Project 3

Gaurav Bagwe

Contents

1 Introduction

The aim of the project is to detect 'scramble game' letters in the English Alphabet. The test image 2 consists of all the letters and has some skew. However, test image 1 consists of subset of the letters of alphabet.

Observing the images, the letters are taken from a diff source and the test images are taken from a diff camera. Which results in inconsistency in contrast, sharpness and features. The processing will need to resize the image and remove the rotation to better match the features. The use of SURF features is excellent choice since SURF features are fast, and are shift and scale invariant.

2 Preprocessing

. The objective detect features from the two set of test images given in the problem description .The letters must be segmented out from the test image. Images consists of three main set of colors, the background being white, Letters are black and scramble block is brown. We take the ROI to set a threshold that would filter the blocks. The dilatation, erosion is done to remove shadows.

2.1 Segmentation

The algorithm shown in Fig 2 is implemented for image segmentation in test image.

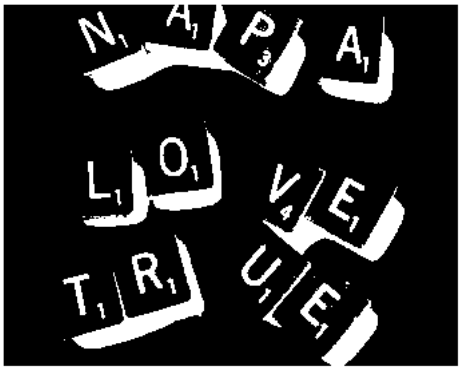

Figure 1 Selection of threshold

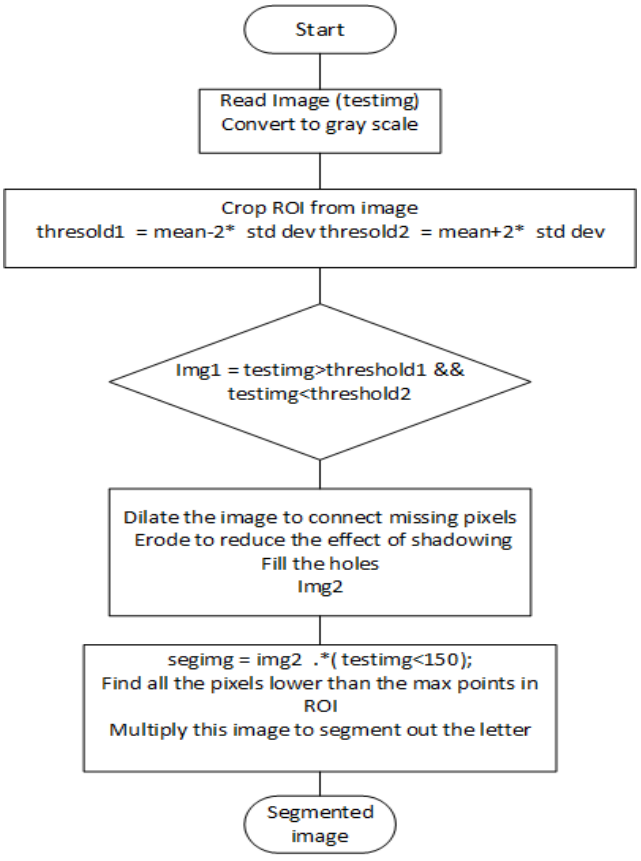

Figure 2 Segmented Image

The results of the preprocessing the both the test images give the following results shown in Fig 3 and Fig 4

