ACKNOWLEDGEMENT

Our report will remain incomplete if we do not mention the efforts of those people who helped us in completing this project.

We would like to thank the Computer Science Department, BITS Pilani for providing us with this excellent opportunity to learn and pursue a Study Oriented project course.

We are thankful to Dr.Sanjay Singh, our guide for his considerable and irreplaceable involvement in the progress of our project and for giving us a chance to learn. We are also thankful to his constant support during desperate times and for his composed approach.

We would also like to thank Mr. Sumeet Saurav, for assisting us in the project and being there whenever assistance and guidance was required.

We are also highly indebted to our mentor Professor K. Haribabu, for giving us this valuable opportunity and mentoring the project.

Finally, we would like to thank each and every one who supported us by any means in the completion of this project.

BAPIRAJU VAMSI TADIKONDA (2013A7PS039P)

KOTHA ACHYUTH REDDY (2013A7PS041P)

SAI KRISHNA JONNALGADDA (2013A7PS141P)

INTRODUCTION:

Computer vision also known as Machine vision, is a branch of Computer science that deals with interpreting, reconstructing and understanding an situation from an image in terms of the properties of the structures present in the scene.

Our Human Vision is very powerful in recognizing and relating things in the real world. But it has its own limitations like we cannot remember each and every small thing and we can analyse things present in the visual spectrum only. To overcome these problems we take the help of computers and storage units.

Object Detection comes in the Middle Layer of the Computer Vision Hierarchy. As the name suggests Object detection deals with identifying objects like Humans, buildings or text in digital images or videos. Special Domain of Object Detection includes Face Detection and Pedestrian Detection.

CHALLENGES

Though the idea is very intriguing, there are many challenges that we need to face.

1. Image Quality: When the image has good image resolution there will be more accuracy in detecting the object.

2. Noise: Dealing with noise is a major concern while detecting an object. Noise hinders the accuracy of detecting the object in an image. Noise can be due to wind, flash in cameras for images in dark areas, motion of camera etc.

3. Lighting: When the image is taken in a good lighting the image has high intensity levels of an object. But when taken in a dark environment the image looks darker. So the Algorithm cannot just look at image intensity values.

4. Within-class variation: Different instances of the same kind of object can look quite different to one another. For example a white Audi and a black Audi are both cars. It should also try to match an Audi vehicle and a Volkswagen vehicle as cars.

5. Aspect: An object can look very different when viewed from different directions. The system should be able to detect the changes and identify both the images as the same object.

6. Deformation: Many objects can change their appearance significantly without their identity changing. For example, when a person is walking his shape changes constantly due to the hands and legs movement.

APPLICATIONS OF OBJECT DETECTION

Object Recognition has many different applications in many spheres:

1. Optical character recognition – This is the process of conversion of typed, handwritten text in the form of images into text that can be used by the computer. Here the objects to be recognised are alphabets and numbers.

2. Content Based Image Retrieval – This is process of searching for an image in a database of objects (already present in google search).
3. Automatic Parking System – as the names suggests here maintaining parking space is done automatic.
4. Identification Systems – Applies in various institutions, offices and organizations where the photo of any new applicant is matched with many databases like criminal database, foreign citizen database, etc. It is a great method for demographic calculations.
5. Surveillance – Used all around the world by Police departments, detectives, security systems, Intelligence agencies, etc. to closely monitor suspects, identify probable criminals at crime sights, etc. They are also used as biometric identification.
6. Pervasive Computing – Computing devices equipped with various sensors and software are being installed in various day-to-day things like phones, cars and home making them smarter and more interactive with the users. These devices will soon be able to get a picture of the virtual world and people around in order to help humans in the real world.

