Matlab Toolbox for Multichannel Blind Deconvolution and Demosaicing version 1.0

Filip Šroubek and Jan Flusser

February, 2007

Contents

1 Introduction

Note: *This MBD toolbox performs only multichannel blind deconvolution. In order to obtain a full version that includes superresolution please contact the authors of the toolbox. However, one step towards superresolution is included and this is demosaicing of color images in Bayer pattern.*

Imaging plays a key role in many diverse application areas, such as astronomy, remote sensing, microscopy or tomography, just to name few. Due to imperfections of measuring devices (optical degradations, limited size of CCD sensors) and instability of observed scene (object motion, media turbulence) acquired images are blurred, noisy and may exhibit insufficient spatial and/or temporal resolution.

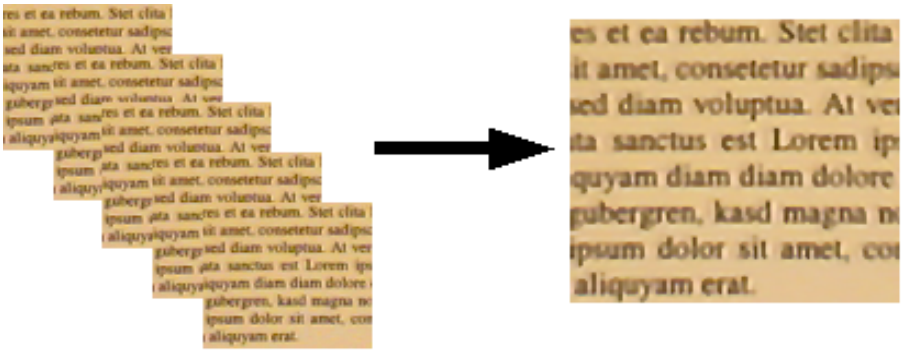
In order to recover the original image, techniques called blind deconvolution and superresolution remove the blur and increase the resolution, respectively. A necessary condition for the methods to be stable is to have more than one image of the scene (multiframe imaging). Differences between images are necessary to provide new information but they can be almost

imperceivable, e.g., subtle spatial shifts or slight modification of acquisition parameters (focus length, aperture size).

Current multiframe blind deconvolution techniques require no or very little prior information about the blurs and they are sufficiently robust to noise to provide satisfying results in most of the real applications. However, they can hardly cope with low-resolution images since in this case a standard convolution model is violated. On the contrary, state-of-the-art superresolution techniques achieve remarkable results in resolution enhancement by estimating the subpixel shifts between images but lack any apparatus for calculating the blurs. The superresolution methods either assume that there is no blur or that it can be estimated by other means.

We propose a unifying system that simultaneously estimates blurs and the original undistorted image, all in high resolution, without any prior knowledge of the blurs or original image. We accomplish this by formulating the problem as constrained least squares energy minimization with appropriate regularization terms, which guarantee close-to-perfect solution in the noiseless case.

In the example below we can see five LR images (left) and the HR image (right) estimated from the LR images using this tool.



2 The math behind BSR

Imaging devices have limited achievable resolution due to many theoretical and practical restrictions. An original scene with a continuous intensity function $o(x, y)$ warps at the camera lens because of the scene motion and/or change of the camera position. In addition, several external effects blur images: atmospheric turbulence, camera lens, relative camera-scene motion, etc. We will call these effects *volatile blurs* to emphasize their unpredictable and transitory behavior, yet we will assume that we can model them as convolution with an unknown point spread function (PSF) $h(x, y)$. Finally, the CCD discretizes the images and produces digitized noisy image $z(i, j)$ (frame). We refer to $z(i, j)$ as a *low-resolution (LR) image*, since the spatial resolution is too low to capture all the details of the original scene. In conclusion, the acquisition model becomes

$$z(i, j) = D[h(x, y) * o(W(x, y))] + n(i, j)\,, \tag{1}$$

where n is additive noise and W denotes the geometric deformation (warping). $D[\cdot] = S[g * \cdot]$ is the *decimation operator* that models the function of the CCD sensors. It consists of convolution with the *sensor PSF* $g(x, y)$ followed by the *sampling operator* S, which we define as multiplication by a sum of delta functions placed on a evenly spaced grid. The above model for one single observation $z(i, j)$ is extremely ill-posed. To partially overcome this difficulty, we assume that multiple LR observations of the original scene are available. Hence we write

$$z_k(i, j) = D[h_k(x, y) * o(W_k(x, y))] + n_k(i, j)\,, \tag{2}$$

where k is the acquisition index and D remains the same in all the acquisitions. In the perspective of this multiframe model, the original scene $o(x, y)$ is a single input and the acquired LR images $z_k(i, j)$ are multiple outputs.

