A Task Description System for Multi-Fingered Robotic Hand with Distributed Touch Sensor

Hajime Sugiuchi[1], Yuki Hasegawa[2], Shinichiro Watanabe[3], Masashi Nomoto[4], and Hirofumi Yagi[5]

[1] Yokohama National University, Yokohama 240-8501, JAPAN,
sugi@srl.me.ynu.ac.jp
[2] Yamatake Techno-Systems Co., Ltd., Yokohama 220-0004, JAPAN,
yuki@yts.yamatake.co.jp
[3] Yokohama National University, Yokohama 240-8501, JAPAN,
nabe@srl.me.ynu.ac.jp
[4] The University of Tokyo, Tokyo 113-8656, JAPAN,
nomoto@jsk.t.u-tokyo.ac.jp
[5] United System Engineers Inc., Shiojiri Nagano 399-0651, JAPAN,
yagi@use-net.co.jp

Abstract. Two types of human mimic robotic hands are developed to achieve dexterous hand works in human mimetic approach. These hands have five fingers and covered with distributed touch sensor that has more than 500 measuring points. The control system of this hand can control the position, orientation, velocity and force of multiple user specified points on the hand simultaneously. The event driven task description system is also developed. By using this system, a pair of scissors handling task and chopstick handling task are implemented successfully.

1 Introduction

In the future, many robots will work together with human while sharing the environment like home for the purpose of housekeeping aid and nursing of aged people and so on. This shared environment is highly optimized for human. It will be very convenient that robots can share human tools such as a pair of scissors, fork, knife, screw driver and so on. The robot should be able to perform many kind of tasks without replacing it's hand because of efficiency. Some robotic hands have been developed [1],[2],[3],[4] but their configuration are different a little from human's because of cost or mechanical reason. Many kinematic analyses have been made for the hand works such as [5], [6]. We consider the human mimic robot hand and the human mimic task implementation approach should be suitable for such tasks at home. This type of robot hand will take advantage for the task implementation because it has same size, same shape and same geometry with human's hand and so can imitate human's way easier than other type of hands. In addition, physical characteristics of hand surface also should be similar to human skin because the softness of human skin supports stable holding of object and the round shape of hand supports the smooth handling. When manipulating something by hand, our hand usually contacts with the handling object on multiple points not only on fingertips but also possibly all side of fingers and palm, and force control at the contact point should be very important.

We developed two types of robotic hands. Both of them have 5 fingers and covered with the distributed touch sensor which is proposed by Prof. Shimojo[7] behind the soft rubber skin. We also developed the control system which can control the position, velocity or force of multiple locations on the hand simultaneously. The control system should also be able to work well enough even if the inconsistent references are given on different points and it will be preferable to notify the confliction between the references to the operator.

The event driven task execution system is also developed. Generally, the object handling task by hand is divided into several actions. Each action has it's own contact condition with the object and control references. This contact condition will change to the next condition while one action is going on. The built in event checking mechanism watches the change of contact condition according to the user specified conditions and the task execution system switches the action to the next action when event fires. This action sequence leads the robotic hand system to the goal of task.

2 Features of developed robotic hand

In this section, we introduce two types of developed hand. Both of them are planned as development platform for control software. They are not so suitable for practical use but have enough performance for slow experimental

task implementation. The new one is the right hand and not attached to the arm yet. This hand has four 3 D.O.F. fingers and a 4 D.O.F thumb. Each finger has 4 joints but the last joint is coupled to the next one as human. In addition, it has one extra joint on the palm. This joint is necessary for handling chopsticks because it extents the movable area of ring finger and little finger. The ring finger cannot reach to the lower chopstick without this joint. So, totally this hand has 17 D.O.F. All active joints are driven by the servo unit for the radio control model. The size of hand is approximately twice of human hand and it's weight is about 2.5kg. The size is adjusted to the size of left hand. The photo. of this hand is shown in figure 1. The thumb, forefinger, middle and ring finger and a part of the palm are covered with distributed touch sensor which can detect the intensity of pressure force and it's location. The arrangement of measuring points are shown in figure 2. All measuring points are scanned within 20ms. The chopstick handling task is performed by this hand.

Fig. 1. The photo. of right hand.

