# Towards Fully Automated Pick and Place Operations of Individual Natural Fibers

Mathias von Essen

Juha Hirvonen

Seppo Kuikka

Pasi Kallio
Department of Automation Science and Engineering
Tampere University of Technology, Tampere, Finland
mathias.vonessen@tut.fi

***Abstract*—This paper reports automated image-based pick and place procedures for manipulation of individual natural fibers. The developed procedures are part of an effort to develop a fully automated microrobotic-based platform for fiber characterization. The presented procedures are divided into unit operations, which can reused in multiple tasks that the platform must perform. Two different demonstrations: pick and place, and coordinated fiber lifting are presented. In addition, a component-based software that promotes reusability of the developed unit operations is presented.**

***Keywords—microrobotics automation, component-based software***

## I. Introduction

Micro- and nanorobotic technologies experienced a rapid evolution during the past decade and they have enabled research and assembly of objects that is out of reach for conventional techniques. The technological maturity has reached sufficient level for commercial applications, however, the breakthrough of micro-and nanorobotic systems in industrial scale high throughput applications has not been seen. Before micro- and nanorobotic systems have the possibility to succeed in high throughput industrial applications, the level of automation must increase, i.e. development of automated or even autonomous systems is essential.

One of the potential high throughput industrial applications of microhandling is the characterization of natural fibrous materials that are construction units for broad spectrum of products. The material properties of fiber-based products are largely determined by the properties of the underlying fiber network and its two main components: the fibers and the bonds between the fibers. Despite this central role of fibers, their research has still many shortcomings. One of the main drawbacks being the lack of reliable measurement data concerning especially the fiber-fiber bonds and fiber networks, but also the characteristics of individual fibers. This paper focuses on manipulation of wood fibers, but the described approach is also applicable to other natural fibrous materials.

In paper research, the current methods for determining material properties are mainly based on sheet level measurements, such as z-directional tensile test, shear cohesion test and Scott bond test. The results of these tests have high correlation with material properties, but they also contain undesirable information that cannot be decoupled. For example: the Z- directional tensile test encompasses both intra- and inter-fiber bonding energies, the shear cohesion test combines the force required to shear the bond with the force acting on the plane of the sheet, and the Scott bond test over evaluates the bond strength because of its dynamic nature [1].

The advances of microrobotics and microsystems technology have provided new tools and methods to manipulate and characterize micro- and nanoscale samples such as living cells [2-7] carbon nanotubes [8, 9], and also promote microassembly approaches [10-13]. In fiber research, the aforementioned technologies enable characterization of individual fibers and fiber-fiber bonds. The methodology provides an attractive alternative to conventional characterization procedures, as the obtained results are not coupled with undesired information. The potential of microrobotic-based characterization of paper fibers has been exhibited in [14-16], where several different teleoperated fiber level characterization procedures, including (i) flexibility measurement of individual fiber and (ii) fiber-fiber bond strength measurement have been reported. In order to fully utilize the benefits of the presented fiber-level characterization capabilities, the described procedures must be automated. Manipulation of individual fibers requires dexterous high-precision operations, which are often beyond human capabilities. High throughput is essential as thousands of fibers must be characterized to produce statistically reliable data.

Complex automated systems are highly dependent on the robustness and quality of their control software. However, the important role of software is often neglected in the development of microrobotic systems. In a typical case, the software is interfacing in-house built custom hardware to perform a specific task. This often results in a custom designed monolithic software architecture that is incapable of accommodating to even slightest changes in the hardware configuration or in the performed task. As a consequence, vast number of the developed robot functionality, algorithms and control schemes cannot be reused as such, but need to be completely rewritten for each application. Constant reinvention of the already existing functionality leads not only to huge waste of resources, but also hinders development of robust source code.

One approach for developing modular and reusable source code for microhandling applications is to divide the performed manipulation tasks into unit operations. The designed control software should then implement and allow integration of the unit operation in such a way that the system integrator can create different sequences from existing unit operation implementations. The goal is that recurring usage of the unit functions should not require writing of additional source code. Moreover, a developer who wishes to reuse existing unit operations should not be required to have in depth knowledge of the unit operation's implementation.