The acquisition model in Eq. (2) embraces three distinct cases frequently encountered in literature. First, we face a registration problem, if we want to resolve the geometric degradation W_k. Second, if the decimation operator D and the geometric transform W_k are not considered, we face a *multichannel* (or multiframe) *blind deconvolution* (MBD) problem. Third, if the volatile blur h_k is not considered or assumed known, and W_k is suppressed up to a subpixel translation, we obtain a classical superresolution (SR) formulation. In practice, it is crucial to consider all three cases at once. We are then confronted with a problem of *blind superresolution* (BSR), which the proposed tool tries to solve.

In our BSR we know the LR images $\{z_k\}$ and we search for a HR estimate $\hat{o}$ of the original image o supposing that only D is known on the right hand side of (2).

In principle, W_k can be a very complex geometric transform that must be estimated by image registration or motion detection techniques. We have to keep in mind that sub-pixel accuracy is necessary for SR to work. Standard image registration techniques can hardly achieve this and they leave a small misalignment behind. Therefore, we will assume that complex geometric transforms are removed in the preprocessing step[1] and W_k reduces to a small translation. We can then perform an important simplification and include the unknown translation into the estimation of volatile blurs, which we will denote in the sequel as $\hat{h}_k$.

In order to solve the BSR problem, i.e, determine the HR image $\hat{o}$ and volatile PSFs $\hat{h}_k$, we adopt a classical approach of minimizing a regularized energy function. This way the method will be less vulnerable to noise and better posed. The energy takes the following form:

$$E(\hat{o}, \{\hat{h}_k\}) = \sum_{k=1}^{K} \|D(\hat{h}_k * \hat{o}) - z_k\|^2 + \lambda_o Q(\hat{o}) + \mu_h \sum_{k=1}^{K} Q(\hat{h}_k) + \lambda_h R(\hat{h}_1, \ldots, \hat{h}_K)\,. \tag{3}$$

The first term measures the fidelity to the data and emanates from our acquisition model (2). The remaining three are regularization terms with positive weighting constants λ_o, λ_h and μ_h. The weighting constants are user-defined parameters; see Section 5.2. Regularization $Q(\hat{o})$ (and $Q(\hat{h}_k)$) is a smoothing term of the form

$$Q(\hat{o}) = \int \phi(|\nabla \hat{o}|)\,, \tag{4}$$

where $|\nabla \hat{o}|$ is the size of the image gradient. Function $\phi(s)$ can have different forms. We have implemented the following forms (see Section 5.2): s^2 (Tichonov regularization), s (total variation) and $\sqrt{1+s^2} - 1$ (hypersurface minimal function). Functional R is a consistency term that binds the different volatile PSFs to prevent them from moving freely and unlike the fidelity term (the first term in (3)) it is based solely on the observed LR images.

To find a minimizer of the energy function, we perform alternating minimizations (AM) of E over $\hat{o}$ and $\hat{h}_k$. The advantage of this scheme lies in its simplicity. Each term of (3) is quadratic and therefore convex (but not necessarily strictly convex) and the derivatives are easy to calculate. This AM approach is a variation on the steepest-descent algorithm. The search space is a concatenation of the blur subspace and the image subspace. The algorithm first descends in the image subspace and after reaching the minimum, it advances in the blur subspace in the direction orthogonal to the previous one, and this scheme repeats.

A detailed description of multichannel blind deconvolution using the AM algorithm and extension to super resolution can be found in our papers given below.

[1]The registration step is not part of this tool.

