IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received July 17, 2021, accepted August 2, 2021, date of publication August 3, 2021, date of current version August 11, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3102404

Two Layer Tensor Form Convolutional Sparse Coding for Stereo Matching

CHUNBO CHENG[1,2] AND WENJING CUI[2]
[1]School of Mathematics and Statistics, Huazhong University of Science and Technology, Wuhan 430074, China
[2]School of Mathematics and Physics, Hubei Polytechnic University, Huangshi, Hubei 435000, China

Corresponding author: Wenjing Cui (wjcui@hbpu.edu.cn)

This work was supported by Hubei Polytechnic University under Grant 21xjz25R.

ABSTRACT Stereo matching is an important research topic in the field of computer vision. It recovers depth information from a pair of color images. Unfortunately, converting multi-dimensional (more than two-dimensional) data into two-dimensional data, such formulations ignore the spatial structure of multi-dimensional images/data. Tensors can be used to describe high-dimensional data structure, which can retain the hidden structure of data, but cannot obtain the deep features that helps to improve the performance of the algorithm. Therefore, it is very important to establish a deep tensor model. In this paper, we propose a two layer tensor form convolutional sparse coding model, which can automatically learn the deep convolutional kernel. Based on the learned two layer convolutional kernels, a two-layer dictionary learning model is established. Then, a new weighted matching cost method is constructed, which combines shallow and deep features. The experimental results on the Middlebury benchmark v2 and Middlebury benchmark v3 show that the proposed two layer tensor convolutional sparse coding is effective for stereo matching.

INDEX TERMS Stereo matching, high-order tensor, convolutional sparse coding, deep learning, dictionary learning.

I. INTRODUCTION

Stereo matching, also known as disparity mapping, is one of the key techniques in stereo vision research area. The core idea is to find all corresponding pixels in a stereo image pair. Stereo matching cost plays an important role in establishing visual matching relationship. Usually, the accuracy of the stereo matching method depends on the accuracy of the stereo matching cost. Commonly used stereo matching costs can be divided into two large categories, including pixel-wise and window-based matching costs. Pixel-wise matching costs include the absolute difference (AD) and truncated absolute difference (TAD) [1]. Window-based matching costs include follows: sum of squared difference (SSD) [2], [3], sum of absolute difference (SAD) [4], normalized cross correlation (NCC), zero mean normalized cross correlation (ZNCC) [5], census (Cen) [6], [7], etc.

To get better matching results, combinations or variations of the above window-based methods are proposed in the literature, such as combination of census and gradient based measures (Cen+G) [8], combination of sum of absolute difference and gradient-based measures (SAD+G) [9], and combination of absolute differences and census measures (AD+Cen) [10]. These stereo matching costs are used by some state-of-the-art stereo matching methods, and have been demonstrated to have very good performance in image regions with smooth terrain. However, they cannot handle regions that are lack of information, such as poorly texture regions, exposure variations, occlusion, depth discontinuities, etc.

Recently, stereo matching methods based on deep learning [11]–[22] had made significant progress in the disparity estimation of stereo images, in which the most prominent and effective deep learning method is the deep convolutional neural network (CNN). Due to the powerful representation capability of deep CNN in poorly texture regions and repetitive texture regions, it has been employed to improve the accuracy of stereo matching. Žbontar and LeCun [21] first introduced CNN to measure the similarity [23]–[25] between two image patches, which used the matching probability between two image patches as the stereo matching cost. Subsequently, a large number of stereo matching methods based on CNN [26]–[29] were proposed. These CNN methods achieved better performance than conventional methods on

The associate editor coordinating the review of this manuscript and approving it for publication was Long Xu.

challenging public benchmark data sets (such as KITTI [30] and Middlebury 2014 [2]).

From the perspective of deep learning, CNN can automatically extract valuable and deep features. Compared with shallow features, high-level features have the capability to represent more abstract and complex structure information, thus, having a stronger robustness and invariance towards local changes of the image. For CNN, deep structure and convolution kernel are its two important parts. Deep structure hierarchically extracts deep features, and convolutional architecture learns spatial structure of images [31]. Unfortunately, these networks train the convolution kernels in an end-to-end manner, which are prone to overfitting. Therefore, many methods [31]–[33] proposed to replace the process of learning these convolutional kernels with the traditional matrix or tensor decomposition method. For example, unsupervised CNN model [34] learns convolutional kernels by employing convolutional sparse coding (CSC) instead of back propagation.

Inspired by the above literature, this paper proposes a two layer tensor convolution sparse coding used in stereo matching, it can be in the form of a kind of unsupervised feature extraction, which can automatically learn the deep convolutional kernels. Based on the learned two layer convolutional kernels, a two-layer dictionary learning model is established. Then, a new weighted matching cost method is constructed, which combines shallow and deep features, and higher matching accuracy can be obtained. Unfortunately, converting multi-dimensional (more than two-dimensional) data into two-dimensional data, such formulations ignore the spatial structure of multi-dimensional images/data. It is still a challenge to learn multi-dimensional dictionaries and their sparse coding or features from multi-dimensional data (e.g. collections of colored images, colored videos, hyperspectral images, or in general videos represented by features of multiple sources) to reconstruct multi-dimensional data. Joint treatment of features [35] acquired from multiple sources (e.g. spectral images, HOG features, and colors) often leads to better performance when compared to treating them separately due to the high order correlations between features. Although the high-order tensor CSC has been studied in the literature [36], it can only extract shallow features and maintain the high order correlations between features, but cannot extract deep features. Compared with shallow features, deep features have the capability to represent more abstract and complex structure information, thus, having a stronger robustness and invariance towards local changes of the image. To this end, this paper proposes a deep high order tensor CSC model.

For this paper, the main contributions are as follows:

(1) A two layer tensor convolutional sparse coding model is proposed, which can automatically learn the deep convolution kernels. The deep feature is helpful to improve the performance of the algorithm, so a deep CSC model is constructed. In the experiment, it is found that the more network layers, the higher the time cost, and the accuracy is not always increasing. To trade off the accuracy and time cost, we constructed a two-layer CSC model.

(2) Based on the learned two layer convolution kernels, a two-layer dictionary learning model is established. Then, the sparse representation coefficients under the first-layer dictionary and the second-layer dictionary are solved, respectively.

(3) A new weighted matching cost method is constructed, which combines shallow and deep features, and higher matching accuracy can be obtained.

The rest of this paper is organized as follows. Section II introduces the preliminary knowledge of high-order tensor convolutional sparse coding. Section III presents two layer tensor convolutional sparse coding for stereo matching. Afterward, experimental results are shown in Section IV. Section V concludes and discusses the whole work.

II. PRELIMINARY

A. NOTATIONS

Let us give some notations used in this paper. The Frobenius squared norm of an N^{th}-order tensor $\mathcal{T} \in \mathbb{R}^{J_1 \times J_2 \times \cdots \times J_N}$ is $\|\mathcal{T}\|_F^2 = \sum_{i_1,i_2,\ldots,i_N} \mathcal{T}(i_1, i_2, \ldots, i_N)^2$, while its $\ell_{1,\ldots,1}$ norm is $\|\mathcal{T}\|_{1,\ldots,1} = \sum_{i_1,i_2,\ldots,i_N} |\mathcal{T}(i_1, i_2, \ldots, i_N)|$. The inner product between two tensors of the same size is $\langle \mathcal{T}_1, \mathcal{T}_2 \rangle = \sum_{i_1,i_2,\ldots,i_N} \mathcal{T}_1(i_1, i_2, \ldots, i_N)\mathcal{T}_2(i_1, i_2, \ldots, i_N)$. The unfold operation can be done along any of its dimensions. A k^{th}-mode fold/unfold of a tensor is defined as $unfold_k(\mathcal{T}) = \mathbf{T}_{(k)} \in \mathbb{R}^{J_k \times (J_1 \cdots J_{k-1} J_{k+1} \cdots J_N)}$, and $fold_k(\mathbf{T}_{(k)}) = \mathcal{T}$. A tensor-matrix product depends on the dimension along which the product is conducted. We denote $\mathbf{T}^{(k)} = \mathcal{T}(:, :, k)$ (using MATLAB notation) as the k^{th} frontal slice. $circ(\mathcal{T})$ is a block circulant matrix of size $n_1 n_3 \times n_2 n_3$, which essentially generates circular shifts out of the blocks of the frontal slices of $\mathcal{T}$. According to [36], the $\circledast_{HT}$ operator is derived by deriving the following three definitions.

Definition 1 (High Order t-Products [36]):

$$\mathcal{A} \circledast_{HT} \mathcal{B}_1 = fold_{HT}(circ_{HT}(\mathcal{A}) MatVec_{HT}(\mathcal{B}_1)). \quad (1)$$

The operators $circ_{HT}(\cdot)$ and $MatVec_{HT}(\cdot)$ apply $circ(\cdot)$ and $MatVec(\cdot)$ recursively on all the dimensions in the order $(3, 4, \ldots, d)$ as follows:

Definition 2 (High Order Recursive MatVec(·) [36]):

$$MatVec_{HT}(\cdot) = MatVec_{(d)}(\cdots(MatVec_{(3)}(\cdot))). \quad (2)$$

Definition 3 (High Order Recursive circ(·) [36]):

$$circ_{HT}(\cdot) = CI_{(d)}(MatVec_{(d)}(\cdots(CI_{(3)}(MatVec_{(3)}(\cdot))))). \quad (3)$$

Simply, $fold_{HT}(\cdot)$: $\mathbb{R}^{(n_1 n_2 \cdots n_d) \times (K n_2 \cdots n_d)} \longrightarrow \mathbb{R}^{n_1 \times K \times \cdots \times n_d}$ refolds the matrix back into a tensor in the same order.

