ELEN644 Homework 1 Report

Justin Goh

January 30, 2023

Introduction

The images used for this assignment were taken from the CIFAR-10 dataset via Tensorflow (see Figure 40). Code was written in Python.

Figure 1: Loaded images from CIFAR-10

1 Problem 1: Hough Transform

1.1 Questions

For 10 images from the 10 classes of the CIFAR-10 dataset:

Apply Hough Line and Circle Transform. Plot line and circle detection outputs for each image.

For both algorithms, explore different parameters and fine-tune them (i.e increasing/decreasing threshold values of the line and circle detector). Justify your choice of values for parameters for both algorithms.

1.2 Analysis

Hough Lines:
The Hough transform can detect (i.e lines, circles) even if the shape is broken or distorted. Lines can be represented in cartesian or polar coordinate system. Lines in the image space (xy space) can be represented as a POINT in the Hough space (mc space). The inverse is also true. Generally, we use polar coordinate system. This is because the cartesian coordinate system cannot represent vertical lines. The general steps of the Hough Line transform involves: 1 Edge detection (i.e Canny edge), 2 Mapping of edge points to Hough space and storage in an accumulator, 3 Interpretation (vias thresholdoing) of the accumulator to yield lines of infinite length, 4 Conversion of infinite lines to finite lines

My function is set up do some preprocessing before determining the Hough Lines. This includes converting the image to grayscale, applying a Gaussian filter and applying a Canny Edge Detector. The function will plot the Hough Lines on the image.

The rho value is the perpendicular distance from origin to the line. Smaller rho values yields a finer grid that can capture more details of a line. After adjusting the values for rho, the general trend in the plotted images is that the smaller the value of rho, the more lines are detected.

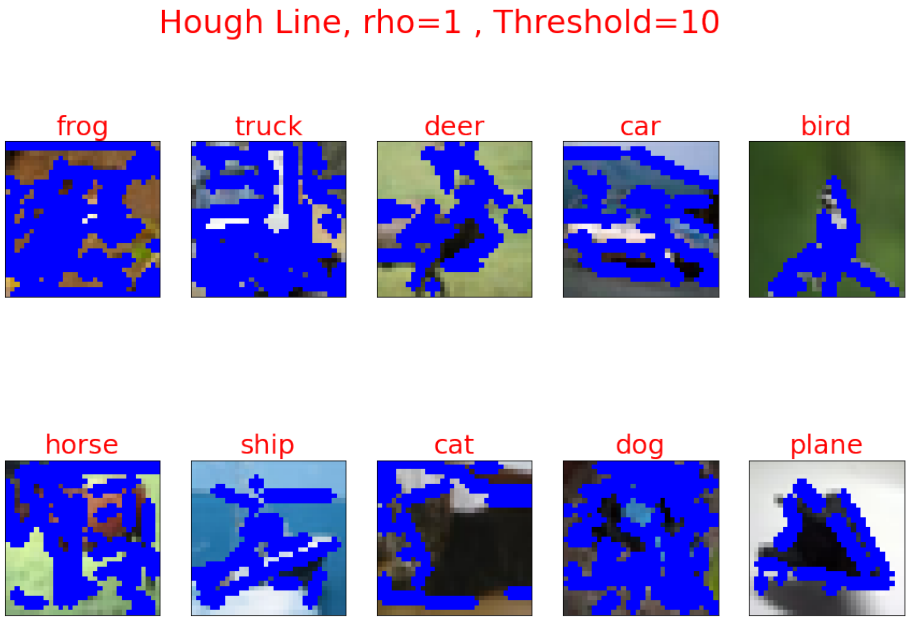

Figure 2: rho=1

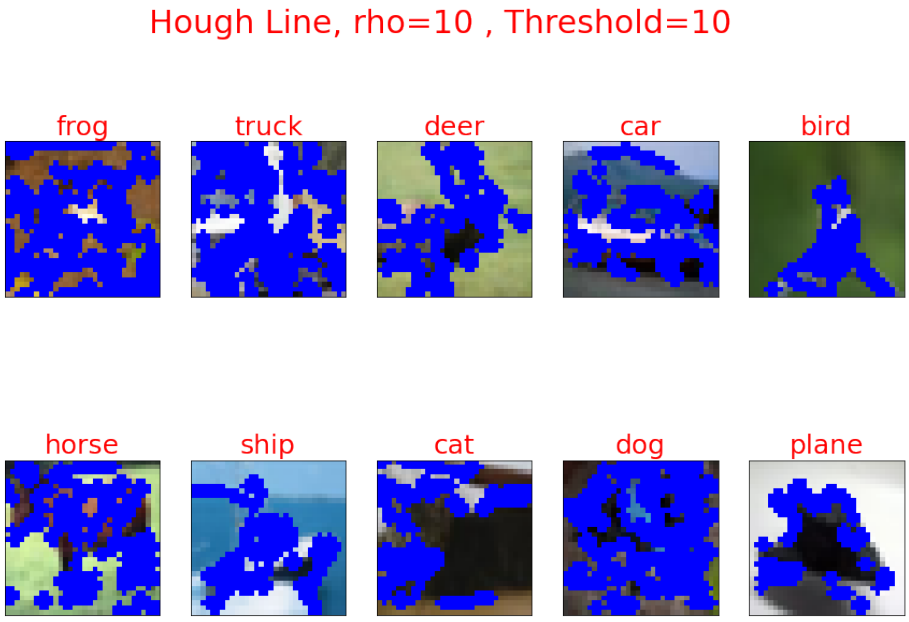

Figure 3: rho=10

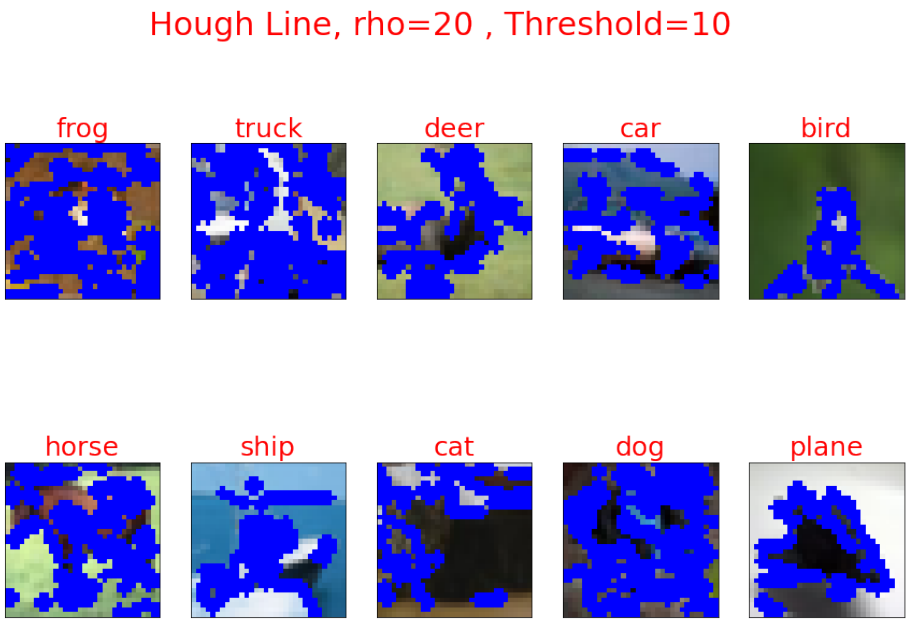

Figure 4: rho=20

Theta is the angle formed by the line rho. The smaller the value of theta, the more line details are captured in the image. As theta increased, the number of lines detected decreased. The effect of different theta is seen as below.

