Computer Vision Project
Winter Term 2023/24
Mathias Zinnen

Sheet 5, starting from January 15th, 2024, due Feb 5th, 2024

This project consists of two exercises. Exercise 5.1 needs to be done by all course participants and can be done in groups. The second exercise (5.2) is mandatory only for those who do the 10 ects version of the course and should be done individually.

Each exercise is accompanied with several questions, that should be answered in written form (preferrably typeset in latex). You don't need to write much. If a question can be answered in one sentence that is fine. If you need more space, that is also okay but try to be concise.

Please create two zip files (one for the 5 ects group exercise, one for the 10ects individual exercise), each containing all code files necessary to run your solution and the writeup of the question answers and upload the zip files to studon. The solution for both individual and group exercises must be uploaded to StudOn by Feb 2, 2024. After uploading, please contact me (mathias.zinnen@fau.de) so we can schedule a date for you to present your solution.

Folder structure and Code details Unziping the exercise zip will create the folder `exercise-5` containing three subfolders, `code/`, `data/`, and `results/`. Unsurprisingly, the code to be modified by you can be found in the `code/` folder. The other two folders contain the input data to process (`data/`) and a place to store the results (`results/`). To install the packages necessary to run the code, please run `pip -r requirements.txt` in the root project folder.

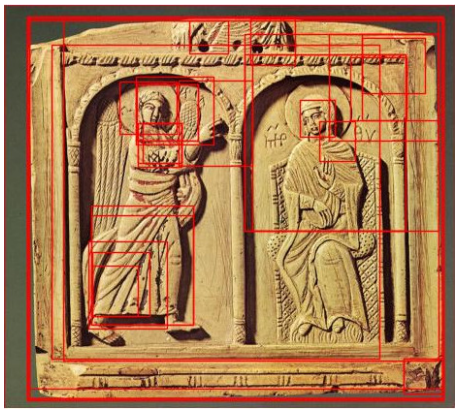
Exercise 5.1: Selective Search

Mandatory for all

In this exercise, you will implement the *selective search* algorithm for object detection. Initial regions are obtained from the Felzenszwalb algorithm as described in this paper. We will test the algorithm on 3 images each from 3 different domains, namely Art history, Christian Archaeology and Classical Archaelogy.

To help you with the implementation it might be helpful to read the paper introducing selective search (specifically section 3).

(a) Christian Arch - Annunciation scene



(b) Resulting image

Initial run: Running `main.py` unchanged will return empty regions for every test image resulting in no boxes detected and the output will be same as input image.

Data: The starter codes have one image loaded from Christian Arch. You must generate the results on all the three images provided under the data/chrisarch, data/arthist and data/classarch folders and store them in results folder.

In total, there are 8 tasks to be implemented.

Details of all of them are provided in the `selective_search.py` file in the code folder. Here's a brief description of all the tasks.

- Task 5.1: Generate initial segmentations. Refer to Felzenszwalb paper mentioned before. You may use the library function provided in sklearn.
- Task 5.2: Extract all the regions from the initial estimate based on four similarities: color, texture, size and fill as was discussed during the lecture. Note that one of the important subtask is to generate histograms, details are explained in the code-skeleton.
- Task 5.3: Extracting list of neighbors which basically are neighboring regions, later used for merging the regions.
- In a hierarchical procedure,
  - Task 5.4: merge the regions.
  - Task 5.5 and 5.6: Mark and remove the old similarities
  - Task 5.7: Calculate the new similarities with the new region
- Task 5.8: Generate the final regions which represents the proposals containing some important object information.

Questions

Q5.1.1 Felzenszwalb can already generate segmentations for the whole image, why do we need selective search at all?

Q5.1.2 Proposal filtering is implemented in `main.py`. What are the criteria based on which the boxes are filtered and what might be the effect? Do you agree with the criteria? Can you think of additional ones?

Q5.1.3 Selective Search iteratively merges regions of arbitrary shapes. How do we obtain rectangles (box proposals) from that?

Exercise 5.2: Detection Pipeline (Bonus)

Optional for 5 ects, mandatory for 10

Selective Search provides you with a set of region proposals where objects can likely be found. However, the task of object detection also consists of determining whether there actually is an object at the propopsed region, and what kind of object that is. In their paper, Uijlings et. al propose a pipeline that extracts features from the candidate regions using SIFT, and classifies the regions with a SVM.

