Received 13 April 2015 Revised 9 July 2015 Accepted 26 September 2015

A random forest application to contact-state classification for robot programming by human demonstration

S. Cabras[a,b*†], M. E. Castellanos[c] and E. Staffetti[c]

This paper addresses the non-parametric estimation of the stochastic process related to the classification problem that arises in robot programming by demonstration of compliant motion tasks. Robot programming by demonstration is a robot programming paradigm in which a human operator demonstrates the task to be performed by the robot. In such demonstration, several observable variables, such as velocities and forces can be modeled, non-parametrically, in order to classify the current state of a contact between an object manipulated by the robot and the environment in which it operates. Essential actions in compliant motion tasks are the contacts that take place, and therefore, it is important to understand the sequence of contact states made during a demonstration, called contact classification. We propose a contact classification algorithm based on the random forest algorithm. The main advantage of this approach is that it does not depend on the geometric model of the objects involved in the demonstration. Moreover, it does not rely on the kinestatic model of the contact interactions. The comparison with state-of-the-art contact classifiers shows that random forest classifier is more accurate.

Keywords: multi-class contact classification; sensor force analysis; supervised learning

1. Introduction

Robot programming by demonstration consists in a set of techniques to generate robot task plans from user examples, and this skill transfer process from humans to robots relies on a real or virtual environment to acquire the crucial information about a task from a demonstration of the task, on a methodology to transform the acquired information into a robot-independent representation, and finally, on a strategy to convert this knowledge learned through observations into a sequence of robot actions that accomplish the same task [1, Ch. 59].

The contribution of this paper is related to one of these aspects, namely, on the methodology to transform the acquired information in robot programming by demonstration of compliant motion tasks into a robot-independent representation. This is done by means of a machine learning algorithm, which is a classifier of sensor signals into several categories of contacts between an object manipulated by the robot programmer and the environment in which the task is performed. We gather the pose of the manipulated object and the interaction wrenches between the object and the environment. With this sensor data, we are able to perform an accurate recognition through classification of the contacts that occurred during the demonstration. With such classification, it is then possible to program the robot in order to repeat the task.

1.1. Compliant motion tasks

Compliant motion tasks are manipulation tasks that involve contacts between the object manipulated by a robot and the environment in which it operates [2, Ch. 7]. In general, interaction wrenches [3] are crucial for the correct execution of the task, and therefore, they have to be controlled along it. Even if the interaction wrenches are not relevant for a particular compliant motion task, these operations must be carried out using passive or active force control to cope with uncertainties,

[a] *Universidad Carlos III de Madrid, Getafe, 28903, Spain*
[b] *Università di Cagliari, Cagliari, 09128, Italy*
[c] *Universidad Rey Juan Carlos, Móstoles, 28933, Spain*
**Correspondence to: S. Cabras, Department of Statistics, Universidad Carlos III de Madrid, C./Madrid 126, 28903, Getafe, Spain.*
[†] *E-mail: stefano.cabras@uc3m.es, s.cabras@unica.it*

because small errors in the object pose can give rise to high interaction forces with possible damages. Moreover, interaction wrenches are useful to control the task of the robot in the hybrid control paradigm discussed in [4, 5].

1.2. Task frame formalism

The task frame formalism [6] is an intuitive and manipulator-independent formalism to specify force controlled in the robotic tasks. In this paradigm, the programmer not only has to specify the task but also to foresee the input sensor signals to control it. For more complex tasks involving multiple contacts, this can be extremely difficult. Therefore, a constraint-based task specification framework has been presented in [7].

Specification and control of compliant motion tasks involving rigid objects relies on calculating velocity and force controlled subspaces, which depend on the kinestatic model of the contact [3]. Therefore, contact modeling is crucial in robot programming by demonstration of compliant motion tasks.

In particular, any contact state between two polyhedra can be described as a combination of the so-called basic contacts (BCs) between object features, namely, vertices, edges, and faces denoted with symbols v, e, and f, respectively. For instance, basic contacts include vertex-face (v-f) and edge-edge (e-e) contacts [8]. A more concise description of contact states is based on the notion of principal contacts (PCs), which was introduced to describe a contact primitive between two surface elements of two polyhedral objects in contact, where a surface element can be a face, an edge, or a vertex. Formally, a PC describes a contact between a pair of surface elements, which are not boundary elements of other contacting surface elements. In this context, the boundary elements of a face are the edges and vertices bounding it, and the boundary elements of an edge are the vertices bounding it. The contacting area can be a single point, as in v-f or e-e contacts, a line, as in an edge-face (e-f) contact, or a polygon, as in a face-face (f-f) contact. Thus, a general contact state between two polyhedral objects can be characterized by a set of PCs, called contact formations (CFs) [9], which are those that we are able to recognize along with the absence of contact denoted by nc. Thus, the relation between BCs, PCs, and CFs is that CFs are defined using PCs, which are defined in terms of BCs. For example, in the CF, represented in Figure 3(f), a face of the cube is in contact with a face of the environment, and an edge of the cube is in contact with another face of the environment. This situation can me described by means of a f-f and a e-f PCs between the cube and the environment. Furthermore, a f-f PC can be described as a combination of four v-f BCs.

1.3. Contact transition monitoring and recognition

Critical aspects of robot programming by demonstration are to capture the intention of the user and to distinguish essential actions of the task demonstration from actions that are not relevant for its correct execution. To successfully transfer skills from humans to robots, in compliant motion tasks, it is necessary to recognize and interpret human actions, which are in general composed of a sequence of operations, each of them directly achieving a specific subgoal representable as a sequence of elementary actions [5]. Therefore, to reproduce compliant motion tasks, it is essential to recognize the sequence of CFs made during the demonstration, to detect transitions between consecutive CFs and to estimate some geometric parameters of interest, such as position of the contact points and direction of the contact normals. These are, for instance, the pose of the manipulated object with respect to the gripper of the robot, the pose of the objects in the environment in which the robot operates and some geometric features of the manipulated object. After the demonstration, this information together with the recorded sequences of poses of the manipulated object, and the interaction wrenches between manipulated object and environment, is translated into a force-based task specification for an hybrid robot controller [10].

Contact recognition of a demonstration experiment, also called segmentation of the task, has been studied in several papers see [11] and references therein.

In [11], a method for contact state classification in robot programming of compliant motion tasks by human demonstration has been described. This contact classifier is based on the supervised learning stochastic gradient boosting (SGB) algorithm. SGB algorithm, in the version implemented in [11], was able to distinguish only one specific contact from the rest of the contacts, and this makes it difficult its use for multi-class CF classification. Thus, multi-class CF classification based on this SGB algorithm is not feasible, and some modifications are needed such as the methods proposed in [12, 13], which are considered in this paper as benchmarks for the classification problem.

In [14] and [15], a fuzzy classifier and a neural network classifier are proposed to identify single-ended CFs, using only force and torque sensor data without taking into account the geometric models of the objects. The fuzzy classifier is compared with the SGB in [11] and resulted to be less accurate.

In this paper, we reconsider the classification problem discussed in [11] and provide a solution that generalize to multi-class CF by using the well-known random forest (RF) algorithm proposed in [16] to perform CF classification. The RF algorithm, which belongs to the class of supervised algorithms, estimates the most probable CF conditionally on sensor data from the task demonstration tool. This is a statistical additive regression model, similar to the SGB, which uses twists

and wrenches sensor data from the task demonstration tool as classification features, in order to learn about the dictionary of CFs, that is, the set of CFs, and recognize them by classification.

The idea behind the RF algorithm is that of combining many possibly unrelated classifiers into a larger model. As in the SGB algorithm, the classifiers are classification trees, but the method used in the RF algorithm to estimate trees is such that the tree diversity is increased with respect to SGB. Essentially, in the SGB trees are much more related among them than in the RF algorithm. Increasing diversity provides an upper bound to the estimation error as shown in [16], while such upper bound does not exists for the SGB method. For this reason, we propose RF algorithm as a more suitable classification method than SGB regardless of the number of CFs in the dictionary. Such number only affects the loss function in the classification algorithm, that is, for one CF the binary logit is needed whereas for more CFs the multinomial logit is used. Additionally, we perform multi-class classification, that is, we classify CFs embedded in a dictionary of CFs that has been specified in the training experiment.

It is worth noting that the difference between supervised and unsupervised learning is estimating from already labeled data versus learning, or estimating, from unlabeled data, that is, a dataset where CFs are not known. In the specific case of robot programming by demonstration, the difference between these two approaches results from the trade-off between the amount of information provided by the training experiment and the amount of information provided in the unsupervised model by an expert that accounts for several factors such as the geometry manipulated objects and their materials. Supervised learning is based on a specific demonstration experiment, while unsupervised models rely on a mathematical model, such as the kinestatic model of the contact and the friction model. The latter can be hard to be formulated even for simple objects and/or soft materials. This explains the need of supervised learning algorithms. In fact, while both approaches can be valuable, the supervised one is always valuable if we have an informative enough demonstration experiment, that is, we are able to capture the essential information; this should be much more practical to be used in classifying contacts than the recognition based on a sophisticated mathematical model that would not generally fit every combination of object's geometry and material.

The rest of the paper is organized as follows. In Section 2, the problem of contact classification is formalized, whereas in Section 3, the setup used for the experiments is described. In Section 4, the RF algorithm used to build the contact classifier is described, and in Section 5 the effectiveness of the method for contact classification is proven by the results obtained in real word experiments. In Section 6, a comparison is made between the results obtained with our classifier and those obtained with other classifiers. Some important remarks and conclusions are given in Section 7.

2. Description of the problem

The problem of contact state classification of human demonstrated robot compliant motion tasks can be stated as follows: given the sensor data collected during the demonstration of a compliant motion task, identify the CFs that took place among a pre-specified set or dictionary of CFs.

We suppose that for each CF that can take place during a demonstration of a compliant motion task, the human demonstrator carries out a training experiment that includes these CFs and records the CFs made together with the corresponding sensor data, that is, sensor data are grouped based on the CF in which they were generated. The dictionary of CFs is specified by the user after the training experiment based on the contact that took place during the experiment itself. For instance, in the experiments 1, 2, and 3, whose results are described in Section 5, the dictionary of CF s is that illustrated in Figures 3, 8, and 12, respectively. In general, a sequence of CFs includes several CFs of the same type. For example, Experiment 1 in Section 5 includes two 3f-f CFs (i.e. the cube in corner CF); Experiment 2 contains two v_1-f (vertex-face) CF and two 3f-f CFs; and Experiment 3 has five f-f (face-face) CF and two 3f-f CFs (Figures 3, 8, and 12).

