Sign Language Recognition

Project Phase II (CSD 401)
End Semester Examination (January 27, 2021 , Thursday)

PROJECT TEAM MEMBERS

1. Sanskruti Nakhale BT18CSE015
2. Niraj Agrawal BT18CSE035
3. Tanmay Ganvir BT18CSE036
4. Nisarg Gogate BT18CSE040
5. Ketan Sarode BT18CSE044

NAME OF SUPERVISOR

Dr. Umesh Deshpande

Let’s Recap

Trained, tested and made a GUI for real time testing of ASL(American Sign Language Recognition).

- Created our own dataset.
- Preprocessing - Adaboost detection, Gaussian blur (Works best with plain background)
- Gesture Classification - Used double layer classifier.
  - Layer 1: To classify in 27 (26 Alphabets + blank).
  - Layer 2: To further classify in similar looking hand signs
- CNN:
  - Conv2d(126,126,32)-MaxPooling(63,63,32)-Conv2d(61,61,48)-MaxPooling(30,30,48) and 3 dense layers + 2 Dropout layers
- Results:
  - Accuracy : 0.973
  - Step-Loss : 0.105
  - Val-Loss : 0.003

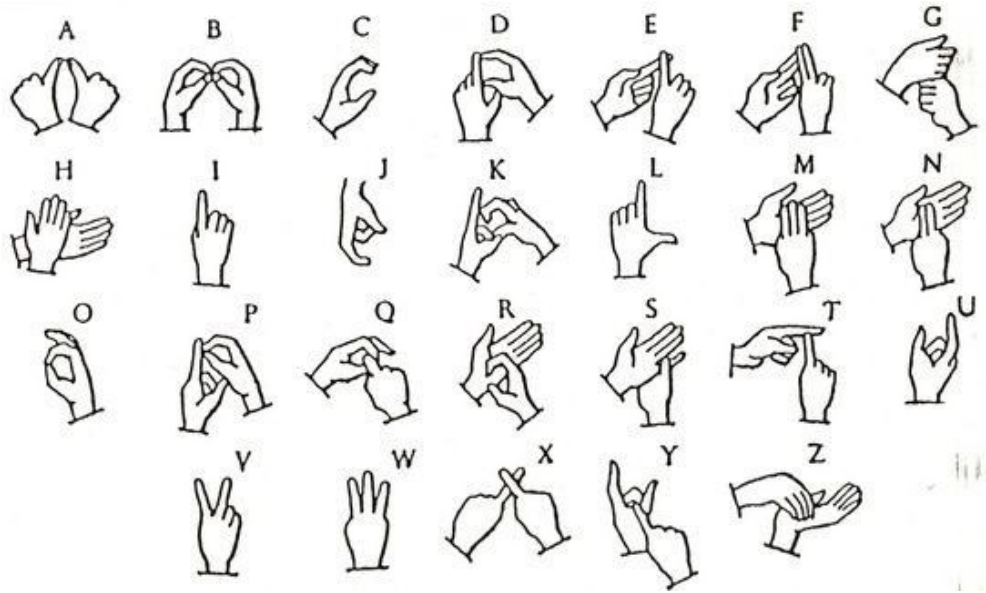
Aim

Our project aims to create a computer application and train a model which when shown a real time video of hand gestures of **Indian Sign Language** shows the output for that particular sign in text format on the screen.

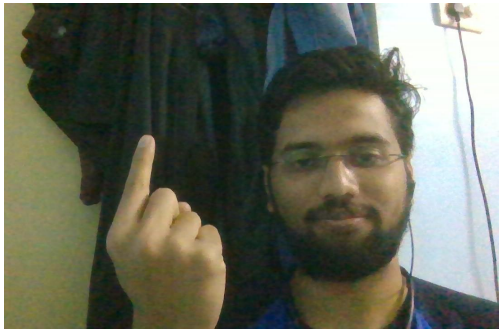
This project aims to predict the 'alphanumeric' gesture of the ISL system.

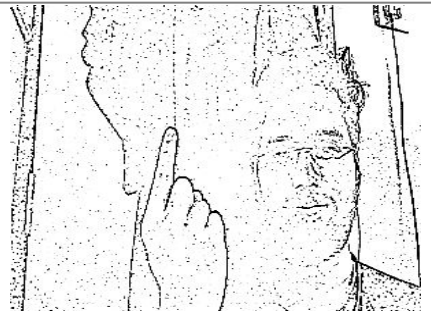
Why do we need a new model for ISL?

- We tested our previous model with new ISL images we got 89.66% accuracy.
- Also real time accuracy was not upto the mark.

