Computer Vision II - Homework Assignment 1

Stefan Roth, Krishnakant Singh, Samin Hamidi,
Visual Inference Lab, TU Darmstadt

May 10, 2023

This homework is due on May 24th, 2023 at 23:59.
Please read the instructions carefully.

General remarks

Your grade not only depends on the correctness of your answer, but also on clear presentation of your results and a 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. You can get partial credit even if you have not completed the task, but addressed some of its challenges. Hence, please hand in enough information demonstrating your effort: 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* (*e.g.* books other than the course books, papers, websites). Acknowledgements will *not* affect your grade. However, not acknowledging a source you used is a clear violation of academic ethics. Note that both the university and the department take any cases of plagiarism very seriously. For more details, see the department guidelines about plagiarism 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 runs with **Python 3.6 or higher** (and **PyTorch 1.8** or higher in later assignments). In order for us to be able to grade the programming assignments properly, insert your code in the provided function definitions. Feel free to experiment with your code in the main() function. However, make sure that your final submission can execute the code originally provided in the main() function. Additionally, comment your code in sufficient detail, so that it is clear 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. In your final submission, please do not use pop-up windows for visualising intermediate results or ask for user input, unless instructed in the assignment task. Of course, you are welcome, in fact even encouraged, to use visualisation while developing your solution. Please be sure to carefully read the comments in the provided code skeleton, as these provide important details on the function's semantics (*e.g.* what arguments it expects and what results it should return), as well as useful tips on implementation. 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 assignments will be made available over Moodle.

What to hand in

Your hand-in should contain a PDF file for any non-programming ("pen & paper") tasks in the assignment. You are encouraged to typeset your solution, but we will also accept scans or photos of your handwritten solution as long as the image quality does not inhibit its readability. For the programming parts, you must not submit any images of your results; your code should be able to generate these instead. Please hand in your completed version of `.py` scripts provided with the assignment. Make sure your code actually executes and that all functions follow the provided definitions, *i.e.* accept the specified input format and return the instructed output with correct types and dimensions.

Handing in

Please upload your solution files as a single `.zip` or `.tgz` file to the corresponding assignment on Moodle. **Please note that we will not accept file formats other than the ones specified!** Your archive should include your write-up (`.pdf`) as well as your code (`.py`). 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 - Probabilities and Statistics - 8 points

1. Show that the following relation is true:

$$p(z \mid y, x) = \frac{p(y \mid z, x)\, p(z \mid x)}{p(y \mid x)}$$

2 points

2. Given the joint distribution $p(x, y)$, how can the density function of x be computed?

1 point

3. For manipulating expectations, we have – among others – the following four relations

$$\begin{aligned}
\mathbb{E}[c] &= c, \\
\mathbb{E}[cf(x)] &= c\,\mathbb{E}[f(x)], \\
\mathbb{E}[f(x) + g(x)] &= \mathbb{E}[f(x)] + \mathbb{E}[g(x)], \\
\mathbb{E}[f(x)g(y)] &= \mathbb{E}[f(x)]\,\mathbb{E}[g(y)], \quad \text{if } x, y \text{ independent},
\end{aligned}$$

where c is a constant, and $\mathbb{E}[\cdot]$ denotes the expectation operator. Prove that the expectation of the product of two independent continuous random variables X and Y is

$$\mathbb{E}[XY] = \mathbb{E}[X]\,\mathbb{E}[Y]$$

2 points

4. There are two boxes. Box 1 contains 7 red and 3 white balls and box 2 contains 3 red and 9 white balls. A box is chosen at random $p(B = 1) = p(B = 2) = 0.5$ and a ball chosen at random from this box turns out to be red. What is the posterior probability that the red ball came from box 1?

3 points

Problem 2 - Modelling - 6 points

When we create models in computer vision, we are encoding prior assumptions about the real world in a mathematical framework. Here, we will have a look at the simple Markov Random Field (MRF) model for binocular stereo that we encountered in the lecture:

$$\begin{aligned}
p(\mathbf{d} \mid \mathbf{I}^0, \mathbf{I}^1) &\propto p(\mathbf{I}^1 | \mathbf{I}^0, \mathbf{d}) p(\mathbf{d} | \mathbf{I}^0) \\
p(\mathbf{I}^1 | \mathbf{I}^0, \mathbf{d}) &= \prod_{i,j} f(I^0_{i,j} - I^1_{i,j-d_{i,j}}) \\
p(\mathbf{d} | \mathbf{I}^0) &= \prod_{i,j} f_H(d_{i,j}, d_{(i+1),j}) f_V(d_{i,j}, d_{i,(j+1)}).
\end{aligned}$$

For the sake of simplicity, we assume for the moment that the disparity $\mathbf{d}$ is independent of $\mathbf{I}^0$. To recap the assumptions that we put in the above model, please briefly answer the following questions **in your own words**:

1. How can we justify to factorise the likelihood $p(\mathbf{I}^1 | \mathbf{I}^0, \mathbf{d})$ into individual terms for each pixel?