B. HIGH-ORDER TENSOR CONVOLUTIONAL SPARSE CODING

Convolutional sparse coding (CSC) has achieved great success as a reconstruction and a classification tool in the field of computer vision and machine learning. High-order tensor

convolutional sparse coding was first proposed by Bibi and Ghanem [36]. A high-order t-product operator is used to give a tensor CSC model, which may reveal the high-dimensional correlation between features/channels in the data. It can learn multi-dimensional dictionaries and their sparse coding or features from multi-dimensional data (e.g. collections of colored images, colored videos, hyperspectral images, or in general videos represented by features of multiple sources) to reconstruct multi-dimensional data. The high-order tensor convolutional sparse coding problem for arbitrary d-th order tensors can be formulated as:

$$\min_{\boldsymbol{D},\boldsymbol{X}} \frac{1}{2}\sum_{n=1}^{N} \|\boldsymbol{Y}_n - \boldsymbol{D} \circledast_{HT} \boldsymbol{X}_n\|_F^2 + \lambda \|\boldsymbol{X}_n\|_{\underbrace{1,\ldots,1}_{d+1}}$$
$$s.t.\ \|\boldsymbol{D}_m\|_F^2 \le 1 \quad \forall m = 1,\ldots,M, \tag{4}$$

where $\circledast_{HT}$ represents a higher-order tensor convolution operator, which is defined in literature [36]. The n-th multidimensional training image is $\boldsymbol{Y}_n \in \mathbb{R}^{n_1\times 1\times n_2\times\cdots\times n_d}$. The filter $\boldsymbol{D}_m \in \mathbb{R}^{n_1\times 1\times n_2\times\cdots\times n_d}$, and the M filters $\{\boldsymbol{D}_m\}_{m=1}^{M}$ are concatenated along the second dimension forming the dictionary $\boldsymbol{D} \in \mathbb{R}^{n_1\times M\times n_2\times\cdots\times n_d}$, and the n-th sparse code is $\boldsymbol{X}_n \in \mathbb{R}^{M\times 1\times n_2\times\cdots\times n_d}$. The Frobenius squared norm of an N-th order tensor $\boldsymbol{Y} \in \mathbb{R}^{J_1\times J_2\times\cdots\times J_N}$ is $\|\boldsymbol{Y}\|_F^2 = \sum_{i_1,i_2,\cdots,i_N} \boldsymbol{Y}(i_1,i_2,\cdots,i_N)^2$, and $\ell_{1,\ldots,1}$ norm of $\boldsymbol{Y}$ is $\|\boldsymbol{Y}\|_{1,\ldots,1} = \sum_{i_1,i_2,\cdots,i_N} |\boldsymbol{Y}(i_1,i_2,\cdots,i_N)|$.

The traditional fixed point strategy [36] is used to solve the high-order tensor convolutional sparse coding in (4), where $\boldsymbol{D}$ and $\boldsymbol{X}$ can be solved alternately. Each subproblem is convex, so the alternating direction method of multipliers (ADMM) can be used to solve it.

III. TWO LAYER TENSOR CONVOLUTIONAL SPARSE CODING FOR STEREO MATCHING

In this section, we will introduce a two layer tensor convolutional sparse coding (CSC) model for stereo matching.

A. DEEP LAYER TENSOR CONVOLUTIONAL SPARSE CODING

The high-order tensor CSC can only extract shallow features, but cannot extract deep features. Compared with shallow features, deep features have the capability to represent more abstract and complex structure information, thus, having a stronger robustness and invariance towards local changes of the image. In this section, we build a two layer tensor CSC model, which can be used to extract deep features efficiently.

First, we build an l-layer tensor CSC model. Given a training set including N stereo image pairs, stereo images will be treated as arbitrary d-dimensional tensors. For example, a colored image $\boldsymbol{Y}_1$ can be represented by a 3^{rd}-order tensor $\boldsymbol{Y}_1 \in \mathbb{R}^{3\times n_2\times n_3}$, where n_2 and n_3 are the spatial dimensions. However, throughout the derivation of our formulation, we always allocate the second dimension for the concatenation of different images and that means all d-dimensional images are represented by a $(d+1)$-dimensional tensor. Thus, our example image is now $\vec{\boldsymbol{Y}}_1 \in \mathbb{R}^{3\times 1\times n_2\times n_3}$, and the set of images $\{\vec{\boldsymbol{Y}}_1, \vec{\boldsymbol{Y}}_2, \ldots, \vec{\boldsymbol{Y}}_N\}$ can now be concatenated along the second dimension with $\vec{\boldsymbol{Y}} \in \mathbb{R}^{n_1\times N\times n_2\times n_3}$. CSC can only be written as a linear sum of t-products, if the images are 1D/2D signals (i.e. vectorized patches). Otherwise, we can build a higher-order form CSC model by using high order t-products for tensors with order higher than three.

The layer l tensor CSC model can be used to decompose stereo image pairs to obtain the convolutional kernels. It can easily be stacked to form a hierarchy by treating the feature maps of layer l as input for layer $l+1$. In other words, layer $l+1$ has as its input an image with K_l feature maps at layer l. The convolutional kernels of layer l can be obtained by minimizing the following cost function of the tensor CSC model:

$$\min_{\boldsymbol{D}^l,\boldsymbol{X}^l} \mathcal{C}(\boldsymbol{D},\boldsymbol{X}) = \frac{1}{2}\sum_{n=1}^{N}\sum_{c=1}^{K_{l-1}} \|\sum_{k=1}^{K_l} \boldsymbol{D}_{k,c}^{l} \circledast_{HT} \boldsymbol{X}_{k,l}^{n} - \boldsymbol{X}_{c,l-1}^{n}\|_F^2 + \lambda \sum_{n=1}^{N}\sum_{k=1}^{K_l} \|\boldsymbol{X}_{k,l}^{n}\|_{1,1,1,1,1}$$
$$\text{s.t. } \|\boldsymbol{D}_{k,c}^{l}\|_F^2 \le 1 \quad \forall k = 1,\ldots,K_l,\ c = 1,\ldots,K_{l-1}, \tag{5}$$

where $\circledast_{HT}$ represents a 4-th order tensor convolution operator. $\boldsymbol{X}_{c,l-1}^{n}$ and $\boldsymbol{X}_{k,l}^{n}$ represent the sparse coding of the $(l-1)$-th and l-th layers, respectively. $\boldsymbol{D}_{k,c}^{l}$ represents the convolutional kernel of the l-th layer. The penalty parameter $\lambda > 0$ controls the tradeoff between good reconstruction and code sparsity.

The l layer tensor form convolutional sparse coding model is trained in a bottom-up manner, the feature maps and convolutional kernels at the layer l obtained by minimizing (5). The next section will introduce the specific solution process of (5).

B. OPTIMIZATION ALGORITHM

For each layer, (5) is a high-order tensor convolutional sparse coding problem in Section II, which can be divided into two subproblems to be solved. In this way, we can solve the sparse coding $\boldsymbol{X}^l$ and the convolutional kernel $\boldsymbol{D}^l$ of each layer, respectively. Specifically, we use ADMM to solve the sparse coding $\boldsymbol{X}^l$ and the gradient descent algorithm to solve the convolutional kernel $\boldsymbol{D}^l$, and the specific solution process is as follows.

Sparse Coding. Fixing $\boldsymbol{D}^l$, we solve $\boldsymbol{X}^l$ in (5). The augmented lagrangian is given as follows:

$$\begin{aligned}\mathcal{L}(\boldsymbol{X}^l,\boldsymbol{Y}^l,\boldsymbol{U}^l) &= \frac{1}{2}\sum_{n=1}^{N}\sum_{c=1}^{K_{l-1}} \|\sum_{k=1}^{K_l} \boldsymbol{D}_{k,c}^{l} \circledast_{HT} \boldsymbol{X}_{k,l}^{n} - \boldsymbol{X}_{c,l-1}^{n}\|_F^2 \\ &+ \frac{\rho}{2}\sum_{n=1}^{N}\sum_{k=1}^{K_l} \|\boldsymbol{X}_{k,l}^{n} - \boldsymbol{Y}_{k,l}^{n}\|_F^2 + \lambda \sum_{n=1}^{N}\sum_{k=1}^{K_l} \|\boldsymbol{Y}_{k,l}^{n}\|_{1,1,1,1,1} \\ &+ \sum_{n=1}^{N}\sum_{k=1}^{K_l} \langle \boldsymbol{U}_{k,l}^{n}, (\boldsymbol{X}_{k,l}^{n} - \boldsymbol{Y}_{k,l}^{n})\rangle. \end{aligned} \tag{6}$$

Update $\boldsymbol{X}^l$: For computational efficiency, the objective (6) is converted into the Fourier domain by using the definition of the operator $\circledast_{HT}$. It can be seen from Equation (6) that the solution for each $\boldsymbol{X}^n_{k,l}$ is independent of other variables. Here we take derivatives of $\mathcal{L}(\hat{\boldsymbol{X}}^l, \hat{\boldsymbol{Y}}^l, \hat{\boldsymbol{U}}^l))$ (Fourier domain) with respect to $\hat{\boldsymbol{X}}^n_{k,l}$:

$$\frac{\partial \mathcal{L}(\hat{\boldsymbol{X}}^l, \hat{\boldsymbol{Y}}^l, \hat{\boldsymbol{U}}^l)}{\partial \hat{\boldsymbol{X}}^n_{k,l}} = \sum_{c=1}^{K_{l-1}} (\hat{\boldsymbol{D}}^l_{k,c})^H (\sum_{k=1}^{K_l} \hat{\boldsymbol{D}}^l_{k,c} \hat{\boldsymbol{X}}^n_{k,l} - \hat{\boldsymbol{X}}^n_{c,l-1}) + \rho(\hat{\boldsymbol{X}}^n_{k,l} - \hat{\boldsymbol{Y}}^n_{k,l}) + \hat{\boldsymbol{U}}^n_{k,l} \quad (7)$$