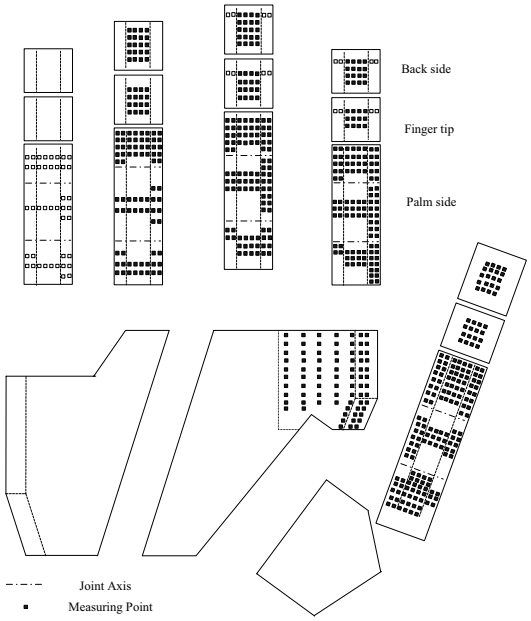

Fig. 2. The measuring points arrangement of right hand.

The other is left hand and attached to the end of industrial robotic arm (Yasukawa K3S). The size of left hand is adjusted to the size of the arm. The left hand is same with right hand except palm joint. So, it has 16 D.O.F and the total D.O.F including the arm is 22. The photo. of this hand is shown in figure 3. The task handling a pair of scissors is performed by this hand arm system.

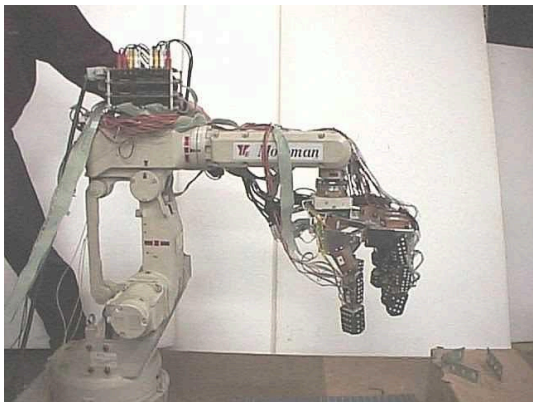

Fig. 3. The photo. of left hand arm system.

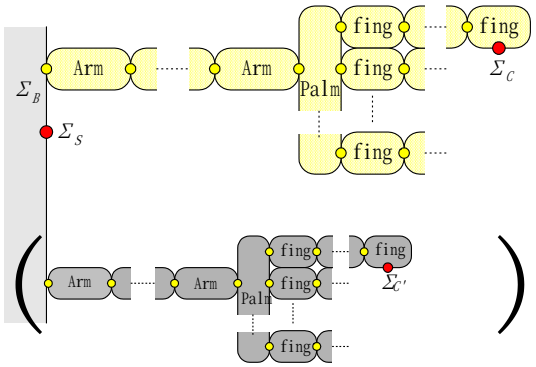

Fig. 4. The supported link configuration in our control system.

3 Control system

In the dexterous handwork task, we use not only the finger tip but also the left, right and back side of finger, palm and various locations on our hand. The hand contacts with the handled object on multiple locations while sensing the position and force of contact points. These mean that the position and force of multiple location on the robot hand should be controlled. These facilities are implemented as follows;

In our hand control system, the position and force references of control points are transformed to the velocity references of corresponding control points. The velocity references of all control points are transformed to the joint angular velocity reference vector by using the Jacobean matrix. The joint angular velocity reference vector is passed to the joint controller. The arm part of our system is usual industrial robot manipulator and the joint angular velocity control is performed by the simple PI controller. The servo unit for the radio control model is used for the finger joint. In the hand part, the joint angular velocity reference is integrated and given to the servo unit because it has joint angle feedback loop internally. The feedback loop gains of RC servos are quite high and they have enough accuracy for slow action. But, if possible, some back drivable torque sources should be preferable for fast motion and, for faster motion, some nonlinear compensation can be implemented to improve the accuracy of joint angular velocity control by using such kind of actuator.

The maximum number of control points is limited by the computational ability of the controller. The Pentium III 500 MHz processor can handle up to about 10 control points within the sampling period (5ms).

