MATLAB User Guide for Depth Reconstruction from Sparse Samples

Lee-Kang Liu, Stanley H. Chan and Truong Q. Nguyen

This MATLAB user guide presents the instructions of how to use the MATLAB functions accompanied with the paper [1].

TABLE I: Table of content.

I. Reconstruction functions

A. Notations and Problem Formulation

Referring to [1], we first discuss three reconstruction functions:

- x_{out}=ADMM_WT(*S*,*b*,*param*)
- x_{out}=ADMM_WT_CT(*S*,*b*,*param*)
- x_{out}=ADMM_outer(*S*,*b*)

The usage of these three functions are demonstrated in the following three MATLAB scripts:

- Demo_ADMM_WT.m
- Demo_ADMM_WT_CT.m
- Demo_Multiscale_ADMM_WT_CT.m.

The ADMM denotes the alternating direction method of multipliers, which is the proposed to solve the reconstruction problem in [1]. WT and CT respectively stand for wavelet and contourlet dictionaries. For three reconstruction functions, the input variables, including a sampling mask "*S*", an observation image "*b*", and a structure, *param*, are unified.

Mathematically, as presented in [1], the generalized model for depth reconstruction is defined as

$$\underset{\boldsymbol{x}}{\text{minimize}} \quad \frac{1}{2}\|\boldsymbol{S}\boldsymbol{x}-\boldsymbol{b}\|_2^2 + \sum_{\ell=1}^{L}\lambda_\ell\|\boldsymbol{W}_\ell\boldsymbol{\Phi}_\ell^T\boldsymbol{x}\|_1 + \beta\|\boldsymbol{x}\|_{TV}. \tag{1}$$

The notations of symbols are shown in Table II. We note that ADMM_WT is for single wavelet dictionary case ($L = 1$). We also note that ADMM_WT_CT and Multiscale_ADMM_WT_CT are for combined dictionary case ($L = 2$). By default, the regularization parameters are set to be $\lambda_1 = 4 \times 10^{-5}$, $\lambda_2 = 2 \times 10^{-4}$ and $\beta = 2 \times 10^{-3}$. The discussion on selecting parameters is presented in [2].

We note that the representations of input variables "*S*" and "*b*" in MATLAB implementation are not the same as $\boldsymbol{S}$ and $\boldsymbol{b}$ in Table II. Mathematically, for simplicity purpose, the image is set to be a 1D signal. However, practically, image is typically represented in 2D and is directly processed. More specifically, the sampling matrix $\boldsymbol{S}$ is a $N \times N$ matrix with 0's at off-diagonal entries and 0's/1's at diagonal entries, whereas the sampling mask "*S*" is a $\sqrt{N} \times \sqrt{N}$ matrix with entries 0's/1's. The measurement vector $\boldsymbol{b}$ has size $N \times 1$, whereas the observation image "*b*" has size $\sqrt{N} \times \sqrt{N}$. As shown in Figure 1, sampling mask "*S*" with size 512×512 contains 1's and 0's; The measurement image "*b*" with size 512×512 contains nonzero values at sampled locations (i, j), where $S(i, j) = 1$.

L. Liu and T. Nguyen are with Department of Electrical and Computer Engineering, University of California at San Diego, La Jolla, CA 92093, USA. Emails: l7liu@ucsd.edu and tqn001@eng.ucsd.edu

S. Chan is with School of Electrical and Computer Engineering and Department of Statistics, Purdue University, West Lafayette, IN 47907, USA. Email: stanleychan@purdue.edu

Symbols	Dimensions	Descriptions
$\boldsymbol{S}$	$\mathbb{R}^{N\times N}$	Sampling matrix
$\boldsymbol{b}$	$\mathbb{R}^{N\times 1}$	Spatial measurements
$\boldsymbol{\Phi}_1$	$\mathbb{R}^{N\times N}$	Wavelet Dictionary
$\boldsymbol{W}_1$	$\mathbb{R}^{N\times N}$	Weighting matrix for wavelet coefficients
$\boldsymbol{\Phi}_2$	$\mathbb{R}^{N\times N}$	Contourlet Dictionary
$\boldsymbol{W}_2$	$\mathbb{R}^{N\times N}$	Weighting matrix for contourlet coefficients
λ_1	$\mathbb{R}$	Regularization parameter for wavelet sparsity
λ_2	$\mathbb{R}$	Regularization parameter for contourlet sparsity
β	$\mathbb{R}$	Regularization parameter for total variation

TABLE II: Notations for symbols in (1).