For a fixed n, setting $\frac{\partial \mathcal{L}(\hat{\boldsymbol{X}}^l, \hat{\boldsymbol{Y}}^l, \hat{\boldsymbol{U}}^l)}{\partial \hat{\boldsymbol{X}}^n_{k,l}} = 0$, for any given k, the optimal $\hat{\boldsymbol{X}}^n_{k,l}$ is the solution to the following linear system:

$$\tilde{\boldsymbol{D}} \begin{pmatrix} \hat{\boldsymbol{X}}^n_{1,l} \\ \vdots \\ \hat{\boldsymbol{X}}^n_{K_l,l} \end{pmatrix} = \begin{pmatrix} \sum_{c=1}^{K_{l-1}} (\hat{\boldsymbol{D}}^l_{1,c})^H \hat{\boldsymbol{X}}^n_{c,l-1} + \rho \hat{\boldsymbol{Y}}^n_{1,l} - \hat{\boldsymbol{U}}^n_{1,l} \\ \vdots \\ \sum_{c=1}^{K_{l-1}} (\hat{\boldsymbol{D}}^l_{K_l,c})^H \hat{\boldsymbol{X}}^n_{c,l-1} + \rho \hat{\boldsymbol{Y}}^n_{K_l,l} - \hat{\boldsymbol{U}}^n_{K_l,l} \end{pmatrix} \quad (8)$$

where

$$\tilde{\boldsymbol{D}} = \begin{pmatrix} \boldsymbol{D}_{11} + \rho \boldsymbol{E} & \cdots & \boldsymbol{D}_{1K_l} \\ \vdots & \vdots & \vdots \\ \boldsymbol{D}_{K_l 1} & \cdots & \boldsymbol{D}_{K_l K_l} + \rho \boldsymbol{E} \end{pmatrix}, \quad (9)$$

$$\boldsymbol{D}_{nm} = \sum_{c=1}^{K_{l-1}} (\hat{\boldsymbol{D}}^l_{n,c})^H \hat{\boldsymbol{D}}^l_{m,c}, \quad n, m = 1, \ldots, K_l. \quad (10)$$

$(\cdot)^H$ is conjugate transpose. Equation (8) can be effectively minimized by conjugate gradient (CG) descent. $\boldsymbol{X}^l$ can be reconstructed back from $\hat{\boldsymbol{X}}^l$ by taking the inverse Fourier transform of $\hat{\boldsymbol{X}}^l$.

Update $\hat{\boldsymbol{Y}}^l$:

$$\hat{\boldsymbol{Y}}^n_{k,l} \longleftarrow \underset{\hat{\boldsymbol{Y}}^n_{k,l}}{\operatorname{argmin}} \lambda \sum_{k=1}^{K_l} \|\hat{\boldsymbol{Y}}^n_{k,l}\|_{1,1,1,1,1} + \frac{\rho}{2} \sum_{k=1}^{K_l} \|\hat{\boldsymbol{Y}}^n_{k,l} - (\overbrace{\hat{\boldsymbol{X}}^n_{k,l} + \frac{1}{\rho} \hat{\boldsymbol{U}}^n_{k,l}}^{\mathcal{B}^l_n})\|^2_F, \quad n = 1, \ldots, N. \quad (11)$$

The soft thresholding operator $\mathcal{S}_{\frac{\lambda}{\xi}}(\alpha) = \text{sign}(\alpha)\max(0, |\alpha| - \frac{\lambda}{\xi})$ is used to solve $\boldsymbol{Y}^n_{k,l} (\forall n = 1, \ldots, N)$. It is applied in an element-wise fashion to tensor $\mathcal{B}^l_n$.

Update $\hat{\boldsymbol{U}}^l$:

$$\hat{\boldsymbol{U}}^n_{k,l} \longleftarrow \hat{\boldsymbol{U}}^n_{k,l} + \rho(\hat{\boldsymbol{X}}^n_{k,l} - \hat{\boldsymbol{Y}}^n_{k,l}). \quad (12)$$

Convolutional Kernel Learning. Fixing $\boldsymbol{X}^n_{k,l}$ and n, we used the gradient descent method to update $\boldsymbol{D}^l$ by using the diagonalization property of $\circledast_{HT}$:

$$\frac{\partial \mathcal{L}(\boldsymbol{D}^l, \mathcal{H}^l, \mathcal{V}^l)}{\partial \boldsymbol{D}^l_{k,c}}$$

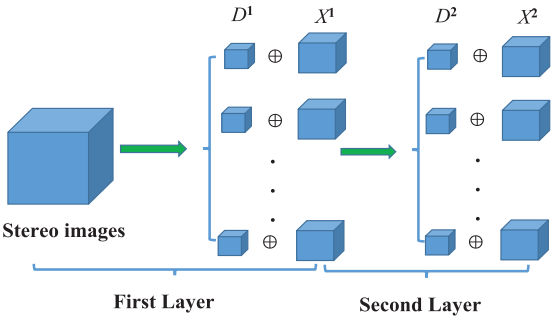

FIGURE 1. The overview of the two layer tensor CSC model.

$$= \sum_{n=1}^{N} \sum_{c=1}^{K_{l-1}} (\boldsymbol{X}^n_{k,l})^T (\sum_{k=1}^{K_l} \boldsymbol{X}^n_{k,l} \boldsymbol{D}^l_{k,c} - \boldsymbol{X}^n_{c,l-1}). \quad (13)$$

When l=2, we can get a two-layer tensor CSC model, as shown in Fig. 1. The learning procedure of the two-layer tensor CSC model is summarized in Algorithm 1. In this way, we can train a two-layer convolutional kernels. After optimization, deep feature can be obtained, which is helpful to improve the stereo matching accuracy. The deep features have the capability to represent more abstract and complex structure information, thus, having a stronger robustness and invariance towards local changes of the image.

Algorithm 1 The Two Layer Tensor CSC Model

Input: Training set $\boldsymbol{I} = \{\boldsymbol{I}^L_j, \boldsymbol{I}^R_j\}_{j=1}^N$, and concatenate into a tensor.
Output: The first layer of convolutional kernel $\boldsymbol{D}^1$; the second layer of convolutional kernel $\boldsymbol{D}^2$.

Initialize: $\boldsymbol{D}_0, \boldsymbol{U}_0 \sim \mathcal{N}(0, 1)$, $\boldsymbol{X}_0 = 0$, $\boldsymbol{Y}_0 = 0$, $t = 0$, $\lambda = \rho = 1$, threshold $\varepsilon = 10^{-8}$;
Precompute Fourier transforms: $\hat{\boldsymbol{I}} = \mathcal{F}(\boldsymbol{I})$, $\hat{\boldsymbol{D}}_0 = \mathcal{F}(\boldsymbol{D}_0)$, $\hat{\boldsymbol{U}}_0 = \mathcal{F}(\boldsymbol{U}_0)$, $\hat{\boldsymbol{X}}_0 = \mathcal{F}(\boldsymbol{X}_0)$, $\hat{\boldsymbol{Y}}_0 = \mathcal{F}(\boldsymbol{Y}_0)$;
while rel $\geq \varepsilon$ **do**
  for $l = 1, 2$ **do**
    foreach $n \in \{1, 2, \ldots, N\}$ **do**
      Update $\boldsymbol{X}^l$ by (7), compute $\boldsymbol{X}^l = \mathcal{F}^{-1}(\hat{\boldsymbol{X}}^l)$;
      Solve $\hat{\boldsymbol{Y}}^l$ by the soft thresholding operator;
      Update $\hat{\boldsymbol{U}}^n_{k,l} \longleftarrow \hat{\boldsymbol{U}}^n_{k,l} + \rho(\hat{\boldsymbol{X}}^n_{k,l} - \hat{\boldsymbol{Y}}^n_{k,l})$;
      Set $t = t + 1$;
      Update rel$= \|\mathcal{C}(\boldsymbol{D}^t, \boldsymbol{X}^t) - \mathcal{C}(\boldsymbol{D}^{(t-1)}, \boldsymbol{X}^{(t-1)})\|_F / \|\mathcal{C}(\boldsymbol{D}^{(t-1)}, \boldsymbol{X}^{(t-1)})\|_F$.
    end
  end
end

C. STEREO MATCHING

In this section, a two-layer dictionary learning model based on the two-layer tensor CSC model is constructed. Specifically, the two-layer high-order tensor CSC model is used

to train a two-layer convolutional kernels, then dictionary learning theory is used to learn a dictionary, and the sparse representation coefficient under the dictionary is taken as the matching feature. Here, we build a two-layer deep stereo matching model, the specific process is as follows.

Firstly, multidimensional convolution operation is carried out between the first-layer convolutional kernel $\boldsymbol{D}^1$ learned in the previous section and the original left image $\boldsymbol{I}_0$ and right image $\boldsymbol{I}_1$:

$$\boldsymbol{F}_0 = convn(\boldsymbol{I}_0, \boldsymbol{D}^1), \quad \boldsymbol{F}_1 = convn(\boldsymbol{I}_1, \boldsymbol{D}^1), \tag{14}$$

where $convn(\cdot)$ represents multidimensional convolution operation.

