Paper

Skill-Based Bimanual Manipulation Planning

Wojciech Szynkiewicz

Institute of Control and Computation Engineering, Warsaw University of Technology, Warsaw, Poland

Abstract—The paper focuses on specification and utilization of manipulation skills to facilitate programming of bimanual manipulation tasks. Manipulation skills are actions to reach predefined goals. They constitute an interface between low-level constraint-based task specification and high level symbolic task planning. The task of the robot can be decomposed into subtasks that can be resolved using manipulation skills. Rubik's cube solving problem is presented as an example of a 3D manipulation task using two-arm robot system with diverse sensors such as vision, force/torque, tactile sensors.

Keywords—bimanual manipulation planning, manipulation skills, robot programming.

1. Introduction

Robots employed in human-centered environments have to be equipped with manipulative, perceptive and communicative capabilities necessary for real-time interaction with the environment and humans. Up to now, robot systems have been only able to deal with the high complexity and the wide variability of everyday surroundings to a limited extent. In this paper we are focused on planning dual-arm/hand manipulation tasks for service robots working in such environments.

In our everyday lives we perform many operations in which our two hands cooperate to manipulate diverse objects. The goal of our research is to understand the nature of two-handed cooperative manipulation and enhancing the manipulative skills of the dual-arm robot. Two cooperative hands, if properly utilized, are capable of grasping and manipulating a much larger class of objects, including long, flexible, irregularly shaped, or complex objects (e.g., with internal degrees of freedom).

Object manipulation tasks typically involve a series of action stages in which objects are recognized, grasped, moved, brought into contact with other objects and released. These stages are usually bound by mechanical events that are subgoals of the task. These events involve the making and breaking of contact between either the hands and the grasped object or the object held in hand and another object or surface. Importantly, these contact events usually produce discrete and distinct sensory events. To simplify a solution of the overall problem, we usually tend to divide the task into a sequence of clearly separated subtasks, each of which accomplishes a specific subgoal. In this case, a task planning focuses on deciding what operations will be needed to execute a particular manipulation task, and in what order the operations should be performed. The operations are considered at an abstract level, i.e., sensory operations, gross and fine motion operations, grasping and releasing operations. In terms of representation, it denotes the smallest entity which is used for describing an action.

Many studies have been devoted to single-handed manipulation, for overview see [1]. Recently, a bimanual manipulation has also attracted more attention, especially in unstructured environments (see for example [2]–[7]). Also, the literature pertaining to the analysis of bimanual operations performed by humans is quite extensive, especially in the field of a human-computer interaction, e.g., [8]. Although many solutions for single-handed manipulation can be easily adopted for bimanual manipulation, the whole potential of two cooperative hands cannot be fully utilized without a deeper understanding of their unique characteristics. In general, two-handed manipulation can be classified into uncoordinated and coordinated tasks [8]. The latter can be further subdivided into symmetric and asymmetric ones.

In this paper we focus on specification and utilization of manipulation skills to facilitate programming of bimanual manipulation tasks. The proposed method uses a hierarchical approach for the decomposition of manipulation skills. Manipulation skills are compositions of basic robot operations to reach some predefined goals. They can serve as an interface between low-level geometric task specification and high level symbolic task planning. If the skills are well-defined and robust, then manipulation planning is simplified because is performed in the "space" of skills rather than in the high dimensional configuration/operational space. The task of the robot can be approximated by a set of parameterized manipulation skills. The approach presented in this paper focuses on tasks, rather than motions, and uses manual programming rather than learning techniques to determine the set of manipulation skills. It should be noted that the concept of using skills to create complex actions is a well-studied topic covering many areas, thus, only its applicability to robot manipulation, especially two-handed manipulation is discussed. In our approach, each individual skill is represented as a hybrid finite state automaton in which each state runs one basic operation, and each violation of a operation can give rise a transition. Each transition is the outcome of the basic operation. Skills can have a set of parameters, which can be used to adapt each skill to a particular use case. Skills are the components of the intermediate level between symbolic and geometric levels. In this paper we focus on solving the coordinated

bimanual tasks, and we propose the solution of the Rubik's cube manipulation as an example of such a task. To implement dual-arm manipulation utilizing vision and force sensing, the MRROC++ robot programming framework is used [9], [10].

The paper is organized as follows. In Section 2 an overview of most representative manipulation planning approaches is given. Section 3 describes the hierarchical representation of manipulation tasks. In Section 4 Rubik's cube problem solving is discussed as an example of two-handed manipulation.

2. Related Work

Manipulation planning is an extension to the classical robot motion planning problem. The robot is able to interact with its environment by manipulating movable objects, while it has to avoid self-collisions or collisions with obstacles. Traditionally, manipulation planning concerns the automatic generation of the sequence of robot motions allowing to manipulate movable objects among obstacles. Existing research in manipulation planning has focused mainly on the geometric aspects of the task, while greatly simplifying the issues of grasping, stability, friction, and uncertainty [4], [11]. Symbolic planning algorithms have typically assumed perfect models of both the environment and the robot, not only at an abstract level but at every level of control. This is a quite reasonable assumption in well-structured and fully controlled environments. However, in everyday environments this is not often the case, which makes that most of the proposed theoretical solutions are not directly applicable. The real world does not behave as expected, and in fact it does not behave predictably.

Most of the research in manipulation planning deals with the creation of the manipulation graph and extraction of a manipulation path from this graph [12]–[15]. The concept of a manipulation graph was introduced by Alami *et al*. [12] for the case of one robot and several movable objects manipulated with discrete grasps and placements. In this case, the nodes of the graph represent discrete configurations and the edges correspond to robot motions moving the grasped object (*transfer path*), or leaving it at rest to get to another grasp position (*transit path*). A solution to the manipulation planning problem is now given by a manipulation path in this graph. This path is solved using PRM (Probabilistic Roadmap Method) planners [13], [14].