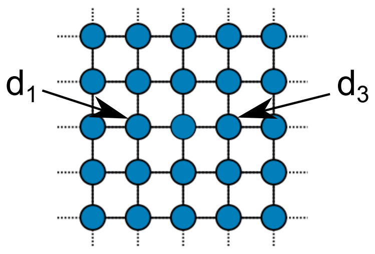

Figure 1: Markov Random Field prior over disparity values.

1 points

2. What are the limitations of the Gaussian likelihood function and what are the alternatives?

1 points

3. What assumptions does the pairwise MRF prior over the disparity values encode? Using 4 nearest neighbours formulation from the lecture, in Figure 1 is d_3 in the Markov blanket of d_1? Disparity values of which nodes (using labels from the image) are required such that d_1 and d_3 are independent?

3 points

4. What are the limitations of the compatibility function (or "potential functions") used in the Pott's model and can you think of alternatives?

1 points

Problem 3 - Stereo Likelihood - 9 Points

In this problem you are going to implement a simple likelihood model for stereo. Use `PIL Pillow` package for loading the images. Here, the `Numpy` package is used.

Write a python script `problem3.py` that does the following:

1. Implement the function `load_data(I0_path, I1_path, gt_path)` which takes the paths to the first and the second image, and the disparity ground truth. The paths to be used are already specified in `main()` from the Tsukuba dataset for your reference. Make sure that the returned image values are 64-bit floating points in the range $[0, 1]$. For the ground truth disparity map $\mathbf{d}_{\text{gt}}$ convert the values to 64-bit floating point. The resulting disparities should be (floating point) integer values in the range $[0, 16]$.

1 point

2. Implement function `rgb2gray(I)`, which converts the RGB image $\mathbf{I}$ to grayscale weighting the RGB channels with (0.1913, 0.4815, 0.3272), respectively.

1 point

3. Implement the function `shift_disparity(I, d)` that takes an image $\mathbf{I}$ and a disparity map $\mathbf{d}$ and shifts each pixel $I_{i,j}$ by the disparity $d_{i,j}$ to get the displaced image $\mathbf{I}^d$.

1 point

4. Since the ground truth disparity is not defined along the image borders, implement the function `crop(I,d)` that crops the image $\mathbf{I}$ to the (central) region where ground truth disparities $\mathbf{d}_{\text{gt}}$ are greater than 0.

1 point

5. Write a function `gaussian_nllh(I0, I1d, mu, sigma)` that computes the negative log of the Gaussian likelihood

$$p(\mathbf{I}^1|\mathbf{I}^0, \mathbf{d}) = \prod_{i,j} \mathcal{N}(I^0_{i,j} - I^1_{i,j-d_{i,j}} \mid \mu, \sigma). \tag{1}$$

Ensure numerical stability of your implementation as discussed in the lecture.
(*Hint:* Avoid direct multiplication of the factors.)

1 point

6. Write a function `laplacian_nllh(I0, I1d, mu, s)` that computes the negative log of the Laplacian likelihood

$$\hat{p}(\mathbf{I}^1|\mathbf{I}^0, \mathbf{d}) = \prod_{i,j} \frac{1}{2s} \exp\left\{ -\frac{|I^0_{i,j} - I^1_{i,j-d_{i,j}} - \mu|}{s} \right\}. \tag{2}$$

Again, ensure numerical stability of your implementation as discussed in the lecture.

1 point

7. Implement the function `make_noise(I, p)` that randomly chooses p% of the pixel positions and replaces the corresponding brightness with values chosen from the normal distribution with $\mu = 0.32$, and $\sigma = 0.78$. Use the numpy functions `np.random.choice()` and `np.random.normal()`. We will denote modified images as $\mathbf{I}_{\text{noise}}$.

1 point

8. Now, we have all components that we need to compute negative log-likelihoods for different corruption strengths. To do so, implement the function `get_nllh_for_corrupted(I0, I1d, p, mu, sigma, s)` that takes the second input image (corresponding to `i1.png`) and artificially generates pixels for which the brightness constancy is violated at p% of all pixels. Next, compute and return the negative log of the Gaussian likelihood $p(\mathbf{I}^1_{\text{noise}}|\mathbf{I}^0, \mathbf{d})$ and the negative log of the Laplacian likelihood $\hat{p}(\mathbf{I}^1_{\text{noise}}|\mathbf{I}^0, \mathbf{d})$ for the noisy and original images using the function parameters μ, σ, and s.

1 point

9. Compare your results and discuss your findings regarding outlier robustness based on the Gaussian and the Laplacian likelihood models. How do those results relate to the findings in the lecture?

1 point