3.1 Supported configuration of robotic hand

The supported link configuration of our system is quite general and flexible. Our system supports the open link chain with multiple branches as shown in figure 4. Links may branch at any link. The control related link parameter description is implemented as the class library in C++ programming language and the robot hand configuration is easily built into the system as the C++ object. Either left and right hand system can be supported by replacing the configuration file. Even the dual hand dual arm system will be supported easily by assuming the first link of each arm as a branch link of base link.

3.2 Control of one point on the hand

Velocity control The velocity control is the base of our control system. The relation between the *i*-th control point velocity vector v_i in Cartesian space and the joint angular velocity vector $\dot{\theta}$ is

$$v_i = \mathbf{J_i}(\theta)\dot{\theta} \tag{1}$$

Here, $\mathbf{J_i}(\theta)$ is the Jacobean matrix of *i*-th contact point. The velocity vector v_i includes both prismatic and angular velocity. This leads the relation between the contact point velocity reference vector v_{ri} and joint angular velocity reference vector $\dot{\theta_r}$ as follows.

$$v_{ri} = \mathbf{J_i}(\theta)\dot{\theta}_r \tag{2}$$

The robot hand will achieves v_{ri} by giving $\dot{\theta}_r$ which satisfies the equation 2 to the joint angular velocity controller.

Position control The reference position p_{ri} and reference orientation $\mathbf{R_{ri}}$ in rotational matrix form are given. The actual position and orientation of the *i*-th control point are p_i and $\mathbf{R_i}$ respectively. The positional error vector Δx_i in 6 dimension is defined as follows.

$$\Delta p_i = p_{ri} - p_i \tag{3}$$

$$\mathbf{R_{ri}}{\mathbf{R_i}}^{-1} \approx \begin{pmatrix} 1 & -\Delta\gamma & \Delta\beta \\ \Delta\gamma & 1 & -\Delta\alpha \\ -\Delta\beta & \Delta\alpha & 1 \end{pmatrix} \tag{4}$$

$$\Delta x_i = \begin{pmatrix} \Delta p_i \\ \Delta\alpha \\ \Delta\beta \\ \Delta\gamma \end{pmatrix} \tag{5}$$

By using Δx_i, the velocity reference of contact point v_{ri} is given as follows.

$$v_{ri} = \mathbf{K_{pi}} \Delta x_i \tag{6}$$

Here, $\mathbf{K_{pi}}$ is user defined 6 by 6 position control gain matrix. It should be chosen not to violate the system stability. Thus the position control is transformed into velocity control.

Force control The reference force f_{ri} is given and the actual force on the *i*-th contact point f_i is detected by touch sensor. Both values are scalar because our distributed touch sensor can detect only normal force on the surface. z_{ni} is the normal vector at control point. The force error vector Δf_i in 6 dimension is defined as follows.

$$\Delta f_i = \begin{pmatrix} (f_{ri} - f_i) z_{ni} \\ 0 \\ 0 \\ 0 \end{pmatrix} \tag{7}$$

By using Δf_i, the velocity reference of contact point v_{ri} is given as follows.

$$v_{ri} = \mathbf{K_{fi}} \Delta f_i \tag{8}$$

Here, $\mathbf{K_{fi}}$ is user defined 6 by 6 force control gain matrix. It also should be chosen not to violate the system stability. Thus the force control is also transformed to velocity control.

3.3 Motion integration of all contact points in one action

All reference values are converted to corresponding velocity references as mentioned above therefore the equation 2 is available for all control points. All equations are piled up and following equation is built.

$$V_r = \mathbf{J}(\theta) \dot{\theta_r} \tag{9}$$

Here, n is the number of control points and

$$V_r = \begin{pmatrix} v_{r1} \\ \vdots \\ v_{rn} \end{pmatrix}, \mathbf{J}(\theta) = \begin{pmatrix} \mathbf{J_1}(\theta) \\ \vdots \\ \mathbf{J_n}(\theta) \end{pmatrix} \tag{10}$$

The integrated velocity reference vector V_r is a quite large 6n dimension vector and $\mathbf{J}(\theta)$ is integrated Jacobean matrix. The size of Jacobean matrix is 6n by 22 in left hand arm system and 6n by 17 in right hand system. We will be able to obtain the joint angular velocity reference vector $\dot{\theta_r}$ that realize V_r by solving the equation 9 but, in the case that the system has many control points, the equation 9 falls into over constrained situation and is impossible to solve. Even in such case, the control system can minimize the velocity error.