In most of the existing algorithms it is assumed that a finite set of stable placements and of possible grasps of the movable object are given in the definition of the problem (e.g., [12], [13]). Consequently, a part of the manipulation task decomposition is thus done by the user since the initial knowledge provided with these finite sets has to contain the grasps, and the intermediate placements required to solve the problem. In [14] the authors proposed a general manipulation planning approach capable of addressing continuous sets for modeling, both the possible grasps and the stable placements of the movable object. The nodes of the manipulation graph (i.e., the places where the connections between the feasible transit and transfer paths have to be searched) correspond to a set of sub-manifolds of the composite configuration space, as opposed to discrete configurations. Cambon *et al*. [15] proposed a specialized integration of a symbolic task planning and geometric motion, and manipulation planning. They extended classical action planning formalism based on a STRIPS-like description where manipulation planning problems in configuration space are introduced.

One of the most intuitive ways to acquire new task knowledge is to learn it from the human user via demonstration and interaction. This approach to task learning is known as Programming by Demonstration (PbD) [2]. It is one of the most often used programming paradigms of two-arm manipulation for humanoid robots [7]. PbD systems generally try to decompose the observed task execution of the human demonstrator into a sequence of tasks that are performable by the robot. Typically, tasks are recorded from human demonstrations, segmented, interpreted and stored using some data representation. Several programming systems and approaches based on human demonstrations have been proposed during the last years, e.g., [2], [3].

In [16] an architecture which uses primitive skills that combine to form a skill, which in turn form a complete task is presented. Each primitive skill is selected by heuristic selection out of many possible primitive skills, based on the sensor signals. A neural network is used to detect the change between the skills. Each primitive skill is executed by a separate controller.

3. Manipulation Planning

Manipulation planning is a very challenging problem in robotics research as it consists of a number of subproblems that themselves are still open issues and subject to ongoing research. Typical manipulation tasks accomplish relative motions and/or dynamic interactions between various objects. Typically, manipulation planning involves motion and grasp planning. The most effective robot motion planning strategies today are built upon sampling-based techniques, including the PRM [13] and Rapidly-exploring Randomized Trees [17], and their variants. Robot motion planning can also be viewed as an optimization problem that aims to minimize a given objective function [5], [11]. To solve such a problem in efficient way appropriate tools have to be used, e.g., [18]. Grasp planning is also an area of intensive research [19].

Several basic components of manipulation task can be distinguished (Fig. 1).

Using intended subgoals as a criterion, three different classes of manipulation tasks can be distinguished [20].

1. *Transport operations*: the simplest class of robot manipulation. This kind of task can be easily distinguishably by the change of the external state (pose) of the manipulated object. Various types of transport

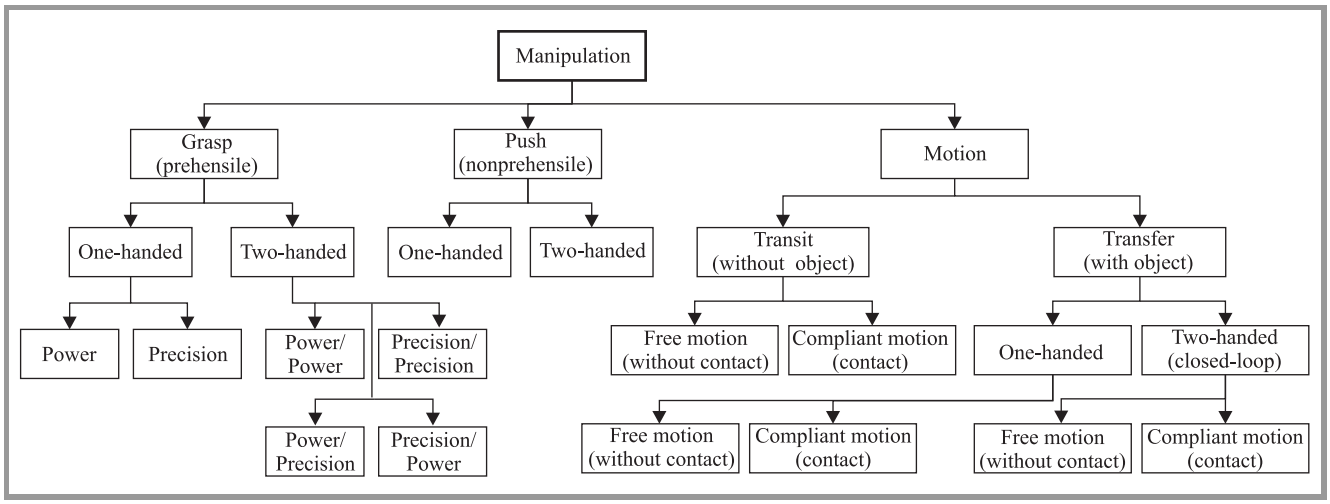

Fig. 1. Components of the manipulation task.

tasks such as pick-and-place or fetch-and-carry are a component part of almost all manipulation operations. Accordingly, the trajectory of the manipulated object has to be considered and modeled in transport actions models.

2. *Object handling*: a more specialized class of manipulation tasks deals with changing the internal state of objects without influencing other objects (like opening doors, pushing a button, manipulating the Rubik's cube, etc.). This class of tasks consists of every task changing an internal state of an object without manipulating another object. In the object-handling tasks, transition actions changing an internal state have to be modeled. Moreover, the object models need to incorporate an adequate description of their internal state.