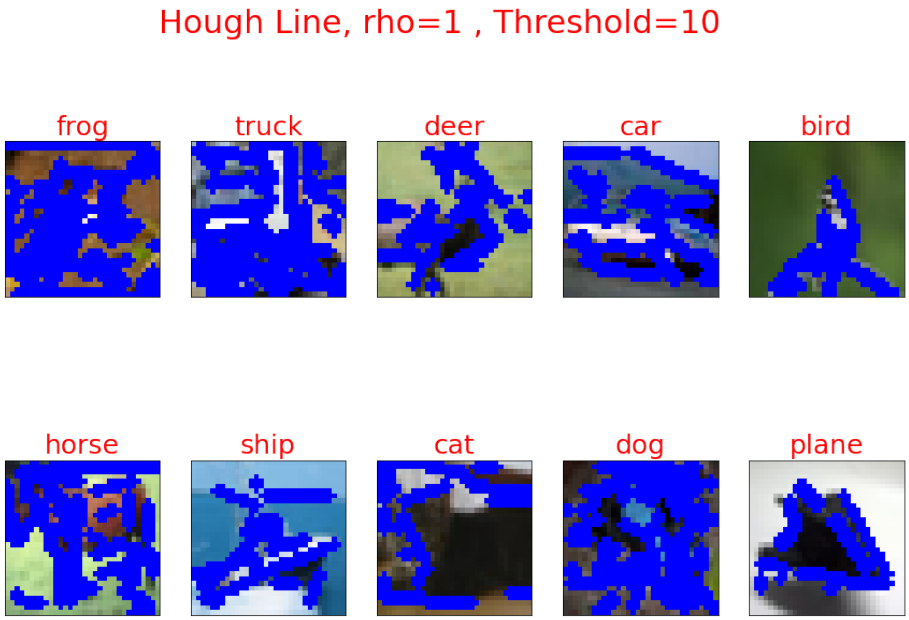

Figure 5: theta=180/pi

Figure 6: rho=20

The threshold parameter serves as a counter/accumulator for how many "votes" a line gets to be detected and considered a line. The line is not considered a line if its' number of "votes" is less than the threshold value. Setting a threshold value too high will result in not many lines being detected. (or no lines being detected) Setting a threshold value too low will result in many lines being detected. One drawback from this is that you may get false lines.

