Computer Vision Recognition

Tassadaq Hussain

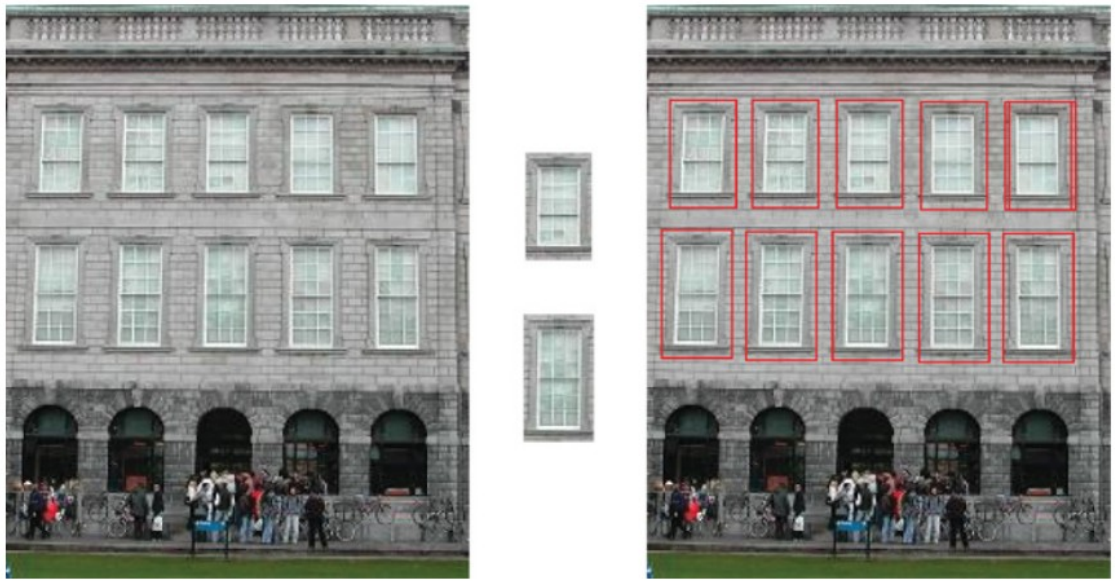
Associate Prof: Riphah Int'l University
Microsoft Barcelona Supercomputing Center Barcelona, Spain
UCERD Pvt Ltd Islamabad

Contents

- Template Matching
- Statistical Pattern Recognition
- Machine Learning Technique
  - Other Recognition Techniques

Template Matching

The Tempalte Matching is a simple technique where a sub-image is searched for within an image.



$$D_{\text{CrossCorrelation}}(i,j) = \sum_{(m,n)} f(i+m, j+n) \cdot t(m,n)$$

Template Matching

```
import cv2
import numpy as np

cap = cv2.VideoCapture(2)
template = cv2.imread("calc_temp.jpg", cv2.IMREAD_GRAYSCALE)
w, h = template.shape[::-1]

while True:
    _, frame = cap.read()
    gray_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    res = cv2.matchTemplate(gray_frame, template, cv2.TM_CCOEFF_NORMED)
    loc = np.where(res >= 0.7)

    for pt in zip(*loc[::-1]):
        cv2.rectangle(frame, pt, (pt[0] + w, pt[1] + h), (0, 255, 0), 3)

    cv2.imshow("Frame", frame)

    key = cv2.waitKey(1)

    if key == 27:
        break

cap.release()
cv2.destroyAllWindows()
```

3.8	2.8	2.4	2	1	0	1	2
3.4	2.4	1.4	1	1	0	1	1.4
3	2	1	0	0	0	0	1
3	2	1	0	0	1	0	1
3	2	1	0	1	1	0	1
3	2	1	0	1	1	0	1
3	2	1	0	0	0	0	1
3.4	2.4	1.4	1	1	1	1	1.4

