Project Descriptions

- Projects need to be done individually and cannot be done in groups.
- There are four project proposals. You can choose freely which you want to work on.
- Each project consists of three parts: A mandatory part, theoretical questions, and an advanced part.
- In order to pass the project with a grade of 3, you need to complete the mandatory part.
- Completing the mandatory part and either the theoretical questions or the advanced part will allow you to pass the project with a grade of 4.
- Completing the mandatory part and both the theoretical questions and the advanced part will allow you to pass the project with a grade of 5.
- The theoretical parts consist of three questions, each of which is worth 1 point. You need 2 points to pass the theoretical part. Your answers will be graded on a continuous scale, i.e., we try to reward partially correct answers.
- Use the MATLAB lab template provided for the labs to hand in your code.
- In addition to the MATLAB code, you will need to write a short report describing your project and providing experimental results. If you submit answers to the theoretical questions, include your answers in the report as well. Please use the paper template from 3DV 2019 conference. Latex and word templates are available here. Submit your report as a PDF file through Canvas.
- Submit all files through Canvas.

1 Project I: Digit Recognition

The goal of this project is to develop a system for digit recognition in images. The system will be based on multi-scale processing with fully convolutional neural networks. Images for evaluation are provided on Canvas.

1.1 Mandatory Part

Design a fully convolutional neural network for digit classification and train it on the MNIST dataset. MNIST provides images of the digits 0 to 9 of size 28×28, together with the label for which digit is shown in a given image. Your fully convolutional neutral should be trained to produce the correct label for the center of the output of your last layer. MNIST provides training and validation data that you should use for training your neural network. Try out different architectures until you achieve at least 97% accuracy on the test set. Since MNIST is a relatively easy dataset, a few layers should be sufficient for this.

Once you have trained your network, you can apply it on one of the evaluation images provided on Canvas. Use non-maximum suppression and thresholding to obtain your digit predictions and write out the detected digits (no particular order required) on MATLAB's output line. Note that you will need to detect digits at multiple scales.

You can find instructions on how to use the MNIST dataset inside MATLAB here.

What to include in the report. Your report should consist of the following:

- A short abstract describing the problem behind the project and outlining your solution.
- A "Method" section describing your approach: What network architecture are you using? How are you handling multi-scale processing? This section should be about one page long.
- A "Experimental Evaluation" section describing the experiments you perform with your approach as well as the results of these experiments:
  - Describe the parameters you used for training, e.g., the learning rate, whether you used momentum, etc.
  - Plot training and validation loss over the number of epochs and explain how you decided when to stop the training process.
  - Measure the accuracy your classifier achieves on the test set of the MNIST dataset.
  - If you experimented with different neural network architectures, also include their results on the MNIST test set.
  - Visualize the evaluation images provided on Canvas and mark which digits were correctly detected by your approach. Also mark false positive detections (e.g., using a different color).

1.2 Theoretical Part

Answer the following questions in your report. Make sure to use your own words and justify your answers.

(a) Fully convolutional neural networks vs. neural networks with fully connected layers. Explain the purpose of fully connected layers in convolutional neural networks. In which scenarios would you apply a fully convolutional neural network instead of a neural network with a fully connected layer?

(b) Converting a fully connected to a fully convolutional layer. Explain how to convert a fully connected layer to a fully convolutional layer such that the fully convolutional layer provides the same functionality.

(c) Scale-space construction. Explain how a scale space that allows the detection of objects at multiple sizes is constructed: Which operations are involved? How is the scale space constructed using these operations? How can a scale space be constructed efficiently?

1.3 Advanced Part

The goal of the advanced part is to extend your initial system from the mandatory part to also handle rotated variants of input images. You should try at least two approaches:

- Evaluate your network from the mandatory part on rotated versions of the evaluation images provided on Canvas and merge the corresponding detections. You might need to adapt the non-maximum suppression part to take the rotated versions into account during merging.
- Use data augmentation to train a variant of your fully convolutional neural network that can be applied on rotated digits.

What to include in the report. Extend your report to include the following:

- Extend the "Method" section by a subsection describing the approaches you implemented to handle rotated versions of the digits.
- Extend the "Experimental Evaluation" section with the results you obtained on the evaluation images showing rotated digits. Present your results by visualizing the true and false positive detections (in different colors).

2 Project II: RANSAC

The goal of this project is to extend your RANSAC implementation from Lab 3 to improve both the quality of the returned solution as well as the run-time.