Further reading:

- Sroubek F., Flusser J.: Multichannel blind iterative image restoration. IEEE Transactions on Image Processing, 12 (2003), 9, 1094-1106.
- Sroubek F., Flusser J.: Multichannel blind deconvolution of spatially misaligned images. IEEE Transaction on Image Processing, 14 (2005), 7, 874-883.
- Sroubek F., Flusser J.: Resolution enhancement via probabilistic deconvolution of multiple degraded images. Pattern Recognition Letters, 27 (2006), 4, 287-293.
- Sroubek F., Flusser J., Cristobal G.: Multiframe blind deconvolution coupled with frame registration and resolution enhancement. In: Blind Image Deconvolution: Theory and Applications. (Campisi P., Egiazarian K. eds.). CRC Press (2007).

3 Implementation

The MBD tool is implemented completely in Matlab. The required version of Matlab is 7.1 or higher including *Optimization Toolbox*. No other toolboxes are necessary.

The MBD code is divided into several Matlab functions (m-files). Each m-file header contains a help section that describes the function usage, input and output parameters and states its role in the MBD application. You can display the help by typing `help <name of the m-file>` on the Matlab command line. Each m-file is also well commented. See the source files.

MBD consists of the following m-files (grouped together by their function):

GUI	
mbd_gui.m	GUI main function
mbd_gui.fig	GUI internal data file
gui_help.pdf	this manual
MBD algorithm	
blindCSR.m	main function performing MBD (AM algorithm)
minUstep.m	1. step of AM; minimization with respect to the HR image ($\hat{o}$)
minHIstep.m	2. step of AM; minimization with respect to the PSFs ($\hat{h}_k$)
hConstr.m	imposes additional constraints on the PSF estimates
uConstr.m	imposes additional constraints on the HR image estimate
Support functions for blur consistency regularization	
fftR2matrix.m	calculates fundamental matrix used in blur consistency regularization R
Support functions for smoothness regularization	
Tichonov.m	Tichonov regularization in Q
TV.m	total variation regularization in Q
CTV.m	color total variation regularization in Q
HyperSurface.m	Hypersurface regularization in Q
Utilities	
bayer2cell.m	converts images stored in Bayer pattern to a set of color channels and its components (see color pattern)
bpattern.m	returns information about the selected Bayer pattern
cog.m	calculates the center of gravity of PSFs
convdown.m	generates degraded (blurred and LR) images from one HR image using a given set of PSFs (for testing purposes)

decmat.m	returns a decimation matrix (D)
dispBPIm.m	displays images stored with Bayer pattern
dispIm.m	displays ordinary 2D matrices as gray-scale images and 3D matrices as color images
fftconv2matrix.m	preprocessing step used in minUstep.m
findH.m	first rough estimate of PSFs
gaussmask.m	returns 1D Gaussian blur used in the decimation matrix
initblur.m	initializes PSFs as delta functions
kron3d.m	Kronecker product for 3D matrices
linscale.m	linear scaling of image intensity values
loadPhotos.m	loads image files into Matlab (for testing purposes)
ncorr.m	calculates normalized correlation in loadPhotos.m to perform simple registration (for testing purposes)
normimg.m	normalizes input images
pmse.m	calculates percentage MSE
unvec.m	converts an image in a vector form back to a 2D matrix
vec.m	vectorizes an image

4 Installation

The MBD tool is implemented in Matlab version 7.1 and is distributed as source m-files. The source files work only, if Matlab is installed.

Unzip the m-files (or p-files) into any directory and run Matlab. In Matlab, change to the directory, where you have the source files and start the tool by typing: `mbd_gui;`

Requirements: The tool was developed in *Matlab version 7.1* and requires *Optimization Toolbox*.

5 How to use the GUI

The MBD tool performs blind deconvolution of blurred images, with very little knowledge of the degradation process. **The fundamental assumption is that we have more then one degraded image of the original scene and that the degradation in the images is sufficiently different.** If the blurs are known in advance they can be loaded into the process and non-blind deconvolution (without estimation of blurs) is then possible as well.