After this training phase, a statistical contact classifier is built using sensor data related to the CFs in the dictionary. With this contact classifier, we are able to classify the CFs that will take place during future demonstrations of compliant motion tasks, which, in this case, are represented by sensor data illustrated in Figures 6, 11, and 15. These data are not used to build the classifier, but only for validation of the classifier built up on sensor data related to the CFs in the dictionary.

In this paper, we suppose that the location of the manipulated object with respect to the demonstration tool is unknown, as well as the location of the objects in the environment in which the demonstration takes place. We also assume that the geometric parameters of the manipulated object are unknown. We only use forces and velocities to perform CFs classification.

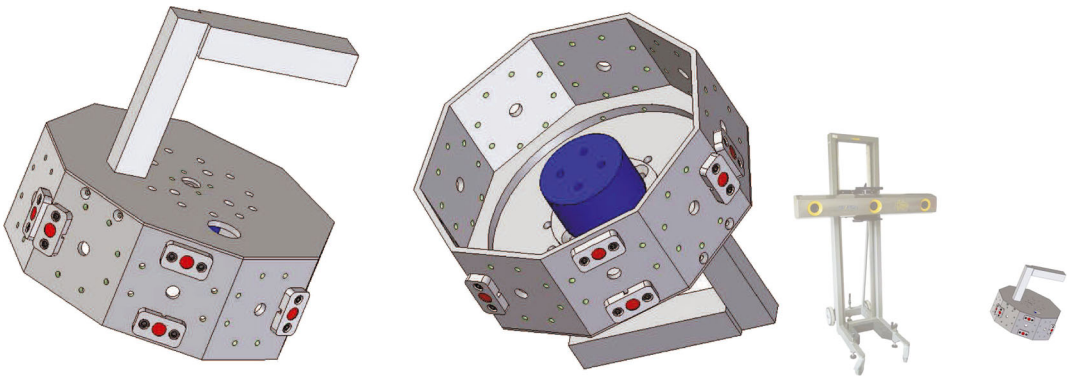
The demonstration setup collects a sequence of poses, twists, that is, linear and angular velocities of the manipulated object, and wrenches, that is, forces and torques that arise during the contact interaction with the environment [3]. We show that in the presence of this kind of uncertainties, we can successfully classify contacts without taking into account the pose information. As suggested in [17], this means that what identifies CFs is the relation between twists and wrenches during the contact rather than the pose of the objects in contact.

3. Experimental setup

To collect sensor data, we used a modular demonstration tool specifically designed for compliant motion tasks [18, 19]. It is attached to the object to be manipulated and located between the hand of the human demonstrator and the object itself. It is equipped with a JR3 (http://www.jr3.com) wrench sensor and a probe for accurate pose tracking with a METRIS K600 optical measurement system (http://www.metris.com) (Figures 1 and 2).

Twists of the manipulated objects are estimated from pose measurements. Data are collected at a frequency of 100 [Hz].

The experimental setup is composed of a cubic manipulated object and an environment consisting of three perpendicular faces. The human demonstrator moves the manipulated object using a demonstration tool, which is assembled onto the manipulated object. A computer-aided design model of the tools is shown in Figure 1, while Figure 2 shows the tool during an experiment. The human demonstrator manipulates the demonstration tool and the object attached to it by means of a handle. The six-dimensional pose of the demonstration tool is measured by the METRIS K600 optical measurement system (Figure 1(b)), measuring the spatial positions of a probe integrated in the demonstration tool, at 100 [Hz] or more, with a volumetric accuracy of 90 [μm]. The demonstration tool has a hollow prismatic shape whose basis is a regular nonagon. A JR3 wrench sensor (Figure 1(a)) is mounted inside the demonstration tool between the demonstration tool and the manipulated object, in order to measure the wrench applied by the human demonstrator to the manipulated object. While the wrench is directly measured by the sensor, the pose and the twist of the manipulated object are estimated. More



(a) The demonstration tool (b) The demonstration equipment

Figure 1. A METRIS K600 measurement system is used to measure the pose of the demonstration tool and to estimate the pose and the twist of the manipulated object. The probe, composed of several light-emitting diode markers, has been integrated in the demonstration tool. A JR3 wrench sensor, assembled inside the demonstration tool, measures the wrench applied by the human demonstrator to the manipulated object. [18, 19].

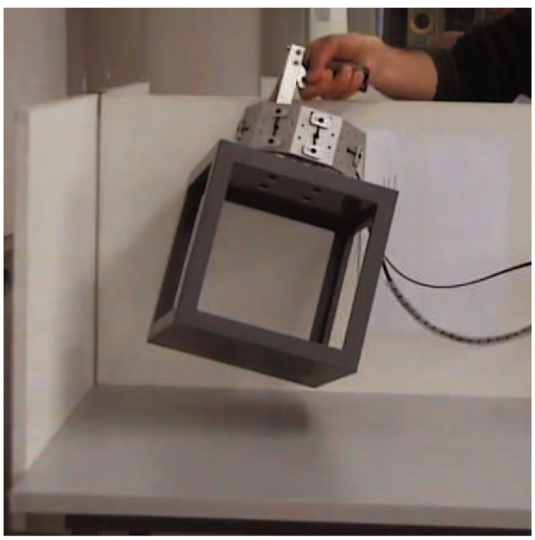


Figure 2. The demonstration tool during an experiment.

precisely, they are estimated by a linear estimator from the measured sequence of pose of the demonstration tool based on a constant acceleration model [20]. This results in a smooth estimation for the pose and twist of the demonstration tool, while deriving the twist from the pose measurements would result in a noisy and inaccurate twist estimation. A Kalman filter is used for this linear estimation problem.

4. Random forest classifier

In the training experiment 12, sensor variables are collected, six of which represent wrenches between the object manipulated by the human demonstrator and the environment in which the demonstration takes place. A wrench

$$\mathbf{w} = (f_x, f_y, f_z, m_x, m_y, m_z)$$

is composed of three forces, f_x,f_y, and f_z along X, Y, and Z axes, respectively and 3 torques m_x, m_y, and m_z about X, Y, and Z axes, respectively. The other six variables represent twists of the manipulated object. A twist

$$\mathbf{t} = (v_x, v_y, v_z, \omega_x, \omega_y, \omega_z)$$

is composed of three linear velocities v_x, v_y, and v_z along X, Y, and Z axes, respectively and three angular velocities ω_x, ω_y, and ω_z about X, Y, and Z axes, respectively.

These observations, along with the indication of the corresponding CFs, are called training set because they are used to build a statistical model, which acts as a CF classifier and approximates the related stochastic process for the sensor data arising from the human demonstration task. To create it, we assume that each CF made in the training experiment is known without uncertainty. Using the same notation as in [21], we denote by $\mathbf{x}_i = \left(x_i^1, x_i^2, \ldots, x_i^{12}\right)$ the i-th observation composed by 12 sensor variables, by $\{\mathbf{x}_i\}_{i=1}^N$ the sequence of N observations made during the training experiment, and by $\{y_i\}_{i=1}^N$ the corresponding response variables, where $y_i \in \mathcal{Y} = \{1, 2, \ldots, k, \ldots, K\}$ is a categorical variable whose values correspond to the indexes of the CFs in the dictionary of CFs with K elements.

Using the data obtained in the training experiment, $\mathbf{D} = \{\mathbf{x}_i\}_{i=1}^N \cup \{y_i\}_{i=1}^N$, our goal is to estimate the function $y = F(\mathbf{x})$ that represents the relation between sensor data from the demonstration tool, $\mathbf{x}$, and the CF, y. In our problem, F is a highly non-linear function in which $\mathbf{x}$ can be thought of as a set of inputs and y as the random output. To estimate this function, we propose a procedure based on the RF algorithm whose statistical properties are detailed in [16]. Here, the description is limited to the standard version of the algorithm.

RF is an additive model given by the ensemble of M non-parametric classifiers, where M has been chosen according to the required precision in estimating y. For our purposes, $M = 1000$ has been found to be adequate even if, as discussed later, the choice of M is not crucial in the RF model compared with other types of classifiers.

In the RF algorithm, the classifiers are classification trees, $\{h_m(\mathbf{x}; \mathbf{a}_m) \in \mathcal{Y}\}_{m=1}^M$, build under the criterion of minimizing of the multinomial deviance. Vector $\mathbf{a}_m$ contains the parameters of the m-th tree, that is, splitting variables and split points that lies in a region of the space spanned by the P sensor variables. In our case, $P = 12$. Parameters $\mathbf{a}_m$ result from the minimization of the tree deviance. For more details on classification trees see, for instance, [22]. Essentially, the RF algorithm randomly extracts a subset of observations, and the tree is grown on subsequent random subsets of sensor variables. This process is repeated M times obtaining thus a forest of classification trees and hence the name "random forest". Once we have estimated the forest in the training experiment, we can predict the response variable y conditioned on a certain vector of sensor data $\mathbf{x}$ in a test experiment, that is, the actual human demonstration task. This estimation just consists in processing $\mathbf{x}$ in all M trees and then taking the most frequent class estimated in all trees. More formally, each h_m is based on a random sample of size $p = \sqrt{P} < P$ of the P variables and a corresponding random sample of size $[N/2]$, where $[\cdot]$ denotes the integer part. This sample is called the in-bag sample and denoted by $\mathbf{D}_{ib}$ to be distinguished from the out-of-bag (OOB) sample denoted by $\mathbf{D}_{oob} = \mathbf{D} \setminus \mathbf{D}_{ib}$. Both sets, $\mathbf{D}_{oob}$ and $\mathbf{D}_{ib}$, represent the m-th random partition of the whole set of states in the training experiment, $\mathbf{D}$, and the classification error of h_m is always estimated with the $\mathbf{D}_{oob}$ sample of size $N - [N/2]$. This is performed in order to avoid overoptimistic prediction errors. The vector $\mathbf{a}_m$ contains splitting variables that are supposed to be the important part in the relation between sensor data and CFs according to the m-th tree. The estimation of the prediction probability for the k-th CF is given by the mean prediction of the M trees

$$\widehat{F}_k(\mathbf{x}) = \frac{\#\{h_m\left(\mathbf{x}; \mathbf{a}_m\right) = k\}}{M},$$

which is just the ratio among the number of trees that predict the k-th CF and the total number of trees in the forest.

Each $h_m(\mathbf{x}; \mathbf{a}_m)$ estimates only a component of the complex kinestatic model that relates CFs and sensor data, while the whole estimation of such complex model is given by $\widehat{F}(\mathbf{x})$. For instance, a single tree may relate one specific wrench or

twist to a specific CF, a set of trees may relate all wrenches and twists with a specific CF and globally the relation among all CFs and all sensor data is given by the $\widehat{F}(\mathbf{x})$ made of thousands of trees.

