Proceedings of the ASME 2016 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference
IDETC2016
August 21-24, 2016, Charlotte, North Carolina, USA

IDETC2016-60499

USING KINECT TO CAPTURE HUMAN MOTION FOR MECHANISM SYNTHESIS, MOTION GENERATION AND VISUALIZATION

Anurag Purwar*, Rumit Desai

anurag.purwar@stonybrook.edu
Computer-Aided Design and Innovation Lab
Department of Mechanical Engineering
Stony Brook University
Stony Brook, New York, 11794-2300

ABSTRACT

In this paper, we are presenting a framework for capturing human motions using Microsoft Kinect sensor for the purpose of 1) generating task positions for mechanism and robot synthesis, and 2) generation and visualization of B-spline interpolated and approximated motion from the captured task positions. The theoretical foundation of this work lies in Kinematic Mapping, Dual and Bi-quaternions, and NURBS (Non-Uniform Rational B-spline) geometry. Lately, Kinect has opened doors for creation of natural and intuitive human-machine interactive (HMI) systems in medicine, robotic manipulation, CAD, and many other fields, where visual-sensing and -capture is a central theme. Kinect has made a huge impact in physical therapy area, achieving new benchmarks in tele-rehabilitation by improving physical exercise assessment, monitoring and supervision using the skeletal data. Moreover, Kinect's depth sensing capability has helped in retrieving depth information required for robotic vision in grasping, object recognition which was previously done using computationally demanding computer vision algorithms. Kinect's point cloud data with interactive gestures has proven to be useful in various CAD software for conceptual design of shapes. Mechanism synthesis is one of the areas in Kinematics, where Kinect-provided skeletal data can be leveraged to design and develop highly customized end-user collaborated mechanism solutions. We demonstrate that using Kinect, OpenGL, and Openframeworks, we can capture discrete (or, key) rigid body displacements, continuous motions, and generate and visualize rational B-spline motions from captured key positions. Capturing only a few key positions results in significant data savings and also provides a natural way to create tasks for mechanism synthesis problems. The output is a set of dual quaternions and 4×4 homogeneous transforms representing a task motion, which can be used as an input for mechanism synthesis applications. The tool produced also allows users to generate trajectories of various points on a moving rigid body interactively. A Kinect-based capture of such motions can help create highly-customized assistive devices for people who suffer from a range of motion-related difficulties due to old age or disabilities.

1 Introduction

Apart from its Xbox gaming system, Microsoft Kinect has found niche applications in physical therapy and rehabilitation, robotics, CAD and computer vision domain. Lars et al. [1] made an argument to use markerless systems for capturing human motions without creating artificial stimuli and artifacts in the capture process. Kinect, a non-intrusive and economical sensor, has given new dimension to physical therapy rehabilitation process by providing developers a simple and time-efficient means to create interactive games to make patients exercise and provide musculoskeletal rehabilitation [2]. Researchers and clinicians have leveraged depth data along with skeletal data to cre-

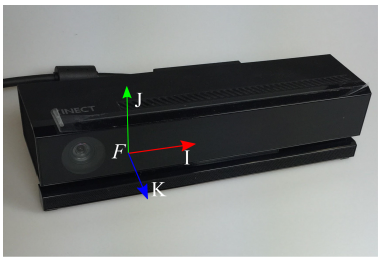

FIGURE 1: Kinect uses a right handed coordinate system as shown to report joint data.

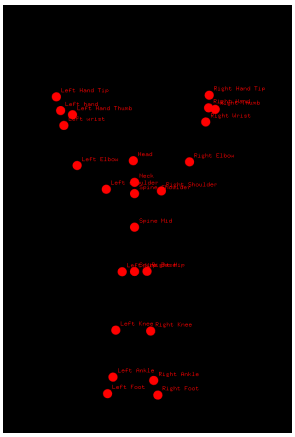

FIGURE 2: Kinect skeletal data shows 25 joints being tracked.

ate many interactive feedback solutions to help patients do exercises correctly [3, 4, 5]. Robotics assistance is another area in which Kinect has replaced conventional visual sensing capabilities using color cameras by providing additional data like infrared frames and depth frames, which otherwise are obtained by post processing the color images. Controlling a robotic arm using human gestures seems non-trivial at first, but Kinect's skeletal data along with its gesture capabilities have made this task possible and making robotic control more user-friendly [6]. It also provides point cloud data and has been used in conjunction with natural hand-gestures for CAD modeling, like sweeping a cylinder [7]. Despite the noise that accompanies the Kinect sensor data, it has been used with filtering algorithms to create 3D reconstruction of scenes, objects and other components in virtual reality [8]. Microsoft's Kinect for Windows page [9] has numerous examples of innovative applications in Physical Computing area, which is concerned with creating interactive physical systems using sensor data from the real-world.