The GUI of the MBD tool consists of one window shown in Fig. 1. In the application menu bar you find two items: *View* and *Help*. The *View* item opens a tool bar from which you can select several tools, such as "zoom in" and "zoom out", for editing images. The *Help* item displays this help.

5.1 Input images

The plot at the top left (see ① in Fig. 1) displays input blurred images. To load images, use a context menu, which is activated whenever the user right-clicks on the plot. Before loading images, select the correct color pattern in *Settings*. The context menu contains the following items:

- *load from workspace* - to load blurred images from a cell array stored in the Matlab workspace. The format of the cell array is $\{image_1, image_2, \cdots, image_K\}$, where $image_k$ are 2D or 3D matrices that contain the images.

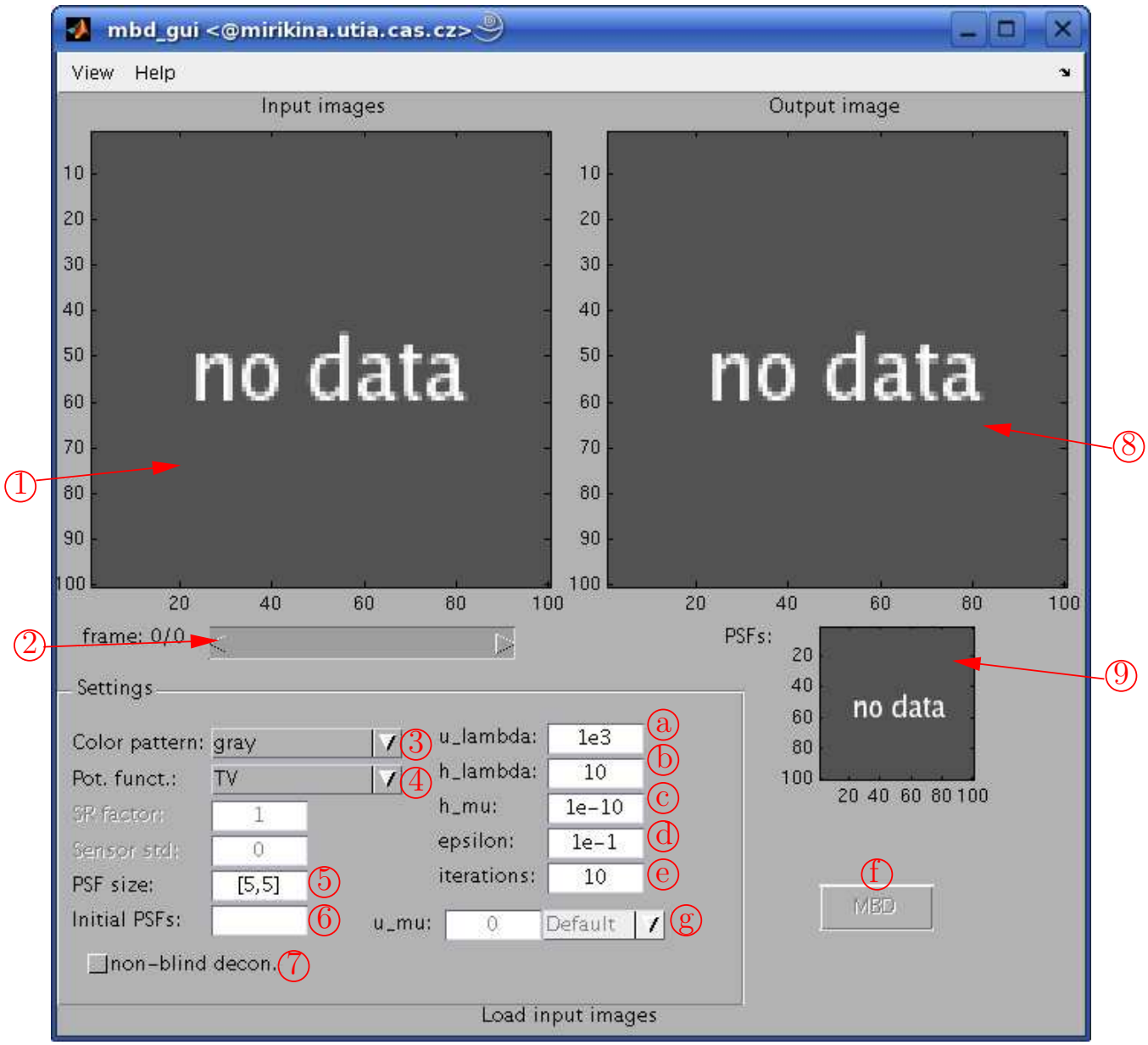


Figure 1: Graphical User Interface

- *load from files* - to load images from image files. The user can (and should) select more than one file. Most of the common image formats (png, tif, jpg, ppm) are supported.
- *info* - to get basic information about the displayed image.

