HAL
open science

Multi-users online recognition of technical gestures for natural Human-Robot Collaboration in manufacturing

Eva Coupeté, Fabien Moutarde, Sotiris Manitsaris

To cite this version:

Eva Coupeté, Fabien Moutarde, Sotiris Manitsaris. Multi-users online recognition of technical gestures for natural Human-Robot Collaboration in manufacturing. Autonomous Robots, 2018, 10.1007/s10514-018-9704-y . hal-01700868

HAL Id: hal-01700868

https://minesparis-psl.hal.science/hal-01700868v1

Submitted on 5 Feb 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Noname manuscript No.
(will be inserted by the editor)

Multi-users online recognition of technical gestures for natural Human-Robot Collaboration in manufacturing

Eva Coupeté · Fabien Moutarde · Sotiris Manitsaris

Received: date / Accepted: date

Abstract Human-Robot Collaboration in industrial context requires a smooth, natural and efficient coordination between robot and human operators. The approach we propose to achieve this goal is to use online recognition of technical gestures. In this paper, we present together, and analyze, parameterize and evaluate much more thoroughly, three findings previously unveiled separately by us in several conference presentations: 1/ we show on a *real prototype* that multi-users continuous real-time recognition of technical gestures on an assembly-line is feasible ($\approx$ 90% recall and precision in our case-study), *using only non-intrusive sensors* (depth-camera with a top-view, plus inertial sensors *placed on tools*); 2/ we formulate an end-to-end methodology for designing and developing such a system; 3/ we propose a method for *adapting to new users* our gesture recognition. Furthermore we present here two new findings: 1/ by comparing recognition performances using several sets of features, we highlight the importance of choosing features that *focus on the effective part of gestures*, i.e. usually *hands movements*; 2/ we obtain new results suggesting that enriching a multi-users training set can lead to higher precision than using a separate training dataset for each operator.

Keywords Real-time online gesture recognition, Human-Robot Collaboration, Multi-users technical gesture recognition, Collaborative robotics, Gesture recognition from depth-video

Center for Robotics,
MINES ParisTech,
PSL Research University
60 boulevard Saint Michel
75006 Paris
France
E-mail: firstname.lastname@mines-paristech.fr

1 Introduction

The development of robots is currently increasing in our society, and also in our industries. Social robots have already been used in various contexts: assistant for elderly people, stimulation for autistic children, guide in museum or sale assistants in stores.

In the industrial context, robots are present since the 1950's. Until recently, they were always located in isolated areas where operators are not allowed to go while robots are running. In the last years, collaborative robots emerged on assembly-lines. These robots are smaller and can work in co-presence or in collaboration with operators, in the same area. Issues arise with the introduction of these robots. The first one is to guarantee the safety of the operators working nearby those collaborative robots. Technology advances allowed to develop "safe" robots, i.e. robots with with a limited strength and embedded sensors to prevent any injury to operators. A second issue is to make the collaboration smooth and efficient between these robots and operators. In this study, we propose to use online recognition of technical gestures to address the issue. We think that gesture recognition can help the robot to synchronize its tasks with the actions of operators, can allow the robot to adapt its speed, and also can make it able to understand if something unexpected happens. In this paper, we use the word "gesture" to refer to the actions needed to perform the tasks on the assembly-line.

The rest of this article is composed of five sections. In Section 2, we present related works, as well as our own previous research, on Human-Robot Collaboration in manufacturing and gesture recognition. In Section 3 we describe our real prototype and use-case, and how to choose the gesture classes that should be recognized for ensuring human-robot coordination. In Section 4,

we summarize our end-to-end methodology to *continuously* recognize technical gestures *in real-time*, including a method to adapt to a new user our learnt gesture recognition system. All our experimental results are presented in Section 5. This section also includes in 5.2 new results highlighting the importance of choosing features that *focus on the effective part of gestures* i.e. usually *hands movements*, and in 5.5.2 new comparative evaluations suggesting that enriching a multi-users training set can lead to higher precision than using a separate training dataset for each operator. Finally, we recapitulate and conclude in Section 6.