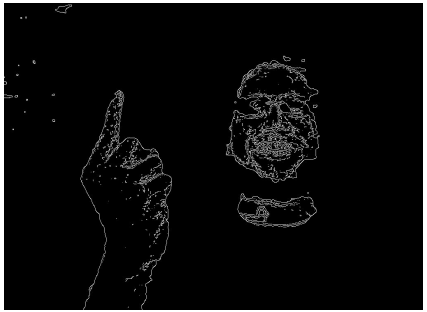


Why new pre-processing?





Old pre-processing



New pre-processing

Pre-processing

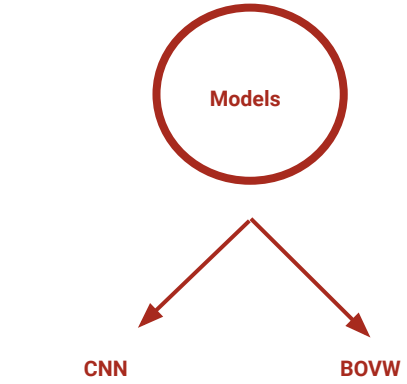
BGR to HSV

Masked

Canny Edge detection

Possible approaches to get better results on ISL?

1. For Hand Detection part preprocessing (hand detection method) YOLO-v3 pre-trained network can be used to find a hand.If any hands are found in the frame, an extended bounding box is created around the hand(s).
2. Skin Segmentation techniques can be used for skin detection (specifically using HSV and YCbCr color-spaces), mostly by its simplicity and performance. However, the skin tones, illumination, and quality is something that could drastically vary between images.
3. Pre-trained Models (via Transfer learning) like Squeeze-Net/ResNet-50/VGG-16 can be used for ISL Sign recognition.
4. BOVW Model

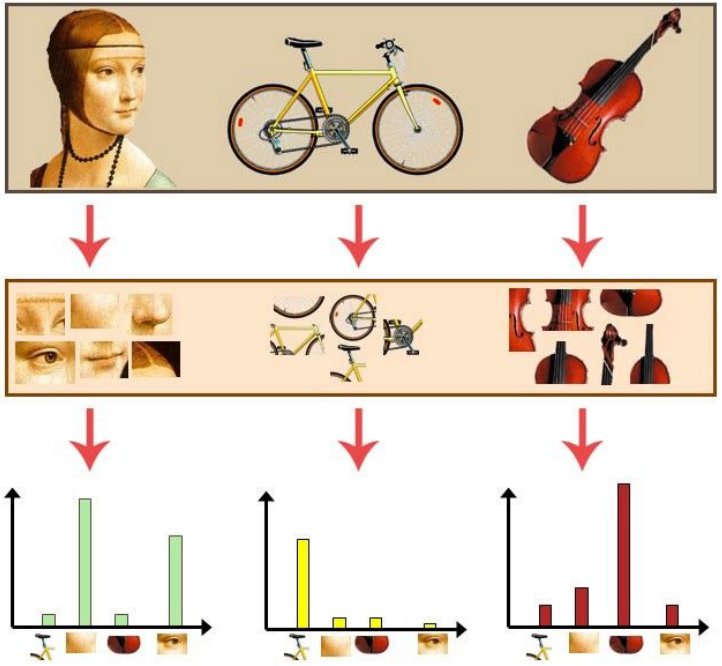


- VGG-16
- ResNet-50
- SqueezeNet

BAG OF VISUAL WORDS MODEL

Bag of Words(**BOW**): This model is mostly used in NLP and IR. We make histogram of most frequent words and try to find the key features that describes the document or corpust the best.

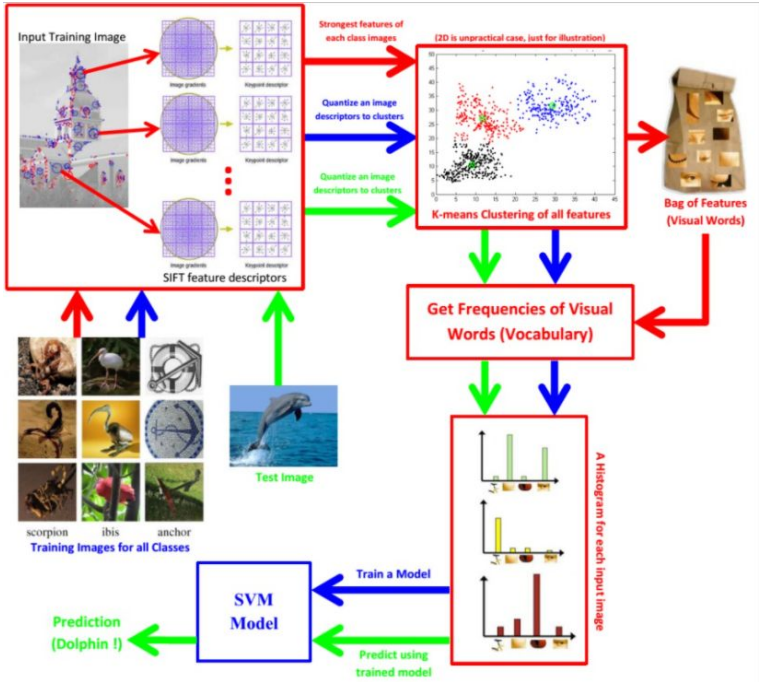
Bag of Visual Words (**BOVW**): Here instead of words we use image features (key points) which are unique patterns in image. Then make frequency histogram based on these patterns. These histograms help us to predict the category of the image.