Sampling Mask "*S*" Observation Image "*b*" Reconstructed Image "x_{out}"

Fig. 1: Input and output data for all reconstruction functions.

B. Usage of Functions

After clarifying the differences between the mathematical models and practical implementations, we in this subsection discuss the usage of three reconstruction functions. For simplicity, in these demonstration code we let "*S*" to be a sampling mask with uniformly random samples. The argument "*sp*." defines the sampling rate, and the code for generating "*S*" is

```
1  % Define Sampling Rate (0¬1)
2  sp = 0.1;
3
4  % uniformly random sampling mask function
5  S        = (rand(rows, cols)≤sp);
6  b        = S.⋆x0;
```

For simplicity, the sampling masks in all demonstration MATLAB scripts are generated by the sampling function mentioned above, and the generation of measurement image "*b*" is shown in line 6, where "x_0" is the ground truth depth image.

1) Using x_{out}*=ADMM_WT(*S,b,param*):* We now discuss the usage of the function x_{out}=ADMM_WT(*S*,*b*,*param*). As mentioned above, input arguments "*S*" and "*b*" are provided. The third argument "*param*" is a structure containing parameter settings for ADMM algorithm and for wavelet dictionary, and its default settings are presented in "Demo_ADMM_WT.m." An example code of its typical settings are

```
1  % Initialize parameters
2      param.wname            = 'db2';       % types of wavelet function
3      param.wlevel           = 2;           % number of levels for wavelet transform
4      param.lambda1          = 4e-5;        % regulization parameter for L1_Wavelet term
5      param.beta             = 2e-3;        % regulization parameter for total variation term
6      param.max_itr          = 400;         % maximum iteration for reconstruction
7      param.tol              = 1e-5;        % tolerance for stop condition
8      param.disp_relchg      = 1;           % display the relative change during process
9      param.disp_img         = 1;           % display image during process
```

Changes on parameters in "*param*" are acceptable. Lines 2-3 are parameter settings for wavelet dictionary, and the options for the settings can be found in MATLAB internal library. Lines 4-7 are settings of regularization parameters for ADMM algorithm, and we set them as defaults according to our experiments [2]. Lines 8-9 are flags for displaying statistics and intermediate depth image x. Setting flag to be 1 means that the target object will be displayed, and the target will not be displayed while flag is set to be 0.

An example of using this function is shown as follows, and users can refer to "Demo_ADMM_WT.m" for detailed information.

```
% ADMM using wavelet dictionary
    xout = ADMM_WT(S,b,param);
```

*2) Using x_{out}=ADMM_WT_CT(*S,b,param*):* The function x_{out}=ADMM_WT_CT(*S*,*b*,*param*) performs ADMM algorithm for combined dictionary, in which wavelet dictionary $\mathbf{\Phi}_1$ and contourlet dictionary $\mathbf{\Phi}_2$ are included. We note that the folder *ContourletSD* is the contourlet toolbox [3], which is required for the MATLAB script "Demo_ADMM_WT_CT.m." To ensure that "x_{out}=ADMM_WT_CT(*S*,*b*,*param*)" operates properly, checking the existence of "ContourletSD" folder and adding the code presented as follows are required.

```
% Set contourlet transform toolbox path
        addpath(genpath('ContourletSD\'));
```

