FreeScup: A Novel Platform for Assisting Sculpture Pose Design

Yirui Wu, Tong Lu, *Member, IEEE*, Zehuan Yuan, and Hao Wang

***Abstract*—Sculpture design is challenging due to its inherent difficulty in characterizing artworks quantitatively; thus, few works have been done to assist sculpture design in the past decades in the multimedia community. We have cooperated with several sculptors on analyzing styles of different artists consisting of Giacometti, Augeuste Rodin, Henry Moore, and Marino Marini from which we find pose editing plays an important role in sculpture design. Motivated by this, we present a novel platform that allows sculptors to edit virtual three-dimensional (3-D) sculptures by a free way. The proposed platform consists of three modules, namely, *sculpture initialization*, *sculptor-sculpture mapping*, and *interactive pose editing*. In sculpture initialization, a virtual 3-D sculpture is first incrementally reconstructed from multiview images. Then, we define Laplace operator and its corresponding spectrum to describe the geometry information of the reconstructed sculpture. During sculptor–sculpture mapping, we apply spectral analysis on the low-frequency parts of the spectrum to search for candidate editing points on the surface of the sculpture. Next, body actions of the sculptor are captured by Kinect and further mapped onto editing points as a predefined configuration set. Finally, during interactive pose editing, a real-time Kinect-driven sculpture pose editing scheme is presented, which not only preserves geometry features of the sculpture but also allows instant changes of sculpture poses. We demonstrate that our platform successfully assists sculptors on real-time pose editing by comparing its performance with those of the existing sculpture assisting methods.**

***Index Terms*—Artistic design, FreeScup, pose editing, sculpture, spectral.**

I. Introduction

DISPLAYING sculpture artworks in public locations where thousands of people can see is the dream of most sculptors [1]. It is true that sculptors have benefited from the recent technologies such as 3D geometric modeling [2], artwork retrieval [3],[4], 3D visualization [5], and 3D printing [6],[7] in producing various sculpture artworks. For example, in sculpture modeling, precisely defined and highly optimized shapes following a clear underlying logic [8] are used in abstract sculpture design, which results in several sculpture families with small physical maquette or large-scale sculpture design [9]. Human-centered interaction techniques have also been adopted, for example, in the "*See me, Feel me, Touch me, Hear me*" project [1], a trajectory crossing sculptures is crafted by combining textual and audio instructions to drive directed viewing, movement and touching in a sculpture garden. In this sense, artists are benefiting from the rapidly developing computer techniques and various multimedia devices.

However, few works have been proposed to assist sculpture design in the multimedia community due to the inherent difficulty in characterizing artworks quantitatively. Therefore, real-life sculpture design now still remains to be a tedious work for most sculptors [9]. In the past years, we have cooperated with several sculptors on developing new tools for assisting sculpture design, especially aiming at simplifying the current artwork design processes to inspire more fantastic ideas. Generally, the traditional sculpture design consists of the following 6 steps from conception, initialization, drafting, molding, coloring to finalization (see Fig. 2):

1) *Step 1: Conception*. A sculptor has to find inspirations for artistic design in this step. In most cases, the sculptor may utilize pen and paper to write down conceptions without the help of any multimedia techniques.
2) *Step 2: Initialization of artwork design*. Once the sculptor has some ideas in designing a sculpture, he/she starts to initialize different aspects of the artwork, such as style, material, size and shape. In this step, the sculptor will initialize his/her ideas to view the artwork in mind through any painting software or directly drawing it on papers.
3) *Step 3: Sculpture drafting*. In this step, the sculptor has to draft a small-scale clay/stone/wood/metal sculpture to decide whether the artwork satisfies himself/herself. In practice, such a physical sculpture has to be modified quite a lot times until it meets all the requirements of artistic design.
4) *Step 4: Molding*. After drafting, the sculptor has to further mold the full-size physical sculpture for viewing.
5) *Step 5: Coloring*. In this step, the sculptor strengthens the artistic style of the full-size sculpture through coloring and texturing.
6) *Step 6: Finalizing the art design*. After all the previous steps, the sculptor finishes artwork design after necessary minor revisions.

Among these steps, we find Step 3, namely, sculpture drafting, is particularly inefficient and time-consuming since even a tiny pose modification often requires re-drafting the whole sculpture

Manuscript received February 29, 2016; revised July 8, 2016; accepted September 11, 2016. Date of publication September 14, 2016; date of current version December 14, 2016. This work was supported by the Natural Science Foundation of China under Grant 61672273, Grant 61272218, and Grant 61321491, and by the Science Foundation for Distinguished Young Scholars of Jiangsu under Grant BK20160021. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Sen-Ching Samson Cheung.

Y. Wu, T. Lu, and Z. Yuan are with the National Key Lab for Novel Software Technology, Nanjing University, Nanjing 210046, China (e-mail: wuyirui1989@163.com; lutong@nju.edu.cn; zhyuan001@gmail.com).

H. Wang is with the Institute of Deep Learning, Baidu, Beijing 100085, China (e-mail: wanghao0711@gmail.com).

This paper has supplementary downloadable material available at http://ieeexplore.ieee.org. This includes a video demo to show how a sculptor uses the proposed platform to edit sculpture poses by a free and real-time way. This material is 37.6 MB in size.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TMM.2016.2609407

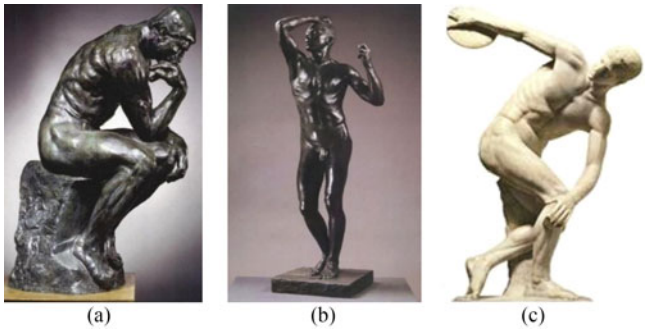


Fig. 1. Different poses of sculptures generally lead to different meanings: (a) *The Thinker* produced by Rodin, (b) *The Age of Bronze* produced by Rodin, and (c) *Discobolos* produced by Myron.

using clay, stone, wood, metal or other materials. Now some sculptors use 3D modeling software such as ZBrush and Maya in this step to avoid re-drafting physical sculptures; however, these tools are still too complicated for most artists to learn.

Essentially, re-drafting in Step 3 is a process of changing the artistic style of the sculpture. With the help of the sculptors, we analyze different styles from classic sculpture artworks produced by Giacometti, Augeuste Rodin, Henry Moore and Marino Marini. We find a fact that sculpture pose plays an important role in distinguishing different sculpture styles. For illustration, the sculptures produced by Giacometti often share the shame shape property of long or thin, symbolizing humans are burned out in wars. For comparison, the shapes of the sculptures produced by Rodin are often plump, implying humans have stepped into a new age of civilization. Fig. 1 gives some examples to show the importance of pose editing in sculpture design, where unseen pressures are hidden in (a) through the shrinking body pose, (b) is relax and stretchable, implying humans have been liberated from an uncultured society, while (c) represents the vitality of humans through a nearly symmetrical "S"-like body pose.

Inspired by above discussions, we decide to utilize multimedia technologies/devices to provide efficient and intuitive tools for sculptors. To decrease the re-drafting time in Step 3, starting with a virtual sculpture shape and then providing tools for assisting free pose editing on it is feasible. That is, we aim at providing a novel platform to assist re-drafting in Step 3 by first initializing the virtual 3D shape of a sculpture, and then providing real-time tools for the sculptor to edit the poses of the virtual sculpture freely. We utilize a RGB-D camera to capture body actions of the sculptor to drive pose editing on the virtual sculpture as shown in Fig. 3, where one can see different poses can be produced for viewing quickly. This helps inspire more innovative artistic ideas for the sculptor through a free and efficient way.

In this paper, we propose a novel sculpture platform named *FreeScup* to assist free sculpture pose design in real time. Fig. 4 shows the workflow of the proposed platform, which consists of three modules, namely, *sculpture initialization*, *sculptor-sculpture mapping*, and *interactive pose editing*. The *sculpture initialization* stage is composed of Fig. 4(a) and 4(b). During this stage, a virtual 3D sculpture is first obtained by either using an existing 3D reconstruction approach from multi-view images or a 3D scanner. Next, we propose to use Laplace operator and its corresponding spectrum to represent the reconstructed 3D geometry for further pose editing. The *sculptor-sculpture mapping* stage includes Fig. 4(c)–(e), where candidate editing points on the surface of sculpture are first automatically searched, and then body joints of the sculptor are manually mapped onto particular editing points as a pre-defined configuration set, which is used to bind 21 body joints of the sculptor with specific editing points on the sculpture to drive further free pose editing. The *interactive pose editing* stage is shown in Fig. 4(f), where we convert the problem of sculpture pose editing to the minimization of local geometry features represented by Laplace coordinates. To solve the minimization problem in real time, we construct a Kinect-driven reduced model based on the spectrum of the sculpture to describe the propagations during pose editing.

The main contribution of the paper is to propose a new sculpture art design assisting platform that supports free sculpture pose modifications. The proposed platform provides artists an efficient approach to edit the poses of sculpture artworks. By this way, we believe human burden can be reduced to a certain extent, which helps sculptors focus more on finding fantastic artwork ideas during their sculpture design steps. Our experimental results and the comparison results demonstrate the effectiveness of the proposed platform. To the best of our knowledge, this is the first work towards assisting sculpture artistic design by using multimedia techniques and devices.

