FREESCUP: A NOVEL PLATFORM FOR ASSISTING SCULPTURE POSE DESIGN

Yirui Wu, Tong Lu, Zehuan Yuan, Hao Wang*

National Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, 210046, China

ABSTRACT

Sculpture design is challenging due to its inherent difficulty in characterizing an artwork quantitatively, and few works have been done to assist sculpture design. We present a novel platform to help sculptors in two stages, comprising *automatic sculpture reconstruction* and *free spectral-based sculpture pose editing*. During sculpture reconstruction, we co-segment a sculpture from real scene images of different views through a two-label MRF framework, aiming at performing sculpture reconstruction efficiently. During sculpture pose editing, we automatically extract candidate editing points on the sculpture by searching in the spectrums of Laplacian operator. After manually mapping body joints of a sculptor to particular editing points, we further construct a global Laplacian-based linear system by adopting the spectrums of Laplacian operator and using Kinect captured body motions for real time pose editing. The constructed system thus allows the sculptor to freely edit different kinds of sculpture artworks through Kinect. Experimental results demonstrate that our platform successfully assists sculptors in real-time pose editing.

Index Terms— Sculpture design, FreeScup, pose editing

1. INTRODUCTION

Displaying sculpture artworks which thousands of people can see is the dream of most sculptors. It is true that they have benefited from the recent technologies such as 3D geometric modeling[1] and artwork retrieval [2] in producing various sculpture artworks. For example, in sculpture modeling, precisely defined and highly optimized shapes following a clear underlying logic [3] are used in abstract sculpture design, which results in several sculpture families with small physical maquette or large-scale sculpture design [4]. Human-centered interaction techniques have also been adopted, for examples, in the "*See me, Feel me, Touch me, Hear me*" project [5], a trajectory crossing sculptures is crafted by combining textual and audio instructions to derive directed viewing, movement and touching in a sculpture garden. In [6], Kinect is also introduced to help produce 3D animations from physical objects. However, sculpture design especially in a virtual 3D environment is still a very tedious process for most sculptors [4].

Providing efficient and natural tools to inspire the ideas of sculptors is the target of this research. Generally, real-life sculpture design consists of the following steps: 1) conception, 2) initial artwork design, 3) drafting using a small-scale clay sculpture for observations and discussions, 4) molding the full-size sculpture, 5) coloring, and 6) finalizing the art design. Among these steps, Step 1 to Step 3 are particularly time consuming since even a tiny pose modification may require the repetition of the design from the beginning. Although art is always believed difficult to characterize quantitatively, we find that sculpture design can be much easier to start with a suitable initial shape and to make incremental changes with a naturally pose editing tool on the shape. Take the classic artworks in Fig. 1 as an example. The three sculptures seem similar in their styles; however, they have very different artistic emotions: unseen pressures are hidden in (a) through the shrinking body gesture of the thinker, while the pose in (b) is relax and stretchable, implying humans have been liberated from an uncultured society. As a comparison, (c) shows the vitality of humans through a nearly symmetrical shape "S". It inspires us that pose editing actually plays an important role in sculpture design. That is, if we provide a real-time and free pose editing platform for sculptors using proper multimedia devices, sculptors can be inspired by liberating them from tedious pose modeling either on a physical sculpture or using a traditional geometry editing program.

Fig. 1. Sculpture artworks: (a) *The Thinker* and (b) *The Age of Bronze* by Rodin, (c) *Discobolos* by Myron.

Based on these considerations, we propose a novel sculpture platform named *FreeScup* for assisting sculpture pose design. Fig. 2 gives the overview of the platform. The input is a set of sculpture images captured from different views in a real scene. It is because most sculptors learn to form his/her