How do we extract these features? How do we use them?

SIFT/SURF/Orb:

These are few of the algorithms which help us to find out the features. Ex. In SIFT (Scale Invert Feature Transform) each patch(feature) is converted to 128-dimensional vector. After this step, each image is a collection of vectors of the same dimension, where the order of different vectors is of not importance.



Bag of Features Model

- K-means clustering technique categorizes m numbers of descriptors into x number of cluster centre.
- The clustered features form the basis for histogram i-e each image is represented by frequency of occurrence of all clustered features.
- This histogram is used to classify in 35 classifies.

Feature Detection

What is feature detection

K-means (minibatch)

Classifiers

- Naive Bayes
- Logistic Regression
- Support Vector Machine

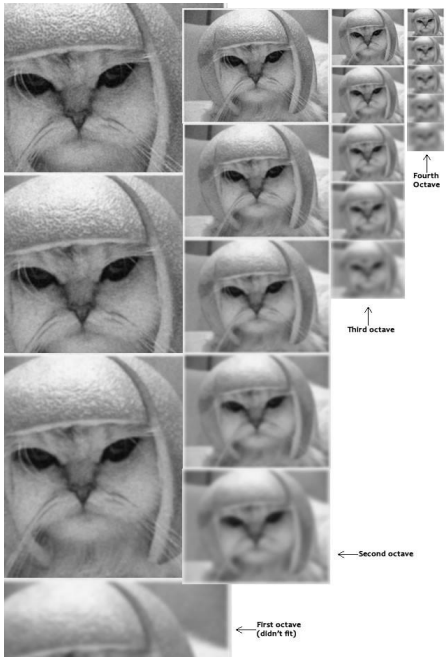
SCALE INVARIANT FEATURE TRANSFORM (SIFT)

SIFT is quite an involved algorithm.

These steps involved in the SIFT algorithm :-

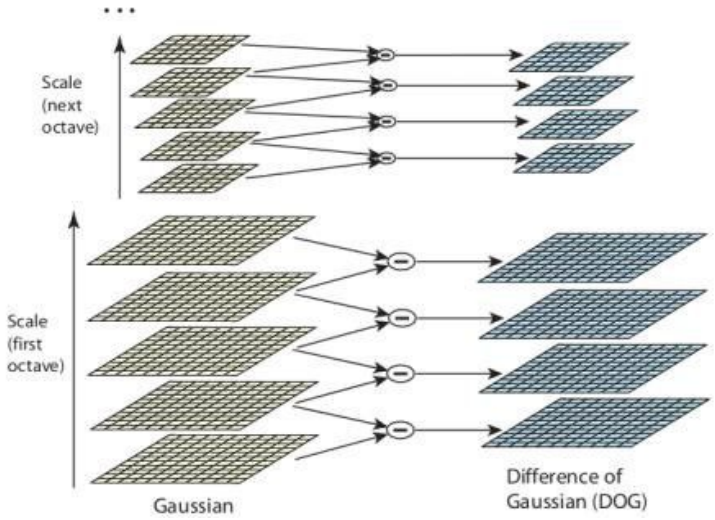
1] Constructing a scale space:-

- The scale space of an image is a function L(x,y,σ) that is produced from the convolution of a Gaussian kernel(Blurring) at different scales with the input image.
- Scale-space is separated into octaves and each octave's image size is half the previous one. Within an octave, images are progressively blurred using the Gaussian Blur operator.



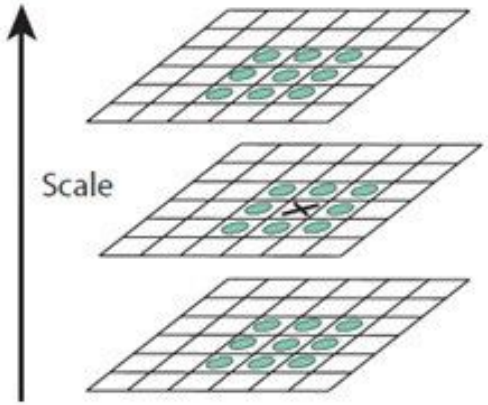
2] DOG(Difference of Gaussian kernel)

- Now we use those blurred images to generate another set of images, the Difference of Gaussians (DoG). These DoG images are great for finding out interesting key points in the image.
- The difference of Gaussian is obtained as the difference of Gaussian blurring of an image with two different σ, let it be σ and *kσ*. This process is done for different octaves of the image in the Gaussian Pyramid.