From software engineering perspective, component-based software engineering (CBSE) is a suitable paradigm to cover the described needs. In software engineering, a component is a unit that encapsulates specific functionality. In the robotics domain, the functionality could be, for example, an algorithm, a control paradigm or a device driver. However, CBSE is mainly a detailed design and implementation approach and as such does not take stance on generic architectural aspects such as distribution or scalability. These supportive, but necessary aspects are responsibilities of software framework. A typical robotic software framework (RSF) provides a component model, communication middleware (usually third-party solution, such as CORBA[1]) and mechanisms to manage components state and life cycle. In industrial and service robotics, the growing need to reuse source code implementing robotic algorithms and device drivers has resulted in development of multiple robotic software frameworks (RSFs). RSFs target at typical robotic problems such as navigation, motion planning, real-time control, distribution or mobility within robotics. Well known open-source RSFs include Player [17], ORCA [18], OROCOS [19], OpenRAVE [20], ROS [21], YARP [22], OpenRTM [23], OPRoS [24], and OpenRDK [25]. Even though microrobotic applications have their own distinctive characteristics when compared to industrial and service robotics, the existing RSFs can provide significant benefits through reduced development time and solid code base. For example, construction of complex real-time systems is extremely time consuming and requires thorough knowledge of operating system, networking, etc. It also requires wide range of testing and validation. In such cases the development cycle can be significantly shortened by utilizing an existing RSF.

[1]Common Object Request Broker Architecture (CORBA) is standard defined by the Object Management Group that allows distributed components to communicate with each other.

The first steps towards automated fiber manipulation were reported in [16]. In this paper, we continue the effort towards a fully automated fiber characterization platform by demonstrating two automated procedures useful in fiber characterization: pick and place; and coordinated lifting. The first procedure is utilized to move individual fibers to a desired location. The latter procedure is required in breaking of fiber-fiber bonds. The procedures are divided into unit operations which are then encapsulated into separate components. The control software presented in this work aims to promote high reusability by applying CBSE. It relies on Orocos RSF, which is an RSF designed for real-time and distributed robot and machine control. The rest of this paper is organized as follows. Section 2 presents set of unit operations and two characterization procedures that utilize the unit operations. Section 3 describes the experimental setup. Section 4 is dedicated for the control software. Section 5 presents results of automated pick and place, and coordinated lifting procedures. Section 6 concludes the paper.

## II. Unit Operations and Fiber Characterization Procedures

Fiber characterization procedures carried out with the developed platform consist of a set of different unit operations. In this work, we present unit operations: Detect, Grasp, Move, Pull, and Release. Two fiber characterization procedures, Fiber Pick and Place (FPP), and Coordinated Fiber Lifting (CFL), are presented as examples of sequences that can be created from the described unit operations. Demonstration of the pick and place and coordinated lifting procedures are provided in Section 5.

### *A. Unit Operations*

Most of the developed unit operation rely on image-based detection of the manipulated fiber or fiber-fiber bond. Detect Fiber is an image-based unit operation which aims to locate and select a fiber that is suitable for characterization. Suitability is determined with a set of parameters, such as curliness and length of the fiber. Moreover, the fiber should not be entangled with other fibers.

Grasping, presented in Fig. 1A, is a unit operation that allows grasping of an individual fiber with the manipulators' microgrippers. Grasping requires utilization of Detect Fiber to determine the suitable grasping points. After successful location of grasping points, microgrippers are moved to the found location (1). The microgrippers are closed after the desired location has been reached (2). Multiple Grasp operations may take place at the same time. This is the case when a fiber is lifted from the substrate in (CFL).