3. *Tool handling*: the most typical characteristic for this type of actions is the interaction between two objects, usually a tool and a workpiece. Interaction is related to the functional role of objects used or the correlation between the functional roles of all objects included in the manipulation, respectively. The object model thus should contain a model of the possible interaction modalities or functional roles the object can take. According to different modalities of interaction, considering contacts, movements, etc., a diversity of handling methods has to be modeled.

We consider a manipulation task to be an activity involving the composition and coordination of an existing set of manipulative skills in order to accomplish a given set of goals. Two representations of robot manipulation skills/tasks can be distinguished symbolic and non-symbolic.

3.1. Manipulation Skills

We make a crucial distinction between tasks, manipulation skills and basic skills or primitive actions in this work. Task is a function to be performed. Manipulation Skill (MS) is a pattern of activity which describes an ability that achieves or maintains a particular goal. A manipulation skill can be defined as an abstraction of a set of basic skills that follow the same control strategy. Basic Skill (BS) is an action that abstract a sensory-motor coupling such as skill motion types (e.g., motion trajectory generators), concrete grasping and releasing strategies, direct and inverse kinematics for the specific robots, etc. The set of the BS serves as an application programming interface. The task of the robot can be represented by a set of parameterized manipulation skills. Therefore the overall planning and control system has a layered hierarchical structure as shown in Fig. 2. It should be noted that hierarchy can exist at all layers. Task is the highest level of abstraction, representing a semantically meaningful task such a solving scrambled Rubik's cube. The task consists of a sequence of MS's, which represent subtasks, such as turning a single face of the Rubik's cube. Skills consist of basic skills or primitive actions which are the lowest level of control in the proposed architecture. Each BS is implemented using a single low-level controller, Control Program (CP) which is responsible for the control of robot hardware.

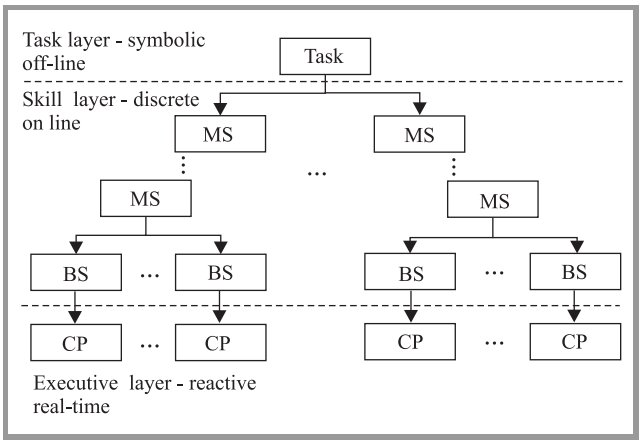

Fig. 2. Planning and control three-layered structure.

To model MS we use a hybrid automaton. A hybrid automaton is a dynamical system that describes the evolution in time of the valuations of a set of discrete and continuous variables. A hybrid automaton H [21], [22] is a collection $H = (Q, X, f, Init, D, E, G, R)$, where

- Q is a set of discrete states;
- $X = \mathbb{R}^n$ is a finite set of continuous states;
- $f(\cdot,\cdot) : Q \times X \to \mathbb{R}^n$ is a vector field;
- $Init \subseteq Q \times X$ is a set of initial states;
- $Dom(\cdot) : Q \to 2^X$ is a domain;
- $E \subseteq Q \times Q$ is a set of edges (events);
- $G(\cdot) : E \to 2^X$ is a guard condition;
- $R(\cdot,\cdot) : E \times X \to 2^X$ is a reset map (relation).

Each state of the automaton has its own low-level controllers and transitions to other states. The proper selection of the set of manipulation skills is a critical step in our approach. The following manipulation skills have been defined to solve the Rubik's cube problem: *Localize*, *Reach*, *Turn*, *Grasp*, *Release*.

Localize – robotic manipulation of an object requires that this object must be detected and located first. If vision is used as the robot's primary source of information about the environment, the object of interest must be identified in the image and subsequently localized in 3D space. Generally, in cluttered environments, detecting a certain object is not an easy problem. Recognition and localization of a known object in the image is based on matching its certain previously defined features such as: shapes, sizes, colors, texture, etc. The choice of features and the matching algorithm is arbitrary and it depends primarily on the specification of the object and it will not be discussed here. This task becomes much more difficult if we want not only to localize the object in the scene (2D localization), but also to find its 3D pose (6D localization) in relation to the camera frame or to the world frame. Typical method used for 6D object localization is to calculate the pose based on the correspondence between 3D model and image coordinates from camera image. Most of the works on grasping and manipulation planning have assumed the existence of a database with 3D models of objects encountered in the robot surroundings and a 3D model of the robot itself [2], [9].

Reach – for reaching an object the *Reach* skill uses motion planning to compute a collision-free trajectory for moving the robot arm from its current pose to one that allows grasping of a specified object with a hand. If both arms are free, then *Reach* can employ each of the arm to move to the vicinity of the object. If one arm is currently grasping an object, *Reach* can be used for the other arm to prepare for grasping. *Reach* skill requires closed-loop execution to permit interaction with the environment. We utilize a position based, end-effector open loop visual servo with stand-alone camera to perform reaching operation [23].