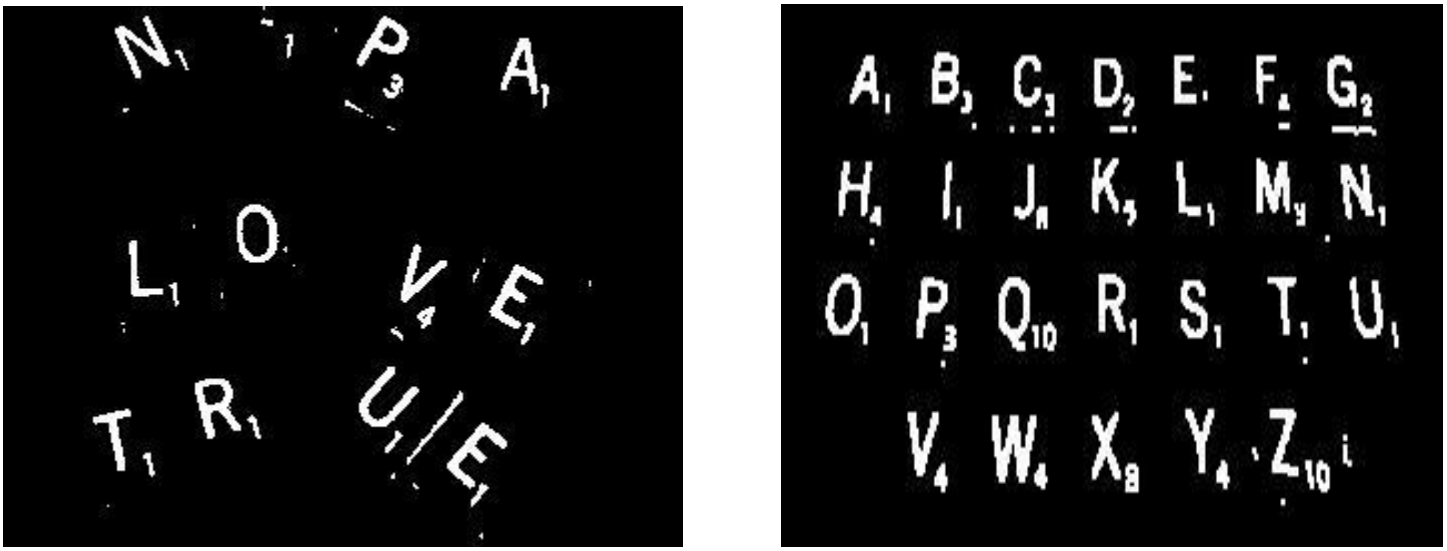

Figure 3 Segmented Test Image 1

Once we have all the major block we apply an area filter which we can remove the unwanted noise. Which further enhances the image.

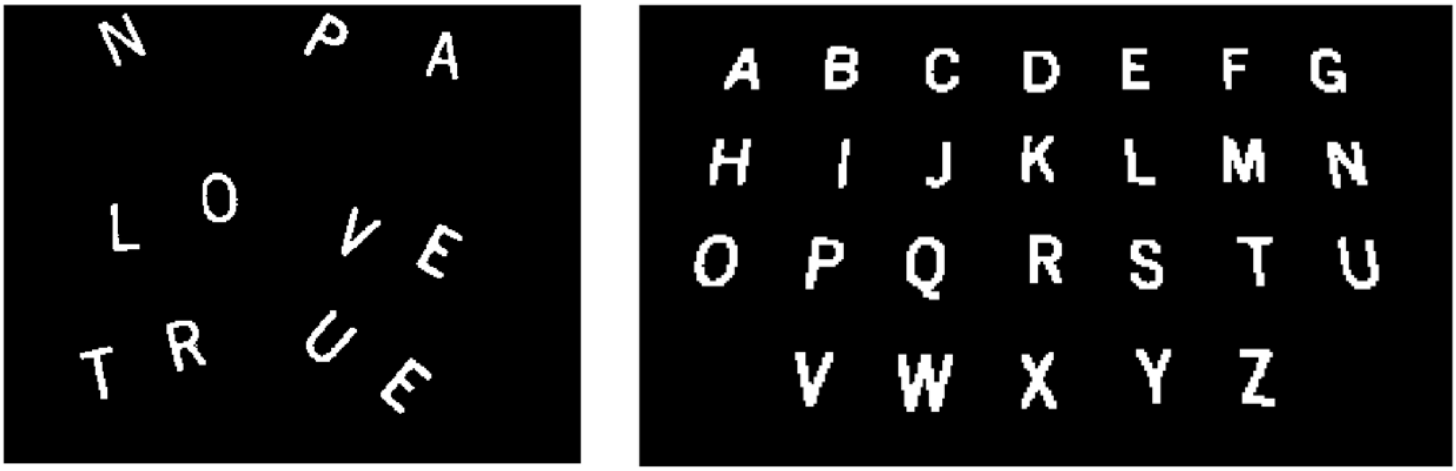

Figure 4 Final Preprocessed Test images

Further, Use of Region Properties lets to segment out the letters from the whole image. MATLAB inbuilt *regionprops block* is very through and provides various kinds of information for blob analysis. *Boundingbox* in regionprops returns the object containing dimensions needed to for smallest rectangle needed to fit the blobs. These dimensions are then used to crop the image resulting in segmented letters.