Secondly, multidimensional convolution operation is carried out between the second-layer convolution kernel $\boldsymbol{D}^2$ learned in the previous section and the original left $\boldsymbol{I}_0$ and right image $\boldsymbol{I}_1$:

$$\boldsymbol{DF}_0 = convn(\boldsymbol{I}_0, \boldsymbol{D}^2), \quad \boldsymbol{DF}_1 = convn(\boldsymbol{I}_1, \boldsymbol{D}^2), \tag{15}$$

Then, the first-layer dictionary $\boldsymbol{D}_1$ is learned on the first-layer stereo image pairs $\boldsymbol{F}_0$ and $\boldsymbol{F}_1$, which can be obtained according to the discriminant dictionary learning method in literature [37]:

$$\min_{\boldsymbol{D},\boldsymbol{W},\alpha,\nu} \frac{1}{2}\|\boldsymbol{P}_{tr} - \boldsymbol{D}\alpha\|_F^2 + \eta\|\boldsymbol{W}\alpha\| - \frac{\|\nu^T(\alpha_l - \alpha_r^{ng})\|_F^2}{\|\nu^T(\alpha_l - \alpha_r^{ps})\|_F^2}, \tag{16}$$

where $\boldsymbol{W} \in \mathbb{R}^{K\times 1}$ is the weighting vector of the sparse representation α, $\boldsymbol{P}_{tr}$ represents input data (training data), and $\nu \in \mathbb{R}^{K\times 1}$ is a projection vector which maximizes the ratio of the distance between the unmatched pairs to the distance between matched pairs.

Similarly, the second-layer dictionary $\boldsymbol{D}_2$ is learned on the second-layer image features $\boldsymbol{DF}_0$ and $\boldsymbol{DF}_1$.

In this way, we can calculate the sparse representation coefficient β_1 under $\boldsymbol{D}_1$, namely,

$$\min_{\beta_1} \frac{1}{2}\|\boldsymbol{H} - \boldsymbol{D}_1\beta_1\|_F^2 + \tau\|\beta_1\|_1. \tag{17}$$

Therefore, the pixel matching cost of the first layer at the position $p = (x, y)$ can be calculated by measuring the ℓ_1 norm between the sparse representation coefficients $\beta_1^L \in \mathbb{R}^{K\times 1}$ and $\beta_1^R \in \mathbb{R}^{K\times 1}$:

$$\boldsymbol{C}_1(p, \hat{d}) = \|\beta_1^L - \beta_1^R\|_1, \tag{18}$$

where $\hat{d}$ is disparity, and the stereo pair is partitioned into patches $\boldsymbol{H}_L$ and $\boldsymbol{H}_R$ via an $d \times d$ sliding window.

Similarly, according to the above equations (17) and (18), the second layer matching cost can be calculated:

$$\min_{\beta_2} \frac{1}{2}\|\boldsymbol{H} - \boldsymbol{D}_2\beta_2\|_F^2 + \tau\|\beta_2\|_1,$$
$$\boldsymbol{C}_2(p, \hat{d}) = \|\beta_2^L - \beta_2^R\|_1. \tag{19}$$

Finally, the pixel matching cost at position (x, y) can be obtained by the weighted summation of the matching cost of the first layer $\boldsymbol{C}_1(p, \hat{d})$ and the second layer $\boldsymbol{C}_2(p, \hat{d})$:

$$\boldsymbol{C}(p, \hat{d}) = \boldsymbol{C}_1(p, \hat{d}) + \omega\boldsymbol{C}_2(p, \hat{d}), \tag{20}$$

where ω is a preset weight value, which measures the effects of the first-layer matching cost and the second-layer matching cost.

D. COMPLEXITY ANALYSIS

1) COMPLEXITY OF TWO-LAYER TENSOR CSC MODEL

The two-layer tensor CSC model needs to train the convolutional kernel $\boldsymbol{D}^l \in \mathbb{R}^{n_1\times K\times n_2\times n_3}$ and sparse coding of each layer. For the sparse coding step, the most expensive part is solving for $\boldsymbol{X}^l$, which involves taking n_1 2D Fourier transforms of size $n_2 \times n_3$ and solving n_2n_3 linear systems each of size $K \times K$. Therefore, the total cost of updating the sparse codes can be estimated to be $\mathcal{O}_s = \mathcal{O}(n_2n_3K^3) + \mathcal{O}(n_1n_2n_3log(n_2n_3))$. Updating convolution kernel $\boldsymbol{D}^l$, the main computational cost comes from calculating the gradient $\frac{\partial\mathcal{L}(\boldsymbol{D}^l,\mathcal{H}^l,\mathcal{V}^l)}{\partial\boldsymbol{D}_{k,c}^l}$ in (13), its computational complexity is $\mathcal{O}_c = \mathcal{O}(N(K_2W_1H_1Kn_1n_2n_3 + K_1W_2H_2Kn_1n_2n_3))$, in which W_2 and H_2 represent the width and height of the 2-th layer sparse coding, respectively. Therefore, we obtain the total computational complexity of the 2-th layer: $\mathcal{O}_s + \mathcal{O}_c$.

2) COMPLEXITY OF STEREO MATCHING COST

Suppose the size of test image is $W \times H$. The computational complexity of the stereo matching cost includes the calculation of the sparse representation coefficient and the matching cost. The computational complexity of the sparse representation coefficient is $\mathcal{O}(2WHK)$. The computational complexity of the matching cost is $\mathcal{O}(2WHd^2)$. As a result, the complexity of stereo matching cost is $\mathcal{O}(2WHK) + \mathcal{O}(2WHn^2)$.

In fact, once the training phase of the convolutional kernels is completed, the complexity of stereo matching cost is $\mathcal{O}(2WHK) + \mathcal{O}(2WHd^2)$. This can greatly reduce the computational cost.

IV. EXPERIMENTS

A. DATASETS DESCRIPTION

The Middlebury 2014 data set [2], [38] from Middlebury benchmark v3 consists of 15 training images and 15 test images without ground truth disparity maps. These images are acquired by different stereo systems and contain different artificial indoor scenes. In addition, The proposed method is experimented on the classic datasets [37] from Middlbury benchmark v2, which contains more various scenes.

KITTI data set [30], [39] contains two sub-datasets (i.e. KITTI 2012 and KITTI 2015), where KITTI 2012 data set contains 194 training images and 195 test images without ground truth disparity maps; KITTI 2015 data set contains 200 training images and 200 test images without ground truth disparity maps. These images are captured from real world dataset with street views.

B. PARAMETER ANALYSIS

In order to compare the performance of different methods, we select percentage of bad matching pixels, which is defined

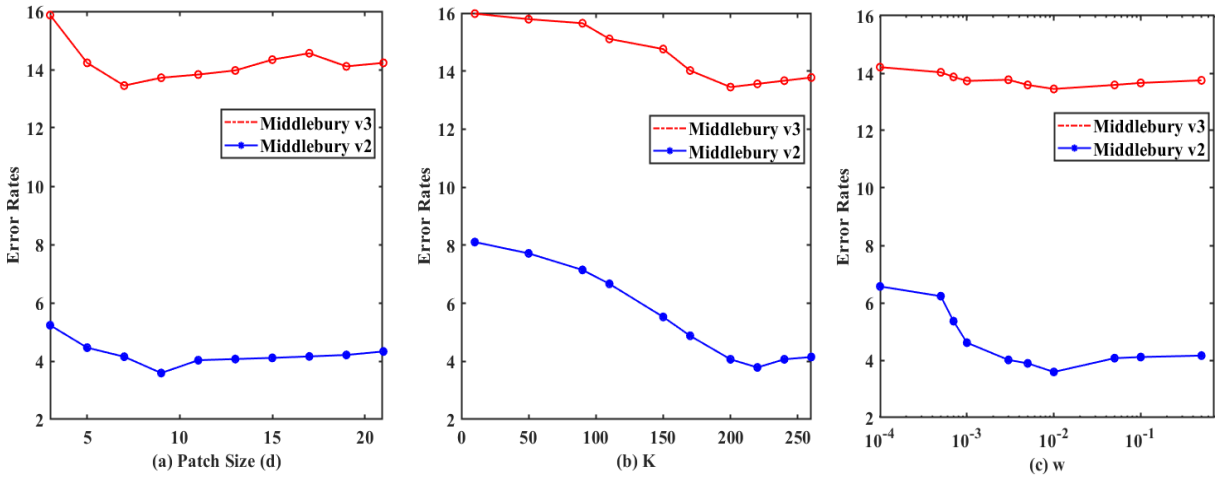

FIGURE 2. Parameter analysis. (a) the patch size *d*; (b) the number of atoms in the dictionary *K*; (c) the weight ω.

as follows:

$$\frac{1}{N}\sum_{(x,y)}(|d_C(x,y)-d_T(x,y)|>\delta_d), \tag{21}$$

where δ_d is a disparity error tolerance. In our current set of experiments, we use $\delta_d = 1$. N is the total number of pixels. $d_C(x, y)$ is the computed disparity map and $d_T(x, y)$ is the ground truth map.

Before doing the experiment, we need to initialize the model parameters. The parameters involved in this paper mainly include: the patch size d, the number of atoms in the dictionary K, and the weight ω.