Grasp – this skill is used for grasping objects for manipulation. Grasps are a special subset of manipulation skills that aimed at constraining the mobility of the object. *Grasp* should allow to perform different types of grasps depending on the hand structure. The parameters of the *Grasp* skill are: *grasp type*, *grasp starting point*, *approaching direction*, *hand orientation*. Grasp type determines the grasp execution control, namely, the hand preshape posture, the way the hand approaches the objects, the hand control strategy. For approaching the object, the hand is positioned at point in the vicinity of the object. The approaching line is determined by the grasp starting point and the approaching direction.

Turn – this skill is equivalent to *Reach* with an object or tool grasped as the end-effector, rather than the hand. Given an object grasped by the robot one hand or two hands, *Transfer* skill utilize motion planning to move the robot to configuration such that the object is at target pose. *Transfer*, like *Reach* requires closed-loop execution.

Release – this skill performs an action opposite to the *Grasp*, it simply release the object in the current configuration.

4. An Example of Two-Handed Manipulation

As an example of the task for two-handed manipulation we chose the manipulation of Rubik's cube puzzle. We used a Rubik's cube as an object to be identified, localized and manipulated. Rubik's cube combinatorial puzzle was invented by Ernő Rubik of Hungary in 1974. The standard $3 \times 3 \times 3$ version of the Rubik's cube consists of 27 sub-cubes, or cubies, with different colored stickers on each of the exposed sides of the sub-cubes. In its goal state each of the six faces has its own color. The total state space for solving a scrambled Rubik's cube is sized at $(3^{8-1} \cdot 8!) \cdot (2^{12-1} \cdot 12!)/2 = 43,252,003,274,489,856,000 \approx 4.3 \times 10^{19}$. Obviously, this number of states is prohibitively large for any sort of a brute force search technique, which is why specialized algorithms are needed to solve the Rubik's cube puzzle. However, the presentation of the algorithms for recognizing and solving Rubik's cube are not discussed in this paper, some information about these algorithms can be found in [9].

In this particular case we are interested in a coordinated manipulation in which both hands are manipulating the same object, thus creating a closed kinematic chain [5]. This task was chosen as it closely resembles the tasks that service robots have to execute. The process of manipulation of the Rubik's cube involves all aspects of visual serving to the vicinity of the cube, alignment of robot arms with the cube, grasping it with the grippers, and finally rotating the adequate face of the cube. The last three actions are repeated as many times as the number of moves is required to solve the scrambled cube. Here, we assume that from the

high-level task planning system, i.e., Rubik's cube solver, a sequence of the turns of the faces is obtained. The goal is to plan a proper sequence of hand movements and grasping actions for both arms.

4.1. Problem Formulation

The task of solving Rubik's cube needs several sensor-based operations such as:

- recognizing the cube in the image and localizing it in the robot workspace – *Localize* skill,
- approaching the cube while avoiding collisions by using visual information – *Reach*,
- grasping the cube using force/torque measurements for stiffness control and eye-in-hand and tactile sensors mounted in the jaws – *Grasp*,
- re-grasping the cube in order to identify the cube's initial state – *Grasp* → *Release* sequence,
- turning the faces of the cube while avoiding jamming using information from force/torque sensor for implementing interaction control – *Grasp* → *Turn* → *Release* sequence performed n times, where n is the number of moves required to solve the cube.

To support the programmer with task specification, object frames and feature frames are introduced, as well as suitable local coordinates to express the relative pose between these frames. Figure 3 presents the geometrical structure of the two-arm manipulation system, and coordinate frames F_i attached to the appropriate components of the system together with the distribution of sensors.

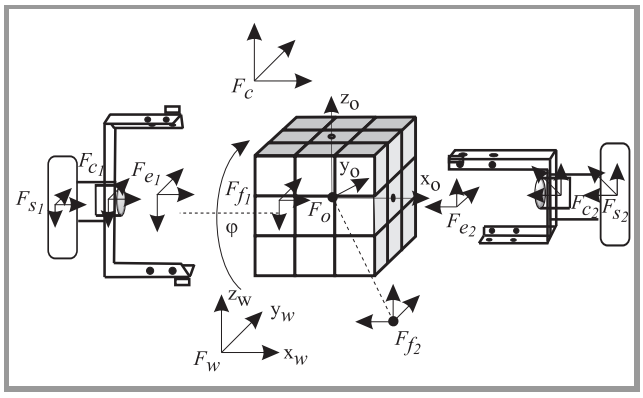

Fig. 3. Coordinate frames attached to the two-handed robotic system.

Coordinate frames $F_{e_i}, i = 1,2$ are attached to the grippers and sensor frames F_{s_i} and F_{c_i} attached to the force sensors and to the eye-in-hand cameras, respectively. Given the view of the scene, the robot should be able to recognize the cube and localize it in the robot workspace. As a result of the visual localization, the position and orientation of the object frame F_o attached to the cube with respect to (w.r.t) world frame F_w is computed (as described earlier).

To describe the task for each hand two feature frames $F_{f_i}, i = 1,2$ are introduced, as shown in Fig. 3. These frames are used to plan manipulation skills such as approach trajectories for both hands, grasp and release operations, and hand movements to turn the cube faces.

The 4×4 matrix ${}^i_jT$ is a homogenous transformation matrix ${}^i_jT \in \mathbb{SE}(3)$ (where $\mathbb{SE}(3)$ is a special Euclidean group of a rigid body motions in $\mathbb{R}^3$ [24]) is a linear operator used in a mapping between the appropriate coordinate frames. Matrix ${}^i_jT$ may be interpreted as the representation of the pose of the frame F_j w.r.t. frame F_i. Left-hand superscript is omitted (i.e., ${}_jT$) when the reference frame is evident from the context, e.g., it is the world frame F_w.