After obtaining both sampling mask "*S*" and measurements "*b*", setting of parameters is the next. The default settings for the input argument "*param*" are shown as below. For distinguishing difference from aforementioned ADMM_WT function, we utilize "*WTCTparam*" as a symbol in the code.

```
1         WTCTparam.wname             = 'db2';  % types of wavelet function
2         WTCTparam.wlevel            = 2;      % number of levels for wavelet transform
3         WTCTparam.nlev_SD           = [5 6];  % directional filter partition numbers
4         WTCTparam.smooth_func       = @rcos;  % smooth function for Laplacian Pyramid
5         WTCTparam.Pyr_mode          = 2;      % redundancy setting for the transformed coefficients
6         WTCTparam.dfilt             = '9-7';  % 9-7 bior filter for directional filtering
7         WTCTparam.lambda1           = 4e-5;   % regularization parameter for L1_Wavelet term
8         WTCTparam.lambda2           = 2e-4;   % regularization parameter for L1_Contourlet term
9         WTCTparam.beta              = 2e-3;   % regularization parameter for total variation term
10        WTCTparam.max_itr           = 400;    % maximum iteration for reconstruction
11        WTCTparam.tol               = 1e-5;   % tolerance for stop condition
12        WTCTparam.disp_relchg       = 1;      % display the relative change during process
13        WTCTparam.disp_img          = 1;      % display image during process
```

Lines 1-2 are parameters for wavelet dictionary. Lines 3-6 are parameters for contourlet transform. Detailed parameter settings of contourlet transform can be found in [3]. Lines 7-11 are parameters for the ADMM algorithm for combined dictionary model. We note that the provided coefficients are default values. Lines 12-13 are flags for displaying statistics and intermediate depth image x. Setting flags to be 1 means that the target object will be displayed, and the target will not be displayed while flag is set to be 0.

An example of using this function is shown as follows, and users can refer to "Demo_ADMM_WT_CT.m" for detailed information.

```
 % main algorithm for dense disparity reconstruction
        xout = ADMM_WT_CT(S,b,WTCTparam);
```

*3) Using x_{out}=ADMM_outer(*S,b*):* The function x_{out}=ADMM_outer(*S*,*b*) is an implementation of multiscale ADMM algorithm proposed in [1]. The multiscale ADMM consists mainly of ADMM_outer(*S*,*b*) and ADMM_inner(*S*,*b*). The latter is typically called in the inner loop for dealing a designate image scale because of the multiscale scheme. More specifically, "ADMM_inner" is a ADMM algorithm for combined dictionary, and "ADMM_outer" is a realization of multiscale scheme for ADMM algorithm. As discussed in Section I-B, the sampling mask "*S*" and measurements "*b*" are obtained by an uniformly random sampling function. Then, the usage of the function is as follows.

```
  % multiscale ADMM reconstruction algorithm
   xout = ADMM_outer(S,b);
```

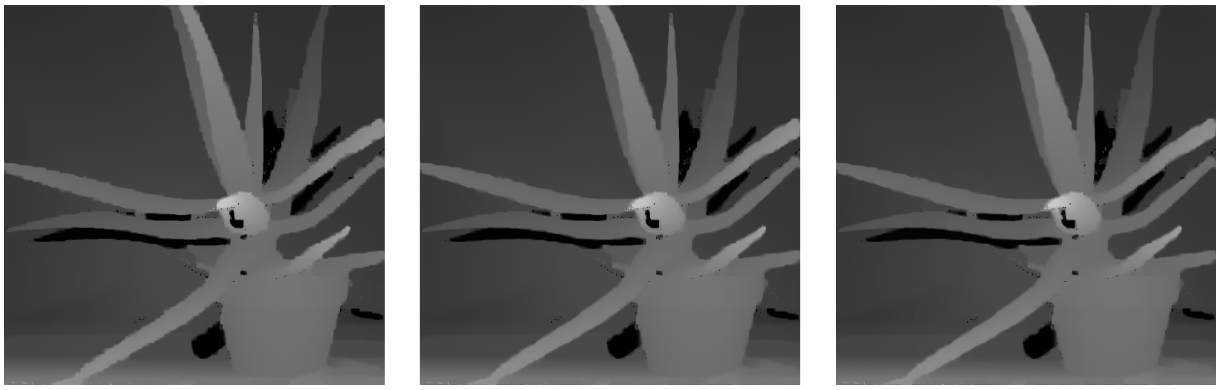
We note that the parameter settings are by default coded in the "ADMM_outer(*S*, *b*)" function. In the following paragraph, we briefly describe the meaning of parameters that are available for tuning, and they are attached as follow.

```
1          param.wname            = 'db2';    % types of wavelet function
2          param.wlevel           = 2;        % number of levels for wavelet transform
3          param.nlev_SD          = [5 6];    % directional filter partition numbers
4          param.smooth_func      = @rcos;    % smooth function for Laplacian Pyramid
5          param.Pyr_mode         = 2;        % redundancy setting for the transformed coefficients
6          param.dfilt            = '9-7';    % 9-7 bior filter for directional filtering
7          param.lambda1          = 4e-5;     % regulization parameter for L1_Wavelet term
8          param.lambda2          = 2e-4;     % regulization parameter for L1_Contourlet term
9          param.beta             = 2e-3;     % regulization parameter for total variation term
10         param.max_itr          = 60;       % maximum iteration for reconstruction
11         param.tol              = 1e-5;     % tolerance for stop condition
12         param.disp_relchg      = 1;        % display the relative change during process
13         param.disp_img         = 1;        % display image during process
14          Q                     = 2;        % scale setting
```