2 Related and previous works

2.1 Human-Robot Collaboration in manufacturing

The first robots useful for men have been introduced in factories in the 1950's. These robots were able to perform repetitive, tiresome, and dangerous tasks. Since then, industrial robots have been very present on assembly-lines, working on specific areas, away from human operators. Although these robots are efficient, they make assembly-lines not very flexible, and cannot be used on assembling tasks where human presence is required. Nowadays, manufacturers tend to create mixed environments, where robots and operators can work on the same area. This new way of working combines human skills (intelligence, adaptability and dexterity) with robot skills (strength and repeatability). The introduction of collaborative robots in factories provides more flexibility and productivity (Hägele et al, 2002), and relieves human operators from physically-demanding tasks and/or from working using undesirable postures, that can lead to musculo-skeletal disorders. These robots are designed to be intrinsically safe: their strength is limited, and they have built-in sensors which prevent them to hurt operators which are nearby.

Sharing work between an operator and a robot can be executed in different ways. They can work in collaboration on the same task, or on two different tasks in the same area, in co-presence. Shi et al (2012) proposed different degrees of work sharing. At the lowest level, robot and operator do not have any contact and work in two different spaces, but without any barriers between them. On the second level, the operator can go into the robot space, but this will automatically halt it. Finally, in the upper level, the robot and the operator cooperate on a common task. Other studies have been done to prove the feasibility of collaborative tasks with a robot, e.g. the assembly and disassembly of pieces (Corrales Ramón et al, 2012), and the assembly of constant-velocity joint (Cherubini et al, 2016).

However, sharing work between an operator and a robot requires an adaptation from the operator. Human-robot collaboration is not as natural as between humans, and new ways of communication must be established. Hoffman and Breazeal (2007), have shown that the anticipation on a future task can improve the efficiency and fluidity of a human-robot collaboration. Dragan et al (2015), have demonstrated that legible motions from the robot during the execution of a known task enable a more fluent collaboration with a human. (Chen et al, 2015) proposed an approach for recognizing hand gestures of a human operator during an assembly task in collaboration with a robot co-worker. Schrempf et al (2005) proposed a method to synchronize robot and human actions using a Dynamic Bayesian Network. Rickert et al (2007) presented a collaborative robot that is equipped with speech recognition and visual object recognition, and is able to follow the operator hands. This robot uses these informations to anticipate on the next task. Bannat et al (2011) introduced the term "Cognitive Factory" for industrial environments with cognitive capacities, in order to make the machines more autonomous. Lenz et al (2008) created a smart collaborative workspace with several sensors to enable the collaborative robot to understand their environment.

2.2 Gesture recognition

Gesture recognition consists in capturing and interpreting human movements, allowing to understand which action is being performed. It is a growing research field, in which new technologies recently brought significant progresses. Indeed, new sensors (like depth-cameras or light and small inertial sensors) now enable an easy and more complete capture of gestures. In the following parts, we review and describe the usual successive steps needed for creating a gesture recognition system.

2.2.1 Motion capture sensors

Different types of sensors have been used to recognize gestures. The oldest, and historically most used, are RGB cameras. With these cameras, it is possible to have an almost complete (if few occlusions) understanding of the scene and to be non-intrusive. Laptev and Lindeberg (2003), Dollar et al (2005) and Oikonomopoulos et al (2005) proposed methods to detect interest points in RGB videos, and use them to describe the filmed action. Wang et al (2009) proposed to use trajectories of sampling points in successive frames to describe an action.

Depth-cameras are more recent, but already commonly used to recognize human actions, because of 3D information they convey, which makes extraction of gestures easier. Chen et al (2013) proposed a survey on motion analysis using depth-data, Zhang and Parker (2011) adapted to depth-video the cuboid RGB video features, Biswas and Basu (2011) used movement of a person filmed with a fixed depth-camera to recognize gestures.