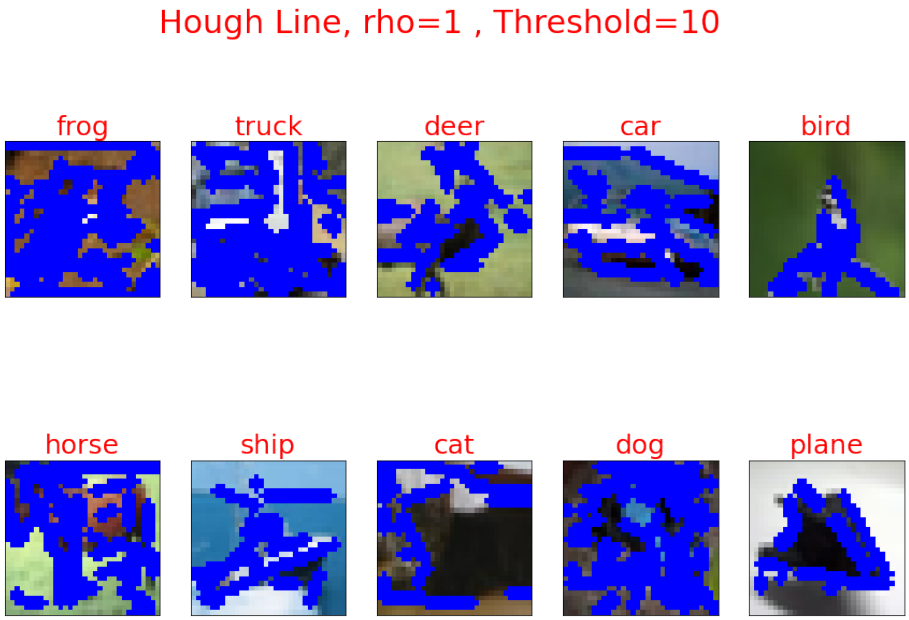

Figure 7: thereshold=10

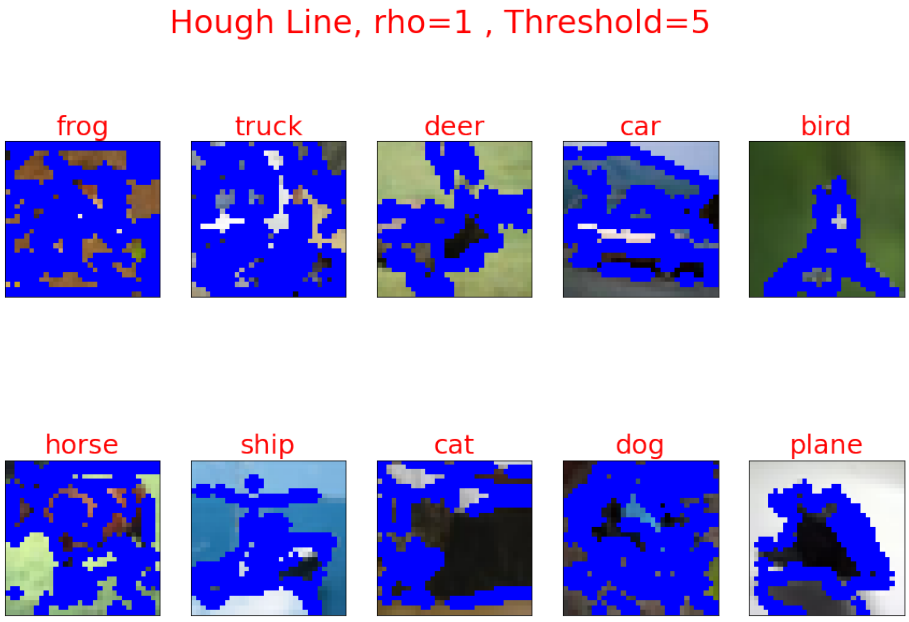

Figure 8: thereshold=5

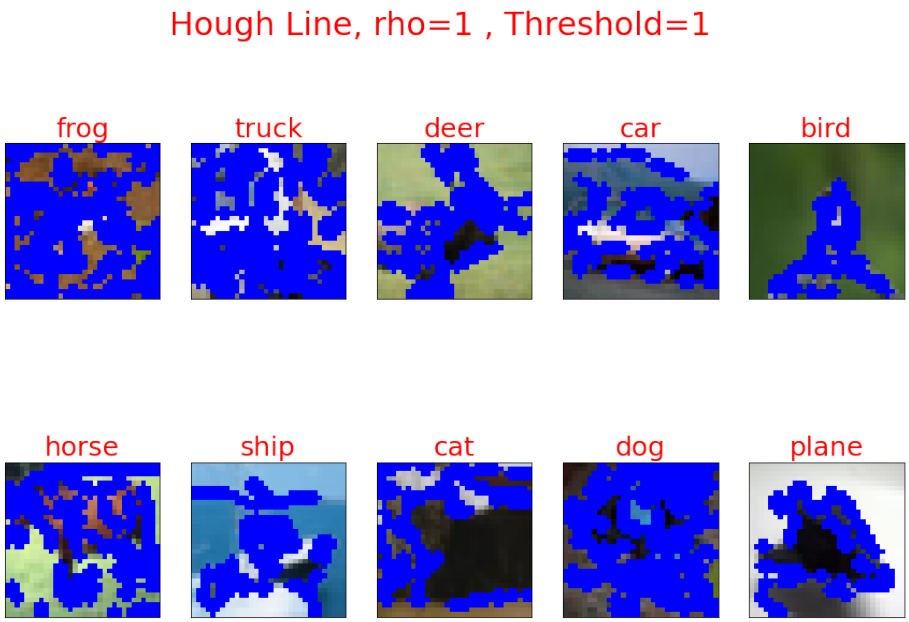

Figure 9: thereshold=1

The minLineLength parameter is the minimum allowed length of line. This parameter was set to a value of 1 since the size of the images were small. This allowed easier formation of lines. If this value was too large, lines could not be formed on the image. The maxLineGap parameter is the maximum allowed gap between line segments that treat them as a single line. This value was set to 1.

After adjusting all the parameters, the plot in Figure 10 the most ideal Hough Line detector. The value of rho wasn't set too low (rho=10) since it was observed earlier that many false lines were created as a result of a low rho value. Theta was set to pi/180, since increasing theta resulted in not many lines being detected. This value was not decreased to avoid the creation of false lines. After adjusting the other parameters, the threshold param was adjusted to try and capture as many edges as possible. The resulting image had mixed results. For the image of the deer and bird, it was able to capture the outline of the animals fairly well. For the image of the frog and truck, there were a lot of background elements and resulted in many lines being formed. It is harder to identify what the image content for the frog and truck are.

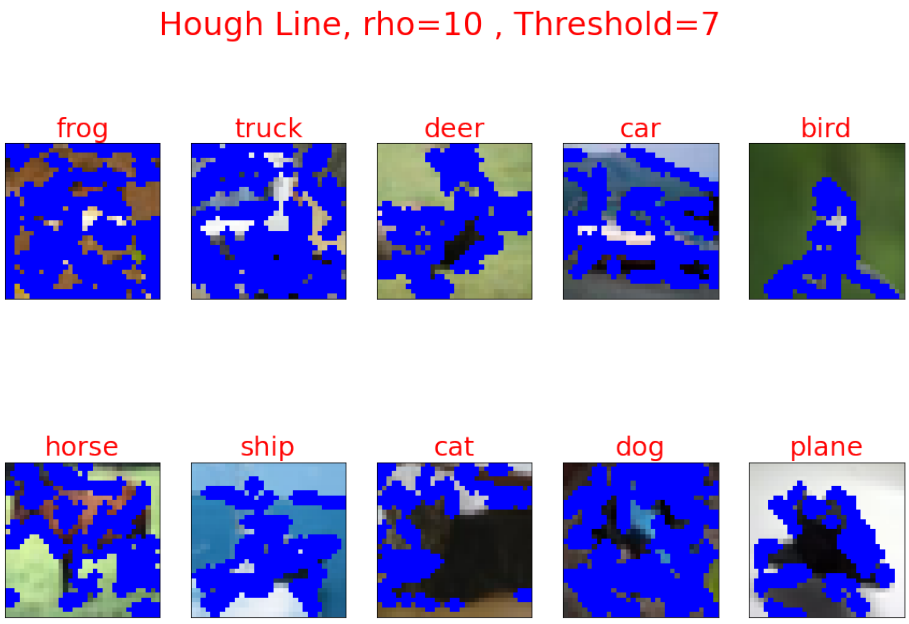

Figure 10: Plot of function with optimized Hough Line parameters for the given image set

Hough Circles:
In my function, some preprocessing is done to the image before applying the Hough Circle Transformation. This includes converting the image to grayscale and applying a Gaussian blur (for noise removal). The parameters experimented with are dp, minDist, param1 and param2.

dp is the inverse ratio of the accumulator resolution to the image resolution. A dp value of 1 indicates that the accumulator has the same resolution as the iamge, whereas a dp value of 2 means the accumulator will have half the resolution. Having a lower dp value will allow you to detect smaller circles that may be present in the image. A higher dp value means yield a lower accumulator resolution. When changing the values between 1 and 2, while other parameters were held constant, it was observed that dp=2 produced Hough circles while dp=1 did not. This could be due to the image having low-resolution and not contain enough information to detect circles. The other parameters may also need to be adjusted to produce more Hough cirlces. For the rest of my exploration I will use a dp value of 2.

Figure 11: dp=1

Figure 12: dp=2

The **minDist** parameter is the minimum distance between centers of detected circles. This will reject circles that are too close together. openCV's HoughCircles function has the default value set to 100. In my function it is reduced to 10 due to the smaller size of my images. Reducing this value all the way down to 1 did not change the number of circles detected as seen in Figure 12. Other parameters may need to be changed to observe any changes in the number of circles detected.

param1 is the higher of the 2 threshold passed into the Canny Edge detection algorithm. Pixels above this threshold are considered strong edges. It sets the sensitivity of how strong an edge of a circle needs to be. If this value is set too high, little to no circles will be detected. Setting this value too low will cause many false circles to be produced. Figures 13 to 15 shows that as I decreased the value of param1, more circles were detected. Figure 16 shows less circles produced when the value of param1 was set to 150.

