VR Assignment2 Report

Srinivasan M, Siddharth Kothari, Sankalp Kothari

February 2024

Contents

1 Part A

1.1 Introduction

This part captures our attempts at solving the image stitching problem. It is described as follows -

1. Image Stitching - to detect interest points in 2 images, then match the interest points in the images. 2 Matchers have been used - Brute Force Matcher and K Nearest Neighbours Matcher. We then estimate the Homography and stitch the images together. Although we have implemented for 2 images, it can be extended to many images.
2. In the submission, we've only included images and outputs corresponding to the Cubbon Park images. The same can be done for the images of Vidhan Soudha and Brigade Road as well.

1.2 Libraries

The following libraries were used in the code :

- Numpy
- Matplotlib.pyplot
- Imageio
- OpenCV(cv2)

1.3 Input Images and Dataset

The input images for image stitching were obtained from the following places -

- *Vidhan Soudha*
- *Cubbon Park*
- *Brigade Road*

1.4 Differences between SURF and SIFT

We shall first look into each of them separately and then tabulate the differences.

1.4.1 SIFT

SIFT (Scale-Invariant Feature Transform)

- **Introduction:** SIFT was developed by D. Lowe in 2004 at the University of British Columbia to address the challenge of scale variance in feature extraction.
- **Components:** The algorithm can be mainly split into two parts namely:
  - **Key-point Detection:** Utilizes Difference of Gaussian (DoG) to approximate Laplacian of Gaussian. Local extrema in a 3x3x3 neighborhood are identified as key-points See Fig 1 for reference.

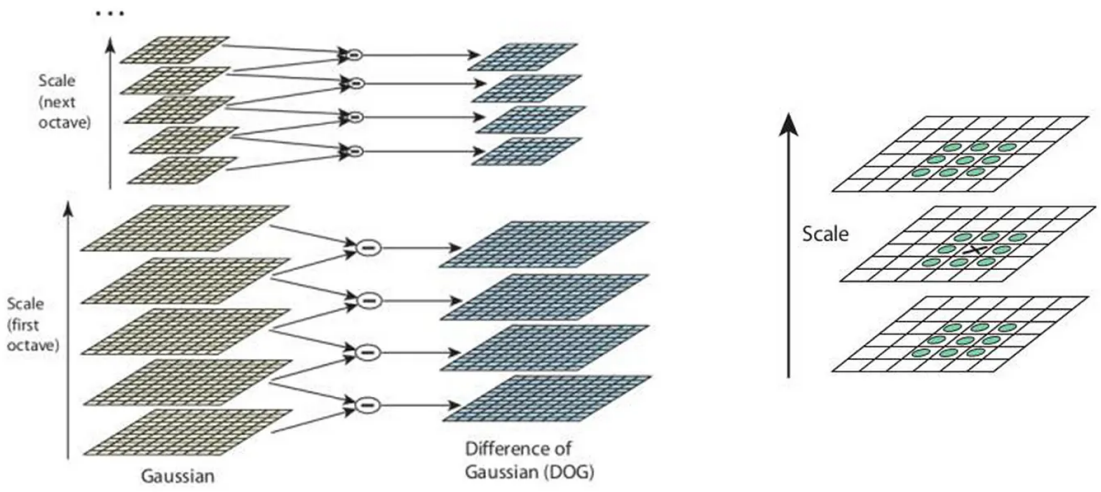

Figure 1: Difference of Gaussians Image

  - **Orientation Assignment:** For each key-point, a neighborhood is examined to create an orientation histogram. The peak of the histogram (or any peak above 80%) is chosen as the orientation.
  - **Descriptor Generation:** A 16x16 neighborhood around each key-point is divided into 4x4 cells. An orientation histogram is computed for each cell, and these histograms are concatenated to form a 128-dimensional feature descriptor. See Fig 2 for better understanding of Descriptor generation.

- **Benefits:** SIFT offers robustness to changes in scale, rotation, and illumination, making it widely used in computer vision tasks such as object recognition and image stitching.

