CS482 Final Project Tutorial

Anup Sinkar

May 2024

1 Introduction

For my project I chose to do gesture recognition using Bag of Words. The idea for my project started out as gesture recognition for device input, due to time and assignments from other classes, it became just gesture recognition using bag of words. Gesture recognition is something that would be worth doing, because its cool and because the applications of it could range from disability aids to cool tech products like the Apple Vision Pro's Gesture Controls or BMW's Infotainment Gesture Control. This project focuses on sign language due to data limitations which are explained in background info. This tutorial would help guide someone through the Bag of Words method and let them experiment with gesture recognition.

2 Related Work

There are quite a lot of similar implementations of gesture recognition using Bag of Words. One of these used high level feature histograms and SVM for a classifier to detect and recognize human actions.[6] Another paper used a Kinect to get both color and depth information along with SPD matrices to do one-shot gesture recognition.[2] Another paper looked at the difference between SIFT and SURF on gesture recognition and also used SVM for classification, they found that SURF was better.[3] Another paper that I looked at used a Kinect and attempted real-time gesture recognition using Bag of Words and machine learning.[1] The other paper that I looked at used Bag of Features and used 3D histograms, Kmeans++, and SVM for their dynamic gesture recognition.[5]

3 Background Info & Methods

For this project, due to time limitations, I chose to look for a dataset with gestures, I then narrowed it down to be static gesture for space limitations. So, for this project, I used the Static Hand Gesture ASL Dataset from IEEE Data-port[4], it contains about 960 images of 24 letter in sign language (j and z are not included as they are apparently dynamic gestures rather than static), the data set includes the original videos, frames, the gray-scale frames, the binary frames, and the rotated & scaled gray-scale frames.

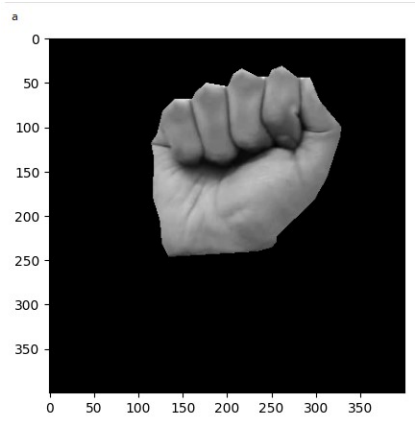

Figure 1: Example of gray-scale image (letter a).

For the project I used the regular gray-scale frames as they isolated the hand and gesture from the rest of the image. I started by using the os library to iterate through all the images and load them with OpenCV in gray-scale and get the ground truth label from the image file name.

```
# get the images from the folder
def getImages(dirpath):
    dpath = os.path.join(os.getcwd(), dirpath)     # make full directory path
    imgs = []
    letters = []

    for img in os.listdir(dpath):
        imgpath = os.path.join(dpath, img)
        letters.append(img[:1])                # get image letter from file name
        imgs.append(cv.imread(imgpath, 0))    # add in gray scale image

    return imgs, letters

imgs, letters = getImages(r"Dataset_ASL_Hand_Gestures\Dataset_ASL_Hand_Gestures\rotated_scaled _grayscale_frames")
plt.figure()
plt.imshow(imgs[0], cmap='gray')          # print an image
print(letters[0])
```

Figure 2: Loading the Images.

Then to get the features from the images, I used both ORB and SIFT to experiment with how differently they would perform; for these I looked to the tutorials from the OpenCV site to help implement them,ORB Tutorial & SIFT Tutorial.

```
# get the features from an image with ORB
def getFeaturesORB(img):
    kp = orb.detect(img, None)
    kp, des = orb.compute(img, kp)    # get the features of the image with orb

    return kp, des

# get features from image with SIFT
def getFeaturesSIFT(img):           # get features with sift
    kp, des = sift.detectAndCompute(img, None)

    return kp, des
```

Figure 3: Detecting features.