Figure 13: param1=25

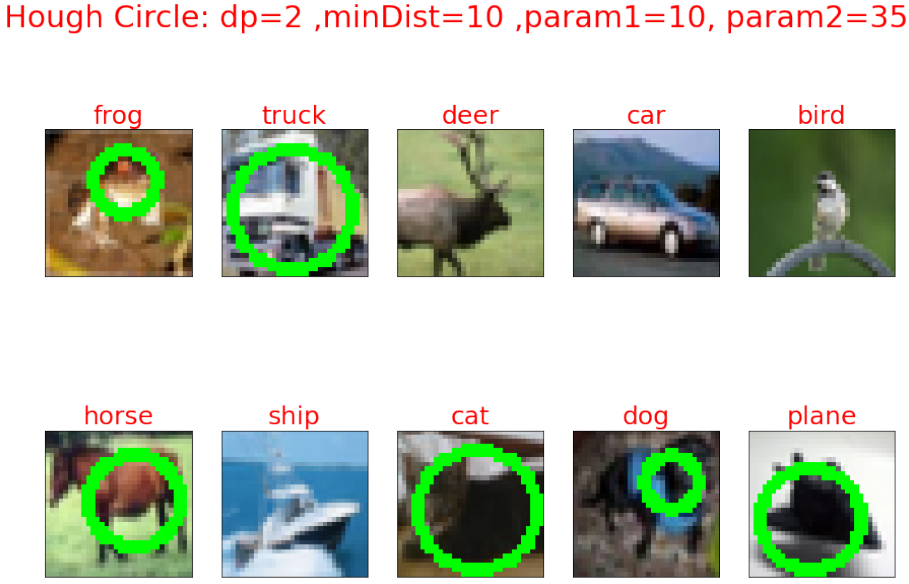

Figure 14: param1=10

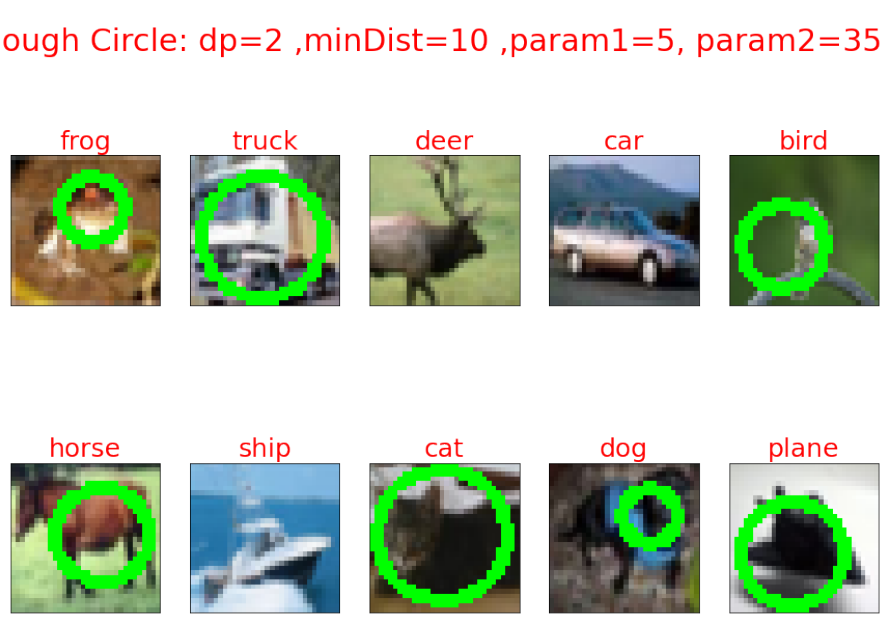

Figure 15: param1=5

Figure 16: param1=150

param2 is the accumulator threshold for the circle centers during the detection stage. The Hough Circle transform accumulates votes for each candidate circle center, param2 serves as the threshold for considering a candidate circle as a legitimate circle. The smaller this value, the more false circles will be detected. Increasing this value will result in less circles being detected. As shown below as the value of param2 increases, the number of circles detected decreases.