The scrollbar (see ② in Fig. 1) below the plot is for browsing through the LR image.

5.2 Input Parameters

The *Settings* group at the left bottom (Fig. 1) contains all the input parameters necessary for correct operation of MBD.

Color pattern: Specifies the format of input images:

- *gray* - gray-scale images; If loaded from the Matlab workspace the cell array must contain images stored as 2D matrices.

- *color (3D matrix)* - standard color images stored as 3D matrices.
- *[bg;gr]* - images stored with Bayer pattern $\left(\begin{smallmatrix} B & G \\ G & R \end{smallmatrix}\right)$. Practically all the digital cameras use color filter arrays and many of them can store the data in raw format. This pattern is for example used in Olympus digital cameras. We have two green channels with the predetermined shift of $(0.5, 0.5)$ pixels and the MBD tool can take advantage of this fact to improve results.

  Note: If the Bayer pattern is selected, the MBD tool will perform not just blind deconvolution but also demosaicing, i.e., it will increase resolution by factor of 2.

Pot. funct.: Potential function $\phi(s)$ in (4): *Tichonov*, *TV*, *HyperSurface* or *CTV*. Tichonov regularization tends to produce over-smoothed results but can be appropriate for images lacking strong edges. Total variation (TV) is probably the best choice as it preserves edges. However, an unwanted patch-like pattern can occur especially for very noisy images. For color images, one can also use the CTV (color TV) option which is a straightforward extension of TV into color that naturally binds the RGB channels. It is defined as $\mathrm{CTV}(r, g, b) = \sqrt{|\nabla r|^2 + |\nabla g|^2 + |\nabla b|^2}$, where r,g and b are the RGB channels of the image.

PSF size: Estimated maximum size of the unknown blurs in the format [<height>,<width>]. The blur size should be large enough to accommodate the expected image (volatile) blur and/or misalignment of the LR images. Large values negatively affect the computational speed. In general, squares up to size 15 are still acceptable.

In reference to the mathematical discussion in Section 2, this option corresponds to the estimated size of the volatile blurs $\hat{h}_k$ in (3).

Initial PSFs: Variable in the Matlab workspace that contains the initial estimates of PSFs. The variable must be a cell array with the same number of elements as the number of input images. Each cell must contain a matrix, which is the initial PSF for the corresponding input image. If a valid cell array is used, PSF size is obsolete, since the size will be determined from the cell array. If this field remains empty or an invalid variable is selected, the PSFs will be initialized as delta functions with the support specified in PSF size.

non-blind deconvolution: By selecting this option, the algorithm will switch to the non-blind mode, i.e., no estimation of blurs. Initial PSFs must be specified and deconvolution of a single image is possible.

u_lambda: Inverse weight of the image smoothness term. It is inversely proportional to the level of noise in the images. Recommended values are, for example, for SNR=40dB 1e4, for SNR=20dB 1e2, etc.

In reference to the mathematical discussion in Section 2, this weight corresponds to $1/\lambda_o$ in (3).

h_lambda: Weight of the blur consistency term. We recommend values roughly 10× smaller than *u_lambda* and in the case of SR factor 1 (no superresolution) 100× smaller than *u_lambda*.

In reference to the mathematical discussion in Section 2, this weight corresponds to λ_h in (3).

h_mu: Weight of the blur smoothness term. In most of the case, the value can remain zero, since the blur reconstruction is controlled by the weight *h_lambda*.

