

TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

PULCHOWK CAMPUS

**(PROJECT REPORT TITLE)**

By:

**(name(s) of students)**

A PROJECT WAS SUBMITTED TO THE DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING IN PARTIAL FULLFILLMENT OF THE REQUIREMENT FOR THE BACHELOR’S DEGREE IN ELECTRONICS & COMMUNICATION / COMPUTER ENGINEERING

DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING

LALITPUR, NEPAL

MONTH, YEAR

**Format of approval page**

TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

PULCHOWK CAMPUS

DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING

The undersigned certify that they have read, and recommended to the Institute of Engineering for acceptance, a project report entitled "Title of the Project" submitted by Name of Student(s) in partial fulfilment of the requirements for the Bachelor’s degree in Electronics & Communication / Computer Engineering.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Supervisor, name of Supervisor

Title

Name of the Department

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Internal Examiner, name of Internal Examiner

Title

Name of the Organization, he belongs to

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

External Examiner, name of External

Title

Name of the Organization, he belongs to

------------------------------------------------------------------------------

Coordinator, Name of Coordinator

Title

Name of the coordinating committee

**DATE OF APPROVAL:** Day.Month.Year

**COPYRIGHT**

The author has agreed that the Library, Department of Electronics and Computer Engineering, Pulchowk Campus, Institute of Engineering may make this report freely available for inspection. Moreover, the author has agreed that permission for extensive copying of this project report for scholarly purpose may be granted by the supervisors who supervised the project work recorded herein or, in their absence, by the Head of the Department wherein the project report was done. It is understood that the recognition will be given to the author of this report and to the Department of Electronics and Computer Engineering, Pulchowk Campus, Institute of Engineering in any use of the material of this project report. Copying or publication or the other use of this report for financial gain without approval of to the Department of Electronics and Computer Engineering, Pulchowk Campus, Institute of Engineering and author’s written permission is prohibited.

Request for permission to copy or to make any other use of the material in this report in whole or in part should be addressed to:

Head

Department of Electronics and Computer Engineering

Pulchowk Campus, Institute of Engineering

Lalitpur, Kathmandu

Nepal

**ACKNOWLEDGEMENT**

Acknowledgement goes here…

**ABSTRACT**

Object detection is one of the fundamental challenges in computer vision. Many researchers and tech giants are spending a lot of resources for the advancement of this field. This report presents the detailed explanation of algorithms and methodology we adopted to complete our minor project entitled ‘Object Detection’. Our system detects few objects and localizes them, with an accuracy of about 60%.

Our method ﬁrst obtains salient features from an input image using a robust local feature extractor called SIFT. After extracting all keypoints and descriptors from the set of training images, they are clustered into N centroids. This operation is performed using the standard K-means unsupervised learning algorithm. The key assumption in this paper is that the extracted descriptors are independent and hence can be treated as a BoW in the image. For a query image, descriptors are extracted using the same robust local feature extractor. Each descriptor is mapped to its visual word equivalent by ﬁnding the nearest cluster centroid in the dictionary. An ensuing count of words for each image is passed into a learning algorithm to classify the image.

Our methodology works only so much, and have limitations. Because the background of image generally create noises, our object detection program is limited to images with plain white background.

**TABLE OF CONTENTS**

PAGE OF APPROVAL.................................................................................................II

COPYRIGHT...............................................................................................................III ACKNOWLEDGEMENTS.........................................................................................IV

ABSTRACT..................................................................................................................V

TABLE OF CONTENTS.............................................................................................VI

LIST OF FIGURES....................................................................................................VII

LIST OF TABLES......................................................................................................VII

LIST OF SYMBOLS AND ABBREVIATIONS......................................................VIII

1 INTRODUCTION.......................................................................................................1

1.1 Sub-heading..................................................................................................1

1.2 Sub-heading..................................................................................................2

1.3 Sub-heading..................................................................................................3

2 LITERATURE REVIEW............................................................................................4

2.1 Sub-heading..................................................................................................5

2.1.1 Sub Sub-heading...........................................................................7

2.1.2 Sub Sub-heading...........................................................................8

2.1.3 Sub Sub-heading.........................................................................10

2.2 Sub-heading................................................................................................13

2.2.1 Sub Sub-heading.........................................................................14

3 OTHERS HEADINGS..............................................................................................16

3.1 Sub-heading................................................................................................16

3.1.1 Sub Sub-heading.........................................................................18

4 RESULTS......…………………………....................................................................30

5 CONCLUSIONS AND FURTHER WORK.............................................................32

5.1 Conclusions................................................................................................32

5.2 Further Work..............................................................................................33

REFERENCES ............................................................................................................34