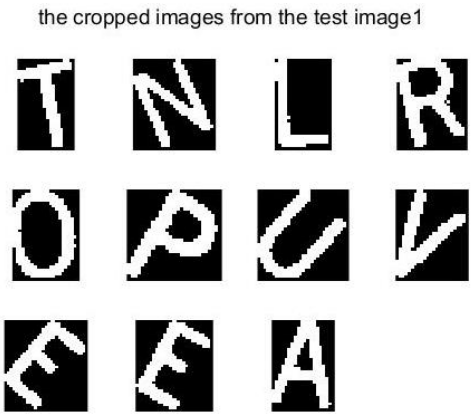

Figure 6 Cropped letters from test image 1

Figure 5 Cropped letters from test image 1

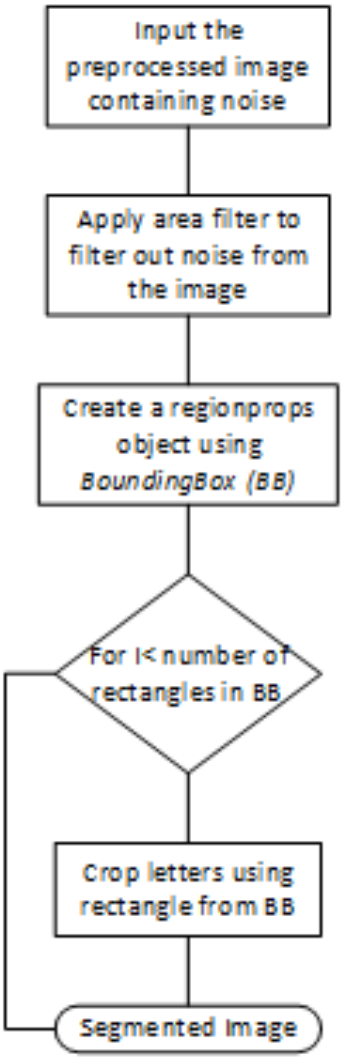

Figure 7 Algorithm to segment letters

2.2 Alignment Rotation Correction

Often the letter segmented from the test images may be rotated. The Surf and Shift features are though shift and scale invariant. However, we still can correct rotation and size of the image to get more matched features when comparing to the letters.

The Region Props inbuilt in MATLAB contains another property of Orientation. The angle between x-axis and the major axis of the ellipse that best fits the image is calculated as shown in Fig . 10. This can be readjusted back and the final image is shown in Fig 8. Fig 8 contains the titled letters obtained from the test1 on left and the corrected image on the Right.