V is the integrated velocity vector which can be realized by the joint angular velocity vector $\dot{\theta}$. The integrated and weighted velocity error E is defined as follows.

$$E = (V_r - V)^t \mathbf{C}^t \mathbf{C} (V_r - V) \tag{11}$$

Here, $\mathbf{C}$ is a user defined weight matrix.

The optimal joint angular velocity reference vector $\dot{\theta}_{ropt}$ that minimizes E is obtained as follows.

$$\dot{\theta}_{ropt} = (\mathbf{J}(\theta)^t \mathbf{C}^t \mathbf{C} \mathbf{J}(\theta))^+ \mathbf{J}(\theta)^t \mathbf{C}^t \mathbf{C} V_r \tag{12}$$

Here, $^+$ means the Moore Penrose's generalized inverse matrix[8].

The Moore Penrose's generalized inverse matrix is unique for any matrix including zero matirx and minimizes not only E but also the norm of $\dot{\theta}_{ropt}$. So, the equation 12 is applicable to both over constrained and redundant cases. Thus the motion of each control point is integrated to the action of hand.

It should be noticed that the system cannot satisfy all of the given references and automatically compromises to minimize the error E in the over constrained case. But you can detect such situation by watching E.

4 Task description system

Our objective is to realize dexterous tasks by using robotic hand. We adopted human mimic approach to implement dexterous tasks with short time. Some system which can describe dexterous tasks easily and efficiently should be necessary for this approach. We call it as task description system. Unfortunately, the task planning has not been automated yet and the task should be programmed manually by using this system.

In order to inherit the implemented tasks to the successive new hardware, it is preferable that the task description has physically recognizable structure. So, we developed the contact position based task description system. In our system, hand action is described as follows; At first user select the contact points to the handling object on the hand, then give the position, velocity or force reference of each contact point to specify the motion. All parameters in the action are physically meaningful.

The easy error recovery description should be considered because the robot program is quite different from computer program on the point that robot works in uncertain real world and, as a necessity, the error recovery is important in the robot programming.

In addition to the mentioned above, the new functions should be implemented easily. The system should start from the minimal function set and gradually grow by implementing various tasks one by one because it is quite difficult to define the necessary and sufficient function set.

4.1 Task programming

The dexterous hand work task is divided into several Actions. The robot hand contacts with the handling object(s) and/or environment at several points. We call this set of contact conditions as State of hand. The State restricts the possible action of hand and, on the contrary, the State will be changed by the Action and so the Action should be associated with the State. The change of State means the change of contact condition of the hand. When the contact condition changes, some signal will be detected from the sensor. We call this signal as Event. Thus, the hand task will be executed as follows. The hand starts form initial State and execute the specified Action. The State of the hand is changed by it's own Action and this is detected by the Event. The Event directs the next State to the system.

The task is described by the network that is constructed from the States, Actions and Events. At first, the task is broken down to the Actions. Each Action is combined to the State and each State is connected by the Event. We call this network as State Action Event Network (SAEN). All SAEN components are implemented as the class library in C++ programming language and the task program is also described in C++ by using those components. It is compiled and linked with other control library.

4.2 Action

It will be enough to specify the motion of contact points in order to specify the motion of hand. All locations on the hand surface where the hand contacts with the object or environment and the motion of each contact point is specified in the Action. The motion is specified by the given position, velocity or force reference. The contact points which have different kind of reference can be mixed as you like in one Action. The position and force reference of each contact point are converted into the velocity reference and velocity references of all contact points are integrated as mentioned before.

The reference trajectory generation is quite simple in our system. At the position controlled point, the destination position and velocity in the Cartesian coordinates and the duration of trajectory should be given and the system generates the reference trajectory on each sampling period by the interpolation with the 3rd order polynomial of trajectory time. The actual position and velocity are used as the initial condition at the beginning of the Action. If the system still remains in the Action beyond specified duration, the system generates the reference trajectory which keeps the destination velocity. At the force controlled point, the destination force value and the duration are given. The reference trajectory is interpolated by the 1st order polynomial and the system keeps the destination value after the specified duration. It is possible to generate complicated reference trajectory by connecting several Actions.

