Real-time Shape Recognition of a Deformable Link by Using Self-Organizing Map

Shan Xu[1,2], Gaofeng Li[1,2], Dezhen Song[3], Lei Sun[1,2], Jingtai Liu[1,2]

***Abstract*— Here we present a novel deformable manipulator composed of active rigid joints and deformable links. The deformable link consists of several passive spherical joints articulated with each other with preload force between socket-ball surfaces. Therefore the manipulator can reach more parts of the task space compared with rigid-link manipulators by bending deformable links according to different tasks. However, frequent changes in the links' shape lead to unknown kinematic parameters, which bring difficulties to the planning and control of the manipulator. In this paper, a real-time shape recognition algorithm is proposed for the deformable link by using Self-Organizing Map (SOM). To avoid topological error and local convergence problem, Least Square Method (LSQ) is utilized for initialization according to the link's current position and shape. The reinitialization process is added to improve the robustness when facing noise and occlusion. To meet the demand of real-time tracking, the GPU parallel computation is applied for acceleration. Moreover, an error metric based on Signed Distance Function (SDF) is presented for evaluation. In this paper, our algorithm is implemented on a deformable link with 11 components. Experimental results validate the feasibility and effectiveness of this method. The processing time descends from 700 ms/frame to 40 ms/frame by using GPU, and the overall average of tracking errors is below 4 mm.**

I. Introduction

The dexterity and safety of robot manipulator have become increasingly important issues for multiple applications especially in home service environments, which is unstructured with human around. Continuum manipulators or soft manipulators offer a number of potential advantages over rigid-link manipulators in safe interaction, as well as adapting to uncertain and complex environments. However, they are often too costly for home service robots due to special materials or actuators.

Here we present a novel low-cost deformable manipulator, which is composed of active rigid joints and deformable links (See Fig. 1). The deformable link consists of passive spherical joints articulated with each other with preload force between socket-ball surfaces. Therefore, the link can be bent under external forces when facing different tasks or moderating unexpected impacts. Compared with rigid-link manipulators, the deformable manipulator can offer extended workspace and be more dexterous, lower-cost and intrinsically safe. On the other hand, it can bear heavier payload than continuum manipulators by choosing a proper preload force.

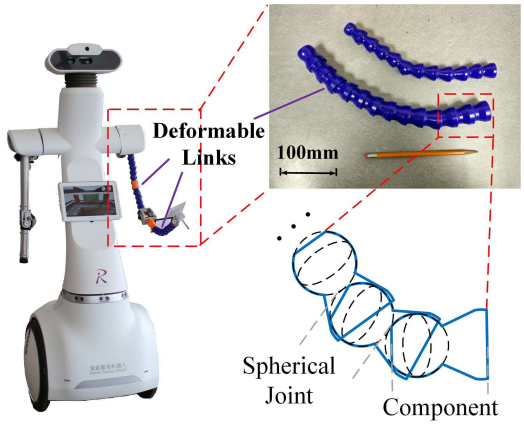

Fig. 1. The deformable manipulator of home service robot is composed of active rigid joints and deformable links. The deformable link consists of passive spherical joints articulated together with preload force between socket-ball surfaces.

However, the shape of the deformable link changes drastically and irregularly during bending operations, making kinematic parameters unknown for modeling. In our prior works, the kinematic-free control framework of a deformable manipulator is well studied [1], [2], including both position and orientation parts. Although the effectiveness of this method is validated, the model-based control is still necessary especially for high accuracy demands such as button-pressing or trajectory tracking tasks [3], [4]. Therefore, the shape measurement of the deformable link is a critical prerequisite for further applications.

In this paper, a real-time shape recognition algorithm is proposed to track the deformable link based on 3D point cloud. First, a concise model representation of the link is built up taking advantage of the rotational symmetry of its component. Then, parameters of the model is optimized by using Self-Organizing Map (SOM). In order to address the local convergence problem of SOM, Least Square Method (LSQ) is utilized for initialization according to the link's current position and shape. It can also guarantee the correct topological order of SOM mapping without reordering. The reinitialization process is added to improve algorithm robustness when facing noise and occlusion. The parallel computation is applied with GPU architecture for real-time tracking. Finally, an error metric based on Signed Distance Function (SDF) is presented to evaluate the tracking accuracy.

