Seminar 4 - Lectures 4a, 4b and 5a
Computer Vision 1, Master AI, 2025

Arnoud Visser, Martin Oswald, Qi Bi, Roan van Blanken

1 Exercise 1: Image Stitching

Q.1.a What is image stitching?

Answer: Image stitching attempts to combine two or more overlapping images into one larger, seamless image. It uses transformations, such as homography, to align the images and blend the overlapping regions, minimizing visible seams or distortions in the final panorama.

Q.1.b Describe the basic steps to stitch two images with the same optical center. Try to list some typical algorithms for each step.

Answer:

1. Compute the transformation matrix between second image and the first one.
   (a) Detect key points. (Typical algorithms: Harris corner, SIFT, SURF, ORB)
   (b) Find matches, i.e. compute the distance between every descriptor in one image and every descriptor in the other image. (Typical algorithms: Euclidean distance, cosine distance)
   (c) Compute transformation. Since the optical centers of two images are the same, we directly compute the homography matrix.
   (d) (RANSAC)
2. Planar mapping. (Typical algorithms: Planar projection.)
3. Blend the two images. (Typical algorithms: Feathering, Pyramid blending, alpha blending)

Q.1.c Describe the basic steps of RANSAC.

Answer:

1. Randomly sample the minimum number of points required to fit the model
2. Solve for model parameters using the minimum set of samples
3. Score by the fraction of inliers within a preset threshold of the model
4. Repeat 1-3 until the best model is found with high confidence

Q.1.d Assume we are using RANSAC to fit a plane in 3D space. Given a set of ten inlier points belonging to a plane, denoted as P_1, P_2, ..., P_{10}, and an additional three outlier points, denoted as O_1, O_2, and O_3. Also, assume that any three points are not collinear. How many points should be sampled each time to define a plane using RANSAC?

Answer: Three. In 3D space, three non-collinear points are necessary and sufficient to determine a plane.

Q.1.e Continuing from the previous question, suppose we repeatedly sample points using RANSAC until we successfully find a plane that contains all ten inlier points $(P_1, P_2, ..., P_{10})$. In the worst-case scenario, how many times do we need to sample to ensure this outcome? (Hint: Use the binomial coefficient formula $C_n^k = \frac{n!}{k!(n-k)!}$ to calculate the number of ways to choose k points from n total points.)

Answer: We sample 3 points from the total of 13 points each time. In the worst-case scenario, every sample includes at least one outlier until the correct plane is found. The binomial coefficient is used here to calculate the different combinations of choosing outliers and inliers in each sample. The worst-case number of samples can be calculated as follows:

$$C_3^1 C_{10}^2 + C_3^2 C_{10}^1 + C_3^3 C_{10}^0 + 1 = 3 \times 45 + 3 \times 10 + 1 \times 1 + 1 = 167 \tag{1}$$

Here, $C_n^k = \frac{n!}{k!(n-k)!}$ represents the binomial coefficient, which is used to count the number of ways to select k elements from a set of n elements. The terms C_3^1, C_3^2, and C_3^3 represent the different ways of including 1, 2, or all 3 outliers in the sample, while C_{10}^2, C_{10}^1, and C_{10}^0 represent the selection of the remaining points from the 10 inliers.

Q1.f Assuming the error boundary δ is set to 0 in the third step of the RANSAC algorithm, what is the minimum fraction of points that need to be inliers to correctly define the plane in this scenario?

Answer: In the worst case, the plane defined by the three outliers might still align with exactly two of the ten inlier points. This scenario suggests that at least 6 out of the 13 points must be correctly identified as inliers to avoid fitting a plane incorrectly based on outliers. Therefore, the minimum fraction of points that must be inliers is:

$$\frac{6}{13} \tag{2}$$

This fraction indicates that in the worst-case scenario, RANSAC needs to identify at least 6 inliers out of the total 13 points to ensure the correct plane is defined.

2 Exercise 2: Bag of Visual Words

Q.2.a Describe the basic steps of the Bag of Visual Words model?

Answer:

1. Build a visual dictionary by clustering the descriptors.
2. Sample patches from the image.
3. Extract local features from the patches.
4. Create a histogram of visual words.

Q.2.b Explain how k-Means clustering can be used to obtain a visual vocabulary.