4.3 Event and EventList

The Event is watching condition of change of the State. Because it should work in uncertain physical environment, the hand does not always transit to the expected State by the Action and sometimes falls into wrong state. The

Event detecting mechanism also accidentally can make some mistakes. You can add more than one Events on the State for the detection of error condition and each error detecting Event directs the State which has corresponding error recovery action. We call this set of Events associated with one State as EventList. Any event in the EventList can cause the State transition so they are ORed. The AND connection of Events is not supported yet. By setting the destination of several Events to the same State, you can probably improve the detecting accuracy. Some additional Events are implemented for the programming aid.

Only a few kinds of Event are implemented to our system in the current version. The ReachedToForce Event fires when the force signal at the specified location on the hand reached and beyond the specified value. It is used to detect the new contact. The appearance of new contact means the change of contact condition. The ReachedToPosition event fires when the specified location on the hand reached to the specified position within the allowable error. The error range is specified by the users. The TimeUp event fires when the specified time has been past in the state. It is used to make long trajectory including via point by connecting several actions. The DistanceWatch event fires when the distance between the specified two locations on the hand approach within specified distance. It is useful to avoid the interfere between fingers. The WatchIntegratedError event fires when the error E in equation 11 exceeds the user specified value. It can be used to watch that the Action is properly arranged.

4.4 State

Originally, the State represents the set of contact constrained conditions. But now, the contact condition is watched by the eventlist and the State works as a container of the Action and EventList. The main control loop of our system is implemented in the State class. The State object controls the hand according to the Action while checking the specified EventList and switches the Action and the EventList when the Event fires.

5 Experiments

The effectiveness of our system is demonstrated by two kinds of experiment. The first experiment is a pair of scissors handling by the left hand arm system. The 22 D.O.F left hand arm system cut a sheet of paper by using a pair of scissors. It is too difficult for our system to fetch a pair of scissors on a desk because our system has no vision system but the remainder part of the task is still complex enough to show the effectiveness of our system. It takes only around one week to implement this task including control parameter tuning. The human mimic approach brings us this short time implementation. The second one is the chopsticks handling experiment. The 17 D.O.F right hand system holds chopsticks stably and open/close them successfully. The implementation of this task was also achieved within one week. Both task programs are quite simple and have no error recovery action as shown below. If some serious error conditions are found, you can easily attach the recovery action by adding proper Event and Action. You need not to implement recovery action for all errors because the necessity of error recovery depends on the possibility of error occurrence.

5.1 A pair of scissors handling task

Here, the execution mechanism of our SAEN system is illustrated. The program flow of this task is shown in figure 5. The task is divided into 7 States(Actions). The task starts from “Start” state and goes on along to the state number. The 4th through 7th states configure loop. Each state has specific Action and EventList. A part of the Action and Event setting of each state is as follows.

In the setup state, the hand goes to the wait posture. The position of 9 control points and the duration for the motion are specified as shown in figure 6. After the specified duration, the time up event fires and the system goes to the next wait state.

In the wait state, the hand system keeps the given posture and waits to be equipped with a pair of scissors by a human operator. By touching the fingertip of little finger, the system goes to the next hold state.

In the hold state, 3 position control points are specified as shown in figure 7. The wrist of robot hand is fixed and the forefinger and middle finger are spread slowly by the given position references to hold a pair of scissors while watching the force at the outer side of forefinger. The force control point may be more suitable for holding a pair of scissors but we adopt position control to ensure the holding form of hand. The system goes to the next close state when the watching condition is satisfied.

In the close state, the system controls the velocities at 6 points as shown in figure 8. A pair of scissors is closed by forefinger, middle finger and thumb. Therefore 3 points at the fingertip of each finger are enough for the close motion. The other control points are added to suppress the unnecessary motion and keep the proper posture.

