Cursor Detection

Daniel Amirtharaj
damirtha@buffalo.edu

1 Objective

To detect a cursor in an image, using a given cursor template image.

2 Analysis

2.1 Template Matching

Template matching is a technique used in image processing to identify pixel locations in an image whose neighbouring region is similar to pixels in a given template image. There are several ways in which template matching can be achieved. The difference in the squared distances of pixels in the input image and the template can give a good idea on similarity. This method is not effective against differences in contrast and intensity between the image and the template. Using Normalized cross correlation for template matching, is a more robust method that is not affected by such intensity differences.

2.2 Normalized cross correlation

The normalized cross correlation between the image and the template is given by the following equation.

$$O(x,y) = \frac{\sum_{(x',y')} T(x',y').I(x+x',y+y')}{\sqrt{\sum_{(x',y')} T(x',y')^2.\sum_{(x',y')} I(x+x',y+y')^2}},\ x'\epsilon(-k,k)\ and\ y'\epsilon(-k,k)$$

Here, O is the output image, T the template and I the input image.

2.3 Image Transforms

Before proceeding with template matching, the image can be transformed to another representation that may yield better results. Since the features that can be discerned directly from the cursor template are limited here, it will be best to find a transform that will highlight only the distinguishing features available, which are the edges. The Sobel operator, LoG, DoG etc. can be applied to the images before template matching to increasing the performance of the program.

3 Method

The following steps were used to detect cursors in the given images using the given template.

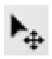
1. Transform each input image (in grayscale) and the template, using an image transformation to get the gradient/edges of the image, and compute the magnitude of the edges detected.

2. Resize the template to ensure scale invariant cursor detection.
3. Use template matching to find regions in the input image similar to the template given. This is done here using normalized cross correlation.
4. Find the maxima of the output image, obtain its pixel location and intensity. This pixel corresponds to the pixel whose neighbours have the best match with the template.
5. Find a threshold to discern between good matches and bad matches. This is done by trial and error here.
6. Based on the threshold chosen, mark good matches by drawing a rectangle around the maximum pixel location.
7. Tweak different parameters, transformations to see which method gives the best performance.

3.1 Templates used

The following cursor templates were used to detect images for sets A and B.

(a)


(b)

(c)

Figure 1: Cursor templates used for set B, (c) was used for set A and B

3.2 Color coding cursor types

To highlight the type of cursor detected in the output images, color codes were used. Green denotes cursor type (a), blue denotes cursor type (b) and red denotes cursor type (c). Cursor types are shown in figure 7.

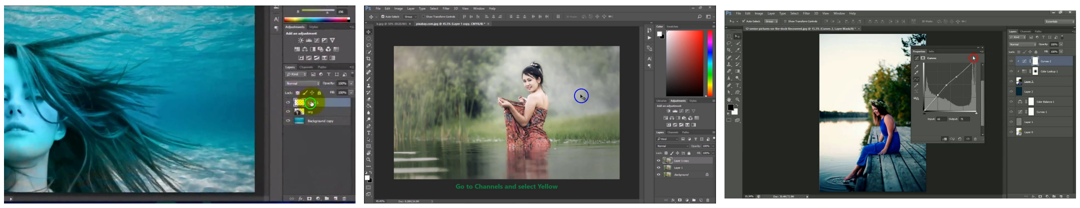


Figure 2: Different cursors highlighted using the color code

3.3 Image scaling

To ensure scale invariance the template was resized to capture matches in varying sizes. A scale of 0.8 to 1.5 was used, increasing and decreasing the size of the template in steps of 0.01. The best match is chosen for the scale that gives the best value for normalized cross correlation.

4 Results

The following results were obtained when template matching was applied on the input images and the cursor template with different image transforms.

4.1 No filtering / grayscale conversion

When template matching was applied on the original color input images using the set template, without any transformation or conversion, the performance was below average and it got a lot of the points wrong. Below image shows how the template was detecting non-cursor points as well.