*This work is supported by National Natural Science Foundation of China (NSFC) under Grant No. 61573198, 61375087. Natural Science Foundation of Tianjin under Grant No. 15JCZDJC31200.

[1]Institute of Robotics and Automatic Information System, Nankai University, Tianjin, China, 300350. [2]Tianjin Key Laboratory of Intelligent Robotics, Nankai University, Tianjin, China, 300350. [3]Department of Computer Science and Engineering, Texas A&M University, College Station, Texas, United States, 77843.

Emails: {xu_shan, gaofengli}@mail.nankai.edu.cn.

*Corresponding authors: liujt@nankai.edu.cn.

II. Related Work

Many related works covering shape recognition of continuum robots especially in surgical intervention field are presented in recent years. Similar to them, the real-time shape sensing of a deformable link is also challenging due to the contradiction between high accuracy and deformation along the whole length. Generally, in surgical applications, the robot diameter is small and the working environment is strict. System sensors are often set as X-Ray apparatus [5], [6], [7], stereo cameras [8], [9] or strain sensors like fiber Bragg gratings sensors (FBG) [10], [11]. In our scenario, manipulators for home service are larger-sized with no attached surroundings. Therefore, the depth camera is chosen in this work as well as meeting demands of low-cost and simplicity in operation. As the shape recognition from 3D point cloud is a nonlinear optimization problem [12], [13], some optimal algorithms such as Gauss-Newton method [14] are presented together with parameterization. However, the convergence is quite limited due to the discontinuous functions. Some others use basic geometric elements to illustrate the shape of continuum robots, such as Bezier patches [14], manifold surfaces [15] or B-Spline surfaces [16]. In those methods, the irregularity of the object could bring in varying densities when sampling. Since the correct topological order is of vital importance in representing a manipulator link, the Self-Organizing Map shows its unique advantage as it can preserve the topology while mapping.

Self-Organizing Map is an unsupervised neural network based on competitive learning proposed by Kohonen [17]. It has the property of creating organized "internal representations" from multi-dimensional inputs and can skeletonize from sparse shapes regardless of size and connectivity [18]. Although this method is relatively robust and straightforward, the complicated training strategy hampers it from online applications [17], [19]. Hence, an accelerated SOM is presented in [9], [20] by decreasing training loops. They use all input points at the same time in every training loop instead of one point a loop as the traditional way. The mapping results are updated with sharp changes, which makes the algorithm converges at an extremely high speed.

As aforementioned, SOM can preserve topology and shape of inputs while mapping, but the point cloud is inputted in an unorganized fashion. In some prior works, the initial order in SOM is given by part of its training process, which is too time-consuming for the real-time tracking [17], [19]. Another reordering strategy is proposed to eliminate topological errors by using adjacency matrix [9], [21]. In this method, reference vectors as mapping results can be iteratively inserted where the performance measurement is not satisfied. In our case, reference vectors are chosen as positions of link components. They are designed to be the same ones without inserting and deleting because of a constant components number. Therefore, the Least Square Method is utilized for correct ordered initialization. By doing this, initial values of reference vectors are given along the corresponding range of input point cloud to avoid local convergence. Different from fixed objects, the deformable link moves around during manipulation, leading to a varying position in a large scale of 3D Euclidean Space. Hence, LSQ is a better choice than using a straight line [9], [18] or sampling randomly from input points [20]. Apart from initialization process, the reinitialization step is needed to improve algorithm robustness, since the noise and occlusion are inevitable in practical visual sensing.

III. Model Representation of the Deformable Link

The deformable link we track is a set of rigid components articulated together as a chain. To describe the shape of the whole body, local frames are attached to all rigid components to represent their poses. Camera frame C is chosen here as the reference frame. Then, every local frame can be identified relative to frame C by a rigid body transformation matrix ${}^{C}_{i}\boldsymbol{T} \in \mathbb{SE}(3)\ , i = 1, \cdots, M$, where M is the number of link components.

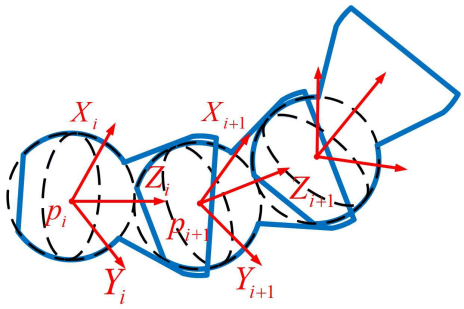

Fig. 2. Every component has a local frame attached to it with the original point set at the center of the spherical part. The Z-axis is pointing to the sphere center of the next component, and the directions of X and Y axes are not settled specifically in this model representation.