Pull is an operation that is utilized to drag fiber in a plane with a single microgripper. The direction of Pull may vary depending on the application, for example in fiber-fiber bond breaking, the direction of Pull should be orthogonal to the fixed fiber. Another type of application of pull is depicted in Fig. 1B, where Pull is utilized to measure friction between fibers. Pull: Fiber is pulled along its x-axis to desired location

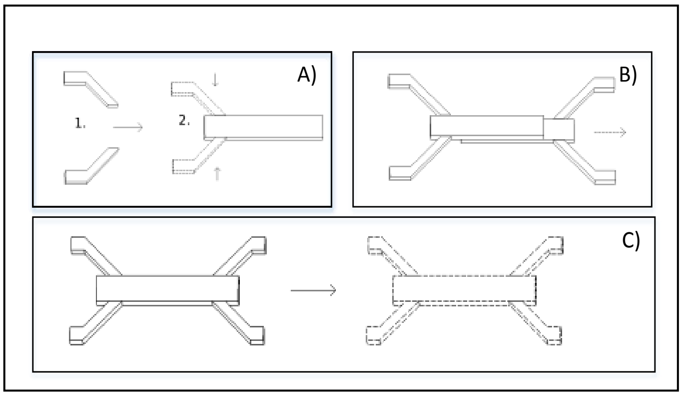


Fig. 1. A) Grasping: Gripper is moved to desired location (1) and closed (2) B) Friction measurement as example of pull; one of the fibers is pulled along its x-axis C) Fiber transported to another location with Coordinated Move operation

Move operation is utilized to move a fiber from one location to another. It requires a successful grasp operation, after which the fiber is lifted up from the substrate and moved into the desired location. Move can be extended into Coordinated Move operation, where the fiber is grasped and moved with two microgrippers, as shown in Fig. 1C.

### *B. Characterization Procedures*

Different fiber characterization procedures can be created by combining the described unit operations.

FPP combines Detect, Grasp, Move, and Release operations. In the first phase, suitable grasping points for a fiber are detected. Next, Grasp operation is executed for the located grasping point. The grasped fiber is then moved to desired location, where the fiber is placed on the substrate by Release.

CFL sequence starts with Detect operation that locates the coordinates of both fiber end-points. After successful detection, Grasp is performed for both end-points of the fiber simultaneously. Finally, the fiber is lifted up from the substrate by performing Coordinated Move operation along z-axis. CFL can be extended to fiber-fiber bond breaking by performing additional Detect, Grasp, Pull sequence. First, the end of the free fiber must be identified with Detect, which is followed by Grasp operation. Finally, the bond is destructed by performing Pull on the fiber that is grasped only from one of its end points.

## III. Experimental Setup

The experimental microhandling platform, presented in Fig. 2, consists of two 4-DOF and one 3-DOF manipulators that are assembled in stacked gantry crane configuration. Grasping and moving of the manipulated fibers is performed with the 4-DOF manipulators, denoted as Manipulator-1 (7) and Manipulator-2 (8), both of which are constructed from three linear micropositioners and a microgripper (SLC-1740, SLC-1730, SG-07, SmarAct GmbH). Samples are picked and placed from special sample areas that are located on the 3-DOF manipulator (Rotary Table). Rotary Table is assembled from two linear and one revolving micropositioner. The linear axes are utilized for correct positioning of a fiber prior to grasping and rotation provides means to align of the fiber along the x-axis of Manipulator-1 and Manipulator-2.

