Inventory change detection project progress report

Jiang Xue
Information Network Institute
4616 Henry St, Pittsburgh

Kaiyue Wu
Information Network Institute
4616 Henry St

Tao Yu
Information Network Institute
4616 Henry St

Abstract

The problem is inspired by the Amazon Fullfillment Technologies Computer Vision Team. Change detection is accomplished by processing digital photographs in order to identify differences in places or other things over time. We need to detect changes in two pictures, which is something removed or added to a bin. Getting the correct answer to this simple problem is incredibly valuable. We could use this technique into the fields of automatic warehourse management.

For pair of images, our goal is to output a score between 0 and 100 that indicates whether an item was removed or added.

1. Methods

1.1. Methods that have been used

1.1.1 Pre processing

We first used histogram equalization to improve the contrast in an image, in order to stretch out the intensity range.

Then we used fastNlMeansDenoisingColored() function, which is the implementation of Non-local Means Denoising algorithm, to get more clear images.

Then we changed the colored image to gray image for further processing.

1.1.2 Feature extraction

We used SIFT(Scale-Invariant Feature Transform) to extract the key points features from two images. Key points between two images are matched by identifying their nearest neighbors. The sample result is shown in Figure 1.1.

But in some cases, the second closest-match may be very near to the first. It may happen due to noise or some other reasons. In that case, ratio of closest-distance to second-closest distance is taken. If it is greater than 0.8, they are rejected.

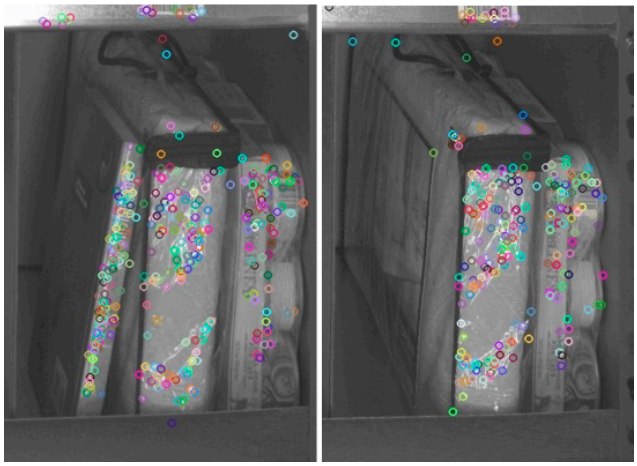

Figure 1.1: feature extraction using SIFT

1.1.3 Feature matching

After the feature extraction, we use Brute-Force matcher, which takes the descriptor of one feature in first set and is matched with all other features in second set using some distance calculation. And the closest one is returned.

We use Brute-Force Matching with SIFT Descriptors and Ratio Test:

```
sift = cv2.SIFT()

kp1, des1 = sift.detectAndCompute(img1_gray, None)
kp2, des2 = sift.detectAndCompute(img2_gray, None)

bf = cv2.BFMatcher()
matches = bf.knnMatch(des1, des2, k=2)

good = []
for m,n in matches:
  if m.distance < 0.75*n.distance:
     good.append([m])

ratio = float(len(good))/float(len(matches))
ratios.append(ratio)
```

1.1.4 Distribution Generation:

For change and unchange image data sets, we run the **sift** and **knnmatch** to calculate a ratio(number of good matched keypoints/total keypoints) for each pair of image. Generate a distribution of ratio value for changed image pairs and unchanged image pairs.

1.1.5 Minimum Probability Of Error Classifier:

We build a minimum error classifier to distinguish changed samples from unchanged samples and find the best ration boundary. For this classifier, we consider only one of the feature: ratio (number of good matched keypoints / total keypoints).

1.2. Methods that plan to use

In the second half of our project, we will keep exploring more methods both on unsupervised and supervised.

For unsupervised method, we have tried using SIFT to extract key points and descriptors, and using these descriptors to do match. In the next step, we will try to use some different key point extraction algorithm. The algorithm are Harris Corner Detector and Shi-Tomasi Corner Detector. Unlike SIFT, Corner Detector like shown in Figure 1.2, finds the corner point rather than key points, as shown in the picture below. This method will have its advantage when there is lightness change. Because corner point is invariant to intensity change. But this could also be a disadvantage when some corner are missing, which is caused by movement of the object.

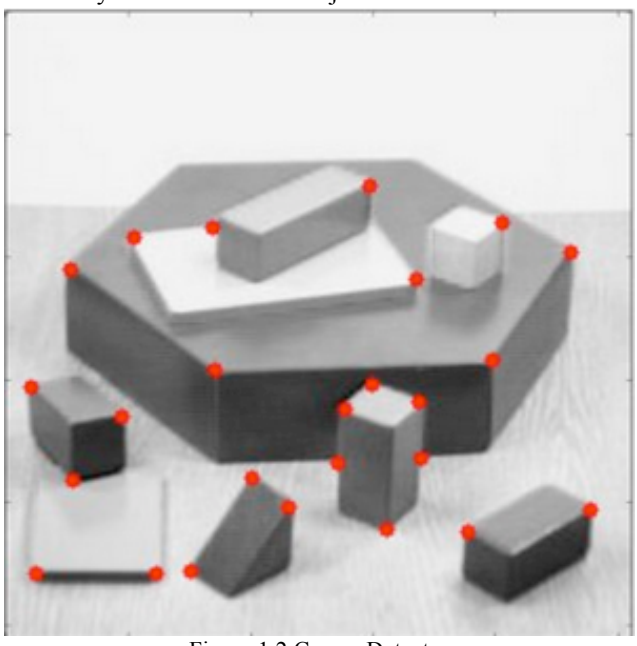

Figure 1.2 Corner Detector

Also, we would like to try a new method called optical flow. Optical flow can be used to track the movement of objects between different frames in a video. In our project, we will take use of function **cv2.calcOpticalFlowPyrLK()** from the opencv library. The function will take a set of key point and the two frames as input and return 0 if the corresponding key points could be found in the next frame or 1 if not. This could be used to determine if an object is missing in the next frame. This could also become a binary feature for the classifier in the supervised implementation below.