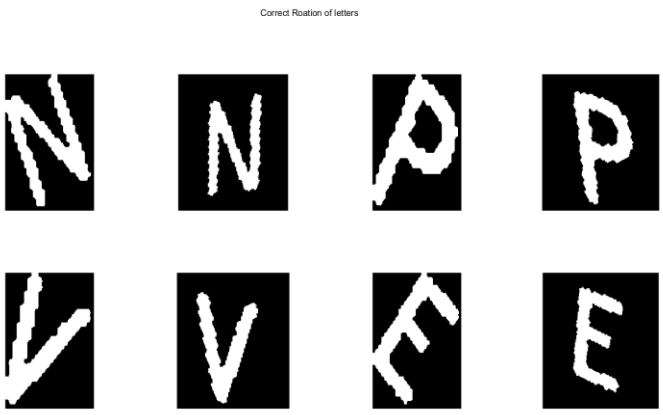

Figure 8 Angle Correction on the Segmented Image

Figure 10 Ellipse fitting for region props in MATLAB

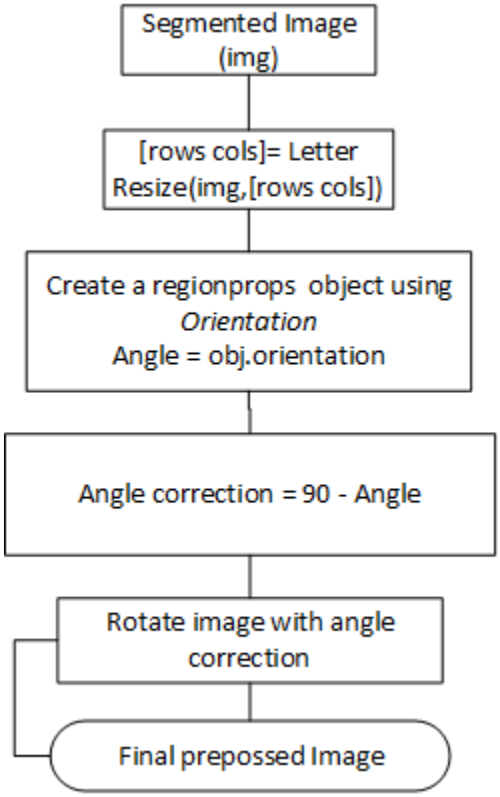

Figure 9Algorithm for angle correction

2.3 Processing on the set of testing Images

The Problem has a subset of alphabet which are to be matched the segmented images we obtained for the images in test 1 and 2. Segmentation on letters is relatively easier compared to test images as they are bigger and have less noise.

Histogram reveals that letter are contained in black so threshold less than 100 in gray scale will filter out the background. Fig11 shows the image obtained after threshold. The letters contain numbers as subscript. We can further remove the numbers that are in smaller areas using *regionprops.area* .

The *region props* object returns the table or areas of each blob present in the image. It an absolute propbabilty that the letter of interest will have maximum area in all the blobs which contain the number as well in the form of subscript. Fig 12 shows the output obtained on selecting the max area in the blob, thus cropping the image.

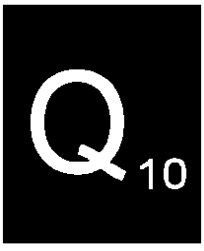

Figure 12 Letter after threshold

Figure 11 Final letter after removing subscript

3 Processing

The processing to detect the letters is done in 3 steps. First, we need to detect the interest points in the letters and the test image it needs to be compared. Second, we extract the features and match the features in the test image and the letter. Lastly, once the features are matched we set a threshold to filter out relevant features that detect the letter.

3.1 Detect Surf points

There are various method to detect features from an image. Speed up Robust Features (SURF), BRISK, SIFT, MSER, Harris, Eigen, FAST,HOG are some of the inbuilt subroutines in MATLAB.

The SURF features are shift invariant and several orders faster than SIFT. The first step while using any feature matching method is to detect interest points. The points returned are dependent on the value of the threshold set. Lower the threshold More surfpoints are detected. NumOctaves control the size of blobs which needed to be detected. Adjusting these parameters in MATLAB built in function returns features.

Fig 13 shows the Strongest 10 Surf features found in the Letters, which would be later compared with the testing images.

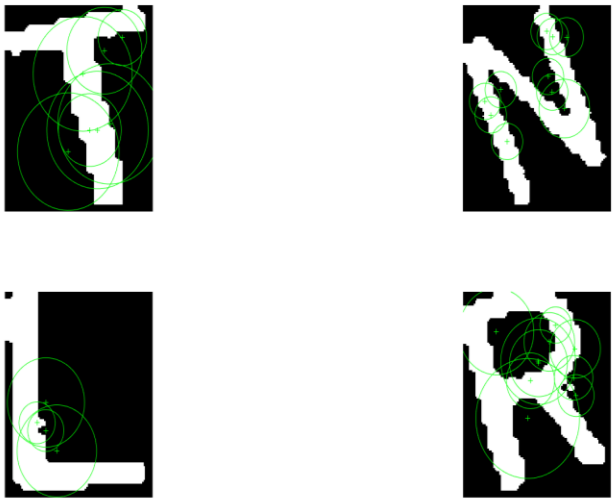

Figure 13 Surf feature points

Similarly, the surfpoints are obtained from the Test Letters set.

The Return type of subroutine *detectSURFfeatures* in MATLAB is a Struct Object, which contains the Scale, Sign of Laplacian, Orientation, Location, Metric and Count of the number of points. Each detect Surf point contains the above information. Further, more Features need to extracted which are than matched.

MATLAB contains other subroutines to extract Features.