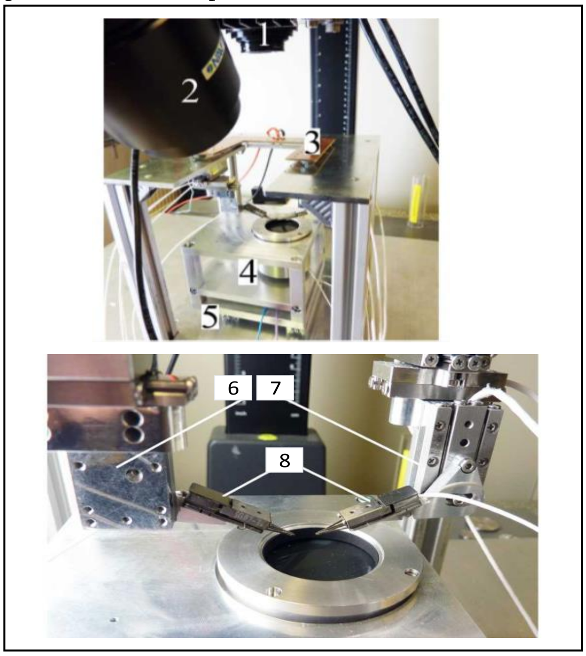


Fig. 2. Experimental setup consists of two cameras equipped with analyzers and motorized optics (1,2), a led array (3), Rotary Table (4,5), Manipulator-1 (6) , Manipulator-2(7) and microgrippers (8)

The manipulated fibers and microgrippers of the manipulators are monitored with imaging system that consists of two cameras (Manta GS-501, Allied Vision Technologies Inc.), both equipped with motorized optics (12x Zoom, Zoom 7000, Navitar Inc.), and two in-house developed illumination elements. The cameras are located above the setup, as depicted in Fig. 2. The illumination is designed so that it allows identification of fibers, but also enables tracking of the pose of the manipulators. Therefore two different approaches are combined in the illumination. To acquire high-contrast images of fibers, a backlight, polarizer and analyzer were used in imaging. When there is a 90˚ difference between the polarizer and analyzer, the polarizer blocks all the light that does not pass the fibers and thus change its polarity. The polarizer and one of the illumination elements are located inside of Rotary Table so that the polarizer's rotation does not change when the table is rotated, as illustrated in Fig. 3. Manipulators are made visible with the additional illumination element which simply provides ambient light.

Fig. 4 presents the overall control scheme of the experimental platform. The low level control of the manipulators is performed with manufacturer specific Simple Control Unit (SCU) and modular control unit (MCS). Where the first control Manipulator-1 and Manipulator-2, and the latter is dedicated for control of Rotary Table. High level control utilizes feed forward control, where image data obtained from the platform's cameras are used as an input.

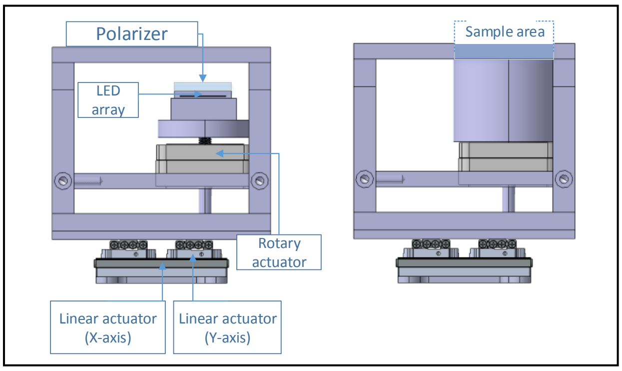

Fig. 3. Rotary Table constructed from two linear and one rotary micropositioners. Sample area is illuminated with stationary led array and polarizer.