Answer:

1. Collect a large set of image patches from the training dataset.
2. For each patch, extract its feature descriptor (e.g., SIFT, SURF).
3. Apply k-Means clustering to these descriptors, where k represents the number of clusters.
4. Each cluster center, or mean, represents a unique *visual word* in the vocabulary.
5. The value of k determines the size of the visual vocabulary and can be adjusted depending on the complexity of the image dataset.
6. After clustering, use these visual words to describe and categorize images in a more compact and meaningful way.

Q.2.c Briefly summarize the major steps involved in using k-Means clustering to cluster the SIFT features into k words in BoW.

Answer:

1. Choose the number of clusters, k, which will represent the number of visual words.
2. Randomly select k initial cluster centroids from the set of SIFT feature vectors.
3. Assign each SIFT feature vector to the nearest cluster centroid based on distance (e.g., Euclidean distance).
4. Recalculate the centroids by averaging the feature vectors assigned to each cluster.
5. Repeat the assignment and centroid recalculation steps until the cluster centroids stabilize, or a stopping criterion is met.
6. The final cluster centroids represent the k visual words used in the Bag of Visual Words model.

Q.2.d Explain the difference between dense sampling and interest point sampling. Now, consider an image retrieval system with a dataset and 100 query images, evaluated using precision@10. If we use a BoW representation with 10K words and SURF descriptors, why might BoW with dense sampled patches perform better on a 1,000-image dataset, but interest point sampling performs better on a 10M-image dataset?

Answer:

1. **Difference between dense sampling and interest point sampling:**
   - **Dense Sampling:** Extracts patches from a regular grid across the entire image, capturing features from all regions.
   - **Interest Point Sampling:** Detects keypoints in salient regions (e.g., corners) using algorithms like Harris, focusing on specific parts of the image.
2. **Reason for performance differences:**
   For smaller datasets (1,000 images)
   - Better coverage. If the interest point detector misses any relevant regions, dense sampling still covers them. With few hard negatives in the small dataset, the extra encoded generic visual words (e.g., patches from background) don't hurt top-10 much.

   For larger datasets (10M images)
   - Distinctiveness matters for precision. Interest point sampling performs better because it focuses on the most distinctive features, which helps in finding precise matches in a large pool of images.
   - Less noise. Dense patches can flood histograms with common visual words (sky/walls), causing many weak matches across many images, which leads to retrieval candidate explosion and lower precision. This is especially the case when the dataset gets larger as any generic visual words make thousands of images look "a bit similar".

Q.2.e What are the weaknesses of the Bag of Visual Words model?

Answer: The Bag of Visual Words model has several weaknesses. Firstly, it lacks spatial information, as it treats an image as an unordered collection of visual words without considering their positions or arrangements. This results in a loss of structural details and context. Secondly, the model has no semantic understanding of the features; it only counts the frequency of visual words, ignoring their relationships and meaning. Lastly, the performance of the model is sensitive to the chosen vocabulary size. If the number of visual words (k) is too small, important details may be lost, while a too-large vocabulary can lead to overfitting and increased computational cost.

3 Exercise 3: Image Retrieval

Q.3.a Assume there is an image retrieval system. We find 4 true positive (TP) samples and 96 true negative (TN) samples. There is 1 false positive (FP) sample and 2 false negative (FN) samples. Calculate the recall, precision, and accuracy (in percent). Based on these outcomes, explain why accuracy is not a good metric.

Answer: The precision, recall, and accuracy for the given image retrieval system can be calculated as follows:

$$\text{precision} = \frac{TP}{TP+FP} = \frac{4}{4+1} = \frac{4}{5} = 0.80\,(\text{or } 80\%) \tag{3}$$

$$\text{recall} = \frac{TP}{TP+FN} = \frac{4}{4+2} = \frac{4}{6} \approx 0.67\,(\text{or } 67\%) \tag{4}$$

$$\text{accuracy} = \frac{TP+TN}{TP+TN+FP+FN} = \frac{4+96}{4+96+1+2} = \frac{100}{103} \approx 0.97\,(\text{or } 97\%) \tag{5}$$

Accuracy measures the proportion of correct assignments, considering both relevant and non-relevant images. However, in image retrieval tasks, most images in the dataset are often not relevant. Therefore, a system that simply returns "no image" for most queries can achieve a high accuracy but fails to serve as an effective retrieval system. Precision and recall, on the other hand, focus on the retrieval of relevant images, providing a more meaningful evaluation.