The rest of the paper is organized as follows. Section II gives an overview of the related work on sculpture design. The initiation of sculpture design is introduced in Section III. Then the mapping from sculptor body joints onto editing points is introduced in Section IV. In Section V, the details of real time Kinect-driven sculpture pose editing are discussed. Section VI shows our experimental results, and finally Section VII concludes the paper.

II. Related Work

Few works have been done to assist sculpture design in the past decades according to our review on the literature as introduced. Existing multimedia techniques related with sculpture design can be roughly categorized into three classes, namely, sculpture modeling, sculpture editing and human-centered interactions.

Sculpture modeling: Sculpture modeling is widely used in sculpture design, which provide methods in initializing virtual 3D sculptures for further processing. The existing methods of sculpture modeling can be further classified into three categories: manual construction with 3D modeling software (e.g., ZBrush and Maya) [10], the reconstruction with 3D scanning devices [11], [12], and image-based 3D reconstruction algorithms [13], [14]. Note that the latter two kinds rely on the existence of a physical sculpture for reconstruction. In these methods, 3D modeling software always bring a lot additional learning costs and sometimes are too complicated for artists. Scanners are widely used for 3D modeling by estimating depth information

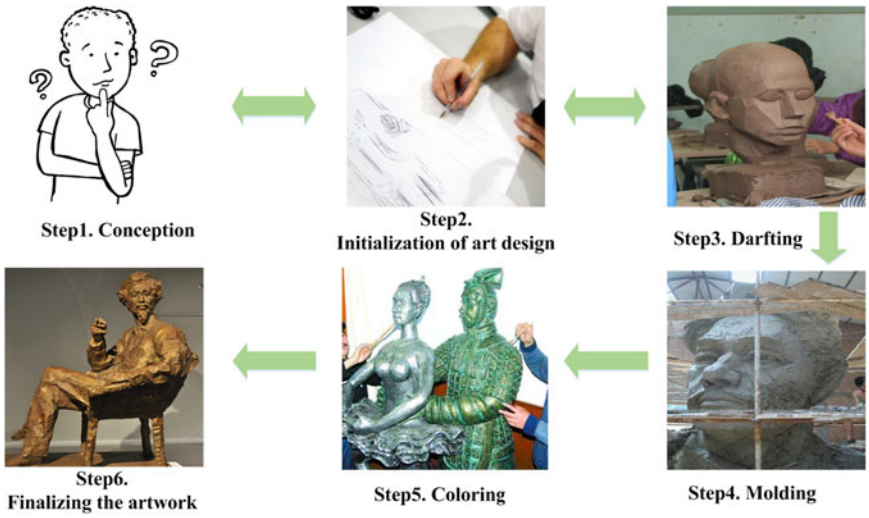


Fig. 2. Traditional way of sculpture design. (1) Conception, (2) Initialization of artwork design, (3) Sculpture drafting, (4) Molding, (5) Coloring, and (6) Finalizing the artwork.

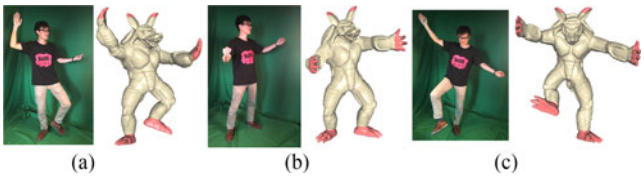


Fig. 3. Pose editing examples on a virtual sculpture driven by body actions of the sculptor: changing the original pose to (a) a Kungfu like pose, (b) a Hug like pose, and (c) a kicking pose.

for each pixel of a physical sculpture through projecting either laser or structured-light onto the sculpture. Note that a laser-based scanner device (e.g., lidar) often requires a relatively long time to generate a high precision 3D model [15] for either outdoor or indoor sculptures. For comparison, the reasonably priced structured-light-based scanners (e.g., Kinect) can now instantly produce 3D models [11] for small-size indoor sculptures. Instead, image-based 3D algorithms utilize multi-view-stereo (MVS) methods [16], [17] to reconstruct dense and detail-abundant surface models from a moderate number of calibrated multi-view images.

Sculpture editing: There are different methods for assisting sculpture editing. Some methods focus on the combination of symbolical sculptures. For example, Séquin *et al.* [2] introduce a tool to help the conception and realization of large-scale bronze geometrical sculptures, which not only allows sculptors to scale each sculpture on different sizes, but also facilitates the design of new sculptures that lie in the same conceptual family. Some other methods aim at shape editing of virtual 3D sculptures, which first convert the given sculpture into a set of controllers such as cage [18] and skeleton [19] embedded in the sculpture itself, and then define the transformation from the controllers to the surface of the sculpture for 3D editing. One problem of such methods is artifacts are sometimes brought due to the fact that local artistic characteristics are not preserved [20]. Such methods are actually designed for professional users with complicated tools that concentrate on geometric editing but not for sculptor artists.

As an alternative of embedded controllers, some other methods formulate sculpture editing as a global variational minimization problem to preserve local artistic characteristics, which are constrained by intuitive controllers defined on surface directly. For example, differential coordinates such as Laplacian coordinates [21], [22] and gradient fields [23], [24] can be adopted to describe particular artistic characteristics. However, a relatively long set-up and updating cost is often required to solve such minimization problem. Therefore, it is difficult to provide an intuitive and real-time framework for sculptors. Au *et al.* [25] construct a reduced model for the minimization problem by sampling isolines to achieving a low updating cost. However, the size of their reduced model is linear with the number of controllers, which leads to a long iterative time if the sculptor sets a lot controllers.

Human-centered interactions: Human-centered interactions have been widely used in sculpture-based applications to help sculpture design or viewing. For example, Barmpoutis *et al.* [26] propose a sculpture-retrieval system to search for sculptures that have specific poses from the Graeco-Roman Sculpture Artwork Dataset by inputting different poses of human bodies captured by a depth sensor. Considering the devices used in interactions, we further classify these methods into the following two categories, namely, tangible methods and stereo vision methods.

Tangible user interfaces [27], [28] intuitively help users edit poses. For example, Jacobson *et al.* [29] develop an articulated device to allow fine-grained controls over a lot degrees of freedom on a character, which results in desirable pose editing results. However, their method requires particular compositions of mechanical parts of the character, which limits the use in other applications.

Stereo vision devices such as Kinect [30] and Intel RealSense do not require additional settings for editing. Moreover, stereo vision methods share the properties of low-cost and free user interfaces that can be controlled by users easily. Now stereo

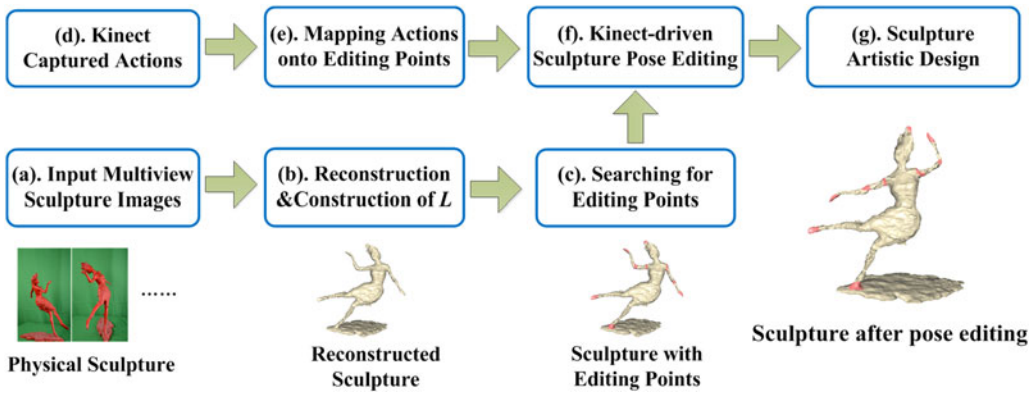


Fig. 4. Proposed platform for assisting sculpture design: (a) inputting a set of multi-view sculpture images, (b) reconstructing a virtual 3D sculpture and calculating its Laplace operator L, (c) searching for candidate editing points on the surface of the sculpture, (d) using Kinect to capture body actions of the sculptor, (e) mapping body actions onto the surface of the sculpture by a pre-defined configuration set, (f) Kinect-driven sculpture pose editing, and (g) obtaining pose editing results.

vision devices are used by a lot real-life applications as input devices. For example, the 3D puperty project [31] offers a natural interface that allows users to create animation stories by directly performing with their puppets before Kinect. Some more real-life systems can also be referred in [32] and [33]. Note that such methods are not well-suited for pose editing of non-humanoid characters. To solve the problem, Seol *et al.* [34] propose the combination of different feature mapping functions to enable natural transformation from human body actions to control handles of non-humanoid characters.

III. Initialization of Sculpture Design

In this section, we first give brief discussions on the reconstruction of a virtual sculpture based on our method presented previously, then we show how to construct Laplace operator and its corresponding spectrum for pose editing with details.