For supervised method, we will try to use some classifier that will take what we extract using SIFT and Harris Corner Detector as features. Currently, we have tried employing minimum error classifier on one-dimension data. Next, we will try to add the dimensionality from one to two or three. The newly added feature will be the new descriptor extracted by Harris Corner Detector or Shi-Tomasi Corner Detector. We may also want to add the third dimension with some cognitive parameters. By cognitive parameters, I mean a value that will include some metric such as difference between the overall intensity, or simply the count of key points.

The first classifier we would like to try is SVM. With different kernel functions, SVM could perform as a non-linear classifier and give a very good performance. SVM can also give a very decent performance when number of training data is limited, which is our current case. Also, we will try to implement LDA for classification.

2. Data

We have three sets of paired images. Each set are separated to two parts of images. One part contains changed images, and the other part contains unchanged images.

2.1. Set 1: add a gray square to original images

In set 1, there are 26 pairs of unchanged images and 25 pairs of changed images. The only difference between original images and changed images is that a changed image has a gray square. The sample data is shown as Figure 2.1. And for a pair of unchanged images, they are the same.

Figure 2.1: changed sample pair in Set 1

2.2. Set 2: add a smaller gray square to original images

Similar with Set 1, in Set 2, there are 26 pairs of unchanged images and 25 pairs of changed images. The only difference between original images and changed images is that a changed image has a gray square. However, the square is smaller than that of Set 1. The sample data is shown as Figure 2.2. And for a pair of unchanged images, they are almost the same, but have slight differences.

Figure 2.2: changed sample pair in Set 2

2.3. Set 3: remove or add items

In set 3, there are 51 pairs of unchanged images and 51 pairs of changed images. The difference between original images and changed images is that in a changed image, an item was removed or added. The sample data is shown as Figure 2.3. And for a pair of unchanged images, no item was removed or added, but the angle or the light of shooting might be different. The sample data is shown as Figure 2.4.

Figure 2.3: changed sample pair in Set 3

Figure 2.4: unchanged sample pair in Set 3

3. Initial results from techniques already implemented

Distribution of ratio with changed and unchanged image samples:

changed_ratio = [0.167442, 0.0661157, 0.235023, 0.392857, 0.107143, 0.0592885, 0.104651, 0.226087, 0.237288, 0.210383, 0.401709, 0.165217, 0.177936, 0.254335, 0.11985, 0.269663, 0.114144, 0.345411, 0.297959, 0.106667, 0.224138, 0.29902, 0.0806794, 0.348624, 0.267241, 0.162011, 0.304721, 0.166667, 0.257143, 0.481013, 0.313333, 0.320359, 0.257962, 0.286726, 0.06621, 0.392226, 0.309735, 0.367816, 0.398693, 0.445333, 0.118182, 0.103352, 0.019305, 0.0609756, 0.163636, 0.252252, 0.173516, 0.128686, 0.388679, 0.394309, 0.37149, 0.555076]

unchanged_ratio = [0.408859, 0.554974, 0.405512, 0.35443, 0.439883, 0.447183, 0.438424, 0.441406, 0.363636, 0.444109, 0.44878, 0.507776, 0.457778, 0.430669, 0.434447, 0.428571, 0.541497, 0.328713, 0.462937, 0.425926, 0.342205, 0.239286, 0.417219, 0.396501, 0.4375, 0.43761, 0.493506, 0.418919, 0.21843, 0.185185, 0.581132, 0.44473, 0.4, 0.474114, 0.460177, 0.452282, 0.446602, 0.330677, 0.467742, 0.321429, 0.503226, 0.470852, 0.325967, 0.432161, 0.375, 0.379147, 0.311377, 0.333333, 0.470817, 0.579787, 0.279412, 0.492647]

Minimum Probability Of Error Classifier as Figure 3.1

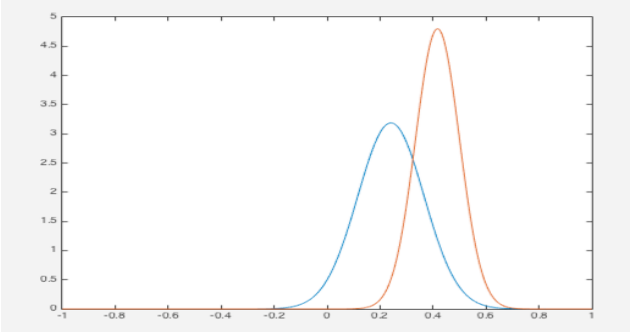

Figure 3.1: changed sample pair in Set 3

When we set the ratio boundary with 0.310942, the total probability of error is 27.8%.

4. Expected results

We think that the final result from our supervised classifiers will perform slightly better than unsupervised ones. The reason is that the dimensionality of our data is very low, and data tend to be very sparse and separate, in which case, a threshold/classifier could be totally unrelated to the training data. We even think, in some situation, the unsupervised classifier will give better performance, because training data will impose some incorrect bias to our classifier, thus affect our final performance.