Accurate and precise motion analysis has always been an integral part of bio-mechanical applications like gait analysis and correction. Kinematic study of human motion involving 2D and 3D motion capture was an involved problem prior to the launch of Kinect sensor. However, Kinect has facilitated three dimensional kinematic analysis of simple human motions by measuring parameters like joint-positions [10].

In this paper, we are presenting a framework which can be used for generating key position and continuous motion data for planar- and spatial-mechanism synthesis and rigid-body motion generation using hand movements. Most mechanism synthesis algorithms rely on input of a few discrete (or, key) positions. Thus, capturing key positions provides 1) a natural way to feed motion task to the synthesis algorithms, and 2) to reduce the storage overhead by not storing position data in every frame.

Our proposed tool has been developed using Microsoft Kinect for Windows v2 in conjunction with Openframeworks [11] and OpenGL 3D graphics API [12]. Kinect is a depth- and RGB-sensing hardware consisting of an RGB camera, infrared camera and a microphone array. It provides RGB images of size 1920×1080 and IR cameras are used to capture real-time depth images of size 512×424 both of them seeding data with a frame rate of 30 Hz. Field of view for IR cameras is 70 degrees in horizontal direction and 60 degrees in vertical direction. To take advantage of this hardware, a free SDK is provided by Microsoft along with various sample programs to give a feel of different types of data provided by Kinect. In our work we have mainly use the joint data provided by its skeleton frame which uses depth data in order to measure the distance in Z-direction. Kinect device uses the principle of structure light in order to get the $Z-$dimension.

Kinect's Software Development Kit (SDK) provides three-dimensional joint coordinates (X, Y, and Z) in meters with respect to Kinect's fixed reference frame as shown in Fig. 1. This data is provided for 25 different joints in human body as shown in Fig. 2. Kinect's depth sensing capability, gesture recognition built in its SDK, and wide range field of views in both horizontal and vertical direction have been instrumental in developing 3D visualization and capture of motions in this work.

We store key positions and continuous motions in real-time by hand gestures and palm movement – assuming a moving coordinate frame attached to left palm and measuring its relative orientation with respect to the Kinect's fixed reference frame, we generate direction-cosine rotation matrix. By combining the rotation matrix with the translation vector from the fixed frame to the moving frame, we generate a 4×4 homogeneous transform and dual quaternion representation of the motion. The user alternates between moving and pausing for a few frames to capture key positions. The beginning of a continuous motion recording is signaled by raising of the right hand and stopped by lowering it. In between, the user moves left palm freely in the space and Kinect continues to record all the frames; the tool generates corresponding dual quaternions and stores them. We show that we can use just a few key positions to generate C^2-continuous rational B-spline motions, which could substitute for continuous recording. An outcome of this work is to provide an interactive tool to input positions and to utilize human motions as interactive medium for mechanism design. The tool and the instructions for its use will be posted at http://me.eng.sunysb.edu/software.

Rest of this paper is organized as follows: In section 2, we review the concept of dual quaternions and discuss how we generate 4×4 homogeneous transformation matrices and dual quaternions from the joint-data. This allows real-time capture of both key positions as well as continuous motions. In sections 3, we discuss methods for interpolating and approximating key positions with cubic B-spline motions, which reduces data storage requirements. Finally, we discuss the details of graphic visual-

ization and tool's user interface in the last section.

2 Pose generation from skeletal data

Mechanism synthesis for the motion generation problem requires pose (position and orientation) data with the constraint that the resulting mechanism should pass through the given poses or come as close as possible. In this section, we show how Kinect-provided skeletal data in the form of joint positions in three-dimensional coordinates is converted into pose data represented as dual quaternions. We first review the concept of quaternions and dual quaternions to the extent necessary for the development of this paper.

