Implementation of Normalised Cross-correlation Based Template Matching Method

GNR 602 - Programing Assignment |

Divyansh Mittal	170040031
Rishabh Arya	180040080
Jos Katiyare	190020054

TEMPLATE MATCHING

- **Template matching**

  Template Matching is a methodology in digital image processing to identify little components of a figure which match a template image.It is used to match a template to an image wherever the template is a sub image which contains the form which we want to find out. Templates are usually employed to print characters, identify numbers, and other little, simple objects. It can be used for detection of edges in figures.

- **Source image**

  An image in which we are hoping to find template image.

- **Template Image**

  The patch image that can be compared to the template image and our objective is to discover the most effective technique for the best matching region.

Template Matching Approaches

- **Featured-based approach**

  The Featured-based method is appropriate while both reference and template images have more connection with regards to features and control points. The features comprises of points, a surface model which needs to be matched,and curves

- **Area-based approach**

  Area-based methods merge the matching part with the feature detection step. These techniques manage the pictures without attempting to identify the remarkable article. Windows of predefined size are used for the estimation of correspondence.

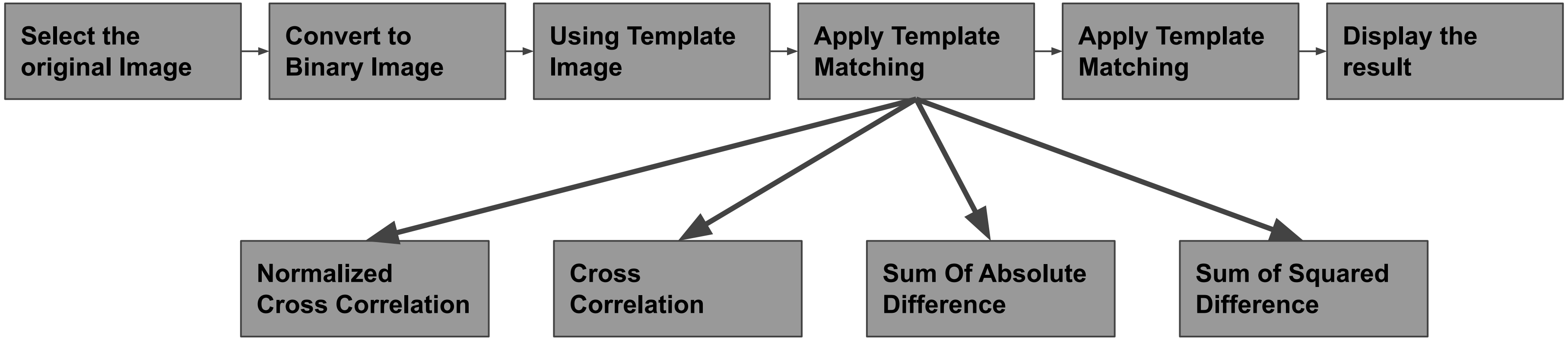
- **Template-based approach**

  Template-based template matching approach could probably require sampling of a huge quantity of points, it's possible to cut back the amount of sampling points via diminishing the resolution of the search and template images via the same factor and performs operation on resulting downsized images

- **Motion Tracking and Occlusion Handling**

  For the templates which can't provide and may not provide an instantaneous match, in that case Eigen spaces may be used, which provides the details of matching image beneath numerous conditions.

Methodology



Techniques

1. **Naive template matching**

   In this technique the template is positioned over the image at every possible location, and each time some numeric measure of similarity between the template and the image segment it currently overlaps with is calculated. Finally the position with best similarity measure is identified.

2. **Image Correlation Matching**

   This method overcomes the Naive template matching problems like computing similarity patterns in aligned pattern images.

   a) Cross-Correlation

   This method is basically a simple sum of pairwise multiplications of corresponding pixel values of the image.

   b) Normalized cross-correlation

   This method is an improvisation on cross correlation. This improvisations are

   - The outcomes are constant to the global brightness changes,that is darkening of whichever figure or consistent brightening have no impact on end results.
   - The final correlation value is scaled to [-1, 1] range, in order that NCC of 2 alike figures is equal to 1.0

Techniques

3. **Sum of squared differences**

   In addition to template matching this technique is also used in noise and rotation detection test.In this method measure of variation or deviation from the mean Is represented by sum of squares.It is calculated as a summations of the squares of the variations from the mean. The performance of this method is done by making the comparison based on the value of correlation coefficient and that produced from different template image

Drawbacks of above techniques

These methods are inefficient as it is time-consuming in computing the pattern correlation image for images varying from medium to large images.