1.4.2 SURF

SURF (Speeded-Up Robust Features)

- **Introduction:** SURF was developed in 2006 as an improvement on SIFT, aiming to enhance the algorithm's speed while maintaining robustness.
- **Key Features:**
  - **Scale Detection:** Utilizes Box Filters instead of Difference of Gaussian to approximate Laplacian of Gaussian. Box filters enable faster computation and simultaneous calculations for different scales.
  - **Orientation Assignment:** Computes Haar-wavelet responses in both x and y directions within a 6s neighborhood around the key-point, with a sampling step proportional to scale. The orientation is determined based on the sum of responses in a sliding scanning area.

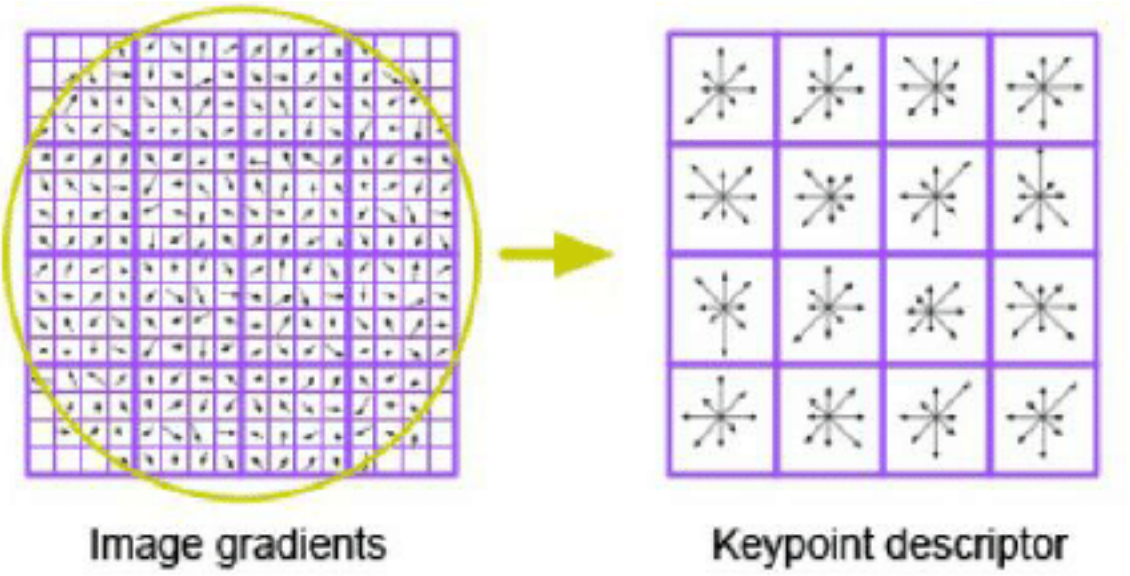

Figure 2: SIFT-Descriptor-generation-2

  - **Descriptor Generation:** Extracts a 20s x 20s neighborhood around each key-point and divides it into 4x4 cells. The x and y wavelet responses are obtained for each cell, and the responses from all cells are concatenated to form a 64-dimensional feature descriptor.

- **Benefits:** SURF offers improved computational efficiency while retaining robustness to scale, rotation, and illumination changes. It is widely used in real-time applications such as object recognition and tracking.

1.4.3 ORB

ORB (Oriented FAST and Rotated BRIEF)

- **Introduction:** Developed in 2011, ORB combines the FAST (Features from Accelerated Segment Test) keypoint detector with the BRIEF (Binary Robust Independent Elementary Features) descriptor. It enhances computational efficiency while maintaining robustness.

- **Key Features:**

  - **Feature Detection:** ORB initially uses the FAST algorithm to identify keypoints in the image. FAST identifies keypoint candidates by comparing pixel intensities in a circular neighborhood.
  - **Orientation Assignment:** Unlike SURF, ORB does not compute a specific orientation for keypoints. Instead, it employs an orientation calculation method based on the intensity centroid of the patch around the keypoint.
  - **Descriptor Generation:** After keypoint detection, ORB generates feature descriptors using the BRIEF algorithm. BRIEF computes binary descriptors by comparing intensity comparisons between pairs of points within a predetermined neighborhood around the

keypoint. This results in a binary feature vector that is more computationally efficient to process compared to the real-valued descriptors used in SIFT and SURF.

