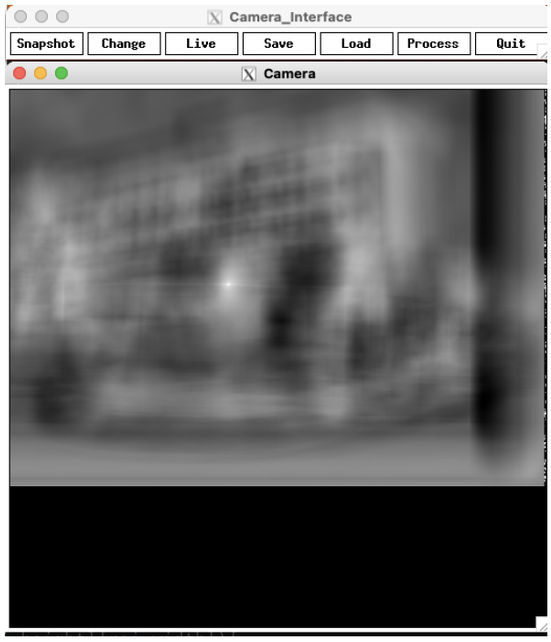


(b) Image after template matching

Figure 3: Template Matching (Window as a template)

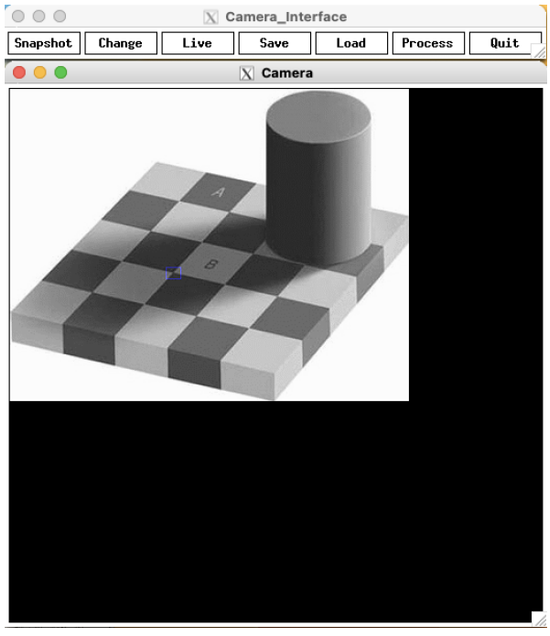


(a) Nedderman Image

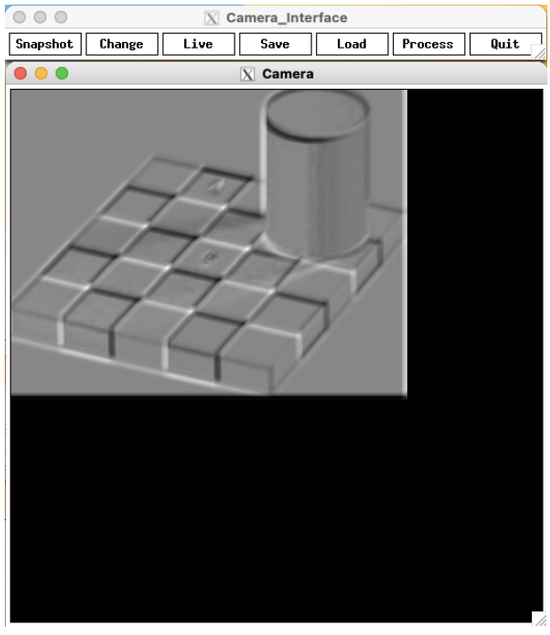


(b) Image after template matching

Figure 4: Template Matching (Tree as a template)

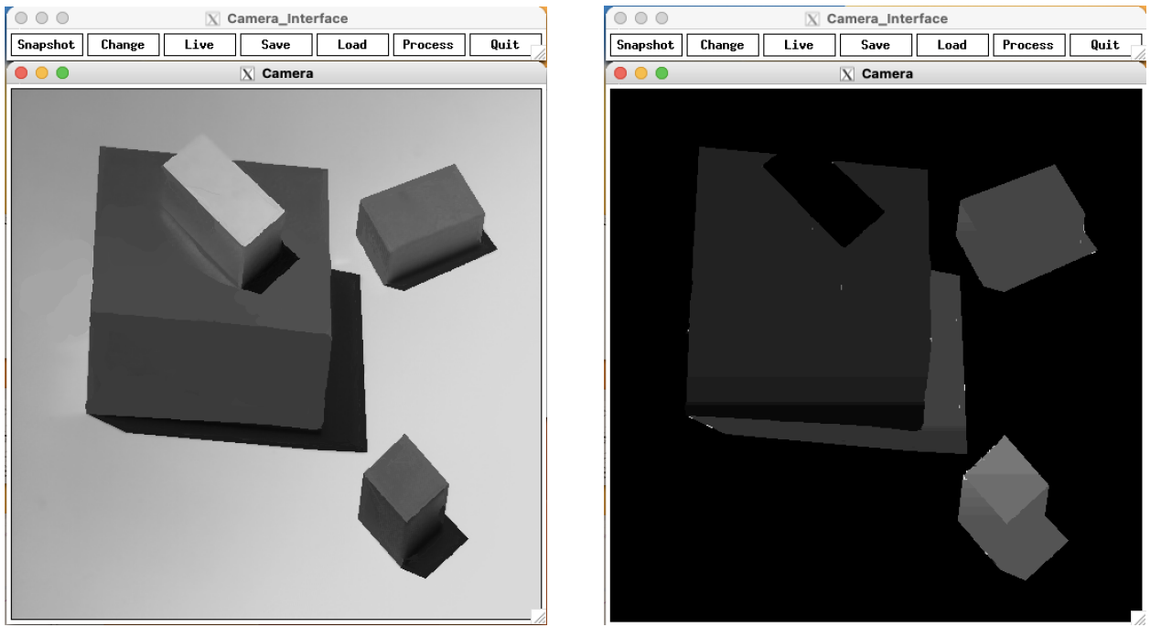


(a) Chess Image



(b) Image after template matching

Figure 5: Template Matching (Corner as a template)



(a) Blocks Image

(b) Image after blob coloring

Figure 6: Blob coloring