Normalized cross-correlation

Normalized cross-correlation is an improved model of the traditional cross correlation methodology. Two major improvement of those are -

- Effect of darkening have no impact on end results. Thus the outcomes are constant wrt brightness changes
- The final correlation value is scaled to [-1, 1] range, in order that NCC of 2 alike figures is equal to 1.0
- Formula of NCC is

$$\frac{1}{N\sigma 1\sigma 2\sigma 3}\sum_{u,v}[(image1(u,v)-\overline{image1})\times(image2(u,v)-\overline{image2})]$$

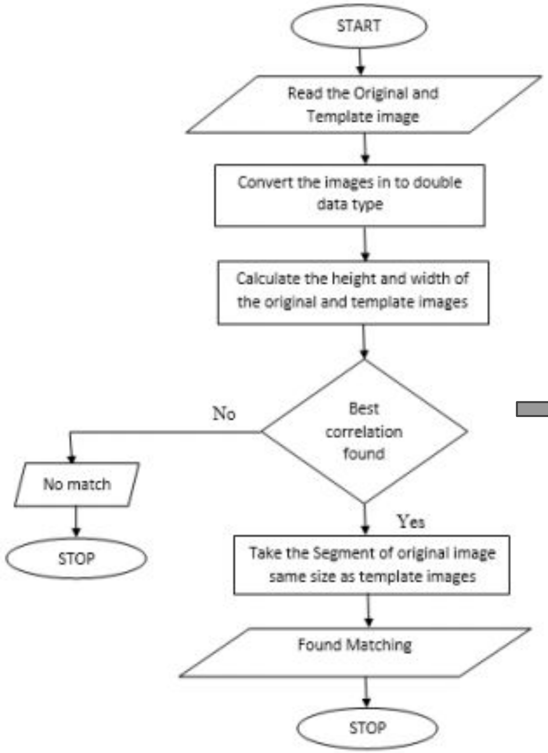
Where image1 and image2 are two images. u and v are their respective pixel coordinates. Σ is a constant

Normalized cross-correlation

- **Benefits**
  Conventional correlation-based image matching techniques not succeed in case if there are significant scale changes or large rotations among the two figures. But Normalized cross-correlation can be give efficient resemblance measure meant for matching applications

- **Drawbacks**
  Normalized Cross Correlation is more complicated as compared to both SSD and SAD algorithms as it entails several division, square root operations and multiplication.In case of NCC, there’s a big increase within the inaccuracy rate which is because of the shaded input figure

Workflow


START
Read the Original and Template image
Convert the images in to double data type
Calculate the height and width of the original and template images
Best correlation found
No
No match
STOP
Yes
Take the Segment of original image same size as template images
Found Matching
STOP

Has two types in our case

Packages Required

- Numpy
- Opencv-python
- Matplotlib

Code Snippets

- This is our function to calculate calculate Normalized Cross correlation value , it takes target image and the cropped source image(with the same number of pixel as of target image) as input

```
def Normalized_Cross_Correlation(roi,
target):
   cor = np.sum(roi * target)
   nor = np.sqrt( (np.sum(roi ** 2))) *
np.sqrt(np.sum(target ** 2))
   #print(cor/nor)
   return cor / nor
```

Code Snippets

Type I

Template Match Function

```
def tm(img, target):

    img_height, img_width = img.shape
    tar_height, tar_width = target.shape
    (max_Y, max_X) = (0, 0)
    MaxValue = 0


    NccVal = np.zeros((img_height-tar_height, img_width-tar_width))

    img = np.array(img, dtype="int")
    target = np.array(target, dtype="int")
```

Storing the Image and target dimensions along with coordinates of point on image plane which have maximum NCC till now along with its value

Converting Image and target into Numpy array time ,Also initializing a zero valued Numpy array for NCC value