Traditionally, 6 parameters are needed to define a spatial pose, with 3 of which for position and the other 3 for orientation. When it comes to the component rotational symmetric about an axis, which is quite universal in articulated objects such as manipulators, the constraint of orientation can be released to some extent. Since the rotation around this axis makes no difference, there is no need to particularly design directions for the other two axes. On the contrary, if directions are settled, difficulties will be brought to the calculation of orientation parameters by redundant constraints. Taking this favorable structure into account, a concise model using only position parameters is proposed to represent local frames.

Fig. 2 is the illustration of local frames. The original point of each frame is set at the center of the spherical part. The Z-axis is pointing to the sphere center of the next component. As known that original point together with Z-axis is the only thing acquired to define a local frame, the parameter is chosen as a $(3M + 3)$ dimensional vector, which contains only original points of local frames:

$$\boldsymbol{\theta} = [p_{01}, p_{02}, p_{03}, p_{11}, p_{12}, p_{13}, \cdots, p_{M1}, p_{M2}, p_{M3}]^{\mathrm{T}}. \quad (1)$$

$\boldsymbol{p}_M = [p_{M1}, p_{M2}, p_{M3}]^{\mathrm{T}}$ is an arbitrary point on Z-axis of frame $(M-1)$ except its original point. It is used to complete the information of the last frame $(M-1)$.

Here is the local frame expression relative to the reference frame C:

$$ {}^{C}_{i}\boldsymbol{T} = \begin{bmatrix} {}^{C}_{i}\boldsymbol{R} & \boldsymbol{p}_i \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{X}_i & \boldsymbol{Y}_i & \boldsymbol{Z}_i & \boldsymbol{p}_i \\ 0 & 0 & 0 & 1 \end{bmatrix}. \quad (2) $$

In this matrix, $\boldsymbol{p}_i$ is the original point of frame *i*, and the rotation matrix ${}^{C}_{i}\boldsymbol{R}$ can be derived from original points of frame *i* and $(i-1)$.

First, Z-axis is the unit vector taking direction from point $\boldsymbol{p}_i$ to $\boldsymbol{p}_{i+1}$:

$$ \boldsymbol{Z}_i = \frac{\boldsymbol{p}_{i+1} - \boldsymbol{p}_i}{\|\boldsymbol{p}_{i+1} - \boldsymbol{p}_i\|} = \begin{bmatrix} Z_{i1} & Z_{i2} & Z_{i3} \end{bmatrix}^{\mathrm{T}}. \quad (3) $$

Then, we utilize (3) with

$$ \begin{aligned} d &= \|\boldsymbol{p}_{i+1} - \boldsymbol{p}_i\| \\ &= \sqrt{(p_{i+1,1} - p_{i1})^2 + (p_{i+1,2} - p_{i2})^2 + (p_{i+1,3} - p_{i3})^2}, \end{aligned} \quad (4) $$

and yield to $Z_1 = \frac{p_{i+1,1} - p_{i1}}{d}$, $Z_2 = \frac{p_{i+1,2} - p_{i2}}{d}$, $Z_3 = \frac{p_{i+1,3} - p_{i3}}{d}$. Next, $\boldsymbol{X}_i = [X_{i1}, X_{i2}, X_{i3}]^{\mathrm{T}}$ can be obtained from following equations:

$$ \begin{cases} X_1 Z_1 + X_2 Z_2 + X_3 Z_3 = 0 \\ X_1^2 + X_2^2 + X_3^2 = 1 \end{cases}. \quad (5) $$