Chamfer Image

1	1	1	1
1			1
1			1
1			1
1	1	1	1

Template

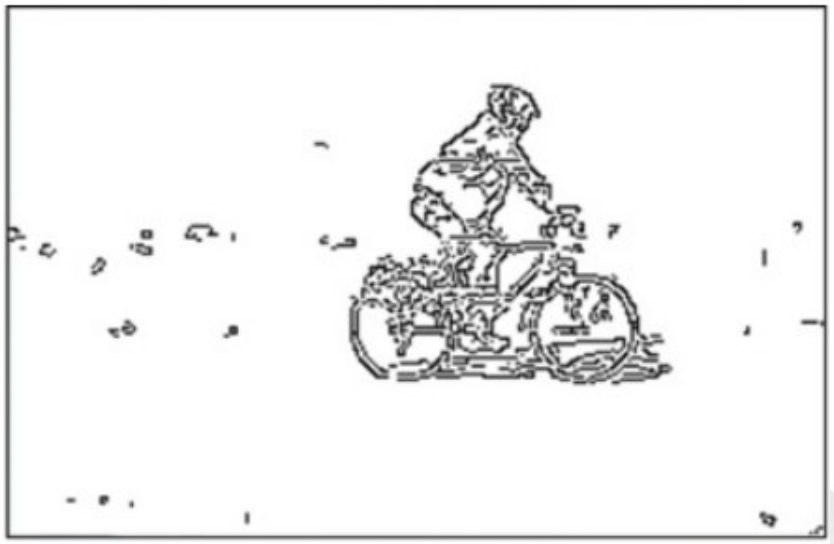
27.4	19.6	12.8	8	27.4
23.2	16.8	10.4	5	9.4
21	14	8	0	6
23.2	16.8	10.4	5	9.4

Matching Space

A contour is a list of points that represent, in one way or another, a curve in an image. Can be different depending on the circumstance at hand. There are many ways to be represented in different ways.

Pattern Recognition

Perform recognition based on the similarity of the features.



Matching Learning

Load Data and Label it

Create Features Extractor

Extract Features

Create Solver (Classification)

Give features to Solver

OBR Features Matching

```
import cv2
import numpy as np

img1 = cv2.imread("the_book_thief.jpg", cv2.IMREAD_GRAYSCALE)
img2 = cv2.imread("me_holding_book.jpg", cv2.IMREAD_GRAYSCALE)

# ORB Detector
orb = cv2.ORB_create()
kp1, des1 = orb.detectAndCompute(img1, None)
kp2, des2 = orb.detectAndCompute(img2, None)

# Brute Force Matching
bf = cv2.BFMatcher(cv2.NORM_HAMMING, crossCheck=True)
matches = bf.match(des1, des2)
matches = sorted(matches, key = lambda x:x.distance)

matching_result = cv2.drawMatches(img1, kp1, img2, kp2, matches[:50], None, flags=2)

cv2.imshow("Img1", img1)
cv2.imshow("Img2", img2)
cv2.imshow("Matching result", matching_result)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

SIFT Matching

```
import cv2
import numpy as np

cap = cv2.VideoCapture(1)


img2 =  cv2.imread("class_temp.jpg", cv2.COLOR_BGR2GRAY)

sift = cv2.xfeatures2d.SIFT_create()



kp_2, desc_2 = sift.detectAndCompute(img2, None)


while True:
    _, img1 = cap.read()
    gray_frame = cv2.cvtColor(img1, cv2.COLOR_BGR2GRAY)
    # ORB Detector
    kp_1, desc_1 = sift.detectAndCompute(gray_frame, None)
    # SIFT Matching
    index_params = dict(algorithm=0, trees=5)
    search_params = dict()
    flann = cv2.FlannBasedMatcher(index_params, search_params)
    matches = flann.knnMatch(desc_1, desc_2, k=2)
    good_points = []
    ratio = 0.6
    for m, n in matches:
        if m.distance < ratio*n.distance:
            good_points.append(m)
    print(len(good_points))
    result = cv2.drawMatches(img1, kp_1, img2, kp_2, good_points, None)
    cv2.imshow("result", result)
    cv2.imshow("Original", img1)
    cv2.imshow("Duplicate", img2)
    key = cv2.waitKey(1)
    if key == 27:
        break

cap.release()
cv2.destroyAllWindows()
```