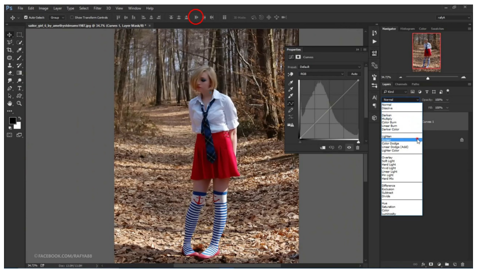
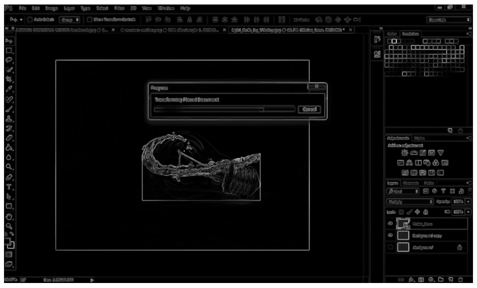


Figure 3: Bad match, when no image transformation is applied

4.2 Sobel filter

In order to enhance the detection of the cursor templates, the sobel operator was used to highlight edges in the image and the template. Below image shows a sobel filtered image and template.



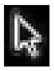


Figure 4: Image and template with the edges highlighted using sobel filter, (not to scale)

Cursor detection improved with the use of the sobel filter. With roughly 70% of cursors (both actual pointers and similar pointers in the image) detected as expected. It can be observed here, that

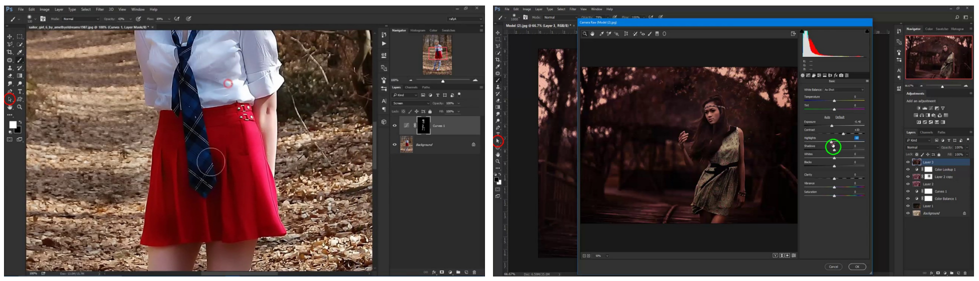


Figure 5: Cursor detection after sobel filter was applied

although some cursors were getting detected, the program was detecting other bad matches as cursors as well, despite setting the threshold at 0.9.

4.3 Difference of Gaussian

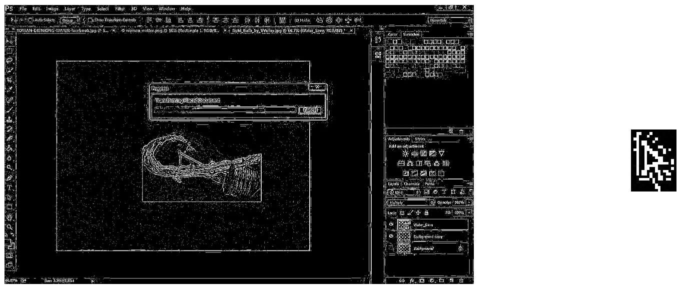


Figure 6: Image and template with the edges highlighted using DoG filter, (not to scale)

The difference of gaussian was also used to highlight edges in the image and the template, to see if it performed better than the sobel operator. Below image shows a DoG filter applied on the image and template.
Cursor detection did not improve here. It can be observed from the template that a lot of noise had been added, resulting in bad matches.

4.4 Laplacian of Gaussian

Since DoG was not very effective LoG was used, to see if it performed better. Below image shows a LoG filter applied on the image and template.

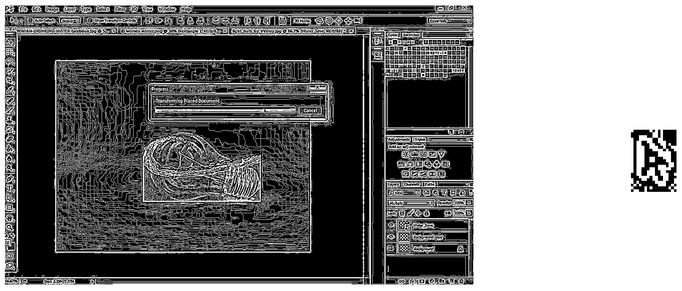


Figure 7: Image and template with the edges highlighted using LoG filter, (not to scale)

The laplacian of gaussian was able to retain intensity of contours better. It performed the best of all the other image filters and got roughly 90% of the cursors (both actual and similar) right, with a good match/ bad match threshold of 0.65.