Q.3.b Compute the Average Precision (in percent) for the following relevance ranking: $[R, R, N, N, R, R, N, N]$, where R denotes relevant and N denotes not relevant.

Answer:

$$\text{AP} = \frac{1}{\sum_{k=1}^{K} \text{rel}_k} \sum_{k=1}^{K} P@k \cdot \text{rel}_k, \tag{6}$$

$$= \frac{1}{4}\left(\frac{1}{1} + \frac{2}{2} + \frac{3}{5} + \frac{4}{6}\right) \approx 0.82\,(\text{or } 82\%), \tag{7}$$

where K is the total number of relevant documents, $P@k$ is the precision at rank k, and rel_k indicates whether the document at rank k is relevant (1) or not (0).

Q.3.c For retrieval, each image is described with 1000 interest points, each interest point is 64-dimensional. How many computations are required when finding matches between 2 images? Now, assume comparing 1M interest points takes 0.8 seconds. How long does it take to compare an image with a dataset containing 2M images (ignoring other factors such as sorting)?

Answer: Here number of computations refer to the elementary operations of multiply(*) and add (+).

1. **Computations required for matching two images:**
   - Each image has 1000 interest points.
   - Each interest point in image 1 is compared with every interest point in image 2.
   - Each comparison involves n computations (depending on the distance metric used, e.g., if using cosine similarity, with precomputed vector magnitudes, there are 64 multiply and 63 add operations).
   - Total computations = $1000 \times 1000 \times n$.
2. **Time required for comparing with a dataset:**
   - Comparing 1 million interest points takes 0.8 seconds.
   - Comparing two images involves 1 million interest point comparisons (1000×1000).
   - Thus, comparing two images takes 0.8 seconds.
   - To compare one image with a dataset of 2 million images: $2 \times 0.8 = 1.6$ million seconds.

Q.3.d After retrieving results with BoW, we use geometrical verification to rerank the top 200 images. However, our evaluation shows no difference in precision and recall measured at $k = 200$. Why? How might you observe a difference in precision and recall?

Answer:

1. **Why no difference at $k = 200$:**
   - Precision and recall at $k = 200$ are evaluated for the entire set of 200 images.
   - Reordering these 200 images does not change the overall precision and recall at this fixed cutoff point.
2. **How to observe differences:**
   - Re-ranking may show improvements in precision and recall at smaller values of k, such as $k = 10$, $k = 50$, or $k = 100$.
   - By evaluating precision and recall at these smaller cutoffs, the impact of reordering becomes more noticeable.

4 Exercise 4: Image Classification & Object Detection

Q.4.a What is the difference between image classification, image localization, and object detection?

Answer: Image classification identifies the category an image belongs to, helping us determine the overall content of the image. Image localization, on the other hand, goes a step further by specifying the location of a single object within the image. Finally, object detection is an extension of localization that identifies and specifies the locations of multiple objects in an image.

Q.4.b For binary linear classification, assume Bag of Words (BoW) is your feature extractor and Support Vector Machine (SVM) is your classifier. How is a "dog" vs "non-dog" classifier trained?

Answer: To train a "dog" vs "non-dog" classifier using BoW and SVM, follow these steps:

1. Collect a large, annotated dataset of images containing "dog" and "non-dog" classes.
2. Decide the test set, for train/val sets, choose to decide it once or run k-fold cross validation to have the train fold and val fold.
3. Compute the BoW visual vocabulary using only the current train set, not with val set, as we don't want to bias the parameter selection on the val set.
4. Generate the BoW representation for all images in the training, validation, and test sets.
5. Use the training set to train the SVM classifier and the validation set to select hyperparameters (e.g., number of words in the vocabulary, regularization parameters).
6. Finally, evaluate the performance of the trained classifier on the test set to measure its performance.

Q.4.c In an object detection task, suppose there are 5 classes and 9 possible locations in an image. We use three aspect ratios (1:1, 3:4, 4:3) and four different scales (0.5, 1, 1.5, 2). How many box evaluations are needed for this image in a multi-class object detection problem?