Lines 1-2 are parameters for wavelet dictionary. Lines 3-6 are parameters for contourlet transform. Lines 7-11 are parameters for the ADMM algorithm for combined dictionary model. Lines 12-13 are flags for displaying statistics and intermediate depth image x. Setting flags to be 1 means that the target object will be displayed, and the target will not be displayed while flag is set to be 0. We also note that these parameter settings are for each designate scale of ADMM algorithm, and are fed to the function "ADMM_inner." Line 14 defines the number of scales utilized in the multiscale ADMM algorithm. For the usage of this multiscale ADMM algorithm, users can refer to the MATLAB script, "Demo_Multiscale_ADMM_WT_CT.m."

C. *Examples*

Three reconstruction functions are presented in this subsection. Figure 2 shows reconstructed "Aloe" disparity map while fixing the sampling rate to be 10% and feeding same sampling map to three reconstruction functions.



ADMM_WT / 28.3056 (dB) ADMM_WT_CT / 28.5114 (dB) ADMM_outer / 28.5889 (dB)

Fig. 2: Results of reconstructed dense disparity maps using three reconstruction functions. Note that we feed the same sampling map with 10% uniformly random samples to three functions.

II. Sampling functions

In this section, we present the usage of sampling functions discussed in [1]. We first show the oracle random sampling (ORS) scheme, which is firstly proposed in [4]. We then show the extension of ORS scheme using principal component analysis (PCA). Finally, we show the proposed 2-Stage sampling scheme which utilizes aforementioned ORS scheme with PCA for practical sampling problem (i.e., ground truth depth data is unknown), and we demonstrate the application for the real depth data reconstruction from sparse samples.

A. *Notations*

According to [1], we first discuss two sampling functions:

- S = Oracle_Random_Sampling(x_0, *sp*)
- S = Oracle_Random_Sampling_with_PCA(x_0, *sp*, S_{pilot})

We note that for former function, ground truth data is typically given, but is unknown for the latter. In the demonstration code, the usage of these two functions are presented in the following two scripts:

- Demo_Oracle_Random_Sampling.m
- Demo_Oracle_Random_Sampling_with_PCA.m

Moreover, we further show the proposed two-stage sampling scheme in the script:

- Demo_Two_Stage_sampling.m

Note that the two stage sampling scheme contains both sampling and reconstruction algorithms presented previously. We finally apply the proposed framework to real application for dense disparity reconstruction from sparse samples, and we demonstrate the work in the MATLAB script:

- Demo_Application_for_Dense_Disparity_Estimation.m

Now, we would like to discuss the mathematical model for the sampling method - Oracle Random Sampling (ORS). Mathematically, the ORS scheme is firstly proposed in [4], and the extension is discussed in [1]. The authors claim that the sampling map is determined by utilizing the optimal probability values obtained from the optimization problem

$$
\begin{aligned}
(P): \qquad & \underset{\boldsymbol{p}}{\text{minimize}} && \frac{1}{N}\sum_{j=1}^{N}\frac{a_j^2}{p_j} \\
& \text{subject to} && \frac{1}{N}\sum_{j=1}^{N} p_j = \xi, \text{ and } 0 \le p_j \le 1.
\end{aligned}
$$

The variable p_j denotes the Bernoulli coin flip probability at jth pixel location, and the variable ξ is the target sampling ratio. The variable a_j denotes the magnitude of the gradient or sum of absolute PCA coefficients at jth pixel location. More specifically, the variable a_j is defined as the gradient value of the jth pixel in the function

- *S* = Oracle_Random_Sampling(x_0, *sp*).