2.1 Dual Quaternions for Spatial Displacement

A spatial displacement (Fig. 3) can be represented as a rigid body transformation from a moving frame M to a fixed frame F in terms of a 4×4 homogeneous matrix:

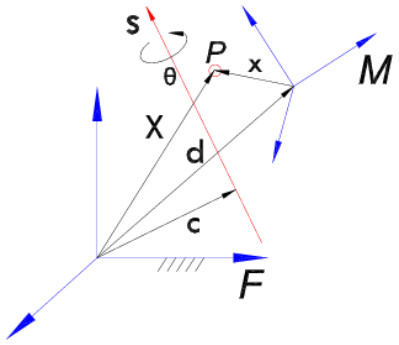

FIGURE 3: A spatial displacement given by rotation axis **s**, angle of rotation θ, and translation vector **d**. M and F indicate moving and fixed frames, respectively.

$$\begin{bmatrix} \mathbf{X} \\ 1 \end{bmatrix} = \left[\begin{array}{ccc|c} & [R] & & \mathbf{d} \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix}, \tag{1}$$

where $\mathbf{X}$ and $\mathbf{x}$ are vectors whose scalar components are the Cartesian coordinates of the point as measured in F and M, respectively.

Matrix representations are, however, verbose and problematic for motion interpolation because the rotation matrix in the interpolation and approximation process has to remain orthonormal; see Fillmore [13] and Roschel [14]. It has been recognized that an effective way of dealing with the problem is to use quaternions [15] and dual quaternions [16, 17].

The rotation matrix $[R]$ can be parameterized with quaternion coordinates $\mathbf{Q} = (q_1, q_2, q_3, q_4)$ related to the unit axis of rotation $\mathbf{s} = (s_x, s_y, s_z)$ and the rotation angle θ by

$$q_1 = s_x \sin(\theta/2),\ q_2 = s_y \sin(\theta/2),\ q_3 = s_z \sin(\theta/2),\ q_4 = \cos(\theta/2). \tag{2}$$

The above given quaternion coordinates satisfy the following relation:

$$\mathbf{Q} \cdot \mathbf{Q} = q_1^2 + q_2^2 + q_3^2 + q_4^2 = 1. \tag{3}$$

When the above condition is satisfied, $\mathbf{Q}$ is called a unit quaternion. Then rotation matrix $[R]$ in terms of quaternion coordinates is given by

$$[R] = \frac{1}{S^2} \begin{bmatrix} q_4^2+q_1^2-q_2^2-q_3^2 & 2(q_1q_2-q_4q_3) & 2(q_1q_3+q_4q_2) \\ 2(q_2q_1+q_4q_3) & q_4^2-q_1^2+q_2^2-q_3^2 & 2(q_2q_3-q_4q_1) \\ 2(q_3q_1-q_4q_2) & 2(q_3q_2+q_4q_1) & q_4^2-q_1^2-q_2^2+q_3^2 \end{bmatrix}, \tag{4}$$

where $S^2 = q_1^2 + q_2^2 + q_3^2 + q_4^2$.

A dual quaternion $\hat{\mathbf{Q}} = \mathbf{Q} + \varepsilon\mathbf{Q}^0$, where ε is the dual unit with the property $\varepsilon^2 = 0$, can represent a spatial displacement [16, 17]. The real part $\mathbf{Q} = q_1\mathbf{i} + q_2\mathbf{j} + q_3\mathbf{k} + q_4$ is defined by the homogeneous Euler parameters of rotation as given in Eq. (2). The dual part, $\mathbf{Q}^0$, which is associated with the translational component of the spatial displacement is given by

$$\begin{bmatrix} q_1^0 \\ q_2^0 \\ q_3^0 \\ q_4^0 \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 0 & -d_3 & d_2 & d_1 \\ d_3 & 0 & -d_1 & d_2 \\ -d_2 & d_1 & 0 & d_3 \\ -d_1 & -d_2 & -d_3 & 0 \end{bmatrix} \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix}. \tag{5}$$

The dual part can also be written as a quaternion product

$$\mathbf{Q}^0 = (1/2)\mathbf{d}\mathbf{Q}, \tag{6}$$

where $\mathbf{d}$ is a vector quaternion associated with the translation vector. Dual quaternions remain invariant when multiplied with a scalar, i.e., same rotation matrix and translation components are obtained. Therefore, Ravani and Roth [18, 19] considered them as a set of four homogeneous dual coordinates that define a point in a projective dual three-space called the *image space of spatial displacements*.