Inertial sensors are also used for gesture recognition, but are intrusive because they must be fixed onto the user for capturing his/her movements. Bulling et al (2014) proposed an "*Activity Recognition Chain*" to recognize gestures with inertial sensors. Dong et al (2007) and Junker et al (2008) used accelerometers to recognize actions.

Data from several types of sensors can be used simultaneously to recognize gestures. Chen et al (2016) used a depth-camera and a wearable inertial sensor to recognize actions. To fuse the data coming from the different sensors, a decision level scheme was adopted.

2.2.2 Computing features from sensors

Depending on the sensor that is used for capture of movements, different types of features can be extracted. In this section we focus on the extraction of features *from depth-images*.

A first group of features are those related to the global posture of the human, for example his skeleton. Shotton et al (2011) used a large database, composed of real and synthetic images maps, to learn a random decision forest which is then able, using depth differences between pairs of pixels in the depth-map, to establish for each body pixel to which body part it belongs. Schwarz et al (2012) proposed another method to find the skeleton of a person filmed with a depth-camera, but which does not require pre-training on a large database. They compute geodesic distances of each point of the body part to the gravity center. Knowing the standard structure of a human body, they estimate locations of the body joints. Another group of methods consists in finding body parts in depth-map without using any global information on the user's posture. Migniot and Ababsa (2013) use particles filtering with a top-view depth-camera to determine the position of a top human body. However, detection of hands in depth-videos is still challenging because of the usually rather low resolution of these sensors: only hands which are close enough to the camera can be easily segmented. Chen et al (2011) track the hands' location and segment them using a region-growing algorithm. Hamester et al (2013) detect hands in depth images based on Fourier descriptors of contours classified using a SVM. Joo et al (2014) use boosting of depth-difference features for detecting hands in depth images.

2.2.3 Machine-learning algorithm for classification of gestures

Several machine-learning techniques have been used in order to train a system to recognize human gestures. SVMs (Support Vector Machines), HMMs (Hidden Markov Models), and DTW (Dynamic Time Warping) have been widely used.

SVMs enable to optimize the separation boundaries between different classes in a feature space. Ke et al (2007), Bregonzio et al (2009) and Schuldt et al (2004) used SVMs to recognize actions in video.

Instead of using a fixed-size temporal window represented as a vector, HMMs can process inputs as a flow of successive values. Furthermore, they are able to recognize gestures independently from their temporal duration. Yamato et al (1992) used HMMs to recognize gestures in video. Xia et al (2012) recognized gestures using the skeletons extracted from depth-video with the method of (Shotton et al, 2011). Zhu and Pun (2012) also used depth-images and HMMs to recognize gestures. They track the locations of hands, and use their trajectories to recognize the gestures performed. Calinon and Billard (2004) used HMMs to learn gesture from demonstration. Aarno and Kragic (2008) proposed a Layered Hidden Markov Model (LHMM) to model human skills and classify motions into basic action primitives.

DTW is actually a method for time-series alignment and similarity measure. For gesture recognition, it is generally used first for selecting for each class a single most representative template gesture. DTW similarities with these templates can then be combined with any similarity-based classification algorithm (a simple Nearest Neighbor method in many case) for predicting class of an unlabbeled gesture. DTW has been used for instance by (Liu et al, 2009) to recognize actions based on output of accelerometers worn by users. Sempena et al (2011) similarly recognize actions with DTW, but applied to 3D joint angles time evolutions estimated by Kinect built-in skeletization. Reyes et al (2011) have shown that recognition performance by this method can be greatly improved by weighting differently each joint angle depending on its impact on executed gesture.

Other methods are also used. Luo et al (2013) classified actions with a Bag-of-Visual-Words framework. More recently, deep Convolutional Neural Networks approach was adapted to recognize actions in depth-maps: Wang et al (2016) used weighted hierarchical depth mo-

tion maps and three-channel deep convolutional neural networks to recognize actions with a small training dataset.