For the computational convenience, we set X_1=0. Then the solution goes to: $X_2 = \frac{Z_3}{\sqrt{Z_2^2 + Z_3^2}}$, $X_3 = \frac{-Z_2}{\sqrt{Z_2^2 + Z_3^2}}$. By defining

$$ h = \sqrt{(p_{i+1,2} - p_{i2})^2 + (p_{i+1,3} - p_{i3})^2}, \quad (6) $$

we can get $X_2 = \frac{p_{i+1,3} - p_{i3}}{h}$ and $X_3 = \frac{p_{i2} - p_{i+1,2}}{h}$. Finally, the Y-axis $\boldsymbol{Y}_i = [Y_1, Y_2, Y_3]^{\mathrm{T}}$ can be given through the cross product operation of X and Z axes:

$$ \begin{aligned} \mathbf{Y} = \mathbf{Z} \times \mathbf{X} &= \begin{bmatrix} -\sqrt{Z_2^2 + Z_3^2} \\ Z_1 Z_2 / \sqrt{Z_2^2 + Z_3^2} \\ Z_1 Z_3 / \sqrt{Z_2^2 + Z_3^2} \end{bmatrix} \\ &= \begin{bmatrix} -h/d \\ (p_{i+1,1} - p_{i1})(p_{i+1,2} - p_{i2})/dh \\ (p_{i+1,1} - p_{i1})(p_{i+1,3} - p_{i3})/dh \end{bmatrix}. \end{aligned} \quad (7) $$

Now it is easy to find that, instead of complicated matrix computation in traditional models, local frames in our model are derived from one transformation matrix with only 6 parameters:

$$ {}^{C}_{i}\boldsymbol{T} = T_i(p_{i1}, p_{i2}, p_{i3}, p_{i+1,1}, p_{i+1,2}, p_{i+1,3}). \quad (8) $$

Although the model presented here is built up based on the deformable link, it can be easily extended to a group of articulated objects especially the ones with rotational symmetric components.

IV. Shape Recognition by Using SOM

A. Initialization by LSQ Fitting

In this algorithm, the point cloud $S = \left\{P_i = [x_i, y_i, z_i]^{\mathrm{T}} \in \mathbb{R}^3, i = 0, 1, \cdots N\right\}$ is captured by one fixed depth camera, and outputs $L = \left\{Q_j = [x_j, y_j, z_j]^{\mathrm{T}}, j = 0, 1, \cdots M\right\}$ are updated reference vectors describing the configuration of the whole link.

As explained in modeling process, representation parameters are original points of all local frames, the number of which is constant in our case. Thus, it's much easier to maintain the number and order of parameters by using the same reference vectors without deleting and inserting. Initial values of reference vectors are supposed to be set along the corresponding range of the point cloud. That allows data points to be properly associated with reference vectors. Different from tracking a fixed object, the deformable link always moves around during manipulator operations. In order to address the link's variable position and shape, LSQ is utilized in this paper for initialization. The captured point cloud S is reused as input signal of LSQ and could be thinned for a faster calculation. In our algorithm, two polynomial functions are fitted out separately from 3D points with the function order set by users:

$$ \begin{cases} y = a_0 + a_1 x + a_2 x^2 + \cdots + a_m x^m \\ y = b_0 + b_1 z + b_2 z^2 + \cdots + b_m z^m \end{cases}. \quad (9) $$

All parameters can be optimized by minimizing the sum of deviation squares, such as the y-x part:

$$ J = \sum_{i=1}^{N} [y_i - (a_0 + a_1 x_i + \cdots + a_m x_i^m)]^2. \quad (10) $$

The proper range of two fitted curves corresponding to the link is figured out according to marginal inputs. Then initial reference vectors can be obtained by sampling from two curve segments with constant intervals. It's also easy to index those vectors sequentially from one end to another with the same topological order of the tracked link. That can provide a correct order for the following SOM mapping.

Although fitting functions as polynomials aren't accurate enough for sketching out an arbitrary link shape, LSQ is only served as initialization here. The accuracy of it is totally sufficient for the request of data association and topology preservation.

B. Iterative Training

In traditional SOM, much efforts are taken to adjust it's training parameters since both initialization part and convergence part are contained. Referring to the accelerated algorithm [9], [20], the main idea of optimization is pulling reference vectors directly to the average location of their neighborhood data points.