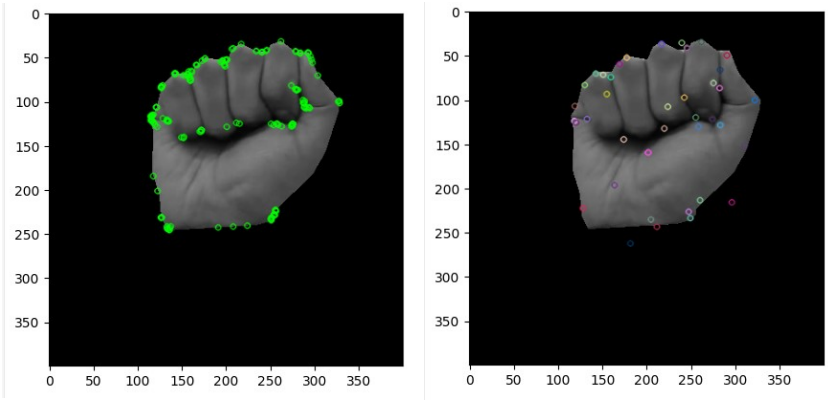

Figure 4: Feature Detection w/ ORB (left) and SIFT (right).

I then used the examples in the sklearn model_selection library to help implement splitting the data into train and test data. I also used sklearn's KMeans function page to help implement the clustering of the features and building of the dictionary.

```
#kmeans clustering
kmeans = KMeans(n_clusters = 100, random_state = 1, n_init = 'auto').fit(i_train_f)
dictionary = kmeans.cluster_centers_
```

Figure 5: KMeans Clustering.

To build the Bag of Words vectors for each of the images I used Numpy's norm function to calculate distance from the image features to the dictionary words to create each image's histogram. I then used sklearn's KNeighborsClassification function documentation to create a KNN classifier.

```
#build BoW vectors
def getBoWvector(img, dictionary):
    hg = np.zeros(len(dictionary))    # create empty histogram

    kp, des = getFeaturesORB(img)        # get features
    num_features = len(des)

    for feature in des:                        # for each feature in image
        dists = []
        for word in dictionary:                    # for each word in the dictionary
            dists.append(np.linalg.norm(feature - word))        # get distance

        cluster = np.argmin(dists)    # nearest word
        hg[cluster] += 1

    hg /= num_features            # normalize histogram

    return hg       # return histogram
```

Figure 6: Encoding Bag of Word Vectors.

```
KNeighborsClassifier
KNeighborsClassifier(metric='euclidean', n_neighbors=2, weights='distance')
```

Figure 7: KNN Classifier.

I trained the classifier on the training images' Bag of Words vectors. I then created the Bag of Words vectors for the test images and then used the classifier to predict the letters for the gestures in the images. I then used the following equation to calculate the accuracy that I was getting from the predictions, more on this in results.

$$Accuracy = \frac{\#\text{ correctly predicted}}{\text{number of predictions}}$$

After doing this, I experimented with different hyper-parameters and the two feature methods as well as the other versions of the images provided with the dataset. The implementation of my project is not to dissimilar to the concepts that we learned in class. For Bag of Words we learned the the very high-level steps for it (Dictionary Encoding, Bag of Words Encoding, and Classifying), for my project, I just implemented everything and experimented with the different hyper-parameters/methods that could have been used for gesture detection.

