Computer Vision Project
Winter 2017/18
Franz Köferl, Peter Fürsattel, Vincent Christlein, André Aichert,
Tobias Geimer, Thomas Köhler, Franziska Schirrmacher

Sheet 5, starting from January 8th, 2018, due February 2nd, 2018, 14h

---

Introduction and General Comments:

- Throughout this exercise, you can test your implementations on a common example dataset[a].
  - Download `lowResData.mat`, which contains a set of low-resolution frames (`LRImages`).
  - We also provide the motion parameters (`motionParams`) for this sequence. Notice that the motion parameters associated with each frame are represented by an affine homography. Each homography models the motion *towards* the first frame selected as reference.
- More hints can be found in `srExample.m`, which is provided as supplementary material. Use this example script and the given parameters as a baseline for your experiments.
- Test your algorithms with different parameters and try to get a deeper understanding regarding their behaviors. Show your implementation and your results to one of the advisors.

[a] More example datasets can be found at https://users.soe.ucsc.edu/ milanfar/software/sr-datasets.

Exercise 5.1: Non-Uniform Interpolation

In this exercise, we will approach multi-frame super-resolution via *non-uniform interpolation*. Below, we implement *motion compensation* and *interpolation* as the two basic steps of super-resolution. Please refer to the lecture slides or [1] regarding a detailed explanation of this approach.

a) Motion Compensation

We use a sequence of K low-resolution frames $\boldsymbol{y}^{(1)}, \dots, \boldsymbol{y}^{(K)}$ with $\boldsymbol{y}^{(k)} \in \mathbb{R}^M$ along with the motion parameters represented as affine homographies $\boldsymbol{H}^{(1)}, \dots, \boldsymbol{H}^{(K)}$ to recover a high-resolution image $\boldsymbol{x} \in \mathbb{R}^N$, where $N = sM$ for the integer magnification factor $s > 1$. First, we implement a motion compensation of the low-resolution frames. For this purpose, use the motion parameters and the magnification factor to warp the (homogenous) pixel coordinates $\boldsymbol{u}_i = (u, v, 1)^\top$, $i = 1, \dots, M$ in each frame $\boldsymbol{y}^{(k)}$ onto the high-resolution gird of $\boldsymbol{x}$ according to:

$$\boldsymbol{u}'_i \cong \tilde{\boldsymbol{H}}^{(k)} \boldsymbol{u}_i, \tag{1}$$

where $\cong$ denotes equality up to scale. Here, $\tilde{\boldsymbol{H}}^{(k)}$ represents a homography determined from $\boldsymbol{H}^{(k)}$ and the magnification factor s to describe this transformation.

b) Interpolation on a High-Resolution Grid

We interpolate the unknown high-resolution image based on the motion compensation. Use the transformed pixel coordinates $\boldsymbol{u}'_i$, $i = 1, \dots KM$ of all low-resolution frames along with the corresponding intensity values y_i, $i = 1, \dots, KM$ to interpolate the high-resolution intensity values x_i, $i = 1, \dots N$. Use bicubic interpolation to obtain each x_i, which leads to the desired high-resolution image $\boldsymbol{x}$.

Hint: You can use `meshgrid` and `griddata` to perform a bicubic interpolation.

Test your implementation with different parameters, e. g. different numbers of input frames or different magnification factors. See Figure 1 for an example based on the provided dataset.

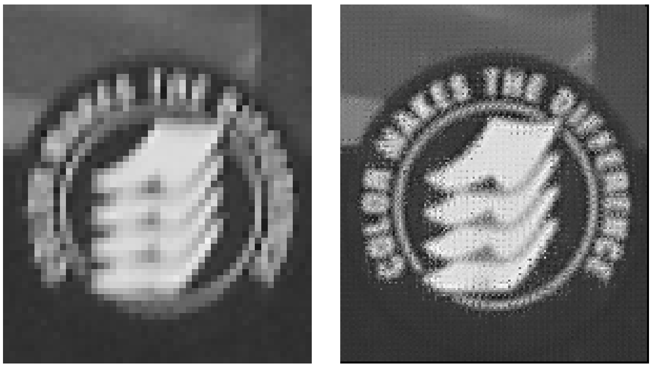

Fig. 1: Left: single low-resolution frame from `lowResData.mat`. Right: result of non-uniform interpolation ($K = 26$ frames, magnification $s = 3$)

Exercise 5.2: Iterative Optimization

In this exercise, we formulate multi-frame super-resolution as an inverse problem that is solved by means of iterative numerical optimization. In particular, we are interested in modeling super-resolution as a statistical parameter estimation problem. Please refer to the lecture slides regarding the mathematical derivation of this method. More details can be found in the work of Elad and Feuer [2]. Below, we develop the different components required for this approach step-by-step.