3] Finding keypoints

- Up till now, we have generated a scale space and used the scale space to calculate the Difference of Gaussians. Those are then used to calculate Laplacian of Gaussian approximations that are scale invariant.
- One pixel in an image is compared with its 8 neighbors as well as 9 pixels in the next scale and 9 pixels in previous scales.If it is a local extrema, it is a potential keypoint. It basically means that keypoint is best represented in that scale.

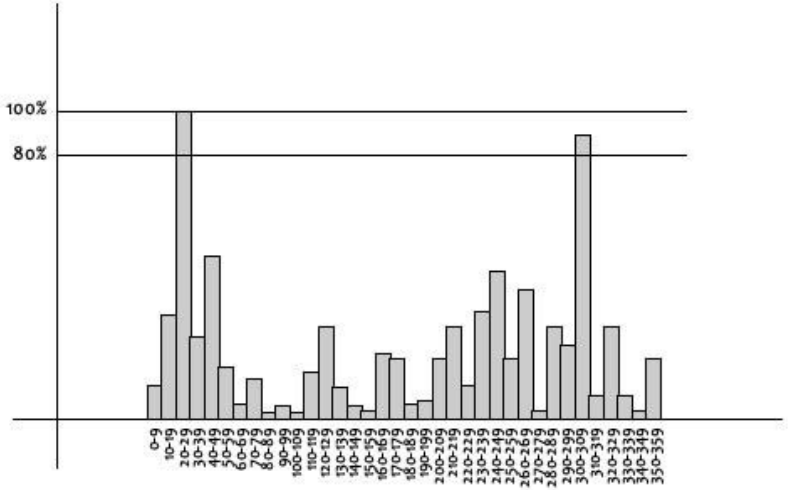


4] Keypoint Localization

- Key points generated in the previous step produce a lot of keypoints. Some of them lie along an edge, or they don’t have enough contrast. In both cases, they are not as useful as features.
- Edges and low contrast features
- Thresholding for low contrast features
- For edges, calculate two gradients at the keypoint. Both perpendicular to each other. Now three possibilities exist : -
    - **A flat region:** If this is the case, both gradients will be small.
    - **An edge:** Here, one gradient will be big (perpendicular to the edge) and the other will be small (along the edge)
    - **A "corner":** Here, both gradients will be big

5] Orientation Assignment

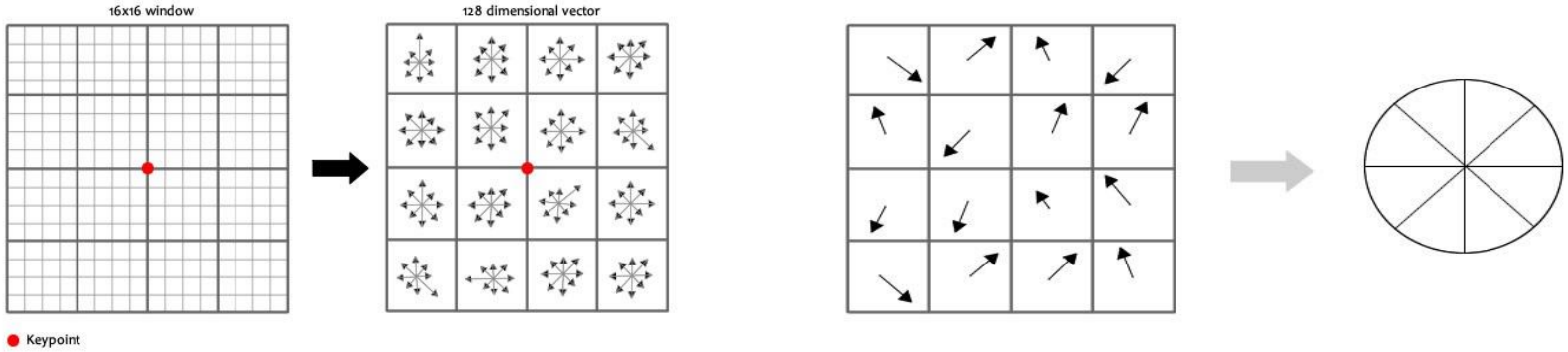
- A neighborhood is taken around the keypoint location depending on the scale, and the gradient magnitude and direction is calculated in that region.
- An orientation histogram with 36 bins covering 360 degrees is created.
- And the “amount” that is added to the bin is proportional to the magnitude of the gradient at that point.
- The highest peak in the histogram is taken and any peak above 80% of it is also considered to calculate the orientation. It creates keypoints with same location and scale, but different directions. It contributes to the stability of matching.