If

$$k = \text{argmax}_i\{\widehat{F}_1(\mathbf{x}), \ldots, \widehat{F}_i(\mathbf{x}), \ldots, \widehat{F}_K(\mathbf{x})\},$$

then, the most probable CF, given the sensor state $\mathbf{x}$, is the k-th CF, which is our estimated CF.

The validation of $\widehat{F}(\mathbf{x})$ in the training experiment is very important because it should provide a realistic measure of the reliability of the estimation algorithm, before using it in a test experiment and in real world applications. This is made by means of the prediction error for the outcome y of the m-th tree, λ_m, that is, the error estimated in the $\mathbf{D}_{oob}$ sample that has not been used to grow the m-th tree. Such estimation for the m-th tree is less conservative than that obtained using $\mathbf{D}_{ib}$ sample, and it is related with the strength of the m-th tree, that is, the ability of the tree to estimate consistently y. The estimation of the overall error in the forest, $\widehat{\Lambda}_M$, is always made over $\mathbf{D}_{oob}$, and it is just the empirical mean of $\left\{\lambda_m\right\}_{m=1}^{M}$. In this way, we avoid overoptimistic error estimation. Let $\Lambda_M = E_{(Y,\mathbf{X})}(\widehat{\Lambda}_M)$ be the expected prediction error of M trees, where the expectation is calculated for the joint unknown distribution of $(Y, \mathbf{X})$. According to Theorems 1.2 and 2.3 in [16], as $M \to \infty$,

$$\Lambda_M \to \Lambda \leqslant \rho(1 - s^2)/s^2 > 0, \text{almost surely},$$

where $s = 1 - E_{(Y,\mathbf{X})}\left(\lambda_m\right)$ is the classifier's strength, and ρ is the correlation among classifiers, also known as classifier's diversity. Under conditions of Theorems 1.2 and 2.3, the RF algorithm does not overfit if we arbitrarily increase M, and $\widehat{F}(\mathbf{x})$ is a consistent estimator of kinestatic model $F(\mathbf{x})$. This results is extremely important as it avoids to carefully choose the parameter M, because it is sufficient to select M large enough, while with other approaches, such as the SGB, the choice of M is crucial to avoid either overfit or a poor fit as discussed in [11]. A more general discussion on consistency of the RF and other classification methods can be found in [23].

With the RF algorithm, we can select those twists and wrenches that are the most important in classifying the specified dictionary of CFs. Such variable selection is performed by means of each h_m, because $\mathbf{a}_m$ is obtained only using those components of twists and wrenches that assure the largest decrease of the prediction error or the largest increase of s. Therefore, we can assess the importance of each of the P variables based on the total decrease of the prediction error induced in all M trees, say $\{\eta_p\}_{p=1}^{P}$. The greater is η_p; the more likely is that the p-th component is relevant in classifying the CFs in the dictionary. We can sort the P variables in decreasing order according to their respective η_p. The interpretation of the set $\{\eta_p\}_{p=1}^{P}$ is that each component contributes to the complex kinestatic model, but some of them are more important than others. Such information is not used in the RF algorithm itself but it is just a by product and can be used to plan future training experiments and demonstration tasks whose effectiveness can be improved by knowing which sensor variables must be excited more than others in order to successfully characterize the CFs in the dictionary. Another advantage of RF algorithm is that it is highly parallelizable, because each of the M classification tree can be calculated separately. Therefore, such algorithm may take advantages of graphical processing units (http://www.gpu.sourceforge.net) parallel architectures, which are becoming cheaper for industrial applications.

Algorithm 1 summarizes the RF algorithm to classify CFs.

Algorithm 1 Random Forest

Require: p and M

1: **for** $m = 1$ to M **do**
2: Randomly draw $[N/2]$ states and p variables;
3: Compute the m-th tree, $h_m(\mathbf{a}_m)$ by repeating step 2 at each branch of the tree;
4: **end for**
5: **return** The probability a certain state $\mathbf{x}$ to be under the k-th CF as $\widehat{F}_k(\mathbf{x}) = \frac{\#\{h(\mathbf{x};\mathbf{a}_m)=k\}}{M}$;
6: The estimated CF is $\arg\max_{k\in 1,\ldots,K} \widehat{F}_k(\mathbf{x})$.

5. Experimental validation

The experimental validation consists in showing the results of the application of the RF algorithm in recognizing through classification, from sensor data, of the different CFs that occur in a certain sequence of CFs during a demonstration of a

compliant motion task. Sensor data have been manually labeled, with the corresponding CF after the training experiment. Such labeling has been done off-line, because the experimental equipment did not allow for on-line labeling. Thus, the labeled training experiment could be subject to errors that can affect the classifier because of the presence of possible unreliable training data. This classification task is much more complicated than that considered in [11], where only one CF had to be recognized among the CFs in the sequence. In this section, we illustrate the results obtained in three different real world experiments. Each experiment is composed by two series of sensor data: one is used for training, that is, to build the classifier over a given dictionary of CFs, and the other is used for testing, that is, for validation purposes.

5.1. Experiment 1

The sequence of CFs in the training experiment is illustrated in Figure 3. This sequence corresponds to the sequence of sensor data shown in Figure 4, with $N = 2187$. Forces and velocities changes according to the CF that occurred during the training experiment, for instance, a linear force along Z axis can be observed when the cube touch the table. Preprocessing sensor data, for example, using a low pass filter, usually needed in other approaches such as [14, 15, 19], is not required in our method. In general, data filtering is needed whenever parametric approaches, for example, those based on normality are employed in data analysis, while our approach is fully non-parametric. In the estimated $\widehat{F}(\mathbf{x})$ for sensor data in Figure 4, the most important sensor variables for classifying the dictionary of CFs (Figure 3) are described in Figure 5.

It reports, in the horizontal axis, the values of η_p for all sensor variables. We can see that forces are the most important sensor variables, being that along Z axis, the direction of the gravity force, the most important one. This is common to all experiments, and it is due to weight of the manipulated object, which is about 4 [Kg] corresponding to a force of about 40 [N] along Z axis combined with the fact that the human operator never applied a similar force in X and Y directions. It is worth noting that all sensor variable are relevant for classifying the dictionary of CFs, but to recognize the dictionary of CFs in Figure 3. It is more important to excite the force sensor along the Z axis rather than, for instance, velocity sensor along the Z axis. The resulting classifier $\widehat{F}(\mathbf{x})$ is also estimated to be very reliable by its own, in fact the estimated OOB error is $\widehat{\Lambda}_M = 0.01$. Applying it to sensor data gathered in a test experiment that contains a subset of CFs in the dictionary, we can see from Figure 6 that the test sequence of CFs is estimated with an accuracy of 95% of CFs correctly classified. In

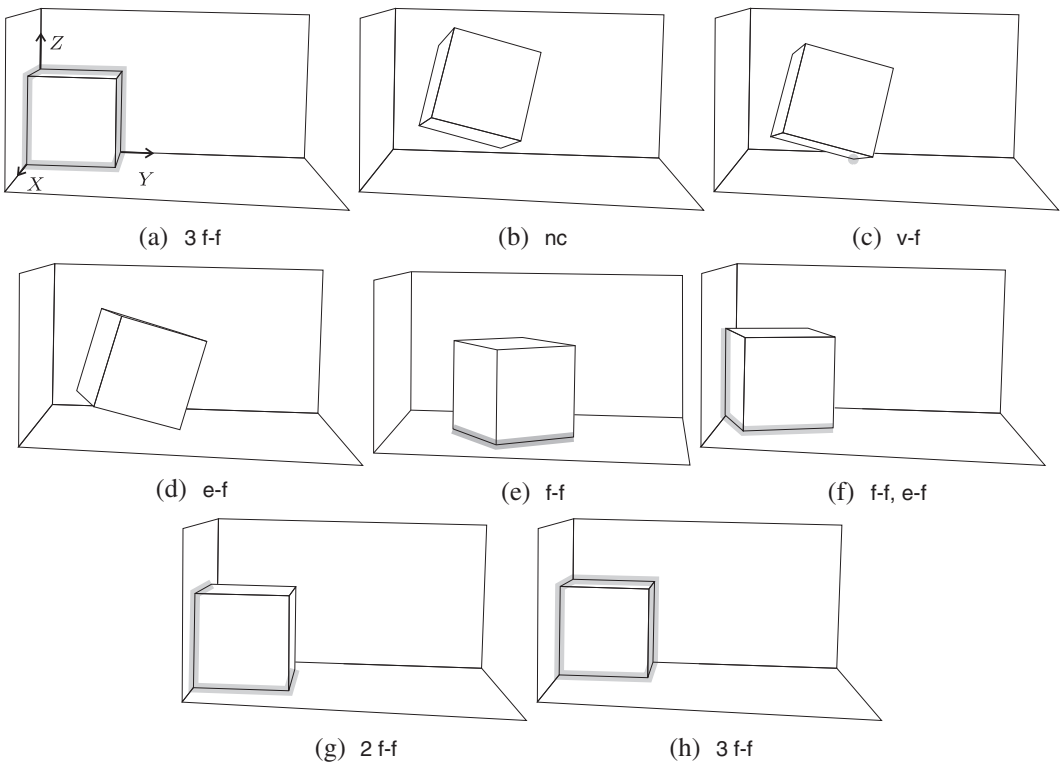


Figure 3. Sequence of contact formations made during the training Experiment 1. Visible elements of the manipulated object that are in contact with the environment are highlighted in gray. The results of the test experiment are shown in Figure 6.