- **Benefits:** ORB offers a balance between computational efficiency and robustness. It is particularly suitable for real-time applications where speed is critical, such as mobile robotics, augmented reality, and object tracking. Despite its simplified approach compared to SIFT and SURF, ORB still maintains good performance in terms of scale, rotation, and illumination changes.

Feature	**SIFT**	**SURF**	**ORB**
Speed	Slow	Faster than SIFT, but slower than ORB	Fast
Robustness	Robust to rotation, scale, illumination changes	Robust to rotation, scale, illumination changes	Robust to rotation, scale, illumination changes
Descriptor Size	Larger (128 to 512 dimensions)	Smaller (64 dimensions)	256 bits (32 bytes)
Scale-Space Extrema Detection	Difference of Gaussians (DoG)	Box Filters	FAST corner detection with pyramid
Computational Complexity	High	Moderate	Low
Keypoint Localization	Sub-pixel accurate	Sub-pixel accurate	Integer-based
Matching Accuracy	High	High	Good
Applications	Image stitching, object recognition	Real-time applications, object tracking	Real-time applications, robotics

1.5 Principles of FLANN matching and RANSAC

1.5.1 FLANN

- **Efficiency:** FLANN (Fast Library for Approximate Nearest Neighbors) is designed for efficiently finding approximate nearest neighbors in high-dimensional spaces.
- **Data Structures:** FLANN employs data structures like KD-trees or hierarchical k-means trees to accelerate the search process, enhancing computational efficiency.
- **Feature Matching:** It is particularly useful for tasks like feature matching in computer vision, where finding similar features in large datasets is essential.
- **Approximate Neighbors:** FLANN utilizes approximate nearest neighbors instead of exact ones, balancing between accuracy and computational efficiency.
- **Large-scale Applications:** Due to its efficient search capabilities, FLANN is suitable for large-scale applications such as image retrieval and object recognition.

1.5.2 RANSAC

- **Robust Estimation:** RANSAC is a robust estimation method used to fit models to data containing outliers, making it suitable for noisy datasets.

- **Random Sampling:** It randomly selects minimal subsets of data points to fit models, which helps in mitigating the influence of outliers.
- **Consensus Set:** RANSAC evaluates the model against the remaining data to find the consensus set, which consists of the inliers that support the estimated model.
- **Iterative Refinement:** It iteratively refines the model by repeating the sampling and fitting process until a termination criterion is met.
- **Applications:** RANSAC is commonly used in tasks such as line fitting, homography estimation, and robust feature matching in computer vision and image processing.

1.6 Experiments in Image Stitching

There were broadly two experiments that we had done in this part. One was trying out the image stitching mechanism using the *RANSAC* method and the other one was using the *FLANN* method.

1.6.1 Using RANSAC

RANSAC stands for Random Sample Consensus. It's an iterative method used for estimating parameters of a mathematical model from a set of observed data that contains outliers.

We first take the two images that have to be stitched together namely the *Query Image* and the *Train Image*.

We then use the *SIFT* and the *ORB* detectors to find the features and keypoints of the images and plot the keypoints on the image using the imshow function of OpenCV. A comparison of the results is provided later.

To match the features across images, we've implemented two kinds of matchers (the distance metric used is L2Norm in both cases) -

1. Brute Force Matcher - this will take one descriptor from the first image and compare it to every descriptor in the second image. This is done for all descriptors in the first image, and then the closest one is matched.
2. Brute Force KNN Matcher - this will perform essentially the same thing, but it will return the k nearest neighbours. We then perform the Lowe's ratio test to return the matches.

We now have access to the keypoints in both the images, and the feature matches. We then compute the points corresponding to each of the matches, and the Homography Matrix is computed using the RANSAC algorithm. RANSAC is used to effectively compute the Homography matrix in presence of outliers.