3.2 Extract Features from the image

The Sub routine returns the descriptor their locations. Depending on the Feature method selected to obtain points the descriptors area calculated. The axis property of the object selects the scale of descriptors meaning the area of the circle, which resembles closest to ellipse form MSER features. The scale of the features is calculated form eq 1.

$$scale = (0.25) * \sqrt{(0.5\ major\ axis\ .*\ 0.5\ minor\ axis)} \qquad \text{.... Eq(1)}$$

Since, the SURF method is independent from scaling the features.

3.3 Feature Matching

Extract features returns MxN matrix containing information of the valid points. The object of the process is to match similar features and return the corresponding indexes pointing to the location of feature. A pairwise distance between each features is calculated. This distance can be sum of Absolute Differences(SAD) or sum of Squared distances(SSD). Under default conditions, subroutine uses SSD. The method specifies how nearest neighbors between features1 and features2 are found. Two feature vectors match when the distance between them is less than the threshold set by the MatchThreshold parameter. Fig 15 from MATLAB helps to understand feature matching.

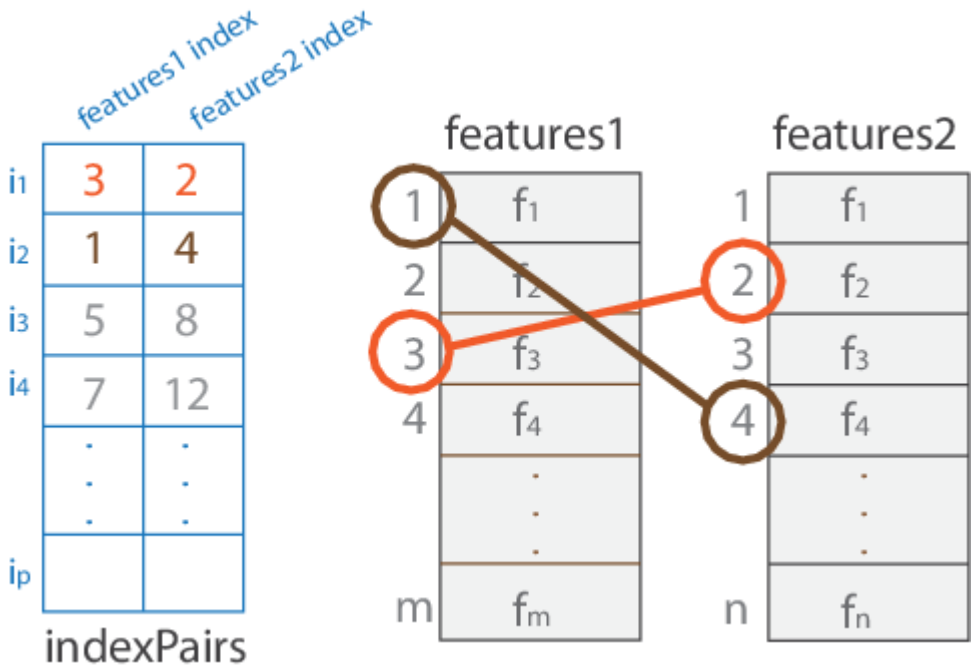

Figure15Match Features

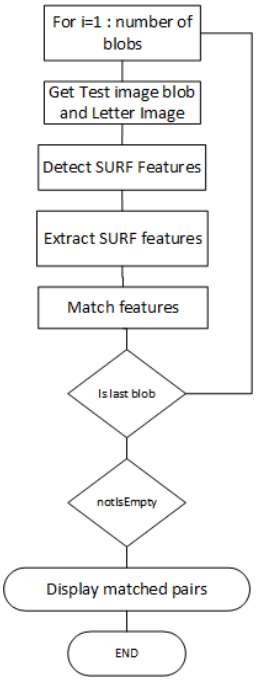

Figure 14 Processing and detection

Index pairs acts as pointers pointing to the features, which are relevant after features matching. *Showmatchfeatures* displays the matched pairs in the image as shown in fig 16