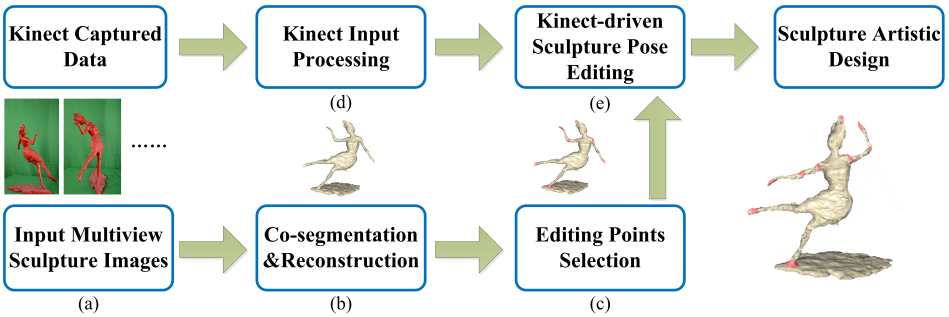

Fig. 2. The proposed framework for assisting sculpture design: (a) inputting a set of multiview sculpture images captured from a real scene, (b) a 3D mesh is reconstructed, (c) searching for candidate editing points on the sculpture, (d) Kinect input processing, and (e) Kinect-driven sculpture pose editing.

own artistic style from imitating and modifying existing artworks. A 3D mesh is then reconstructed from the input by co-segmenting the images through a two-label MRF framework. Next, our platform automatically searches for candidate editing points on the reconstructed sculpture. After mapping his/her joints to particular editing points, the proposed Kinect-driven sculpture pose editing scheme allows sculptors to edit the artwork in a free way through a Kinect device.

The main contribution of the paper is to propose a new sculpture design assisting platform that supports free sculpture pose modifications. Sculptors are allowed to freely edit their artworks through multimedia devices and the experimental results demonstrate the effectiveness of the proposed platform. By this way, human burden can be greatly reduced either in tedious clay molding or in geometry editing, which helps sculptors focus more on finding fantastic artwork ideas. To the best of our knowledge, this is the first work towards assisting sculpture artistic design by using multimedia techniques.

2. INITIALIZATION OF SCULPTURE DESIGN

In this section, we initialize sculpture design by two steps: *sculpture image co-segmentation* and *3D reconstruction*.

2.1. Co-segmentation from Different Views

The backgrounds in different view images tend to reduce the quality in reconstructing the foreground sculpture. To solve this problem, we perform image co-segmentation technique on all the views simultaneously, namely, segmenting all the images that contain the same sculpture by making use of the similar foreground visual characteristics.

If we assign label 1 to the sculpture and 0 to backgrounds, assuming that L is the binary pixel label of all the images, the co-segmentation of the sculpture can be modeled by the following two-label MRF framework:

$$E(L) = \sum_i E_a(L_i) + \sum_i E_d(L_i) + \sum_{i,j} E_p(L_i, L_j) \quad (1)$$

where E_a encourages sculpture pixels to follow its global appearance, E_p is a potts model to assign penalties to different label combinations of two neighboring pixels in the same image, and E_d introduces an objectness measurement [7] by the reconstructed 3D shape and calibrated cameras, formulating the intuition that the foreground pixel has a consistent occurrence in all the input sculpture images. To solve it, an alternated s-t minimized cut between updating foreground appearance model and performing labeling is adopted. We use Photo tourism [8] to calibrate all the images, where camera intrinsic and external parameters are initialized by establishing a two-view geometry model through matching sparse features across images.

2.2. Sculpture Shape Reconstruction

We adopt an incremental 3D reconstruction method [9] to generate the sculpture mesh $\mathcal{M}$ on different granularity over these calibrated segmented images in real-time. The method initializes a sculpture model by a set of uniformly distributed views and incrementally updates the mesh using new view images by minimizing

$$E(\mathcal{M}) = \kappa E_p(\mathcal{M}) + E_s(\mathcal{M}) \quad (2)$$

where E_p measures the photometric consistency of $\mathcal{M}$ within all the images, E_s enforces the smoothness of the reconstructed sculpture, and κ balances the importance of the two functions. $E(\mathcal{M})$ will be minimized by conjugate gradient descent to generate the desired 3D sculpture mesh.

3. REAL-TIME SCULPTURE EDITING

In this section, we aim at providing sculptors an efficient tool for liberating them from tedious 3D pose editing. We first automatically search for candidate editing points on the 3D shape, and then edit sculpture poses in real-time freely by mapping Kinect captured sculptor body motions onto particular editing points using a spectral-based method.