Finally, once we have the Homography matrix, we use the warpPerspective function of OpenCV to stitch together the images using the Homography matrix. This gives us the stitched image.

1.6.2 Using FLANN

FLANN (Fast Library for Approximate Nearest Neighbors): FLANN is a library that provides fast algorithms for approximate nearest neighbor searches in high-dimensional spaces. It offers various methods for indexing and searching high-dimensional data efficiently. FLANN is particularly useful in tasks such as image matching, clustering, and similarity search.

We begin by taking two images to be stitched together: the Query Image and the Train Image.

Using the SIFT detector and the ORB detector, we extract keypoints and compute feature descriptors for both images. These keypoints represent distinctive points in the images that can be matched across them.

To match features across the images, we employ FLANN (Fast Library for Approximate Nearest Neighbors) for efficient nearest neighbor search. FLANN operates by building an index structure to perform fast approximate nearest neighbor queries.

1. FLANN-Based Matcher: FLANN is utilized to match features between the images. It employs a k-d tree algorithm for approximate nearest neighbor search.

2. Matching Strategy: FLANN's K-nearest neighbor (KNN) matching approach is employed. This method returns the K nearest neighbors for each descriptor in one image from the descriptors in the other image.

3. Lowe's Ratio Test: To ensure high-quality matches, we apply Lowe's ratio test to the nearest neighbor matches obtained from FLANN. This test helps filter out unreliable matches by comparing the distance ratios between the best and second-best matches for each query descriptor.

After obtaining the keypoints in both images and filtering for good matches, we compute the points corresponding to each of these matches.

Homography Estimation with RANSAC: Using the matched keypoints, we estimate the homography matrix that describes the transformation between the images. RANSAC (Random Sample Consensus) algorithm is employed to robustly estimate this transformation in the presence of outliers or mismatches.

Finally, once we have the homography matrix, we use the warpPerspective function from OpenCV to stitch together the images. This function applies the computed homography matrix to warp one image onto the other, aligning them properly. The result is a seamless panorama of the stitched images.

1.7 Observations

The following were the observations that we saw while using both the methods,

1. Both were very fast in-terms of computation.

2. They return very similar results in terms of quality and the image stitching.

Overall we feel that both of them can be used in normal scenarios but then from what we understood reading about these methods we feel that :

1.7.1 RANSAC over FLANN

RANSAC is often employed in scenarios where model parameter estimation is required from noisy data contaminated by outliers. Here are some situations where RANSAC might be preferred over FLANN:

1. **Homography Estimation:** In image registration tasks such as panorama stitching or object recognition, RANSAC is commonly utilized to estimate the homography matrix that describes the transformation between two images. RANSAC's robustness to outliers is crucial in such scenarios where feature correspondences may contain mismatches or outliers.

