Homework 5 Writeup

Code Implementation and Details

Feature Extraction

```python
def feature_extraction(img, feature):
    """
    This function computes defined feature (HoG, SIFT) descriptors of the target image.

    :param img: a height x width x channels matrix,
    :param feature: name of image feature representation.

    :return: a number of grid points x feature_size matrix.
    """

    if feature == 'HoG':
        # HoG parameters
        win_size = (32, 32)
        block_size = (32, 32)
        block_stride = (16, 16)
        cell_size = (16, 16)
        nbins = 9
        deriv_aperture = 1
        win_sigma = 4
        histogram_norm_type = 0
        l2_hys_threshold = 2.0000000000000001e-01
        gamma_correction = 0
        nlevels = 64

        hog = cv2.HOGDescriptor(win_size, block_size, block_stride, cell_size, nbins, deriv_aperture, win_sigma, histogram_norm_type, l2_hys_threshold, gamma_correction, nlevels)
        hog_features = hog.compute(img)

        return hog_features.reshape(-1, 36)
    elif feature == 'SIFT':
        sift = cv2.SIFT_create()
        step_size = 20
        keypoints = [cv2.KeyPoint(x, y, step_size) for y in range(0, img.shape[0], step_size) for x in range(0, img.shape[1], step_size)]
        keypoints, descriptors = sift.compute(img, keypoints)


        if descriptors is None:
            return np.zeros((0, 128), dtype=np.float32)
        return descriptors
```

- **HoG Descriptor** For HoG Descriptor, I simply used cv2.HoGDescriptor function to obtain HoG feature points. Then I reshaped the output into -1*36 size, which automatically makes the row number.

- **SIFT Descriptor** For SIFT Descriptor, first I devided the image into 20*20 size grids, and for each grid made keypoints with cv2.Keypoint function. Then for each keypoints I computed the SIFT descriptor. If there are no descriptors found, the code gives zero array for output.

Principal Component Analysis

```python
def get_features_from_pca(feat_num, feature):

    vocab = np.load(f'vocab_{feature}.npy')

    vocab_mean = np.mean(vocab, axis=0)
    vocab_centered = vocab - vocab_mean

    covariance_matrix = np.cov(vocab_centered, rowvar=False)

    eigen_values, eigen_vectors = np.linalg.eigh(covariance_matrix)

    idx = np.argsort(eigen_values)[::-1]
    principal_components = eigen_vectors[:, idx[:feat_num]]

    reduced_vocab = np.dot(vocab_centered, principal_components)

    return reduced_vocab
```

- First, from the loaded vocabulary I centered it by extracting the mean.
- Then, with the centered vocabulary I calculated the covariance matrix, and performed eigenvalue decompostition to obtain eigenvalues and vectors.
- I sorted the eigenvalues from largest one to the smallest one, and then chose the eigenvectors corresponding to largest feat_num indices.
- Finally, I obtained the reduced vocabulary by making a dot product with the centered vocabulary and selected eigenvectors.

Bag of Words

```python
def get_bags_of_words(image_paths, feature):
    vocab = np.load(f'vocab_{feature}.npy')

    vocab_size = vocab.shape[0]

    bags_of_words = np.zeros((len(image_paths), vocab_size))

    for i, path in enumerate(image_paths):
        img = cv2.imread(path)
        features = feature_extraction(img, feature)

        distances = pdist(features, vocab)
        closest_vocab_indices = np.argmin(distances, axis=1)

        for idx in closest_vocab_indices:
            bags_of_words[i, idx] += 1

        bags_of_words[i, :] /= linalg.norm(bags_of_words[i, :])

    return bags_of_words
```

In the beginning...

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum. See Equation 1.

$$a = b + c \tag{1}$$

Interesting Implementation Detail

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

My code snippet highlights an interesting point.

```
import numpy as np
one = 1
two = one + one
if two == 2:
    # This is comment
    print('This computer is not broken.')
else:
    print('This computer is broken.')
```

A Result

1. Result 1 was a total failure, because...

2. Result 2 (Figure 1, left) was surprising, because...
3. Result 3 (Figure 1, right) blew my socks off, because...

Figure 1: *Left:* My result was spectacular. *Right:* Curious.

My results are summarized in Table 1.

Condition	Time (seconds)
Test 1	1
Test 2	1000

Table 1: Stunning revelation about the efficiency of my code.