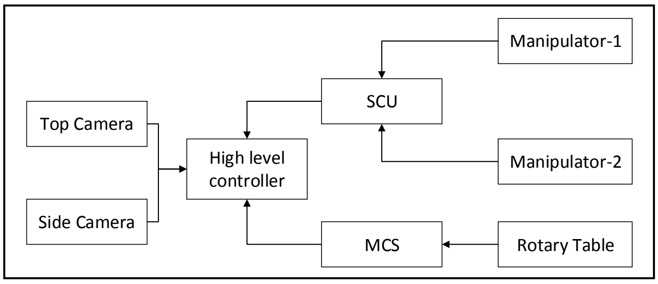

Fig. 4. Control scheme of the experimental platform

## IV. Control Software

The developed control software is based on Orocos RSF. The aspects supporting selection of Orocos were (i) well defined component model, (ii) support for real time operating systems, (iii) active English speaking user group, (iv) good documentation, and (v) possibility to integrate with ROS. Other investigated potential options were OpenRTM-aist and OPRoS. The first is a reference implementation of Robotic Technology Component (RTC) specification from Object Management Group (OMG). The latter is a Korean effort for developing a component-based RSF. Orocos was preferred over OpenRTM-aist, because Orocos hides the middleware implementation from component developer. Middleware specific knowledge is only required when new complex data types have to be created. Further, Orocos has a very active English speaking community which has proven to be the best source of information in many implementation related issues. OPRoS does not currently provide real-time capabilities. However, there are ongoing efforts on this field [27]. Further, OPRoS did not provide integration with ROS at the time of deciding upon the utilized RSF. The rationale of requiring integration with ROS is that it is a widely used RSF and most of other RSFs integrate with ROS. The authors do not want to limit themselves to one single RSF and ability to integrate with ROS provides certain flexibility as different systems or subsystems can be built on different RSFs and intercommunication can be implemented with ROS messages.

### A. Orocos Component Model

Orocos component model defines two distinctive concepts for communication: services and flow ports. Service interfaces are access points through which the component provides or requires operations. Service operations are used in a similar manner as function calls are utilized in programming languages. Flow ports implement a publish-subscribe scheme, where a component with an output port acts as a data publisher and components with input ports can subscribe to listen the incoming data. Life cycle of a component is managed with a simple state-machine each component implements. Upon instantiation, Orocos components enter PreOperational state, where the component remains until it is configured. After successful configuration, a transition to Stopped state is performed. When started, the component enters Running state, where it remains until it is stopped and transitions to Stopped state. The final state of components lifecycle is Cleanup after which the component is desctructed. Orocos component model also includes recoverable and unrecoverable error states, which allow the developers to define appropriate actions in case of exceptions.

### B. Developed Components

In this work we have developed a set of components that implement the unit operations as Orocos components. The manipulation sequences required in the characterization of individual fibers can then be composed by connecting these components appropriately. The developed components can be categorized into two different levels: low level hardware interface components and high level components that implement supportive functionality, algorithms, and characterization procedures.

In the current phase, our component library includes two hardware interface components, which enable communication with the microhandling platform's hardware. *SaController* is responsible for communication with SmarAct's SCU and MCS controllers. It provides a generic service interface through which other components can control the micropositioners. Internally, *SaController* consists of two control classes ScuTcpController and McsController. The first enables socket-based communication with SCUs and the latter controls local MCSs[2]. An instance of *SaController* represents one SmarAct controller, the selection of the internal controller type is performed during the component instance's configuration phase. The second hardware interface component is *GigeAvt*, which provides the ability to control GigE Vision cameras that are manufactured by Allied Vision Technologies Inc. Unlike *SaController*, *GigAvt* does not provide a service interface, but relies on Orocos' flow port concept, where data flows from one component to another. Camera configuration is performed with Orocos' properties. Similarly to *SaController* component, an instance of *GigeAvt* represents a single camera.

The high level components are divided into three groups; supportive, algorithmic and task execution components. Supportive components include image visualization component *ImageViewer* and manipulator abstraction component *SaManipulator*. The latter allows the users to control a set of

[2] Communication with SCUs is currently performed over with socket communication through virtual machine due to operating system related limitations; the currently used SCU supports only Windows operating systems. The MCS controller is supported in Linux and Windows environments.

micropositioners using a simplified interface without having to specify detailed information about the micropositioners. For example, manipulator's pose can be obtained with a single function call and movements can be executed by defining a joint index and a desired position.

The algorithmic components implement different image processing algorithms that are required by the manipulation unit operations. *LocateGraspingPoint* is a component that obtains potential grasping points from images and returns the found coordinate points in manipulator's own coordinate system. *LocateGripper* is a similar algorithmic component that locates markers that are attached to microgrippers. It provides location information of the gripper jaw's location in global coordinate system.

Task execution components utilize other high level components to perform a unit operation. *GraspFiber* is a component which first locates potential grasping point, moves the gripper to the found location and finally closes the gripper. The current implementation does not validate success of the operation.