Pre-computed sequence of turns of the faces can be described in the F_o coordinate frame as a sequence of rotations about unit vectors $\hat{\boldsymbol{x}}_o, \hat{\boldsymbol{y}}_o, \hat{\boldsymbol{z}}_o$ of its axes:

$$ {}^oR(\boldsymbol{u}, \varphi) = \mathrm{Rot}(\boldsymbol{u}, \varphi), \tag{1}$$

where $\boldsymbol{u} = \hat{\boldsymbol{x}}_o, \hat{\boldsymbol{y}}_o$ or $\hat{\boldsymbol{z}}_o$, and $\varphi = -\frac{\pi}{2}, -\pi, \frac{\pi}{2}$ or π.

The desired grasp configurations w.r.t. coordinate frame F_o are described by the following matrices:

- ${}^o_{f_1}T$ – to grasp a single slice,
- ${}^o_{f_2}T$ – to grasp two slices simultaneously.

Locations of the possible contact regions on the cube are imposed by the specific shape of the gripper jaws. The shape of each of the jaws of the gripper matches the form of the corner of the cube. The cube is being grasped diagonally in such a way that either one or two layers are immobilized, where the corner pieces of one layer define the diagonal.

Now, we have to plan such a sequence of admissible grasps $({}^o_{f_1}T, {}^o_{f_2}T)$ that enable each single turn of the face without re-grasping:

$$ {}_{f_1}T = {}_oT\,{}^o_{f_1}T; \quad {}_{f_2}T = {}_oT\,{}^o_{f_2}T \tag{2}$$

The conditions of grasp feasibility are as follows:

$$ {}_{f_1}T = {}_{b_1}T\,{}^{b_1}_{e_1}T(\boldsymbol{q}_1)\,{}^{e_1}_{o}T\,{}^o_{f_1}T; \quad {}_{f_2}T = {}_{b_2}T\,{}^{b_2}_{e_2}T(\boldsymbol{q}_2)\,{}^{e_2}_{o}T\,{}^o_{f_2}T, \tag{3}$$

or equivalently

$$ {}_{f_1}T = {}_{b_2}T\,{}^{b_2}_{e_2}T(\boldsymbol{q}_2)\,{}^{e_2}_{o}T\,{}^o_{f_1}T; \quad {}_{f_2}T = {}_{b_1}T\,{}^{b_1}_{e_1}T(\boldsymbol{q}_1)\,{}^{e_1}_{o}T\,{}^o_{f_2}T, \tag{4}$$

where ${}_{b_i}T, i = 1,2$ is the homogenous transformation matrix from the world frame F_w to the robot base coordinate frame F_{b_i}. Matrix ${}^{b_i}_{e_i}T(\boldsymbol{q}_i), i = 1,2$ represents direct kinematics of the robot arm i, and $\boldsymbol{q}_i$ is the vector of joint coordinates of the arm i.

In this case grasp stability conditions are of a geometric nature and grasp synthesis is reduced to the choice of four contact regions on the cube (two for each gripper) from the given set of contacts and computing desired poses of both grippers, i.e., ${}_{e_1}T$ and ${}_{e_2}T$ which guarantee firm grasps. In fact, grasp synthesis comes down to the proper positioning of the grippers. Therefore grasp configurations can be described in the operational space as well as in the joint space.

When both grippers firmly hold the cube the closed kinematic chain is established. Now the motion planning problem is complicated by the need to maintain the closed loop structure, described by the loop closure constraint.

$$ {}^{b_1}_{e_1}T(\boldsymbol{q}_1)\,{}^{e_1}_{o}T(\varphi) - {}^{b_1}_{b_2}T\,{}^{b_2}_{e_2}T(\boldsymbol{q}_2)\,{}^{e_2}_{o}T = \mathbf{0} \qquad (5) $$

However, in our case, the motion of the closed chain linkage can be described in the F_o coordinate frame as a single rotation about its axes (i.e., the elementary turn of the cube's face). For the frame F_o chosen as it is shown in Fig. 3 these moves are rotations around its axes described in (1).

These moves can be easily transformed to the motions of the grippers. However, due to kinematic calibration errors, the two robot arms cannot be position controlled while executing the turns. This would cause excessive build-up of force in a rigid closed kinematic chain due to small misalignments. Therefore at this stage the motions have to be executed in position-force control mode.

4.2. Implementation of the Two-Handed Manipulation in the MRROC++ Framework

The control system of the two-handed system equipped with special end-effectors, each composed of an electric gripper and diverse sensors, was implemented by using the MRROC++ [1] robot programming framework.

MRROC++ is a robot programming framework, thus it provides a library of software modules (i.e., classes, objects, processes, threads and procedures) and design patterns according to which any multi-robot system controller can be constructed. This set of ready made modules can be extended by the user by coding extra modules in C++ [9], [23], [25]. MRROC++ based controllers have a hierarchic structure composed of processes (Fig. 4) (some of them consisting of threads) supervised by the QNX Neutrino real time operating system. The underlying software is written in C++.

From the point of view of the executed task MP is the coordinator of all effectors present in the system. It is responsible for trajectory generation in multi-effector systems where the effectors cooperate tightly – as is the case in the presented system. The manipulation planning system contained in the MP transforms the solution obtained from Rubik's cube solver into a proper sequence of manipulation skills. In the MRROC++ framework these skills are implemented as motion generators, which are used by the Move instructions. Therefore the MP is responsible both for producing the plan of the motions of the faces of the cube and subsequently the trajectory generation for both manipulators. This trajectory can be treated as a crude reference trajectory for both arms. At a later stage this trajectory is modified by taking into account the force readings.