A. Reconstruction of 3D Virtual Sculptures

On the proposed platform, the sculptor can either directly use an existing virtual sculpture or build his/her a virtual sculpture using an existing 3D reconstruction algorithm from multi-views. The latter has a relatively lower cost and higher convenience for artists especially compared with geometric modeling methods. To achieve desirable quality of reconstruction, we first perform co-segmentation [35] on multi-view images to separate the sculpture from background. In other words, we utilize visually similar foreground visual features of multi-view images for segmentation in this step by calibrating multi-view images [36] and constructing a two-label MRF framework to assign label 1 to the sculpture and 0 to its background. Note that the global appearance smoothness, the penalty to local different labels, and the objectness measurement in [37] are involved. After calibrating and segmenting, we adopt our previous incremental 3D reconstruction method [14] to generate the virtual sculpture $\mathcal{M}$ over these multi-view images. That is, we first roughly initialize the sculpture by a set of uniformly sampled multi-view images, and then incrementally update the sculpture by involving more multi-view images to minimize the energy function to guarantee the quality of reconstruction. More details can be referred in [14].

B. Constructing Laplace Operator and Its Spectrum

In this subsection, we show how to construct the Laplace operator and its corresponding spectrum for the virtual sculpture. Our goal is to construct Laplacian coordinates to represent both the local characteristics and the global shape of the sculpture using the components of the spectrum of the Laplacian matrix for further pose editing.

Specifically, for the sculpture reconstructed from multi-view images, we first perform post refinements to reduce artifacts through filling holes and removing noises. We then represent the virtual sculpture as a graph of $\mathcal{M} = (X, E)$ with a vertex set X and an edge set E, where $X = [x_1^T, x_2^T, ..., x_{n_{\text{ver}}}^T]^T, x_i \in \mathcal{R}^3$, while n_{ver} denotes the number of the vertices on the virtual sculpture. Supposing δ_i gives the Laplacian coordinate of x_i, the result of applying the discrete Laplacian operator to x_i is

$$\delta_i = \sum_{\{i,j\}\in E} w_{\text{ij}}(v_j - v_i) = \left[\sum_{\{i,j\}\in E} w_{\text{ij}} v_j\right] - v_i \qquad (1)$$

where $\sum_{\{i,j\}\in E} w_{\text{ij}} = 1$, and w_{ij} are the weights of δ_i. We thus adopt a cotangent weighting scheme [38], which gives an approximation for local characteristics such as surface normals, curvatures and local triangle shapes. The cotangent weights are respectively represented by $w_{\text{ij}} = \frac{\varpi_{\text{ij}}}{\sum_{\{i,k\}\in E} \varpi_{\text{ik}}}$ and $\varpi_{\text{ij}} = \cot\alpha + \cot\beta$, where α and β are the two angles opposite to edge $\{i, j\}$. We use an $n \times n$ Laplacian matrix L^c defined as follows to obtain Laplacian coordinates for the sculpture:

$$\mathbf{L}_{\text{ij}}^c = \begin{cases} -1 & i = j \\ w_{\text{ij}} & (i,j) \in E \\ 0 & \text{otherwise.} \end{cases} \qquad (2)$$

Next, we compute x, y and z coordinates of Laplacian coordinates $\sigma(X), X \in \{x, y, z\}$ separately by

$$\sigma(X) = L^c X. \qquad (3)$$

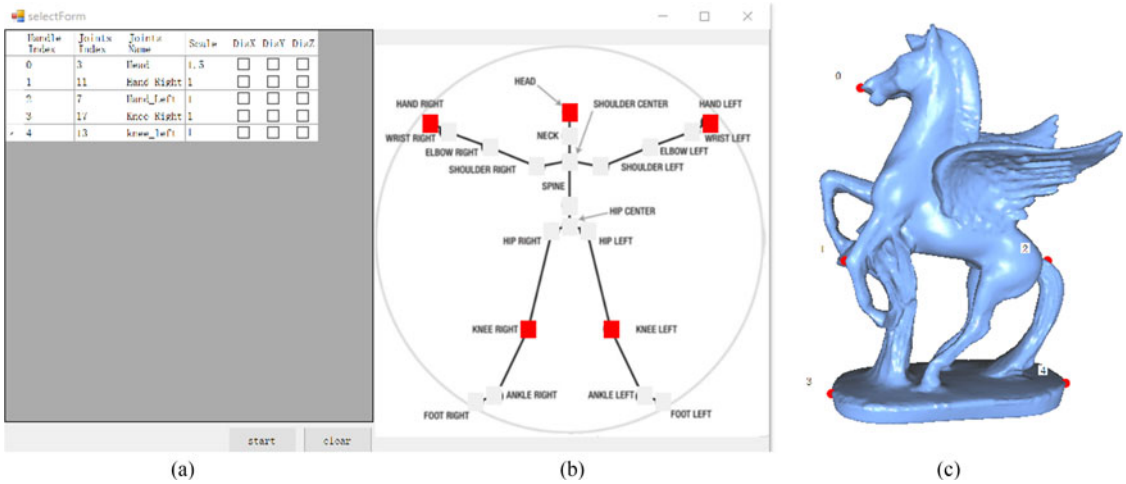


Fig. 5. User interface to customize the mapping from Kinect captured body joints to the editing points on the surface of sculpture.

After constructing the Laplacian matrix L^c for the virtual sculpture, we compute eigenvectors of L^c using the ARPACK solver. We sort normalized non-zero eigenvectors $[\phi_1, \phi_2, \ldots]$ with an increasing order of eigenvalues of $[\lambda_1, \lambda_2, \ldots]$. Each eigenvector ϕ_i represents a scalar function on the surface of the sculpture with the value ϕ_i^v assigned to each vertex v. Essentially, the eigenvectors of the Laplacian matrix form a spectrum, which are regarded as 'Shape DNA' representing the geometry of the virtual sculpture on different scales [39]. Among the spectrum, we focus on its low-frequency components, namely, the eigenvectors with smaller eigenvalues. This is due to the fact that low-frequency eigenvectors can smoothly describe the global shape [40] and thus a smooth sculpture can be approximately characterized using the first several eigenvectors. Based on the experiments on a lot sculptures, we find generally the first 15 eigenvectors $(n_e = 15)$ are sufficient to represent geometry information of sculptures. As a result, we utilize the constructed Laplacian spectrum $\{\phi_i\}_{i=1}^{n_e}$ as low-frequency geometry features in searching for pose editing points and freely editing different styles of sculpture poses. Note that during pose editing, the high-frequency details represented by Laplacian coordinates of the sculpture will be well preserved.

IV. Mapping Sculptor Actions Onto Sculpture Artwork

In this section, we first apply spectral analysis on the low-frequency Laplacian spectrum to search for candidate editing points on the surface of the sculpture. After that, we allow the sculptor to customize the mapping from his/her body joints to these editing points. Then during interactive pose editing, body actions captured by Kinect will be transformed to the editing points to drive pose editing.

A. Searching for Editing Points

For free pose editing, we need first automatically select proper points as handles for editing. Through the experimentations on selecting different types of editing points on sculpture artworks, we basically summarize the following two criteria to guide the automatic discovery of editing point candidates:

1) Protrusion or prominent regions of the sculpture are essential since actions on these places often lead to different poses.
2) Joint or junction regions of the sculpture are also essential since these regions reflect the range of propagation from one part to another on the surface of the sculpture during pose editing.

Base on these two criteria, for a given sculpture with any shape, we search for its candidate editing points automatically in the low-frequency components $\{\phi_i\}_{i=1}^{n_e}$ as follows:

1) *Searching for extreme points using eigenvectors:* Due to the intrinsic harmonic behavior of eigenvectors, the point that has a locally maximum/minimum value in each eigenvector usually locates in a protrusion or prominent region of the sculpture.
2) *Searching for saddle points:* Take a humanoid sculpture as an example, saddle points most probably appear at junction regions like human limbs. Therefore, a saddle point here is a stationary one in eigenvectors, the value of which is the minimum one along one axial direction and remains the maximum on the crossing axis at the same time.
3) *Searching for the maximum points in the gradient field of eigenvectors:* The gradients values of eigenvectors, i.e. $\{\mathbf{g}_i\}_{i=1}^{n_e}$, is regarded as the fields defined on the sculpture manifold. Essentially, the gradient field is sensitive in indicating local variations on the surface of the sculpture. A point with a large value in the gradient field implies a rapid variation, which is usually considered as a joint on the sculpture.

Fig. 6(a) shows some results of candidate editing points on a Neptune sculpture, where we can find that most of the candidate editing points are on joint or junction parts. This is valid since it coincides with our expect. Manual interactions are also provided by our platform to add/delete/drag editing point candidates on the sculpture. As a result, we obtain a set of verified editing

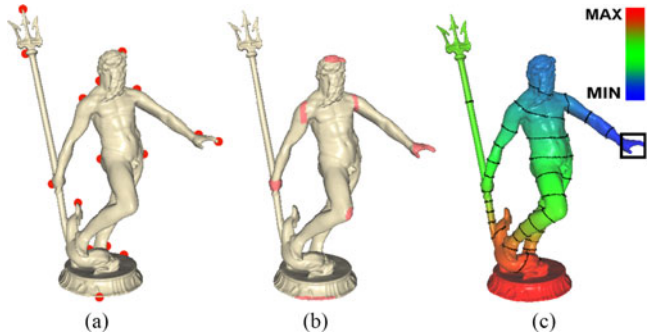


Fig. 6. Example of Kinect-driven pose editing: (a) automatically searched editing point candidates on the sculpture, (b) extended editing regions from the editing points, and (c) an example of action propagation from the left hand to the whole sculpture in one eigenvector, where eigenvector values from the maximum to the minimum are shown by different colors from red to blue.

points of $H = \{h_j\}_{j=0}^{n_s}$, where n_s represents the number of candidate points.