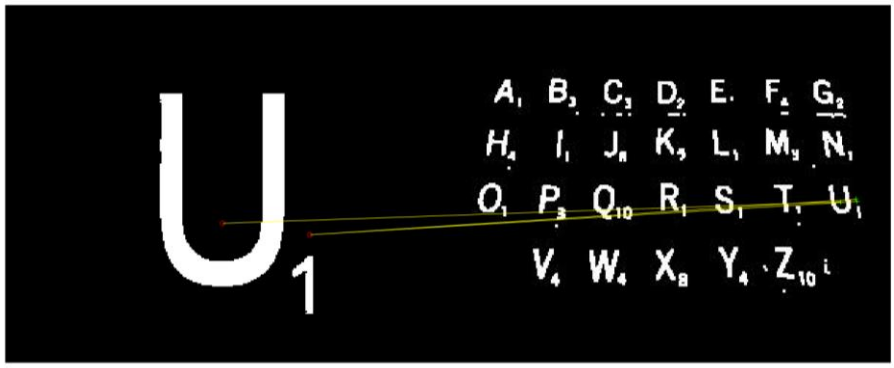

Figure 16 Feature Matching example

3.4 Threshold Selection

Selection of Threshold important throughout the preprocessing and Processing. It helps in removes noise, reducing false and missed detection. Histogram is used to set up threshold to detect letters and separate it from the background. Proper selection results in fewer noise and fewer shadow effects. Comparison between Fig 1 and Fig 3 shows the importance in the threshold selection. Fig 1 contains image which has effects of shadow and need more processing to remove it.

A proper threshold is also selected to area filter to keep the images greater than threshold. We can find the argmin (ROI) to find the minimum area for the threshold.

Importance of threshold is much more import in the processing of the features.

The Subroutines *detecSURFfeatures , extractfeatures and matchfeatures* are highly dependent on the threshold selection for blob analysis. Decreasing the Metric threshold results in the weaker but more blobs. Number of Octaves is controls the size of the blobs.

As discussed earlier, the match features works on finding the distance using SSD or SAD. A nearest neighbor search results or Pairwise distance calculation results in a detection problem. The distance is compared to a threshold. The tolerance or the variance of the in the distance also controls the number of features detected. A proper threshold of the these two is necessary to match as many correct features resulting in fewer errors. The effects of threshold are discussed in the next section.

4 Results

The detected letters on the test images are calculated as the max of strongest features. When the thresold is low and more tolerance is allowed the more features are matched as shown in fig 17. This results to more errors and false detection of letters.

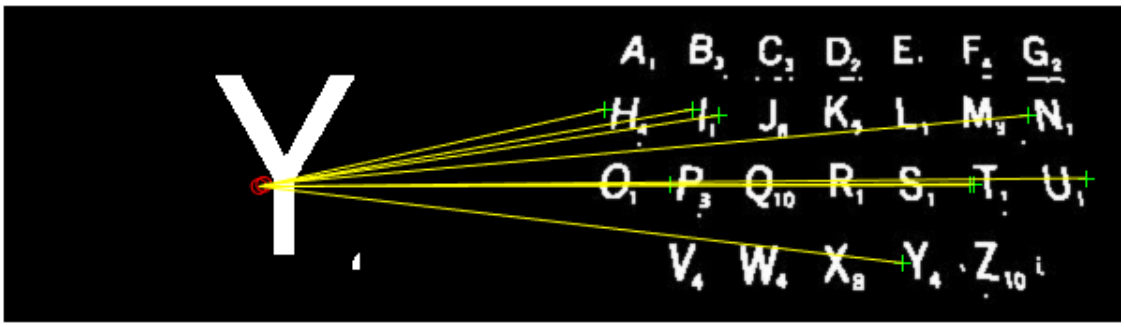

Figure 17 Low threshold

If we set the proper threshold, we can detect letters correctly as shown in fig 16 and fig 18.