For the patch size d, we set d to 3, 5, 7, 9, 11, 13, 15, 17, 19 and 21 to analyze its influence on the performance of the proposed method, as shown on the left of Fig. 2. It can be seen that there is a sharp increase in error rates when d ranges from 9 to 17 on Middlebury benchmark v3. The error rates decrease gradually when d ranges from 3 to 9 on Middlebury benchmark v3. While there is a sharp increase in error rates when d ranges from 9 to 17 on Middlebury benchmark v2, and the error rates decrease gradually when d ranges from 3 to 9 on Middlebury benchmark v2. As a trade-off between the error rates and computational cost, we set d as 9 for the all Middlebury dataset.

For the number of atoms in the dictionary K, we set K to 10, 50, 90, 110, 150, 170, 200, 220, 240, and 260 to analyze its influence on the performance of the proposed method, as shown in the middle of Fig. 2. It can be seen that the error rates increase when K ranges from 200 to 260 on Middlebury benchmark v3. The error rates decrease gradually after K ranges from 10 to 200 on Middlebury benchmark v3. While the error rates increase when K ranges from 200 to 260 on Middlebury benchmark v2, and the error rates decrease gradually after K ranges from 10 to 220 on Middlebury benchmark v2. For Middlebury benchmark v3 and Middlebury benchmark v2, $K = 200$ is not the best choice ($K = 220$ is the best choice for the benchmark v2). But a bigger K means more computation, therefore, choose the smaller $K = 200$ for all Middlebury dataset.

As for the weight ω, we set ω to 0.0001, 0.0005, 0.0007, 0.001, 0.003, 0.005, 0.01, 0.05, 0.1, and 0.5 to analyze its influence on the performance of the proposed method, as shown on the right of Fig. 2. It can be seen that the error rates increase when ω ranges from 0.01 to 0.5 on Middlebury benchmark v3 and Middlebury benchmark v2. The error rates decrease gradually after ω ranges from 0.0001 to 0.01 on Middlebury benchmark v3 and Middlebury benchmark v2. Therefore, we set ω as 0.01 for all Middlebury dataset.

C. COMPARISON WITH OTHER STATE-OF-THE-ART METHODS

In this section, the proposed method is compared with some closely related stereo matching methods on Middlebury benchmark v3 and Middlebury benchmark v2. At the same time, we also compared the state-of-the-art deep learning methods. The results are shown in Table 1 and Table 2, respectively. In addition, we also show disparity maps of different methods, as shown in Fig. 3 and Fig. 4.

1) RESULTS ON MIDDLEBURY BENCHMARK v3

It can be seen from Table 1 that that the proposed method obtains the best results in terms of Avgerr-All and Rms-All on the Middlebury benchmark v3. The proposed method on training set is 10.8% lower than SM-AWP and DSGCA in Avgerr-All and Rms-All, and the proposed method on test set is 29% lower than SM-AWP and DSGCA in Avgerr-All and Rms-All. The proposed method on training set is 22% lower than MTS in Avgerr-All and Rms-All, and the proposed method on test set is 32% lower than MTS in Avgerr-All and Rms-All. The proposed method on training set is 4.9% lower than MDP, LPS, MBM, SGBMP, and LAMC_DSM in Avgerr-All and Rms-All, and the proposed method on test set is 5% lower than MDP, LPS, MBM, SGBMP, and LAMC_DSM in Avgerr-All and Rms-All. Compared with

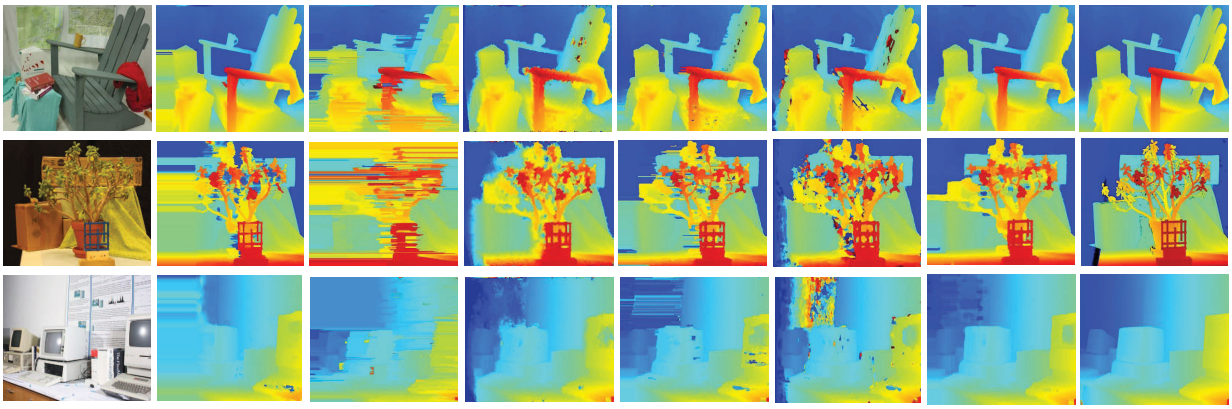

FIGURE 3. Qualitative comparison results on Middlebury 2014 dataset. The first column: input color images from Middlebury 2014; From second to seventh column: the results achieved by SGBMP, MTS, LAMC-DSM, MBM, DSGCA and Ours, respectively; last column: ground truth disparity maps.

TABLE 1. Comparison of our method and some selected state-of-the-art supervised methods and existing unsupervised methods on Middlebury 2014 test dataset.

Methods	Avgerr-All [a](%)		Rms-All [b](%)		Environment	Runtime
	training set	test set	training set	test set		
CBMV [14]	11.5	14.4	34.9	46.9	Nvidia GTX TitanX (CUDA, Python, C/C++)	1001 s
JMR [40]	9.57	15.7	32.0	49.0	NVidia Titan X (C++/CUDA)	4.46 s
MC-CNN-arct [21]	11.8	17.9	36.6	55.0	Nvidia GTX Titan (CUDA, Lua/Torch7)	150 s
MC-CNN-fst [21]	12.8	19.3	37.5	55.7	Nvidia GTX Titan (CUDA, Lua/Torch7)	1.69 s
MC-CNN-WS [15]	13.7	19.9	38.3	56.6	Nvidia (GPU, Lua/Torch)	3.19 s
LW-CNN [12]	10.9	19.3	35.3	58.8	Nvidia Geforce GTX Titan X (Torch)	314 s
DSGCA [41]	18.7	26.9	45.7	66.6	Nvidia Geforce GTX 1080(GPU, MATLAB)	10.2 s
RBES-GC [42]	-	10.7	-	-	Intel core i5 @3.0 GHz	3.9 s
UpBIU(NSP) [43]	-	-	-	36.1	Nvidia Geforce GTX TitanX(CUDA, C++/MATLAB)	492 s
Chen et al. [44]	-	20.0	-	-	Nvidia TitanX(CUDA, C++/Torch7)	80 s
MDP [45]	10.8	13.6	32.6	43.4	Nvidia GTX Titan X (Python)	79.2 s
LPS [46]	12.8	19.7	30.0	44.7	4 cores i7 @3.6GHz (c/c++)	9.52 s
MBM [47]	10.1	12.9	27.1	37.5	Nvidia GTX TITAN X (GPU)	**0.01** s
SGBMP [48]	11.2	11.4	26.7	30.8	Nvidia GTX TITAN X (Python/c++)	9.12 s
LAMC_DSM [49]	14.6	23.1	38.4	60.1	1 core Intel Xeon @2.5 GHz (MATLAB)	704 s
MTS [50]	27.6	38.5	49.2	68	4 cores i7 @3.4GHz(C++)	2.6 s
SM-AWP [51]	16.0	35.6	39.9	70.5	i7 core @2.7GHz(MATLAB)	3.3 s
Ours	**5.15**	**6.4**	**13.6**	**22.3**	Nvidia Geforce GTX 1060(CPU, MATLAB)	13.2 s

[a] **Average error in all pixels**.
[b] **Root mean square error in all pixels. Best results are shown in bold**.

the state-of-the-art deep learning methods CBMV, JMR, MC-CNN-arct, MC-CNN-fst, MC-CNN-WS and LW-CNN, the performance of the proposed methods is better than that of their methods in terms of Avgerr-All and Rms-All. The proposed method on training set is 4.4% lower than CBMV, JMR, MC-CNN-arct, MC-CNN-fst, MC-CNN-WS and LW-CNN in Avgerr-All and Rms-All, and the proposed method on test set is 8% lower than CBMV, JMR, MC-CNN-arct, MC-CNN-fst, MC-CNN-WS and LW-CNN in Avgerr-All and Rms-All. In terms of the disparity maps of different methods in Fig. 3, disparity maps of the proposed method are the closest to the ground-truth disparity maps (last column), and the proposed method produces a smooth and dense disparity maps with the least noise. The disparity maps achieved by MTS produces too much noise. The disparity maps achieved by SGBMP do not perform well in geometric details and discontinuous disparity areas. Compared with other methods, the disparity map generated by the proposed method is not only visually closer to ground-truth disparity maps, but also performs well in geometric details and discontinuous disparity areas.