B. Mapping Sculptor Body Joints Onto Editing Points

Since sculptor actions captured by Kinect are represented by a combination of human joints, the sculptor is required to customize the mapping between body joints and editing points before editing. Based on the definition of body actions by Kinect, we represent sculptor actions as a time-series structure of 21 body joints, including positions in X-Y plane, depth information and joint orientations. Note that the number of 21 here refers to the predefined maximal types of body joints that Kinect can recognize. Considering that there are no exact correlations between the parts of a non-humanoid sculpture and human body joints, we allow the sculptor to customize the mapping through the user interface as shown in Fig. 5, which gives an example of mapping sculptor joints onto the surface of a non-humanoid Pegasus sculpture. Fig. 5(b) allows the sculptor to directly click on the Kinect-captured joint image to choose joints, while Fig. 5(c) allows to view, select, add or delete editing points from the automatically searched candidates. After defining joints and editing points, Fig. 5(a) allows to bind joints with particular editing points. Note that the proposed platform supports pose editing of both humanoid and non-humanoid sculptures. We show some pose editing results of non-humanoid sculptures produced by our platform in the second column of Fig. 11. Note that our platform supports pose editing for both humanoid and non-humanoid sculptures through manually mapping body joints onto specific editing points. For illustration, we show a non-articulated sculpture example in Fig. 7, which is produced on our platform by transforming joint actions of the sculptor onto the non-articulated sculpture with a configured mapping set.

During interactive pose editing, we notice that the input of sculptor actions derived from Kinect sensors may be inaccurate due to occlusions or outrange of sensor. Therefore, we further propose the following two filters on the captured raw data to stabilize sculptor actions:

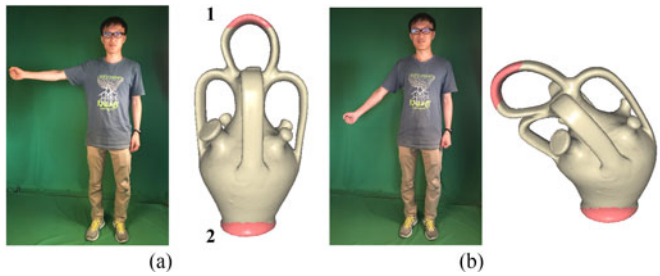


Fig. 7. Non-articulated sculpture pose editing example produced by the proposed platform: (a) the initialization of a vase sculpture and (b) the result of pose editing driven by sculptor body actions. Note that the editing region labeled with '1' is mapped to joint 'Hand Right', while '2' is mapped to joint 'Foot Right'.

1) *Holt-Winters double exponential smoothing filter:* This filter applies Holt-Winters double exponential smoothing to historical joint positions and orientations to get the predictions on upcoming action data under a reasonable assumption that there exists a trend in the captured action data. By the interpolation between the prediction and upcoming action data, this filter stabilizes joint locations and orientations to remove most jitters and noises brought by Kinect.
2) *Limbs inferring filter:* This filter is applied to the positions of occlusive limbs, aiming at preventing the jumpy of limbs. Kinect can provide rough predictions for clipped limb joints; however, the inference may be incorrect since it is based on a limited depth image. We thus linearly interpolate the previous smoothed joint positions and the inferred positions to predict convinced positions for clipped limbs by this filter.

After filtering, we compute the actions respected to the kth joint as

$$\begin{cases} d_k^{t+1} = \alpha_k \cdot g(f(p_k^{t+1}) - f(p_k^t)) \\ r_k^{t+1} = \text{orth}(f(o_k)^{t+1} * f(o_k^t)^{-1}) \end{cases} \tag{4}$$

where d_k^t and r_k^t represent the difference of joint positions and the transformation matrix of joint orientations at time t, respectively. p_k^t and o_k^t respectively denote the position in x, y, z plane and the orientation described by a group of unit quarternion, α_k is the combination of scale and discard?> parameters, function $f()$ denotes the discussed filters, function orth() represents orthogonal normalization, which is designed to transform the original orientation transformation to a rigid one [41], and function $g()$ represents the action reshape transform, which converts actions from the Kinect camera space to the sculpture space based on the size portion of the Kinect input image and the sculpture. Note that the time interval between t and $t+1$ which is named as one iteration equals to the updating frequency of Kinect sensor (set as 30Hz).

The mapping process from joint actions of to editing points is represented as a conversion matrix $\text{Map}_{n_h \times 21}$, where the entry at the jth row and the kth column of Map is set to 1 if the sculptor define the mapping between the jth editing point and the kth body joint, otherwise it is set to 0. Formally, we compute

the actions of the editing points in the following matrix form as

$$R^t = \text{Map}_{n_p \times 21} * S^t_{21\times 4}, \tag{5}$$

where S^t is constructed from the set of joint actions $\{d^t_k\}^{21}_{k=1}$ and $\{r^t_k\}^{21}_{k=1}$, and each row of R^t represents the actions of one specified editing point h_j, including translations d^t_j and 4×4 rotation transformation matrix r^t_j.

V. Real-Time Kinect-Driven Sculpture Pose Editing

In this section, we first transform the problem of pose editing to the minimization of local artistic features. Next, we propose a Kinect-driven reduced model to achieve real-time pose editing based on the low-frequency Laplacian spectrum. Finally, we present an iterative Laplacian editing framework by cooperating with our Kinect-driven reduced model to allow free pose editing in real time. The details are as follows.

A. Problem Describing

To preserve abundant surface details of the reconstructed sculpture, we convert the Kinect-driven sculpture pose editing problem to the minimization of time-series variances of local shape features (e.g., curvature, normal and local rigidity) of the sculpture. By encoding local shape features with Laplacian coordinates, we further formulate the minimization as a global optimization problem for solving the proper position of each vertex under the Laplacian coordinates system. After positioning and orientating the mapped editing points, we rewrite (3) to the following linear system to maintain Laplacian coordinates during sculpture pose editing:

$$\begin{aligned} \text{A}X^{t+1} &= b(X^t), \qquad \text{with} \\ A &= \begin{bmatrix} L^D \\ \omega\Phi \end{bmatrix} \qquad \text{and} \qquad b(X^t) = \begin{bmatrix} \sigma(X^t) \\ \omega H \end{bmatrix} \end{aligned} \tag{6}$$

where X^t and X^{t+1} respectively refer to vertex positions before editing and after editing, L^D is the Dual-Laplacian operator [21] constructed from the original sculpture, $\sigma(X^t)$ represents the Laplacian coordinates that depend on X^t, ω is a constant to enforce the soft constraint in [24], H gives the locations of mapped editing points, and $\Phi X^t = H$ indicates that the minimization problem is subject to the constraints of editing points. The resulting sculpture is thus equivalent to solve (6) using the following Gauss-Newton method:

$$X^{t+1} = (A^T A)^{-1} A^T b(X^t). \tag{7}$$

where transformation matrix A is formed by transformations, which change current vertex positions to result positions.

By analyzing (7), we find the computation of each iteration ranging from t to $t+1$ mainly depends on the size of $A^T A$, which can be defined before editing and remained unchanged during editing. $A^T A$ is a sparse matrix with the size up to $n_{\text{ver}} \times n_{\text{ver}}$. This result may be very large especially for complex sculptures (for example, the vertex number for the reconstructed ChinaRed sculpture in Fig. 4 reaches 73539). Therefore, solving (7) to achieve the latest position X is infeasible. Our target is to provide a real time sculpture pose editing platform for sculptors by solving the problems consisting of memory bottleneck, expensive per-iteration cost, and slow convergence. Therefore, we need find an appropriate way to decrease the size of transformation matrix A during sculpture pose editing to reduce the per-iteration computation cost.

B. Kinect-Driven Reduced Model

In this section, we describe how to construct the Kinect-driven reduced model to achieve real-time pose editing. Essentially, the Kinect-driven reduced model $\tilde{A}$ is simplified from transformation matrix A with two aspects, that is, 1) the construction of editing space by a subset of low-frequency Laplacian spectrum, and 2) the construction of a reduced Kinect-driven propagation scheme to describe how actions captured by Kinect are propagated on the surface of the sculpture. Note that the proposed platform provides the sculptor with functions that extend editing point to editing regions formed by neighboring vertices for editing. For illustration, we show an example of editing regions on the Neptune sculpture in Fig. 6(b). Each editing region offers six degrees of freedom for both translation and rotation, while each editing point offers three degrees of freedom for translation only.

Specifically, inspired by the reduced model in [25], [22] and the inherent geometry information of Laplacian spectrum described in Section III-B, we propose to construct a proper and reduced transformation matrix $\tilde{A}$ from the Laplacian spectrum. To construct $\tilde{A}$, we need keep a low information loss from the original Laplacian editing space described by A to reduced editing space described by $\tilde{A}$ with a large number of eigenvectors. Meanwhile, we need decrease the number of adopted eigenvectors to a reasonable size to achieve low computational cost (previously we use $n_e = 15$). Therefore, we choose to keep a balance between information loss and computational cost by selecting a sufficient number of eigenvectors to construct the editing space. Essentially, for the purpose of sculpture pose editing, the main concern of the sculptor lies on the global consistence of the shape of the sculpture. In another word, our reduced transformation matrix $\tilde{A}$ has to characterize the editing space of the sculpture through smooth and global-sensitive eigenvectors.