3.1. Construction of Laplacian operator

Spectrum is known as 'Shape DNA' since it describes inherent geometry features on a 3D shape [10], we thus introduce a novel spectral method for sculpture pose editing, the motivation of which stems from exploiting abundant geometry information in the spectral space for real time pose editing.

We define the following cotangent weighted Laplacian operator for $\mathcal{M}$:

$$\mathbf{L}^c_{ij} = \begin{cases} -1 & i = j \\ w_{ij} & (i,j) \in E \\ 0 & otherwise \end{cases} \tag{3}$$

where α_{ij} and β_{ij} are opposite orientations to edge (i, j), and

$$w_{ij} \propto \cot \alpha_{ij} + \cot \beta_{ij} \tag{4}$$

The cotangent Laplacian $\mathbf{L}^c$ is an approximation of both surface normals and curvatures, and its eigenvectors $[\phi^L_1, \phi^L_2, \ldots]$ with an increasing order of eigenvalues $[\lambda^L_1, \lambda^L_2, \ldots]$ can serve as an informative tool for shape analysis. In the following subsections, we will introduce our novel Kinect driven scheme by using the constructed Laplacian operator.

3.2. Searching for Editing Points

Basically, there are two kinds of editing points on the reconstructed shape for pose editing, namely, the points on protrusion or prominent regions, and the points on joints or junction parts. We search for such points by utilizing the first N eigenvectors $\{\phi^L_i\}^N_{i=1}$ in the spectral space as follows:

1. **Searching for extreme points of eigenvector fields**. Here $\{\phi^L_i\}^N_{i=1}$ can be treated as the fields defined on the sculpture mesh manifold. Due to the intrinsic harmonic behavior, the point that has a locally maximum/minimum field value exists on a protrusion or prominent region on the sculpture.

2. **Searching for maximum points on the gradient field of eigenvectors**. The gradient of eigenvectors $\{\mathbf{g}^L_i\}^N_{i=1}$ can also be treated as the fields defined on the sculpture manifold. The gradient field that has a large field value is considered sensitive in indicating a local variation on the sculpture surface. Such a local variation will be considered as a joint on the sculpture.

3. **Searching for saddle points on eigenvector fields**. A saddle point on eigenvector fields has the property that its neighborhood is not entirely on any side of the tangent space. For example, for a human-like sculpture, saddle points most probably appear on the junction regions of human limbs.

After selecting the editing points, a sculptor can manually add/delete/modify the editing points. We thus get a set of candidate editing points $\{h_j\}^M_{j=0}$, where M presents the final number of such points. We show the searched candidate editing points on a Neptune sculpture in Fig. 3(a).

3.3. Kinect Input Processing

Kinect has been proved useful in producing low-cost multimedia systems as it enables free human-machine interactions without any kind of additional devices. In our platform, we hope to capture body motions of the sculptor. The problem here is that there may exist noises and occlusions, which often makes the capture of body motions inaccurate. We thus apply two filters on the captured motion data:

1. **Holt-Winters double exponential smoothing filter**. It helps predict new motions under a reasonable assumption that there often exists a trend in the captured data. By interpolating between the prediction and the captured motions, our platform is able to compute stable joint positions and orientations to remove most jitters and noises caused by Kinect.

2. **Limbs filter**. This filter is applied to prevent the jumpy of limbs. Kinect can provide rough predictions for the clipped motions; however, the inference can occasionally be erroneous since it is based on a limited depth image. We thus linearly interpolate the previous smoothed joint positions and the inferred positions to predict new positions for clipped limbs.

After filtering, the sculptor manually maps Kinect captured body joints to particular editing points of the sculpture, the style of which can be either human-like or not. These points are named as mapped editing points. This process is represented by a conversion matrix $Map_{M\times 25}$, where the entry from jth row and kth column of Map is set to 1 if the sculptor defines the mapping between the editing point h_j and the kth body joint, otherwise it is set to 0. The number of 25 here refers to the predefined maximal number of the body joints that a Kinect device can recognize.