6] Keypoint descriptor

- To do this, a 16x16 window around the keypoint is taken. It is divided into 16 sub-blocks of 4x4 size.
- For each sub-block, 8 bin orientation histogram is created(4x4x8 = 128 bin values). It is represented as a feature vector to form keypoint descriptor. This feature vector introduces a few complications.

1. **Rotation dependence** To achieve rotation independence, the keypoint's rotation is subtracted from each orientation. Thus each gradient orientation is relative to the keypoint's orientation.
2. **Illumination dependence** If we threshold numbers that are big, we can achieve illumination independence. So, any number (of the 128) greater than 0.2 is changed to 0.2. This resultant feature vector is normalized again.



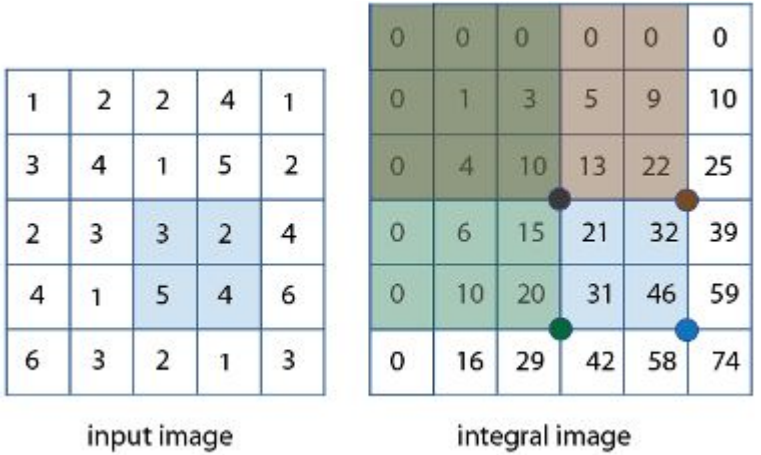
SURF (Speeded-Up Robust Features)

The SURF method (Speeded Up Robust Features) is a fast and robust algorithm for local, similarity invariant representation and comparison of images.

SURF is composed of two steps

- **Feature Extraction**
- **Feature Description**

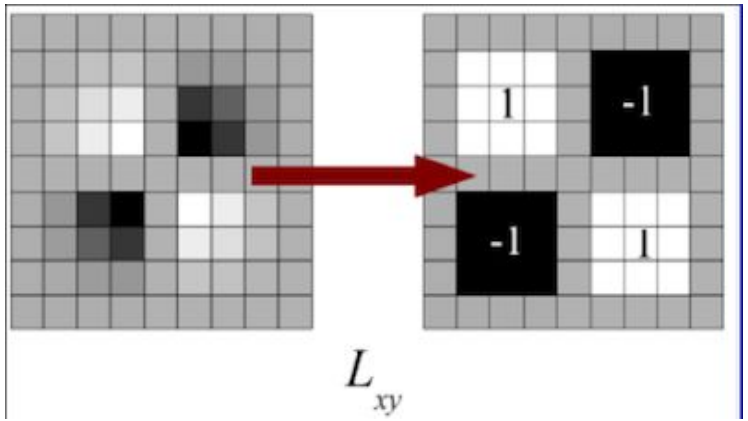
Integral images

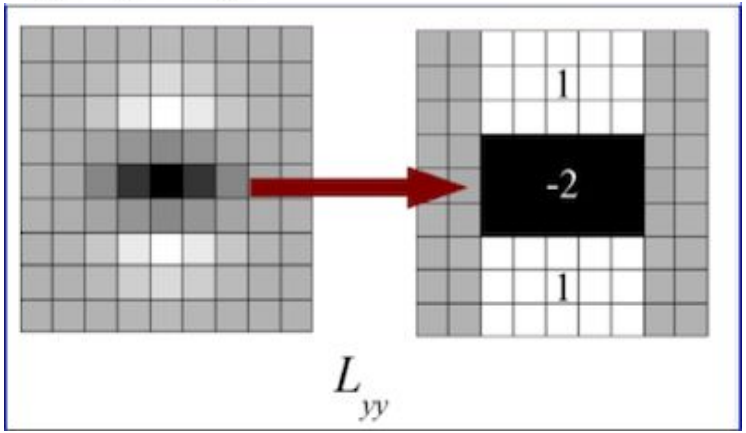


1] Feature Extraction

- The approach for interest point detection uses a very basic Hessian matrix approximation.
- In order to calculate the determinant of the Hessian matrix, first we need to apply convolution with Gaussian kernel, then second-order derivative. These approximate second-order Gaussian derivatives and can be evaluated at a very low computational cost using integral images and independently of size.
- The 9 × 9 box filters in the above images are approximations for Gaussian second order derivatives with σ = 1.2. We denote these approximations by Dxx, Dyy, and Dxy. Now we can represent the determinant of the Hessian (approximated) as:

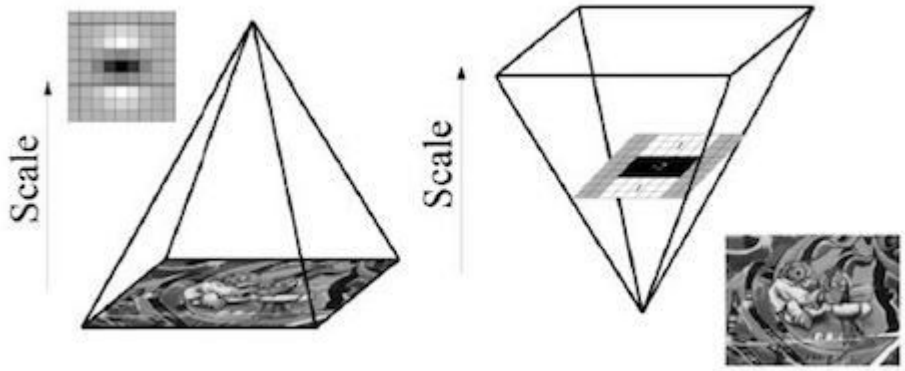
$$\det(\mathcal{H}_{\text{approx}}) = D_{xx}D_{yy} - (wD_{xy})^2.$$





2] Scale-space representation

- Due to the use of box filters and integral images, surf does not have to iteratively apply the same filter to the output of a previously filtered layer but instead can apply such filters of any size at exactly the same speed directly on the original image, and even in parallel. Therefore, the scale space is analyzed by up-scaling the filter size(9×9 ⟶ 15×15 ⟶ 21×21 ⟶ 27×27, etc) rather than iteratively reducing the image size.
- In order to localize interest points in the image and over scales, a non-maximum suppression in a 3 × 3 × 3 neighborhood is applied.



3] Feature Description

The creation of SURF descriptor takes place in two steps. The first step consists of fixing a reproducible orientation based on information from a circular region around the keypoint. Then, we construct a square region aligned to the selected orientation and extract the SURF descriptor from it.

Orientation Assignment

1. Surf first calculate the Haar-wavelet responses in x and y-direction, and this in a circular neighborhood of radius 6s around the keypoint, with s the scale at which the keypoint was detected. Also, the sampling step is scale dependent and chosen to be s, and the wavelet responses are computed at that current scale s. Accordingly, at high scales the size of the wavelets is big. Therefore integral images are used again for fast filtering.
2. Then we calculate the sum of vertical and horizontal wavelet responses in a scanning area, then change the scanning orientation (add $\pi/3$), and re-calculate, until we find the orientation with largest sum value, this orientation is the main orientation of feature descriptor.

Descriptor Components

1. The first step consists of constructing a square region centered around the keypoint and oriented along the orientation we already got above. The size of this window is 20s.

2. Then the region is split up regularly into smaller 4 × 4 square sub-regions. For each sub-region, we compute a few simple features at 5×5 regularly spaced sample points. For reasons of simplicity, we call **dx** the Haar wavelet response in the horizontal direction and **dy** the Haar wavelet response in the vertical direction (filter size 2s). To increase the robustness towards geometric deformations and localization errors, the responses dx and **dy** are first weighted with a Gaussian (σ = 3.3s) centered at the keypoint.

Then, the wavelet responses **dx** and **dy** are summed up over each subregion and form a first set of entries to the feature vector. In order to bring in information about the polarity of the intensity changes, we also extract the sum of the absolute values of the responses, **|dx|** and **|dy|**. Hence, each sub-region has a four-dimensional descriptor vector v for its underlying intensity structure **V = ($\sum$ dx, $\sum$ dy, $\sum$|dx|, $\sum$|dy|)**. This results in a descriptor vector for all 4×4 sub-regions of **length 64**(In **Sift**, our descriptor is the **128-D vector**, so this is part of the reason that SURF is faster than Sift).

CLUSTERING ALGORITHM

- K-means clustering is one of the unsupervised machine learning algorithm to cluster the required features.
- A cluster refers to a collection of data points aggregated together because of certain similarities.
- So here first, Target number k is decided, which refers to the number of centroids you need in the dataset. A centroid is the imaginary or real location representing the center of the cluster.
- Every data point is allocated to each of the clusters through reducing the in-cluster sum of squares.