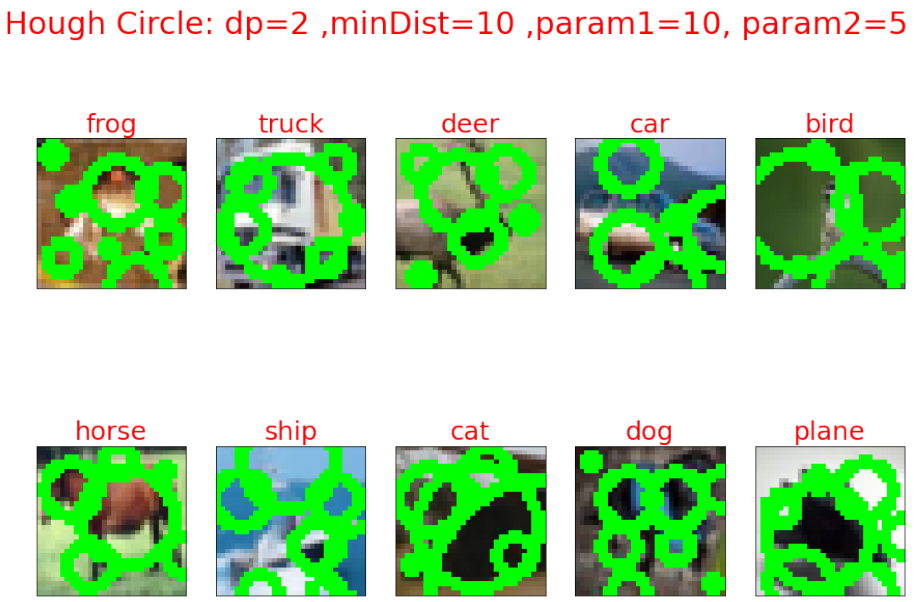

Figure 17: param2=5

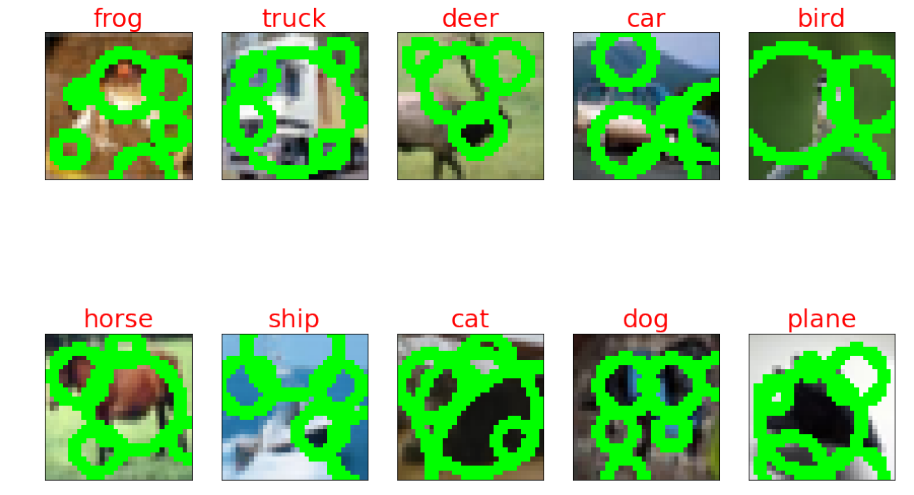

Figure 18: param2=10

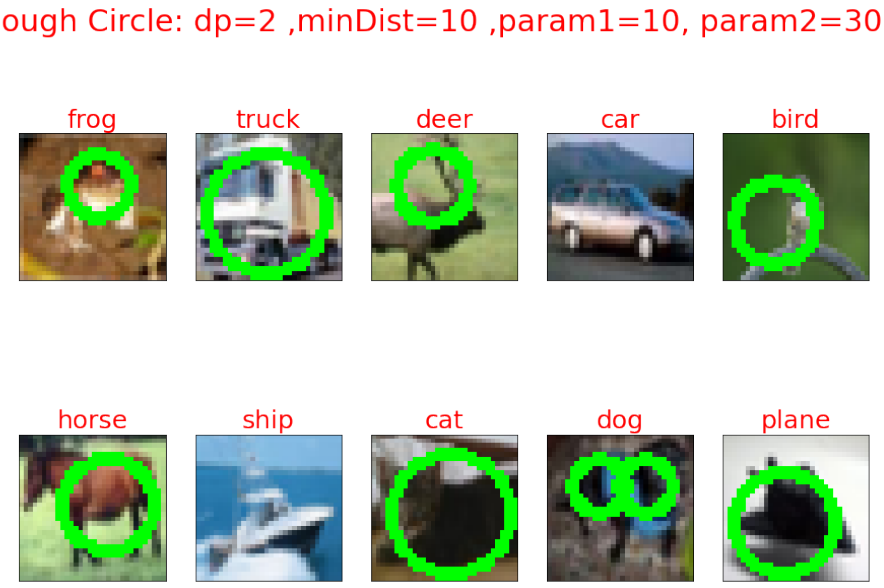

Figure 19: param2=30

After adjusting the parameters to create the best possible circle detector, the resulting Figure was produced. dp was set to 2 and minDist was set to 1 due to the low resolution images I am working with. param1 was given a relatively low value of 10. This is to allow edges to be detected more easily in my image set. param2 was set to 25. The resulting plot formed circles in areas where there was a bit of a curvature, however it should be noted on the image with a truck and car, the Hough Circle detector struggled to detect the curvature of the wheels. This could be due to the image having a low resolution and wheels looking more like lines than a curve.

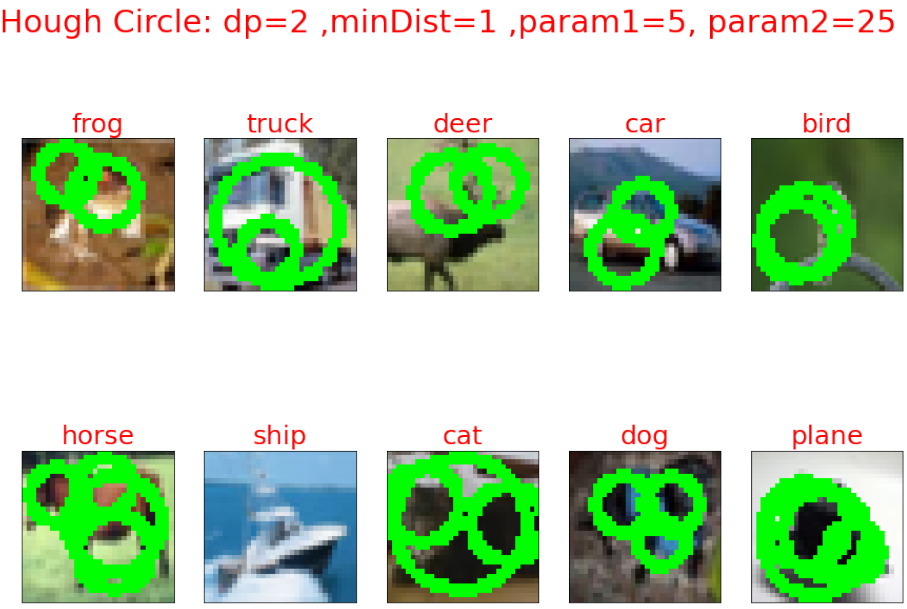

Figure 20: Plot of function with optimized Hough Circle parameters for the given image set

2 Problem 2: SIFT Feature Detector and Descriptor

2.1 Questions

Apply SIFT to your 10 images. Study the effects oF sigma, edge threshold and contrast threshold
Plot the interest points or key points detected by SIFT for all the images. Justify your choice of your parameter values.

2.2 Analysis

Sigma:
The value of sigma here is the sigma term applied to a Gaussian filter. In this context, the larger the sigma value, the more "blurred" the resulting image becomes.

In the SIFT algorithm, the sigma value is used to determine the scale of the blobs detected. A larger sigma value can detect larger blobs (larger keypoints), at the risk of smaller image details being smooothed out. A smaller sigma can detect smaller blobs (smaller keypoints). Using multiple sigma values allows

SIFT to detect scale-invariant keypoints in an image. The effectiveness of the sigma value in detecting keypoints in the image depends on the image content.

Since the images from CIFAR-10 are only 32x32 in size, having too large of a sigma will result in no blobs being detected. Increasing the value of sigma will result in less keypoints being detected in the image. Decreasing the value of sigma will result in more keypoints being detected. This is due to the image already being small and having a low resolution. With lower sigma values, we are able to detect "blobs" that are smaller which allows us to detect more keypoints on a small image. If the CIFAR-10 image was larger, we may be able to detect more keypoints if we increased the value of sigma. Below shows the different number of keypoints being detected as the value of sigma changes. All other parameters are held constant.