APPROACHES OF OBJECT DETECTION

It can be done in various methods. Some of them are

- Appearance Based Methods
    1. Edge Matching
    2. Divide and Conquer Search
    3. Grey Scale Matching
    4. Gradient Matching

5. Histogram of Receptive Field Responses
6. Large Model Bases

- Feature Based Methods
    1. Interpretation Trees
    2. Hypothesize and Test
    3. Pose Consistency
    4. Pose Clustering
- Bag of Words Representation

We used Histogram of Oriented Grades for the purpose of Object Detection.The technique counts occurrences of gradient orientation in localized portions of an image. This method is similar to that of edge orientation histograms, scale-invariant feature transform descriptors, and shape contexts, but differs in that it is computed on a dense grid of uniformly spaced cells and uses overlapping local contrast normalization for improved accuracy.The HOG descriptor has a few key advantages over other descriptors. Since it operates on local cells, it is invariant to geometric and photometric transformations, except for object orientation. Such changes would only appear in larger spatial regions. Moreover, as Dalal and Triggs discovered, coarse spatial sampling, fine orientation sampling, and strong local photometric normalization permits the individual body movement of pedestrians to be ignored so long as they maintain a roughly upright position. The HOG descriptor is thus particularly suited for human detection in images.

HOG Algorithm Implementation

Gradient computation

The first step of calculation in many feature detectors in image pre-processing is to ensure normalized color and gamma values. As Dalal and Triggs point out, however, this step can be omitted in HOG descriptor computation, as the ensuing descriptor normalization essentially achieves the same result. Image pre-processing thus provides little impact on performance. Instead, the first step of calculation is the computation of the gradient values. The most common method is to apply the 1-D centered, point discrete derivative mask in one or both of the horizontal and vertical directions. Specifically, this method requires filtering the color or intensity data of the image with the following filter kernels:

$$[-1, 0, 1] \text{ and } [-1, 0, 1]^T.$$

Dalal and Triggs tested other, more complex masks, such as the 3x3 Sobel mask or diagonal masks, but these masks generally performed poorer in detecting humans in images. They also experimented with Gaussian smoothing before applying the derivative mask, but similarly found that omission of any smoothing performed better in practice.

Orientation binning

The second step of calculation is creating the cell histograms. Each pixel within the cell casts a weighted vote for an orientation-based histogram channel based on the values found in the gradient computation. The cells themselves can either be rectangular or radial in shape, and the

histogram channels are evenly spread over 0 to 180 degrees or 0 to 360 degrees, depending on whether the gradient is “unsigned” or “signed”. Dalal and Triggs found that unsigned gradients used in conjunction with 9 histogram channels performed best in their human detection experiments. As for the vote weight, pixel contribution can either be the gradient magnitude itself, or some function of the magnitude. In tests, the gradient magnitude itself generally produces the best results. Other options for the vote weight could include the square root or square of the gradient magnitude, or some clipped version of the magnitude.

Descriptor blocks

To account for changes in illumination and contrast, the gradient strengths must be locally normalized, which requires grouping the cells together into larger, spatially connected blocks. The HOG descriptor is then the concatenated vector of the components of the normalized cell histograms from all of the block regions. These blocks typically overlap, meaning that each cell contributes more than once to the final descriptor. Two main block geometries exist: rectangular R-HOG blocks and circular C-HOG blocks. R-HOG blocks are generally square grids, represented by three parameters: the number of cells per block, the number of pixels per cell, and the number of channels per cell histogram. In the Dalal and Triggs human detection experiment, the optimal parameters were found to be four 8x8 pixels cells per block (16x16 pixels per block) with 9 histogram channels. Moreover, they found that some minor improvement in performance could be gained by applying a Gaussian spatial window within each block before tabulating histogram votes in order to weight pixels around the edge of the blocks less. The R-HOG blocks appear quite similar to the scale-invariant feature transform (SIFT) descriptors; however, despite

their similar formation, R-HOG blocks are computed in dense grids at some single scale without orientation alignment, whereas SIFT descriptors are usually computed at sparse, scale-invariant key image points and are rotated to align orientation. In addition, the R-HOG blocks are used in conjunction to encode spatial form information, while SIFT descriptors are used singly.