The variable a_j is defined as the sum of absolute PCA coefficients of the jth pixel in the function

- *S* = Oracle_Random_Sampling_with_PCA(x_0, *sp*, S_{pilot}).

Finally, we note that in all demonstration MATLAB scripts, we consistently utilize multiscale ADMM reconstruction function (i.e., ADMM_outer), as we discuss various sampling functions. We also note that users are free to apply different combinations of sampling functions and reconstruction algorithms.

B. Oracle Sampling

1) Using "S = Oracle_Random_Sampling(x_0, sp)": Referring to the function inputs, the variable "*sp*" stands for the sampling ratio, and "x_0" denotes the reference image. The image could be either a gray scale view image, a depth image, or a disparity map. First, the function estimates the gradients of the input image"x_0" and defines them as the coefficient $a_j, \forall j$. Then, the sampling map "*S*" is determined as each pixel having the sampling probability p_j. A code segment of the sampling function is attached as follows.

```
% sampling ratio
sp        = 0.1;
% Oracle Random Sampling with Magnitude of gradients
S         = Oracle_Random_Sampling( x0, sp);
b         = S.*x0;
```

2) Using "S = Oracle_Random_Sampling_with_PCA(x_0, sp, S_{pilot})": Similar to the previous function, the variable "*sp*" stands for the sampling ratio, and "x_0" denotes the reference image. The image could be either a gray scale view image, a depth image, or a disparity map. First, the function estimates the PCA basis by constructing a collection of patches $\{\boldsymbol{y}_j \in \mathbb{R}^d\}_{j=1}^N$ centered at jth pixel and by estimating the eigenvectors of the covariance matrix $\boldsymbol{C} = \frac{1}{N}\sum_{j=1}^{N}\boldsymbol{y}_j\boldsymbol{y}_j^T$. Since the covariance matrix is symmetric and positive semi-definite, $\boldsymbol{C}$ can be defined as

$$\boldsymbol{C} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^T.$$

Then, the sum of absolute of PCA coefficients for each pixel is estimated by

$$a_j = \sum_{i=2}^{d'} |\langle \boldsymbol{u}_i, \boldsymbol{y}_j \rangle|,$$

where $\boldsymbol{u}_i$ denotes the ith column vector of the matrix $\boldsymbol{U}$, and d' is a scalar value with $d' \le d$. Then, the sampling map "*S*" is determined as each pixel having the sampling probability p_j. A code segment of the sampling function is attached as follows.

```
% Oracle Random Sampling with Magnitude of gradients
S_pilot   = zeros(rows,cols);
S         = Oracle_Random_Sampling_with_PCA( x0, sp, S_pilot);
b         = S.*x0;
```

Figure 3 shows examples of different sampling patterns while sampling rate is set to be 10%. We note that in line 2, the variable S_{pilot} is set to be zero as there is no additional pilot information. The further usage of the pilot signal will be discuss in the next subsection.