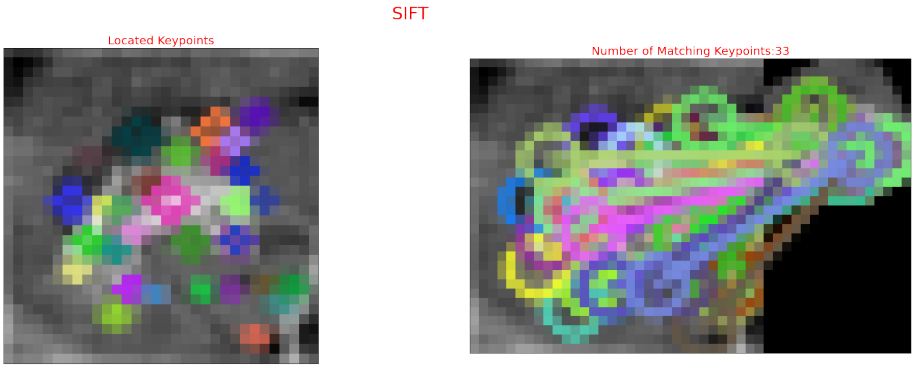

Figure 21: $\sigma = 1, keypoints = 33$

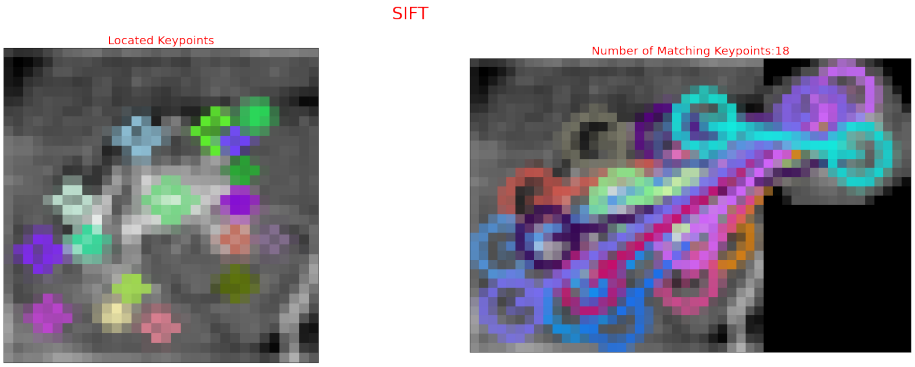

Figure 22: $\sigma = 1.6, keypoints = 18$

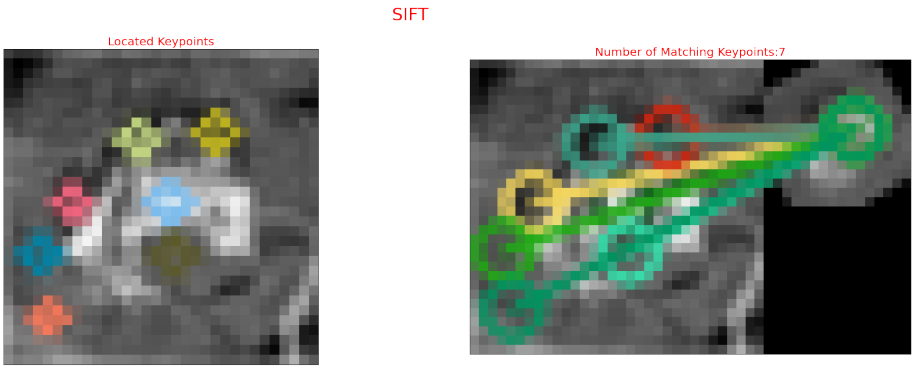

Figure 23: $\sigma = 2.5, keypoints = 7$

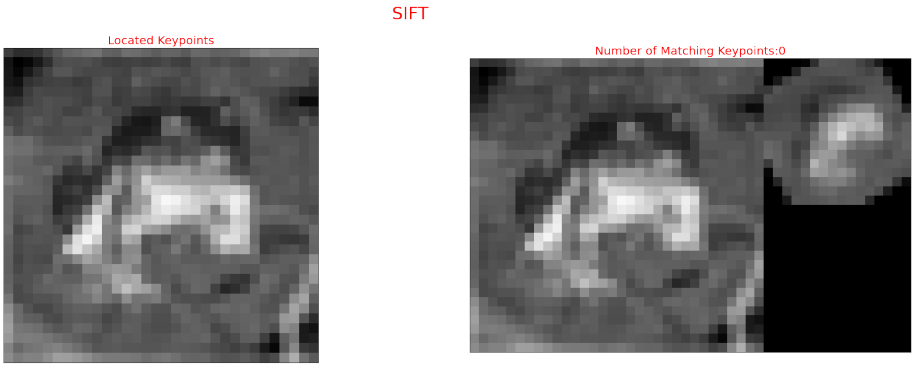

Figure 24: $\sigma = 4, keypoints = 0$

Edge Threshold:
The edge threshold parameter filters out edge-like features. The larger the edgeThreshold, more features are retained. Depending on the image content, the importance of edge-like features may vary. It was observed that increasing the edgeThreshold to a certain point would not increase the number of keypoints detected. Below shows how the number of keypoints change as value of the edgeThreshold increases. All other parameters are held constant.