Formally, we compute the filtered body motions as the difference of joint positions $d^t_k = g(f(p^t_k) - f(p^{t-1}_k))$ and the transformation of joint orientations $r^t_k = orth(f(o_k)^t * f(o^{t-1}_k)^{-1})$, where p^t_k and o^t_k respectively denote the position and orientation of the kth joint at time t, function f denotes the filters, $orth$ represents the orthogonal normalization that achieves a rigid transformation, and g represents the motion scale transform based on the size portion of Kinect input image and sculpture size. We thus rewrite the motions of the mapped editing points in matrix form as follows:

$$H^t = Map_{M\times 25} S^t_{25\times 3} \tag{5}$$

where S^t is constructed from the filtered body motion set $\{d^t_k\}^{25}_{k=1}$ and $\{r^t_k\}^{25}_{k=1}$, and each row of H^t represents the motion of one editing point, including the translations in x, y, z coordinates and $4 * 4$ rotation matrix entries.

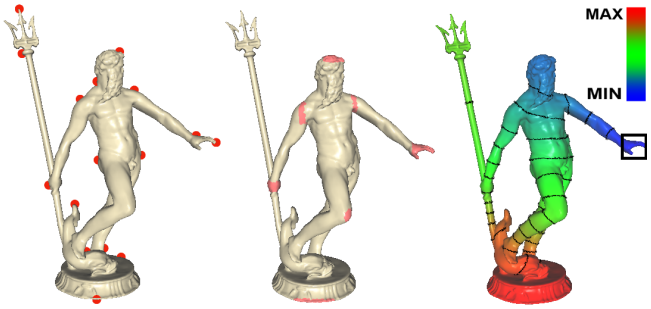

Fig. 3. Kinect-driven pose editing: (a) automatically searched editing points, (b) extended regions from the mapped editing points, (c) the example propagation from the left hand to the whole sculpture in one spectrum, where Eigenvector values from the maximum to the minimum are shown by different colors from red to blue.

3.4. Kinect-driven Sculpture Pose Editing

We convert Kinect-driven sculpture pose editing problem to the minimization of the variances of local shape features (i.e., curvature, normal and local rigidity) before and after editing. We construct the following linear system to preserve these features on the 3D surface during editing:

$$AX = b(X), \qquad with$$
$$A = \begin{bmatrix} L^D \\ \Phi \end{bmatrix} \qquad and \qquad b(X) = \begin{bmatrix} \sigma(X) \\ H \end{bmatrix} \tag{6}$$

where L^D is the Dual-Laplacian operator [11] constructed from the original sculpture before editing, $\sigma(X)$ represents the Laplacian coordinates that depend on the deformed vertex positions X, and $\Phi X = H$ indicates that the minimization problem is subject to the constraints of the mapped editing points controlled by the sculptor using Kinect.

After defining the linear system, we need find appropriate transformations for each vertex on the sculpture to make its coordinates fit with the changes of surface orientations during pose editing. We first expand the mapped editing points to regions, which offer six degrees of freedom to reflect the rotations and translations of the body motions of the sculptor. Then, we propose a Kinect-driven propagation scheme to describe time-varying transformations. Real-time interactions will be achieved by decreasing the size of the transformations during editing. We illustrate the scheme in Fig. 3, where (b) shows the editing regions extended from the mapped editing points, and (c) gives example propagations from the left hand to the whole sculpture in one specified spectrum.

Specifically, we first uniformly sample the spectrum ϕ_i^L to calculate isolines $\{iso_{i,n}\}_{n=1}^K$, each of which is represented by a black circle as shown in Fig. 3(c). Note that the vertices on an isoline always receives the same value of transformations. The transformations associated with any calculated isoline are determined by the body motions captured by Kinect in our scheme, while the rests are interpolated to support real time pose editing. We thus compute the transformation values of isolines induced by Kinect captured body motions H_k of the kth joint by

$$s_{i,n}^t = \frac{w * H_k^t}{|\phi_{i,n} - \phi_{i,h_j}|} \tag{7}$$

where $s_{i,n}$ represents the transformation value assigned to the nth isoline in the ith spectrum, $\phi_{i,n}$ and ϕ_{i,h_j} denote the eigenvector values of the nth isoline and the jth editing points, respectively, and w is a preset parameter. For the spectrum in Fig. 3(c), the isoline that has a smaller eigenvector value (near the left hand) gets a larger transformation value.