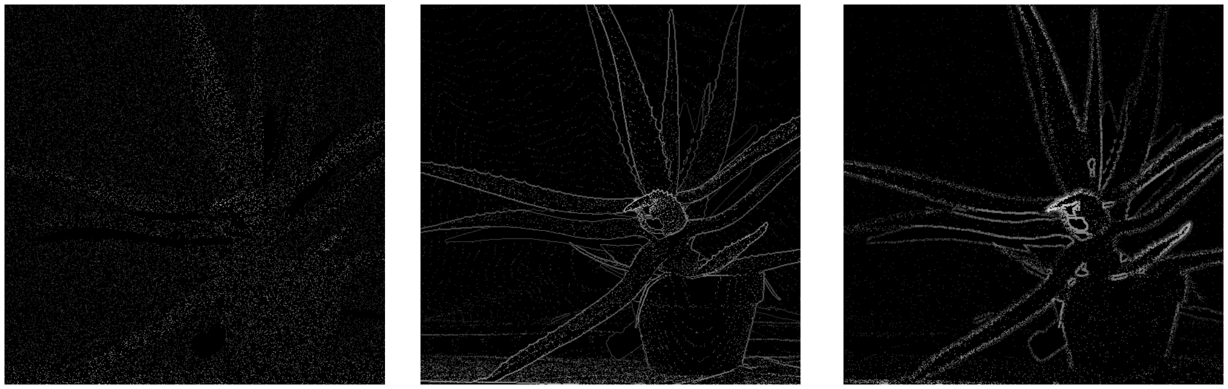


Fig. 3: Examples of sampling maps. We set the sampling rate to be 10% for each function.

C. Practical Sampling (2-Stage Sampling)

In this subsection, we demonstrate the 2-Stage sampling algorithm proposed in [1]. The MATLAB script describing the propose algorithm is presented in

- DEMO_Two_Stage_Sampling.m

In the 1st stage, the uniformly random sampling scheme is applied to obtain sampling map S_{pilot}, then the pilot signal x_{out_pilot} is estimated. The code of 1st stage is attached as follows.

```
    % 1st Stage
    S_pilot      = (rand(rows,cols)≤sp*0.5);
    b            = S_pilot.*x0;
    xout_pilot   = ADMM_outer(S_pilot,b);
```

In ths 2nd stage, the pilot signals S_{pilot} and x_{out_pilot} are utilized as input variables to the ORS with PCA function. Then, the sampling map, *S*, at the 2nd stage is obtained. The code of 2nd stage is attached as follows.

```
    % 2nd Stage
    S            = Oracle_Random_Sampling_with_PCA( xout_pilot, 0.5*sp, S_pilot);
    b            = S.*x0;
    xout         = ADMM_outer(S,b);
```

We note that in the 2-Stage sampling demonstration code, 50% of the sampling budgets is utilized in the 1st stage and the rest 50% is utilized in the 2nd stage. Since we assume the ground truth signal x_0 is unknown, we utilize uniformly random sampling for obtaining the pilot sampling map. Once a pilot signal x_{out_pilot} is acquired, we then apply ORS with PCA function for constructing the optimal sampling map. Figure 4 shows an example with intermediate results of 2-Stage algorithm.

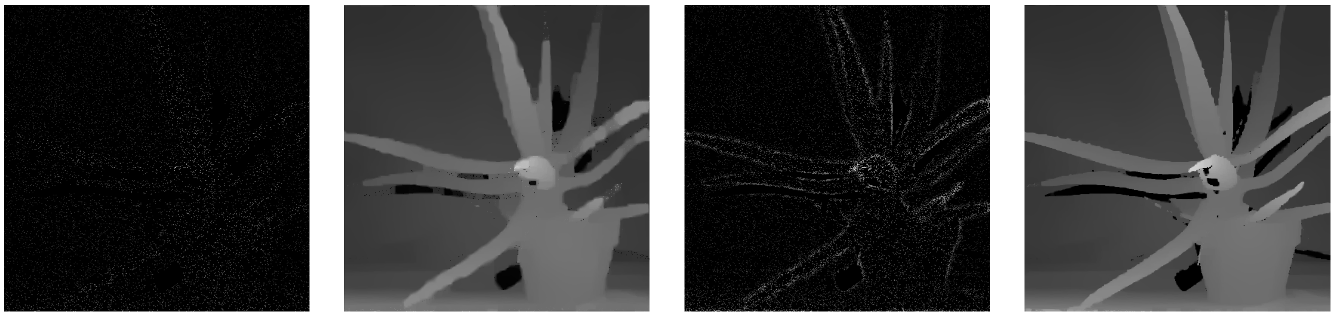

1st Stage / $\zeta = 0.0494$
x_{out_pilot} / 25.0999 (dB)
2nd Stage / $\zeta = 0.0997$
x_{out} / 31.5065 (dB)

Fig. 4: Example of 2-Stage algorithm .