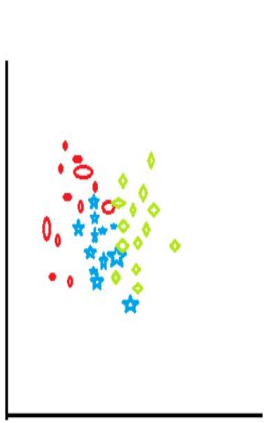


fig 1: before applying k-means clustering

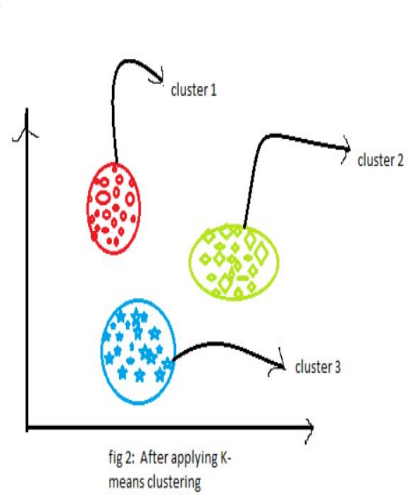


fig 2: After applying K-means clustering

How the K-means algorithm works

It starts with a first group of randomly selected centroids, which are used as the beginning points for every cluster, and then performs iterative (repetitive) calculations to optimize the positions of the centroids.It halts creating and optimizing clusters when either:
1. The centroids have stabilized — there is no change in their values because the clustering has been successful.
2. The defined number of iterations has been achieved.

Suppose there are X objects, that are to be divided into K clusters. The input can be a set of features, X={x1,x2,...,xn}.

$$\arg\min_{\mathbf{S}} \sum_{i=1}^{k} \sum_{\mathbf{x} \in S_i} \|\mathbf{x} - \boldsymbol{\mu}_i\|^2$$

The goal is basically to minimize the distance between each point in the scatter cloud and the assigned centroids.where μ is mean of points for each Si(cluster) and S denotes set of points partitioned into clusters of {S1,S2,...Si}.

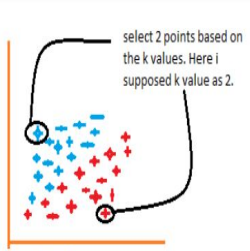
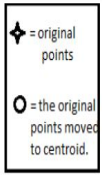
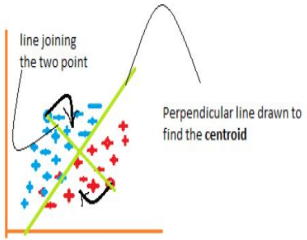


Figure 1

F2: Find the average of all the blue points and red points and move the selected points to **centroid.**





F3: Some of the red points changed to blue points, that means they belong to the group blue now. Again the repeat the same process.

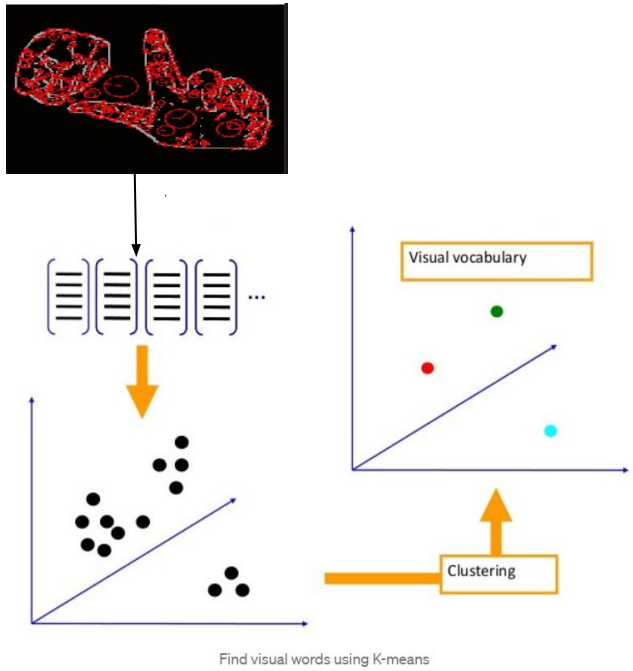

F4: The same process has been applied here. This process will be continued until we get the **two complete different cluster.**

Role of K-Means in our BOVW Model

Here we quantize the feature space (Descriptor list returned by Feature descriptor). Making this operation via clustering algorithms such as K-means. The center points, that we get from the clustering algorithm, are our visual words.

How it works :

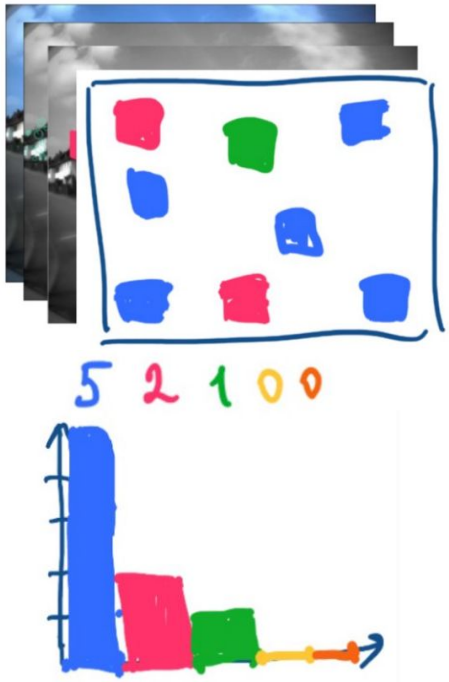
- As discussed earlier, these techniques detect the keypoints in our image and compute their values(descriptors) for the input image.
- These feature detectors return an array containing the descriptors.We do this for every image in our training dataset.
- Now, we will be having N(no of images in training dataset) arrays.
- we then stack these arrays vertically. Now we will use clustering algorithm like K-Means to form K clusters.
- K-Means groups the data points into K groups and will return the center of each group(see image below).Each cluster center(centroid) acts as a visual word.All these K centroids form our required codebook.