2.2 Position Capture

We attach the moving frame to the palm of user's left hand as shown in the Fig. 4 and use three joint-data information ($\mathbf{P}_{hand}$, $\mathbf{P}_{handTip}$, and $\mathbf{P}_{handThumb}$) provided by the Kinect SDK to construct this frame. The choice of left hand is arbitrary to input a task position; alternatively, the right hand could be chosen while ensuring the right-handedness of a coordinate frame attached to the hand. The plane of the palm provides the X-Y plane, while the Z-axis is obtained by taking a cross-product of unit vectors in the X-Y plane using right-hand rule as follows:

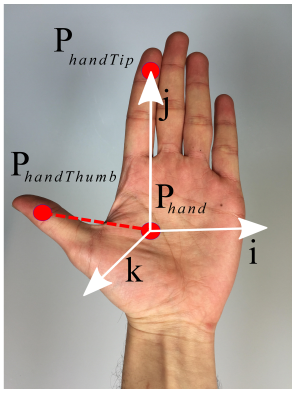

FIGURE 4: The moving frame is attached to the palm of user's hand; **i,j,k** are the unit basis vectors computed using hand joints.

$$
\begin{aligned}
\mathbf{j} &= \vec{u}/|\vec{u}|, \\
&\quad \text{where } \vec{u} = \mathbf{P}_{handTip} - \mathbf{P}_{hand} \\
\mathbf{k} &= \mathbf{j} \times \vec{v}/|\vec{v}|, \\
&\quad \text{where } \vec{v} = \mathbf{P}_{handThumb} - \mathbf{P}_{hand} \\
\mathbf{i} &= \mathbf{j} \times \mathbf{k}
\end{aligned}
$$

The 4×4 homogeneous matrix representing this moving frame M with respect to the Kinect-located fixed frame F can be written as

$$
\mathbf{T} = \begin{bmatrix} \mathbf{I}\cdot\mathbf{i} & \mathbf{I}\cdot\mathbf{j} & \mathbf{I}\cdot\mathbf{k} & d_1 \\ \mathbf{J}\cdot\mathbf{i} & \mathbf{J}\cdot\mathbf{j} & \mathbf{J}\cdot\mathbf{k} & d_2 \\ \mathbf{K}\cdot\mathbf{i} & \mathbf{K}\cdot\mathbf{j} & \mathbf{K}\cdot\mathbf{k} & d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{7}
$$

where $\mathbf{I}$, $\mathbf{J}$, and $\mathbf{K}$ are the unit basis vectors of F, while $\mathbf{d} = (d_1, d_2, d_3)$ represents the translation vector from F to M. In the above matrix, upper 3×3 matrix is the well-known direction cosine matrix. A dual quaternion representation for this matrix can be obtained by computing rotation part $\mathbf{Q} = (q_1, q_2, q_3, q_4)$ of the dual quaternion by solving Eq. (4) [20, 21]. The dual part $\mathbf{Q}^0 = (q_1^0, q_2^0, q_3^0, q_4^0)$ is obtained from Eq. (5). Thus, a dual quaternion $\hat{\mathbf{Q}}$ is obtained for every key position.

Planar linkage synthesis usually requires strictly planar motion data. In that case, we can ignore translations in $Z-$direction and set $d_3 = 0$ in the Eq. (1), while the upper 2×2 matrix would represent a rotation in the X-Y plane. The corresponding homogeneous matrix reduces to

$$
[H] = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & d_1 \\ \sin\theta & \cos\theta & 0 & d_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{8}
$$

where θ is the angle of rotation in the X-Y plane.

For a planar displacement, a dual quaternion reduces to a planar quaternion $\mathbf{Z} = Z_1 i\varepsilon + Z_2 j\varepsilon + Z_3 k + Z_4$ given by following four components [17]:

$$
\begin{aligned}
Z_1 &= \frac{d_1}{2}\cos\frac{\theta}{2} + \frac{d_2}{2}\sin\frac{\theta}{2}, \\
Z_2 &= -\frac{d_1}{2}\sin\frac{\theta}{2} + \frac{d_2}{2}\cos\frac{\theta}{2}, \\
Z_3 &= \sin\frac{\theta}{2}, \\
Z_4 &= \cos\frac{\theta}{2}.
\end{aligned} \tag{9}
$$

Therefore, for n key positions, we obtain n dual- or planar-quaternions. Their corresponding homogeneous transforms can be obtained by solving for rotation matrix $[R]$ from Eq. (4) and translation vector $\mathbf{d}$ by inverting Eq. (5) and then substituting in Eq. (1).

2.3 Key Position Recognition

For a position to be recognized as a key position, we use the following algorithm, where we measure the deviation of a particular position over time. If the position does not change as evidenced by a pause in the motion over a certain number of frames, we record that position as a key position. As shown in Algorithm 1, we are computing Δ representing the difference between two successive positions and a numerical measure of this difference is to be obtained either from its corresponding homogeneous matrix $\mathbf{T}$ or dual quaternion representation $\hat{\mathbf{Q}}$. If the difference remains below a threshold and a certain number of frames have elapsed, the position is recorded as a key position.