APPENDIX A: Title of Appendix...............................................................................37

APPENDIX B: Title of Appendix................................................................................38

**LIST OF FIGURES**

Figure 1.1 System Overview..........................................................................................1

Figure 2.1 Intersection of two lines................................................................................6

Figure 2.2 Intersection of two lines in 3D......................................................................6

Figure 2.3 Intersection of N lines...................................................................................7

Figure 2.4 Single Camera...............................................................................................7

Figure 2.5 Stereo Camera Geometry..............................................................................8

Figure 3.1 Image and it’s SIFT features.........................................................................9

Figure 3.2 Determining stable keypoints.....................................................................10

Figure 3.3 Corner extractions for camera calibration...................................................11

Figure 3.4 Cameras mounted on the board..................................................................12

Figure 3.5 Matching right figure with left one.............................................................13

Figure 3.6 Test System.................................................................................................14

**LIST OF TABLES**

Table 3.1 Calibration result of first camera..................................................................11

Table 3.2 Calibration Data...........................................................................................20

**LIST OF SYMBOLS / ABBREVIATIONS**

θ Orientation Vector

p Position Vector

SIFT128 The 128-dimension SIFT descriptors

2D 2 Dimensional

3D 3 Dimensional

BoW Bag of Words

SIFT Scale Invariant Feature Transform

SMO Sequential Minimal Optimization

SURF Speeded Up Robust Features

SVM Support Vector Machine

1. **INTRODUCTION**

**Computer vision** is an interdisciplinary field that deals with how computers can be made to gain high-level understanding from digital images or videos. From the perspective of engineering, it seeks to automate tasks that the human visual system can do. Sub-domains of computer vision include scene reconstruction, event detection, video tracking, object recognition, object pose estimation, learning, indexing, motion estimation, and image restoration. In this report, we intend to explore the methodologies we adopted to realize object recognition.

**Object detection** is a computer technology related to computer vision and image processing that deals with detecting instances of semantic objects of a certain class (such as humans, buildings, or cars) in digital images and videos.. Object detection algorithms rely on matching, learning, or pattern recognition algorithms using appearance-based or feature-based techniques. Computer vision tasks include methods for acquiring, processing, analyzing and understanding digital images, and in general, deal with the extraction of high-dimensional data from the real world in order to produce numerical or symbolic information, *e.g.*, in the forms of decisions. Understanding in this context means the transformation of visual images (the input of the retina) into descriptions of the world that can interface with other thought processes and elicit appropriate action.

**1.1 Applications of Object detection**

There are numerous applications for object recognition and classiﬁcation in images. The leading uses of object classiﬁcation are in the ﬁelds of robotics, photography, and security. Robots commonly take advantage of object classiﬁcation and localization in order to recognize certain objects within a scene. Using object detection , in the future, we might be able to use object detection to identify anomalies in a scene such as bombs or explosives (by making use of a quadcopter).

Object detection can also be used for people counting, which is used for analyzing store performance or crowd statistics during festivals, for manufacturing industry, where identification of products can be done using object detection, for classifying images found online, for automated vehicle parking systems ,for face detection, etc.

* 1. **Object detection methods in Computer Vision**

Object detection algorithms typically use extracted features and learning algorithms to recognize instances of an object category.

Objects can be recognized using a variety of models, including:

* Extracted features and boosted learning algorithms
* Bag-of-words models with features such as SURF and MSER
* Gradient-based and derivative-based matching approaches
* Viola-Jones algorithm
* Template matching
* Image segmentation and blob analysis

**1.3 Our Methodology in Brief**

Our method ﬁrst obtains salient features from an input image using a robust local feature extractor. The leading techniques for such a purpose include the SIFT and SURF. After extracting all keypoints and descriptors from the set of training images, our method clusters these descriptors into N centroids. This operation is performed using the standard K-means unsupervised learning algorithm. The key assumption in this paper is that the extracted descriptors are independent and hence can be treated as a BoW in the image. This BoW nomenclature is derived from text classiﬁcation algorithms in classical machine learning. For a query image, descriptors are extracted using the same robust local feature extractor. Each descriptor is mapped to its visual word equivalent by ﬁnding the nearest cluster centroid in the dictionary. An ensuing count of words for each image is passed into a learning algorithm to classify the image.