The transformation value s of vertex v that is not on any isoline will be interpolated as follows:

$$s_v^t = \sum_{i=1}^{i=N} \theta_i * \delta_{i,v} * s_{i,n}^t + \theta_i * (1 - \delta_{i,v}) * s_{i,n+1}^t. \tag{8}$$

where v is supposed between the nth and the (n+1)th isolines. $\delta_{i,v}$ denotes a linear weight determined by eigenvector value difference between v and its neighboring nth isoline, and θ_i represents the weight of the ith spectrum as quadratic sum of $\beta_{X,i}$ after Eigenspace projections, i.e., $\beta_{X,i} = X \cdot \phi_i$ in x, y, z coordinates. Kinect captured body motions are firstly propagated to the isolines by Equ.7, then are propagated to each vertex by Equ.8.

Essentially, we apply isolines to decrease the size of transformations, which are directly related with the Kinect captured body motions in the platform. After that, we only need calculate a much smaller number of the transformations associated with the isoline set during pose editing, which ensures the editing with low computing cost and fast convergence. The number of eigenvectors N and the number of isolines K are empirically decided, and in our platform we set $N = 8$ and $K = 15$ through many tests. We thus rewrite the Kinect-driven propagation scheme as $X^t = W \cdot E \cdot H^t$ in matrix form, where E is a column vector constructed from the eigenvector values set $\{\phi_{i,n}\}$ and $\{\phi_{i,h_j}\}$, and W is a weight matrix constructed from $\{\theta_{i,v}\}$ and $\{\delta_i\}$.

To achieve real time feedback of editing, we adopt an iterative Laplacian updating strategy [12], which provides intermediate pose editing results guide artwork design. We iteratively and alternatively update S and X based on the current deformed surface, and use the Kinect-driven scheme to update Laplacian coordinates $\sigma(X^t)$ based on the rotations of the triangles that are crossed by isolines. Note that only S requires to be updated during pose editing. The size of S is related with the number of the triangles crossed by isolines. Compared with the original transformation size, namely, the whole mesh vertex $n_{ver} \times n_{ver}$, S is much smaller after decreasing,

Group	Bear							ChinaRed							Armadillo				
	t_r	t_e	n_r	n_e	n_c	n_s	P	t_r	t_e	n_r	n_e	n_c	n_s	P	t_e	n_e	n_c	n_s	P
A_1	83.6	10.7	823	136	-	-	-	*fail*	23.3	*fail*	471	-	-	-	19.6	395	-	-	-
A_2	6.0	2.1	12	18	16	5	80	20.2	4.6	16	26	25	8	75	3.7	15	17	6	75
B_1	121	18.2	1434	241	-	-	-	*fail*	36.7	*fail*	889	-	-	-	27.6	527	-	-	-
B_2	6.2	3.2	14	29	16	6	83.3	25.4	5.2	18	32	25	13	54.6	4.2	17	17	8	60

Table 1. User studies of FreeScup. Superscripts 1 and 2 denote using ZBrush and FreeScup, respectively. P is defined as $P = n_s^c/n_s$, where n_s^c represents the number of the mapped editing points. Duration is measured by minutes, while P is measured by $\%$.

Model	n_{ver}	n_{tri}	n_c	n_s	$t_{p,1}(s)$	$t_{p,2}(s)$	$\tilde{t_{p,2}}(s)$	$t_{i,1}(ms)$	$t_{i,2}(ms)$	$n_{i,1}$	$n_{i,2}$
Bear	10233	963	16	6	1.04	1.03	**0.271**	63	**9.03**	64	**17**
ChinaRed	73539	1562	23	8	20.9	18.8	**4.03**	806	**38.9**	213	**36**
Armadillo	60000	1250	17	6	17.2	15.1	**3.31**	928	**37.4**	93	**28**
Feline	15744	1293	16	6	2.88	2.86	**0.613**	118	**13.8**	78	**22**
Raptor	25102	1339	20	6	6.31	6.08	**1.21**	173	**20.8**	173	**34**
Human	15154	1327	16	9	1.60	1.42	**0.421**	98	**11.3**	88	**25**
Neptune	28052	1312	16	8	6.57	6.38	**1.23**	190	**21.7**	98	**28**

Table 2. Subscripts p and i respectively refer to pre-computing and iterative updating, while subscripts 1 and 2 respectively refer to Dual-Laplacian editing [11] and our method.

which significantly speeds the per-iteration updating and convergence rate, resulting in real-time sculpture pose editions.