However, this is not a trivial problem – it is well-known that a metric for a planar- or spatial-displacement should be *ideally* bi-invariant (invariant with respect to change in moving or fixed reference frames) and meaningful so that the disparate units of translation and rotation can be combined [22]. McCarthy [23] showed that planar and spatial rigid-body motions can be regarded as special cases of spherical and 3-spherical motions, respectively and using this notion, an *approximately* bi-invariant metric can be obtained. Larochelle et al. [24] used Polar Decomposition (PD) and singular value decomposition (SVD) of a homogeneous transformation matrix associated with an element

Algorithm 1 Position Capture algorithm

```
procedure RECORDPOSITION
    for i = 1 do
        if body → tracked then
            if Δ < Threshold then
                if fameBuffer = frameCount then
                    KeyPosition ← Position
                else
                    frameBuffer++
            else
                frameBuffer=0
    return KeyPosition
```

of SE(n) to find the nearest approximating element in SO($n+1$) and then used a metric in the space of rotations, which leads to an approximately bi-invariant metric. Purwar and Ge [25] proposed an analytical method, which converts a dual quaternion into a biquaternion that represents a four-dimensional rotation. We briefly review this method for the computation of a deviation in hand position. Introducing a pair of unit quaternions, $\mathbf{G}$ and $\mathbf{H}$, called biquaternions such that

$$\mathbf{G} = \mathbf{DQ}, \quad \mathbf{H} = \mathbf{D}^*\mathbf{Q}, \tag{10}$$

where $\mathbf{D}$ is a unit quaternion given by

$$\mathbf{D} = \frac{2R}{\sqrt{4R^2+d^2}} + \bar{\mathbf{d}}\frac{d}{\sqrt{4R^2+d^2}}, \tag{11}$$

and its conjugate

$$\mathbf{D}^* = \frac{2R}{\sqrt{4R^2+d^2}} - \bar{\mathbf{d}}\frac{d}{\sqrt{4R^2+d^2}}. \tag{12}$$

In above equations, R is chosen to be a sufficiently large number to achieve better approximation [24], while $\bar{\mathbf{d}}$ is a unit vector along translation vector $\mathbf{d}$ and d is the magnitude of the translation vector. Then, given two poses defined by two sets of biquaternions, say $(\mathbf{G}_1, \mathbf{H}_1)$ and $(\mathbf{G}_2, \mathbf{H}_2)$, the distance Δ between them is given by

$$\Delta = \sqrt{(\mathbf{G}_1 - \mathbf{G}_2).(\mathbf{G}_1 - \mathbf{G}_2) + (\mathbf{H}_1 - \mathbf{H}_2).(\mathbf{H}_1 - \mathbf{H}_2)} \tag{13}$$

In the algorithm 1, we have used a Threshold value of 0.2 and the frameCount was set to 200 frames, after which we declare a position to be a key position. On the other hand, for continuous motion capture, we record all the frames and store the resulting displacements as a set of dual quaternions.

3 Motion Generation and Visualization

Once the key positions have been captured, we can employ motion design algorithms to produce smooth motions that either approximate given positions or interpolate them. We apply Computer Aided Geometric Design (CAGD) algorithms in the image space of spatial displacements to obtain a representation for rational motions in the Cartesian space [26, 27, 28]. For motion visualization, we use OpenGL 3D graphics API, which requires a homogeneous transform. The flow chart in Fig. 5 illustrates the process from capture to visualization.

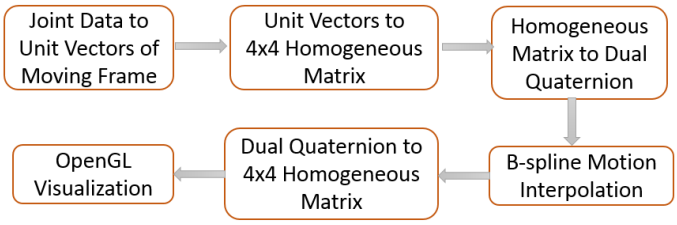

FIGURE 5: Data Flow from Kinect sensor to the final output on OpenGL screen

3.1 Rational B-spline Motion

Given $n+1$ key positions and their corresponding dual quaternions $\mathbf{Q}_i$., a rational B-spline motion can be obtained by designing a B-spline curve in the image space of spatial displacements. A B-spline curve represented by dual quaternions is given as [28]

$$\hat{\mathbf{Q}}(t) = \sum_{i=0}^{n} N_{i,p}(t)\hat{\mathbf{Q}}_i \tag{14}$$

where $N_{i,p}(t)$ are p-th degree B-spline basis functions [29]. Substituting for $\mathbf{Q}(t)$ in Eq. (1), we obtain a Cartesian motion of degree $2p$ due to quadratic functions of quaternion parameters. Such a motion approximates the given key positions, which can be useful for generating a smooth motion from the given key positions. Figure 6 shows six recorded key positions along with trajectory resulting from an approximating B-spline motion, while the approximating degree-six rational B-spline motion is shown in Fig.7. Table 1 gives the dual quaternions for the key positions.