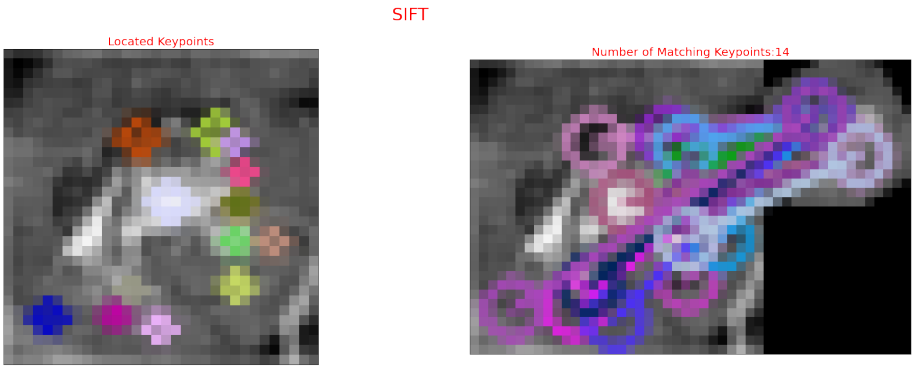

Figure 25: edgeThreshold=3 , keypoints=14

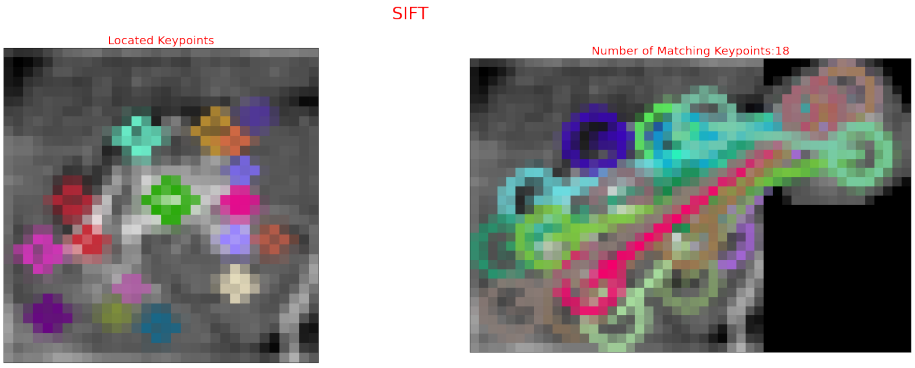

Figure 26: edgeThreshold=9 , keypoints=18

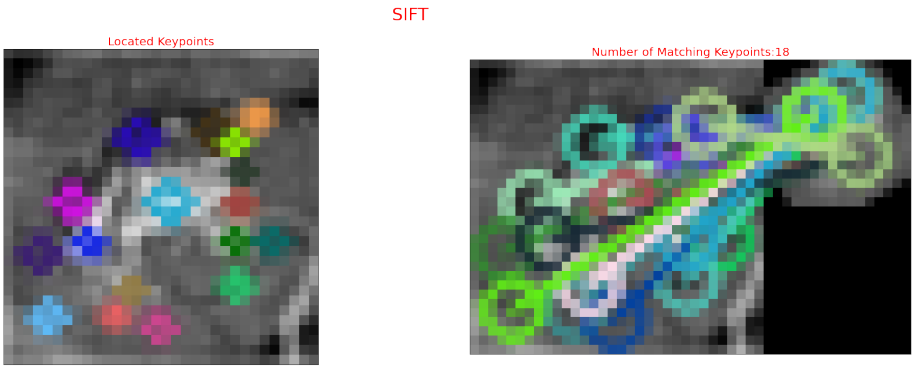

Figure 27: edgeThreshold=15, keypoints=18

Contrast Threshold:
The contrast threshold filters out weak features in low-contrast regions. The purpose for filtering out low-contrast regions is because it is like that features in low-contrast regions are unreliable descriptors. Contrast can provide information on the image such as the sharpness of the change in image intensity at a certain location. The larger this threshold, the less features are detected. This is shown in the figures below. All other parameters are held constant.

Figure 28: contrastThreshold=0.02 , keypoints=20

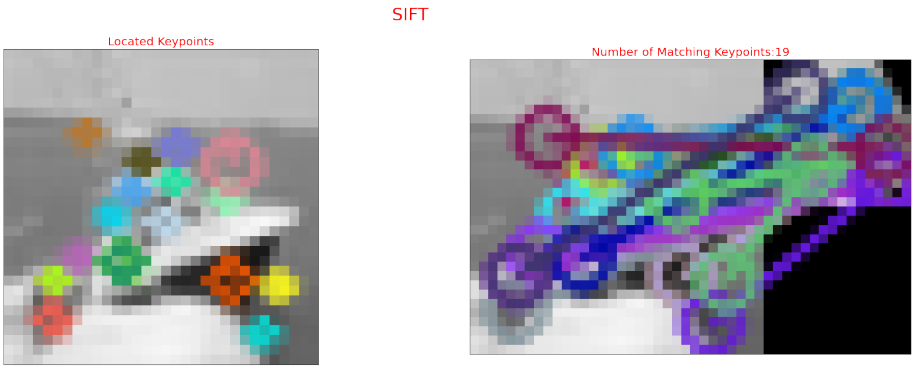

Figure 29: edgeThreshold=0.04 , keypoints=19

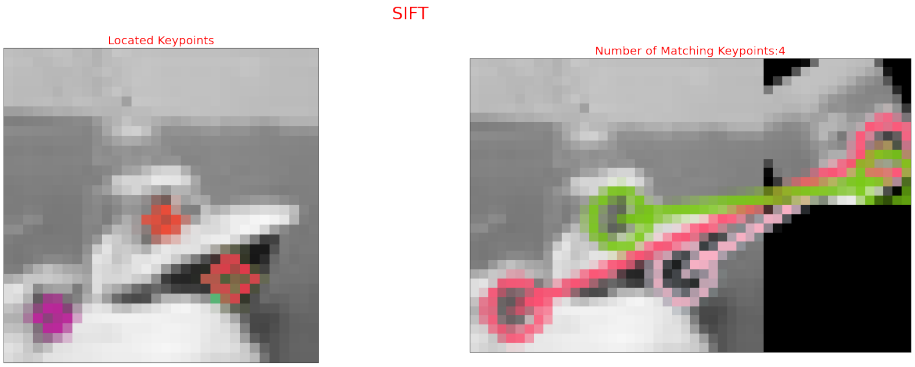

Figure 30: edgeThreshold=0.15, keypoints=4

In choosing parameters for a SIFT detector I would choose parameters that increases the number of keypoints detected since the images I have are small (low resolution). I would choose a smaller value of sigma. This will ensure that smaller "blobs" can be detected. I will also want to consider edges more heavily since it is likely that edges in these images have meaningful features. For the contrast threshold, I would choose a lower value threshold. A lower threshold will allow me to detect more features. For my SIFT function I used the following parameters: sigma=1, edgeThreshold=10 and contrastThreshold=0.02 .