4. EXPERIMENTAL RESULTS

We have cooperated with 8 students majoring in sculpture for more than one year. We conduct experiments on real sculpture artworks by comparing the proposed way with a well known 3D sculpture modeling platform named ZBrush.

4.1. User Study for Assisting Sculpture Design

In this experiment, we conduct a series of user studies to evaluate the effectiveness of our platform by paying special attention to the learning cost of 3D pose editing. We divide the sculptors into two groups, namely, 4 sculptors who are familiar with ZBrush are arranged into Group A, while the rests who are not familiar with any 3D modeling software are arranged into Group B. Both the two groups are asked to create virtual 3D sculptures respectively with our platform and ZBrush by imitating two existing clay artworks named as bear and ChinaRed. In addition, they are asked to edit the pose of an existing virtual 3D sculpture named Armadillo. We record all the data of the 8 sculptors, such as the duration t_r and the interactions (e.g., mouse clicks and keyboard typing) n_r for creating the sculptures, the duration t_e and the interactions n_e for pose modifications, and the total number of candidate and mapped editing points, i.e., n_c and n_s.

The results are shown in Table 1, from which we can see that the creation of a 3D sculpture using ZBrush averagely requires much more interactions. Another interesting phenomena is that the sculptors in both Group A and B fail in creating the ChinaRed sculpture, which is more complex than the other sculpture artworks. For comparison, our platform achieves lower durations $\{t_{r,A_2}, t_{r,B_2}\}$ and fewer interactions $\{n_{r,A_2}, n_{r,B_2}\}$ by comparing them with $\{t_{r,A_1}, t_{r,B_1}\}$ and $\{n_{r,A_1}, n_{r,B_1}\}$, illustrating the effectiveness of the reconstruction of the two clay sculptures from multiview photos. Essentially, the most time-consuming step of 3D reconstruction lies on the post refinement of the reconstructed 3D shape, such as filling holes and removing noises. We also notice that the manually created Bear sculpture by ZBrush only contains with nearly 2,000 vertices averagely, while our reconstruction offers a sculpture with nearly 10,000 vertices. Moreover, the high precision P, which defined as $P = n_s^c/n_s$, is up to near 60%, proving that the automatically searched editing points are acceptable for the sculptors.

In Table 1, we also notice that 3D pose editing by ZBrush always requires more time $\{t_{e,A_2}, t_{e,B_2}\}$ and more interactions $\{n_{e,A_2}, n_{e,B_2}\}$ comparing with the proposed platform. That means only after time-consuming and repeatedly editing on plenty of vertices, the sculptors can finally observe pose editing effects. This is not good for inspiring fantastic ideas during artwork design. It can also be found that ZBrush itself is a complex 3D modeling environment for learning. This can be proved by the more modeling duration t_{all,B_1} and the more interactions n_{all,B_1} in comparison with t_{all,A_1} and n_{all,A_1}. Note that we represent $t_{all} = t_r + t_e$ and $n_{all} = n_r + n_e$. For our platform, the learning phase is quite short since the real-time visual feedback helps the sculptors correctly understand

their interactions. We thus find a much smaller learning cost of our platform by comparing t_{all,B_2} and n_{all,B_2} with t_{all,A_2} and n_{all,A_2}.