Considering that the low-frequency components of Laplacian spectrum share the properties like smoothness and global-sense, we select the eigenvectors from the low-frequency Laplacian spectrum $\{\phi_i\}^{n_e}_{i=1}$ to construct the editing space. Specifically, we utilize eigengap, which denotes the difference between two successive eigenvalues $|\lambda_i - \lambda_{i+1}|$, to determine the number of eigenvectors n_e by minimizing the information loss from A to $\tilde{A}$. According to the Davis-Kahan theorem [42], for any symmetric matrix A and its perturbed version $\tilde{A}$, i.e., $\tilde{A} = A + G$, the distance between the original subspace V and the perturbed subspace $\tilde{V}$, which are both formed by the first m eigenvectors, is bounded by $d(V, \tilde{V}) = \|G\|/\delta$, where δ coincides with the eigengap $|\lambda_m - \lambda_{m+1}|$. That is, regarding the potential editing on our constructed cotangent-weight Laplacian matrix L^C as $\tilde{L^C}$, the distance $d(L^C, \tilde{L^C})$, i.e., the information loss from L^C to $\tilde{L^C}$, is determined by eigengap. Therefore, we construct the subspace of L^C with eigenvectors from the low-frequency

Laplacian spectrum, the number of which is determined by the following maximal eigengap:

$$\tilde{V} = \{\phi_1, \ldots, \phi_{n_m} \mid n_m = \arg\max_i |\lambda_i - \lambda_{i+1}|, 1 \le i \le n_e\} \tag{8}$$

where n_m refers to the number of eigenvectors to construct the sculpture editing space.

After constructing the editing space, we further propose a reduced Kinect-driven propagation scheme to describe how actions captured by Kinect propagate on the surface of the sculpture. We assume that the vertices on an isoline always receive the same values of transformation in one eigenvector. We thus uniformly sample the eigenvectors to locate a set of isolines, each of which is represented by a black circle as shown in Fig. 6(c). The transformation associated with any isoline is directly determined by the Kinect captured body actions in our scheme, while the transformation values of the rest vertices, which are not on isolines, are interpolated from the transformation of isolines. Essentially, isolines greatly decrease the size of transformations from vertex number to isoline number during each iteration of sculpture pose editing, which makes per-iteration editing faster and easier to converge.

Supposing the kth joint is mapped to the jth editing point, we calculate the transformation of the lth isoline in the ith eigenvector as

$$\left\{ p_{i,l}^t = R_j^t - \frac{2R_j^t \cdot |\phi_{i,l}^v - \phi_{i,j}^v|}{|2\phi_{i,j}^v - 1| + 1} \mid 1 \le i \le m, 1 \le l \le n_{\text{iso}} \right\} \tag{9}$$

where R_j^t is the jth row of Matrix R formed by the translations d_k^t and rotations r_k^t of the kth joint, $\phi_{i,l}^v$ and $\phi_{i,j}^v$ denote the eigenvector values of the lth isoline and the jth mapped editing point, respectively. We set the number of sampling isolines $n_{\text{iso}} = 15$ since it is a good balance between editing quality and computation cost by experiments. Essentially, (9) regards the ratios of eigenvector values as the ratios of transformations, and properly computes rotations and translations of the triangles set Θ based on Kinect captured body actions, where Θ refers to the triangles that are crossed by isolines in the selected eigenvectors. We show one example of action propagation from left hand to the whole sculpture in Fig. 6(c), where we can notice the isoline that has a smaller eigenvector value (e.g., near the left hand) gets a larger transformation value.

Once transformations of isolines are determined by Kinect captured body actions in (9), we calculate the transformation of vertices based on the transformation of isolines $p_{i,l}^t$ by

$$p_v^t = \sum_{i=1}^{i=n_m} \theta_i * \omega_{i,v} * p_{i,l}^t + \theta_i * (1 - \omega_{i,v}) * p_{i,l+1}^t \tag{10}$$

where v is supposed to locate between the lth and the $(l+1)$th isolines, $\omega_{i,v}$ denotes a linear weight determined by the difference of the eigenvector values between v and its lth neighboring isoline. θ_i denotes the weight of the ith eigenvector with respect to the spectrum. Regarding x-coordinates of the sculpture as a signal and e_i as a basis of the x signal, we compute their coefficients $\beta_{x,i}$ by Eigenspace projections, i.e., $\beta_{x,i} = x \cdot \phi_i$. Similarly, we obtain the coefficients for $y-$ and $z-$coordinates as $\beta_{y,i}$ and $\beta_{z,i}$, respectively. It is a natural way to use the combination of these spectral coefficients to quantify the weight θ_i of eigenvector $\theta_i = \sqrt{\beta_{x,i}^2 + \beta_{y,i}^2 + \beta_{z,i}^2}$.

C. Iterative Sculpture Pose Editing

Based on the proposed Kinect-driven reduced model $\tilde{A}$ and the mapped Kinect-captured action input R^t defined in (5), we rewrite (7) in a matrix form to compute the resulting sculpture:

$$\begin{cases} \tilde{A} = \text{AW} \\ P = \text{ER}^t \\ X^{t+1} = (\tilde{A}^T \tilde{A})^{-1} \tilde{A}^T b(\text{PX}^t) \end{cases} \tag{11}$$

where W is a weight matrix constructed from $\{\theta_i\}$, $\{\omega_{i,v}\}$ and $\{\phi^i\}$, which reduce the original Laplacian space to the spectrum-based and isoline-based editing space. P represents the propagation of transformations from Kinect recognized joints to isolines, where E is a column vector constructed by the set of eigenvector values, i.e., $\{\phi_{i,l}^v\}$ and $\{\phi_{i,j}^v\}$. Function $b()$ computes the local transformation, thus the Laplacian coordinates at time t can properly be changed to fit the orientations of sculpture surface at time $t+1$. The transform process of Laplacian coordinates can also be reduced by sampled isolines. We thus compute the Laplacian coordinates at time $t+1$ as $\sigma(PX^t) = \text{WP}\Psi^t$, where Ψ^t represents Laplacian coordinates at time t.

To solve (11) for obtaining pose editing results, we adopt an iterative Laplacian editing framework [24], [21] to iteratively and alternatively update P and X, which provides intermediate pose editing results to guide artwork design. During updating P, (9) implies the computational cost of updating P is relatively small since the updating of P only involves the triangles in Θ that are crossed by isolines. During updating X^{t+1}, compared with the original transformation size, namely, the whole mesh vertex $n_{\text{ver}} \times n_{\text{ver}}$, we achieve a reduced transformation matrix $\tilde{A}^T \tilde{A}$ with the size of $4n_e n_{\text{iso}} \times 4n_e n_{\text{iso}}$ after the projection of the Kinect-driven reduced model, which significantly speeds the per-iteration updating and convergence rates and results in real-time sculpture pose editions.

During pose editing, the sculptor can instantly view visual appearances of the result sculpture to decide whether it satisfies himself/herself. If the result shape coincides with the artistic idea of the sculptor, he/she can move to the molding step directly to complete the full-size sculpture. With the rapid development of 3D printing [43], [44], the sculptor can also use such a device to convert the result virtual sculpture to a physical one. Actually, the existing 3D printing technologies not only allow sculptors to rebuild sculpture artworks by a convenient and cheap way, but also support creating particular instances of artworks using multi-material printing [6] or multi-style design [45].

VI. Experimental Results

In this section, we show the effectiveness of the proposed platform in helping sculpture pose editing. As shown in Fig. 8, several sculpture pose editing results produced by the proposed

Fig. 8. Proposed platform supports sculpture pose editing by imitating the poses from sculptors. Left: the sculptor's poses. Right: pose editing results of an example sculpture.

platform are illustrated. Experiments on user study for assisting sculpture design, computational cost and quality measuring of the proposed platform are further introduced as follows.

A. User Study for Assisting Sculpture Design

We first conduct user studies to evaluate the effectiveness of our platform for assisting sculpture design. The user studies mainly focus on the efficiency of sculpture construct and sculpture pose editing by comparing the proposed platform with ZBrush version 4R7, which is one of the most popular systems in sculpture design. We tested the performances of the two systems on a laptop with 1.7GHz i5 core2 and 6GB of RAM. The Kinect v2 version was used in our experiments.

Participants: Eight university students majoring in sculpture join our user study experiments. We pay a special attention on the learning cost to use ZBrush and FreeScup by dividing participants into two groups. Four sculptor students who are familiar with ZBrush are arranged into Group A, while the rest who have no experiences in 3D modeling are arranged into Group B. Note that none of them has used the proposed Kinect-driven pose editing platform before experiments.

Tasks: Both groups are first asked to create virtual 3D sculptures based on two physical clay artworks (named as Bear and ChinaRed) with our platform and ZBrush, respectively. Then, they are required to imitate two predefined poses for their created virtual sculptures and two existing sculpture artworks (named as Armadillo and Neptune). In summary, our pose editing experiments involved 8 (participants) $\times$ 2 (systems) $\times$ 4 (models) $\times$ 2 (predefined poses) $=$ 96 trials. Note that the participants are given a short warm-up session to practice a bit before testing.

Performance Measures: We record various types of information of the eight sculptors to quantify the performances, including the number of the vertices of each sculpture n_{ver}, the number of selected editing points n_s, and automatically found point candidates n_c, the working time t_r or interactions n_r (e.g., mouse clicks and keyboard typing) for initializing each sculpture, and the completion time t_i or interactions n_i for pose modifications. Note that the completion time for pose modifications include the cost of conceptions on how to reproduce the given pose.

Results: The results on user studies are shown in Table I, where subscripts z and f denote sculptor students using ZBrush and FreeScup, respectively, and n_c' refers to the number of automatically found point candidates. We thus define the ratio between editing points and automatically found candidates as $P_c = n_c'/n_s$. From Table I, one can find that the proposed platform requires fewer interactions in sculpture pose editing. This is due to the fact that the proposed platform is built on Kinect-based interactions, while ZBrush needs a lot mouse click and keyboard typing operations.