As detailed in [9], every data point P_i should be associated to its nearest reference vector according to a distance metric, such as the Euclidean distance used in our method:

$$ Q_c(P_i) = \arg\min_{Q_j \in L} \|P_i - Q_j\|. \quad (11) $$

Algorithm 1 Shape Recognition Algorithm

loop
  Get current camera frame
  Extract link's 3D point cloud $S_0 = \{P_i | i = 1, \cdots, N\}$
  if $initialization$ **then**
    Fit curve C_0 from S_0 by LSQ
    Get reference vectors L_0 by sampling C_0 with constant interval
  end if
  for $i = 0$; $i < 4$; $i++$ **do**
    for all $P_i \in S$ **do**
      Find P_i's associated reference vector $Q_c(P_i)$
      Accumulate position information of $Q_c(P_i)$
    end for
  end for
  for all $Q_j \in L$ **do**
    Get neighborhood $N_j = \{P_i | Q_j = Q_c(P_i)\}$
    if $|N_j| == 0$ **then**
      Reinitialize reference vector L by LSQ
    else
      Update reference vector $Q_j^* = \sum_{i \in N_j} P_i / |N_j|$
    end if
  end for
end loop

Following this, the neighborhood of a reference vector Q_j is defined as a 3D point set $N_j = \{P_i | Q_j = Q_c(P_i)\}$, which consists of all data points whose closest reference vector worked out to be Q_j. Then, the new position of reference vector Q_j is updated by a straightforward average operation:

$$Q_j^* = \frac{\sum_{i \in N_j} P_i}{|N_j|}. \tag{12}$$

In order to meet the accuracy requirement, the updating process also iterates multiple times to guarantee convergence. Unlike the conventional one, a few loops are adequate to pull reference vectors close enough to real positions, thus making it particularly efficient and also simple to be applied.

C. Robustness to Noise and Occlusion

Since every captured point contributes to the output during updating procedure, the point cloud is preprocessed to be extremely clear. Otherwise, reference vectors may be deviated far away from correct positions by noise points. The preprocess is to extract the object from the whole image using the information of color and depth. However, noise from background is difficult to be totally removed. To address this, the reinitialization part is added to improve robustness. The procedure of reinitialization is exactly the same as the initialization part, which is sampling from fitted curves of the current link to refresh invalid reference vectors.

Another situation that matters is when some components are blocked by obstacles or even other components of the link. The blocked reference vectors may be miss-associated and pulled away, which makes them extremely difficult to be rematched in the next loop. For these two situations, the trigger of reinitialization is set when a blank neighborhood is observed. Algorithm 1 is the pseudocode of the entire shape recognition algorithm.

V. GPU Implementation

As mentioned in Section IV, proper data association is imperative for good recognition results. In order to use previous outputs as initial values for the next data association, high computation efficiency is required for little changes between two requisite frames. Even though some acceleration has been implemented within SOM algorithm, big amount of data from dense point cloud still blocks the way of real-time application. From SOM pseudocode illustrated above, simple identical executions are applied to all points repeatedly, and little data correlation is needed in this framework. Therefore, the parallel computation using GPU architecture is applied for a further speeding up.

GPU can launch thousands of threads executing the same code at the same time. It exceeds its CPU counterpart in

Algorithm 2 GPU architecture for SOM

Null all the reference vectors' neighborhood
Memory allocation in GPU
Copy data from CPU to GPU
Kernel1 // associate data point P_i to their closest reference vector Q_{P_i}
for thread $i = 0$; $i < N$; $i++$ **do**
  Give d_0 a big initial value
  for all do$Q_j \in L$
    $d_{ij} = \|P_i - Q_j\|$
    if $d_{ij} < d_0$ **then**
      $d_0 = d_{ij}$
      $Q_{p_i} = Q_j$
    end if
  end for
end for
Kernel2 // accumulate position information of reference vector Q_{P_i}
for thread $i = 0$; $i < N$; $i++$ **do**
  atomicAdd$(X[Q_{P_i}], x_{P_i})$
  atomicAdd$(Y[Q_{P_i}], x_{P_i})$
  atomicAdd$(Z[Q_{P_i}], x_{P_i})$
  atomicAdd$(num[Q_{P_i}], 1)$
end for
copy results from GPU to CPU
if $N_j = \emptyset, \forall Q_j$ **then**
  Reinitialization
else
  for all $Q_j \in L$ **do**
    $x_{Q_i} = X[Q_i]/num[Q_i]$
    $y_{Q_i} = Y[Q_i]/num[Q_i]$
    $z_{Q_i} = Z[Q_i]/num[Q_i]$
  end for
end if

computation efficiency by a large margin, and is recently introduced into real-time tracking of complex 3D objects [14]. In our implementation, GPU operations are mainly conducted in two kernels. The first one associates all data points to the closest reference vector. The second one updates positions of reference vectors according to neighborhood data points. In both kernels, each thread is responsible for one data point independently. Those two kernels replace massive serial iterations in conventional CPU architecture, and the modified algorithm structure is outlined in Algorithm 2.