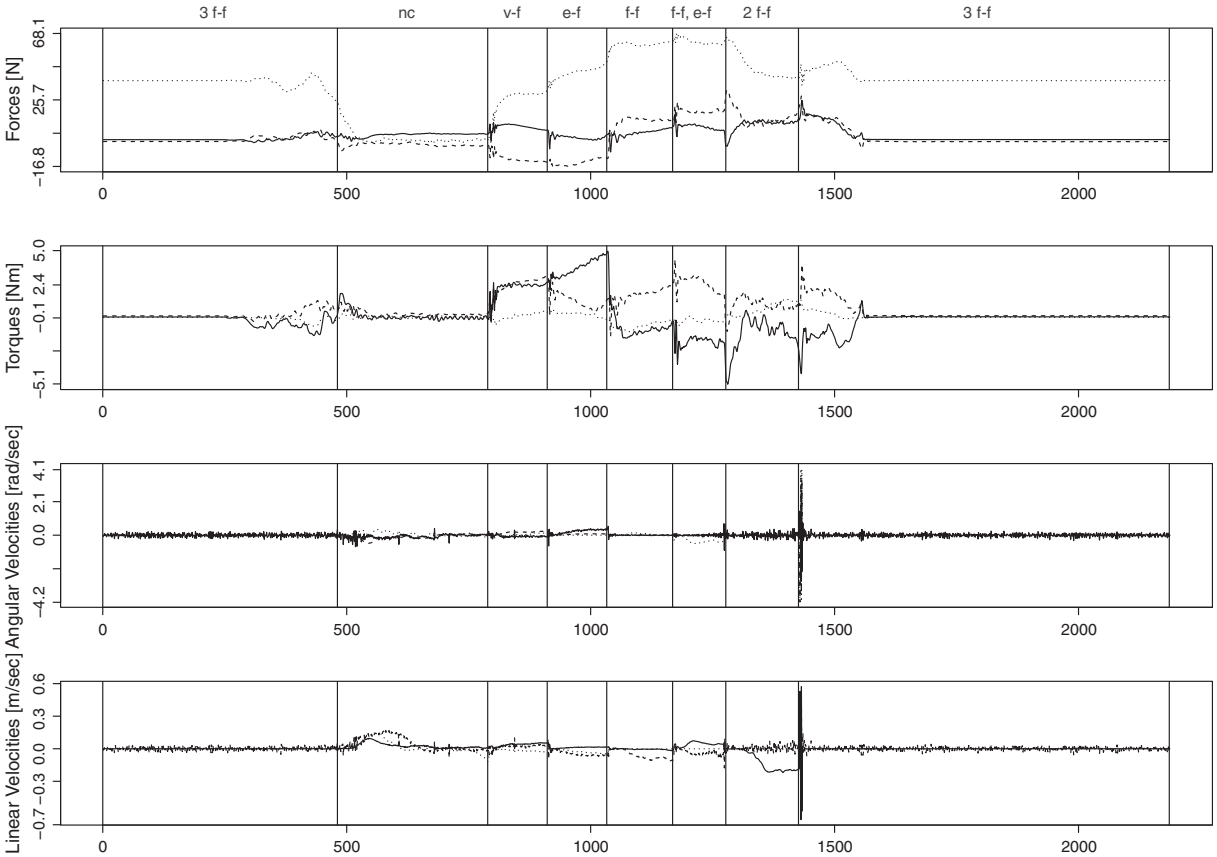


Figure 4. Sensor data for training Experiment 1. Horizontal axis represents time during the demonstration, and vertical axes are the values of the sensor data expressed in the unit appearing into axis label.

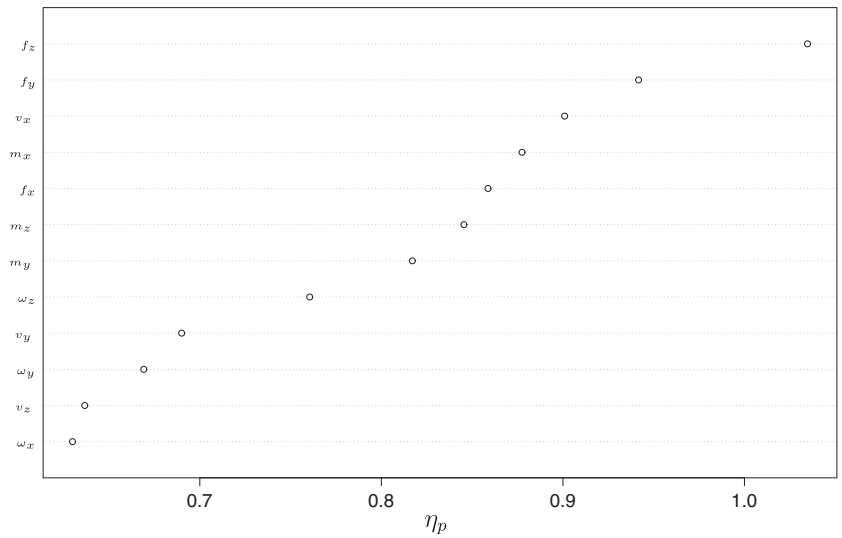


Figure 5. Importance of sensor variables for the classification of the contact formations in the training Experiment 1. Horizontal axis is the sensor variable importance η_p, while vertical axis indicates the specific variable.

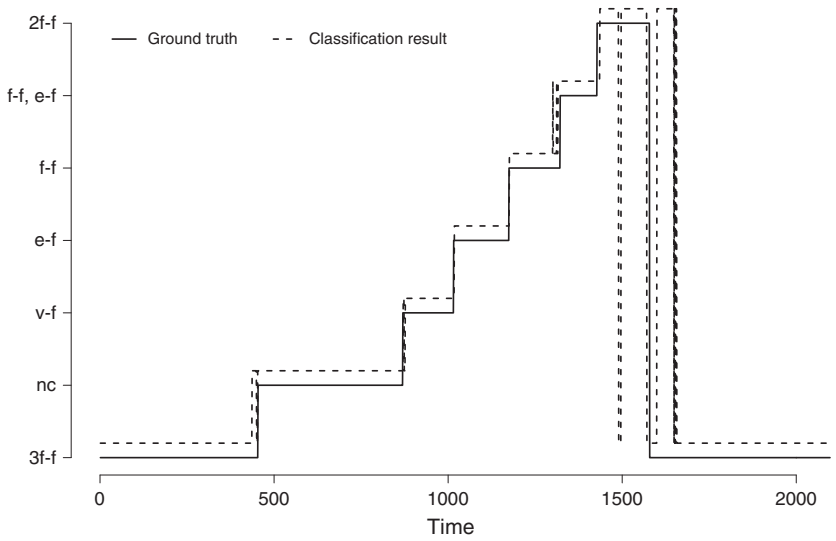


Figure 6. Validation on a test experiment for the dictionary of the contact formations in Experiment 1. Horizontal axis represents time during the validation experiment, while vertical axis indicates the specific CF.

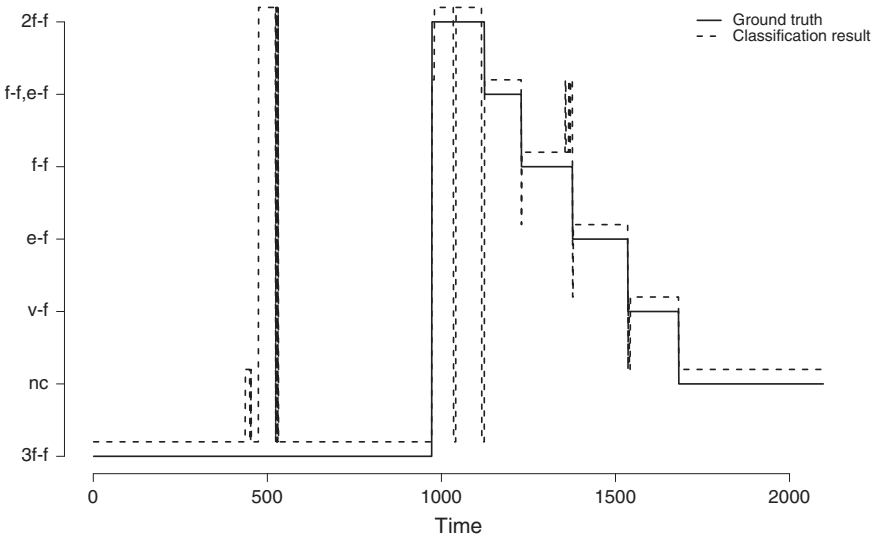


Figure 7. Validation on a simulated test experiment for the dictionary of the contact formations in Experiment 1. In contrast to experiment in Figure 6, this is not a real experiment but just a random permutation of the CFs.

order to highlight that the time and the sequence of CFs are not used in the classifier. Figure 7 illustrates the performance of the classifier in an experiment generated by random permutation of the CFs of validation experiment in Figure 6. We can see that, as expected, the accuracy is not altered.

5.2. Experiment 2

The sequence of CFs in the training experiment is reported in Figure 8, and this corresponds to the sequence of sensor data shown in Figure 9, with $N = 2691$.

For this dictionary of CFs, the most important sensor variable is the force along Z axis as shown in Figure 10. Because $\widehat{\Delta}_M = 0.0115$, the resulting $\widehat{F}(\mathbf{x})$ is very accurate, which is also verified by the results of the application of this classifier to the sensor data gathered in the test experiment. In Figure 11, we can see that almost 94% of CFs are correctly classified.

Note that, in this example, some CFs, such as the e-f contacts in Figure 8(d), (f), (h), and (j), take place in the training experiment for a very short time with respect to the other CFs. However, in spite of this, our algorithm is able to recognize them in the test experiment. Discrepancies in classifying the last part of the test experiment is mostly because of a difficulty