In reference to the mathematical discussion in Section 2, this weight corresponds to μ_h in (3).

epsilon: Relaxation parameter for the potential functions TV and CTV. The default value 0.1 is correct in most of the cases. By decreasing the value, the algorithm preserves more edges.

iterations: Number of iterations. By clicking the *Stop* button, the user can terminate the reconstruction process at anytime.

u_mu: Weight of the color correlation term. Not yet implemented. Instead for color images it is recommended to use pot. function CTV.

5.3 Running MBD

The MBD algorithm starts by click the *MBD* button (see ⓯ in Fig. 1). A new window appears that will monitor the iterative process. After the given number iteration, the results (estimated HR image and blurs) are sent to the main window. Clicking the *Stop* button, the process terminates immediately and the current reconstruction is displayed in the main window.

5.4 Output

The reconstructed image appears in the top-right plot and the estimated blurs in the bottom-right plot (see ⑧ and ⑨ in Fig. 1, respectively). The scrollbar (see ②) browses through the blurs and matches the blurs to the input images.

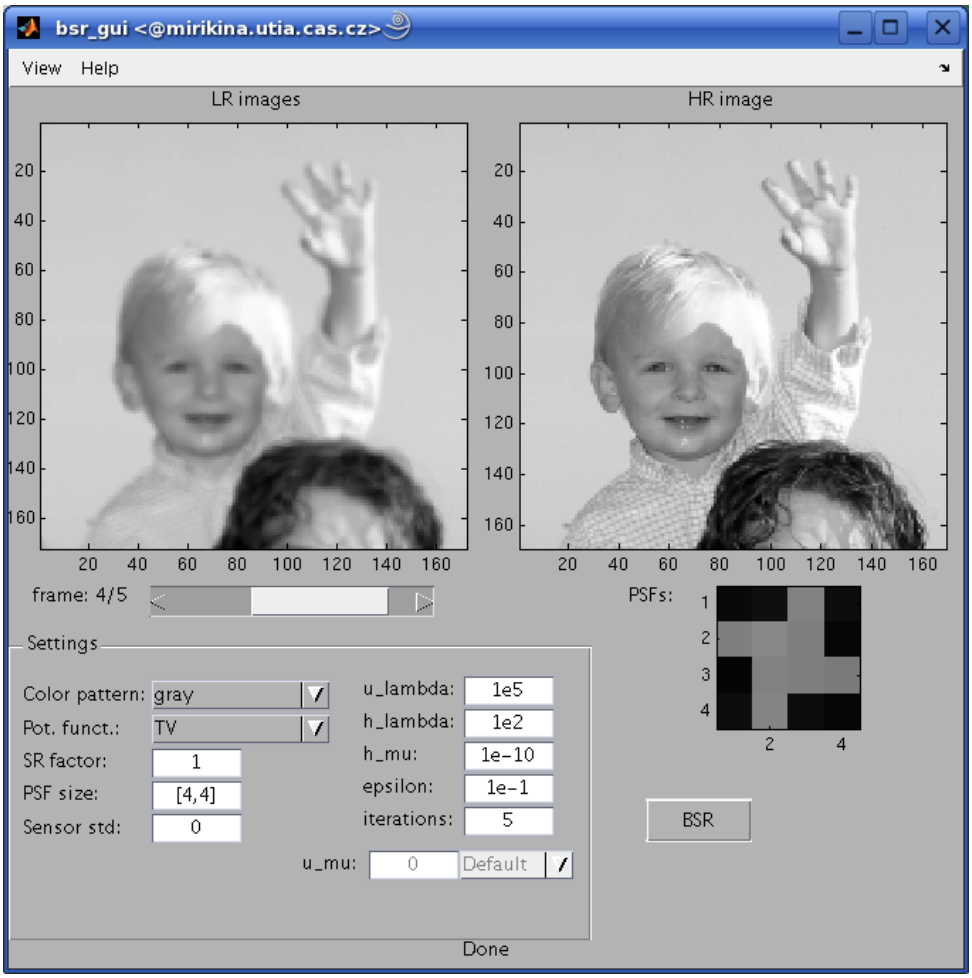
To save the reconstructed image or blurs, use context menus, which are activated whenever the user right-clicks on the corresponding plot.