*ManiCalibration* performs actuator based calibration, where the purpose is to establish a relationship between image coordinates of the cameras and the coordinate systems of both manipulators. The relation of camera's image coordinates and metric coordinates can be described with 3x4 camera matrix, which fulfills the following equation:

$$\mathbf{x} = P\mathbf{X} \qquad (1)$$

, where **x** is a homogeneous image coordinate point and **X** is a homogeneous metric 3D coordinate point defined as

$$\mathbf{x} = \begin{bmatrix} x & y & 1 \end{bmatrix}^T \quad \mathbf{X} = \begin{bmatrix} X & Y & Z & 1 \end{bmatrix}^T$$

where $x$ and $y$ are the horizontal and vertical image coordinates and $X$, $Y$ and $Z$ are the 3D coordinates [1]. When the camera matrices have been defined for both of the cameras, metric 3D coordinates can be calculated from image point-image point correspondences between the images of the cameras by solving the equation pair produced from (1).

Internally, *ManiCalibrator* executes a sequence where it first reads a manipulator trajectory from file. In the next phase the component moves a manipulator through the trajectory by calling a *SAManipulator*. In each of the trajectory's point, *ManiCalibrator* stores image data from the cameras and the manipulator's actual pose, which is obtained from the micropositioners' position sensors. After completing the trajectory, *ManiCalibrator* calculates the camera matrix pair from the cameras' image coordinates to manipulator's coordinate system. The component instances and their connections during the calibration process are illustrated in Fig. 5. *ManiCalibrator* has two data flow (input) ports that subscribe to the image streams of the *GigeAvt* instances topCamera and sideCamera. The images are also visualized through *ImageViewer* components.

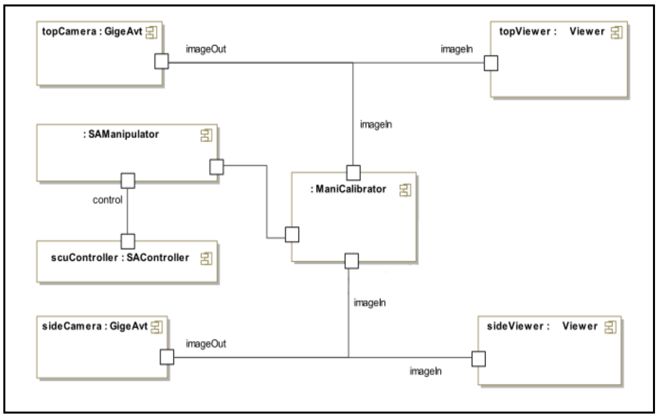


Fig. 5. Connections between component instances during calibration. A ManiCalibrator performs the calibration sequence by moving a SAManipulator through predefined trajectory. GigAvt instances record images in each point of the trajectory

## V. EXPERIMENTS AND RESULTS

Two different experiments were performed to evaluate the performance of the developed FPP and CFL sequences. In the first experiment, the performance of the developed FPP was evaluated by automatically picking individual fibers from the sample area of Rotary Table, moving them into predefined location, and placing the fibers on adhesive substrate. In the first step, Detect operations outputs were tested with a simple rule that defined the boundary conditions for acceptable z-directional coordinates for the manipulator to prevent the manipulator from colliding into the sample area. In case of unacceptable coordinate points, the system cancelled the sequence and informed the operator with an error message. If the coordinate points were found suitable, the system performed automatic sequence of Grasp, Move, and Release operations. The experiment was repeated 10 times out of which 6 were successful. The maximum number of consecutive successful operations was three.

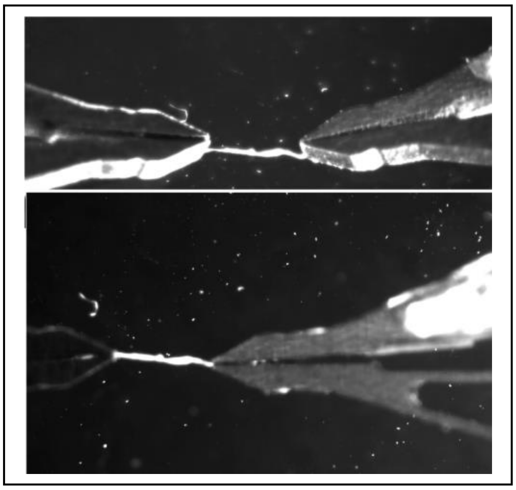

Fig. 6. Successful grasping of a fiber from top-view Camera (upper) and side-view camera (lower). The fiber, illuminated with polarized light is seen as a white line.