2.3 Our previous work

We have been working since 2012 on technical gesture recognition for collaborative robotics in factories. All our research is conducted on *real prototype "cells" of factory collaborative robotics* developed by french automaker PSA (see Acknowledgements). After a feasibility study using inertial sensors worn by operators (Coupeté et al, 2014), we have conducted a first experimentation of a less intrusive approach: using only a top-viewing depth-camera for capture of gestures (Coupeté et al, 2015). We have then highlighted in (Coupeté et al, 2016b) the significant recognition rate improvement achievable by complementing gesture capture from depth-camera with data from inertial sensors *placed on tools*. Finally in (Coupeté et al, 2016a) we began investigating the multi-users issue, and proposed a simple but efficient way of adapting our gesture recognition module to new operators.

In this paper, we put together all our methods and algorithms recalled above as a proposed generic methodology and pipeline for technical gesture recognition. Furthermore, we compare several feature sets (hands positions only vs. idem + arms postures, etc...), which we had not done in our previous work, and show that best results are obtained with descriptors related only to the *effective* part of gestures (i.e. hands movements). We also conduct new comparative study on user adaptation providing new results suggesting that enriching a multi-users training set can lead to higher precision than using a separate training dataset for each operator.

3 Our Human-Robot Collaboration prototype

We work on a real-life scenario where the worker and the robot share the same workspace and cooperate. The task is inspired from the assembly of motor hoses on production-line supplies preparation. Presently, the assembly process of motor hoses has some drawbacks: the worker has to find the appropriate parts of motor hoses among other motor parts, which is a lack of time and increase the cognitive load of the worker. In our set-up, a dual-arm robot and the worker are facing each other, with a table separating them, see Figure 1. More details on this *real* prototype are given below, and in (Coupeté et al, 2015) and (Coupeté et al, 2016b).

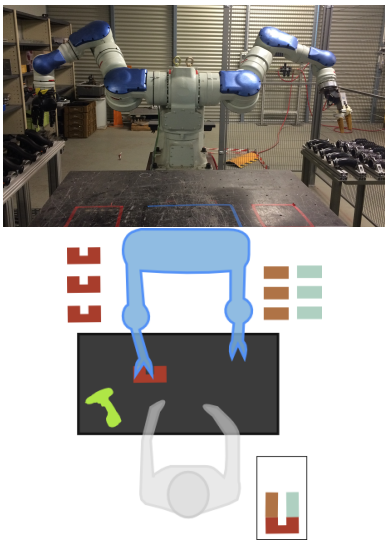

Fig. 1 On top, our human-robot collaboration prototype. On bottom, schematic description of our use-case: an operator is standing in front of a table, taking and assembling parts that are "handed" to him/her by a robot placed on the opposite side of the table.

On an assembly-line, the mounting operations must be executed quickly through a rather strictly-defined succession of elementary and standardized sub-tasks. To ensure a *natural* human-robot collaboration, the robot has to perform its actions according to the task that the operator is currently executing, in order to be useful at the right time, without delaying the worker. In our use-case, the assembling of motor hoses requires the worker to take two hose parts respectively from left and right claw of the robot, join them, screw them, take a third part from the right claw, join it, screw it, and finally place the mounted motor hose in a box. The only actions performed by the robot are giving a piece with the right claw and giving a piece with the left claw. The order of these sub-tasks and how the robot and the operator should be coordinated is presented Figure 2. Such an analysis of the human-robot collaborative work is essential to determine the gesture types that the robot needs to recognize.