Below are a few images, where cursors were detected using the LoG filter. Negative images are also shown. Some patches in the images that were similar to the cursor templates were also detected, as can be seen here.

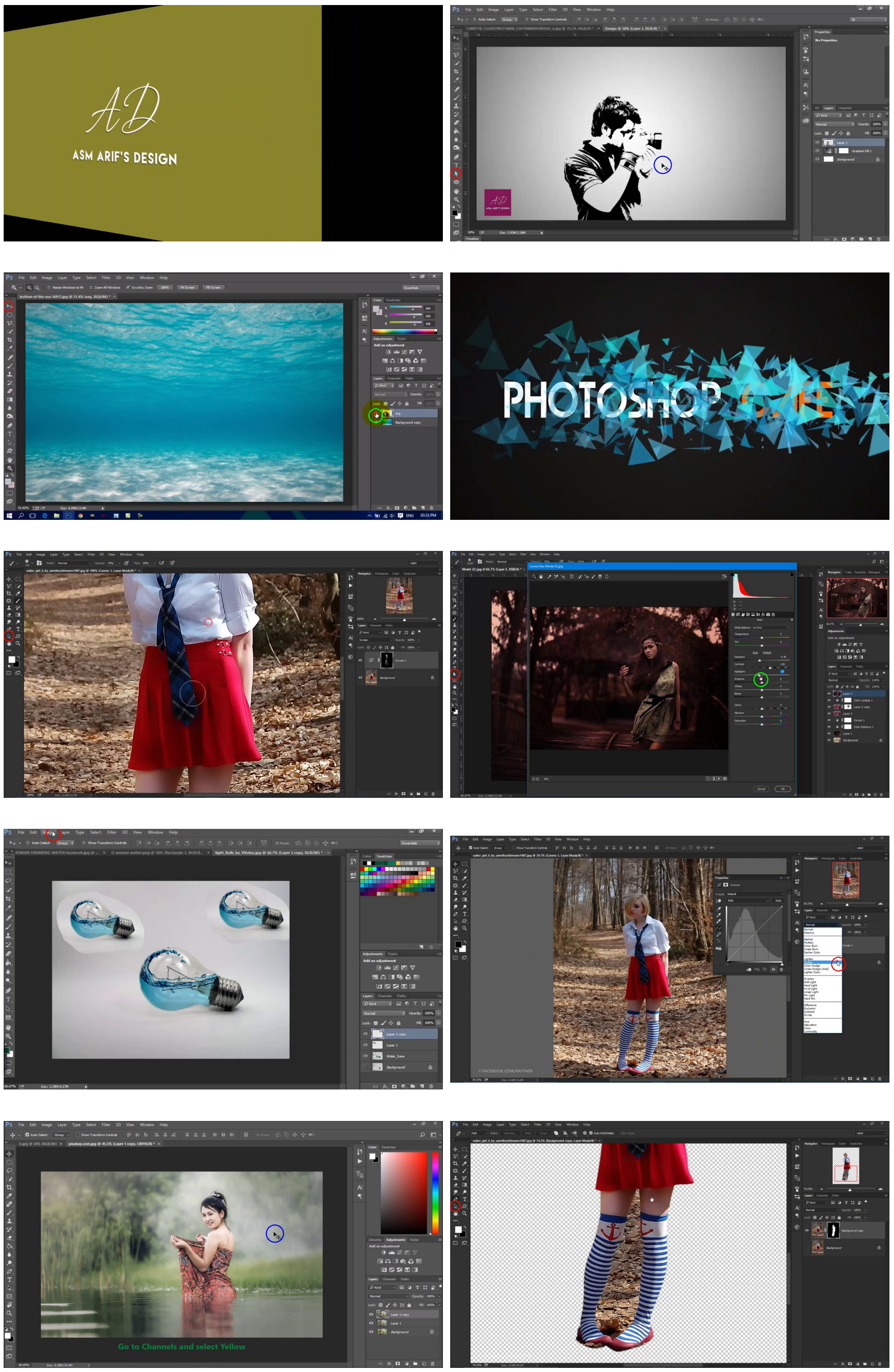


Figure 8: Output images after cursor detection from both set A and B