In disparity estimation application, stereo images are used to estimate disparity values, and these view images can be used to infer optimal sampling locations by the ORS with PCA function. Therefore, we show how to practically apply the proposed algorithms to reconstruct dense disparity maps from sparse samples as stereo images are given. The demonstration for the real application is presented in the MATLAB script:

- Demo_Application_for_Dense_Disparity_Estimation.m

In this MATLAB script, as one view image, y_{0_bflt}, is known, we set it as the initial pilot signal for the ORS with PCA function, and we use only 50% of the sampling budgets in the first stage. Then, the reconstruction algorithm is applied to obtain the pilot signal, x_{out_pilot}, and these pilot signals S_{pilot} and x_{out_pilot} are further set as input variables of the ORS with PCA function to estimate additional 50% of sampling budgets in the 2nd stage. Finally, the dense disparity map is estimated using the proposed ADMM algorithm. The code for reconstructing dense depth data from 10% measurements using 2-Stage algorithm is present as follows:

```
    % 1st Stage
    S_pilot      = zeros(rows,cols);
    S_pilot      = Oracle_Random_Sampling_with_PCA( y0_bflt, 0.5*sp, S_pilot);
    b            = S_pilot.*x0;
    xout_pilot   = ADMM_outer(S_pilot,b);

    % 2nd Stage
    S            = Oracle_Random_Sampling_with_PCA( xout_pilot, 0.5*sp, S_pilot);
    b            = S.*x0;
    xout         = ADMM_outer(S,b);
```

Figure 5 and Figure 6 show the intermediate and final results of the demonstration code for real dense disparity estimation application. Figure 6 also shows a comparison between densely estimated disparity map using [5] and the proposed algorithm using 10% of measurements of [5].