In order to minimize the runtime of data transferring between CPU and GPU, data points are first copied to GPU memories allowing efficient reading and writing of GPU kernels. After the computation of all data points, the GPU outcomes are transferred back to CPU memories for next executions. The updating of a shared data in GPU by multiple threads can lead to a data disorder without requiring synchronization. Hence in kernel 2, atomic operation is used to lock the signal asserted by instructions.

VI. Error Metric for Evaluation

In order to evaluate the tracking accuracy, an error metric based on SDF is utilized by importing geometrical information of object in an implicit way. SDF can determine the distance of a given point x to the boundary of a geometry: $SDF\left(x\right) : \mathbb{R}^3 \rightarrow \mathbb{R}$, which takes on negative value inside the geometry, positive value outside and has a value of zero at the surface. A precomputed SDF is used to remove explicit computation of the nearest point, replacing it with an extremely fast looking-up [22]. In our work, SDF is generated by the combination of some analytically simple shape functions, while it can also be obtained through 3D scanning or so.

Referring to [14], the local version of SDF is applied for articulated objects. Instead of describing the whole body in one global SDF, local SDFs of rigid components are chained together with parameters in modeling process. Therefore, the shape changes can be symbolized by variable parameters, with no massive recomputation of global SDF is needed. The data association part in this metric is the same one as that in SOM. It allows inputted 3D points transferred into associated local frames directly by reusing current model parameters:

$$\boldsymbol{Q}_k = \boldsymbol{Q}_c\left(\boldsymbol{P}_i\right) = \arg\min_{Q_j \in L} \left\|\boldsymbol{P}_i - \boldsymbol{Q}_j\right\|, \tag{13}$$

$$^{k}\boldsymbol{P}_i = {}^{k}_{C}\boldsymbol{T}\left(\boldsymbol{\theta}\right){}^{C}\boldsymbol{P}_i. \tag{14}$$

After getting distance errors from transferred points:

$$e_i = \mathrm{SDF}^{k}\left({}^{k}_{C}\boldsymbol{T}\left(\boldsymbol{\theta}\right){}^{C}\boldsymbol{P}_i\right), \tag{15}$$

the final accuracy error is defined as the mean value of the absolute sum.

$$e = \frac{\sum\limits_{N} \left\|\mathrm{SDF}\left(\boldsymbol{P}_i;\boldsymbol{\theta}\right)\right\|}{N}. \tag{16}$$

VII. Experiments

Experiments are implemented on a 3.4GHz Intel Core i7, 16GB RAM computer, with NVIDIA's GeForce GTX 1080 to execute the parallel device codes. As illustrated in Fig. 3, the deformable link consists of 11 identical components. The depth camera is Intel RealSense SR300 and can be changed to other types for different demands in depth, vision field or resolution. Our method is compared with the accelerated SOM presented in [9], [20]. Each experiment lasts for a period of time including a sequence of recognition trials. Once a trial is completed, the next will be triggered immediately using the current camera frame.

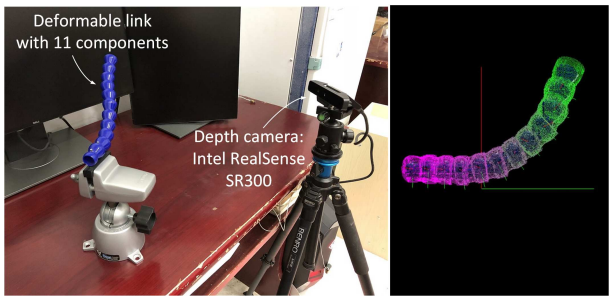

Fig. 3. Experiment devices include the deformable link with 11 components and the Intel RealSense SR300 as the depth camera.

Fig. 4 shows the recognition accuracy evaluated by SDF metric. Our method can provide high quality tracking since mean errors are generally smaller than 4 mm. Some sudden increases as outliers refer to the noise or occlusion. And the value can return back to normal in few loops due to the instant error elimination. In contrast, the error of accelerated SOM is much higher with apparent accumulation for moving link, and the result depends heavily on the initial value.