2) RESULTS ON MIDDLEBURY BENCHMARK v2

Similarly, the quantitative results of the proposed method on Middlebury benchmark v2 are shown in Table 2. As not all the codes of compared methods are available, we obtained the quantitative results either by performing the codes from the authors' homepages without modification or from the corresponding papers directly. Specifically, the results of the non-local methods (MST [52] and Cross_E [53]) and the local method based on guided filter [9] are obtained by using the codes provided by authors. The results of the local methods AEGF [54] and the data-driven method MC-CNN [21] are directly cited from the paper [54]. We can see our proposed method outperforms all the five compared methods, even the data-driven method based on MC-CNN. The average error of the proposed method is 2.2% lower than the average errors

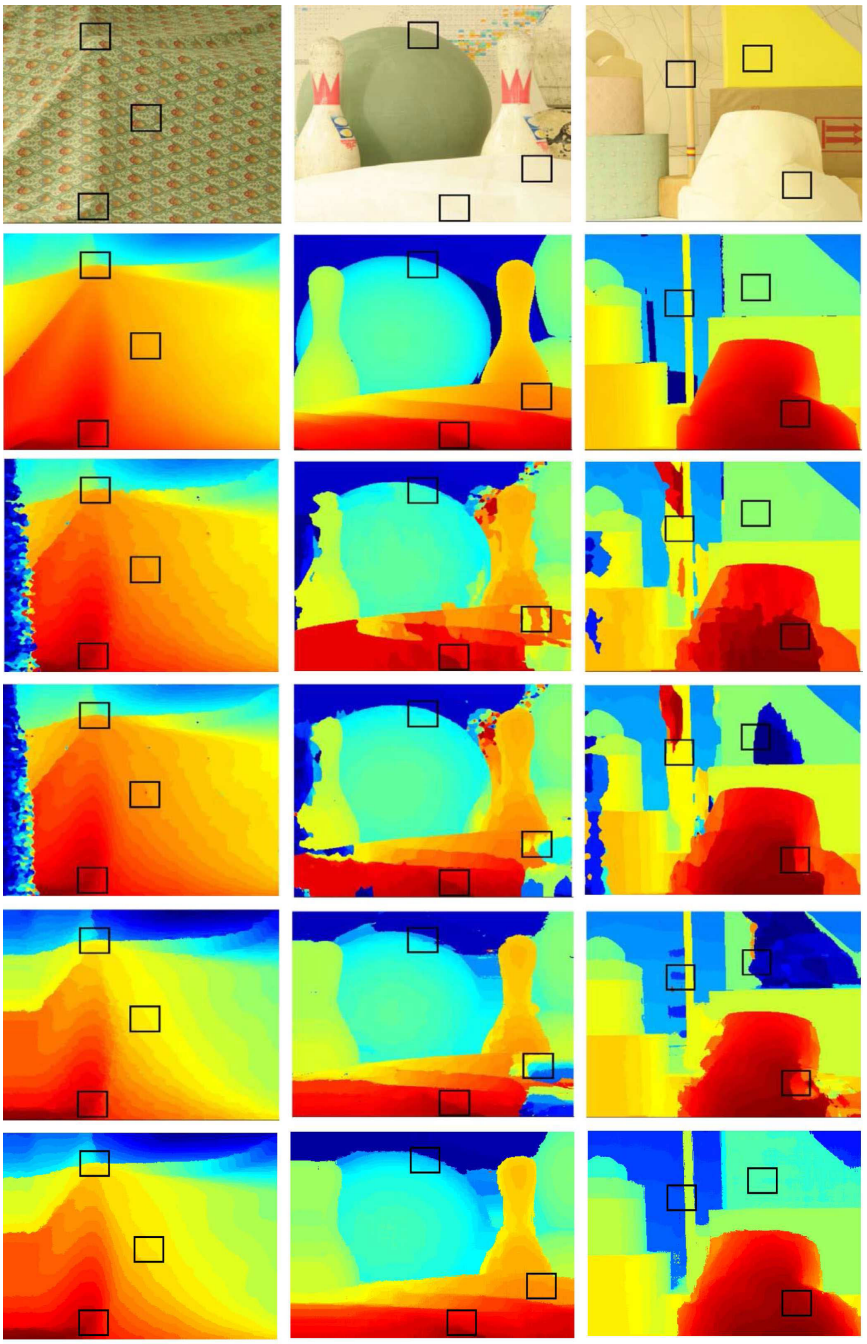

FIGURE 4. Disparity maps of datasets Cloth1, Bowling1, and Lampshade2 obtained by using MST, Cross_E, GF, and the proposed method from top to bottom except for the left view color image and the ground truth disparity maps in the first two rows.

of MST, Cross_E and GF, the average error of the proposed method is 1.1% lower than the average error of AEGF, and the average error of the proposed method is 0.4% lower than the average error of MC-CNN. In terms of the disparity maps of MST, Cross_E, GF, AEGF, and MC-CNN in Fig. 4, disparity maps of the proposed method are the closest to the ground-truth disparity maps, and the proposed method produces a smooth and dense disparity maps with the least noise. For example, in the black box area, the proposed method is visually superior to other methods. Disparity maps generated by MST and Cross_E not only generate too much noise, but also perform poorly in discontinuous disparity areas and occluded areas. Compared with other methods, the proposed method produces less noise, and the proposed method performs better than other methods in occlusion areas and discontinuous disparity areas.

TABLE 2. Quantitative evaluation on Middlebury benchmark v2.

Datasets	MST	Cross_E	GF	AEGF	MC-CNN	Ours
Aloe	8.37	8.06	6.70	5.25	4.75	5.13
Art	14.6	12.6	9.43	7.58	5.51	9.34
Baby1	12.2	12.1	4.06	3.18	3.14	2.26
Baby2	21.8	9.23	5.31	4.15	3.83	2.32
Baby3	6.95	7.34	7.26	4.34	3.92	3.10
barn1	2.34	3.06	0.57	0.51	0.55	0.74
barn2	3.87	4.25	0.46	0.35	0.38	0.68
Books	13.2	11.6	10.3	8.05	7.28	8.02
Bowling1	25.6	17.1	13.9	13.3	11.6	4.15
Bowling2	16.2	12.2	8.35	7.13	6.57	4.21
bull	0.88	1.39	0.34	0.41	0.35	0.61
Cloth1	3.48	3.79	1.18	0.49	0.85	0.21
Cloth2	6.55	4.40	3.88	2.88	3.12	1.58
Cloth3	4.23	4.14	2.09	1.31	1.35	1.67
Cloth4	2.29	1.61	2.04	1.54	1.27	1.11
Cones	5.70	6.10	2.91	2.01	1.87	4.15
Dolls	8.27	6.64	5.19	5.36	3.95	5.23
Flowerpots	20.5	18.4	15.6	10.4	8.29	6.25
Lampshade1	14.2	12.6	11.5	11.4	10.4	5.18
Lampshade2	17.6	18.1	21.3	17.0	14.7	4.01
Laundry	16.0	16.8	15.6	14.3	10.3	11.8
Moebius	11.9	12.7	10.3	8.71	7.50	9.22
poster	1.43	2.06	0.62	1.38	0.79	0.88
Reindeer	12.5	8.87	5.58	3.57	2.87	4.72
Rocks1	7.34	6.77	4.17	2.35	2.21	3.91
Rocks2	5.40	5.87	2.35	2.44	2.05	3.13
sawtooth	7.48	8.23	0.96	1.02	0.83	1.93
Teddy	7.85	8.50	6.90	5.21	4.80	6.27
Tsukuba	2.42	2.58	1.86	1.32	1.06	2.03
Venus	1.63	2.10	0.25	0.26	0.21	0.54
Average Error/%	9.42	8.31	6.03	4.91	4.21	**3.81**

TABLE 3. The average error of the proposed method with different number of layers on different datasets.

Datasets	Average Error		
	1-th layer	2-th layer	1-th+2-th layers
Middlebury benchmark v3	6.34	7. 11	**5.15**
Middlebury benchmark v2	4.07	4.16	**3.81**

In terms of runtime, it can be observed from Table 1 that most of the other methods use GPU/CUDA accelerations, except for our method, RBES-GC, LAMC_DSM [49], MTS, SM-AWP and LPS [46]. The running time of our method is significantly lower than that of LAMC_DSM, but higher than that of MTS, SM-AWP, and LPS. Besides, we achieve a lower computational expense, even when it is compared with deep learning methods CBMV [14] and LW-CNN [12] which work on Nvidia GTX Titan with CUDA acceleration and deep learning method MDP [45] which works on Nvidia GTX Titan X. Our method is done on the CPU and an accelerated version of the proposed method on the GPU will be considered in the future.

D. ANALYSIS AND DISCUSSION

It can be seen from Fig. 3 that disparity maps of the proposed method obviously produce less noise than other methods. Quantitatively, the results in the Table 1 are consistent with the visual results. Next, we further investigate the effectiveness of the proposed method from the following ablation study.

TABLE 4. The average error of the proposed method with different l.

Datasets		l=1	l=2	l=3	l=4
Middlebury benchmark v3	Error	6.34	5.15	6.02	6.75
	Time	9.5s	13.2s	25.7s	46.3s
Middlebury benchmark v2	Error	4.69	3.81	4.02	4.79
	Time	10.1s	17.5s	32.8s	58.9s

TABLE 5. Quantitative evaluation of different methods on KITTI dataset.