For comparisons, our platform achieves lower durations for sculpture construction $\{t_{r,A_f}, t_{r,B_f}\}$ comparing with $\{t_{r,A_z}, t_{r,B_z}\}$ achieved by ZBrush, which illustrates the effectiveness of the reconstructions of virtual sculptures from multi-view photos. In addition, the sculptors in both Group A and B fail in creating the ChinaRed sculpture by ZBrush since it is too complex for manually creating. Due to the failure in creating ChinaRed sculpture, we have to adopt the ChinaRed sculpture produced by our platform for later pose editing experiments. Note that the time costs of post refinement of reconstructed 3D shapes are included in the completion time here.

We notice that the manually created Bear sculpture only contains nearly 340 vertices, while our platform offers a Bear sculpture with almost 1400 vertices. Essentially, manually creating sculptures is a time-consuming work for inexperienced sculptors, which results in low-quality sculptures with less geometry details. On the contrary, geometry details are related to the resolution and the number of input multi-view images on our platform. Sculptors thus can incrementally add images to our platform to increase the quality of the reconstructed sculpture. Note that the high precision $P_{c,s}$, which is up to nearly 60%, shows that most of the automatically found candidates are selected to induce pose editing.

We also notice that our platform requires less time to complete pose editing $\{t_{i,A_f}, t_{i,B_f}\}$ comparing with the $\{t_{i,A_z}, t_{i,B_z}\}$ achieved by ZBrush, which is represented in the third bin of Fig. 9 as well. During pose editing, we find the most time-consuming step of using ZBrush lies in slightly adjusting sculpture poses, which usually requires quite a lot selection or drag operations on vertices. Another problem is only after adjusting can the sculptors view the appearances of sculptures, which is not good for inspiring fantastic ideas of sculptors during artwork design. On the contrary, the proposed platform enables real-time feedback of pose editing results with the help of Kinect.

With the help of Kinect, the proposed platform offers an intuitive and effective method for pose editing. Moreover, the platform is friendly for sculptors to reduce their learning cost, which is important in artistic design. We represent the learning

TABLE 1
COMPARISON OF USER STUDY RESULTS BETWEEN OUR SYSTEM AND ZBRUSH

Group	Model	$t_r\,(m)$	n_r	n_{ver}	$t_i(m)$	n_i	n_c	n'_c	n_s	$P_c\ (\%)$	Model	$t_i\ (m)$	n_i	n_c	n'_c	n_s	$P_c\ (\%)$
$A_z:1$	Bear	53.6	823	431	10.7	136	-	-	-	-	Armadillo	19.6	395	-	-	-	-
$A_f:1$	Bear	6.0	12	1310	2.0	18	16	4	5	80.0	Armadillo	3.6	15	17	5	6	83.3
$B_z:1$	Bear	91.0	1434	246	18.2	241	-	-	-	-	Armadillo	27.6	527	-	-	-	-
$B_f:1$	Bear	6.2	14	1519	3.1	29	16	5	6	83.3	Armadillo	4.1	17	17	5	8	62.5
$A_z:2$	ChinaRed	*fail*	*fail*	*fail*	23.3	471	-	-	-	-	Neptune	13 6	273	-	-	-	-
$A_f:2$	ChinaRed	20.2	16	35461	4.4	26	25	6	8	75.0	Neptune	2.4	17	16	6	9	66.7
$B_z:2$	ChinaRed	*fail*	*fail*	*fail*	36.7	889	-	-	-	-	Neplune	19.3	348	-	-	-	-
$B_f:2$	ChinaRed	25.4	18	42781	5.1	32	25	7	13	53.8	Neptune	2.8	19	16	6	10	60.0

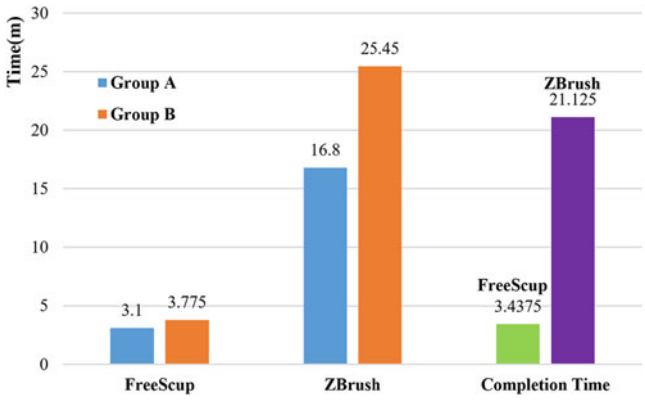


Fig. 9. Comparison of the learning cost and completion time. The first two bins represent the comparison of the learning cost, while the latest bin denotes the comparison of completion time.

costs using difference values of editing durations t_i of Group A and B, namely, $l = |t_{i,A} - t_{i,B}|$. We show comparisons between FreeScup and ZBrush in the first two bins of Fig. 9. For ZBrush, we can notice the editing durations of Group A t_{i,A_z} is much larger than that of Group B t_{i,B_z}, which implies inexperienced sculptors have to spent lots of time in practising their skills for pose editing. For our platform, the learning cost is relatively low since we offer real-time visual feedbacks to help sculptors real time.

B. Computational Cost for Sculpture Pose Editing

In this experiment, we further compare our spectral-based pose editing algorithm with another editing algorithm named as Dual-Laplacian [21]. Dual-Laplacian is a baseline algorithm for 3D pose editing, which also adopts an iterative Laplacian strategy to solve the non-linear Laplacian coordinates minimization problem. We divide the computing times of the two algorithms into the pre-computing time, including pre-computing matrix and eigen-decomposition, and the iteration updating time to solve (6). Table II gives the detailed results of the two methods on a 1.7GHz i5 core2 with 6GB of RAM machine, where subscripts p and i respectively refer to the pre-computing and the iterative updating time, subscripts 1 and 2 refer to the Dual-Laplacian editing method and our method, respectively. The row signed with $*$ represents the computing results by adopting the number of eigenvectors decided by our algorithm.

In Table II, we find the proposed platform achieves almost the same pre-computing time $t_{p,2}$ with that of Dual-Laplacian method $t_{p,1}$. As indicated by (11), matrix $\tilde{A}$, E and W are fixed in each iteration during editing. In other words, these matrices representing the construction of editing space and weights scheme, can be pre-computed and reused for one specific sculpture. We thus store these matrices for reusing to decrease the pre-computation time from $t_{p,2}$ to $t_{p,3}$. The low value of P_{t_p}, defined as a ratio between $t_{p,3}$ and $t_{p,1}$, proves the improvement of efficiency for pre-computing.

By decreasing the transformation size of A from n_{ver} to n_{tri} in each iteration, our method achieves a much lower iterative updating time $t_{i,2}$ than $t_{i,1}$ achieved by Dual-Laplacian method as shown in Table II. Since Kinect captures 30 frames per second, an iterative pose editing method need react within around $1/30$ seconds for each iteration to ensure real-time response. If not, the sculptor may feel a latency after each of his/her interactions. In our experiments, the sculptors felt obvious latency when adopting the Dual-Laplacian method to edit large sculptures such as Chinared and Armadillo since the per-iteration computing time can be up to almost 800ms. Instead, the proposed platform keeps a low per-iteration cost to ensure real-time visual feedback even for detail-abundant sculptures. In Table II, we can find the iterative updating time of Dual-Laplacian method $t_{i,1}$ is insufficient to support real-time feedback, while the proposed platform supports real-time pose editing by a much lower iterative updating time $t_{i,2}$. Note that P_{t_i} is a ratio between $t_{i,2}$ and $t_{i,1}$ to show the improvement in per-iteration computing of the proposed platform.

Another advantage brought by the reduced transformation matrix A is that proposed platform achieves a smaller convergency number $n_{i,2}$, which contributes to real-time editing as well. We define convergency number n_i as the iteration times when the change of gradients reaches a pre-defined threshold. We notice that the Dual-Laplacian method needs a large convergency number $n_{i,1}$ for converging due to the fact that high nonlinearity of $b(X)$ often exists in protruding regions of complex sculptures. For comparison, the proposed platform benefits from the simplification of iteratively updating on transformations and Laplacian coordinates, which results in a much smaller convergency number $n_{i,2}$.