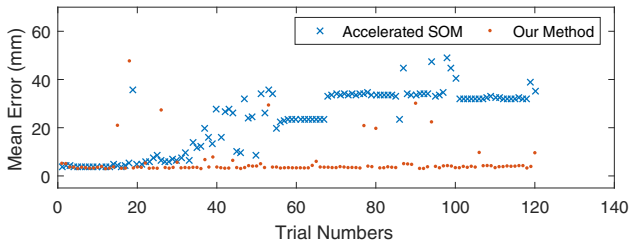

Fig. 4. Algorithm accuracy of our method and accelerated SOM method measured by SDF error metric.

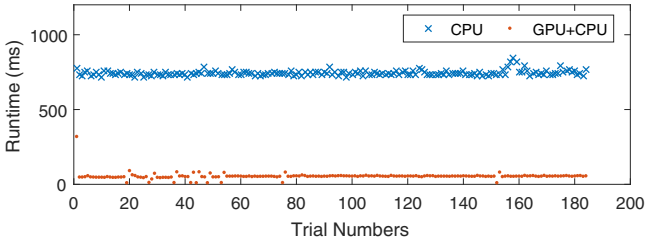

Fig. 5. Time-consuming of shape recognition with and without GPU acceleration.

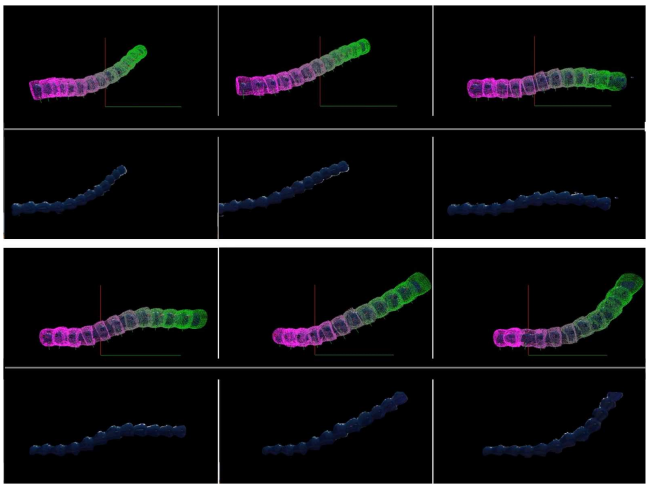

Fig. 6. Online visualization of recognized 3D shapes of deformable link together with corresponding camera images. 6 different shapes are shown here, with the upper one the recognition result and lower one the image obtained by depth camera. A video of shape recognition experiments by accelerated SOM method and our method can be found at https://youtu.be/NPrLjGaDgzA.

Fig. 5 shows the time consuming of shape recognition implemented by serial and parallel computation. It's obvious that parallel computation with GPU architecture can highly improve the execution speed consistently. In our platform, the processing time is about 40 ms, which is fast enough for the real-time demand of manipulator operation.

Fig. 6 is the 3D visualization of recognition results rendered by OpenGL, and images captured by camera are also presented simultaneously for monitoring. In current implementations, the background of the experimental scene is set simple for a easy preprocess, thus some improvements of obtaining a clear point cloud in varying brightness and complex background should also be considered for further applications.

VIII. Conclusion

In this paper, a real-time shape recognition algorithm using SOM is proposed to track a deformable link in home service environments. First, a concise kinematic model suitable for articulated objects with rotational symmetrical components is presented. Then in our proposed SOM algorithm, LSQ is utilized as initialization for correct topological order and proper data association. It is also used in reinitialization part in order to improve the robustness to noise and occlusion. Furthermore, the algorithm is accelerated for real-time demand by applying parallel computation with GPU. The recognition accuracy is verified by our error metric SDF together with the online visualization.

Although this method is designed based on the deformable link, it can be easily extended to a group of soft objects or articulated objects. It can meet the real-time demand with high accuracy, and experiment devices are easy to set up for practical applications. In our future work, this method will be extended for the kinematic calibration of the whole deformable manipulator by dealing with multiple links in the same frame.

References

[1] G. Li, S. Xu, L. Sun, and J. Liu, "Kinematic-free position control for a deformable manipulator," in *Chinese Control Conference*, 2016, pp. 10 302–10 307.

[2] G. Li, L. Sun, S. Xu, D. Song, and J. Liu, "A hybrid model and kinematic-free control framework for a low-cost deformable manipulator using in home service," in *IEEE International Conference on Automation Science and Engineering (CASE)*, 2016, pp. 1002–1007.