2.1 Mandatory Part

Start with your implementation from Lab 3, where you implemented a RANSAC algorithm for estimating an affine transformation, as well as the function `affine_test_case(outlier_rate)` from the same lab. Modify your function `affine_test_case` as to be able to add Gaussian noise to the measurements in your affine test cases.[1] In addition, write a function that can be used to evaluate the quality of an estimated affine transformation. This function should compute the average residual length after the transformation for all true inliers (i.e., the inliers generated by `affine_test_case`).
Next, you will extend your RANSAC implementation by two techniques:

- Given a transformation M estimated by the minimal solver, the $T_{d,d}$ test randomly selects d correspondences. M is only evaluated on all correspondences if all of these d matches are inliers to M. Otherwise, M is discarded and the next iteration of RANSAC begins. Notice that this is equivalent to drawing a random sample of size $3 + d$ in each iteration of RANSAC, i.e., you will need to adjust the number of required iterations accordingly.
- Given a new best model $M_{\text{new best}}$ found by RANSAC, together with its set I of inliers, *local optimization* efficiently tries to improve this model. First, the best model M_{best} found using the minimal solver is updated to $M_{\text{new best}}$. Next, *local optimization* introduces an inner optimization loop. For each of its $k = 10$ inner iterations, it draws a non-minimal sample S of size $\min(I, 12)$ from the inliers I of $M_{\text{new best}}$. Next, all correspondences in the sample S are used to obtain a new transformation estimate (use the function `least_squares_affine` from Lab 3 for this). Ultimately, RANSAC returns the transformation with the largest number of inliers among all models found by local optimization and M_{best}.
Note that the models found by local optimization are not used to replace M_{best}. However, it is used to update the current estimate of the inlier ratio.

Use your modified version of `affine_test_case(outlier_rate)` to generate examples for experimental verification for the report.

What to include in the report. Your report should consist of the following:

- A short abstract describing the problem behind the project and outlining your solution.
- A "Method" section describing your approach: Provide pseudo code for your extended version of RANSAC (including the $T_{d,d}$ test and *local optimization*) and explain the algorithm in the text. Take special care to introduce all variables you use. This section should be about one page long.
- A "Experimental Evaluation" section describing the experiments you perform with your approach as well as the results of these experiments:
  - For the $T_{d,d}$ test (without local optimization), create the following two figures:

[1]To make it easier to add noise, adjust your function to generate points in an image of size 640×480.

1. The y-axis of the first plot corresponds to the number of RANSAC iterations taken while the x-axis corresponds to an increasing outlier ratio (from 0.0 to 0.9).
2. As 1), but the y-axis of the first plot corresponds to the run-time of RANSAC.

For each figure, run RANSAC with the $T_{d,d}$ test for $d = 0$, $d = 1$, $d = 2$, $d = 3$, i.e., each plot should contain 4 curves. To create a realistic setting, add some small noise (standard deviation of 1) to your correspondences. To account for RANSAC's random nature, repeat the experiments 10 times and report average values for the number of iterations and run-times.
Hint: You can measure timings in MATLAB using the `tic` and `toc` commands.

– Fix $d = 1$ for the following experiments. To evaluate RANSAC with the $T_{1,1}$ test and *local optimization*, perform the following experiments:

  1. For outlier ratios of 0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, run RANSAC with and without local optimization and report the run-time, the number of iterations, the inlier ratio, the number of inliers, and the average residual length among all true inliers in a table. To account for RANSAC's random nature, repeat each experiment 10 times and report average values. Use a small amount of noise on your correspondences (standard deviation of 1) .
  2. For a fixed outlier ratio of 0.4, vary the amount of noise you add to the correspondences (standard deviation $\sigma = 0.5$, $\sigma = 1$, $\sigma = 2$, $\sigma = 4$, $\sigma = 8$, $\sigma = 16$) and again report the same statistics as above in a table. Again, repeat your experiments 10 times to report average numbers.
  3. Make sure to explain how you set up your experiments in the text and to reference the figures in the text.

2.2 Theoretical Part

Answer the following questions in your report. Make sure to use your own words and justify your answers.

(a) The impact of the $T_{d,d}$ test. Sampling larger minimal samples increases the number of necessary RANSAC iterations. However, for $d = 1$ you should observe that RANSAC is faster than for $d = 0$. Explain this behavior.

(b) The impact of *local optimization.* You should observe that when using local optimization, RANSAC requires less iterations compared to not using local optimization. At the same time, you should observe that local optimization allows RANSAC to better handle noise on the image measurements. Explain this behavior.
Hint: Determine which assumptions are made by a minimal solver and the equation for determining the number of RANSAC presented in the lecture. Do they hold in practice? If not, how can local optimization help in these cases?

(c) Local optimization at low outlier ratios. Local optimization introduces a computational overhead. For higher outlier ratios, this overhead has little impact on the overall run-time. However, for low outlier ratios the increase in run-time is noticeable. Devise a strategy to avoid this problem. Make sure to justify your answer.