[1]The name is derived from the fact that this programming framework is the basis for the design of Multi-Robot Research-Oriented Controllers and that the underlaying software is coded in C++.

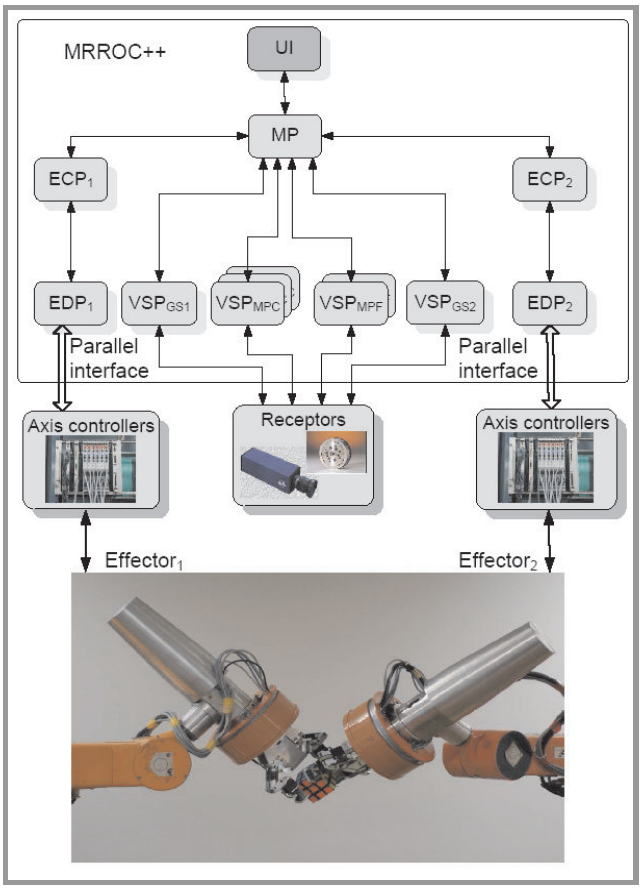

Fig. 4. MRROC++ based controller for the two-arm system.

Each effector has two processes controlling it: Effector Control Process ECP and Effector Driver Process EDP. The first one is responsible for the execution of the user's task dedicated to this effector (in our case the task is defined by the MP – it is defined by the reference trajectory that is to be executed by the manipulator), and the other one for direct control of this effector. The EDP is responsible for direct and inverse kinematics computations, as well as for both position and force servo-control.

4.3. Experiments

The overall experimental setup consists of two 6 degree of freedom (dof) modified IRp-6 robot arms, each with a parallel jaw gripper Fig. 5.

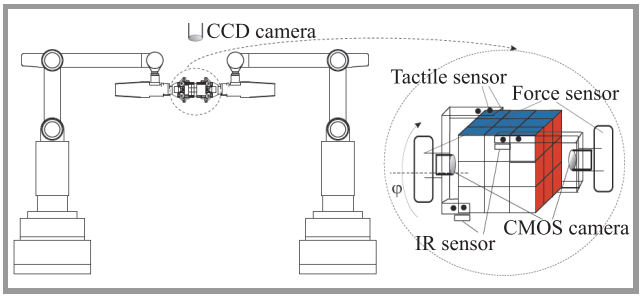

Fig. 5. Sensors used to locate and manipulate the Rubik's cube.