6 Examples

Package `examples.zip` contains input images used in the following examples.

6.1 Blind deconvolution of artificially blurred images

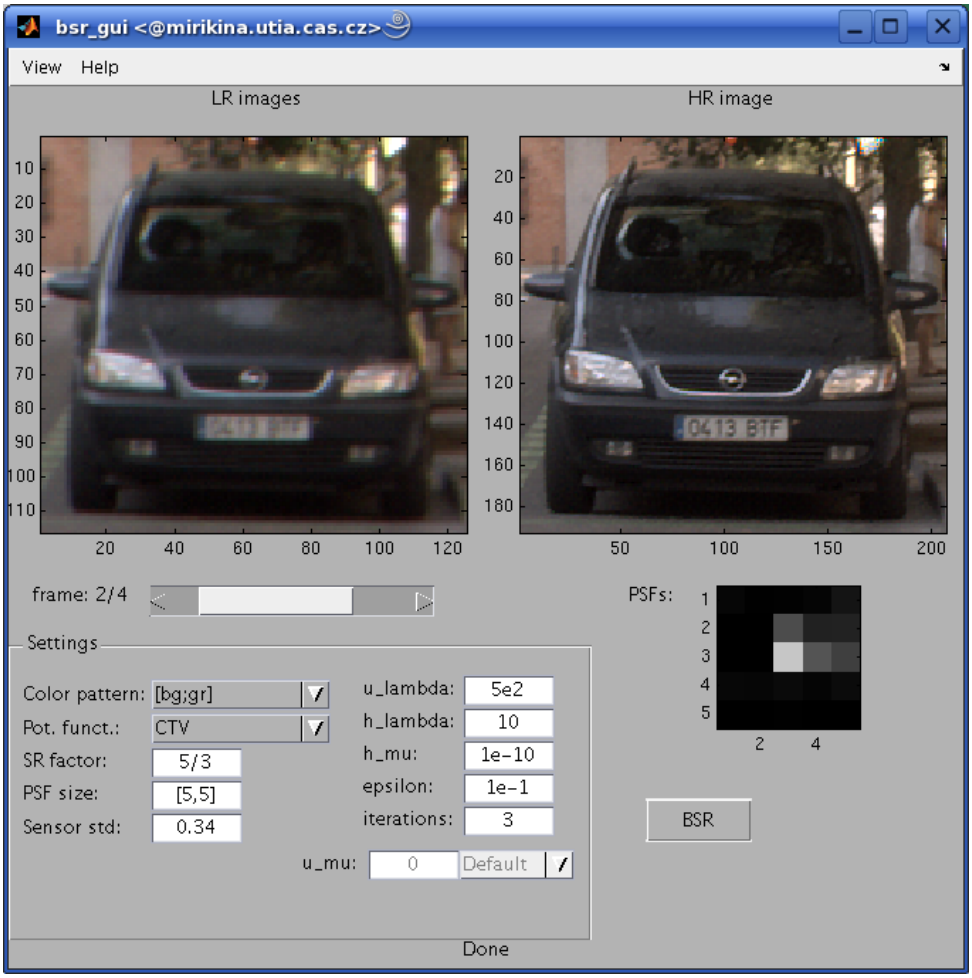
Set the MBD parameters according to the figure below:



Load at least two images images from the set `kiko1.png`, ..., `kiko5.png`. These experimental images were created by convolving the original image (`kiko_orig.png`) with 5 different synthetic blurs, no noise was added and no resolution decimation was performed. Run MBD and in couple of seconds (depending on the computer speed) you should obtain perfect reconstruction.

6.2 Blind deconvolution and demosaicing of color photos

Set the MBD parameters as follows:



Load all four images `opel1.png`, ..., `opel4.png`. These images cover a small section of four photos shot from hand with a 5 Mpixel digital camera (Olympus 5050). Photos were stored in raw format and therefore we can take advantage of the Bayer pattern. The MBD reconstruction takes about 1 minute (depending on the computer speed). Estimated blurs indicate that the hand was not perfectly still. The reconstructed HR image demonstrates the true power of the MBD tool by performing blind deconvolution and superresolution simultaneously.