In order for the robot to be properly synchronized with the human, it should be able to recognize several gesture classes, that can be deduced from Figure 2. The first two are "to take a piece in the right claw" and "to take a piece in the left claw". The operator can screw after the first gesture "to assemble", or can chose to screw later during the last assembly sub-task. Therefore, the robot should be able to recognize "to assemble" and "to screw", so as to give at the correct moment the third motor piece with its right arm. Finally, at the end of the assembly task, the operator puts the assembled piece in a box, so it is interesting to rec-

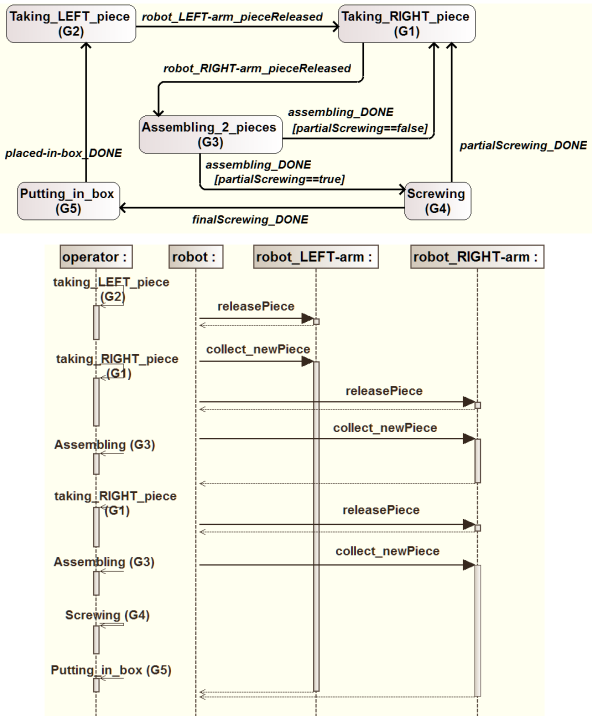

Fig. 2 Analysis of required coordination between Human and Robot: on top, state-transition diagram for the operator tasks; on bottom, sequence diagram of operator-robot interactions.

ognize this gesture in order to understand that a cycle has just ended.

The set of gestures classes to be recognized by our system is therefore rather straightforwardly deduced from above-mentioned sub-tasks as:

1. to take a motor hose part in the robot right claw (G1)
2. to take a motor hose part in the robot left claw (G2)
3. to join two parts of the motor hose (G3)
4. to screw (G4)
5. to put the final motor hose in a box (G5)

Note that in this set-up, the operator chooses the pace during the execution of his sub-tasks, and the robot adapts to it.

4 Methodology for recognition of technical gestures

In this section, we detail our end-to-end methodology for *online* recognition of technical gestures *in real-time*. In the first part 4.1, we present our pipeline (improved and more general than our first versions already presented in (Coupeté et al, 2015) and (Coupeté et al, 2016b)) to recognize gestures, from extraction of features to the classification algorithm. In part 4.2 we describe the two criteria we use to evaluate our gesture recognition system. In part 4.3, we explain how we equipped the scene with an inertial sensor on a tool, and how we refine output from gesture recognition by taking into account the tool-movement information. Finally, in part 4.4, we propose an approach for *adapting to a new user* our system, by limited enrichment of the training database.

4.1 Gesture recognition pipeline

In order to capture the gestures of the operator, we decided to use non-intrusive sensors, for avoiding any discomfort to the operators. Moreover, we want to monitor relative positions of the operator and the robot, while capturing all the operator's movement without potential occlusions. For these reasons, we chose to use a *depth-camera*, filming with a *top-view* for capturing the scene without occlusion. Note that visits on real assembly-lines, and preliminary work on another prototype use-case (door-elements mounting on a continuous line) have convinced us that this choice of sensor type and viewpoint configuration is transferable to most future human-robot collaborative assembly areas. In this section, we explain how we extract body-movement information from the depth-camera.

4.1.1 Posture estimation from depth images

A depth-camera provides information about *3D geometry* of the scene: the value of each pixel corresponds to the distance between the camera and the filmed object to which the pixel belongs.

