Whenever you describe something, always explain *how* it's being done.

Question 1, 4p

Flo is the artist who created a giant inflatable duck (see photo). The duck is typically placed in random harbours. For security reasons, we would like to track the duck but we're dealing with a highly dynamic environment due to the weather conditions and objects such as boats passing by. Background subtraction using a color model of the background is therefore not very suitable. Instead, we can build a color model of the duck and use that to find the foreground. The fuck is yellow with an orange beak. Due to the lighting and shadow conditions, the shades of yellow and orange vary to some extent. In our case, with "foreground" we refer to the duck shape.

We take 20 "training" images with the duck. We cluster the color values of all pixels in these images using k-means with k=15 clusters. Once we have determined the cluster centers, we use these as the bins of our histogram. So a color value "belongs" to the bin corresponding to the closest cluster center. We then take only the pixels of the duck in all "training" images, find the closest cluster center and "vote" for that bin in our histogram. We then normalize the histogram.

a) Explain how we can decide, for each pixel, whether it is foreground (duck) or background, given the duck histogram. This step will produce a binary foreground mask. Use pseudo-code or test to describe your pipeline. Clearly explain and initialize your data structures. If there are parameters in your approach, mention these explicitly and explain what they do. From your text, I should be able to reproduce your pipeline. No need to do post-processing, we do this in subquestion b)

Answer key points: Loop over all pixels (1p), distance to GMM (2p), decision for foreground/background (1p)

b) After the process in subquestion a), we end up with a binary image that hopefully contains a duck-like foreground shape but probably also contains some holes in this shape, and spurious pixels that are erroneously attributed to be part of the duck. To this end, we will introduce spatial information. Using clustering on the binary foreground image, we plan to figure out which pixels are likely to be erroneously classified as foreground. Describe how to do this, using pseudo-code or text. Again, clearly explain and initialize data structures and parameters. We're ignoring the duck's eyes so you don't have to pay attention whether these are properly segmented. A solution without any parameters is preferred.

Answer key points: Clustering applied, distance to center used, non-parametric (all 1p)

Question 2, 4p

Which statements about SIFT are true?

The orientation of a keypoint is calculated from the gradients of a small area around the keypoint (correct) (2p)
A SIFT descriptor encodes the gradient directions around a keypoint (correct) (2p)
SIFT descriptors are normalized for scale and rotation before matching (incorrect) (-1.33p)
After the keypoint selection stage, keypoints are approximately evenly distributed over the image (incorrect) (-1.33p)
The scale of two SIFT keypoints is determined as the inverse distance between them (incorrect) (-1.33p)

Question 3, 4p

We have calibrated a webcam using a chessboard. We webcam's position and orientation do not change, i.e. the camera doesn't move. In the online phase, we're now reading each frame from the webcam, similar to Assignment 1, and apply the *findChessboardCorners* function from OpenCV. We're moving the chessboard around, which can cause motion blur. The *findChessboardCorners* therefore doesn't always find the corners. To still estimate the position of the corners, we calculate dense optical flow using the Lukas-Kanade algorithm with image pyramids (*calcOpticalFlowPyrLK*), which results in a dense optical flow map. You may assume that the chessboard is never occluded and is always fully in view. You mal also assume that previous frames are available and that there are never more than five consecutive frames without a detection from *findChesscoardCorners*.

Describe an algorithm that can be used to estimate the location of the chessboard corners, using either *findChessboardCorners* or *calcOpticalFlowPyrLK*. Use pseudo-code or text to describe your pipeline. Clearly explain and initialize your data structures. If there are parameters in your approach, mention these explicitly and explain what they do. From your text, I should be able to reproduce your pipeline.

Answer key points: Selection chessboard/OF (1p), use of optical flow (1p), filtering (2p)

Question 4, 4p

Which of the following statements about bag-of-words are true?

Codewords are found by clustering a representative set of local features (correct, 1.33p)
Bag-of-words are calculated over sets with an arbitrary number of local features (correct, 1.33p)
With the same codebook, the bag-of-word vector length of two images is equal (correct, 1.33p)
Spatial information is retained when using bag-of-words (incorrect, -2p)

Question 5, 4p

We want to detect faces in an image. To this end, we use the Selective Search algorithm to iteratively merge clusters of pixels. Each pixel initially forms a cluster. In each iteration of the algorithm, clusters are merged based on color and texture cues. Consequently, the number of clusters decreases in every iteration of the process. You may assume that there is a function *mergeClusters* that takes care of this process. You may also assume that there is a function *getClusterSize* available that, for a given cluster, returns the height and width of that cluster in the images. Similarly, *getClusterBoundingBox* returns the bounding box of the cluster.