a) Modeling the Image Formation Process

We model the formation of low-resolution frames from a high-resolution image according to:

$$\begin{aligned} \boldsymbol{y} &= \boldsymbol{W}\boldsymbol{x} + \boldsymbol{\epsilon} \\ \text{where } \boldsymbol{y} &= \begin{pmatrix} \boldsymbol{y}^{(1)} & \dots & \boldsymbol{y}^{(K)} \end{pmatrix}^\top \text{ and } \boldsymbol{W} = \begin{pmatrix} \boldsymbol{W}^{(1)} & \dots & \boldsymbol{W}^{(K)} \end{pmatrix}^\top . \end{aligned} \tag{2}$$

In this model, $\boldsymbol{W} \in \mathbb{R}^{KM \times N}$ denotes the system matrix that is assembled from motion and imaging parameters and $\boldsymbol{\epsilon} \in \mathbb{R}^{KM}$ denotes additive observation noise. For the sake of convenience, the image $\boldsymbol{X} \in \mathbb{R}^{N_u \times N_v}$ given in matrix notation is represented as *parameter vector* $\boldsymbol{x} \in \mathbb{R}^N$, $N = N_u \cdot N_v$ using line-by-line scanning according to $\boldsymbol{x} = \begin{pmatrix} X_{1,1} & X_{1,2} & \dots & X_{N_u,N_v} \end{pmatrix}^\top$.

1. In the supplementary material for this exercise, we provide the code for the composition of $\boldsymbol{W}^{(k)}$ in (2) as a sparse matrix. Get familiar with the use of this function based on the provided documentation (`help composeSystemMatrix`).
2. Assemble the system matrix $\boldsymbol{W}$ and the low-resolution observation vector $\boldsymbol{y}$ from a set of low-resolution frames and the corresponding motion and imaging parameters. Hints:
   - Implement utility functions to convert from a matrix representation of an image to a parameter vector (`imageToVector`) and vice vera (`vectorToImage`).
   - Use `spalloc` to initialize a sparse matrix in MATLAB.

b) Maximum Likelihood Estimation

We can now employ the image formation model to determine a *maximum likelihood* (ML) estimate for the unknown high-resolution image $\boldsymbol{x}$. That is, we reconstruct this image according to:

$$\boldsymbol{x}_{\mathrm{ML}} = \arg\min_{\boldsymbol{x}} F(\boldsymbol{x}), \qquad \text{where } F(\boldsymbol{x}) = ||\boldsymbol{y} - \boldsymbol{W}\boldsymbol{x}||_2^2 , \tag{3}$$

assuming independent and identically distributed (i.i.d.) observations $\boldsymbol{x}$ and zero-mean normal distributed noise $\boldsymbol{\epsilon}$ in (2). Implement the minimization of this energy function using *gradient descent*

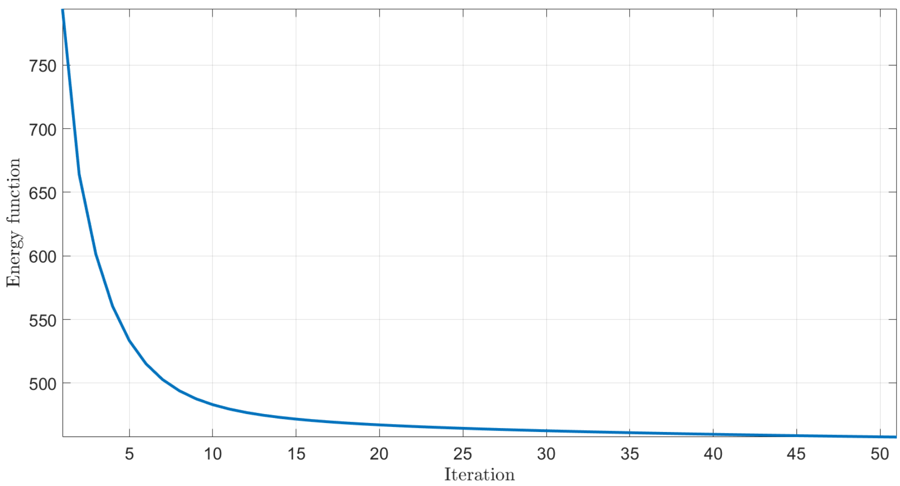

Fig. 2: Energy function $F(\boldsymbol{x}^t)$ over the gradient descent iterations ($\alpha = 0.05$).

iterations. For this purpose, we update an estimate $\boldsymbol{x}^t$ at iteration t to $\boldsymbol{x}^{t+1}$ according to:

$$\boldsymbol{x}^{t+1} = \boldsymbol{x}^t + \alpha \boldsymbol{p}^t, \tag{4}$$

where $\alpha > 0$ is a constant step size parameter and $\boldsymbol{p}^t$ denotes the search direction.