Code Snippets

Type I

```
for y in range(0, img_height-tar_height):


  for x in range(0, img_width-tar_width):

      roi = img[y : y+tar_height, x : x+tar_width]
      NccVal[y, x] = Normalized_Cross_Correlation(roi, target)

      if NccVal[y, x] > 0.975:
         MaxValue = NccVal[y, x]
         (max_Y, max_X) = (y, x)
         print(MaxValue)
         top_left = (max_X, max_Y)
         cv2.rectangle(image, top_left, (top_left[0] + target.shape[1], top_left[1] + target.shape[0]), 0, 3)
return (max_X, max_Y)
```

Calculating NCC for every valid pixel coordinate in image. Here in Type 1 we have made a rectangle for every pixel coordinate which gave NCC value greater than 0.975
Might be used to recognize more than one instance

Code Snippets

Type II

```
def tm(img, target):

    img_height, img_width = img.shape
    tar_height, tar_width = target.shape
    (max_Y, max_X) = (0, 0)
    MaxValue = 0


    NccVal = np.zeros((img_height-tar_height, img_width-tar_width))

    img = np.array(img, dtype="int")
    target = np.array(target, dtype="int")
```

Storing the Image and target dimensions along with coordinates of point on image plane which have maximum NCC till now along with its value

Converting Image and target into Numpy array time ,Also initializing a zero valued Numpy array for NCC value

Code Snippets

Type II

```
for y in range(0, img_height-tar_height):


  for x in range(0, img_width-tar_width):

       roi = img[y : y+tar_height, x : x+tar_width]
       NccVal[y, x] = Normalized_Cross_Correlation(roi, target)

           if NccVal[y, x] > MaxValue:
            MaxValue = NccVal[y, x]
            (max_Y, max_X) = (y, x)
            print(MaxValue)
 top_left = (max_X, max_Y)
 cv2.rectangle(image, top_left, (top_left[0] + target.shape[1], top_left[1] + target.shape[0]), 0, 3)
 return (max_X, max_Y)
```

Calculating NCC for every valid pixel coordinate in image
Here in type 2 we are only drawing a rectangle corresponding to pixel with highest NCC value
Thus only one instance recognized

Code Snippets

Main Function

```
if __name__ == '__main__':

    ap = argparse.ArgumentParser()
    ap.add_argument("-i", "--image", required = True, help = "Path to input image")
    ap.add_argument("-t", "--target", required = True, help = "Path to target")
    args = vars(ap.parse_args())

    image = cv2.imread(args["image"], 0)
    target = cv2.imread(args["target"], 0)

    #cv2.imshow("image", image)
    #cv2.waitKey(0)

    height, width = target.shape
    top_left = tm(image, target)

    #draws a rectangle based on the info received
    cv2.rectangle(image, top_left, (top_left[0] + width, top_left[1] + height), 0, 3)

    pl.subplot(111)
    pl.imshow(image)
    pl.title('result')
    pl.show()
```

Taking input of source and target image through command line

Drawing the rectangle based on the response received

How to run?

- From the Command Line Change Directory(through cd) to the project directory
- From there use the command

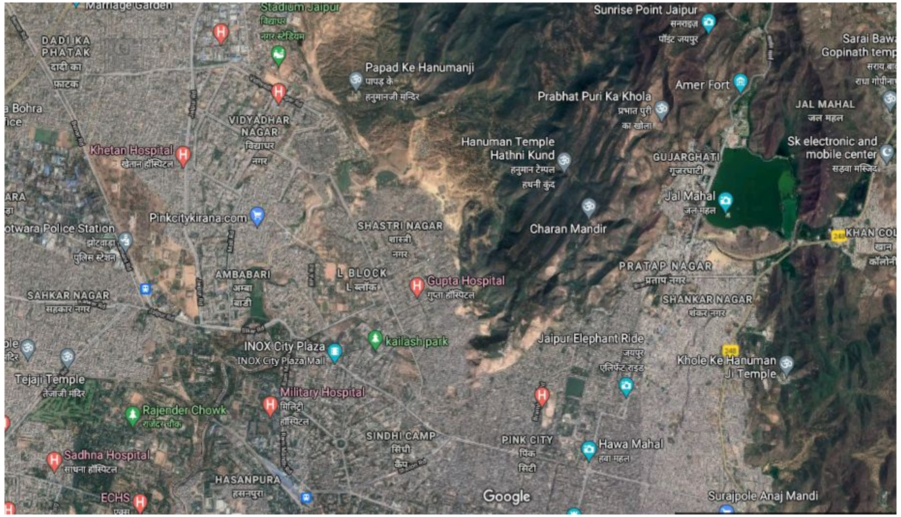
“python Type1.py -i B.jpg -t A.jpg” for Type 1