Circular HOG blocks (C-HOG) can be found in two variants: those with a single, central cell and those with an angularly divided central cell. In addition, these C-HOG blocks can be described with four parameters: the number of angular and radial bins, the radius of the center bin, and the expansion factor for the radius of additional radial bins. Dalal and Triggs found that the two main variants provided equal performance, and that two radial bins with four angular bins, a center radius of 4 pixels, and an expansion factor of 2 provided the best performance in their experimentation. Also, Gaussian weighting provided no benefit when used in conjunction with the C-HOG blocks. C-HOG blocks appear similar to shape context descriptors, but differ strongly in that C-HOG blocks contain cells with several orientation channels, while shape contexts only make use of a single edge presence count in their formulation.

Block normalization

Dalal and Triggs explored four different methods for block normalization. Let v be the non-normalized vector containing all histograms in a given block, $\|v\|_k$ be its *k*-norm for

$k = 1, 2$ and e be some small constant (the exact value, hopefully, is unimportant). Then the normalization factor can be one of the following:

L2-norm: $$f = \frac{v}{\sqrt{\|v\|_2^2 + e^2}}$$

L2-hys: L2-norm followed by clipping (limiting the maximum values of v to 0.2) and renormalizing, as in

L1-norm: $$f = \frac{v}{(\|v\|_1 + e)}$$

L1-sqrt: $$f = \sqrt{\frac{v}{(\|v\|_1 + e)}}$$

In addition, the scheme L2-hys can be computed by first taking the L2-norm, clipping the result, and then renormalizing. In their experiments, Dalal and Triggs found the L2-hys, L2-norm, and L1-sqrt schemes provide similar performance, while the L1-norm provides slightly less reliable performance; however, all four methods showed very significant improvement over the non-normalized data.

DETAILS OF DATABASE

This database contains 213 images in total. There are 10 subjects and 7 facial expressions for each subject. Each subject has about twenty images and each expression includes two to three images. The seven expressions are angry, happy, disgust, sadness, surprise, fear and neutral respectively.

Those 7 expressions are

1. Neutral
2. Angry
3. Sad
4. Happy
5. Surprise
6. Disgust
7. Fear

The HOG features are extracted from these images and the SVM is trained with them.

Different features including SIFT , Gabor filters , Local Binary Patterns (LBP) and HOG (Histogram of Oriented Gradient) have been proposed for facial expression recognition. Facial

expressions result from muscle movements and these movements could be regarded as a kind of deformation. For example, the muscle movements of the mouth cause the mouth open or close, and cause brows raiser or lower. These movements are similar to deformations. Considering that HOG features are pretty sensitive to object deformations. In this paper, we propose to use the HOG features to encode facial components. HOG was first proposed by Dalal and Triggs in 2005 . It is well received by computer vision community and widely used in many object detection applications, especially in pedestrian detection. HOG numerates the appearance of gradient orientation in a local patch of an image. The idea is that the distribution of the local gradient intensity and orientation could describe the local object appearance and shape.

Compared with other features such as LBP and Gabor filters, HOG is also very useful in facial expression recognition. HOG can characterize the shapes of important components constitute facial expressions. So we apply the HOG to encode these facial components. In our experiments, we set cell size to 8×8, the number of bin size to 9, the orientation range to 0 -180

Support Vector Machine

Support Vector Machine (SVM) has been widely used in various pattern recognition tasks. It is believed that SVM can achieve a near optimum separation among classes. In our study, we train SVMs to perform facial expression classification using the features we proposed. In general, SVM builds a hyperplane to separate the highdimensional space. An ideal separation is achieved when the distance between the hyper plane and the training data of any class is the largest. Given a training set of labeled samples, a SVM tries to find a hyperplane to distinguish the samples

with the smallest errors. For a input vector , the classification is achieved by computing the distance from the input vector to the hyperplane. The original SVM is a binary classifier. However, we can take the one-against-rest strategy to perform the multi-class classification

Training set and testing set

Machine learning is about learning some properties of a data set and applying them to new data. This is why a common practice in machine learning to evaluate an algorithm is to split the data at hand into two sets, one that we call the training set on which we learn data properties and one that we call the testing set on which we test these properties.