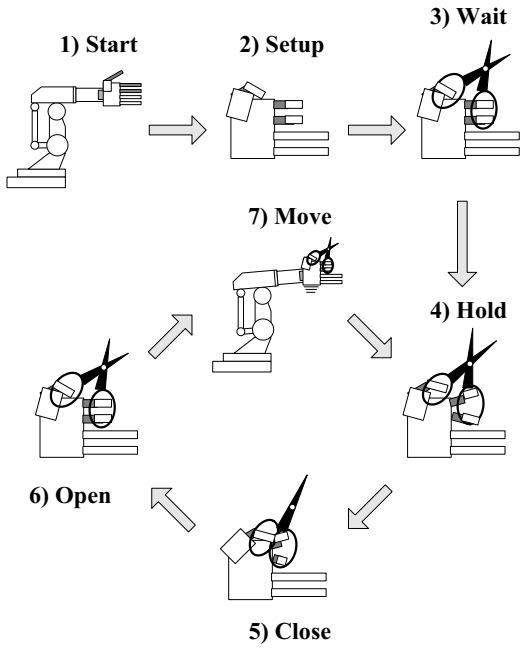

Fig. 5. Task description of a pair of scissors handling experiment.

2 events are specified in the event list to detect the end of this action. The forces are watched at the foreside of fingertip of forefinger and thumb. Each event can have it's own destination state but the same open state is specified here. So the system goes to the open state when either event fires. You can write a branch code by setting a different state as the destination of each event.

In the open state, the control points setup is similar to the close state as shown in figure 9. But the direction of velocity references are inverted and only one force detect event is specified at the back side of the forefinger because this state is not so stable as the close state and the force detected at the backside of the thumb is not suitable for the reliable action. The system goes to the next move state when the fingers reached to the open end and the detected force reached to the specified value.

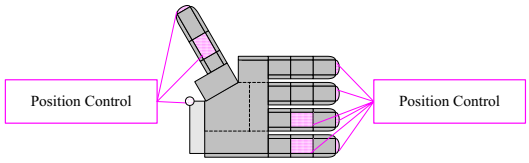

Fig. 6. The references and events setting in the setup state.

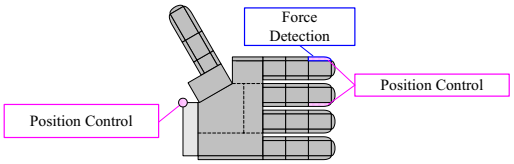

Fig. 7. The references and events setting in the hold state.

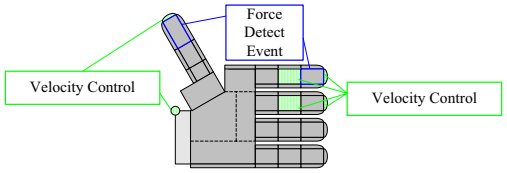

Fig. 8. The references and events setting in the close state.

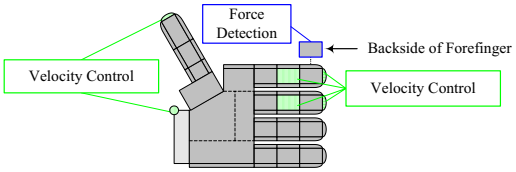

Fig. 9. The references and events setting in the open state.

In the move state, only one velocity control point on the wrist is specified. The hand moves forward for the next stroke. No control point exists on any finger and all fingers seem to be free. But no finger moves in this state

because the system works to minimize the norm of the angular velocity reference vector. The system goes back to the hold state after the specified duration.

The experiment in which our robotic hand holds a pair of scissors and achieves a paper cutting task by using the task program is shown in figure 10 through 13.

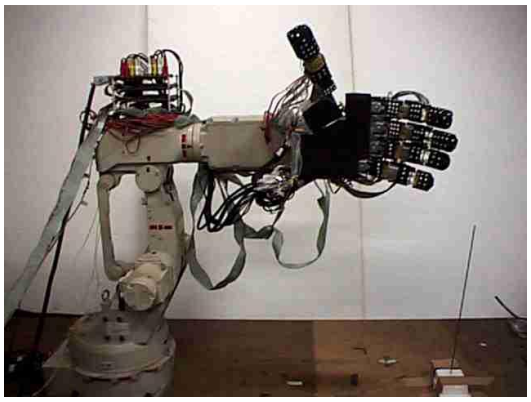

Fig. 10. The photo. of robotic hand in the start state.