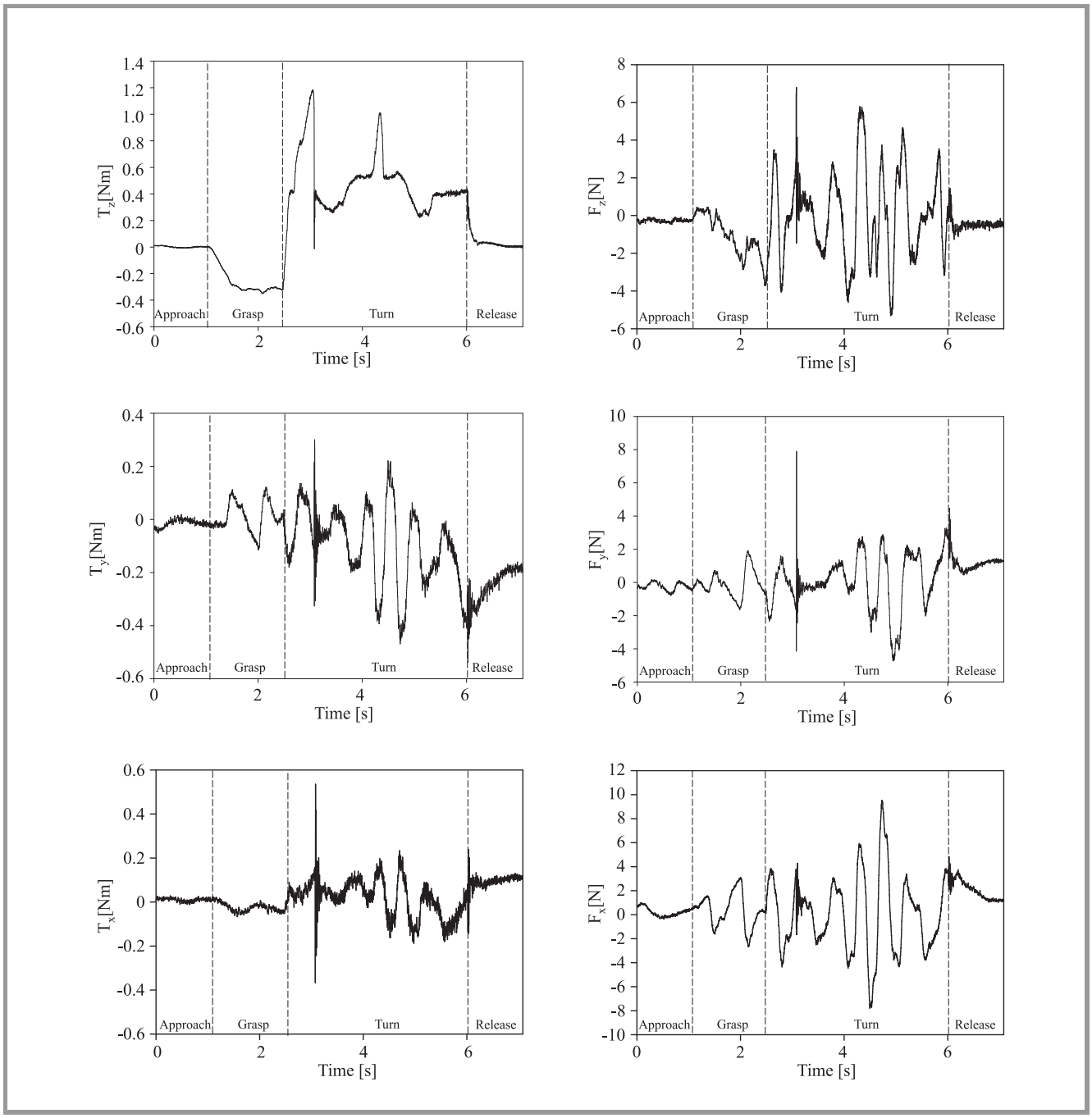

Fig. 6. Measured force and torque components while manipulating the Rubik's cube.

Each jaw was instrumented with tactile sensors which detect the presence of contacts with the grasped object. Moreover, each hand was equipped with a wrist-mounted six-axis force-torque sensor, and an eye-in-hand miniature CCD color camera [9]. Additionally, a global vision system with fixed-mount color camera and Digital Video Processor for fast image acquisition and realtime processing of the incoming data was used.

During the task execution either pure position control or position–force control is used, depending on the current task execution stage. Typically, these execution stages are position controlled in which there in no simultaneous contact between the two end-effectors and the cube, or between one of the end-effectors and the cube held by the operator. The stages, where such contact is present or expected to occur, are position–force controlled.

Cube grasping starts with one of the manipulators initiating the closing of the gripper jaws to catch the cube already held by the other manipulator or the operator. The manipulator currently holding the cube is commanded to keep the current position, hence it is position controlled. Figure 6 presents the force and torque plots for three stages of manipulation for the second manipulator, which is currently force-controlled.

Force/torque (F/T) sensors provide information about the magnitude and direction of the forces and torques that appear when the robot arms and the object are in contact. Free motion can be observed in the first phase (reaching the object), this stage occurs when one of the manipulators is currently holding the cube and the second one is approaching to gain a direct contact with the other side of the cube. Then, after the contact, grasping phase begins. The visible oscillations occur due to arms and Rubik's cube compliance. Once the cube is grasped firmly the torque stabilizes (Fig. 6). The rapid change in torque appears when the rotation of the cube face is initiated (turn phase), because initially the rotated face was jammed – this can be seen from the plot. In the release phase the gripper is opened, and the closed kinematic chain is disjoined.

5. Conclusion

In this paper, a framework for the description of two-arm/hand manipulation task based on the definition of a priori specified manipulation skills was proposed. The whole task was decomposed into a set of subtask each of which is resolved by a set of manipulation skills. To manage the task or environment variations the skills were parameterized. The parameters are generally related to the task variations, such as: type of a motion, grasping rule, an initial and final points, etc. Rubik's cube solving problem was used as a 3D manipulation task using two-arm robot system with diverse sensors such as vision, force/torque, tactile sensors. The manipulation task was implemented in the MRROC++ framework.

References

[1] A. M. Okamura, N. Smaby, and M. R. Cutkosky, "An overview of dexterous manipulation" in *Proc. IEEE Int. Conf. Robot. Autom. ICRA 2000*, San Francisco, USA, 2000, pp. 255–262.

[2] T. Asfour, P. Azad, N. Vahrenkamp, K. Regenstein, A. Bierbaum, K. Welke, J. Schröder, and R. Dillmann, "Toward humanoid manipulation in human-centred environments", *Robot. Autonom. Sys.*, vol. 56, no. 1, pp. 54–65, 2008.

[3] A. G. Billard, S. Calinon, and F. Guenter, "Discriminative and adaptive imitation in uni-manual and bi-manual tasks", *Robot. Autonom. Sys.*, vol. 54, no. 5, pp. 370–384, 2006.

[4] Ch. Smith, Y. Karayiannidis, L. Nalpantidis, X. Gratal, P. Qi, D. V. Dimarogonas, and D. Kragic, "Dual arm manipulation – a survey", *Robot. Autonom. Sys.*, vol. 60, no. 10, pp. 1340–1353, 2012.

[5] W. Szynkiewicz and J. Blaszczyk, "Optimization-based approach to path planning for closed-chain robot systems" *Int. J. Appl. Mathem. Comp. Sci.*, vol. 21, no. 4, pp. 659–670, 2011.

[6] N. Vahrenkamp, D. Berenson, T. Asfour, J. Kuffner, and R. Dillmann, "Humanoid motion planning for dual-arm manipulation and re-grasping tasks", in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Sys. IROS 2009*, St. Louis, MO, USA, 2009, pp. 2464–2470.

