Computer Vision II - Homework Assignment 2

Stefan Roth, Shweta Mahajan, Faraz Saeedan
Visual Inference Lab, TU Darmstadt

May 20, 2020

This homework is due on June 3rd, 2020 at 23:59.
Please read the instructions carefully!

General remarks

Your grade does not only depend on the correctness of your answer but also on clear presentation of your results and good writing style. It is your responsibility to find a way to *explain clearly how* you solve the problems. Note that we will assess your complete solution and not exclusively the results you present to us. If you get stuck, try to explain why and describe the problems you encounter – you can get partial credit even if you have not completed the task. Hence, please hand in enough information so that we can understand what you have done, what you have tried, and how your final solution works.

Every group has to submit its own original solution. We encourage interaction about class-related topics both within and outside of class. However, you are not allowed to share solutions with your classmates, and *everything you hand in must be your own work*. Also, you are not allowed to just copy material from the web. You are required to *acknowledge any source of information you use to solve the homework* (*i.e.* books other than the course books, papers, websites, etc). Acknowledgments will *not* affect your grade. Not acknowledging a source you rely on is a clear violation of academic ethics. Note that both the university and the department are very serious about plagiarism. For more details, see the department guidelines about plagiarism at https://www.informatik.tu-darmstadt.de/studium_fb20/im_studium/studienbuero/plagiarismus/index.de.jsp and http://plagiarism.org.

Programming exercises

For the programming exercises you will be asked to hand in Python code. Please make sure that your code complies with **Python 3.6 or higher / PyTorch 1.1**. In order for us to be able to grade the programming assignments properly, stick to the function names and type annotations that we provide in the assignments. Additionally, comment your code in sufficient detail so that it will be easy to understand for us what each part of your code does. Sufficient detail does not mean that you should comment every line of code (that defeats the purpose), nor does it mean that you should comment 20 lines of code using only a single sentence. Your Python code should display your results so that we can judge if your code works from the results alone. Of course, we will still look at the code. If your code displays results in multiple stages, please insert appropriate `sleep` commands between the stages so that we can step through the code. Group plots that semantically belong together in a single figure using subplots and don't forget to put proper titles and other annotations on the plots. Please be sure to name each file according to the naming scheme included with each problem. This also makes it easier for us to

grade your submission. And finally, please make sure that you included your name and email in the code.

Files you need

All the data you will need for the problems will be made available in Moodle.

What to hand in

Your hand-in should contain a PDF file (a plain text file is ok, too) with any textual answers that may be required. You must not include images of your results; your code should display these instead. For the programming parts, please hand in all documented `.py` scripts and functions that your solution requires. Make sure your code actually works and that all your results are displayed properly!

Handing in

Please upload your solution files as a single `.zip` or `.tar.gz` file to the corresponding Moodle area at https://moodle.tu-darmstadt.de/course/view.php?id=19436. **Please note that we will not accept file formats other than the ones specified!** Your archive should include your write-up (`.pdf` or `.txt`) as well as your code (`.py` scripts). If *and only if* you have problems with your upload, you may send it to `cv2staff@visinf.tu-darmstadt.de`

Late Handins

We will accept late hand-ins, but we will deduct 20% of the total reachable points for every day that you are late. Note that even 15 minutes late will be counted as being one day late! After the exercise has been discussed in class, you can no longer hand in. If you are not able to make the deadline, *e.g.* due to medical reasons, you need to contact us *before* the deadline. We might waive the late penalty in such a case.

Code Interviews

After your submission, we may invite you to give a code interview. In the interview you need to be able to explain your written solution as well as your submitted code to us.

Problem 1 – Markov random fields with Gaussian potentials 10 points

In class we have seen how to model a disparity prior using Markov random fields (MRFs). In particular we use a pairwise MRF which is a simple yet effective tool for this task. In this model we represent pixels by nodes in a graphical model and connect every node with its 4 nearest neighbors (left, right, top, bottom). Edges between horizontal and vertical neighbors induce a factor in the probability density which describes the compatibility of the neighboring pixels. For a disparity map $\boldsymbol{x}$ of height M and width N we write such a prior model of disparities as

$$p(\boldsymbol{x}) = \frac{1}{Z} \prod_{i=1}^{M-1} \prod_{j=1}^{N} f_V(x_{i+1,j} - x_{i,j}) \cdot \prod_{i=1}^{M} \prod_{j=1}^{N-1} f_H(x_{i,j+1} - x_{i,j}).$$

To model the horizontal and vertical potentials f_H and f_V we use Gaussian distributions:

$$f_{H/V}(d) = e^{-\frac{(d-\mu_{H/V})^2}{2\sigma_{H/V}^2}}.$$

For simplicity, we assume that the parameters of the horizontal and vertical potentials are the same, *i.e.* $\sigma_H = \sigma_V = \sigma$ and $\mu_H = \mu_V = \mu$. Since we will not need the normalization term $1/Z$, we will ignore it in the following, *i.e.* you can assume that it is 1.