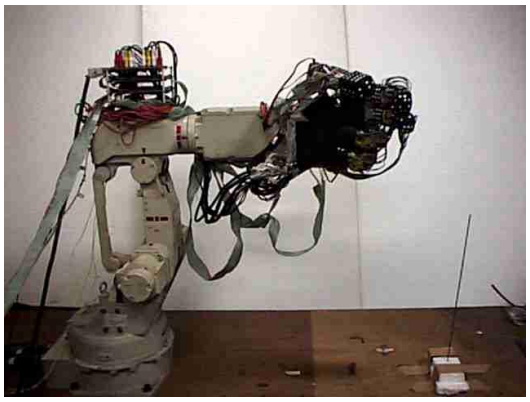

Fig. 11. The photo. of robotic hand in the setup state.

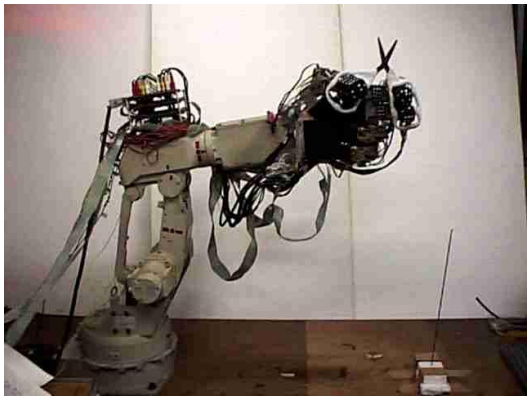

Fig. 12. The photo. of robotic hand in the hold state.

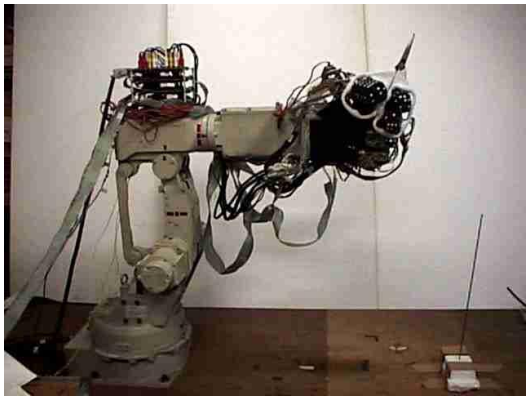

Fig. 13. The photo. of robotic hand in the close state.

5.2 Chopsticks handling task

The program flow of chopsticks handling task by the right hand is shown in figure 14. The task is divided into 6 States(Actions). The task starts from the initial state and goes on along to the state number. The position references of 6 locations on the hand are specified for preparation from state 1 to 3 as shown in figure 15. During these states, the operator inserts chopstick one by one according to the posture of the right hand.

The 4th through 6th states have the same control points setting with each others as shown in figure 16. The hand holds chopsticks in the 4th state. The index, middle and ring fingers form a proper posture by using position control and The thumb holds both chopsticks by using force control.

In the 5th state, the hand opens chopsticks by the position control specification on the fingertips of index and middle finger. In the 6th state, the positional references of index and middle fingers are updated and the hand close chopsticks while watching the force exerted on the fingertip of index finger as shown in figure 17. The 5th and 6th state configure the loop and the hand repeats open and close actions.

The photo. of right hand are shown in figure 18 and 19. In figure 18, the hand holds chopsticks successfully. In figure 19, the hand picks a soft sponge cube by chopsticks. By watching the arising force on the forefinger fingertip, the system can detect that a soft sponge cube is picked on the end of chopsticks.

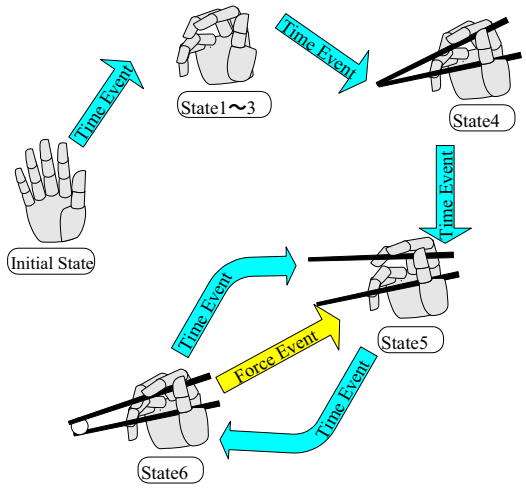

Fig. 14. Task description of chopsticks handling experiment.