**2. LITERATURE REVIEW**

Computer vision (image understanding) is a discipline that studies how to reconstruct, interpret and understand a 3D scene from its 2D images in terms of the properties of the structures present in the scene. It has been studied since late 1970s and has lot more to go. Signiﬁcant eﬀorts have been paid to develop representation schemes and algorithms aiming at recognizing generic objects in images taken under diﬀerent imaging conditions (e.g., viewpoint, illumination, and occlusion). Within a limited scope of distinct objects, such as handwritten digits, ﬁngerprints, faces, and road signs, substantial success has been achieved. Object recognition is also related to content-based image retrieval and multimedia indexing as a number of generic objects can be recognized. In addition, signiﬁcant progress towards object categorization from images has been made in the recent years. Object recognition has also been studied extensively in psychology, computational neuroscience and cognitive science.

**2.1 Geometry-based approaches**

Early attempts on object recognition were focused on using geometric models of objects to account for their appearance variation due to viewpoint and illumination change. The main idea is that the geometric description of a 3D object allows the projected shape to be accurately predicted in a 2D image under projective projection, thereby facilitating recognition process using edge or boundary information (which is invariant to certain illumination change). Much attention was made to extract geometric primitives (e.g., lines, circles, etc.) that are invariant to viewpoint change. Nevertheless, it has been shown that such primitives can only be reliably extracted under limited conditions (controlled variation in lighting and viewpoint with certain occlusion).

**2.2 Appearance-based approaches**

In contrast to early eﬀorts on geometry-based object recognition works, most recent eﬀorts have been centered on appearance-based techniques as advanced feature descriptors and pattern recognition algorithms are developed. Most notably, the eigen face methods have attracted much attention as it is one of the ﬁrst face recognition systems that are computationally eﬃcient and relatively accurate. As the goal of object recognition is to tell one object from the others, discriminative classiﬁers have been used to exploit the class speciﬁc information. Classiﬁers such as k-nearest neighbor, neural networks with radial basis function (RBF), dynamic link architecture, Fisher linear discriminant, support vector machines (SVM), sparse network of Winnows (SNoW), and boosting algorithms have been applied to recognize 3D objects from 2D images. While appearance-based methods have shown promising results in object recognition under viewpoint and illumination change, they are less eﬀective in handling occlusion. A single exemplar is unlikely to succeed. As such approaches require pattern to be identified from the image and object looks different under varying conditions (illumination, viewing direction, change in shape/size), all possible appearances of an object is impossible to be represented from a single exemplar. Thus, use of multiple example images of the objects is used to perform recognition. In addition, a large set of exemplars needs to be segmented from images for generative or discriminative methods to learn the appearance characteristics. These problems are partially addressed with parts-based representation schemes.

**2.3 Feature-based approaches**

The central idea of feature-based object recognition algorithms lies in ﬁnding interest points, often occurred at intensity discontinuity, that are invariant to change due to scale, illumination and aﬃne transformation. It correctly classifies images of objects under perturbation by noise, rotation and scaling. FBR uses a set of feature detectors to build a representation vector for images. The feature detectors are learned from the dataset itself. The scale-invariant feature transform (SIFT) descriptor, proposed by Lowe, is arguably one of the most widely used feature representation schemes for vision applications. The SIFT approach uses extrema in scale space for automatic scale selection with a pyramid of diﬀerence of Gaussian ﬁlters, and keypoints with low contrast or poorly localized on an edge are removed. Next, a consistent orientation is assigned to each keypoint and its magnitude is computed based on the local image gradient histogram, thereby achieving invariance to image rotation. At each keypoint descriptor, the contribution of local image gradients are sampled and weighted by a Gaussian, and then represented by orientation histograms. For example, the 16×16 sample image region and 4×4 array of histograms with 8 orientation bins are often used, thereby providing a 128-dimensional feature vector for each keypoint. Objects can be indexed and recognized using the histograms of keypoints in images. Numerous applications have been developed using the SIFT descriptors, including object retrieval. Although the SIFT approach is able to extract features that are insensitive to certain scale and illumination change, vision applications with large base line change entail the need of aﬃne invariant point and region operators. Finally, SIFT-based methods are expected to perform better for objects with rich texture information as suﬃcient number of keypoints can be extracted. On the other hand, they also require sophisticated indexing and matching algorithms for eﬀective object recognition.

**2.4 Recent Development**

With more reliable representation schemes and recognition algorithms being developed, tremendous progress has been made in the last decade towards recognizing objects under variation in viewpoint, illumination and under partial occlusion.

Together with SIFT feature extraction, Support Vector Machines or SVM has been very influential factor for the sudden rise of Computer Vision. Support vector machines are a generic purpose, out-of-the-box classifiers. We can see major promising projects going on like ImageNet, Microsoft COCO and many more.