Supervised Methods	KITTI 2012		KITTI 2015	
	Out-noc	Avg-all	D1-bg	D1-all
MC-CNN-WS fst [15]	13.9%	-	14.1%	-
MC-CNN-fst [21]	15.4%	-	15.4%	-
Deep Embed [55]	8.9%	4.9 px	7.3%	2.0%
USCNN [17]	-	-	11.2%	-
OASM-Net [56]	**6.4**%	2.0 px	6.9%	9.0%
ELAS [57]	8.2%	1.7 px	7.9%	9.7%
CostFilter [9]	20.0%	5.4 px	17.5%	18.4%
StereoGAN [58]	-	1.94 px	-	8.78%
FC-DCNN [59]	-	5.61 px	-	7.71%
Lu et al. [60]	6.66%	-	6.54%	7.44%
USMC [33]	6.4%	-	6.2%	-
Liu et al+L_p [61]	5.87%	-	-	9.83%
Ours	**5.8**%	**1.4** px	**5.5**%	**1.3**%

It can be seen from Table 3 that the proposed method only uses the first layer and the second layer to obtain an error rate of 6.34 and 7. 11 on Middlebury benchmark v3, respectively; while the proposed method uses the first layer and the second layer to obtain an error rate of 4.07 and 4.16 on Middlebury benchmark v2, respectively. When the proposed method uses the first layer and the second layer at the same time, an error rate of 5.15 and 3.81 is obtained on Middlebury benchmark v3 and Middlebury benchmark v2, respectively. This shows the effectiveness of the proposed method with two layers.

In addition, we also discuss the influence of different l on the performance of the proposed method, and the results are shown in Table 4. According to the results in the table, when $l = 2$, the proposed method achieves the smallest average error. When l is greater than 2, the average error of the proposed method gradually increases, and the time cost also increases. The time cost when $l = 3$ is about twice the time cost when $l = 2$. Considering the time cost and average error, choosing $l = 2$ is a better choice. This also shows the effectiveness of the proposed method with two layers.

In order to further verify the performance of the proposed method on other data sets, we also conducted experiments on the KITTI dataset. The experimental results are shown in the Table 5. It can be seen from the table that our method achieves the smallest error among all comparison methods. This shows that the proposed method has good generalization performance.

Here, Out-noc, Avg-all, D1-bg, and D1-all are used as evaluation metrics for different methods, in which ''Out-noc'' is percentage of erroneous pixels in non-occluded, ''Avg-all'' is average disparity (end-point error) in total areas, ''D1-bg'' is percentage of outliers averaged only over background regions in first frame, and ''D1-all'' is percentage of

TABLE 6. Comparison results under other metrics on KITTI 2012 dataset.

Methods	5-PE	4-PE	2-PE	Para.num
FC-DCNN [59]	3.71	4.40	8.81	0.37M
OASM-Net [56]	4.32	5.11	9.01	0.95M
SGBM [48]	5.03	6.03	10.6	-
ADSM [62]	6.20	7.09	13.13	-
GF(Census) [63]	8.49	9.57	16.75	-
Ours	**3.53**	**4.21**	**7.76**	**0.28**M

outliers averaged over all ground truth pixels in first frame, respectively.

In addition, we introduce more other evaluation indicators to evaluate the experimental results, and the results are shown in Table 6. In Table 6, the x-PE error measures the percentage of bad pixels whose error is larger than x pixels, and Para.num is the number of parameters. From the results in the table, the result of our method is the best, and the number of parameters in our model is also the least. Under the x-PE measurement, the error of the OASM-Net method is about 1.2 times the error of our method, and the parameter amount of the OASM-Net method is about 3.4 times that of our method. The error of the GF(Census) is about 2.2 times the error of our method. This shows that the proposed method still maintains good performance even under other metrics. This shows the effectiveness of the proposed method.

V. CONCLUSION

In this paper, a deep high-order tensor convolutional sparse coding model is proposed, which can automatically learn the deep convolutional kernel. Based on the learned deep convolutional kernel, a two-layer dictionary learning model is established. Then, the sparse representation coefficients under the first-layer dictionary and the second-layer dictionary are respectively solved, and a new weighted matching cost method is constructed, which combines shallow and deep features. The experimental results on the Middlebury benchmark v3 and Middlebury benchmark v2 show the effectiveness of the proposed method.

In the future, we will design an efficient solution algorithm for deep high-order tensor convolution sparse coding, and study deeper matching costs. In addition, GPU version of the proposed method will also be considered.

REFERENCES

[1] X. Yang, X. Chen, and J. Xi, "Block based dense stereo matching using adaptive cost aggregation and limited disparity estimation," in *Proc. IEEE Int. Instrum. Meas. Technol. Conf.*, May 2017, pp. 1–6.

[2] D. Scharstein and R. Szeliski, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," *Int. J. Comput. Vis.*, vol. 47, nos. 1–3, pp. 7–42, Apr. 2002.

[3] R. Ben-Ari and N. Sochen, "A geometric approach for regularization of the data term in stereo-vision," *J. Math. Imag. Vis.*, vol. 31, no. 1, pp. 17–33, May 2008.

[4] J.-H. Mun and Y.-S. Ho, "Guided image filtering based disparity range control in stereo vision," *Electron. Imag.*, vol. 2017, no. 5, pp. 130–136, Jan. 2017.

[5] H. Hirschmüller, P. R. Innocent, and J. Garibaldi, "Real-time correlation-based stereo vision with reduced border errors," *Int. J. Comput. Vis.*, vol. 47, pp. 229–246, Apr. 2002.

[6] Y. Zhan, Y. Gu, K. Huang, C. Zhang, and K. Hu, "Accurate image-guided stereo matching with efficient matching cost and disparity refinement," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 26, no. 9, pp. 1632–1645, Sep. 2016.

[7] R. Zabih and J. Woodfill, "Non-parametric local transforms for computing visual correspondence," in *Proc. Eur. Conf. Comput. Vis.*, 1994, pp. 151–158.

[8] K. Yamaguchi, D. McAllester, and R. Urtasun, "Efficient joint segmentation, occlusion labeling, stereo and flow estimation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jan. 2014, pp. 1–16.

[9] A. Hosni, C. Rhemann, M. Bleyer, C. Rother, and M. Gelautz, "Fast cost-volume filtering for visual correspondence and beyond," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 2, pp. 504–511, Feb. 2013.

[10] S. Hermann and T. Vaudrey, "The gradient—A powerful and robust cost function for stereo matching," in *Proc. Int. Conf. Image Vis. Comput.*, Nov. 2011, pp. 467–474.

[11] W. Luo, A. G. Schwing, and R. Urtasun, "Efficient deep learning for stereo matching," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Dec. 2016, pp. 5695–5703.

[12] H. Park and K. M. Lee, "Look wider to match image patches with convolutional neural networks," *IEEE Signal Process. Lett.*, vol. 24, no. 12, pp. 1788–1792, Dec. 2017.

[13] M. El-Khamy, H. Ren, X. Du, and J. Lee, "Multitask deep neural networks for tele-wide stereo matching," *IEEE Access*, vol. 8, pp. 184383–184398, Oct. 2020.

[14] K. Batsos, C. Cai, and P. Mordohai, "CBMV: A coalesced bidirectional matching volume for disparity estimation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 2060–2069, doi: 10.1109/CVPR.2018.00220.

[15] S. Tulyakov, A. Ivanov, and F. Fleuret, "Weakly supervised learning of deep metrics for stereo reconstruction," in *Proc. IEEE Int. Conf. Comput. Vis.*, Oct. 2017, pp. 1348–1357.

[16] K. Achour and L. Mahiddine, "Segmentation-based stereo matching using combinatorial similarity measurement and adaptive support region," *J. Math. Imag. Vis.*, vol. 16, pp. 17–29, May 2002.

[17] A. Ahmadi and I. Patras, "Unsupervised convolutional neural networks for motion estimation," in *Proc. IEEE Int. Conf. Image Process.*, Sep. 2016, pp. 1629–1633.

[18] Y. Chen, L. Liu, J. Tao, X. Chen, R. Xia, Q. Zhang, J. Xiong, K. Yang, and J. Xie, "The image annotation algorithm using convolutional features from intermediate layer of deep learning," *Multimedia Tools Appl.*, vol. 80, no. 3, pp. 4237–4261, Jan. 2021.

[19] Y. Chen, H. Zhang, L. Liu, X. Chen, Q. Zhang, K. Yang, R. Xia, and J. Xie, "Research on image inpainting algorithm of improved GAN based on two-discriminations networks," *Appl. Intell.*, vol. 51, no. 6, pp. 3460–3474, Jun. 2021, doi: 10.1007/s10489-020-01971-2.

[20] Y. Chen, V. Phonevilay, J. Tao, X. Chen, R. Xia, Q. Zhang, K. Yang, J. Xiong, and J. Xie, "The face image super-resolution algorithm based on combined representation learning," *Multimedia Tools Appl.*, pp. 1–23, Nov. 2020, doi: 10.1007/s11042-020-09969-1.

[21] J. Žbontar and Y. LeCun, "Stereo matching by training a convolutional neural network to compare image patches," *J. Mach. Learn. Res.*, vol. 17, no. 1, pp. 2287–2318, Oct. 2016.

[22] P. Brandao, E. Mazomenos, and D. Stoyanov, "Widening Siamese architectures for stereo matching," *Pattern Recognit. Lett.*, vol. 120, pp. 75–81, Apr. 2019.

[23] Y. Chen, H. Zhang, L. Liu, J. Tao, Q. Zhang, K. Yang, R. Xia, and J. Xie, "Research on image inpainting algorithm of improved total variation minimization method," *J. Ambient Intell. Hum. Comput.*, pp. 1–10, Jan. 2021, doi: 10.1007/s12652-020-02778-2.

[24] Y. Chen, L. Liu, V. Phonevilay, K. Gu, R. Xia, J. Xie, Q. Zhang, and K. Yang, "Image super-resolution reconstruction based on feature map attention mechanism," *Appl. Intell.*, vol. 51, no. 7, pp. 4367–4380, Jul. 2021, doi: 10.1007/s10489-020-02116-1.