Adopting more eigenvectors means cooperating the high-frequency components of sculpture geometry to construct the editing space, which will be suitable for editing local geometry

TABLE II
PERFORMANCE COMPARISON BETWEEN OUR METHOD AND DUAL-LAPLACIAN EDITING [32]

Model	n_ver	n_tri	n_m	n_c	n_s	$t_{p,1}\ (s)$	$t_{p,2}\ (s)$	$t_{p,3}\ (s)$	$P_{t_p}\ (\%)$	$t_{i,1}\ (ms)$	$t_{i,2}\ (ms)$	$n_{i,1}$	$n_{i,2}$	$P_{t_i}\ (\%)$
ChinaRed	73539	782	**4**	23	8	20.9	11.3	**3.76**	18.0	806	**20.3**	2L3	**29**	2.52
ChinaRed*	73539	1562	**8**	23	8	20.9	18.8	**4.03**	19.3	806	**38.9**	213	**36**	4.83
ChinaRed	73539	2412	**12**	23	8	20.9	31.3	**4.21**	20.1	806	**69.2**	2L3	**42**	8.59
Armadillo	60000	1269	**6**	17	6	17.2	12.7	**3.12**	18.1	928	**28.0**	93	**25**	3.02
Armadillo*	60000	2166	**10**	17	6	17.2	20.2	**3.56**	20.7	928	**50.0**	93	**31**	5.39
Armadillo	60000	3083	**14**	17	6	17.2	35.1	**3.82**	22.2	928	**82.9**	93	**39**	8.93
Neptune	28052	641	**3**	16	8	6.57	3.91	**0.89**	13.5	190	**7.13**	98	**19**	3.75
Neptune*	28052	1096	**5**	16	8	6.57	5.13	**1.05**	16.0	190	**13.1**	98	**22**	6.89
Neptune	28052	1554	**7**	16	8	6.57	5.97	**1.20**	18.3	190	**19.9**	98	**26**	10.5
Bear	10233	963	**8**	16	6	1.04	1.03	**0.231**	22.2	63.0	**9.03**	64	**17**	14.3
Feline	15744	1074	**6**	16	6	2.88	2.43	**0.523**	18.1	118	**12.1**	78	**21**	10.3
Raptor	25102	2975	**13**	20	6	6.31	10.1	**1.68**	26.6	183	**40.0**	143	**33**	21.9
Human	15154	2943	**13**	16	9	5.60	7.43	**0.877**	15.7	98.2	**37.5**	88	**29**	38.3
Elephant	42321	2623	**12**	19	7	14.5	11.7	**3.31**	22.8	682	**44.8**	119	**37**	6.57
Centaur	30001	1737	**8**	21	8	6.37	8.53	**1.28**	20.1	213	**21.6**	103	**28**	10.1
Frog	11206	2295	**10**	15	6	2.03	3.67	**0.439**	21.6	78.3	**25.7**	93	**23**	32.8

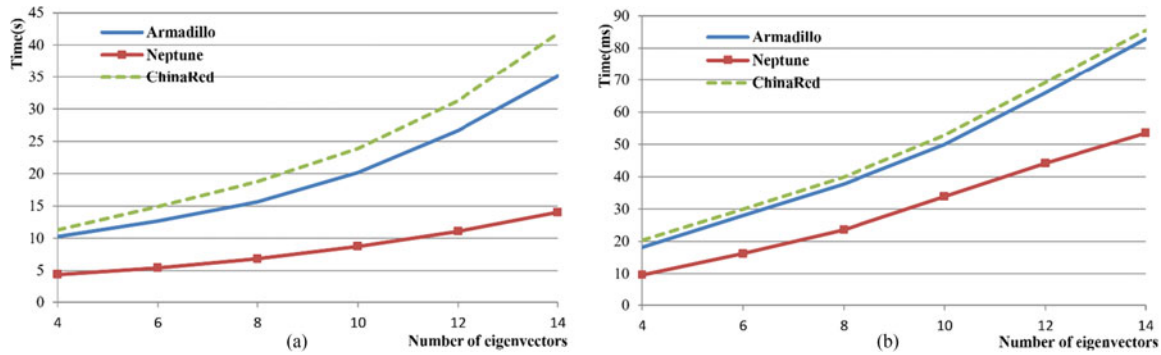


Fig. 10. Comparison of (a) pre-computation time and (b) per-iteration computing time for different numbers of eigenvectors on various meshes.

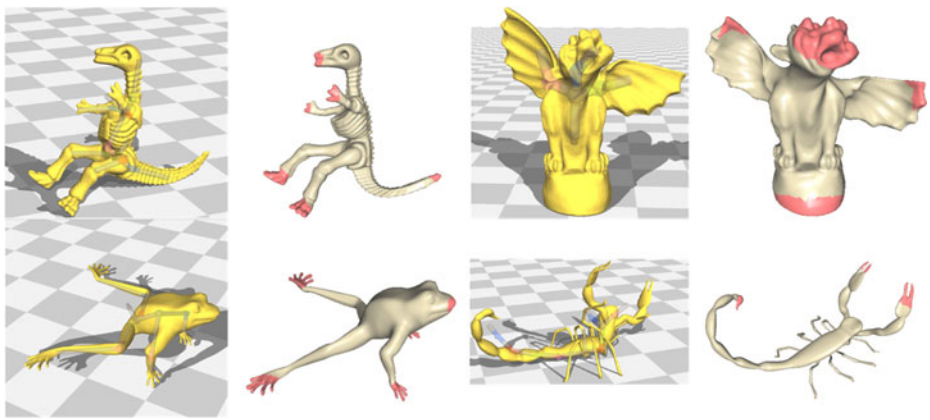


Fig. 11. Comparisons of pose editing results on four different sculptures. For each sculpture model: (left) the result produced by the tangible method in [29] and (right) the result produced by the proposed platform.

details. Due to the fact that in sculpture design people tend to edit the global pose of each sculpture, we discard high-frequency components of Laplacian spectrum. Meanwhile, choosing more eigenvectors requires a large computation cost for both pre-computing and per-iteration updating. Figs. 10(a) and (b) respectively show the comparisons of the pre-computation time $t_{p,2}$ and the iterative updating time $t_{i,2}$ with different numbers of eigenvectors n_m. We find that pre-computing and iterative updating time will increase sharply if given more eigenvectors. Therefore, it is necessary to select a proper number of eigenvectors to construct the editing space, which keeps a reasonable computing time.

C. Pose Editing Quality Analysis

We further compare the qualities of pose editing results respectively using the proposed platform and the tangible method in [29] on the same dataset. As discussed, the tangible method is presented for real-time pose editing based on tangible input devices. We show several pose editing results produced by the proposed platform and the tangible method in Fig. 11, respectively. It can be found that the tangible method constructs transformations to explicitly enforce globally rigid rotations, thus it produces desirable results and preserves volumes well even with rotations. However, to achieve a large degrees of freedom for rotations, skeletons need to be embedded sub-optimally into volumes. Sculptors thus can not well control those poses of small protruding regions such as legs as shown in Fig. 11. Besides, the tangible method requires particular compositions of mechanical parts for each sculpture, which is tedious for sculptors. Instead, the proposed method can edit the poses of small protruding regions by manually specifying editing points on such regions directly. This is because we well preserve geometry details through Laplacian coordinates, and Kinect-captured actions are also properly propagated by geometry-aware low-frequency Laplacian eigenvectors. Therefore, the proposed platform produces desirable pose editing results as respectively shown in Figs. 3, 8, and 11.

VII. Conclusion

In this paper, we present a novel platform named *FreeScup* for assisting sculpture designs. The proposed platform consists of three modules, namely, *sculpture initialization*, *sculptor-sculpture mapping*, and *interactive pose editing*. In sculpture initialization, we first reconstruct a virtual sculpture from multi-view images, and then define Laplacian operator and low-frequency Laplacian spectrum for the reconstructed virtual sculpture. During sculptor-sculpture mapping, candidate editing points on the reconstructed 3D surface are first automatically searched through spectral analysis, and then the mapping from Kinect-captured body joints of the sculptor onto the surface of the virtual sculpture are defined manually. Finally during interactive pose editing, a Kinect-driven sculpture pose editing scheme is proposed to allow for free and real-time sculpture pose editing. User studies and experimental results show the effectiveness in both improving the efficiency and inspiring artistic ideas in sculpture design for the proposed platform. In our future work, we will imitate different kinds of sculpturing operations such as smoothing, pinching, cutting, beating and carving to further assist both humanoid and non-humanoid sculpture designs.

References

[1] L. Fosh, S. Benford, S. Reeves, B. Koleva, and P. Brundell, "See me, feel me, touch me, hear me: Trajectories and interpretation in a sculpture garden," in *Proc. ACM Conf. Human Factors Comput. Syst.*, 2013, pp. 149–158.

[2] C. H. Séquin, "Computer-aided design and realization of geometrical sculptures," *Comput.-Aided Design. Appl.*, vol. 4, no. 5, pp. 671–681, 2007.

[3] S. Philipp-Foliguet, M. Jordan, L. Najman, and J. Cousty, "Artwork 3D model database indexing and classification," *Pattern Recognit.*, vol. 44, no. 3, pp. 588–597, 2011.

[4] M. Alcoverro, S. Philipp-Foliguet, M. Jordan, L. Najman, and J. Cousty, "Region-based 3D artwork indexing and classification," in *Proc. IEEE 3DTV: True Vis., Capture, Transm. Display 3D Video*, 2008, pp. 393–396.

[5] C. H. Séquin, "Rapid prototyping: A 3D visualization tool takes on sculpture and mathematical forms," *ACM Commun.*, vol. 48, no. 6, pp. 66–73, 2005.

[6] D. Chen, D. I. W. Levin, P. Didyk, P. Sitthi-Amorn, and W. Matusik, "Spec2fab: A reducer-tuner model for translating specifications to 3D prints," *ACM Trans. Graph.*, vol. 32, no. 4, pp. 135:1–135:10, 2013.

[7] J. Vanek *et al.*,"Packmerger: A 3D print volume optimizer," *Comput. Graph. Forum*, vol. 33, no. 6, pp. 322–332, 2014.

[8] Carlo H. Séquin, "Sculpture design," in *Proc. IEEE Int. Conf. Virtual Syst. Multimedia*, 2001, pp. 832–843.

[9] C. H. Séquin *et al.*, "Art, math, and computers: New ways of creating pleasing shapes," in *Proc. Bridges Math. Connection Art, Music, Sci. Conf.*, 1998, pp. 1–10.

[10] Z. Ji, L. Liu, and Y. Wang, "B-mesh: A modeling system for base meshes of 3D articulated shapes," *Comput. Graph. Forum*, vol. 29, no. 7, pp. 2169–2177, 2010.