The input key positions can be construed as interpolating positions as well. A B-spline interpolating motion can be generated using the Global Motion Interpolation Algorithm for curve design [29]. In this algorithm, the known quantities are the interpolating positions $\mathbf{Q_i}$ and we want to interpolate these given

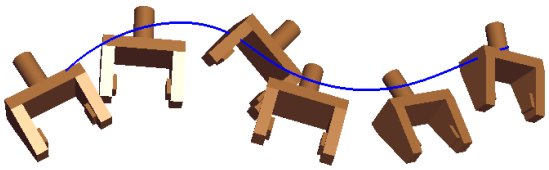

FIGURE 6: Recorded 6 key positions to approximate a cubic B-spline, trajectory passing through center of each input position

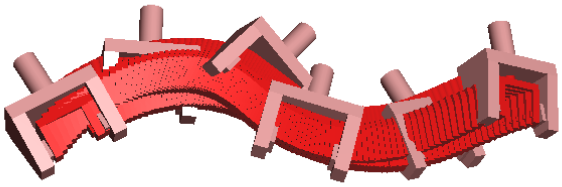

FIGURE 7: A cubic B-spline approximating motion with 6 positions

TABLE 1: Dual Quaternion key positions data in the form $(\mathbf{Q};\mathbf{Q}^0)$ for spatial cubic B-spline motion; rotation part of the Quaternion is dimensionless, while in the dual part, the translation vector coordinates are measured in meters.

$\hat{\mathbf{Q}}_0$	(-0.2216, 0.1610, 0.2679, 0.9236); (-0.6333, 0.0625, 1.0043, -0.4541)
$\hat{\mathbf{Q}}_1$	(-0.2388, 0.1204, -0.0151, 0.9634); (-0.5363, -0.0185, 1.0275, -0.1145)
$\hat{\mathbf{Q}}_2$	(-0.2688, 0.2099, -0.3813, 0.8591); (-0.5295, -0.1459, 0.9375, 0.2861)
$\hat{\mathbf{Q}}_3$	(-0.3482, 0.0896, -0.1204, 0.9252); (-0.2439, -0.2519, 1.0781, 0.0729)
$\hat{\mathbf{Q}}_4$	(-0.3161, 0.3156, 0.1661, 0.8791); (-0.1911, -0.2026, 0.9585, -0.1771)
$\hat{\mathbf{Q}}_5$	(-0.2571, 0.3291, -0.0142, 0.9084); (-0.0797, -0.0363, 0.9960, 0.0061)

positions with a degree p B-spline curve in the image space.

$$\mathbf{Q}_i = \sum_{i=0}^{n} N_{i,p}(\bar{u}_k)\mathbf{P}_i \tag{15}$$

Here the control positions $\mathbf{P}_i$ are the unknowns. With above equation, we need to solve a $(n+1)\times(n+1)$ system of linear equation to obtain $\mathbf{P}_i$. For the same key positions as given in Table 1, a cubic B-spline interpolating motion is shown in Fig. 8.

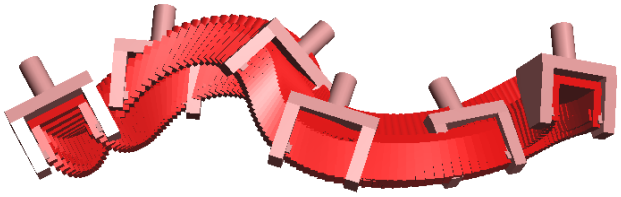

FIGURE 8: A cubic B-spline *interpolation* with 6 positions

4 Graphic Visualization and Tool Interface

Backbone of our application is OpenFrameworks 0.8.4 [11], which is an open source C++ toolkit developed for creative application development. We have used its GUI toolkit for implementing simple interaction like changing the degree of motion and selecting between motion approximation and interpolation. Moreover, options are made available to see specific motions like screw motion and B-spline motion. We have used OpenGL for motion visualization – its strong API support for graphical rendering and matrix manipulation makes it possible to represent interpolating motion in a more natural way.