[25] Y. Chen, L. Liu, J. Tao, R. Xia, Q. Zhang, K. Yang, J. Xiong, and X. Chen, "The improved image inpainting algorithm via encoder and similarity constraint," *Vis. Comput.*, vol. 37, no. 7, pp. 1691–1705, Jul. 2021.

[26] H. Xu and J. Zhang, "AANet: Adaptive aggregation network for efficient stereo matching," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2020, pp. 1956–1965.

[27] F. Zhang, X. Qi, R. Yang, V. Prisacariu, B. Wah, and P. Torr, "Domain-invariant stereo matching networks," in *Proc. Eur. Conf. Comput. Vis.*, Nov. 2020, pp. 420–439.

[28] Z. Wu, X. Wu, X. Zhang, S. Wang, and L. Ju, "Semantic stereo matching with pyramid cost volumes," in *Proc. IEEE Int. Conf. Comput. Vis.*, Oct. 2019, pp. 7483–7492.
[29] Z. Liang, Y. Guo, Y. Feng, W. Chen, L. Qiao, L. Zhou, J. Zhang, and H. Liu, "Stereo matching using multi-level cost volume and multi-scale feature constancy," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 1, pp. 300–315, Jan. 2021.
[30] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The KITTI dataset," *Int. J. Robot. Res.*, vol. 32, pp. 1231–1237, Sep. 2013.
[31] J. Chen, W. Zhang, Y. Qian, and M. Ye, "Deep tensor factorization for hyperspectral image classification," in *Proc. IEEE Int. Geosci. Remote Sens. Symp.*, Jul. 2018, pp. 4788–4791.
[32] M. Xi, L. Chen, D. Polajnar, and W. Tong, "Local binary pattern network: A deep learning approach for face recognition," in *Proc. IEEE Int. Conf. Image Process.*, Sep. 2016, pp. 3224–3228.
[33] C. Cheng, H. Li, and L. Zhang, "Two-branch convolutional sparse representation for stereo matching," *IEEE Access*, vol. 9, pp. 21910–21920, 2021.
[34] X. Ren, K. Chen, X. Yang, Y. Zhou, J. He, and J. Sun, "A new unsupervised convolutional neural network model for Chinese scene text detection," in *Proc. IEEE China Summit Int. Conf. Signal Inf. Process.*, Jul. 2015, pp. 428–432.
[35] Y. Chen, J. Tao, L. Liu, J. Xiong, R. Xia, J. Xie, Q. Zhang, and K. Yang, "Research of improving semantic image segmentation based on a feature fusion model," *J. Ambient Intell. Hum. Comput.*, pp. 1–13, May 2020, doi: 10.1007/s12652-020-02066-z.
[36] A. Bibi and B. Ghanem, "High order tensor formulation for convolutional sparse coding," in *Proc. IEEE Int. Conf. Comput. Vis.*, Oct. 2017, pp. 1790–1798.
[37] J. Yin, H. Zhu, D. Yuan, and T. Xue, "Sparse representation over discriminative dictionary for stereo matching," *Pattern Recognit.*, vol. 71, pp. 278–289, Nov. 2017.
[38] D. Scharstein, H. Hirschmüller, Y. Kitajima, G. Krathwohl, N. Nešić, X. Wang, and P. Westling, "High-resolution stereo datasets with subpixel-accurate ground truth," in *Proc. German Conf. Pattern Recognit.*, Oct. 2014, pp. 31–42.
[39] M. Menze and A. Geiger, "Object scene flow for autonomous vehicles," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2015, pp. 3061–3070.
[40] P. Knöbelreiter, C. Reinbacher, A. Shekhovtsov, and T. Pock, "End-to-end training of hybrid CNN-CRF models for stereo," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jul. 2017, pp. 2339–2348.
[41] W. Williem and I. K. Park, "Deep self-guided cost aggregation for stereo matching," *Pattern Recognit. Lett.*, vol. 112, pp. 168–175, Sep. 2018.
[42] Y. Fu, W. Chen, K. Lai, Y. Zhou, and J. Tang, "Rank-based encoding features for stereo matching," *IEEE MultimediaMag.*, vol. 26, no. 4, pp. 28–42, Oct. 2019.
[43] X.-B. Meng, M. Zhang, Z.-X. Zhang, R. Wang, Z. Geng, and F.-Y. Wang, "Efficient confidence-based hierarchical stereo disparity upsampling for noisy inputs," *IEEE Access*, vol. 7, pp. 4067–4082, Oct. 2019.
[44] L. Chen, L. Fan, J. Chen, D. Cao, and F. Wang, "A full density stereo matching system based on the combination of CNNs and slanted-planes," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 50, no. 2, pp. 397–408, Feb. 2020.
[45] A. Li, D. Chen, Y. Liu, and Z. Yuan, "Coordinating multiple disparity proposals for stereo computation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 4022–4030.
[46] S. N. Sinha, D. Scharstein, and R. Szeliski, "Efficient high-resolution stereo matching using local plane sweeps," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2014, pp. 1582–1589.
[47] Q. Chang and T. Maruyama, "Real-time stereo vision system: A multi-block matching on GPU," *IEEE Access*, vol. 6, pp. 42030–42046, Jul. 2018.
[48] Y. Hu, W. Zhen, and S. Scherer, "Deep-learning assisted high-resolution binocular stereo depth reconstruction," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2020, pp. 467–474.
[49] C. Stentoumis, L. Grammatikopoulos, I. Kalisperakis, and G. Karras, "On accurate dense stereo-matching using a local adaptive multi-cost approach," *ISPRS J. Photogram. Remote Sens.*, vol. 91, pp. 29–49, May 2014.
[50] R. Brandt, N. Strisciuglio, N. Petkov, and M. H. F. Wilkinson, "Efficient binocular stereo correspondence matching with 1-D max-trees," *Pattern Recognit. Lett.*, vol. 135, pp. 402–408, Jul. 2020.
[51] S. S. A. Razak, M. A. Othman, and A. F. Kadmin, "The effect of adaptive weighted bilateral filter on stereo matching algorithm," *Pattern Recognit. Lett.*, vol. 8, no. 3, pp. 2249–8958, 2019.
[52] Q. Yang, "A non-local cost aggregation method for stereo matching," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2012, pp. 1402–1409.
[53] F. Cheng, H. Zhang, M. Sun, and D. Yuan, "Cross-trees, edge and superpixel priors-based cost aggregation for stereo matching," *Pattern Recognit.*, vol. 48, no. 7, pp. 2269–2278, 2015.
[54] S. Zhu, Z. Wang, X. Zhang, and Y. Li, "Edge-preserving guided filtering based cost aggregation for stereo matching," *J. Vis. Commun. Image Represent.*, vol. 39, pp. 107–119, Aug. 2016.
[55] Z. Chen, X. Sun, L. Wang, Y. Yu, and C. Huang, "A deep visual correspondence embedding model for stereo matching costs," in *Proc. Int. Conf. Comput. Vis.*, Dec. 2015, pp. 972–980.
[56] A. Li and Z. Yuan, "Occlusion aware stereo matching via cooperative unsupervised learning," in *Proc. Asian Conf. Comput. Vis.*, Cham, Switzerland, May 2018, pp. 197–213.
[57] A. Geiger, M. Roser, and R. Urtasun, "Efficient large-scale stereo matching," in *Proc. Asian Conf. Comput. Vis. Comput. Vis.*, Nov. 2010, pp. 25–38.
[58] R. Liu, C. Yang, W. Sun, X. Wang, and H. Li, "StereoGAN: Bridging synthetic-to-real domain gap by joint optimization of domain translation and stereo matching," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2020, pp. 12754–12763.
[59] D. Hirner and F. Fraundorfer, "FC-DCNN: A densely connected neural network for stereo estimation," in *Proc. IEEE Int. Conf. Pattern Recognit.*, Jan. 2021, pp. 2482–2489.
[60] Z. Lu, J. Wang, Z. Li, S. Chen, and F. Wu, "A resource-efficient pipelined architecture for real-time semi-global stereo matching," *IEEE Trans. Circuits Syst. Video Technol.*, early access, Feb. 6, 2021, doi: 10.1109/TCSVT.2021.3061704.
[61] P. Liu, I. King, M. R. Lyu, and J. Xu, "Flow2Stereo: Effective self-supervised learning of optical flow and stereo matching," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2020, pp. 6647–6656.
[62] O. Zeglazi, M. Rziza, A. Amine, and C. Demonceaux, "Accurate dense stereo matching for road scenes," in *Proc. IEEE Int. Conf. Image Process.*, Sep. 2017, pp. 720–724.
[63] K. Zhang, Y. Fang, D. Min, L. Sun, S. Yang, S. Yan, and Q. Tian, "Cross-scale cost aggregation for stereo matching," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2014, pp. 1590–1597.

CHUNBO CHENG received the B.Sc. degree in mathematics and applied mathematics from Yangtze University, Jingzhou, China, in 2015. He is currently pursuing the Ph.D. degree with the School of Mathematics and Statistics, Huazhong University of Science and Technology, Wuhan, China. His research interests include pattern recognition, machine learning, and computer vision.

WENJING CUI received the B.S. degree from Henan Normal University, Henan, China, in 2015, and the M.S. degree from Wuhan University of Technology, Wuhan, China, in 2018. She is currently an Assistant with the College of Mathematical and Physical Sciences, Hubei Polytechnic University. Her research interests include machine learning and econometric analysis.