[3] S. Xu, G. Li, J. Liu, and J. Hao, "Inverse kinematics solution of deformable manipulator for point touching task," *Robot*, vol. 39, no. 4, pp. 405–414, 2017.

[4] G. Li, L. Sun, X. Lu, J. Hao, and J. Liu, "A practical, fast, and low-cost kinematic calibration scheme for a deformable manipulator by using leap motion," in *IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 2016, pp. 719–724.

[5] A. Vandini, C. Bergeles, B. Glocker, P. Giataganas, and G.-Z. Yang, "Unified tracking and shape estimation for concentric tube robots," *IEEE Transactions on Robotics*, vol. 33, no. 4, pp. 901–915, 2017.

[6] E. J. Lobaton, J. Fu, L. G. Torres, and R. Alterovitz, "Continuous shape estimation of continuum robots using x-ray images," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2013, pp. 725–732.

[7] A. Vandini, C. Bergeles, F.-Y. Lin, and G.-Z. Yang, "Vision-based intraoperative shape sensing of concentric tube robots," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015, pp. 2603–2610.

[8] A. Reiter, A. Bajo, K. Iliopoulos, N. Simaan, and P. K. Allen, "Learning-based configuration estimation of a multi-segment continuum robot," in *IEEE RAS & EMBS International Conference on Biomedical Robotics and Biomechatronics (BioRob)*, 2012, pp. 829–834.

[9] J. M. Croom, D. C. Rucker, J. M. Romano, and R. J. Webster, "Visual sensing of continuum robot shape using self-organizing maps," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2010, pp. 4591–4596.

[10] S. Elayaperumal, J. C. Plata, A. B. Holbrook, Y.-L. Park, K. B. Pauly, B. L. Daniel, and M. R. Cutkosky, "Autonomous real-time interventional scan plane control with a 3-d shape-sensing needle," *IEEE transactions on medical imaging*, vol. 33, no. 11, pp. 2128–2139, 2014.

[11] C. Shi, S. Giannarou, S.-L. Lee, and G.-Z. Yang, "Simultaneous catheter and environment modeling for trans-catheter aortic valve implantation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2014, pp. 2024–2029.

[12] J. Hoschek, D. Lasser, and L. L. Schumaker, *Fundamentals of computer aided geometric design*, 1993.

[13] T. Speer, M. Kuppe, and J. Hoschek, "Global reparametrization for curve approximation," *Computer Aided Geometric Design*, vol. 15, no. 9, pp. 869–877, 1998.

[14] T. Schmidt, R. Newcombe, and D. Fox, "Dart: dense articulated real-time tracking with consumer depth cameras," *Autonomous Robots*, vol. 39, no. 3, pp. 239–258, 2015.

[15] B. Guo, "Surface reconstruction: from points to splines," *Computer-Aided Design*, vol. 29, no. 4, pp. 269–277, 1997.

[16] H. Hoppe, T. DeRose, T. Duchamp, M. Halstead, H. Jin, J. McDonald, J. Schweitzer, and W. Stuetzle, "Piecewise smooth surface reconstruction," in *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, 1994, pp. 295–302.

[17] M. Eck and H. Hoppe, "Automatic reconstruction of b-spline surfaces of arbitrary topological type," in *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, 1996, pp. 325–334.

[18] T. Kohonen, "The self-organizing map," *Neurocomputing*, vol. 21, no. 1-3, pp. 1–6, 1998.

[19] R. M. Palenichka and M. B. Zaremba, "Multi-scale model-based skeletonization of object shapes using self-organizing maps," in *International Conference on Pattern Recognition*, vol. 1, 2002, pp. 143–146.

[20] A. Baader and G. Hirzinger, "A self-organizing algorithm for multisensory surface reconstruction," in *IEEE/RSJ/GI International Conference on Intelligent Robots and Systems (IROS)*, vol. 1, 1994, pp. 81–88.

[21] J. Barhak and A. Fischer, "Parameterization and reconstruction from 3d scattered points based on neural network and pde techniques," *IEEE Transactions on Visualization and Computer Graphics*, vol. 7, no. 1, pp. 1–16, 2001.

[22] P. Felzenszwalb and D. Huttenlocher, "Distance transforms of sampled functions," Cornell University, Tech. Rep., 2004.