In this exercise, you will implement a simplified version of their approach to obtain a complete object detection algorithm.

To train an object detection algorithm, you need ground truth annotations specifying where in the training images which object can be found. Since the images used in exercise 5.1 are not annotated with ground truth, you first need an annotated dataset to train your model. In this exercise, we will

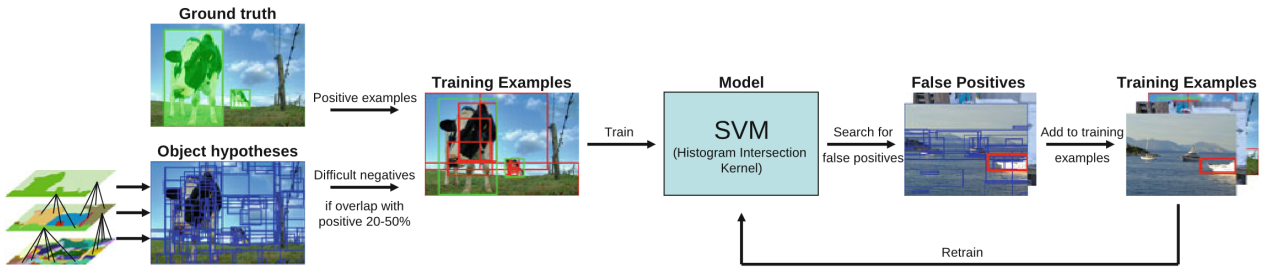


Fig. 2: Object Detection Pipeline by Uijlings et. al.

use a small toy dataset for the detection of balloons in photographs.

You can find a zipped version of the dataset in the `data` folder. Below, you will find three folders `train/`, `valid/`, and `test/`, each containing the images of the respective split and an annotation file (`_annotations.coco.json`). The data is given in the coco json, which is a very common data format for object detection annotations. For instructions on how to interpret the data, please refer to the official specification.

The general idea is that you create region proposals using your implementation of selective search from exercise 5.1, create positive and negative training samples based on overlap of the proposals and the ground truth annotations. Then extract features for these regions and train a SVM classifier to classify the proposed regions. Concretely this boils down to the following steps:

- Task 5.2.1: Generate proposals for each image in training and validation set using your implementation of exercise 5.1. Note that this step is taking the most time, it is thus advisable to save the resulting region proposals to a file and continue the following steps from there.
- Task 5.2.2: Load the previously generated proposals and create positive and negative training examples. One straightforward approach would be to consider the overlap between ground truth boxes (from the annotations files) and box proposals. Define thresholds t_p and t_n for positive and negative samples, respectively. You can then define proposals with an overlap below t_n as negative training samples (i.e. class background), and proposals with an overlap above t_p as postive (i.e. class balloon). Play with different thresholds.[1] Two sensible values to start with are $t_p = 0.75$ and $t_n = 0.25$.
- Task 5.2.2: Extract features for each negative and positive training sample to obtain an image representation that you can feed to the SVM classifier. Feel free to use an extraction method of your choice, for example the VLAD feature extractor you implemented in exercise 3, or a pre-trained CNN[2].
- Task 5.2.3: Train an SVM with the flattened feature vectors and the labels obtained during proposal generation. Feel free to use library functions, e.g. from sklearn.
- Task 5.2.4: Implement an inference script which enables region proposal, classification of proposed regions, and visualization of detected objects on arbitrary input images. To re-use the classifier trained in 5.2.3, you need to persist the model parameters. If you are using the sklearn, instructions can be found in the documentation.

Questions

Q5.2.1 Your implementation is a simplified version of Uijlings et. al approach (cf. figure 1b). In what ways does their version differ from yours?

[1] Feel free to use the ground truth data to generate more positive samples.
[2] Torchvision provides a potential starting point

Q5.2.2 What is the effect of changing the thresholds for positive and negative samples? Why do we need two thresholds instead of just one?

Q5.2.3 The balloon dataset is quite small and does not provide many training samples. Can you think of ways to increase the amount of training data?