Experiment No : 9

Name :Pratham Ingawale

Roll no: 25

Aim: To Creating and Training an Object Detector

Objective: Bag of Words BOW in computer version Detecting cars in a scene

Theory :

Creating and Training an object detector

The aim of this project is to develop an object detector, a computer vision system capable of recognizing and localizing specific objects within images or video frames. This involves training a machine learning model to identify objects of interest accurately.

Bag-of -words

In this theoretical discussion, the Bag of Words (BOW) technique in computer vision is explored. It covers the principles of feature extraction and representation using BOW, which involves breaking down an image into visual words or features to analyze and compare.

BOW in Computer Vision

The objective of this project is to apply the Bag of Words (BOW) technique to the field of computer vision. BOW is a methodology used to extract and represent visual features within images, and the objective is to employ it for various computer vision tasks.

Detecting Cars

This theory section specifically focuses on the process of detecting cars within a scene. It includes discussions on how BOW can be applied to recognize car features, the importance of feature matching, and the overall workflow for car detection in computer vision applications.

Example

Code:

```
import cv2

import numpy as np

import os



# Check if the 'CarData' directory exists
```

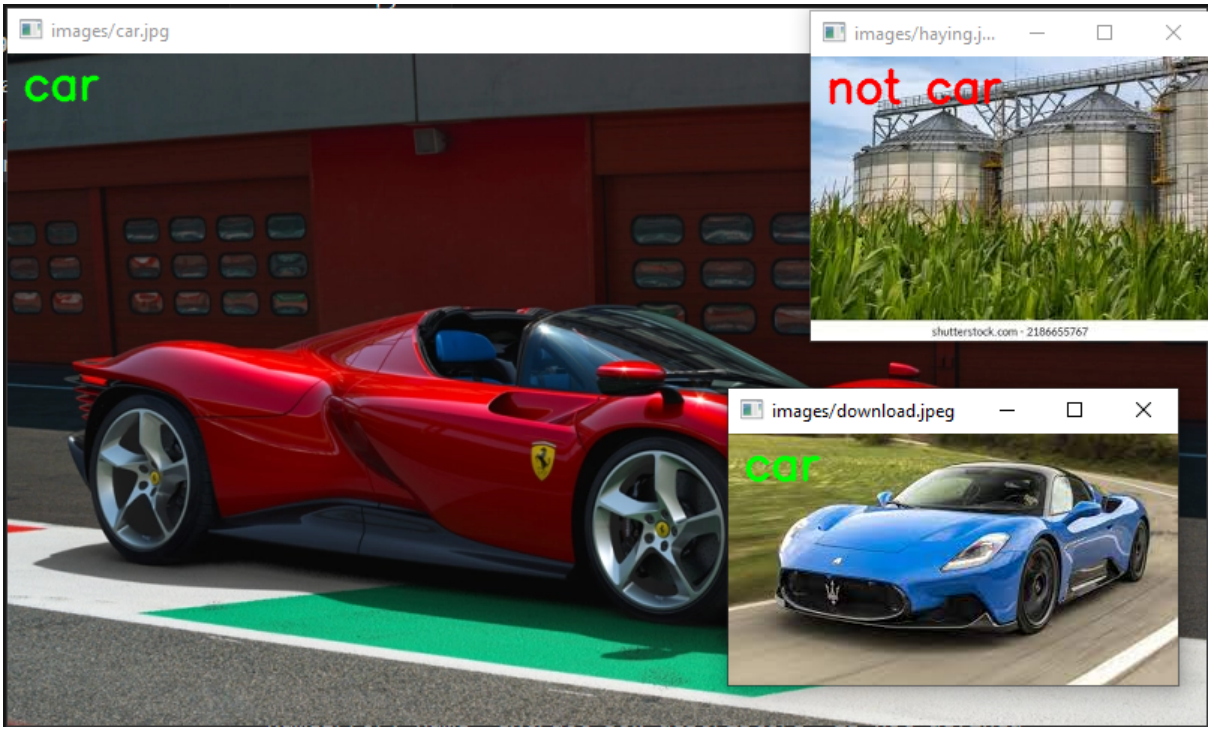
```
if not os.path.isdir('CarData'):
    exit(1)

# Define constants for the number of training samples and BoW clusters
BOW_NUM_TRAINING_SAMPLES_PER_CLASS = 10
SVM_NUM_TRAINING_SAMPLES_PER_CLASS = 110
BOW_NUM_CLUSTERS = 40

# Create a SIFT detector
sift = cv2.SIFT_create()

# Define FLANN parameters for BoW matching
FLANN_INDEX_KDTREE = 1
index_params = dict(algorithm=FLANN_INDEX_KDTREE, trees=5)
search_params = dict(checks=50)
flann = cv2.FlannBasedMatcher(index_params, search_params)

# Create a BoW K-Means trainer and BoW image descriptor extractor
bow_kmeans_trainer = cv2.BOWKMeansTrainer(BOW_NUM_CLUSTERS)
bow_extractor = cv2.BOWImgDescriptorExtractor(sift, flann)

# Function to get positive and negative image paths
def get_pos_and_neg_paths(i):
    pos_path = 'CarData/TrainImages/pos-%d.pgm' % (i+1)
    neg_path = 'CarData/TrainImages/neg-%d.pgm' % (i+1)
    return pos_path, neg_path

# Function to add SIFT descriptors to the BoW trainer
def add_sample(path):
```

```
    img = cv2.imread(path, cv2.IMREAD_GRAYSCALE)
    keypoints, descriptors = sift.detectAndCompute(img, None)
    if descriptors is not None:
        bow_kmeans_trainer.add(descriptors)

# Loop to add samples to the BoW trainer
for i in range(BOW_NUM_TRAINING_SAMPLES_PER_CLASS):
    pos_path, neg_path = get_pos_and_neg_paths(i)
    add_sample(pos_path)
    add_sample(neg_path)

# Cluster the SIFT descriptors to form the vocabulary
voc = bow_kmeans_trainer.cluster()
bow_extractor.setVocabulary(voc)

# Function to extract BoW descriptors
def extract_bow_descriptors(img):
    features = sift.detect(img)
    return bow_extractor.compute(img, features)

# Lists to store training data and labels
training_data = []
training_labels = []

# Loop to extract BoW descriptors for training data
for i in range(SVM_NUM_TRAINING_SAMPLES_PER_CLASS):
    pos_path, neg_path = get_pos_and_neg_paths(i)
    pos_img = cv2.imread(pos_path, cv2.IMREAD_GRAYSCALE)
    pos_descriptors = extract_bow_descriptors(pos_img)
```

```
    if pos_descriptors is not None:
        training_data.extend(pos_descriptors)
        training_labels.append(1)  # Positive class
    neg_img = cv2.imread(neg_path, cv2.IMREAD_GRAYSCALE)
    neg_descriptors = extract_bow_descriptors(neg_img)
    if neg_descriptors is not None:
        training_data.extend(neg_descriptors)
        training_labels.append(-1)  # Negative class

# Create an SVM classifier
svm = cv2.ml.SVM_create()

# Train the SVM using the training data
svm.train(np.array(training_data), cv2.ml.ROW_SAMPLE,
np.array(training_labels))

# Loop to test the classifier on test images
for test_img_path in ['CarData/TestImages/test-0.pgm',
                      'CarData/TestImages/test-1.pgm',
                      'images/car.jpg',
                      'images/haying.jpg',
                      'images/download.jpeg']:
    img = cv2.imread(test_img_path)
    gray_img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    descriptors = extract_bow_descriptors(gray_img)
    prediction = svm.predict(descriptors)
    if prediction[1][0][0] == 1.0:
        text = 'car'
        color = (0, 255, 0)
```

```
    else:
        text = 'not car'
        color = (0, 0, 255)
    cv2.putText(img, text, (10, 30), cv2.FONT_HERSHEY_SIMPLEX, 1,
color, 2, cv2.LINE_AA)
    cv2.imshow(test_img_path, img)


# Display the test results
cv2.waitKey(0)
```

Output:-



Conclusion

In summary, this project aims to create an object detector by applying the Bag of Words (BOW) technique in computer vision, with a specific focus on car detection. This demonstrates the versatility of computer vision for real-world applications and highlights the potential of BOW for accurate feature extraction and representation in object detection tasks.