The second experiment aimed to test the performance of the CFL. Like in FPP experiment, the fibers were placed one at the time on the sample area and Detect operation was executed. In the next phase, the manipulators were moved to the found end points, and Grasp was performed on both manipulators simultaneously. Finally, the fiber was lifted up with Coordinated Move operation. The result of a successful Grasp operation is presented in Fig. 6. The CFL sequence was performed successfully 12 times out of 15 separate tests. All three unsuccessful tests failed due to the movement of the grasped fiber in Grasp phase. In its current state, the duration of CFL sequence is approximately 30 seconds in total. Where majority of the time is spent for calculation of suitable grasping points. The time is significantly shorter than what is required in teleoperated CFL sequence [26], where the sequence was estimated to take 67 seconds.

## VI. Conclusions and Future Work

In this paper we have demonstrated automated unit operations that are essential part of the effort to develop fully automated microrobotic platform for characterization of natural fibrous materials. Automated fiber pick and place, and coordinated fiber lifting procedures were successfully demonstrated. The coordinated lifting sequence had success rate of 80% and it performed twice as fast as the tele-operated lifting sequence with the same hardware. The presented work utilized mature robotic software framework as a basis of the control software implementation. The approach has proven to increase productivity, as software components have significantly increased the reuse of the developed source code.

## References


[1] Y. Sun, M. A. Greminger and B. J. Nelson, "Investigating protein structure with a microrobotic system," in Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference On, 2004, pp. 2854-2859.

[2] J. Park, S. Jung, Y. Kim, B. Kim, S. Lee, B. Ju and K. Lee, "An integrated bio cell processor for single embryo cell manipulation," in Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference On, 2004, pp. 242-247.

[3] P. Kallio and J. Kuncova-Kallio, "Capillary pressure microinjection of living adherent cells: challenges in automation," Journal of Micromechatronics, vol. 3, pp. 189-220, 2006.

[4] A. Georgiev, P. K. Allen and W. Edstrom, "Visually-guided protein crystal manipulation using micromachined silicon tools," in Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference On, 2004, pp. 236-241.

[5] F. Arai, T. Sakami, H. Maruyama, A. Ichikawa and T. Fukuda, "Minimally invasive micromanipulation of microbe by laser trapped micro tools," in Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference On, 2002, pp. 1937-1942.

[6] K. Inoue, T. Arai, T. Tanikawa and K. Ohba, "Dexterous micromanipulation supporting cell and tissue engineering," in Micro-NanoMechatronics and Human Science, 2005 IEEE International Symposium On, 2005, pp. 197-202.

[7] K. Carlson, K. N. Andersen, V. Eichhorn, D. H. Petersen, K. Mølhave, I. Bu, K. Teo, W. Milne, S. Fatikow and P. Bøggild, "A carbon nanofibre scanning probe assembled using an electrothermal microgripper," Nanotechnology, vol. 18, pp. 345501, 2007.

[8] V. Eichhorn, K. Carlson, K. N. Andersen, S. Fatikow and P. Boggild, "Nanorobotic manipulation setup for pick-and-place handling and nondestructive characterization of carbon nanotubes," in Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference On, 2007, pp. 291-296.

[9] B. Tamadazte, N. Le Fort-Piat, S. Dembélé and G. Fortier, "Robotic micromanipulation for microassembly: modelling by sequencial function chart and achievement by multiple scale visual servoings," Journal of Micro-Nano Mechatronics, vol. 5, pp. 1-14, 2009.