The **ImageNet** project is a large visual [database](https://en.wikipedia.org/wiki/Database) designed for use in [visual object recognition software](https://en.wikipedia.org/wiki/Outline_of_object_recognition) research. As of 2016, over ten million URL's of images have been hand-annotated by ImageNet to indicate what objects are pictured; in at least one million of the images, bounding boxes are also provided.

Computer Vision has arrived to the today, where it can finally start fulfilling its prehistoric promises. All of a sudden, the lessons tought from the previous decade, the abundance of data and the tremendous power of modern hardware brought neural networks, the forgotten child of AI, to the surface again. Today, based on the modern version of neural networks, namely deep learning networks , we are able to classify the image content very, very accurately. In fact, it could be that computers are now reaching a human level of accuracy in certain tasks.

**3. METHODOLOGY**

There are appearance-based and feature-based algorithms for object detection. Feature-based algorithm, in general, selects a small number of critical visual features from a large set of training images, using machine learning algorithms and yields extremely efficient classifiers. Objects are detected by matching image features with the learned classifier.

**3.1 General methodology for object detection.**

 Fig.1. Block Diagram For Training



Fig.2. Block Diagram for Testing

**3.2 Methodology adopted in Detail**

**3.2.1 Extracting features using SIFT**

For our object detection project, we used a feature-based algorithm called **Scale-invariant feature transform** (or **SIFT**). The algorithm was published by [David Lowe](https://en.wikipedia.org/wiki/David_Lowe_(computer_scientist)) in 1999.

For any object in an image, interesting points on the object can be extracted to provide a "feature description" of the object. This description, extracted from a training image, can then be used to identify the object when attempting to locate the object in a test image containing many other objects. To perform reliable recognition, it is important that the features extracted from the training image be detectable even under changes in image scale, noise and illumination.

SIFT can robustly identify objects even among clutter and under partial occlusion, because the SIFT feature descriptor is invariant to [uniform scaling](https://en.wikipedia.org/wiki/Scaling_(geometry)), [orientation](https://en.wikipedia.org/wiki/Orientation_(geometry)), and partially invariant to [affine distortion](https://en.wikipedia.org/wiki/Affine_transformation) and illumination changes.

Steps involved for applying SIFT algorithm in feature extraction are as follows:

**3.2.1.1 Creating Scale Space**

Suppose we need to detect a tree. We need to get rid of minor detail from images like the leaves, twigs, etc. While getting rid of these details, we must ensure that we do not introduce new false details. The only way to do that is with the Gaussian Blur (it was proved mathematically, under several reasonable assumptions).

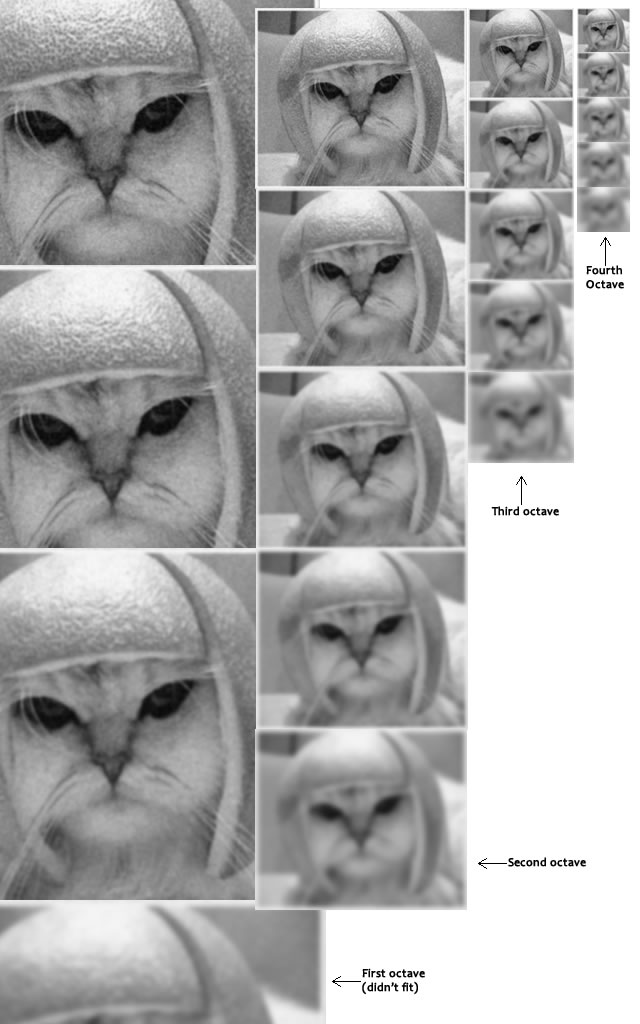


Fig. Scale and Octave

**Blurring**

Mathematically, "blurring" is referred to as the convolution of the gaussian operator and the image. Gaussian blur has a particular expression or "operator" that is applied to each pixel. What results is the blurred image.