The user interface of the tool is shown in Fig. 9. The top left window shows the OpenGL visualizations, i.e. recording motion and respective interpolated motion. Bottom left corner shows the spatial position data in the form of coordinate frames using Openframeworks, while on the right, we see the Kinect camera and skeletal tracking.

Gestures are used to initialize and stop recording of continuous motion recording. Movement of hands from initial position to final position results into a gesture that will initiate the continuous motion capture – when we move our hand in vertical direction from initial to final position, recording for continuous motion will start, and same gesture when repeated stops the recording as well.

An additional feature is to provide export of dual quaternion data in XML format, which can be imported in a mechanism synthesis application for designing mechanisms.

5 Conclusions and Future Work

In this paper, we have reported a method for capturing human poses and motion using Microsoft Kinect for ready con-

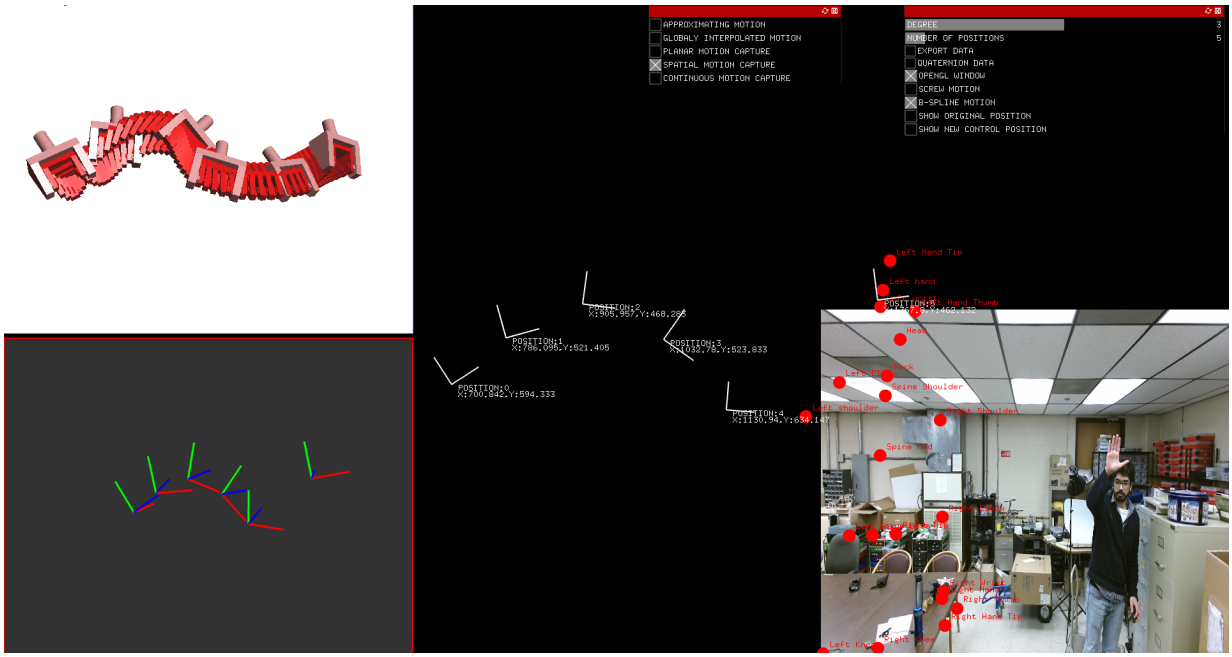

FIGURE 9: User Interface

sumption by mechanism synthesis applications. We have also developed a prototype tool that provide motion generation and visualization capabilities. Tool proposed in this paper is an initial effort to use the power of human interaction device like Kinect to improve the quality of motion visualization, which is an integral part in any mechanism synthesis. Moreover, its user interface allows designer to extract important data required for motion design. In our future research, we plan to compare the position results provided by Kinect with other standard methods of tracking joint positions. Moreover, we also expect to improve the GUI along with additional features which can be a complete package for human collaborated mechanism synthesis applications and extend its usability in medical domain for designing mechanisms for physically challenged patients.

REFERENCES

[1] Mndermann, L., Corazza, S., and Andriacchi, T. P., 2006, "The evolution of methods for the capture of human movement leading to markerless motion capture for biomechanical applications", Journal of NeuroEngineering and Rehabilitation.

[2] Lange, B., Koenig, S., McConnell, E., Chang, C., Juang, R., Suma, E., Bolas, M., and Rizzo, A., "Interactive game-based rehabilitation using the Microsoft Kinect", Virtual Reality Short Papers and Posters (VRW), 2012 IEEE, pp. 171–172.