4 Results & Future Work

The results that I first got were not great, I started with 20 clusters for Kmeans and 20 neighbors for KNN, this resulted in an accuracy around 50%. I then experimented with different parameters, I found that using very few neighbors was more accurate and found 2 neighbors to net the best result. I also found that for the clusters, that a higher amount was good, but not too high, and found 100 to be the most optimal amount of clusters. Using 2 neighbors and 100 clusters, I was able to get the accuracy to 96.53%.

```
['g' 'w' 'l' 'k' 'i' 'h' 'w' 'g' 'a' 'b' 'q' 'i' 'e' 'c' 's' 's' 'c' 'u'
 'p' 'c' 'f' 'k' 't' 'c' 'r' 's' 'g' 'k' 'a' 'p' 'k' 'f' 'b' 'k' 'a' 'l'
 'h' 'o' 'f' 'n' 'y' 'o' 'f' 'o' 'e' 'r' 'h' 't' 'h' 'x' 'c' 'd' 'y' 'l'
 'l' 'x' 'y' 'w' 't' 'x' 'm' 'q' 'a' 'n' 'm' 'y' 'o' 'k' 'g' 'q' 'e' 'n'
 'o' 'i' 'n' 'm' 'y' 'd' 'y' 'b' 'v' 'w' 't' 'k' 'd' 'p' 'a' 'r' 'p' 'o'
 's' 'q' 'x' 'g' 't' 'c' 'r' 'h' 's' 'a' 'n' 'b' 'u' 'o' 'u' 'u' 'g' 'x'
 'h' 'u' 'd' 't' 't' 'q' 'a' 'd' 'b' 'l' 'i' 'm' 'y' 'c' 'k' 'x' 'e' 'l'
 'r' 'g' 'g' 'o' 'u' 'n' 'a' 'u' 'b' 'y' 'l' 'i' 'c' 'w' 'e' 'd' 'g' 'l']
['g', 'w', 'l', 'k', 'i', 'h', 'w', 'g', 'a', 'b', 'q', 'i', 'e', 'c', 's', 's', 'c', 'u', 'p', 'c', 'f', 'k', 't', 'c', 'r', 's', 'g', 'k', 'a', 'p',
'k', 'f', 'b', 'k', 'a', 'l', 'h', 'o', 'f', 'n', 'y', 'o', 'f', 'i', 'e', 'r', 'h', 's', 'h', 'x', 'c', 'd', 'y', 'l', 'l', 'x', 'y', 'w', 't', 'x',
'm', 'q', 'a', 'n', 'm', 'y', 'o', 'k', 'h', 'q', 'e', 'n', 'o', 'n', 'n', 'm', 'y', 'd', 'y', 'b', 'v', 'w', 't', 'k', 'd', 'p', 'a', 'r', 'p', 'o',
's', 'q', 'x', 'g', 't', 'c', 'r', 'h', 's', 'a', 'n', 'b', 'u', 'o', 'u', 'u', 'g', 'x', 'h', 'u', 'd', 't', 't', 'q', 'a', 'd', 'b', 'l', 'i', 'm',
'y', 'c', 'r', 'x', 'e', 'l', 'r', 'g', 'g', 'o', 'u', 'n', 'a', 'u', 'b', 'y', 'l', 'i', 'c', 'w', 'e', 'd', 'g', 'l']
The accuracy was 0.9652777777777778.
```

Figure 8: Optimal Output for Regular Gray-scale Images.

I also experimented with using the other image types, using the original frames with the optimal parameters reduced the accuracy to 92.36% and the binary frames reduced it a substantial amount resulting in 75.69% accuracy. I also tried with the rotated and scaled gray-scale images and got a very good accuracy of 99.3%, though it is important to mention that this may have been because the rotated scaled gray-scale folder had 4 times as many images (each image got 3 duplicates: 20° rotation, -20° rotation, and resized). In addition to trying the other image types, I also used the SIFT feature detection method and also got a substantial decrease in accuracy, giving me a 78.47% accuracy. So the system worked best with the ORB feature detection system with rotated and scaled images. This is not that surprising, as the original images had too much detail and the feature detectors picked up irrelevant features. The binary frames had too little features to find. The gray scale pictures were the hands isolated from the rest of the image and the rotated and scaled ones made all the images more uniform allowing the classifier and clustering functions to better differentiate the gestures.

In terms of future work, the obvious starting point would be to get the images straight from a camera and isolate the images myself and then scale and rotate them. Beyond that

maybe experimenting with dynamic gestures rather than static ones, and possible trying to get it to something close to real time gesture recognition. I would also want to extend this to get to the original goal of this project which was for gesture recognition for device control.

References

[1] Muhammad R. Abid, Feng Shi, and Emil M. Petriu. Dynamic hand gesture recognition from bag-of-features and local part model. In *2012 IEEE International Workshop on Haptic Audio Visual Environments and Games (HAVE 2012) Proceedings*, pages 78–82, 2012.

[2] Guoming Chen, Qiang Chen, Yiqun Chen, and Xiongyong Zhu. Hand gesture recognition via bag of visual words. In *2016 12th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD)*, pages 1159–1163, 2016.

[3] Guoming Chen, Qiang Chen, Yiqun Chen, and Xiongyong Zhu. Hand gesture recognition via bag of visual words. In *2016 12th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD)*, pages 1159–1163, 2016.

[4] Raimundo Farrapo Pinto Junior and Ialis Cavalvante de Paula Junior. Static hand gesture asl dataset, 2019.

[5] Pasquale Foggia, Gennaro Percannella, Alessia Saggese, and Mario Vento. Recognizing human actions by a bag of visual words. In *2013 IEEE International Conference on Systems, Man, and Cybernetics*, pages 2910–2915, 2013.

[6] Lei Zhang, Shengping Zhang, Feng Jiang, Yuankai Qi, Jun Zhang, Yuliang Guo, and Huiyu Zhou. Bomw: Bag of manifold words for one-shot learning gesture recognition from kinect. *IEEE Transactions on Circuits and Systems for Video Technology*, 28(10):2562–2573, 2018.