“python Type2.py -i B.jpg -t A.jpg” for Type 2

*Replace A and B by Image names A-Source Image   B-Target Image

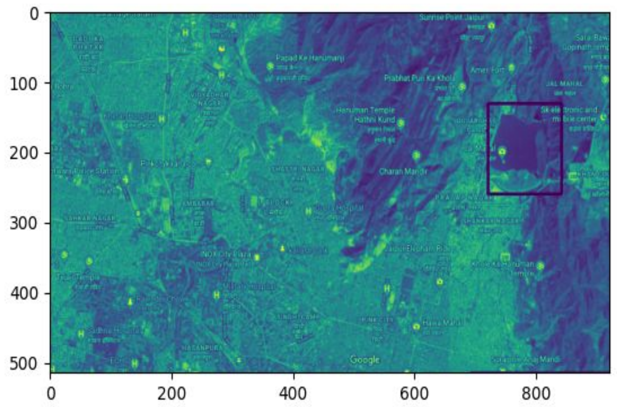
Results

1.Input

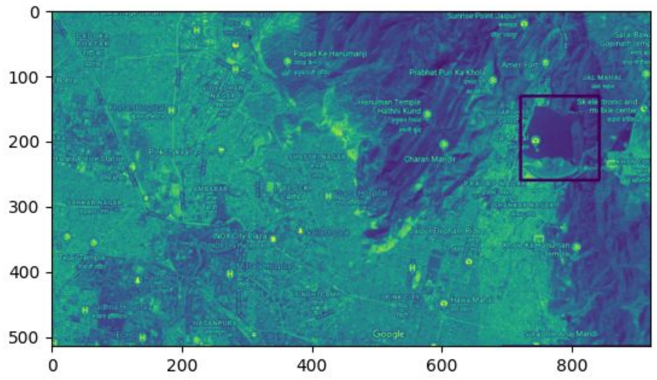


Results

Type I

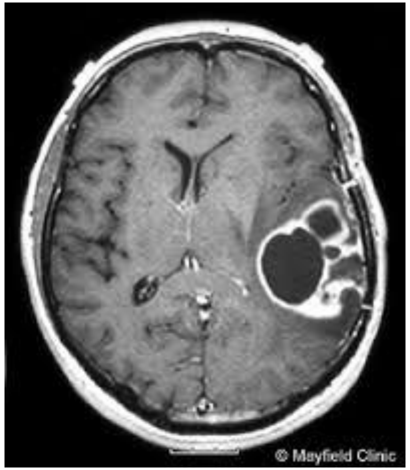


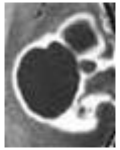
Type II



Results

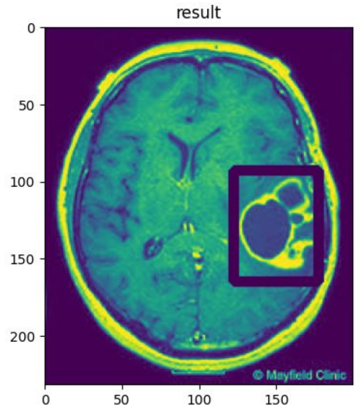
2. Input



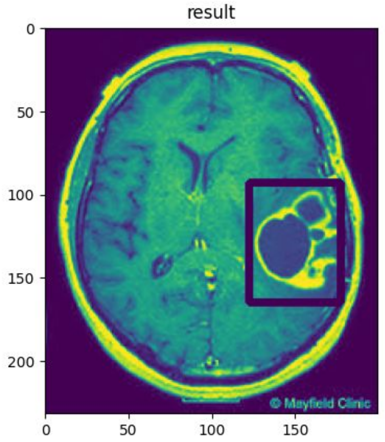


Results

Type I

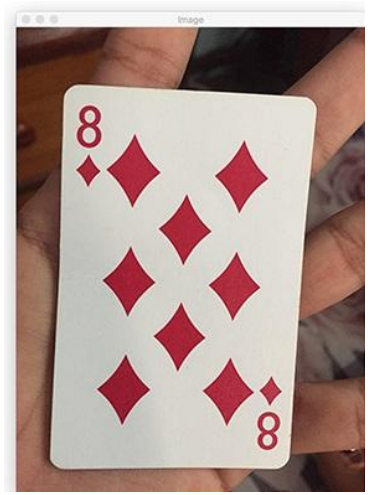


Type II



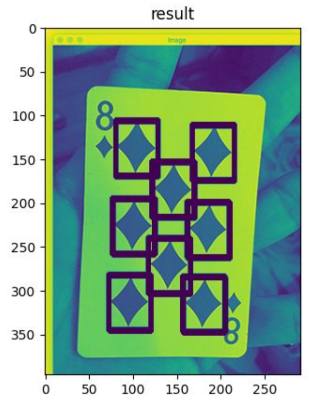
Results

3. Input

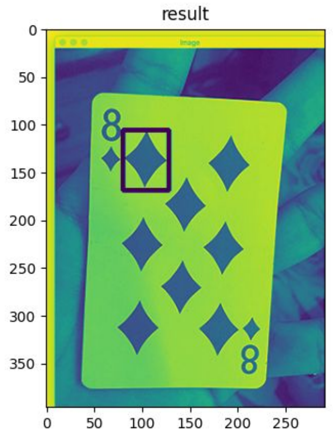


Results

Type I



Type II

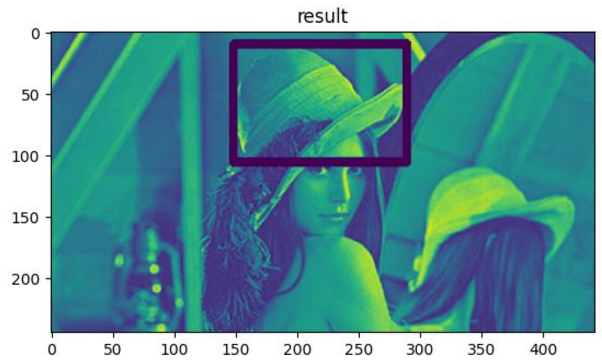


Results

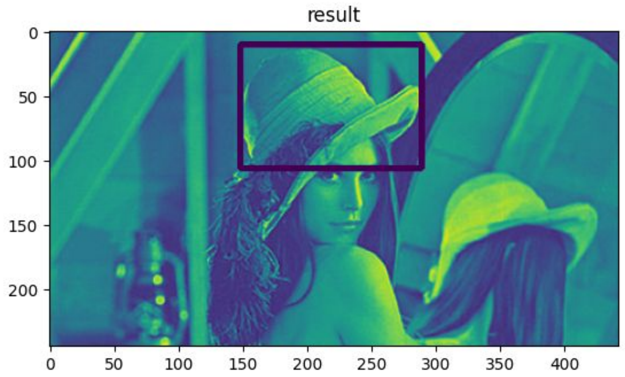
4. Input

Results