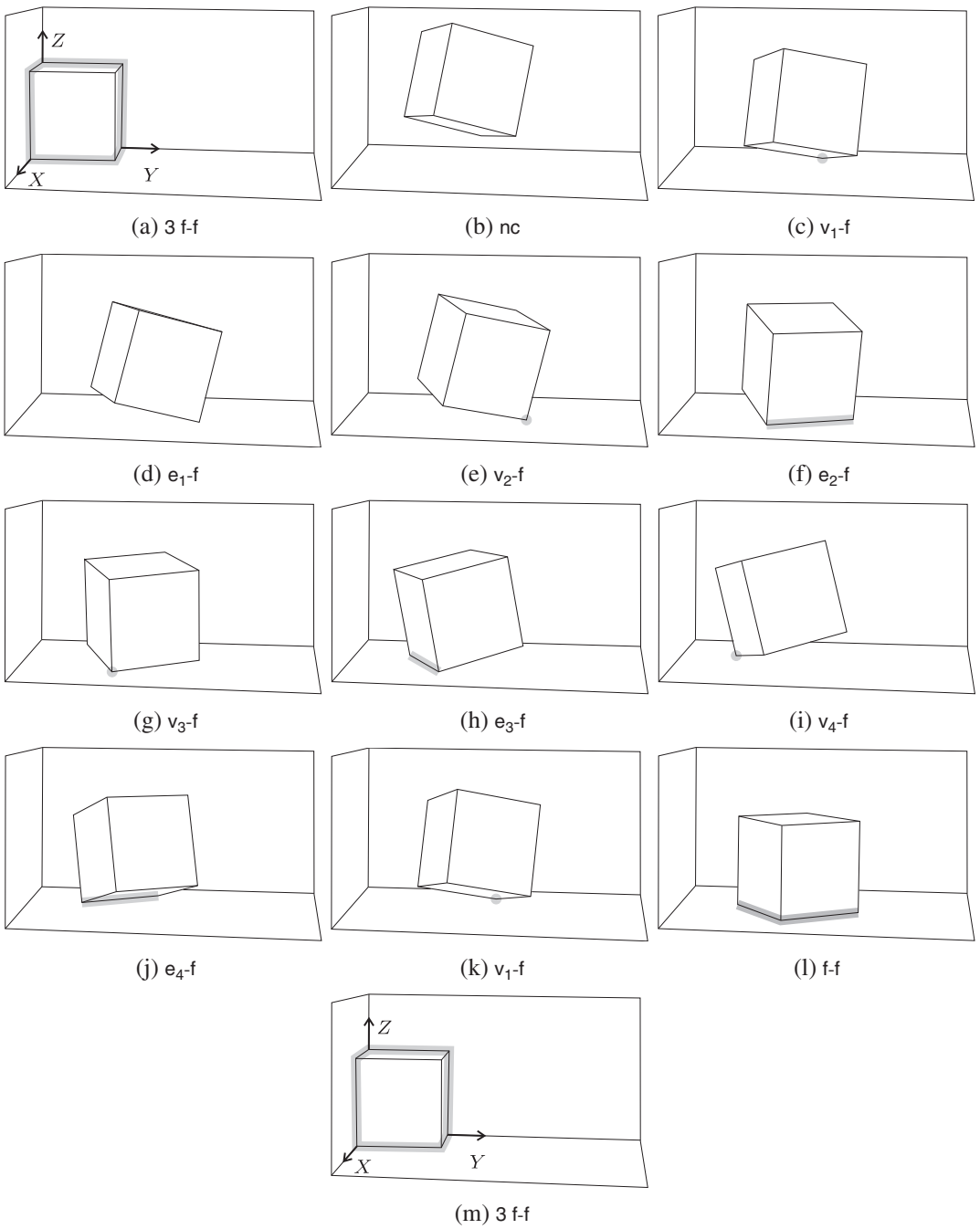


Figure 8. Sequence of contact formations made during the training Experiment 2. Visible elements of the manipulated object that are in contact with the environment are highlighted in gray.

of performing an accurate labeling of the sensor data in the training experiment rather than a defect in $\widehat{F}(\mathbf{x})$, despite this, $\widehat{F}(\mathbf{x})$ is still able to adequately classify the CFs. Moreover, in this case, the sequences of CFs made in the training experiment and in the corresponding test experiment do not coincide. In the training experiment, we have the sequence of contacts v_1-f (k), f-f (l), 3 f-f (m), whereas in the test experiment, there is a transition from the v_1-f contact (k) to nc and then directly to the 3 f-f contact (m). Transitions from nc to 3 f-f are almost impossible to occur directly, especially in human demonstration of compliant motion tasks, where because of an intrinsic lack of precision of the human operator, many other CFs take place [9]. In this case, many intermediate CFs of very short duration occur, for which an accurate off-line labeling is very difficult. However, the classifier is able to recognize some of them, those already included in the dictionary of CFs.

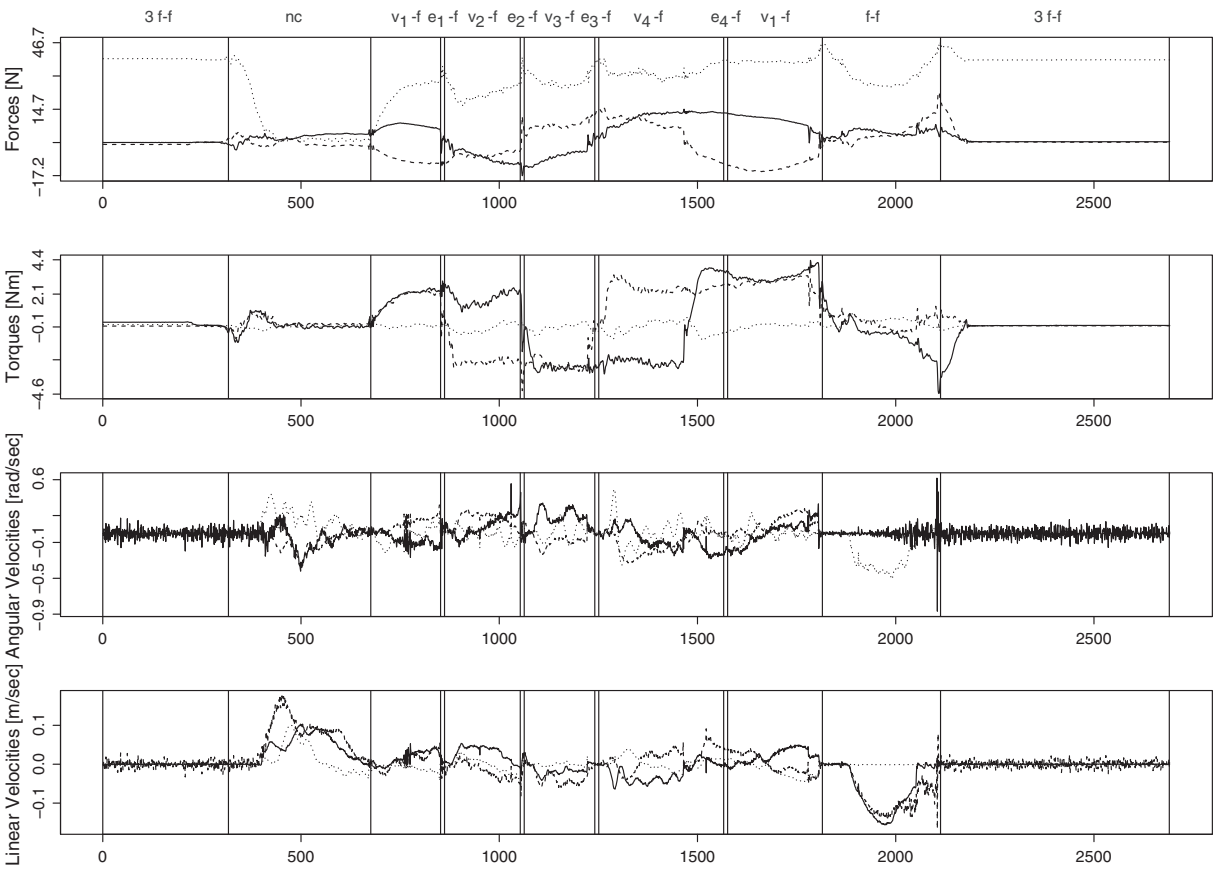


Figure 9. Sensor data for training Experiment 2. Horizontal axis represents time during the demonstration, and vertical axes are the values of the sensor data expressed in the unit appearing into axis label.

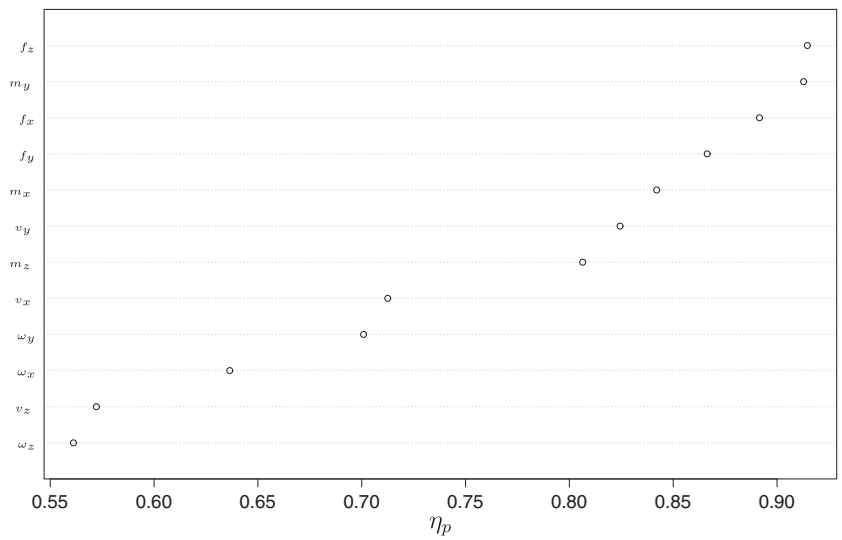


Figure 10. Importance of sensor variables for the classification of the contact formations in the training Experiment 2. Horizontal axis is the sensor variable importance η_p, while vertical axis indicates the specific variable.

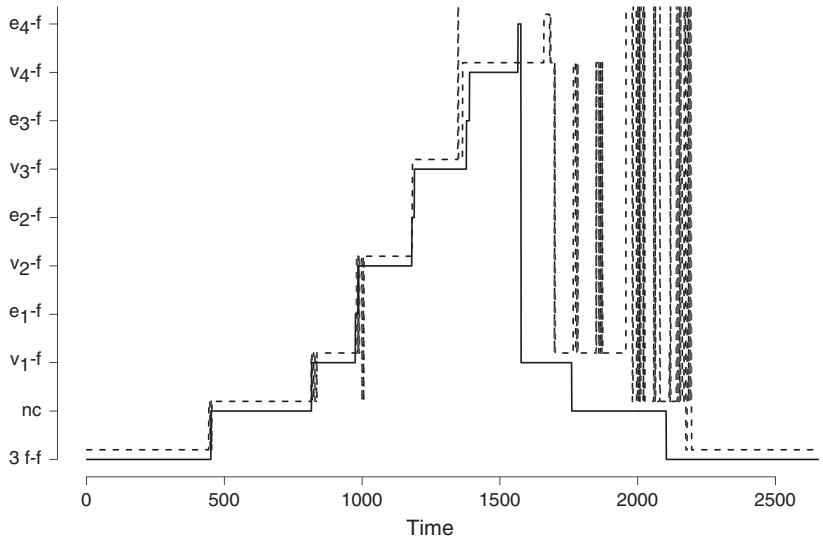


Figure 11. Validation on a test experiment for the dictionary of contact formations in Experiment 2. Horizontal axis represents time during the validation experiment, while vertical axis indicates the specific CF.