We also have a face template ***HOG_face*** available in the form of a HOG descriptor. The standard parameters that lead to a feature vector of dimensionality of 640 apply. We are only interested in faces with a width between 50-80 pixels and a height between 60-140 pixels. You may use a function *calculateHOG* that produces a HOG descriptor given an input image and a bounding box.

Describe a pipeline that uses Selective Search and HOG template matching to detect faces of the right size. The output of your algorithm should be a list of bounding boxes. You can use the functions described above but you explicitly need to explain how the matching is performed. Clearly explain and initialize your data structures. If there are parameters in your approach, mention these explicitly and explain what they do. From your text, I should be able to reproduce your pipeline.

Answer key points: Proper use of Selective Search (2p), proper matching and thresholding (2p)

Question 6, 7p

As input for an automated apple picker, we develop an apple detection algorithm. Given a camera image, we use a sliding window approach to detect apples. At each location offset, we extract a window with a fixed size (100x100). We use this window as input for a trained CNN with inputs of size 100x100x3 that has two neurons in the output layer: one for “apple”, one for “other”. The activation of the output later is Softmax. You may assume that the apples in the images are not overlapping and each apple is between 70x70 and 110x110 in size. After processing an image, we have a (large) list of bounding boxes with an associated Softmax score for the neuron that corresponds to the “apple” class. We do not filter this list yet. Also, we do not vary the size of the detection window in our approach, so we only traverse the image once.

Initially, the stride in the sliding window approach is 1. This means we could have duplicate detections at a slightly different location within the image. We will use non-maximum suppression. This algorithm uses an IoU measure for the overlap between two bounding boxes. What would be a suitable threshold for IoU so that, given a bounding box that is perfectly centered on the target, we don't miss any apple. Explain your answer.

Answer key points: Correct overlap (1p), explanation (1p). IoU threshold = 4900/10000 = 0.49

Explain how the non-maximum suppression algorithm works, either in text or using pseudo-code. If you introduce parameters or data structures, mention these explicitly and explain what they do. From your text, I should be able to reproduce your pipeline.

Answer key points: Sorting (1p), use of threshold (1p), processing entries (1p)

We decide to increase the stride of the sliding window approach to 20. We still apply non-maximum suppression. With the same IoU threshold, explain what will happen to 1) the number of false positives and 2) the number of false negatives

Answer key points: 1) fewer total steps, so false positives increase. 2) increases, since we might not capture an entire apple

Question 7, 4p

We're trying to train a CNN but our training accuracy doesn't improve at all, while the loss also doesn't decrease at all. So we're not training. What could be possible causes>

Learning rate is fixed to 0 (correct, 4p)
When using SGD with momentum as the optimizer, the momentum factor is 0 (incorrect, -1.33p)
The output labels of the training samples are shuffled (incorrect, -1.33p)
The activation function of the output layer is ReLU (incorrect, -1.33p)

Question 8, 9p

We are to classify handwritten digits using a CNN. The input is a 28x28x3 RGB image of a digit. There are 10 outputs, corresponding to digits 0-9. Your network should, realistically, be able to learn the relevant patterns with expected variations in the input. Design a CNN according to the following restrictions

1. You can only use these layers: conv, fully connected, flatten, input, output. (Global)pooing layers or any other type not mentioned here are not allowed.
2. Your network has to have at laest one fully connected layer one input layer and one output layer
3. The maximum number of layers is 7 layers. All layers count towards that sum
4. The total number of inputs to a single neuron in the output layer cannot exceed 50
5. All activation functions are ReLU except for the output, which is softmax
6. Number of filters in the first conv layer is 8 and doubles for every next conv layer
7. Input to a conv layer can only be one dimensional
8. Spatial size of the output of the last conv layer cannot exceed 5x5

Provide the sequence of layers. For each conv layer, provide the relevant hyper-parameters; filter size, stride, padding. for each fully connected layer, give the number of neurons

Answer key points: Sequence adhering to criteria (3p), hyperparameters conv correct (1p), hyperparams fully connected correct (1p)

Provide for each layer 1) the number of params and 2) the size of the output activation volume. Your calculations should match the choices you made in subqestion a). Provide the calculation but you don't have to motivate your answer.

Answer key points: Number of params (2p), output volume (2p)

Question 9, 4p

Which of the following measures will combat overfitting when training a deep CNN model?

Reduce number of params in the model (correct, 1.33p)
Add skip connections to the conv layers (correct, 1.33p)
Use pre-training on images of a similar image domain (correct, 1.33p)
Replace ReLU activations with softmax (incorrect, -2p)
Increase number of conv layers (incorrect, -2p)