Choosing the parameters of the model

It is possible to automatically find good values for the parameters by using tools such as *grid search* and *cross validation*.

Cross-validation, sometimes called **rotation estimation**, is a model validation technique for assessing how the results of a statistical analysis will generalize to an independent data set. It is mainly used in settings where the goal is prediction, and one wants to estimate how accurately a predictive model will perform in practice. In a prediction problem, a model is usually given a dataset of *known data* on which training is run (*training dataset*), and a dataset of *unknown data* (or *first seen* data) against which the model is tested (*testing dataset*). The goal of cross

validation is to define a dataset to "test" the model in the training phase (i.e., the *validation dataset*), in order to limit problems like overfitting, give an insight on how the model will generalize to an independent dataset (i.e., an unknown dataset, for instance from a real problem), etc.

k-fold cross-validation

In *k*-fold cross-validation, the original sample is randomly partitioned into *k* equal sized subsamples. Of the *k* subsamples, a single subsample is retained as the validation data for testing the model, and the remaining $k - 1$ subsamples are used as training data. The cross-validation process is then repeated *k* times (the *folds*), with each of the*k* subsamples used exactly once as the validation data. The *k* results from the folds can then be averaged (or otherwise combined) to produce a single estimation. The advantage of this method over repeated random sub-sampling (see below) is that all observations are used for both training and validation, and each observation is used for validation exactly once. 10-fold cross-validation is commonly used, but in general *k* remains an unfixed parameter.

When $k=n$ (the number of observations), the *k*-fold cross-validation is exactly the leave-one-out cross-validation.

In *stratified k*-fold cross-validation, the folds are selected so that the mean response value is approximately equal in all the folds. In the case of a dichotomous classification, this means that each fold contains roughly the same proportions of the two types of class labels.

Results

Accuracy on training set: 1.0

Accuracy on testing set: 0.642857142857

Classification Report:

	PRECISION	RECALL	F1-SCORE	SUPPORT
1	1.00	0.90	0.95	10
2	0.70	0.70	0.70	10
3	0.33	0.50	0.40	10
4	1.00	0.60	0.75	10
5	0.67	0.40	0.50	10
6	0.46	0.60	0.52	10
7	0.73	0.80	0.76	10
avg/total	0.70	0.64	0.65	70

Confusion Matrix:

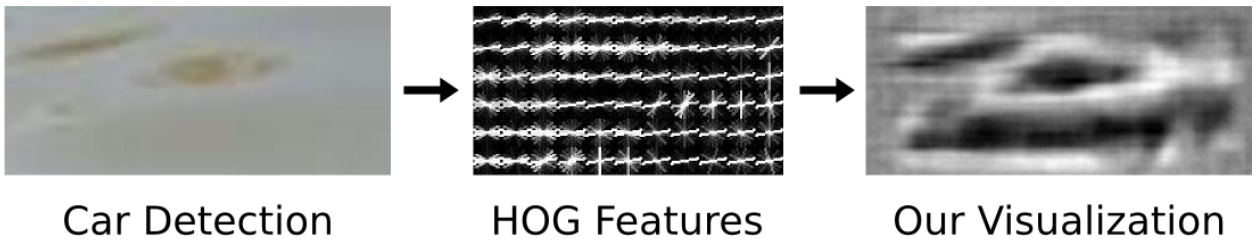
	ANGRY	DISGUST	FEAR	HAPPY	NEUTRAL	SAD	SURPRISE
ANGRY	9	1	0	0	0	0	0
DISGUST	0	7	0	0	0	3	0
FEAR	0	0	5	0	1	2	2
HAPPY	0	0	1	6	1	2	0
NEUTRAL	0	0	5	0	4	0	1
SAD	0	2	2	0	0	6	0
SURPRISE	0	0	2	0	0	0	8

CONCLUSION

We got a low testing accuracy. Many of the results are correct.To improve results we must take a new methods into consideratons such as extracting histogram oriented gradient features from individual parts of the face like eyebrows, eye ,teeth, nose etc. and making a combined feature vector of them. HOG features can be used for object detection.