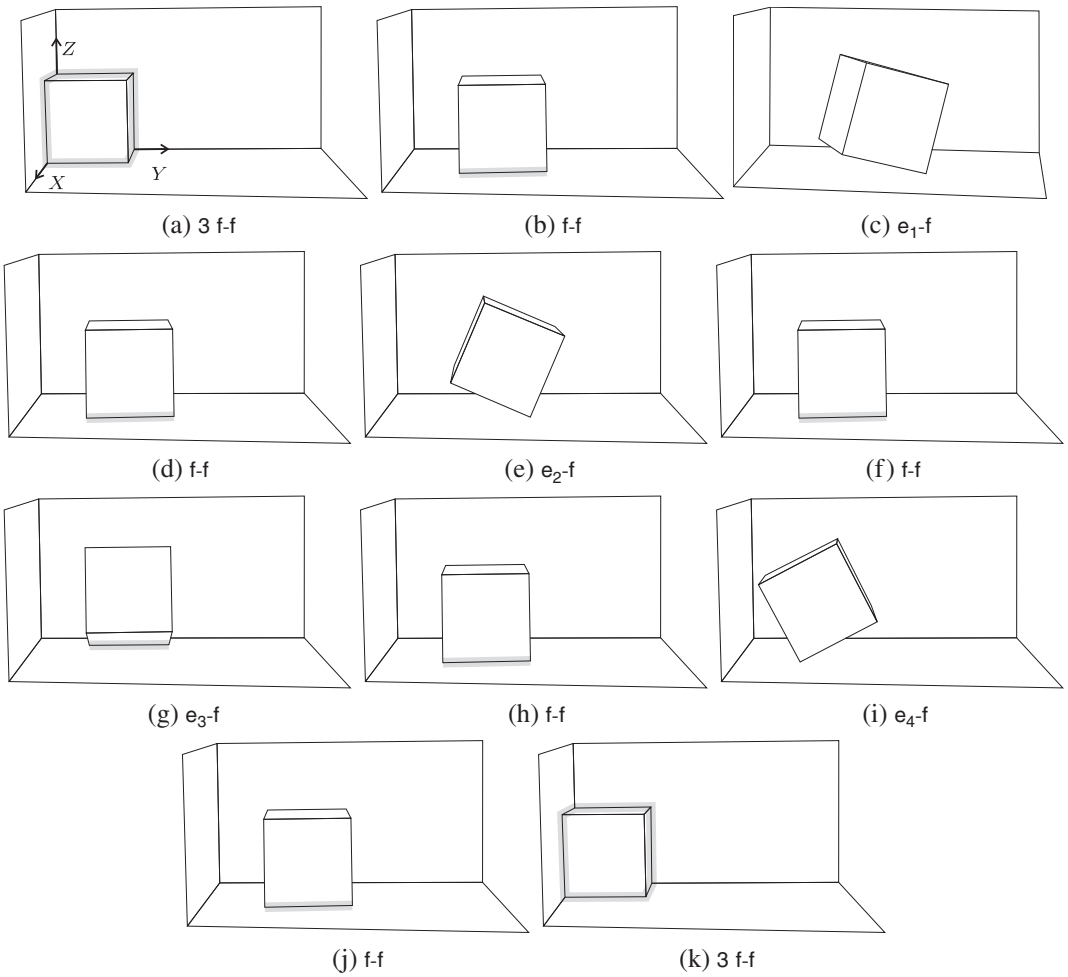


Figure 12. Sequence of contact formations made during the training Experiment 3. Visible elements of the manipulated object that are in contact with the environment are highlighted in gray.

5.3. Experiment 3

The sequence of CFs for the training experiment is shown in Figure 12, which corresponds to the sequence sensor data shown in Figure 13, with $N = 2415$.

Figure 14 reports the most important variables, which essentially are the forces followed by the moments.

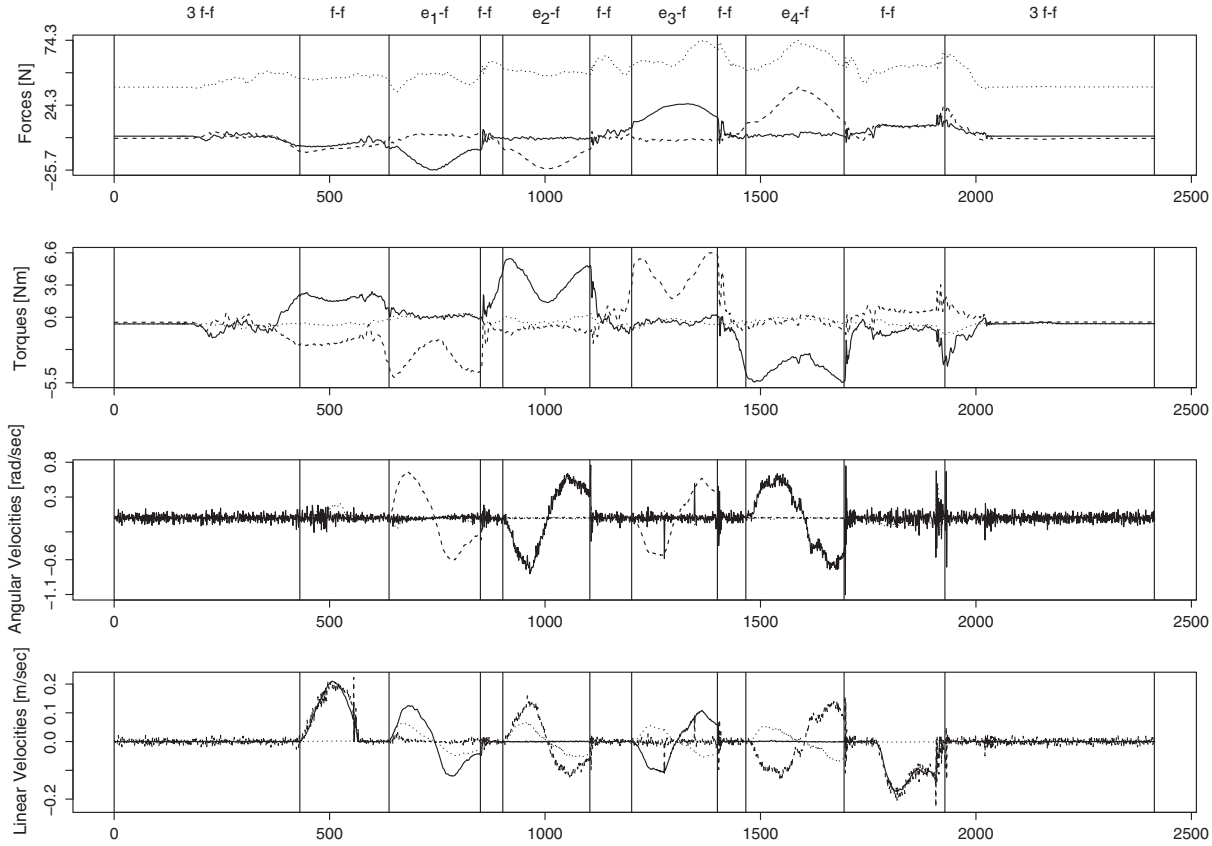


Figure 13. Sensor data for training Experiment 3. Horizontal axis represents time during the demonstration, and vertical axes are the values of the sensor data expressed in the unit appearing into axis label.

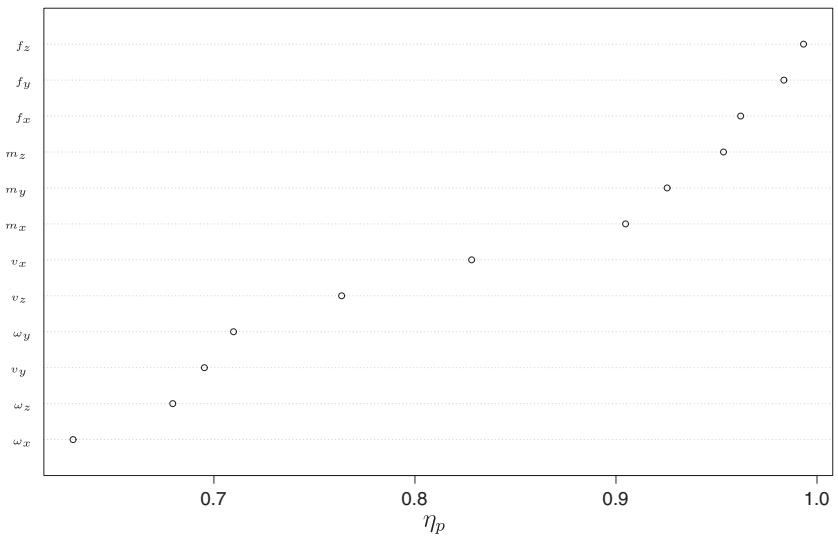


Figure 14. Importance of sensor variables for the classification of the contact formations in the training Experiment 3. Horizontal axis is the sensor variable importance η_p, while vertical axis indicates the specific variable.

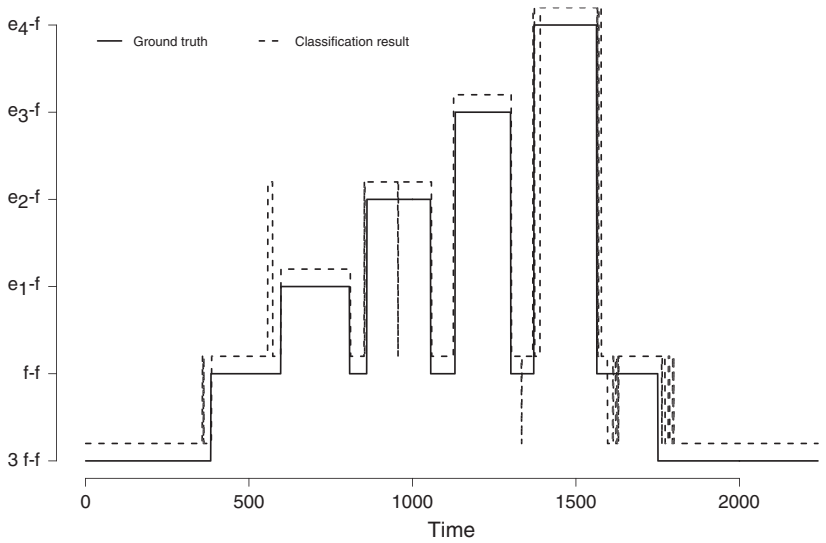


Figure 15. Validation on a test experiment for the dictionary of contact formations in Experiment 3.