In many related work on gesture recognition using depth-cameras, input features are simply the successive states of the global human skeleton posture estimation provided by APIs of Kinect for horizontal viewpoint. In our approach using a vertical viewpoint from the top, it was not possible. We therefore needed to extract upper-body (in particular hands) movements from the raw depth-video. We make the assumption that, from top viewpoint, the farthest upper-body parts from the top of the head, using a geodesic measurement, are the two hands. Based on this hypothesis, we have designed an algorithm to locate the operators' hands and estimate arms' postures in the depth-map.

Our algorithm, inspired but significantly modified from (Schwarz et al, 2012) (in which Schwarz et al. estimate global posture, but *only for facing horizontal viewpoint*), is based on estimation of geodesics on body 3D surface. First, we create a 2D graph of the upper-body of the person filmed. Each pixel of this

graph are connected with its eight neighbors. We associate for each connection a weight equal to the depth difference between the two pixels, i.e. the difference of the two pixels values. Then, we use Dijkstra algorithm (Dijkstra, 1959) to compute the geodesic distance between each pixel of the upper-body and the top of the head. We are thereby able to detect the two hands positions, and also obtain the geodesically shortest paths between each hand and the top of the head's (which can be used as approximations of arms' postures). Figure 3 illustrates this algorithm, and we refer readers to (Coupeté et al, 2015) for further details on our algorithm. One advantage of this approach for localization of hands is that it is relatively immune to hand occlusion: even when one hand is occasionally hidden from the depth camera (e.g. by another arm), the forearm and arm for this hand are generally still visible, so the geodesic from the head is still properly found and just stops around wrist instead of hand; therefore with this method, hand occlusion leads to only slightly erroneous hand position rather than to absence of information in the data stream.

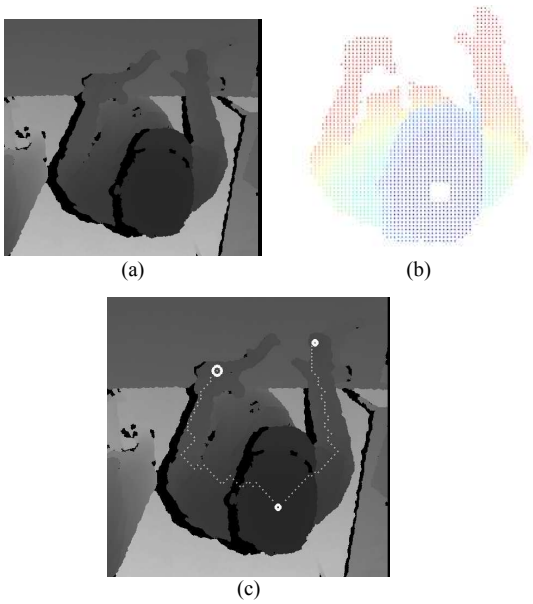

Fig. 3 Our hands localization and upper-body posture estimation algorithm. (a): depth-map from the camera filming an operator with a top-view; (b): geodesic distance for each pixel of the upper-body to the head's top; (c): estimation of the head and two hands locations, plus the geodesically-shortest paths between hands and the head's top (all produced by our algorithm).

4.1.2 Choice of features

In most machine-learning and pattern-recognition tasks, the attainable classification acuracy is strongly depending on the choice of features extracted from raw data and fed into the algorithm. In gesture recognition, it is rather natural and often adopted to use the estimated movements of body parts and joints as features. However, not all of them are equally important, depending on what gesture types should be recognized. Furthermore, it is well-known that inclusion of irrelevant features can reduce recognition rate, either by just adding "noise" to the machine-learning input, or worse by introducing spurious correlations. It is therefore highly recommended to either perform preliminary features selection, or at least to compare recognition performances attained with various sets of human-body related features.

In our case, as exposed above, our dedicated depth-image processing algorithm provides as output:

- 3D location of the head's top;
- estimated 3D locations of the two hands;
- the two 3D geodesically-shortest paths from head to each hand (providing rough approximations of approximations of arms' postures).