Type I

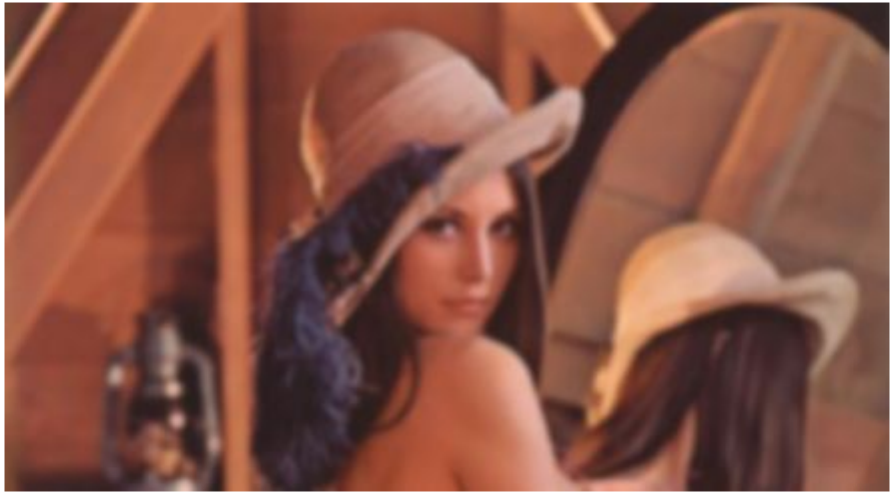


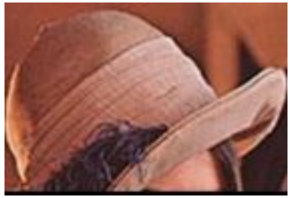
Type II



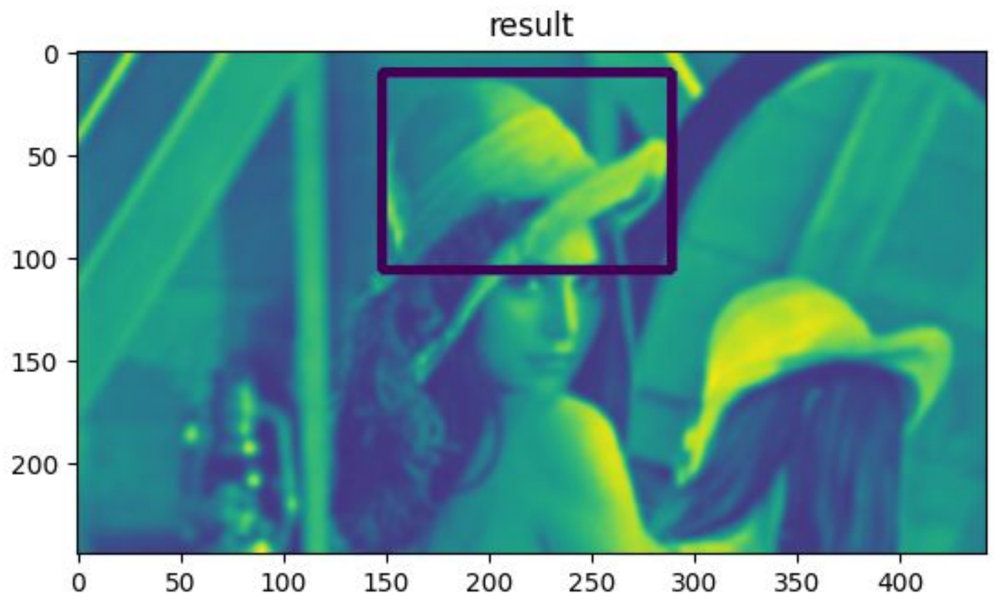
Results

5. Input [Gaussian Blur with Kernel=13]





Results



Observation: Max NCC Value was reduced

Further Possibilities

- By taking advantage of the exhaustive search and high overlap between windows required for high-quality imaging, this method can avoid redundant calculation in motion estimation and thereby significantly reduce the computational cost.
- Finding correspondences always remains a very active area of research. The Registration and correspondences of images in turn is based on cross correlation and normalized cross correlation techniques have been studied the least and they are an active area of research.
- The algorithm can be further developed to achieve conversion to many number systems such as hexadecimal and octal.
- This method could thus prove very useful in real-time motion estimation in medical ultrasound as well as other medical imaging modalities or computational fields.
- The betterment or deterioration of the severity of a patient’s disease, can also be monitored on a time to time basis by NCC.

References

1. https://docs.adaptive-vision.com/4.7/studio/machine_vision_guide/TemplateMatching.html
2. https://www.ijcaonline.org/archives/volume153/number10/swaroop-2016-ijca-912165.pdf
3. https://www.ripublication.com/acst17/acstv10n1_07.pdf
4. https://www.youtube.com/watch?v=ngEC3sXeUb4