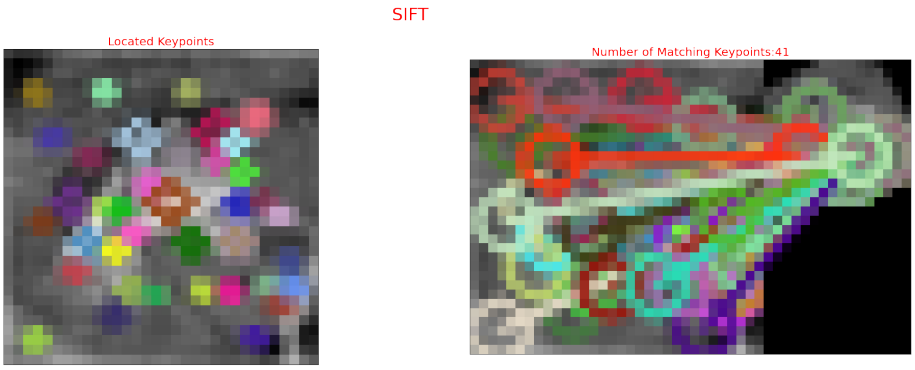

Figure 31: Frog, keypoints=41

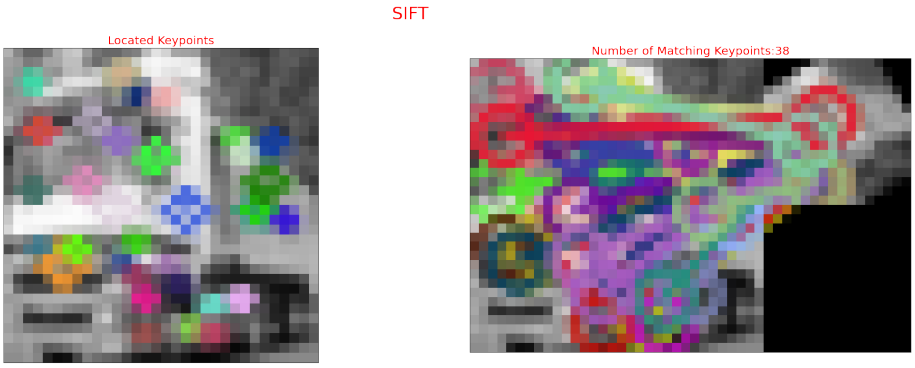

Figure 32: Truck, keypoints=38

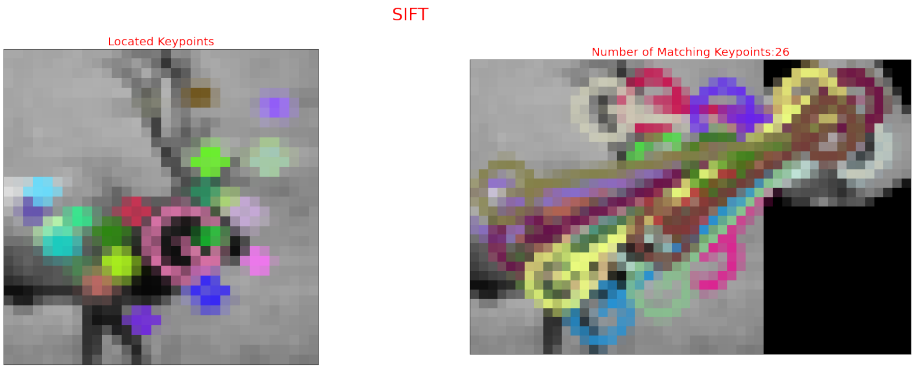

Figure 33: Deer, keypoints=26

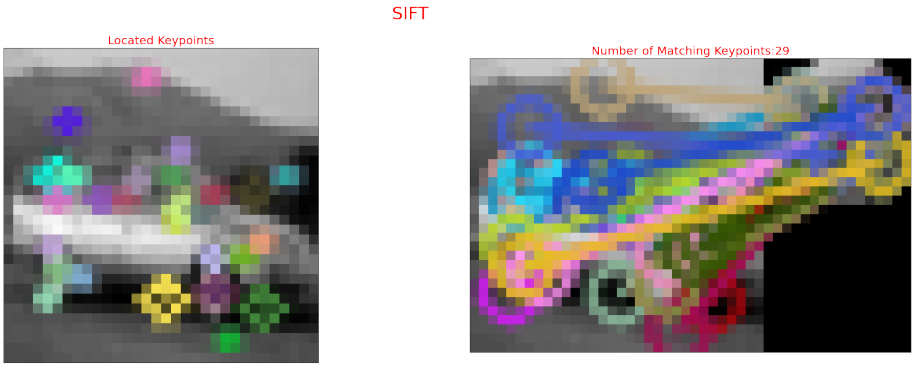

Figure 34: Car, keypoints=29

Figure 35: Bird, keypoints=12

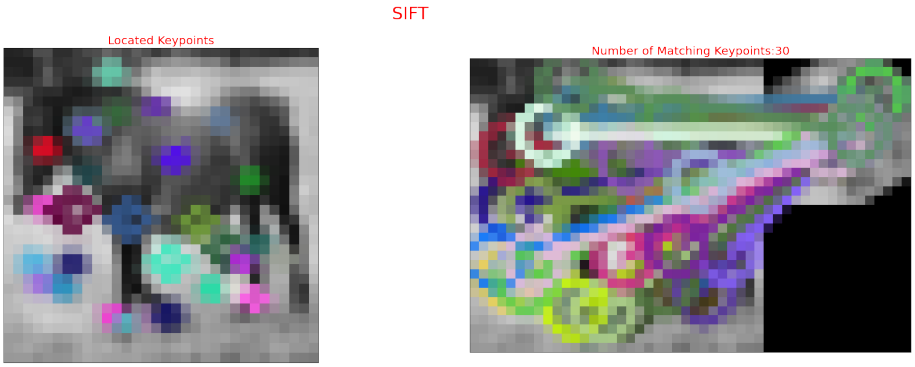

Figure 36: Horse, keypoints=30

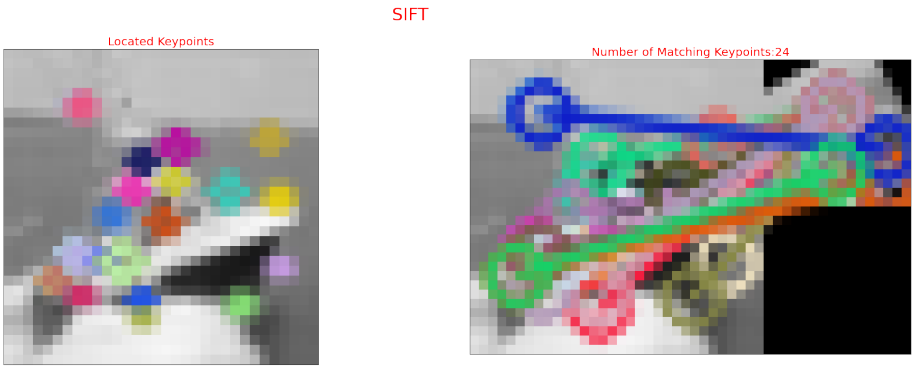

Figure 37: Boat, keypoints=24

Figure 38: Cat, keypoints=16

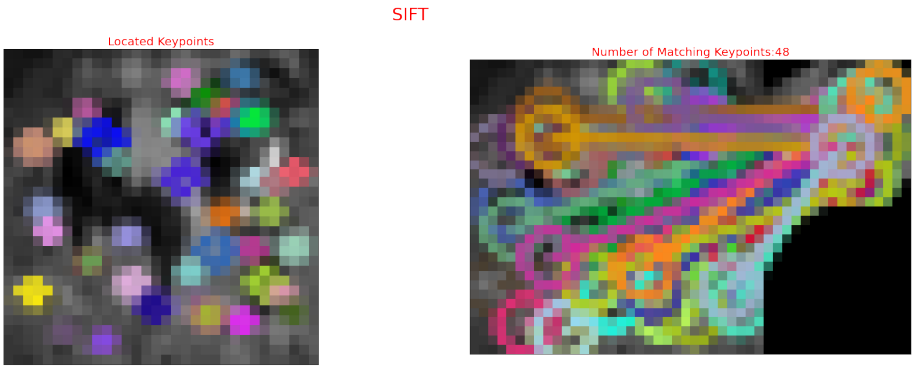

Figure 39: Dog, keypoints=48

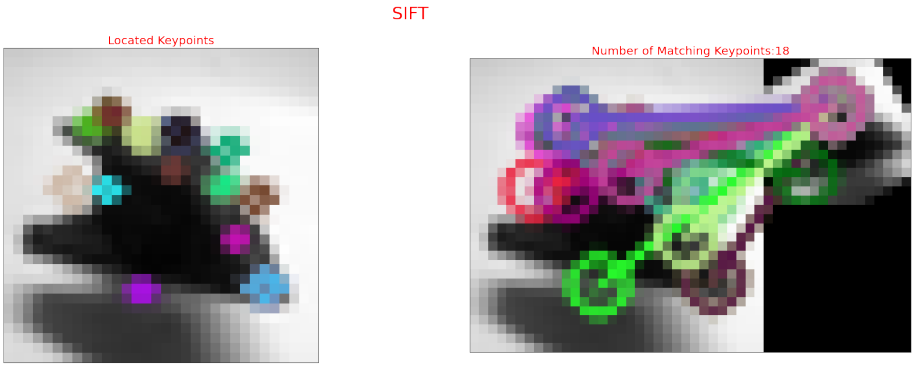

Figure 40: Plane, keypoints=18