The symbols:

* L is a blurred image
* G is the Gaussian Blur operator
* I is an image
* x,y are the location coordinates
* σ is the "scale" parameter. Think of it as the amount of blur. Greater the value, greater the blur.
* The \* is the convolution operation in x and y. It "applies" gaussian blur G onto the image I.

This is the actual Gaussian Blur operator

**3.2.1.2 LoG Approximations**

Now we use those blurred images to generate another set of images, given be Laplacian of Gaussian(LoG) , given approximately by the Difference of Gaussians (DoG). These DoG images are a great for finding out interesting key points in the image.

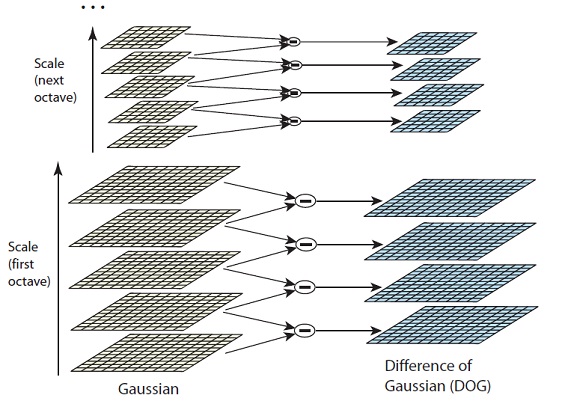


Fig. DoG images

**3.2.1.3 Finding Key Points**

Finding key points is a two part process

1. Locate maxima/minima in DoG images
2. Find subpixel maxima/minima

The first step is to coarsely locate the maxima and minima. This is simple. We iterate through each pixel and check all it's neighbours. The check is done within the current image, and also the one above and below it. Something like this:

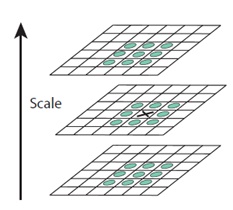


Fig. Calculating maxima/minima

X marks the current pixel. The green circles mark the neighbours. This way, a total of 26 checks are made. **X is marked as a "key point" if it is the greatest or least of all 26 neighbours.**

**Find subpixel maxima/minima**

Using the available pixel data, subpixel values are generated. This is done by the Taylor expansion of the image around the approximate key point.

Mathematically, it's like this.

We can easily find the extreme points of this equation (differentiate and equate to zero). On solving, we'll get subpixel key point locations. These subpixel values increase chances of matching and stability of the algorithm.

**3.2.1.4 Removing low contrast features**

This is simple. If the magnitude of the intensity (i.e., without sign) at the current pixel in the DoG image (that is being checked for minima/maxima) is less than a certain value, it is rejected.

Because we have subpixel keypoints (we used the Taylor expansion to refine keypoints), we again need to use the taylor expansion to get the intensity value at subpixel locations. If it's magnitude is less than a certain value, we reject the keypoint.

**3.2.1.5 Finding KeyPoint Orientations**

We already know the scale at which the keypoint was detected (it's the same as the scale of the blurred image). So we have scale invariance. The next thing is to assign an orientation to each keypoint. This orientation provides rotation invariance.

Gradient magnitudes and orientations are calculated using these formulae:

The magnitude and orientation is calculated for all pixels around the keypoint.

Then, a  [histogram](http://aishack.in/tutorials/histograms-from-simplest-to-the-most-complex/) is created for this.In this histogram, the 360 degrees of orientation are broken into 36 bins (each 10 degrees). Lets say the gradient direction at a certain point (in the "orientation collection region") is 18.759 degrees, then it will go into the 10-19 degree bin. And the "amount" that is added to the bin is proportional to the magnitude of gradient at that point.

Once we've done this for all pixels around the keypoint, the histogram will have a peak at some point.

Below, we see the histogram peaks at 20-29 degrees. So, the keypoint is assigned orientation 3 (the third bin)

Also, any peaks above 80% of the highest peak are converted into a new keypoint. This new keypoint has the same location and scale as the original. But it's orientation is equal to the other peak.

So, orientation can split up one keypoint into multiple keypoints.