2.3 Advanced Part

While easy to implement, the $T_{d,d}$ test is sub-optimal. In the advanced part, you will implement a better test for rejecting a model if is seems unlikely that this model will yield to a new best model, the sequential probability ratio test (SPRT).
Read the paper describing the test and implement the test accordingly.

What to include in the report. Extend your report to include the following:

- Extend the "Method" section by a subsection that describes the SPRT in your own words (the level of detail that can be covered in about a 1/2 page is sufficient and it is not necessary to provide all theory from the paper).
- Extend the "Experimental Evaluation" section. Add another entry to the tables you created for RANSAC with $T_{d,d}$ and RANSAC with $T_{d,d}$ and local optimization: RANSAC with SPRT and local optimization. Provide the corresponding results (run-time, the number of iterations, the inlier ratio, the number of inliers, and the average residual length among all true inliers). Again, repeat the experiments 10 times to average out RANSAC's random nature.

3 Project III: Structure-from-Motion in 2D

In this project, you will implement a Structure-from-Motion pipeline in 2D, which is equivalent to stitching together multiple 2D images of a planar surface.

3.1 Mandatory Part

You can start with modifying your RANSAC implementation from Lab 3. Rather than estimating an affine transformation, you should implement a function that can estimate the homography between four correspondences (see the DLT algorithm presented in the lecture and do not forget to use the normalization discussed in the lecture). Given your RANSAC-based method for estimating the homography between two images, implement the following pipeline:

- For a given set of images, extract SIFT features in each image. Next, match SIFT features between pairs of images and estimate a homography for each pair. Store each estimated homography, together with the inlier matches to it. Use code from Lab 3 if possible (e.g., for feature extraction and matching).
- Select the image pair with the largest number of inliers as the starting pair. The first image thereby provides a global coordinate system and the homography $\mathtt{H}$ provides the mapping from the second image into the global coordinate system.[2]
- Iteratively add the other images: Among all images not yet included, select the image I that has the largest number of homography inliers with any of the images already included in the reconstruction. Let J be the corresponding image in the reconstruction and $\mathtt{H}_{J,I}$ the homography from I to J. Use $\mathtt{H}_{J,I}$ and the known mapping $\mathtt{H}_J$ from the coordinate system of J into the global coordinate system to compute the global homography $\mathtt{H}_I$ for image I. This step is the 2D equivalent to the "Extend motion" step from the Structure-from-Motion pipeline presented in the lecture.
- Finally, create a sparse map of the scene: For each image I in the reconstruction, use $\mathtt{H}_I$ to map all SIFT features found in I into the global coordinate system. Visualize the resulting 2D point cloud.

You can use the "graf" and "wall" (both showing changes in viewpoint), as well as the "bark" and "boat" (both showing zoom and rotation changes) datasets from here to evaluate your approach.

What to include in the report. Your report should consist of the following:

- A short abstract describing the problem behind the project and outlining your solution.
- A "Method" section describing your approach: Provide pseudo code for your 2D Structure-from-Motion system and explain each step in your own words. This section should be about one page long.
- A "Experimental Evaluation" section describing the experiments you perform with your approach as well as the results of these experiments:

[2]Assuming that your homography maps pixels in the second into the first image. If this is not the case, you need to invert the homography.

  - List the thresholds that you use for your RANSAC-based homography estimation, i.e., the maximum number of RANSAC iterations you run and the inlier threshold that you use.
  - Visualize the sparse maps you obtain with your method.
  - The dataset also provide a ground truth homography from the first image in the sequence to each other image. Use these ground truth homographies to create sparse maps and visualize them next to your maps.
  - For each dataset, measure the average distance between each sparse point in your map and the corresponding point position in the ground truth map. Report the results in a table.

3.2 Theoretical Part

Answer the following questions in your report. Make sure to use your own words and justify your answers.

(a) Triangulation. Triangulation is used in order to create 3D points in Structure-from-Motion. Given a set of 2D pixel positions $\mathbf{x}_i$, $i = 1, ..., k$, in k images and the projection matrices $\mathtt{P}_i$ of these images, explain how to obtain the 3D point that best represents these 2D measurements. Assume that $k > 2$.
How large does k need to be in order to be able to trust that the triangulated 3D point is not an outlier? Justify your answer.

(b) "Triangulation" in 2D. The 2D Structure-from-Motion algorithm that you implemented does not use any notion of 2D points (unlike the 3D points in classical Structure-from-Motion). How can you adapt the algorithm to use a notion of 2D points created from multiple 2D features? In particular, how would your minimal solver look like to generate a 2D point hypothesis? How would you refine a 2D point hypothesis given a set of inlier measurements from other images?