Answer: To calculate the total number of box evaluations, use the formula:

$$\text{Total evaluations} = \#\text{locations} \times \#\text{aspect ratios} \times \#\text{scales} \times \#\text{classes}. \tag{8}$$

Substitute the given values:

$$\text{Total evaluations} = 9 \times 3 \times 4 \times 5 = 540. \tag{9}$$

Thus, 540 box evaluations are needed if using binary classifier; if it's a multi-class classifier, we run a single inference for every box, leading to $9 \times 3 \times 4 \times 1$.

Q.4.d What is the key idea of Selective Search, and how does it increase variations in object detection?

Answer: Selective Search is a technique used to find a small set of class-agnostic object bounding boxes. The key idea is to use hierarchical clustering to group similar regions in an image, considering different color spaces and other features. By employing various grouping strategies, Selective Search generates region proposals that capture objects at multiple scales, sizes, and locations, thereby increasing the variations and improving the robustness of the object detection process.

Q.4.e What are the benefits of using Selective Search in object detection pipelines?

Answer: Selective Search offers several benefits in object detection:

1. It captures objects at different scales, addressing the fact that objects in images can vary in size and orientation.
2. It diversifies the grouping of regions according to different metrics, enhancing the likelihood of identifying the objects.
3. It is computationally efficient compared to an exhaustive search over all possible bounding boxes, speeding up the object detection process.

5 Exercise 5: Sliding Window Approach

Q.5.a What is the basic pipeline for window-based object detection? What are the weaknesses of window-based object detection?

Answer:
The basic pipeline for window-based object detection involves the following steps:

1. Inspect every window and generate candidates.
2. Extract features from each window.
3. Classify each window. Accept the window if the classification score exceeds a threshold.
4. Perform post-processing to clean up overlapping detections.

The weaknesses of window-based object detection include:

1. High computational complexity due to inspecting a large number of windows.
2. Not all objects are "box"-shaped, leading to less accurate detections.
3. Non-differentiable, making it harder to integrate into modern deep learning pipelines.

Q.5.b For an image of size 512×512, how many windows are required to detect objects for 16 different orientations and 4 scales?

Answer: To detect objects for 16 orientations and 4 scales:

$$\text{Total windows} = 512 \times 512 \times 16 \times 4 = 16,777,216. \tag{10}$$

Thus, 16,777,216 windows are required for this object detection task.

Q.5.c Assume a non-linear SVM on your computer requires about 0.02 seconds to classify each window. How many hours does it take to detect objects in $6,000$ images? How can you reduce the number of bounding boxes for detection?

Answer: The total time required for detecting objects in 6,000 images can be calculated as:

$$\text{Total time} = 16,777,216 \times 0.02 \times 6000/3600 = 559,241\,\text{hours}. \tag{11}$$

To reduce the number of bounding boxes for detection, you can:

1. Use strides to jump positions during the sliding window process.
2. Apply segmentation techniques to focus on regions of interest.

Q.5.d Consider the application of a sliding window strategy in convolutional neural networks for a five-class object detection task on a 28×28 image. Assume the sliding window size is 14×14 with a stride of 2. No padding is used, meaning the convolutional kernels cannot exceed the image. How many outcomes are generated after processing by the sliding window?

Answer: First, we need to determine how many times the window slides across each row and column:

$$\text{Number of slides} = \frac{28 - 14}{2} + 1 = 8. \tag{12}$$

The total number of outcomes generated is:

$$\text{Total outcomes} = 8 \times 8 \times 5 = 320. \tag{13}$$

Q.5.e In the case of a five-class object detection task on a 28×28 image, suppose the sliding window size is 14×14 and the stride is 2. Unlike in the previous, where no padding was used, now assume a padding size of 2. How many outcomes are generated after processing by the sliding window?

Answer: In Q.5.d, we used a sliding window of size 14×14 with a stride of 2 on a 28×28 image without padding, resulting in 320 outcomes. Now, with a padding size of 2, the image extends by 2 pixels on each side.

The number of times the window slides across each row and column is:

$$\text{Number of slides} = \frac{28 - 14 + 2 \times 2}{2} + 1 = 10. \tag{14}$$

Given there are 5 classes, the total number of outcomes generated is:

$$\text{Total outcomes} = 10 \times 10 \times 5 = 500. \tag{15}$$

Thus, with the added padding of 2 pixels, the number of outcomes increases from 320 (without padding) to 500.