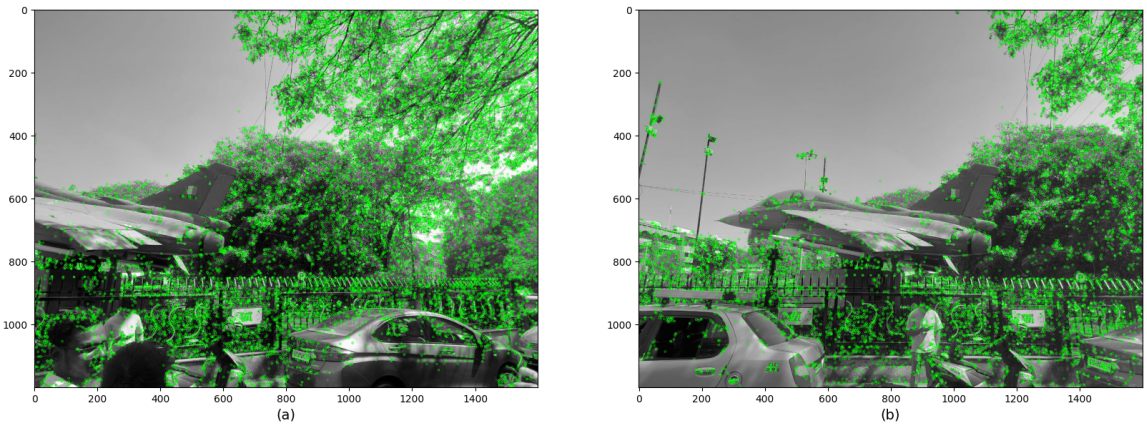

Figure 3: Keypoints of the Cubbon Park Image Using SIFT Detector

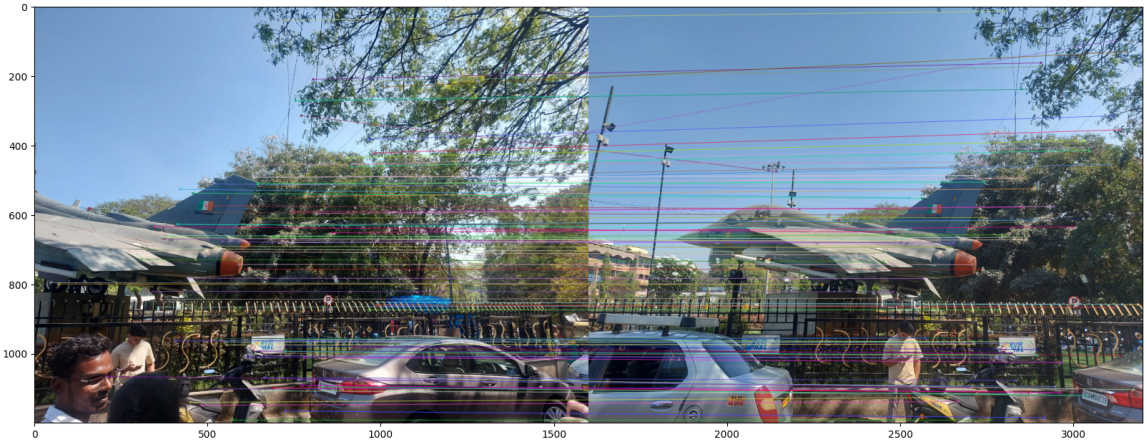

Figure 4: Keypoints Matching of the Cubbon Park Image (SIFT Detector)

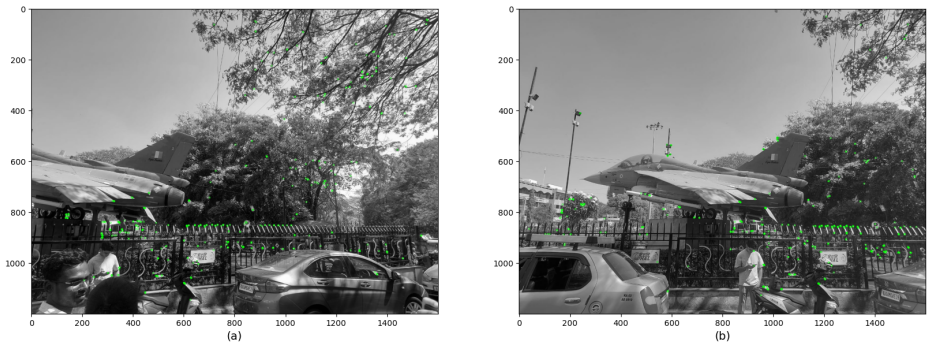

Figure 5: Keypoints of the Cubbon Park Image Using ORB Detector

Figure 6: Keypoints Matching of the Cubbon Park Image (ORB Detector)

Figure 7: Ransac stitched image of the Cubbon Park Image(SIFT Detector)

Figure 8: FLANN stitched image of the Cubbon Park Image

2. **Pose Estimation:** In the context of structure from motion or visual odometry, RANSAC can be employed to estimate the camera pose from feature correspondences between consecutive frames. This is particularly useful in scenarios where the correspondence data might be noisy or contain outliers.