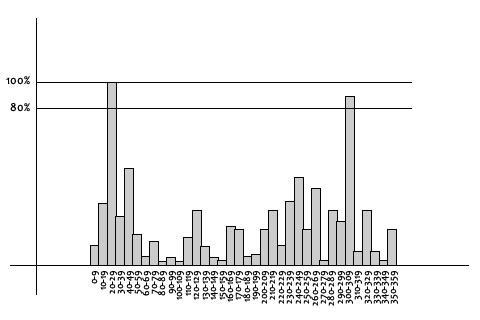


Fig. Histogram for finding KeyPoint Orientations

**3.2.1.6 Generating Feature**

To do this, a 16x16 window around the keypoint is taken . This 16x16 window is broken into sixteen 4x4 windows. Within each 4x4 window, gradient magnitudes and orientations are calculated. These orientations are put into an 8 bin histogram.

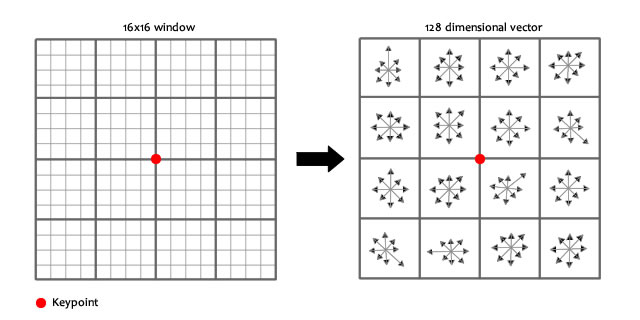


Fig. Creating a feature description vector of a keypoint

Doing this for all 16 pixels, we would've "compiled" 16 totally random orientations into 8 predetermined bins. We do this for all sixteen 4x4 regions. So we end up with 4x4x8 = 128 numbers. Once we have all 128 numbers, we normalize them (just like we would normalize a vector in school, divide by root of sum of squares). These 128 numbers form the "feature vector". This keypoint is uniquely identified by this feature vector.

Because of time factor and complexity of this method, we used a library to extract SIFT features.

**3.2.2 Generating single feature vector for each image**

A key development in image classiﬁcation using keypoints and descriptors is to represent these descriptors using a BoW model. Although spatial and geometric relationship information between descriptors is lost using this assumption, the inherent simpliﬁcation gains make it highly advantageous. The descriptors extracted from the training images are grouped into N clusters of visual words **using K-means clustering Algorithm**. A descriptor is categorized into its cluster centroid using a Euclidean distance metric. For our purposes, we choose a value of N = 500. This parameter provides our model with a balance between high bias (underﬁtting) and high variance (overﬁtting). For a query image, each extracted descriptor is mapped into its nearest cluster centroid. A histogram of counts is constructed by incrementing a cluster centroid’s number of occupants each time a descriptor is placed into it. The result is that each image is represented by a histrogram vector of length N. It is necessary to normalize each histogram by its L2-norm to make this procedure invariant to the number of descriptors used.

**K-means Clustering Algorithm**

K-means is  one of  the simplest unsupervised  learning  algorithms  that  solve  the well  known clustering problem. The procedure follows a simple and  easy  way  to classify a given data set  through a certain number of  clusters (assume k clusters) fixed apriori. The  main  idea  is to define k centers, one for each cluster. These centers  should  be placed in a cunning  way  because of  different  location  causes different  result. So, the better  choice  is  to place them  as  much as possible  far away from each other. The  next  step is to take each point belonging  to a  given data set and associate it to the nearest center. When no point  is  pending,  the first step is completed and an early group age  is done. At this point we need to re-calculate k new centroids as barycenter of  the clusters resulting from the previous step. After we have these k new centroids, a new binding has to be done  between  the same data set points  and  the nearest new center. A loop has been generated. As a result of  this loop we  may  notice that the k centers change their location step by step until no more changes  are done or  in  other words centers do not move any more. Finally, this  algorithm  aims at  minimizing  an objective function know as squared error function given by:

where,  
                           *‘||xi- vj||’* is the Euclidean distance between *xi* and *vj.*

*‘ci’* is the number of data points in *ith* cluster.

*‘c’* is the number of cluster centers.

**Algorithmic steps for k-means clustering**

Let  X = {x1,x2,x3,……..,xn} be the set of data points and V = {v1,v2,…….,vc} be the set of centers.

1) Randomly select *‘c’* cluster centers.

2) Calculate the distance between each data point and cluster centers.

3) Assign the data point to the cluster center whose distance from the cluster center is minimum of all the cluster centers..

4) Recalculate the new cluster center using:

where,*‘ci’* represents the number of data points in *ith* cluster.

5) Recalculate the distance between each data point and new obtained cluster centers.