[10] S. J. Ralis, B. Vikramaditya and B. J. Nelson, "Micropositioning of a weakly calibrated microassembly system using coarse-to-fine visual servoing strategies," Electronics Packaging Manufacturing, IEEE Transactions On, vol. 23, pp. 123-131, 2000.

[11] M. Probst, C. Hürzeler, R. Borer and B. J. Nelson, "A microassembly system for the flexible assembly of hybrid robotic MEMS devices," International Journal of Optomechatronics, vol. 3, pp. 69-90, 2009.

[12] M. Probst, M. Fluckiger, S. Pané, O. Ergeneman, Z. Nagy and B. J. Nelson, "Manufacturing of a hybrid acoustic transmitter using an advanced microassembly system," Industrial Electronics, IEEE Transactions On, vol. 56, pp. 2657-2666, 2009.

[13] P. Saketi, M. von Essen, M. Mikczinski, S. Heinemann, S. Fatikow and P. Kallio, "A flexible microrobotic platform for handling microscale specimens of fibrous materials for microsopic studies," Journal of Microscopy, vol. accepted, 2012.

[14] P. Saketi and P. Kallio, "Measuring bond strengths of individual paper fibers using microrobotics," in Progress in Paper Physics Seminar, Graz, Austria, Verlag Der Technischen Universität Graz, 2011, pp. 199-203.

[15] P. Saketi and P. Kallio, "Microrobotic platform for making, manipulating and breaking individual paper fiber bonds," in Assembly and Manufacturing (ISAM), 2011 IEEE International Symposium On, 2011, pp. 1-6.

[16] M. Essen von, J. Hirvonen, P. Saketi and P. Kallio, "Automated grasping in manipulation of individual paper fibers," in The First International Conference on Manipulation, Manufacturing and Measurement on the Nanoscale, Changchun, China, 2011, .

[17] B. P. Gerkey, R. T. Vaughan, K. Stoy, A. Howard, G. S. Sukhatme and M. J. Mataric, "Most valuable player: A robot device server for distributed control," in Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference On, 2001, pp. 1226-1231 vol.3.

[18] A. Brooks, T. Kaupp, A. Makarenko, S. Williams and A. Orebäck, "Orca: A Component Model and Repository," vol. 30, pp. 231-251, 2007.

[19] P. Soetens and H. Bruyninckx, "Realtime hybrid task-based control for robots and machine tools," in Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference On, 2005, pp. 259-264.

[20] R. Diankov. Automated construction of robotics manipulation programs. Dissertations, Paper 32 [http://repository.cmu.edu/dissertations/32]. 2010.

[21] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler and A. Y. Ng, "ROS: An open-source robot operating system," in ICRA Workshop on Open Source Software, 2009, .

[22] G. Metta, P. Fitzpatrick and P. Natale, "YARP: Yet Another Robot Platform," International Journal of Advanced Robotic Systems, 2006.

[23] N. Ando, T. Suehiro and T. Kotoku, "A Software Platform for Component Based RT-System Development: OpenRTM-Aist," vol. 5325, pp. 87-98, 2008.

[24] B. Song, S. Jung, C. Jang and S. Kim, "An introduction to robot component model for OPRoS (open platform for robotic services)," in Proceedings of International Conference on Simulation, Modeling and Programming for Autonomous Robots (SIMPAR 2008), Venice, Italy, 2008, pp. 592-603.

[25] D. Calisi, A. Censi, L. Iocchi and D. Nardi, "OpenRDK: A modular framework for robotic software development," in Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference On, 2008, pp. 1872-1877.

[26] P. Saketi, M. von Essen, M. Miczinski, S. Heinemann, S. Fatikow, P. Kallio, "A flexible microrobotic platform for handling microscale specimens of fibrous materials for microscopic studies", Journal of Microscopy, vol. 248, pp. 167-171, 2012

[27] C. Jang, B. Song, S. Jung, K.-H. Lee, S. Kim, "Real-time supporting of OPRoS component Platform", in Proceedings of 8th International Conference on Ubiquitous Robots and Ambient Intelligence, 2011, pp. 640-641