1.7.2 FLANN over RANSAC

FLANN, on the other hand, is primarily utilized for efficient approximate nearest neighbor search in high-dimensional spaces. Here are some scenarios where FLANN might be preferred over RANSAC:

1. **Feature Matching:** In tasks such as image stitching or object recognition, FLANN can efficiently match keypoints or feature descriptors between images. This accelerates the matching process, especially with large datasets, compared to exhaustive search methods.
2. **Clustering:** FLANN finds utility in clustering applications where approximate nearest neighbor search is necessary, such as in k-means clustering algorithms or hierarchical clustering methods.

In summary, while **RANSAC** is typically employed for robust model estimation in the presence of outliers, **FLANN** is used for efficient approximate nearest neighbor search in high-dimensional spaces. The choice between these techniques depends on the specific requirements and constraints of the task at hand.

2 PART B

2.1 Introduction

This part of the projects captures our aim of solving the *Image classification problem.* It is described as follows:

- Image Classification - to carry out image classification into two categories - bike or horse, using bag of visual words approach. Interest points and features are detected using SIFT detector, following which the features are clustered, and classification is carried out using KNN, SVM and Logistic Regression, and the results are compared. We also extend the approach to CIFAR10 dataset.

2.2 Libraries

The following libraries were used in the code :

- Numpy
- Tensorflow(for CIFAR)
- Opencv(cv2)
- Sklearn

2.3 Input Images and Dataset

The input images for image classfication were obtained from the following places -

- Dataset containing pictures of Bikes and Horses.
- CIFAR10 Dataset

2.4 Procedure for Bag of Words in Image processing

The following are the steps in the *Bag of Words* approach:

1. **Data Collection**: Gather a balanced dataset of images for different classes.
2. **Preprocessing**: Resize and preprocess images (e.g., grayscale conversion).
3. **Feature Extraction**: Extract local features (e.g., SIFT descriptors) from images.
4. **Codebook Generation**: Apply K-Means clustering to create a visual vocabulary.
5. **Conversion to Vector representation**: Quantize image features based on the codebook to create histograms. Represent images as feature vectors
6. **Model Training**: Train a classifier (e.g., SVM, KNN) on training data.
7. **Model Evaluation**: Evaluate classifier performance on testing data.
8. **Parameter Tuning**: Fine-tune model parameters if needed.

2.5 Bikes and Horses

2.5.1 Approach

Our approach consisted of the following:-

1. First load the appropriate datasets.
2. Then use the Feature extractor and then get the descriptors and the features of the image. We again try out two detectors for this, and compare the results in the observations section. We do this for all of the images that we have gotten from the dataset.
3. Then we are using the *Kmeans* to fit on the descriptors, which will help in creating the histogram.
4. Each of the cluster that the KMeans returns will be the labels for the histogram (in the vector representation).
5. Then we will build the histogram and then use the necessary models to fit the data and classify the images when required.
6. We use four main classifiers for this purpose -
   (a) SVM (without any kernel function)
   (b) SVM with linear kernel
   (c) Logistic Regression
   (d) K Nearest Neighbours Classifier (neighbours = 5)

2.5.2 Observations

A comparative study of the results of SIFT and ORB detectors is provided here.

The following were the observations when we used the SIFT Detector.

Table 1: Comparison of Models based on Accuracy with SIFT Detector

Model	**Accuracy**
SVC	0.9722
SVC (linear kernel)	0.9722
Logistic Regression	1
K-Nearest Neighbors (k=5)	0.9722

The following were the observations when we used the ORB Detector.

Table 2: Comparison of Models based on Accuracy with ORB Detector

Model	**Accuracy**
SVC	0.9444
SVC (linear kernel)	0.8333
Logistic Regression	0.8888
K-Nearest Neighbors (k=5)	0.8055

1. On the whole, K Nearest Neighbours was not suited for this task, for both the types of detectors.
2. SVC and Linear Regression performed the best out of all the models, with Logistic Regression giving a perfect accuracy when used with the SIFT Detector.
3. SIFT provided much better results than ORB over all the models.
4. We include some images in this report for which all the models performed poorly, to demonstrate shortcomings of this approach.