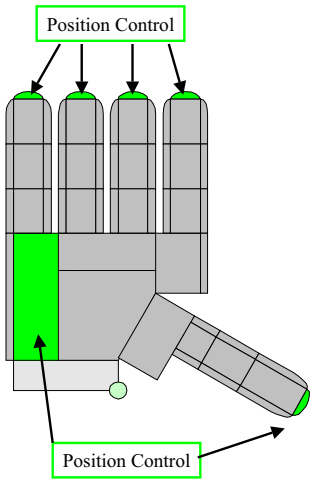

Fig. 15. The control points setting of State 1 to 3.

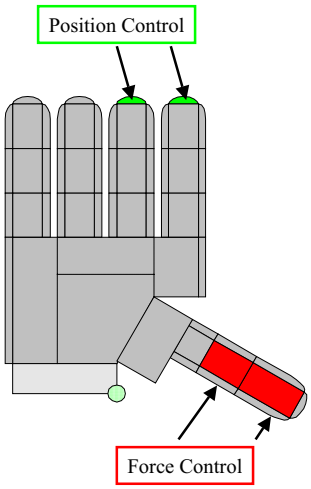

Fig. 16. The control points setting of State 4 to 6.

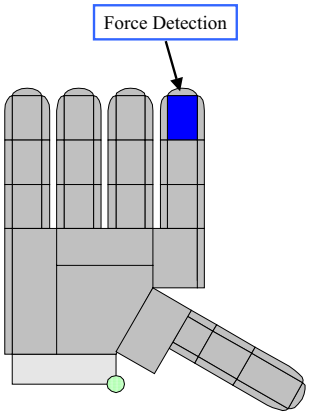

Fig. 17. The event setting for object picking.

6 Conclusions

We have developed two multi-finger robotic hand system. Each developed hand has five fingers and the surface of hand is covered with the distributed touch sensor. A control system for them is proposed. This system is able to control the position, orientation, velocity or force of multiple points on the hand surface simultaneously. A task description system is also developed. The system is able to detect the change of constrained condition of the hand and change the control points setting dynamically according to the detected event. It is demonstrated by the experiment that the proposed control system has enough ability to execute complicated and dexterous hand work. It is also shown by the rapid implementation of complicated and dexterous hand work that our task description system is very effective for dexterous hand task programming and our human mimic approach is suitable for the task realization.

Acknowledgments

The authors wish to gratefully acknowledge Toshihiro Sano, Toshiaki Ishihara and Takeharu Kitagawa for the development of joint level control and the force sensing system.

Fig. 18. The photo. of right hand in holding chopsticks.

Fig. 19. The photo. of right hand in picking a sponge cube by chopsticks.

References

1. Salisbury, J.K.: Design and Control of an Articulated Hand. Int'l. Symp. on Design and Synthesis, Tokyo, July 1984.
2. André van der Ham, A Dexterous Teleoperator for Hazardous Environments. the Universiteitsdrukkerij Delft, 1997.
3. Oomichi, T. et al, Mechanics and Multiple Sensory Bilateral Control of a Fingered manipulator. 3rd ISRR, MIT Press, pp.148-149, 1987.
4. Oomichi, T. et al, Development of Working Multifinger Hand Manipulator. IROS'90, IEEE, pp.873-880, 1990.
5. Bicchi, A. et al, On the Mobility and Manipulability of General Multiple Limb Robots. IEEE Trans. Robot. and Automat., vol. 11, no. 2, Apr. 1995.
6. Sinha, P. R. et al, A Contact Stress Model for Multifingered Grasps of Rough Objects. IEEE Trans. Robot. and Automat., vol. 8, no. 1, Feb. 1992.
7. Shimojo, M. et al, A Flexible High Resolution Tactile Imager with Video Signal Output. IEEE Int. Conf. Robotics and Automation, pp. 384-391, 1991.4.
8. Penrose, R., A generalized inverse for matrices. Proc. Cambridge Philos. Soc. 51, 406-413, 1955.