6) If no data point was reassigned then stop, otherwise repeat from step 3).

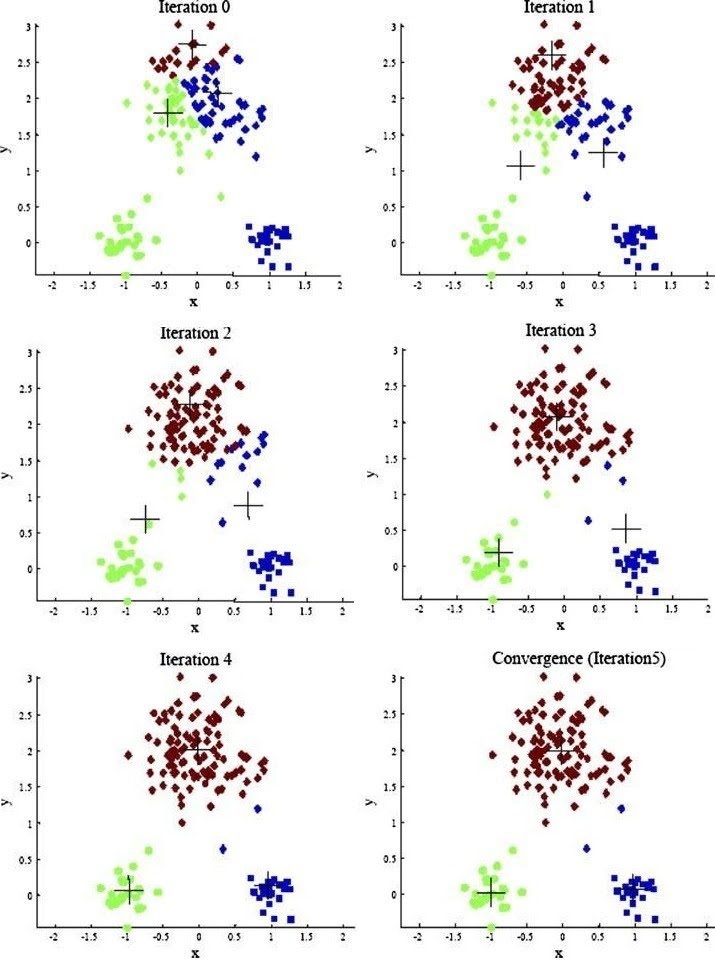


Fig. Iterations in K-means clustering Algorithm

Here , + denote cluster centers.

**3.2.3 Training Algorithm – SMO**

We have used a simpliﬁed version of the Sequential Minimal Optimization (SMO) algorithm for training images and creating classifiers. SMO receives its input: Design matrix, a n\*m matrix, where n is the number of training examples, and m is the number of features for each example. Using BoW model, we have 500-dimensional vector (i.e m=500) to represent each image. Other input is a n \*1 matrix, where n is the number of examples , and the value on each row is either 0 or 1. 0 is for negative training example and 1 is for positive training example. Other inputs are various constant parameters and Kernel function used for optimizing the classifier.

A support vector machine computes a linear classifier of the form

………………………………….(1)

Since we want to apply this to a binary classification problem, we will ultimately predict y = 1 if

f(x) ≥ 0 and y = −1 if f(x) < 0, but for now we simply consider the function f(x). By looking

at the dual problem we see that this can also be expressed using inner products as

….........(2)

is Kernel Function. For our purpose , we use Gaussian Kernel, so as to yield efficient non-linear classifier. Gaussian kernel function is given by,

The SMO algorithm gives an efficient way of solving the dual problem of the (regularized)

support vector machine optimization problem. We avoided the mathematical derivation of the optimization problem of SMO, and simply followed the following algorithm:

**Algorithm: Simplified SMO**

Input:

C: regularization parameter

tol: numerical tolerance

max\_passes: max # of times to iterate over α’s without changing

(x(i)),y(i)),. . . , (x(m), y(m)): training data

Output:

α ∈ Rm: Lagrange multipliers for solution

b ∈ R : threshold for solution

◦ Initialize αi = 0, ∀i, b = 0.

◦ Initialize passes = 0.

◦ while (passes < max passes)

◦ num\_changed alphas = 0.

◦ for i = 1, . . .m,

◦ Calculate Ei = f(x(i)) − y(i)  using (2).

◦ if ((y(i)Ei < −tol && αi < C) || y(i)Ei > tol && αi >0))

◦ Select j ≠ i randomly.

◦ Calculate Ei = f(x(j)) − y(j) using (2).