1. Initialize $\boldsymbol{x}^0$ with an upsampled version of the reference frame $\boldsymbol{y}^{(1)}$ using *bicubic* interpolation.
2. Implement and compare two methods to compute the search direction $\boldsymbol{p}^t$ in (4).
   - Compute the search direction according to the gradient of the energy function $F(\boldsymbol{x})$ (*steepest descent*), i. e. $\boldsymbol{p}^t = -\nabla_{\boldsymbol{x}} F(\boldsymbol{x}^t)$ (hint: $\nabla_{\boldsymbol{x}} ||\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}||_2^2 = -2\boldsymbol{A}^\top(\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x})$).
   - Compute the search direction for gradient descent according to *Zomets method* [3]:

$$\boldsymbol{p}^t = -K \cdot \text{median}_{k=1,\ldots,K}\, g(\boldsymbol{y}^{(k)}, \boldsymbol{x}^t), \quad \text{where } g(\boldsymbol{y}^{(k)}, \boldsymbol{x}) = \nabla_{\boldsymbol{x}} \left|\left| \boldsymbol{y}^{(k)} - \boldsymbol{W}^{(k)}\boldsymbol{x} \right|\right|_2^2. \tag{5}$$

3. Find a usable termination criterion for the gradient descent iterations. Visualize the progress of the estimation by plotting the energy function values $F(\boldsymbol{x}^t)$ at different iterations, see Figure 2. Test the behavior for different optimization related parameters (e. g. different step sizes α).
4. Test your implementation on our example dataset. Compare Zomets method to simple steepest descent under several parameter settings. In particular, test with different numbers of input frames (e. g. using 4, 8 or 26 frames) and parameter settings (e. g. different magnification factors).

c) Maximum A-Posteriori Estimation

Let us now extend the ML method to *maximum a-posteriori* (MAP) estimation. Thus, we aim at estimating a high-resolution image according to:

$$\boldsymbol{x}_{\text{MAP}} = \arg\min_{\boldsymbol{x}} F(\boldsymbol{x}), \qquad \text{where } F(\boldsymbol{x}) = ||\boldsymbol{y} - \boldsymbol{W}\boldsymbol{x}||_2^2 + \lambda R(\boldsymbol{x}), \tag{6}$$

where $R(\boldsymbol{x})$ denotes a regularization term and $\lambda \geq 0$ is the corresponding regularization weight.

1. The regularization term $R(\boldsymbol{x})$ models prior knowledge on the appearance of the image $\boldsymbol{x}$. Typically, this term is defined via the image gradient to penalize intensity variations in $\boldsymbol{x}$. In this exercise,

Fig. 3: From left to right: super-resolution based on ML estimation, MAP estimation with steepest descent and TV regularization as well as MAP estimation using Zomets method and TV regularization ($K = 26$ frames, magnification $s = 3$).

we model the image gradient $\nabla_u \boldsymbol{x}$ in *horizontal* direction according to:

$$\begin{aligned} \nabla_u \boldsymbol{x} &= \boldsymbol{Q}_u \boldsymbol{x} \\ \text{where } \boldsymbol{Q}_u &= \begin{pmatrix} +1 & -1 & & & & \\ & +1 & -1 & & & \\ & & & \ddots & & \\ & & & & +1 & -1 \\ & & & & & +1 \end{pmatrix} \in \mathbb{R}^{N \times N}. \end{aligned} \tag{7}$$

Similar, we can model the image gradient $\nabla_v \boldsymbol{x}$ in *vertical* direction by $\nabla_v \boldsymbol{x} = \boldsymbol{Q}_v \boldsymbol{x}$.

Implement the composition of the matrices $\boldsymbol{Q}_u$ and $\boldsymbol{Q}_v$ for a given image size (hint: use sparse matrices to store $\boldsymbol{Q}_u$ and $\boldsymbol{Q}_v$).

2. Extend your gradient descent optimization with the following image priors to regularize the estimated high-resolution image.
   - Gaussian prior: $R(\boldsymbol{x}) = ||(\boldsymbol{Q}_u + \boldsymbol{Q}_v)\,\boldsymbol{x}||_2^2$
   - Isotropic Total Variation (TV): $R(\boldsymbol{x}) = \sqrt{||\boldsymbol{Q}_u \boldsymbol{x}||_2^2 + ||\boldsymbol{Q}_v \boldsymbol{x}||_2^2 + \epsilon}$, where $\epsilon > 0$ is a small constant (e. g. $\epsilon \approx 10^{-4}$) to make TV differentiable.

   Notice that $\boldsymbol{Q}_u$ and $\boldsymbol{Q}_v$ do not depend on $\boldsymbol{x}$. Thus, these matrices can be pre-computed and re-used at each iteration to compute the search direction.

3. Determine (by experiments) reasonable settings for the regularization weights λ.
4. Test your implementation on our example dataset. Compare MAP estimation to ML estimation using different parameter settings, see e. g. Figure 3.

References

[1] Sung Cheol Park, Min Kyu Park, and Moon Gi Kang. Super-resolution image reconstruction: a technical overview. *IEEE Signal Processing Magazine*, 20(3):21–36, 2003.

[2] M. Elad and A. Feuer. Restoration of a single superresolution image from several blurred, noisy, and undersampled measured images. *IEEE Trans Image Process*, 6(12):1646–1658, 1997.

[3] A. Zomet, A. Rav-Acha, and S. Peleg. Robust super-resolution. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1, 2001.