Tasks:

- Implement this MRF prior with Gaussian potentials. In particular write a function

  `logp = mrf_log_prior(x)`,

  that computes the log of the unnormalized MRF prior density for a disparity map `x`. For the gaussian potentials set the parameters to $\sigma = 0.8$ and $\mu = 0.0$. Make sure that your implementation contains terms for *all* horizontal and vertical potentials. Please implement the calculation of the log of the Gaussian potential in a function

  `value = log_gaussian(x)`

  2 points

- Compute and report the log-prior density of the ground truth disparity of the Tsukuba dataset (from assignment 1).

  1 point

- Create a random "noise map" of the same dimensions as the Tsukuba disparity map by drawing uniform random numbers in $[0, 14]$ independently for each pixel. Please implement the corresponding function in the code template. Compute and report the log-prior density for this "noise map".

  2 points

- Also create a "constant map" ($\boldsymbol{x} \equiv 8.0$) (with same dimensions) and report its log-prior density. Please implement the corresponding function in the code template.

  2 points

- Compare the values of the log-prior density for these three disparity maps and comment on what the relative values say about how well this MRF model captures the properties of the visual world.

3 points

Problem 2 – Stereo with gradient-based optimization 17 points

In this problem we will put things together and implement an algorithm for stereo estimation using gradient-based MAP estimation. We use the model of the previous problem consisting of:

- A pairwise MRF prior over disparity values with Gaussian potentials
- Our stereo likelihood with Gaussian potentials

In the current and previous exercise you have implemented code for evaluating the (unnormalized) log-probability of a disparity map given two input images. For the gradient-based optimization we now need *gradients*! For the implementation of following tasks, make sure that your code works with continuous disparities.

Tasks:

- Implement the function

  ```
  [value, grad] = log_gaussian(x, sigma, mu)
  ```

  that calculates the value and the gradient (w.r.t. $\boldsymbol{x}$) of the log-density of the Gaussian distribution. Write the function such that it can also operate on matrix-valued inputs for which it should calculate the gradient element-wise. Here, we want to be more flexible and use given parameters σ and μ.

  1 point

- Now implement the MRF log-prior. Write a function

  ```
  [value, grad] = stereo_log_prior(x)
  ```

  that computes the value and gradient of the log-prior density w.r.t. every disparity value in $\boldsymbol{x}$. Here, set $\sigma = 0.8$ and $\mu = 0.0$. Note that the output gradient `grad` should have the same dimensions as the input disparity map. You can (and should) make use of the previously implemented function `log_gaussian`.

  1 point

- Now implement the stereo log-likelihood. Write a function

  ```
  [value, grad] = stereo_log_likelihood(x, im0, im1)
  ```

  that computes the value and gradient w.r.t. every disparity value given the input images $\boldsymbol{I}^0$ and $\boldsymbol{I}^1$. Here, set $\sigma = 1.2$ and $\mu = 0.0$. Again, `log_gaussian` should help you with that.

  5 points

- Now, put things together by implementing log-posterior. Write a function

  ```
  [value, grad] = stereo_log_posterior(x, im0, im1)
  ```

  that computes the value and gradient of the stereo posterior by combining your stereo prior and likelihood.

1 point

- With the log-posterior and its gradient at our fingertips, we can now estimate the disparity map. To that end, implement a function

  ```
  x = stereo(x0, im0, im1)
  ```

  that takes an initial guess of the disparity map and two input images $\boldsymbol{I}^0$ and $\boldsymbol{I}^1$. It should optimize the log-posterior to a local optimum by a gradient-based method. We recommend that you use BFGS optimizer from `Scipy.optimize` for this task.

2 points

Now, evaluate your algorithm with the supplied stereo image pair `i0.png` and `i1.png` with ground truth `gt.png`.

Tasks:

- Write the main function of the script `problem2` that loads the images and invokes your stereo method. After your algorithm has finished, show the ground truth disparity, your estimated disparity map, and the differences between both. What are your findings when you initialize the disparity map with a constant value, *i.e.* $\boldsymbol{x} \equiv 8$, or alternatively when you initialize with uniform noise in $[0, 14]$? What do you observe when you initialize with ground truth disparity? Why does your algorithm (probably) perform so poorly?

4 points

- Now implement coarse-to-fine estimation. For this, use subsequently downsampled versions[1] of each input image and run the algorithm first on the coarsest scale. For the next iteration initialize with an upscaled version of the current estimate of the disparity map (do not forget to scale the disparity values after upsampling).

2 points

- Show the all estimated disparity maps (of the different resolutions) in a single figure next to each other.

1 point

[1] If available, consider reusing your CV1 code for computing the Gaussian pyramid.