[3] Microsoft, "Kinect app for physical therapy put to the test", URL https://blogs.msdn.microsoft.com/kinectforwindows/2015/06/05/kinect-app-for-physical-therapy-put-to-the-test/.

[4] Smisek, J., Jancosek, M., and Pajdla, T., 2011, "3D with Kinect", .

[5] Pinto, A. M., Costa, P., Rocha, A. P. M. L. F., Veiga, G., and Moreira, E., 2015, "Evaluation of Depth Sensors for Robotic Applications", .

[6] Liying, C., Qi, S., Han, S., Yang, C., and Shuying, Z., "Design and implementation of human-robot interactive demonstration system based on Kinect", Control and Decision Conference (CCDC), 2012 24th Chinese, pp. 971–975.

[7] Inc., A., 2016, "Reality Capture Inside AutoDesk using Kinect Fusion", .

[8] Alexiadis, D. S., Zarpalas, D., and Daras, P., 2013, "Real-Time, Full 3-D Reconstruction of Moving Foreground Objects From Multiple Consumer Depth Cameras", Multimedia, IEEE Transactions on, **15(2)**, pp. 339–358.

[9] Microsoft, "Kinect for Windows Product Blog", URL https://blogs.msdn.microsoft.com/kinectforwindows/.

[10] Wham, R. M., 2012, "Three-Dimensional Kinematic Analysis Using the Xbox Kinect", Honors thesis projects.

[11] Lieberman, Z., Watson, T., and Castro, A., "OpenFrameworks", URL http://openframeworks.cc/about/.

[12] Group, K., 2016, "OpenGL", .

[13] Fillmore, J. P., 1984, "A note on rotation matrices", IEEE Computer Graphics & Application, **4(2)**, pp. 30–33.

[14] Röschel, O., 1998, "Rational motion design - a survey",

Computer-Aided Design, **30(3)**, pp. 169–178.
[15] Shoemake, K., "Animating rotation with quaternion curves", SIGGRAPH85, ACM Computer Graphics, volume 19, pp. 245–254.
[16] Bottema, O. and Roth, B., 1979, Theoretical Kinematics, Dover Publication Inc., New York.
[17] McCarthy, J. M., 1990, Introduction to Theoretical Kinematics, The MIT Press, Cambridge, MA.
[18] Ravani, B., 1982, "Kinematic mapping as applied to motion approximation and mechanism synthesis", Ph.d. dissertation.
[19] Ravani, B. and Roth, B., 1983, "Motion Synthesis Using Kinematic Mappings", Journal of Mechanisms Transmissions and Automation in Design-Transactions of the Asme, **105(3)**, pp. 460–467.
[20] Dam, E. B., Koch, M., and Lillholm, M., 1998, "Quaternions, interpolation and animation", Technical Report DIKU-TR-98/5, University of Copenhagen.
[21] Klumpp, A. R., 1976, "Singularity-free extraction of a quaternion from a direction-cosine matrix", Journal of Spacecraft and Rockets, **13(12)**, pp. 754–755.
[22] Park, F. C., 1995, "Distance Metrics on the Rigid-Body Motions with Applications to Mechanism Design", ASME Journal of Mechanical Design, **117(1)**, pp. 48–54.
[23] McCarthy, J. M., 1983, "Planar and Spatial Rigid Body Motion as Special Cases of Spherical and 3-Spherical Motion", ASME Journal of Mechanism, Transmission, Automation, **105**, pp. 569–575.
[24] Larochelle, P., Murray, A., and Angeles, J., 2007, "A distance metric for finite sets of rigid-body displacements via the polar decomposition", ASME Journal of Mechanical Design, **129(8)**, pp. 883–886.
[25] Purwar, A. and Ge, Q. J., 2013, "Polar Decomposition of Unit Dual Quaternions", ASME Journal of Mechanisms and Robotics, **5(3)**, pp. 031001–031001, 10.1115/1.4024236.
[26] Ge, Q. and Ravani, B., 1991, "Computer aided geometric design of motion interpolant", Proceedings of 17th ASME Design Automation Conference, pp. 33–41.
[27] Juttler, B. and Wagner, M., 1996, "Computer Aided Design With Spatial Rational B-Spline Motions", ASME Journal of Mechanical Design, **119(2)**, pp. 193–201.
[28] Purwar, A. and Ge, Q. J., 2005, "On the Effect of Dual Weights in Computer Aided Design of Rational Motions", ASME Journal of Mechanical Design, **127(5)**, pp. 967–972.
[29] Piegl, L. and Tiller, W., 1995, The NURBS Book, Springer, Berlin.