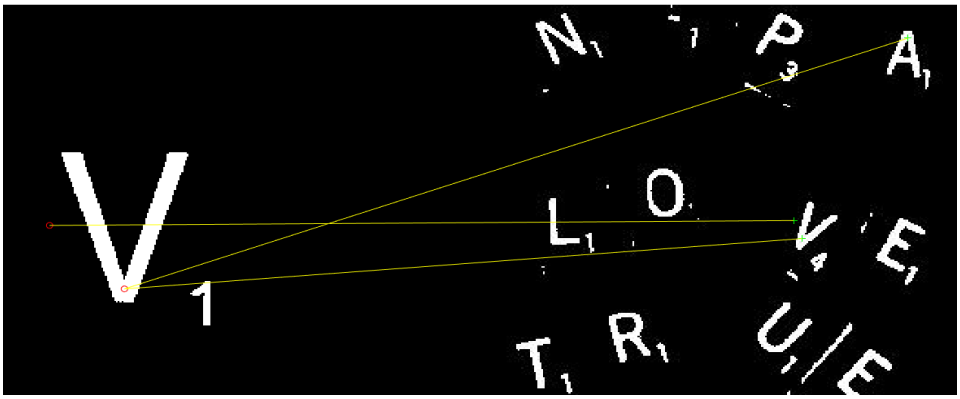

Figure 18 detecting letter V

Problems due to similar features

Many letters have similar features. It can be observed in fig 18 that the corners in V are similar to the corners in A. Hence, the feature matches both to V and A. Fig 18 was processed in presence of the Subscript to see the performance of the process in presence of these features.

The effect of subscript and observation that SURF is Scale invariant is shown in Fig 19. Q contains a sub script number '0', which matches highly with the letter 'O'. This results to a false detection. Furthermore, the hollow space in Q matches to hollow space in P and lower curve in) matches to U. These errors can be reduced simply by setting a higher matching threshold, which creates more strict detection for distances. Fig 20 detects letter 'R' higher matching.

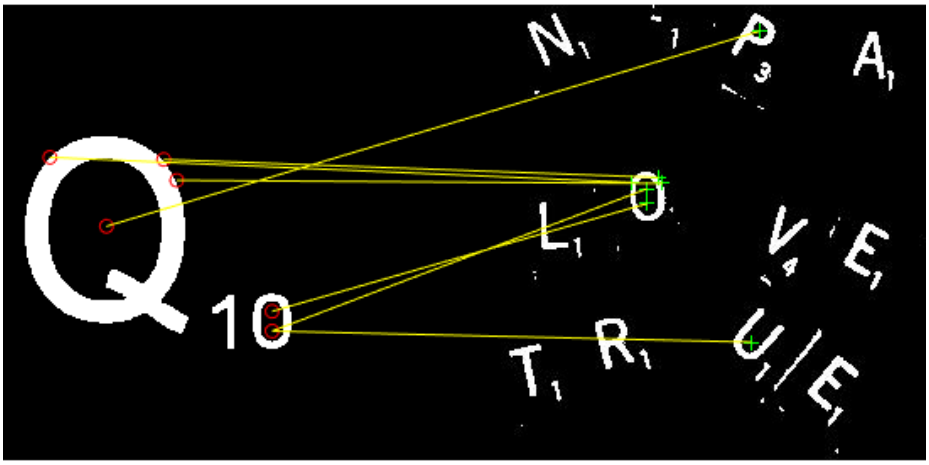

Figure 19 detection of letter Q

Results after thresolding

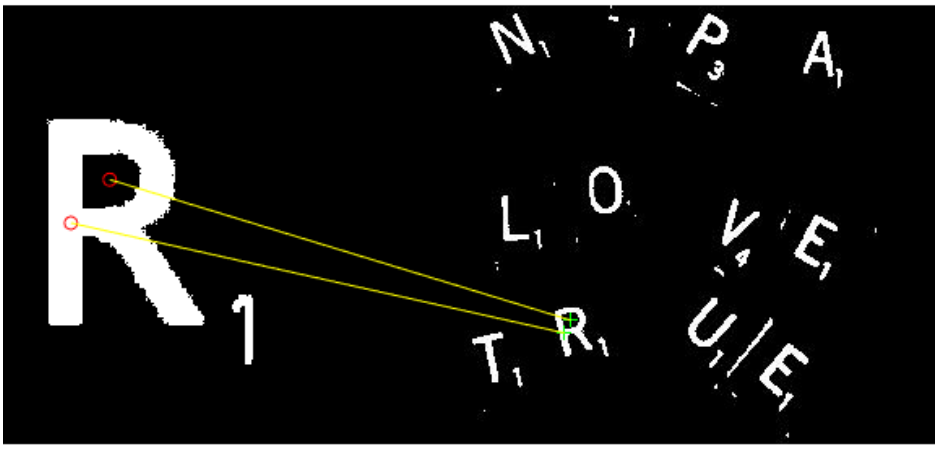

Figure 20 detection of letter R

The errors is reduces by processing and removing the subscript. This results in fewer irrelevant features. The results of further processing is shown Fig 4 and Fig 8.