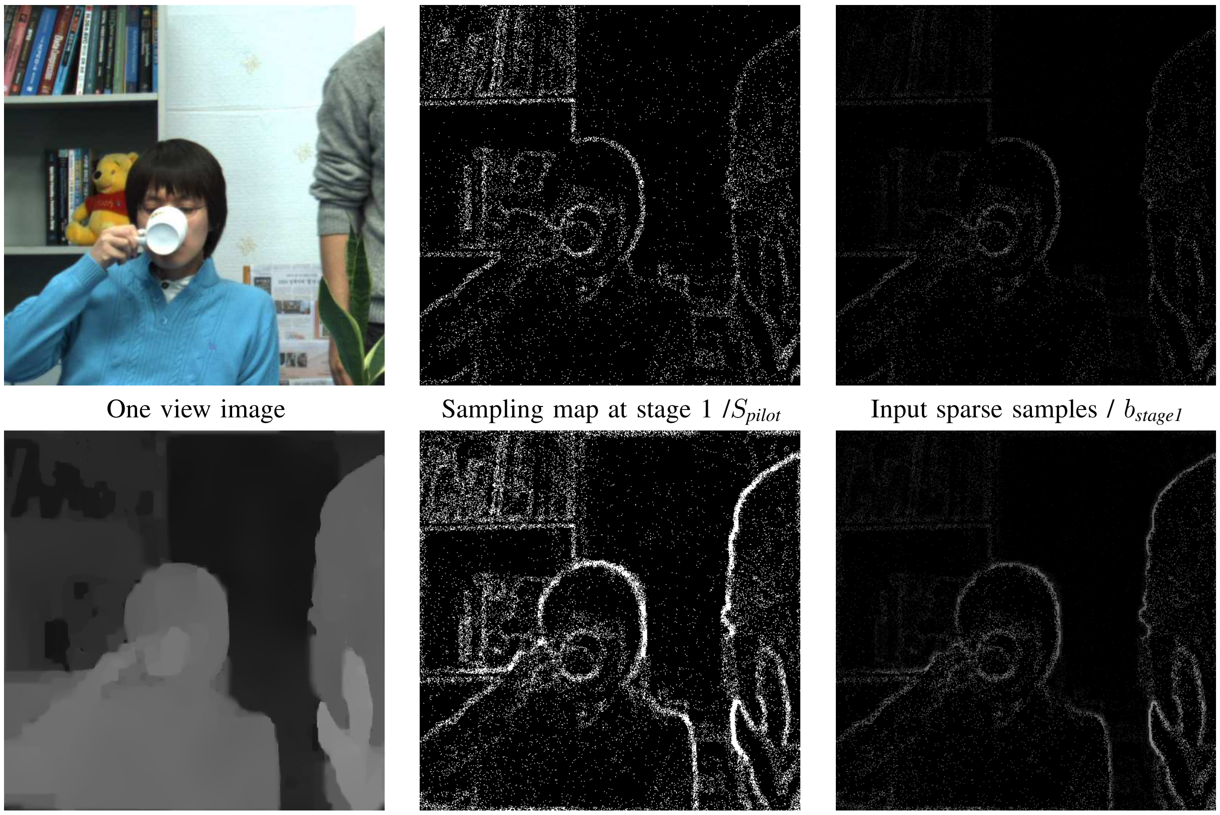

One view image
Sampling map at stage 1 /S_{pilot}
Input sparse samples / b_{stage1}
Pilot Signal / x_{out_pilot}
Sampling map at stage 2 / S
Input sparse samples / b_{stage2}

Fig. 5: Intermediate results of the demonstration code for dense disparity reconstruction.

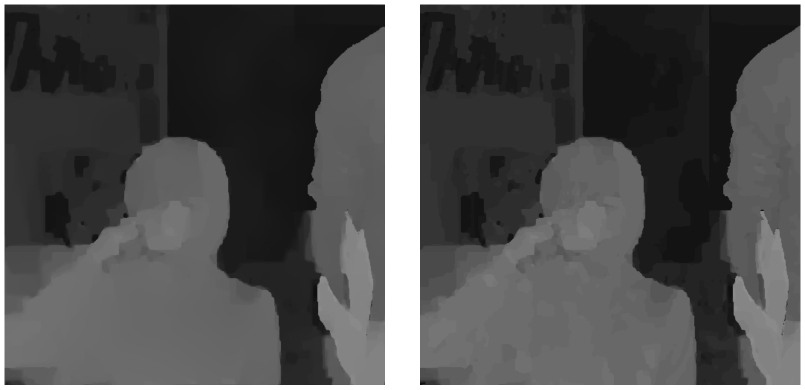


Fig. 6: Final results of the demonstration code for dense disparity reconstruction.

III. Compatibility

This toolbox has been tested under the following machine, and it works properly.

- MATLAB 2010a, 64-bit Windows 7 Professional
- MATLAB 2012b, 64-bit Windows 7 Professional
- MATLAB 2014a, 64-bit Windows 7 Professional

References

[1] L.-K Liu, S.H. Chan, and T.Q. Nguyen, "Depth reconstruction from sparse samples: Representation, algorithm, and sampling," *IEEE Trans. on Image Process.*, vol. 24, no. 6, pp. 1983–1996, Jun. 2015.

[2] L.-K Liu, S.H. Chan, and T.Q. Nguyen, "Depth reconstruction from sparse samples: Representation, algorithm, and sampling (supplementary material)," Available online at http://arxiv.org/abs/1407.3840.

[3] M.N. Do and M. Vetterli, "The contourlet transform: An efficient directional multiresolution image representation," *IEEE Trans. Image Process.*, vol. 14, no. 12, pp. 2091–2106, Dec. 2005.

[4] S.H. Chan, T. Zickler, and Y.M. Lu, "Monte Carlo non-local means: Random sampling for large-scale image filtering," *IEEE Trans. Image Process.*, vol. 23, no. 8, pp. 3711–3725, Aug. 2014.

[5] Z. Lee, J. Juang, and T.Q. Nguyen, "Local disparity estimation with three-moded cross census and advanced support weight," *IEEE Trans. Multimedia*, vol. 15, no. 8, pp. 1855–1864, Dec. 2013.