Find visual words using K-means

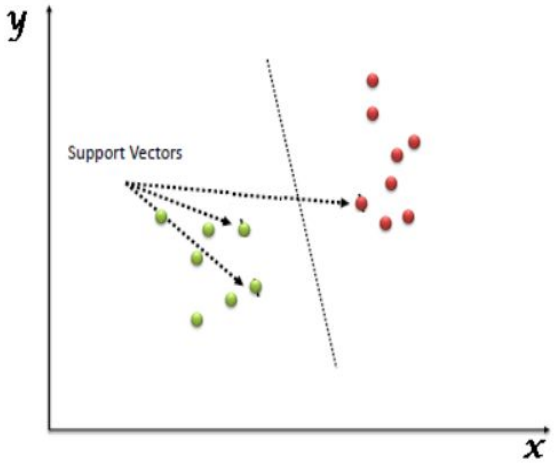
Creating Histogram

- Now we will create a 2D-array of zeros of shape (N,K) , Then we iterate through our images again and look for words in the image present in our dictionary.
- Once we detect a word present in both the dictionary and image, we increase the count of that particular word(i.e array[i][w]+=1 where i is the current image and w is the word).
- This is how we create the histograms for the images. Similarly,all the images will be converted to histograms.



Classifiers

- Having generated a BOVW histogram for each of our training images, the final step in setting up our image classifier is to train a SVM (Support Vector Machine) or any other classifier on the BOVW histograms.
- Once trained, the intention is that the SVM is then able to take the BOVW histogram for a new image which was not part of the training set and correctly classify it.
- Support Vector Machine" (SVM) is a supervised machine learning algorithm that can be used for both classification or regression challenges. However, it is mostly used in classification problems.
- So In the SVM algorithm, we plot each data item as a point in n-dimensional space (where n is a number of features you have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyper-plane that differentiates the respective classes very well.



Future Work

- Different clustering methods.
- Different Classifiers to get better accuracy.
- If all goes well according plan we will add YOLO and make a real time app else we will stick to ROI method with some other CNN model(via transfer learning) like Squeeze-net.


VISVESVARAYA NATIONAL INSTITUTE OF TECHNOLOGY
NAGPUR
ॐ: कर्मसु कौशलम्

Thank You !!!

References

[1]Manuel Eleazar Martínez-Gutiérrez, José Rafael Rojano-Cáceres, Edgard Benítez-Guerrero, Héctor Eduardo Sánchez-Barrera,"Data Acquisition Software for Sign Language Recognition" in Universidad Veracruzana, Facultad de Estadística e informática, Mexico

[2]Mathavan Suresh Anand, Nagarajan Mohan Kumar, Angappan Kumaresan, "An Efficient Framework for Indian SignLanguageRecognition Using Wavelet Transform" Circuits and Systems, Volume 7, pp 1874-1883, 2016

[3]Suharjitoa, Ricky Andersonb , Fanny Wiryanab , Meita Chandra Ariestab , Gede Putra Kusumaa ,"Sign Language Recognition Application Systems for Deaf-Mute People: A Review Based on Input-Process-Output" in 2nd International Conference on Computer Science and Computational Intelligence 2017, ICCSCI 2017, 13-14 October 2017, Bali, Indonesia

[4]T. Yang, Y. Xu, and "A. , Hidden Markov Model for Gesture Recognition", CMU-RI-TR-94 10, Robotics Institute, Carnegie Mellon Univ.,Pittsburgh,PA, May 1994

References

[5]Pigou L., Dieleman S., Kindermans PJ., Schrauwen B. (2015) “Sign Language Recognition Using Convolutional Neural Networks”. In: Agapito L., Bronstein M., Rother C. (eds) Computer Vision - ECCV 2014 Workshops. ECCV 2014. Lecture Notes in Computer Science, vol 8925. Springer, Cham

[6]Byeongkeun Kang , Subarna Tripathi , Truong Q. Nguyen ”Real-time sign language fingerspelling recognition using convolutional neural networks from depth map” 2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR)

[7]Zaki, M.M., Shaheen, S.I.: “Sign language recognition using a combination of new vision based features”. Pattern Recognition Letters 32(4), 572–577 (2011)