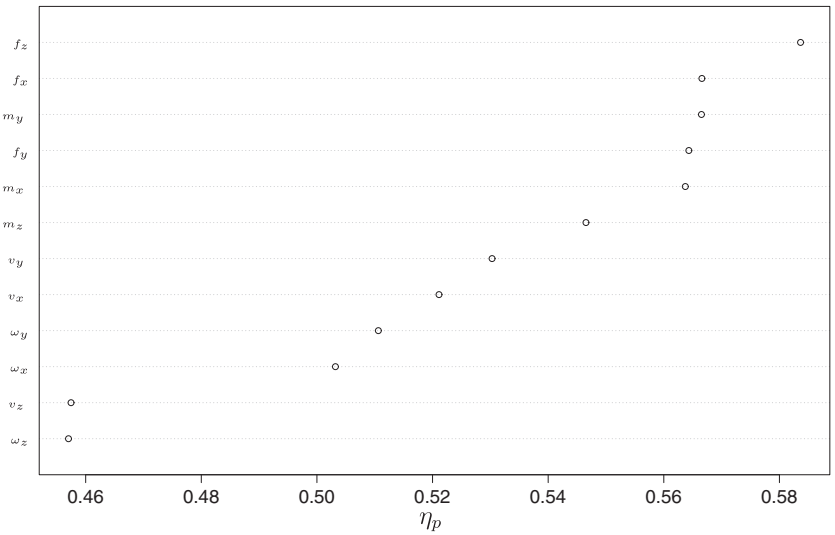


Figure 16. Importance of sensor variables for the classification of the contact formations in the training Combined Experiment. Horizontal axis is the sensor variable importance η_p, while vertical axis indicates the specific variable.

This is due to the fact that all CFs involve small displacements and large rotations about all three axes. The estimated classification error is $\widehat{\Lambda}_M = 0.0108$, which is supported by the accuracy of classification in the corresponding test experiment (Figure 15), that is, 94% of CFs correctly classified.

5.4. Combined experiment

In this case, we created a training experiment by merging sensor data collected in the training experiments described beforehand in order to classify the combination of the sensor data gathered in the test experiments described earlier. In this case, we created a training data and a test data set by merging the data sets of the training and test experiments described previously. More specifically, the sequence of training data is obtained aggregating the training sequence of Experiment 2 after that of Experiment 1 and the training sequence of Experiment 3 after that of Experiment 2 ($N = 7293 = 2187 + 2415 + 2691$). Similarly, the sequence of sensor data to be classified is obtained by combining the sensor data of the corresponding test experiments in the same order. Figure 16 reports the most important variables in this classification, which essentially are forces followed by moments.

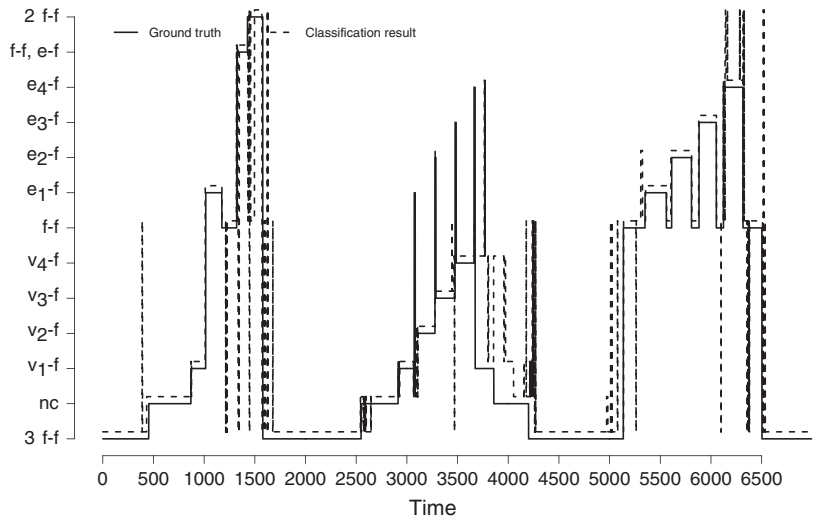


Figure 17. Validation on a test experiment for the dictionary of contact formations in the Combined Experiment. Horizontal axis represents time during the validation experiment, while vertical axis indicates the specific CF.

The estimated classification error is $\widehat{\Lambda}_M = 0.0145$, which is also reflected in the accuracy of classification in the test experiment whose results are given in Figure 17, where 89% of CFs correctly classified.

The fact that the estimated classification error, $\widehat{\Lambda}_M = 0.0145$, of the Combined Experiment does not significantly differ from those of the individual experiments reported previously indicates that the contact classifier is able to learn from shorter training experiments, too. This is an interesting property, which can be used to speed-up demonstrations of compliant motion tasks.

6. Comparison with stochastic gradient boosting classifier for more than two contacts

In this section, we compare the classification error on the test set obtained with the RF algorithm against the classification error obtained with the SGB algorithm.

Before making this comparison, we first discuss the generalization of the SGB algorithm developed in [11] to the general problem of classifying $K > 2$ CFs. Such generalization is based on the original algorithm [12] that extends the AdaBoost algorithm to the multi-class classification problem. It consists in transforming the multi-class classification problem, that implies to recognize K contacts, into K two-class classification problems. Essentially, we apply the SGB algorithm, in [11], to each problem obtaining K different probabilities, and we choose the CF with the highest probability. More specifically, each boosting model has been estimated with the SGB algorithm with shrinkage parameter (the ρ quantity in [11]) equal to 0.01, the optimal number of boosts has been chosen between 1 and 10,000 trees, which have been grown up to 6 way interactions. Half of the sample has been used as OOB sample; final nodes are composed by no less than 50 states; and finally, the OOB error has been estimated with 10-fold cross validations. This algorithm is computationally very demanding because we have to estimate a SGB model for each CF.

We also considered the multi-class AdaBoost algorithm proposed in [13], which will be called AdaBoost-SAMME algorithm or just SAMME in order to avoid confusion. Such algorithm is another generalization of the boosting algorithm for the multi-class classification problem in which the parameter $\alpha^{(m)}$, that controls the weight of the at the m-th boost, is $\alpha^{(m)} = \log(1 - \lambda_m) - \log(\lambda_m) + \log(K - 1)$, where λ_m is the prediction error of the m-th tree.

For the three experiments described earlier, Figures 18, 19, and 20 report the evolution, with respect to the first 100 values of M (i.e. boosts or number of trees), of the error for the training and test sets with the SGB and SAMME algorithms, along with the evolution of $\widehat{\Lambda}_M$ for the RF model in the training set. Finally, the horizontal dashed line represents the actual classification error, on the test set, of the estimated RF with $M = 1000$ trees.

We can see that the latter is always smaller than that obtained with the SGB classifier and with the AdaBoost-SAMME. In particular, the AdaBoost-SAMME seems to make a severe under estimation of the error in the training set, being large in the test experiment. Therefore, we may say that there is a relevant improvement in the RF model with respect to the extension of the SGB model discussed in the final section of [11] and also with respect to the SAMME algorithm considered in [13]. The reason of this relies on the increased classifier diversity of the RF with respect to boosting algorithms.

The RF approach, compared with the SGB methods, has the following advantages:

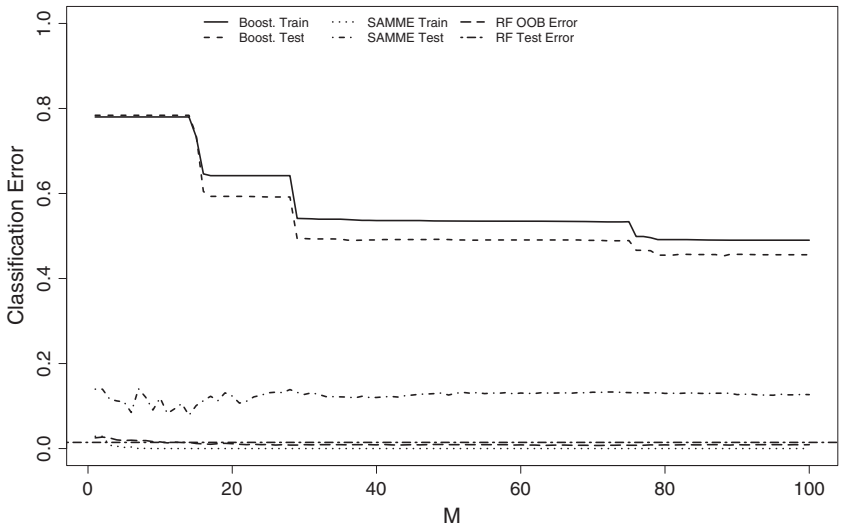


Figure 18. AdaBoost classification errors for training and test Experiment 1. Comparison with the random forest out-of-bag error, $\widehat{\Lambda}_M$. Horizontal axis represents the number of boosts M, while the vertical axis the classification error.

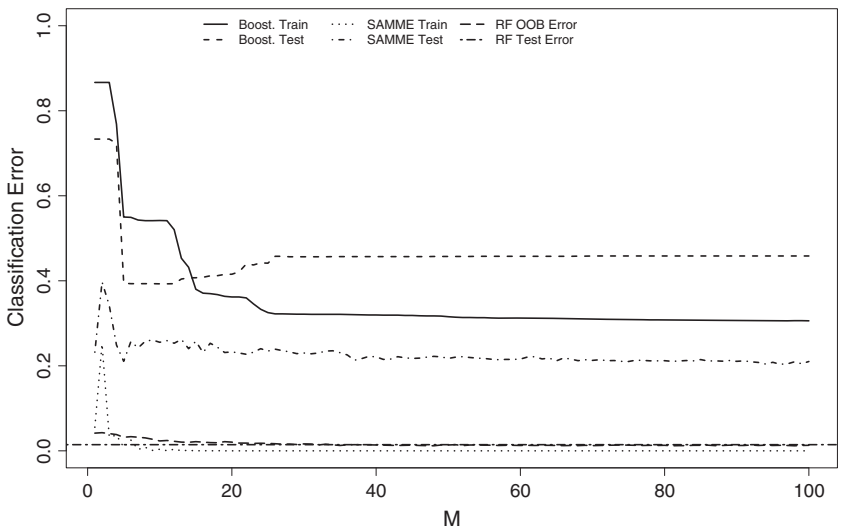


Figure 19. AdaBoost classification errors for training and test Experiment 2. Comparison with the random forest out-of-bag error, $\widehat{\Lambda}_M$. Horizontal axis represents the number of boosts M, while the vertical axis the classification error.

- RF classifiers are computationally faster to be evaluated because they can be parallelizable, while the boosts in the SGB method must be calculated in a sequence of related steps.
- RF approach does not overfit as the number of trees is increased, while the SGB method does. Thus, the tuning, for example, the choice of the number of boosts, is a crucial aspect of the SGB algorithm, while this is not true for the RF algorithm.

For all these reasons, it is possible to conclude that the advantages in using the RF classifier are as follows: ease of implementation (no tuning parameters), suitability for industrial applications (lower costs for parallel computing), and better accuracy. The latter aspect can be relevant or not depending on the specific application of the Human demonstration task, for example, the error can be relevant if the task involves fragile materials or not, explosives or inert materials, and others.