4.2. Computational Cost for Sculpture Pose Editing

In this experiment, we compare our spectral based pose editing scheme with another editing algorithm named Dual-Laplacian [11]. 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 test both the two methods on the 8 sculptures. Table. 2 gives the detailed results of the two methods on a 1.7GHz i5 core2 PC with 6GB of RAM machine. In our platform, W and L are pre-computed before editing, which will not change during pose editing. We adopt a strategy by storing these matrices for reusing to save the pre-computation cost t_p. We thus decrease $t_{p,2}$ to a much shorter time $\tilde{t_{p,2}}$. By decreasing the transformation size in each iteration, we achieve a much lower iterative updating cost $t_{i,2}$ and convergency number $n_{i,2}$ than $t_{i,1}$ and $n_{i,1}$ as shown in Table. 2, especially in producing complex sculptures with a larger number of vertices.

Note that Kinect captures 30 frames per second, which requires a pose editing algorithm reacts within about $1/30$ seconds, else the sculptors may feel latency after his/her interactions. In Table. 2, we find the iterative updating duration $t_{i,1}$ of Dual-Laplacian are not small enough to support real-time editing feedbacks, while our method is proved effective in such interactions.

5. CONCLUSIONS

We present a novel platform (*FreeScup*) for assisting sculpture design. After reconstructing a sculpture from multi-view images, we propose a spectral approach to help sculptors freely edit different kinds of sculpture artworks through Kinect through a novel iterative Laplacian updating strategy. User studies and experimental results show the effectiveness in both improving the efficiency and inspiring new ideas in sculpture design of the proposed platform. Our further work includes the imitations of sculpturing actions consisting of smoothing, pinching, cutting, beating and carving using multimedia techniques.

Acknowledgments

The work described in this paper was supported by the Natural Science Foundation of China under Grant No. 61272218 and No. 61321491, and the Program for New Century Excellent Talents under NCET-11-0232. We thank Guozhu Liang and Longfei Qin for their working on experiments.

6. REFERENCES

[1] S. H. Carlo, "Computer-aided design and realization of geometrical sculptures," *Computer-Aided Design and Applications*, vol. 4, no. 5, pp. 671–681, 2007.

[2] P. F. Sylvie, J. Michel, N. Laurent, and C. Jean, "Artwork 3d model database indexing and classification," *Pattern Recognition*, vol. 44, no. 3, pp. 588–597, 2011.

[3] H. S. Carlo, "Sculpture design," *In: Proc. of International Conference on Virtual Systems and Multimedia*, 2001.

[4] H. S. Carlo, "Art, math, and computers: news ways of creating pleasing shapes," *Educator's TECH Exchanges*, 1996.

[5] 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. of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 149–158, 2013.

[6] Robert Held, Ankit Gupta, Brian Curless, and Maneesh Agrawala, "3d puppetry: a kinect-based interface for 3d animation," in *The 25th Annual ACM Symposium on User Interface Software and Technology, UIST '12, Cambridge, MA, USA, October 7-10, 2012*, 2012, pp. 423–434.

[7] Adarsh Kowdle, Sudipta N. Sinha, and Richard Szeliski, "Multiple view object cosegmentation using appearance and stereo cues," in *ECCV*, 2012, pp. 789–803.

[8] Noah Snavely, Steven M. Seitz, and Richard Szeliski, "Photo tourism: exploring photo collections in 3d," *ACM Transactions on Graphics*, vol. 25, no. 3, pp. 835–846, 2006.

[9] Ze-Huan Yuan and Tong Lu, "Incremental 3d reconstruction using bayesian learning," *Appl. Intell.*, vol. 39, no. 4, pp. 761–771, 2013.

[10] Martin Reuter, Franz-Erich Wolter, and Niklas Peinecke, "Laplace-beltrami spectra as 'shape-dna' of surfaces and solids," *Computer-Aided Design*, vol. 38, no. 4, pp. 342–366, 2006.

[11] Oscar Kin-Chung Au, Chiew-Lan Tai, Ligang Liu, and Hongbo Fu, "Dual laplacian editing for meshes," *IEEE Trans. Vis. Comput. Graph.*, vol. 12, no. 3, pp. 386–395, 2006.

[12] Jin Huang, Xiaohan Shi, Xinguo Liu, Kun Zhou, Li-Yi Wei, Shang-Hua Teng, Hujun Bao, Baining Guo, and Heung-Yeung Shum, "Subspace gradient domain mesh deformation," *ACM Trans. Graph.*, vol. 25, no. 3, pp. 1126–1134, 2006.