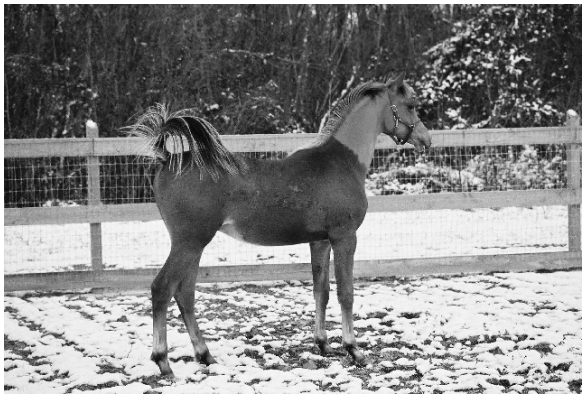

Figure 9: Classified consistently as Bike

Figure 10: Classified consistently as Horse

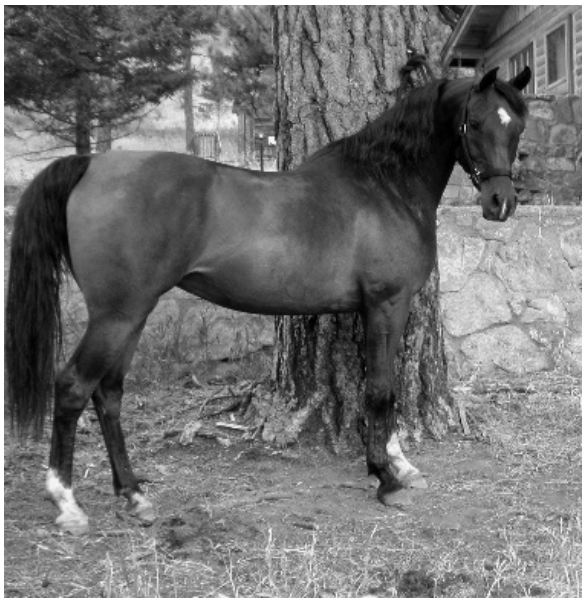

Figure 11: Classified consistently as Bike

Figure 12: Classified consistently as Bike

2.6 Cifar-10

2.6.1 Approach

We had tried two different datasets for this:-

- Dataset from Tensorflow
- Dataset from *Toronto - Education* website

Only acquiring the datasets were different because we had to try various methods and write out experiments and observations in the report. The steps for the approach is same as the previous one.

2.7 Observations

The comparative study for the two datasets is described below:
The following is the results for the *Tensorflow* dataset.

Table 3: Comparison of Models based on Accuracy with *Tensorflow* dataset

Model	**Accuracy**
SVC	0.1004
Logistic Regression	0.104
K-Nearest Neighbors (k=5)	0.099

The following is the results for the *Toronto Education*

Table 4: Comparison of Models based on Accuracy with *Toronto Education* dataset

Model	**Accuracy**
SVC	0.2322
Logistic Regression	0.2207
K-Nearest Neighbors (k=5)	0.1614

The values that we got from training the

3 References

References

[1] Toronto Education Wesbsite for Cifar - 10 classes: https://www.cs.toronto.edu/~kriz/cifar.html

[2] Bag of Words Approach https://towardsdatascience.com/bag-of-visual-words-in-a-nutshell-9ceea97ce0fb

[3] Panorama Stitching https://towardsdatascience.com/image-panorama-stitching-with-opencv-2402bde6b46c

[4] Feature Detection and Matching https://medium.com/@deepanshut041/introduction-to-feature-detection-and-matching-65e27179885d

[5] Matplotlib Documentation https://matplotlib.org/stable/index.html

[6] OpenCV Documentation https://docs.opencv.org/4.x/index.html

[7] Scikit-Learn Documentation https://scikit-learn.org/

[8] Numpy Documentation https://numpy.org/doc/

[9] Pandas Documentation https://pandas.pydata.org/docs/

[10] TensorFlow Documentation https://www.tensorflow.org/api_docs