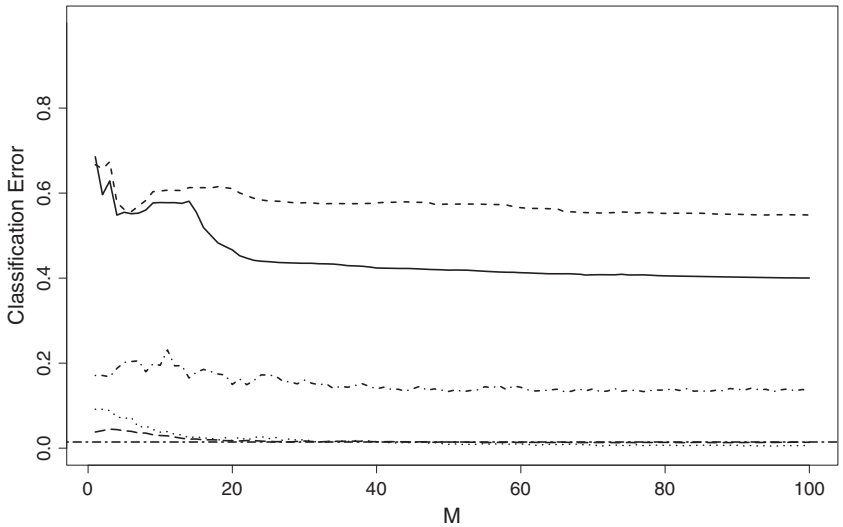


Figure 20. AdaBoost classification errors for training and test Combined Experiment. Comparison with the random forest out-of-bag error, $\widehat{\Lambda}_M$. Horizontal axis represents the number of boosts M, while the vertical axis the classification error.

7. Concluding remarks

In this work, we improved state-of-the-art contact state classification algorithms for robot programming of compliant motion tasks by human demonstration, in particular, the binary stochastic gradient boosting classifier proposed in [11] and its generalizations to multi-class contact classification. Our approach is based on the RF algorithm, which has never been applied to contact state classification. Such improvement is essentially because of the fact that in classifier aggregation, as in the boosting and in the RF algorithms, it is important not only the strength of each classifier, which is a tree in both cases, but also the classifier diversity. If we increase classifier diversity then we avoid overfitting and improve the estimation [16]. In this way, the RF algorithm tends to produce, at each iteration, trees that are more independent to each other by randomly choosing subsets of sensor data and subsets of sensor variables. This is why RF performs better than boosting algorithms (i.e. SGB, AdaBoost, or AdaBoost-SAMME).

In the approach presented in this paper, the sensors data have not been regarded as a time series for two reasons: firstly because this would make the model more complicated and unsuitable for industrial applications; secondly because the improvements in the CFs classification that could be achieved with this approach, would be marginal with respect to the large accuracy yet reached without time information (i.e. around 90% of CFs correctly classified). Typically, time information is used in particle filters together with all possible contacts between two polyhedral objects, which are, in general ,hundreds even for very simple objects; see e.g. [19]. Such unsupervised approaches are based on topological information contained in the CF Graph, that is, the graph that connects all possible CFs [9], which is used to derive the probability of being in a certain CF at certain time t, CF_t, conditionally on CF_{t-1}. However, automatic computation of the CF Graph [9] depends on the geometry of the objects that is not always available, except for very simple objects. Knowledge about the contact states that took place during a demonstration can be incorporated into an estimator of the unknown geometric parameters of the objects involved in the demonstration to build a system that first performs contact classification and then geometric parameter estimation. In such a system, the contact formations would not be random variables as in [19], which performs contact classification and geometric parameter estimation at the same time. This represents a significant simplification, which demonstrates the necessity of a contact classification algorithm.

The advantages of CF estimation based on supervised learning, such as the RF or SGB algorithms, with respect to unsupervised methods, in particular those based on particle filtering as in [10], are as follows:

- A description of the geometry of the objects is not required.
- The calculation of the CF Graph is not needed. This is an advantage because its computation has prohibitive complexity even for very simple geometric object models [9].
- The specification of the kinestatic model is totally avoided, while in other approaches it must be built for every type of contact, which is very complicated in case of multiple contacts [3].
- Our classifier works in real time, whereas, for instance, a sequential Monte Carlo analysis takes hours on a standard computer.

- Our approach to CF classification is based on a non-parametric statistical model that differs from parametric models in the support space of the parameter, which is a subset of the real space for parametric models and a subset not necessarily of the real space for non-parametric models [24]. If enough data are available, non-parametric models are more accurate than parametric ones because they are more flexible, while they can perform worse than parametric models when substantial information on the real parameters is available, and very few observations are considered. However, in the case considered in this paper there is very few information on how forces and velocities allow to recognize CFs, while, at the same time, a large amount of sensor data are available. For instance, suppose that a parametric model were based on a certain model of friction between the manipulated object and the environment. If the assumed friction model does not fit the actual friction during the demonstration of a compliant motion task, then the parametric classification would fail. On the contrary, we do not assume any friction model, but this is intrinsic in the estimated relation between contact velocities and forces that take place during the training experiment. In this sense, the available information in the problem considered in this paper is enough for a non-parametric model. This is a qualitative statement as it depends the on the specific application, for instance, the materials of manipulated object. In fact, CFs between rigid objects are easier to be detected than between deformable objects. We can say that the number of used sensor data, N, is enough to achieve classification accuracy between 90% and 95%.
- In our approach, it is possible to understand what sensor data are most important in classifying a dictionary of CFs and this information can be used to improve the classification process. However, such information must be provided beforehand in unsupervised methods.

Acknowledgements

We would like to thank Joris De Schutter and Herman Bruyninckx for making available to us the robotics laboratory of the Department of Mechanical Engineering of the Katholieke Universiteit Leuven, Belgium, where the experiments have been conducted.

Stefano Cabras and María Eugenia Castellanos have been partially supported by Ministerio de Ciencia e Innovación grants MTM2013-42323, ECO2012-38442, RYC-2012-11455, by Ministero dell'Istruzione, dell'Univesità e della Ricerca of Italy and by Regione Autonoma della Sardegna CRP-59903.

Ernesto Staffetti have been partially supported by the project TRA2013-47619-C2-2-R of the Spanish Ministerio de Economía y Competitividad (2014-2016).

References

1. Billard A, Calinon S, Dillmann R, Schaal S. Robot programming by demonstration. In *Handbook of Robotics*, Siciliano B, Khatib O (eds), chapter 59. Springer: Secaucus, NJ, USA, 2008; 1371–1394.
2. Villani L, De Schutter J. Force control. In *Handbook of Robotics*, Siciliano B, Khatib O (eds), chapter 7. Springer-Verlag: Berlin Heidelberg, 2008; 161–185.
3. Staffetti E. Analysis of rigid body interactions for compliant motion tasks using the Grassmann–Cayley algebra. *IEEE Transactions on Automation Science and Engineering* 2009; **6**(1):80–93.
4. Raibert M, Craig JJ. Hybrid position/force control of manipulators. *ASME Journal of Dynamical Systems, Measurements, and Control* 1981; **102**:126–133.
5. Bruyninckx H, De Schutter J. Specification of force-controlled actions in the "Task Frame Formalism": a survey. *IEEE Transactions on Robotics and Automation* 1996; **12**(5):581–589.
6. Mason MT. Compliance and force control for computer controlled manipulators. *IEEE Transactions on Systems, Man, and Cybernetics* 1981; **11**(6):418–432.
7. De Schutter J, De Laet T, Rutgeerts J, Decré W, Smits R, Aertbeliën E, Claes K, Bruyninckx H. Constraint-based task specification and estimation for sensor-based robot systems in the presence of geometric uncertainty. *The International Journal of Robotics Research* 2007; **26**(5):433–455.
8. Latombe JC. *Robot Motion Planning*. Kluwer Academic Publishers: Boston, MA, 1991.
9. Xiao J, Ji X. On automatic generation of high-level contact state space. *The International Journal of Robotics Reseach* 2001; **20**(7):584–606.
10. Meeussen W, Staffetti E, Bruyninckx H, Xiao J, De Schutter J. Integration of planning and execution in force controlled compliant motion. *Robotics and Autonomous Systems* 2008; **56**(5):437–450.
11. Cabras S, Castellanos M-E, Staffetti E. Contact-state classification in human demonstrated robot compliant motion tasks using the boosting algorithm. *IEEE Transactions on Systems, Man, and Cybernetics, Part B* 2010; **40**(5):1372–1386.
12. Schapire RE, Singer Y. Improved boosting algorithms using confidence-rated predictions. *Machine Learning* 1999; **37**:297–336.
13. Zhu J, Zou H, Rosset S, Hastie T. Multi-class adaboost. *Statistics and Its Interface* 2009; **2**:349–360.
14. Everett LJ, Ravuri R, Volz RA, Skubic M. Generalized recognition of single-ended contact formations. *IEEE Transactions on Robotics and Automation* 1999; **15**(5):829–836.
15. Skubic M, Volz RA. Identifying single-ended contact formations from force sensor patterns. *IEEE Transactions on Robotics and Automation* 2000; **16**(5):597–603.
16. Breiman L. Random forests. *Machine Learning* 2001; **45**:5–32.

17. Bruyninckx H, Demey S, Dutré S, De Schutter J. Kinematic models for model based compliant motion in the presence of uncertainty. *The International Journal of Robotics Reseach* 1995; **14**(5):465–482.
18. Rutgeerts J, Slaets P, Schillebeeckx F, Meeussen W, Verdonck W, Stallaert B, Princen P, Lefebvre T, Bruyninckx H, De Schutter J. A demonstration tool with Kalman filter data processing for robot programming by human demonstration. *Proceedings of the 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Edmonton, Alberta, Canada, 2005,3592–3597.
19. Meeussen W, Rutgeerts J, Gadeyne K, Bruyninckx H, De Schutter J. Contact state segmentation using particle filters for programming by human demonstration in compliant motion tasks. *IEEE Transactions on Robotics* 2007; **23**(2):218–231.
20. Bar-Shalom Y, Li X. *Estimation and Tracking, Principles, Techniques, and Software*. Artech House: Norwood, MA, 1993.
21. Friedman JH. Stochastic gradient boosting. *Computational Statistics and Data Analysis* 2002; **38**:367–378.
22. Hastie T, Tibshirani R, Friedman JH. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer-Verlag: New York, NY, 2001.
23. Biau G, Devroye L, Lugosi G. Consistency of random forests and other averaging classifiers. *Journal of Machine Learning Research* 2008; **9**:2015–2033.
24. Bickel PJ, Doksum KA. *Mathematical Statistics: Basic Ideas and Selected Topics* (2nd edn). Pearson Prentice Hall: Upper Saddle River, N.J., 2007.