[7] R. Zöllner, T. Asfour, and R. Dillmann, "Programing by demonstration: Dual-arm manipulation tasks for humanoid robots", in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Sys. IROS 2004*, Sendai, Japan, 2004, vol. 1, pp. 479–484.

[8] Y. Guiard, "Asymmetric division of labor in skilled bimanual action: the kinematic chain as a model" *J. Motor Behav.*, vol. 19, no. 4, pp. 486–517, 1987.

[9] C. Zieliński, W. Szynkiewicz, T. Winiarski, M. Staniak, W. Czajewski, and T. Kornuta, "Rubik's cube as a benchmark validating MRROC++ as an implementation tool for service robot control systems", *Industr. Robot: An Int. J.*, vol. 34, no. 5, pp. 368–375, 2007.

[10] C. Zieliński and T. Winiarski, "Motion generation in the MRROC++ robot programming framework", *The Int. J. Robot. Res.*, vol. 29, no. 4, pp. 386–413, 2010.

[11] S. M. LaValle, *Planning Algorithms*. Cambridge, U.K.: Cambridge University Press, 2006.

[12] R. Alami, T. Siméon, and J. P. Laumond, "A geometrical approach to planning manipulation tasks. The case of discrete placements and grasps", in *Proc. Fifth Int. Symp. Robot. Res.*, Cambridge, MA, USA, MIT Press, pp. 453ℓ-463, 1990.

[13] Ch. Nielsen and L. Kavraki, "A two-level fuzzy prm for manipulation planning", in *Proc. EEE/RSJ Int. Conf. Intell. Robots Sys. IROS*, Takamatsu, Japan, 2000, pp. 1716–1721.

[14] T. Siméon, J.-P. Laumond, J. Cortés, and A. Sahbani, "Manipulation planning with probabilistic roadmaps", *The Int. J. Robot. Res.*, vol. 23, no. 7–8, pp. 729–746, 2004.

[15] S. Cambon, R. Alami, and F. Gravot, "A hybrid approach to intricate motion, manipulation and task planning", *Int. J. Robot. Res.*, vol. 28, pp. 104–126, 2009.

[16] G. Milighetti, H. B. Kuntze, C. W. Frey, B. Diestel-Feddersen, and J. Balzer, "On a primitive skill-based supervisory robot control architecture", in *Proc. IEEE Int. Conf. Robot. Autom. ICRA 2005*, Barcelona, Spain, 2005, pp. 141–147.

[17] J. J. Kuffner and S. M. LaValle, "RRT-connect: An efficient approach to single-query path planning", in *Proc. IEEE Int. Conference Robot. Autom. ICRA 2000*, San Francisco, USA, 2000, pp. 995–1001.

[18] E. Niewiadomska-Szynkiewicz and M. Marks, "Software Environment for Parallel Optimization of Complex Systems", in *Applied Parallel Scientific Computing, Lecture Notes in Computer Science LNCS 7133*, K. Jonasson, Ed. Springer, 2012, pp. 86–96.

[19] A. Bicchi and V. Kumar, "Robotic grasping and contact: a review", in *Proc. IEEE Int. Conf. Robot. Autom. ICRA 2000*, San Francisco, CA, USA, 2000, pp. 348–352.

[20] M. Pardowitz, S. Knoop, R. Dillmann, and R. D. Zoellner, "Incremental learning of tasks from user demonstrations, past experiences, and vocal comments", *IEEE Trans. Sys. Man, Cybernet., Part B*, vol. 37, no. 2, pp. 418–432, 2007.

[21] T. A. Henzinger, "The theory of hybrid automata", in *Proc. Eleventh Ann. IEEE Sympo. Logic Comp. Sci. LICS'96*, New Brunswick, USA, 1996, pp. 278–292.

[22] M. S. Branicky, V. S. Borkar, and S. K. Mitter, "A unified framework for hybrid control: Model and optimal control theory", *IEEE Trans. Autom. Contr.*, vol. 43, pp. 31–45, 1998.

[23] W. Szynkiewicz, C. Zieliński, W. Czajewski, and T. Winiarski, "Control Architecture for Sensor-Based Two-Handed Manipulation", in *CISM Courses and Lectures – 16th CISM-IFToMM Symposium on Robot Design, Dynamics and Control, RoManSy'06*, T. Zielińska and C. Zieliński, Eds., no. 487, pp. 237–244. Wien, New York: Springer, 2006.

[24] R. M. Murray, Z. Li, and S. S. Sastry, *A Mathematical Introduction to Robotic Manipulation*. CRC Press, 1994.

[25] C. Zieliński, W. Szynkiewicz, and T. Winiarski, "Applications of MRROC++ robot programming framework", in *Proc. Fifth Int. Worksh. Robot Motion Control RoMoCo'05*, Dymaczewo, Poland, 2005, pp. 251–257.

Wojciech Szynkiewicz received Ph.D. degree in Robotics in 1996 from the Warsaw University of Technology (WUT). He is an Assistant Professor employed in the Institute of Control and Computation Engineering of WUT. From 1999 to 2003, he was the Deputy Director and Secretary to the Scientific Council of the Research Center for Automation and Information-Decision Technology-CATID. His research activities concentrate on multi-robot/multi-agent systems, motion planning, autonomous mobile robots, robot controller structures, and real-time and distributed systems. He works on sensor-based motion planning and control algorithms for multi-robot systems, including service, personal and mobile robots.

E-mail: W.Szynkiewicz@elka.pw.edu.pl
Institute of Control and Computation Engineering
Warsaw University of Technology
Nowowiejska st 15/19
00–665 Warsaw, Poland