Hog detector failures

Our visualizations offer an explanation. Below we show the output from our visualization on the HOG features for the false car detection. This visualization reveals that, while there are clearly no cars in the original image, there is a car hiding in the HOG descriptor.



HOG features see a slightly different visual world than what humans see, and by visualizing this space, we can gain a more intuitive understanding of our object detectors.

REFERENCES

http://www.pyimagesearch.com/2014/11/10/histogram-oriented-gradients-object-detection/

Histograms of Oriented Gradients for Human Detection-Navneet Dalal and Bill Triggs

http://scikit-learn.org/stable/documentation.html

http://opencv.org/

APPENDIX

The Code

```
from __future__ import division

import cv2

import os

import numpy as np

from sklearn import svm

def facecrop(image):

    facedata = "haarcascade_frontalface_default.xml"

    cascade = cv2.CascadeClassifier(facedata)



    img = cv2.imread(image)
```

```
    minisize = (img.shape[1],img.shape[0])
    miniframe = cv2.resize(img, minisize)

    faces = cascade.detectMultiScale(miniframe)

    for f in faces:
        x, y, w, h = [ v for v in f ]
        cv2.rectangle(img, (x,y), (x+w,y+h), (255,255,255))

        sub_face = img[y:y+h, x:x+w]
        #face_file_name = "crop.jpg"
        #cv2.imwrite(face_file_name, sub_face)
        #cv2.imshow(image, sub_face)
        #cv2.imshow(image, img)

    return sub_face

def extracthog(image):
```

```
    croppedface = facecrop(image)
    winStride = (8,8)
    padding = (8,8)
    locations = ((10,20),)
    hog = cv2.HOGDescriptor()
    hist = hog.compute(croppedface,winStride,padding,locations)
    hist = hist.transpose()
    hist = hist[0]
    return hist




imageformat=".tiff"
basepath = "imgdata/traindata/"
traindata = []
labels = []
i=1
while(i<8):
```

```
    path=basepath+str(i)+"/"
    imfilelist=[os.path.join(path,f) for f in os.listdir(path) if f.endswith(imageformat)]
    count = 0
    for el in imfilelist:
        traindata.append(extracthog(el))
        labels.append(i)
        count += 1
        print (path + str(count))
    i += 1


X_train = np.asarray(traindata)
y_train =  np.asarray(labels)




testpath = "imgdata/testdata/"
testdata = []
testlabels = []
i=1
while(i<8):
```

```
    path=testpath+str(i)+"/"
    imfilelist=[os.path.join(path,f) for f in os.listdir(path) if f.endswith(imageformat)]
    count = 0
    for el in imfilelist:
        testdata.append(extracthog(el))
        testlabels.append(i)
        count += 1
        print (path + str(count))
    i += 1


X_test = np.asarray(testdata)
y_test =  np.asarray(testlabels)


import numpy as np
from sklearn import cross_validation
from sklearn import datasets
from sklearn import svm
from sklearn import metrics


clf1 = svm.SVC(kernel='linear', C=1)
```

```
def train_and_evaluate(clf, X_train, X_test, y_train, y_test):

    clf.fit(X_train, y_train)

    print "Accuracy on training set:"
    print clf.score(X_train, y_train)
    print "Accuracy on testing set:"
    print clf.score(X_test, y_test)

    y_pred = clf.predict(X_test)

    print "Classification Report:"
    print metrics.classification_report(y_test, y_pred)
    print "Confusion Matrix:"
    print metrics.confusion_matrix(y_test, y_pred)

train_and_evaluate(clf1, X_train, X_test, y_train, y_test)
```