◦ Save old α’s: (old)

αi (old) = αi

αj (old) = αj

◦ Compute L and H by

• If y(i )≠ y(j), L = max(0, αj –αi), H = min(C,C+ αj –αi)

• If y(i)= y(j), L = max(0, αi + αj − C), H = min(C, αi + αj)

◦ if (L == H)

continue to next i.

◦ Compute η as

◦ if (η >= 0)

continue to next i.

◦ Compute and clip new value for αj using (a) and (b)

……………..(a)

……(b)

◦ if (|αj − αj(old) | < 10-5)

continue to next i.

◦ Determine value for αi using

◦ Compute b1 and b2 using

◦ Compute b by

◦ num\_changed alphas := num\_changed alphas + 1.

◦ end if

◦ end for

◦ if (num\_changed alphas == 0)

passes := passes + 1

◦ else

passes := 0

◦ end while

**3.2.4 Testing**

**Localization**

There can be multiple object in a test image. For simplicity of localization, we take input as test images, only with plain white background. Identifying the boundary of non-white pixels, we localize each image. Each image is then temporarily cropped.

**Generating histogram using BoW model**

Using BoW model, we represent each cropped image by 500-dimensional histogram vector.

**Prediction**

Using trained models obtained from SMO training algorithm, we predict the class of the cropped image (eg, pendrive, keys, etc).

**3.2.5 Reinforcement Learning**

We have implemented reinforcement learning, so that the system can keep improving. After prediction of test image is given, we allow the user to make corrections, if object identified is incorrect. Then the histogram vector for that image is appended in a list of histograms of the class it belongs, so that correct prediction can be made next time.



Fig. Block diagram for Testing – Detailed



Fig. Block diagram for Training - Detailed

**REFERNCES**

1. M. Betke, L.Gurvits, “Mobile Robot Localization Using Visual Landmarks,” *IEEE Transaction**on Robotics and Automation,* Vol.13, No.2, pp. 251-263, 1997.
2. . Sim, G. Dudek, “Mobile robot localization from learned landmarks,” *In Proceedings of the International Conference on Intelligent Robots and Systems*, Vol.2, pp. 1060-1065, 1998.
3. S. Thrun, “Finding landmarks for mobile robot navigation,” *In Proceedings of the*

*International Conference on Robotics and Automation,* Vol. 2, pp. 958-963, 1998.

1. Desouza, G. N., Kak, A. C., “Vision for mobile robot navigation: a survey” *IEEE Transaction on Pattern Analysis and Machine Intelligence*, Vol.24, No.2, pp.237-267, 2002.
2. D.G. Lowe, “Object recognition from local scale invariant features”, *In Proceedings of the Seventh International Conference on Computer Vision*, pp.1150-1157, September 1999.
3. R.Hartley, A.Zisserman, “Multiple View Geometry in Computer Vision”, *Cambridge University Pre*ss, 2000.
4. D. Lowe, “Distinctive Image Feature from Scales-Invariant Keypoints,” *International Journal of Computer Vision*, pp. 91-110, 2004.
5. R. Cipolla, P. J. Giblin, “Visual Motion of Curves and Surface”, *Cambridge University Press*, 2001.
6. D.G. Lowe “Demo Software: SIFT Keypoint Detector”, <http://people.cs.ubc.ca/~lowe/keypoints/>, 2005.
7. J. Bouguet “Camera Calibration Toolbox for Matlab”, http://www.vision.caltech.edu/bouguetj /calib\_doc/, 2008.
8. Drew Schmitt, Nicholas McCoy, “Object Classiﬁcation and Localization Using SURF Descriptors,” 2011.
9. Efstratios Gavves,”A Brief History Of Computer Vision,” 2014.
10. http://aishack.in/tutorials/sift-scale-invariant-feature-transform-introduction/
11. The Simplified SMO Algorithm, CS 229, Autumn 2009.

**APPENDIX B: FORMAT OF DISKETTES/CD CONTAINING COMPUTER SOFTWARE**

Listings of software should be given in a diskette/CD. The diskette/CD should contain files containing the source code, one or more sample input and corresponding output separately. Other than there, there must be another file named “README”. In this ASCII text file, the following sections must appear:

* Files in the Disk: In this section, the names of the files together with their contents must be listed.
* Hardware Requirements: In this section, the equipment, graphics card, mouse, disk capacity, RAM capacity etc necessary to run the software must be noted.
* Software Requirements: In this section the operating system, the compiler, the linker and the libraries etc necessary to compile and link the software must be listed.

Please note that no copyrighted material file (compiler, library etc.) should be put on the disk without obtaining the necessary license from the copyright owner.