[11] S. Izadi *et al.*, "KinectFusion: Real-time 3D reconstruction and interaction using a moving depth camera," in *Proc. 4th Annu. ACM Symp. User Interface Softw. Technol.*, 2011, pp. 559–568.

[12] D. S. Alexiadis, D. Zarpalas, and P. Daras, "Real-time, full 3-D reconstruction of moving foreground objects from multiple consumer depth cameras," *IEEE Trans. Multimedia*, vol. 15, no. 2, pp. 339–358, Feb. 2013.

[13] F. Remondino and S. El-Hakim, "Image-based 3D modelling: A review," *Photogramm. Rec.*, vol. 21, no. 115, pp. 269–291, 2006.

[14] Z.-H. Yuan and T. Lu, "Incremental 3D reconstruction using Bayesian learning," *Appl. Intell.*, vol. 39, no. 4, pp. 761–771, 2013.

[15] B. Curless, "From range scans to 3D models," *ACM SIGGRAPH Comput. Graph.*, vol. 33, no. 4, pp. 38–41, 1999.

[16] Y. Furukawa and J. Ponce, "Accurate, dense, and robust multiview stereopsis," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 8, pp. 1362–1376, Aug. 2010.

[17] Y. Guo, F. A. Sohel, M. Bennamoun, J. Wan, and M. Lu, "An accurate and robust range image registration algorithm for 3D object modeling," *IEEE Trans. Multimedia*, vol. 16, no. 5, pp. 1377–1390, Aug. 2014.

[18] F. G. García, T. Paradinas, N. Coll, and G. Patow, "*cages: : A multilevel, multi-cage-based system for mesh deformation," *ACM Trans. Graph.*, vol. 32, no. 3, pp. 24, 2013.

[19] X. Shi *et al.*, "Mesh puppetry: Cascading optimization of mesh deformation with inverse kinematics," *ACM Trans. Graph.*, vol. 26, no. 3, 2007, Art. no. 81.

[20] R. Gal, O. Sorkine, N. J. Mitra, and D. Cohen-Or, "Iwires: An analyze-and-edit approach to shape manipulation," *ACM Trans. Graph.*, vol. 28, no. 3, 2009, Art. no. 33.

[21] O. K.-C. Au, C. -L. Tai, L. Liu, and H. Fu, "Dual Laplacian editing for meshes," *IEEE Trans. Vis. Comput. Graphics*, vol. 12, no. 3, pp. 386–395, 2006.

[22] Y. Wu, O. K.-C. Au, C.-L. Tai, and T. Lu, "HIRM: A handle-independent reduced model for incremental mesh editing," *Comput. Aided Geom. Design*, vol. 35–36, pp. 56–68, 2015.

[23] Y. Yu *et al.*, "Mesh editing with poisson-based gradient field manipulation," *ACM Trans. Graph.*, vol. 23, no. 3, pp. 644–651, 2004.

[24] J. Huang *et al.*,"Subspace gradient domain mesh deformation," *ACM Trans. Graph.*, vol. 25, no. 3, pp. 1126–1134, 2006.

[25] O. K.-C. Au, H. Fu, C.-L. Tai, and D. Cohen-Or, "Handle-aware isolines for scalable shape editing," *ACM Trans. Graph.*, vol. 26, no. 3, pp. 83, 2007.

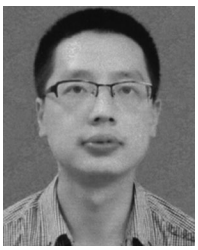
[26] A. Barmpoutis, E. Bozia, and D. Fortuna, "Interactive 3D digitization, retrieval, and analysis of ancient sculptures, using infrared depth sensors for mobile devices," in *Proc. 9th Int. Conf. Human-Comput. Interact.*, 2015, vol. 9178, pp. 3–11.

[27] W. Yoshizaki *et al.*, "An actuated physical puppet as an input device for controlling a digital manikin," in *Proc. SIGCHI Conf. Human Factors Comput. Syst.*, 2011, pp. 637–646.

[28] T. Shiratori, M. Mahler, W. Trezevant, and J. K. Hodgins, "Expressing animated performances through puppeteering," in *Proc. IEEE 3D User Interfaces*, 2013, pp. 59–66.

[29] A. Jacobson *et al.*, "Tangible and modular input device for character articulation," *ACM Trans. Graph.*, vol. 33, no. 4, pp. 82:1–82:12, 2014.

[30] J. Han, L. Shao, D. Xu, and J. Shotton, "Enhanced computer vision with microsoft kinect sensor: A review," *IEEE Trans. Cybern.*, vol. 43, no. 5, pp. 1318–1334, Oct. 2013.
[31] R. Held, A. Gupta, B. Curless, and M. Agrawala, "3D puppetry: A kinect-based interface for 3d animation," in *Proc. 25th Annu. ACM Symp. User Interface Softw. Technol.*, 2012, pp. 423–434.
[32] P. Phamduy, M. DeBellis, and M. Porfiri, "Controlling a robotic fish via a natural user interface for informal science education," *IEEE Trans. Multimedia*, vol. 17, no. 12, pp. 2328–2337, Dec. 2015.
[33] Y. Akagi, M. Furukawa, S. Fukumoto, Y. Kawai, and H. Kawasaki, "Interactive 3D animation system based on touch interface and efficient creation tools," in *Proc. IEEE Int. Conf. Multimedia and Expo*, 2013, pp. 1–7.
[34] Y. Seol, C. O'Sullivan, and J. Lee, "Creature features: Online motion puppetry for non-human characters," in *Proc. ACM Symp. Comput. Animation*, 2013, pp. 213–221.
[35] J. C. Rubio, J. Serrat, A. M. López, and N. Paragios, "Unsupervised co-segmentation through region matching," in *Proc. IEEE Comput. Vis. Pattern Recognit.*, 2012, pp. 749–756.
[36] N. Snavely, S. M. Seitz, and R. Szeliski, "Photo tourism: Exploring photo collections in 3D," *ACM Trans. Graph.*, vol. 25, no. 3, pp. 835–846, 2006.
[37] A. Kowdle, S. N. Sinha, and R. Szeliski, "Multiple view object cosegmentation using appearance and stereo cues," in *Proc. 12th Eur. Conf. Comput. Vis.*, 2012, pp. 789–803.
[38] A. Nealen, T. Igarashi, O. Sorkine, and M. Alexa, "Laplacian mesh optimization," in *Proc. 4th Int. Conf. Comput. Graph. Interactive Techn. Australasia Southeast Asia*, 2006, pp. 381–389.
[39] M. Reuter, F.-E. Wolter, M. E. Shenton, and M. Niethammer, "Laplace-Beltrami eigenvalues and topological features of eigenfunctions for statistical shape analysis," *Comput.-Aided Design*, vol. 41, no. 10, pp. 739–755, 2009.
[40] Z. Karni and C. Gotsman, "Spectral compression of mesh geometry," in *Proc. 27th Annu. Conf. Comput. Graph. Interactive Techn.*, 2000, pp. 279–286.
[41] O. Sorkine and M. Alexa, "As-rigid-as-possible surface modeling," in *Proc. ACM Symp. Geom. Process.*, 2007, vol. 257, pp. 109–116.
[42] U. von Luxburg, "A tutorial on spectral clustering," *Statist. Comput.*, vol. 17, no. 4, pp. 395–416, 2007.
[43] C. H. Séquin, "Rapid prototyping: A 3D visualization tool takes on sculpture and mathematical forms," *ACM Commun.*, vol. 48, no. 6, pp. 66–73, 2005.
[44] M. Vaezi, H. Seitz, and S. Yang, "A review on 3D micro-additive manufacturing technologies," *Int. J. Adv. Manuf. Technol.*, vol. 67, no. 5–8, pp. 1721–1754, 2013.
[45] B. Bickel *et al.*, "Design and fabrication of materials with desired deformation behavior," *ACM Trans. Graph.*, vol. 29, no. 4, pp. 63:1–63:10, 2010.



Yirui Wu is currently working toward the Ph.D. degree in the Department of Computer Science and Technology, Nanjing University, Nanjing, China.

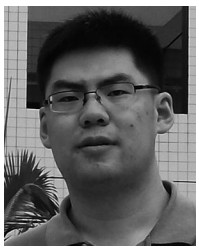
His current research interests include the areas of multimedia, computer vision, and computer graphics.

Tong Lu (M'15) received the B.Sc. and M.Sc. degrees and Ph.D. degree in computer science from Nanjing University, Nanjing, China, in 1993, 2002, and 2005, respectively.

He was a Visiting Scholar with the National University of Singapore and in the Department of Computer Science and Engineering, Hong Kong University of Science and Technology. In 2005 and 2007, he became an Assistant Professor and Associate Professor, respectively, with the Department of Computer Science and Technology, Nanjing University, where he is currently a Full Professor. He is also a Member of the National Key Laboratory of Novel Software Technology, China. He has published more than 60 papers and authored two books in his area of interest, and issued more than 20 international or Chinese invention patents. His current interests include the areas of multimedia, computer vision and pattern recognition algorithms/systems.

Zehuan Yuan is currently working toward the Ph.D. degree in the Department of Computer Science and Technology, Nanjing University, Nanjing, China.

His current research interests include the areas of pattern recognition and computer vision.



Hao Wang received the M.S. degree in computer science from Nanjing University, Nanjing, China, in 2014.

He is currently with the Institute of Deep Learning, Baidu, China.