Figure 21 Detection of some letters

Confusion Matrix

The system detects almost all the letter in the subset. The problem arises as discussed above, the features in V matches that of W and A. Similar problems occur for I, K, L. The confusion matrix shown in Fig 22 shows the table where the letters are matched to the letters in the test image.

Confusion Matrix	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	0	1	1	0	0	0	1	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0
C	0	0	1	1	0	0	1	0	0	0	0	0	0	0	1	0	0	0	1	0	1	0	0	0	0	0
G	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
H	0	0	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
I	0	0	0	1	0	0	0	0	1	1	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0
J	0	0	0	1	0	0	0	0	1	1	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0
K	1	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
M	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0
N	1	0	0	0	0	1	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1	0	0	0	1	0
Q	0	0	0	0	0	0	1	0	0	0	0	0	0	1	1	1	1	0	1	0	1	0	0	0	0	0
R	0	1	0	0	1	1	0	0	0	1	1	0	0	0	1	1	0	1	0	0	0	0	0	0	0	0
S	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0
T	0	0	0	1	0	0	0	1	1	1	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0
U	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
V	1	0	0	0	0	1	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	1	1	0	1	1
W	1	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	1	0	1	0	0
Y	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	

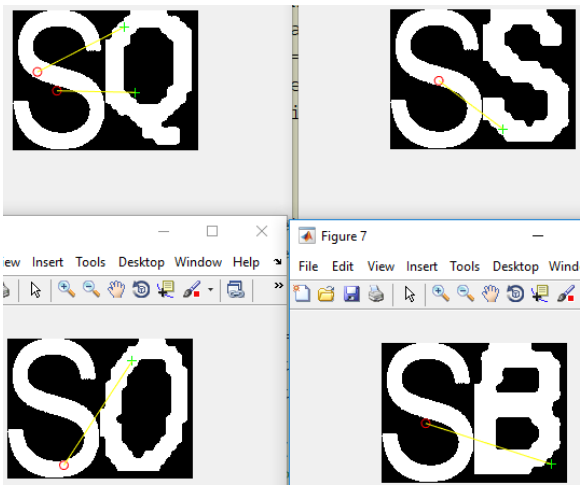

Figure 22 Letter S matched with features in test 2

Figure 23confusion matrix test2

Confusion Matrix	N	P	A	L	O	V	E	T	R	U
B	0	0	0	0	0	0	0	0	0	0
C	0	0	0	0	1	0	0	0	0	0
G	0	0	0	0	0	0	0	0	0	0
H	0	0	0	0	0	0	0	0	0	0
I	0	0	0	1	0	0	0	0	0	0
J	0	0	0	1	0	0	0	0	0	0
K	0	0	0	0	0	0	0	0	1	0
M	0	0	0	0	0	0	0	0	0	0
N	1	0	0	0	0	1	0	0	0	0
Q	0	0	0	0	0	0	0	0	0	0
R	0	0	0	0	0	0	0	0	1	0
S	0	0	0	0	0	0	0	0	0	0
T	0	0	0	1	0	0	0	1	0	0
U	0	0	0	0	0	0	0	0	0	0
V	1	0	0	0	0	1	0	0	0	0
W	0	0	0	0	0	0	0	0	0	0
Y	0	0	0	0	0	0	0	0	0	0

Figure 24 confusion matrix test1

Fig 23 show the number of letters, letter S matches in the testing image. Out of which S is correct detection while all other are false detection. Each sub image is for example Q was matched given that S was tried to match. Since the letter we tried to check are limited, the testing of a letter can only be checked once. This process can be repeated with more sample images and testing images, resulting in a more complex and informative confusion matrix. The current confusion matrix shown in Fig 21, 23 show false detection or matching of the features. Probability can be calculated for detection as

$$P_{G|N} = \frac{num\ of\ times\ G\ was\ matched\ with\ N}{Sum\ of\ all\ the\ time\ N\ was\ matched}$$

Overall, the system performs well for detecting the image with feature matching. The limitation can be removed with training various letters, or if the Images were taken from the same camera, resulting in similar features, resolution and information.