(c) Bundle Adjustment in 2D. Given a set of 2D points and the homographies mapping points from a global coordinate system to the coordinate system of each image, derive the equation that should be minimized during Bundle Adjustment. As for standard Bundle Adjustment, the goal is to minimize a sum of squared reprojection errors between 2D points and the corresponding measurements in the images. Explain your equation and make sure to explain all variables that you use.

3.3 Advanced Part

As a post-processing step, adjust your 2D Structure-from-Motion pipeline to create 2D points instead of using individual features:

- Form tracks of features: If feature i in image I and feature j in image J form an inlier correspondence, and if feature k in image K and feature j in image J form an inlier correspondence, then i, j, and k form a feature track. This definition can then be extended to more images.
- Given the positions of all features in a track in the global coordinate system, compute the 2D point position by averaging the positions.

Given the 2D points and the known homographies, perform non-linear optimization of the 2D point positions and the homographies using the 2D equivalent of bundle adjustment (as derived above). Use Gauss-Newton for the optimization.
After optimization, recreate the sparse map by mapping all features from all images into the global coordinate system.
Hint 1: You can use finite differences to avoid the need to analytically derive the Jacobians. However, note that this comes at the prize of increased computational requirements.
Hint 2: In order to accelerate the computations, use a random subset of all 2D points.

What to include in the report. Extend your report to include the following:

- Extend the "Method" section by a subsection by adding a description of the method that you implemented in your own words.
- Extend the "Experimental Evaluation" section by adding visualizations of the 2D maps generated with your more advanced pipeline. Also measure the errors in your 2D point positions with respect to the ground truth map you generated before.

4 Project IV: Generative Neural Networks

In this project, you will implement a generative neural network, more precisely a Variational Autoencoder (VAE).

4.1 Mandatory Part

Implement a Variational Autoencoder (VAE) to generate images of digits between 0 and 9. For inspiration for network architectures, please have a look at this paper (in particular, Section 3 and Appendices B and C.2) and this tutorial. Also experiment with variations of this architecture to see whether you can find a network that performs better. For simplicity, use a 2D latent space. For training and testing, use the MNIST dataset. You can find instructions on how to use the MNIST dataset inside MATLAB here.

What to include in the report. Your report should consist of the following:

- A short abstract describing the problem behind the project and outlining your solution.
- A "Method" section describing your approach: Explain how VAEs work in your own words and present and discuss the cost function that is optimized. Make sure to explain all variables that you use. You do not need to derive the cost function. Also discuss the network architectures that you use for experiments. This section should be about one page long.
- A "Experimental Evaluation" section describing the experiments you perform with your approach as well as the results of these experiments:
  - Describe the parameters you used for training, e.g., the learning rate, whether you used momentum, etc., as well as the number of epochs that you train your networks.
  - Show example images generated by the network architectures that you tried. Discuss the results, i.e., point out which network performs best and point out failure cases.

4.2 Theoretical Part

Answer the following questions in your report. Make sure to use your own words and justify your answers.

(a) The ELBO. In your own words, explain why we optimize the Empirical Lower BOund (ELBO) rather than $\log(p(\mathbf{x}))$ when training a VAE.

(b) VAE-GANs. In the lecture, we used the tendency of VAEs to generate blurry images to introduce Generative Adversarial Networks (GANs) as a way of learning a better loss function for the reconstructed images. In your own words, explain how the discriminator part of a GAN can be integrated into a VAE.

(c) Task driven losses. In some cases, VAEs or GANs are used to create additional training data. For example, in the advanced part below, we will use the VAE to generate training data for a digit classification problem. Following this example, can you think of a way of using a classifier as part of the loss when training the VAE? Explain your solution and the motivation behind your solution.

4.3 Advanced Part

In this part, you will experiment with your VAE in the context of creating training data. To this end, create a classification network for the digits 0 to 9 and train it on the training set of the MNIST dataset. Your network should be able to achieve a test accuracy of at least 90% on the test set. Since MNIST is a relatively easy dataset, your network does not need to be too deep (two convolutional layers (with a max-pooling layer in between) followed by two fully connected layers should be sufficient).
Next, perform the following experiment: Use half of the MNIST training set to train your VAE. For the other half, use 20%, 50%, 100% of its images, together with an equal amount of images generated by your VAE, to train your classifier.

What to include in the report. Extend your report to include the following:

- Extend the "Experimental Evaluation" section: Report the accuracy of the classifier trained with 20%, 50%, 100% of the images in the other half of the training set on the MNIST test set. How does the accuracy change when also training on synthetic images?