Exercises for Probabilistic Graphical Models
Sheet No. 3

Bernt Schiele

Due Date: 4th January

Hand in: by 11:59pm by email to Apratim Bhattacharyya (abhattac at mpi-inf mpg de). Begin the subject of your e-mails with [PGM_Exercise3]. You should specify your first and last name as well matriculation number on submission. Please send your source code and all your results in a short report as a pdf. Explain your observations in this report.

In this exercise you are introduced to Markov random fields and their application to image denoising. We solve

$$p(true|noisy) = p(T|N) = p(N|T)P(T) \tag{1}$$

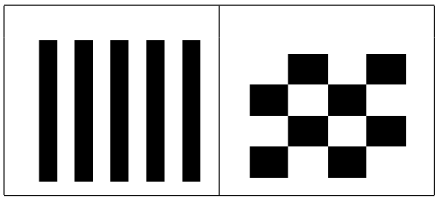
with gradient ascent method and compare our results to median filtering method. Finally, we have a look at different image priors $p(T)$ and question the meaning of this priors as well as the independence assumption.

1 Evaluation

Before we start with image denoising, we need an evaluation framework that generates noisy images and calculates a performance measure to compare our algorithms.

Tasks: (2 points)

- Generate artificial images with binary pixel values $x_{ij} \in \{0, 255\}$ that shows stripes (see left image), i.e.,



`T = toy_stripes(n, m, sSize)`,

and a checkerboard pattern (right image), i.e.,

`T = toy_checkerboard(n, m, cSize)`.

T is the true image of size $n \times m$ with stripe width $sSize$ or black square size $cSize$.

- Write a function that adds Gaussian noise to an existing image T:

  `N = add_noise(T, sigma)`

  Also, write a function which adds salt and pepper noise (randomly occurring black and white pixels):

  `N = add_sp_noise(T, p)`

- Write a function that calculates peak signal-to-noise ratio

$$PSNR = 20 \log_{10} \left(\frac{255}{\sqrt{err}} \right) \tag{2}$$

  with reconstruction error $err = (T - N)^2/(nm)$.

  `psnr = calc_psnr(T, N)`

Hints:

- You can use matlab function `random()` with $\mu = 0$ to add or substract noise values from each pixel. Be sure that your pixel values are still in the range of $[0, 255]$.
- You should fix the random seed in your functions with Matlab function `RandStream`. This makes all results comparable to each other.

2 Median Filtering

A simple image denoising technique is median filtering. As you will find out, it often leads to blurred images.

Tasks: (3 points)

- Write a function `T = median_filter(N, nsize)` that replaces each pixel in a noisy image N with the median of the pixel values in a window of size $nsize \times nsize$ around it. Take care at the image borders.
- Evaluate this denoising procedure on our different artificial examples with 10% salt'n'pepper noise and with the image `la.png` downloadable from the course website after adding Gaussian noise with $\sigma = 25$ (10% of the range). Show images before and after denoising and document PSNR for these images. What do you observe?
- Vary the amount of noise for the image `la.png`.

3 MRF-based Denoising with Gradient Ascent

A better denoising technique is a MRF-based method with gradient ascent:

$$T^{t+1} \leftarrow T^t + \eta \nabla_T \log p(T|N) \tag{3}$$

with $\log p(T|N) = \log p(N|T) + \log p(T) + const.$ *const* is ignored in all further computations as it doesn't depend on T. We need the gradient of both the likelihood $\log p(N|T)$ and the prior $\log p(T)$. For simplicity, let's start with a Gaussian log-likelihood

$$\log p(N|T) = \sum_{i,j} \left(-\frac{1}{2\sigma^2} (N_{i,j} - T_{i,j})^2 \right), \tag{4}$$

as well as a Gaussian log-prior

$$\log p(T) = \sum_{i,j} \log(f_H(T_{i,j}, T_{i+1,j})) + \log(f_V(T_{i,j}, T_{i,j+1})) \tag{5}$$

with

$$\log f_H = -\frac{1}{2\sigma^2} (T_{i,j} - T_{i+1,j})^2 \tag{6}$$

for horizontal neighbors and $\log f_V$ analogously.

Tasks: (8 points)

- Write a function

  `lp = denoising_lp(T, N, sigma)`

  to compute the log-posterior,

  `g = denoising_grad_llh(T, N, sigma)`

  to compute the gradient of the log likelihood $\log p(N|T)$, and

  `g = mrf_grad_log_gaussian_prior(T, sigma)`

  to calculate the gradient of the Gaussian prior $\log p(T)$.

- Finally, implement the gradient ascent to denoise the image

  `T = denoising_grad_ascent(N, sigma, eta)`.

  You should initialize the gradient ascent with the noisy image N.

- Explain your parameter tuning. Which observation do you make for different σ and η? (See hints.)

- Evaluate your implementation with the noisy images from Task 2 for the same noise parameters. Check the PSNR and the increasing log-posterior curve. Compare your results with median filtering - which algorithm shows better results and why?

- What happens if you initialize gradient ascent with the output of median filtering? Is there an improvement in performance or a faster convergence observable?

Hints:

- The gradient `g` should have the same size as the input image while the log-posterior `lp` is just a scalar.
- You may need many iterations to reach approximate convergence (> 1000). You can verify your algorithm by displaying the log-posterior curve.
- Start with small artificial images to see the correctness of your algorithm and to get a feeling for the parameters. Usually, it is recommended to test different powers of 10, i.e., $\sigma, \eta \in \{..., 10^{-1}, 10^{0}, 10^{1}, ...\}$.

4 A Different Prior

As we know from the lecture, a Gaussian distribution does not match the statistics of a natural image very well. A more appropriate distribution is the Student-t distribution:

$$f_H(T_{ij}, T_{i+1j}) = \left(1 + \frac{1}{2\sigma^2}(T_{i,j} - T_{i+1,j})^2\right)^{-\alpha} \tag{7}$$

Tasks: (5 points)

- Implement the gradient of this log-prior given above as
  `g = mrf_grad_log_student_prior(T, sigma, alpha)`.
  Do not forget the log in your partial derivatives.
- Evaluate your denoising algorithm with this new prior, $\alpha = 1$ works fine. What do you observe?
- Display the gradient of the log-prior for the test images. Explain your results.

5 Independence Assumption

Finally, we question the assumption that the noise in each pixel is independent.

Tasks: (2 points)

- Is this assumption reasonable?
- What happens if you add spatially dependent noise to your image, e.g., a "noisy stripe" like in old movies.
- Show results for both priors (Gaussian and Student-t distributions).

6 Bonus

(2 points) So far, we have used MRF-based image denoising for gray images only. How would you denoise a color image?