We therefore test (which we had not done in our previous works) five different sets of features, listed in Table 1 and illustrated on Figure 4. They all contain 3D locations of the two hands, completed with varying number of other upper-body posture information.

Table 1 Definitions of the five sets of features compared.

featureSet 1	15 samples of each shortest path + head and two hands 3D locations
featureSet 2	7 samples of each shortest path + head and two hands 3D locations
featureSet 3	3 samples of each shortest path + head and two hands 3D locations
featureSet 4	head and two hands 3D locations
featureSet 5	two hands 3D locations

4.1.3 Gesture classification algorithm

To classify the technical gestures performed, we have chosen to use *discrete* Hidden Markov Models (HMM). They are probabilistic models for classification of sequential discrete data. Given a continuous-valued vector of features deduced from estimated top-viewed posture (see part 4.1.2), we first need to quantize these data

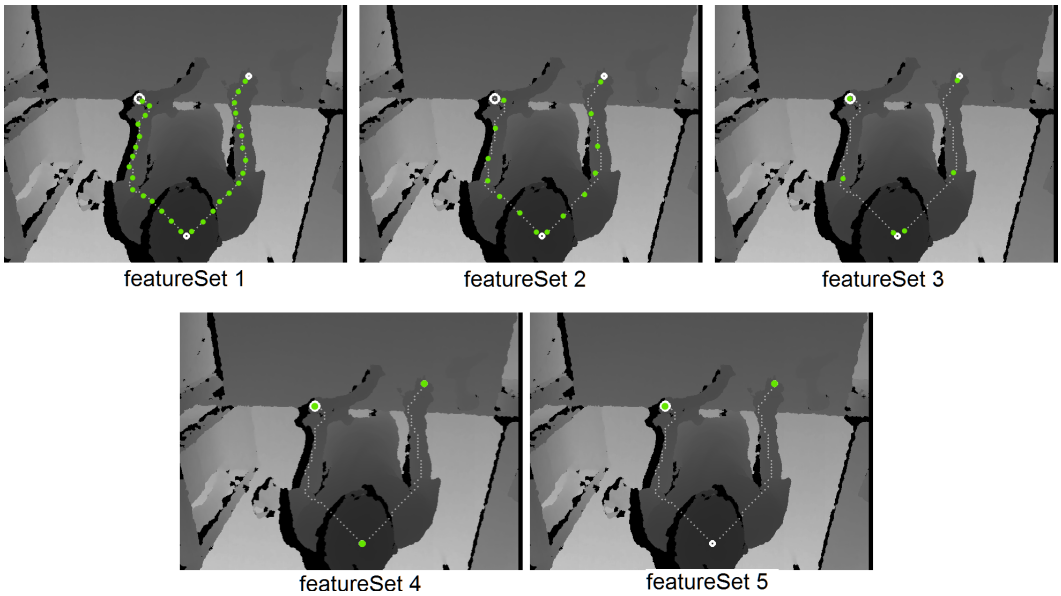

Fig. 4 Illustration of our five sets of features tested and compared for recognition of technical gestures (see 4.1.2 for their definition).

in order to obtain *discrete-valued* observations. For this step, we use K-Means clustering. This method aims at partitioning observations into a fixed number K of clusters. Each observation belongs to the cluster with the nearest centroïd. For our study, as already described in (Coupeté et al, 2015) and (Coupeté et al, 2016b), we partition all computed posture-estimation feature vectors into K clusters, so each cluster corresponds to an approximate top-viewed posture. After clustering, a gesture is represented as a temporal sequence of cluster IDs, corresponding to a sequence of approximate postures.

We use these quantized data to train our discrete HMMs, one HMM for each gesture class. For recognition, each feature vector extracted from depth-image is quantized by the previously learned K-Means, and the obtained labels are afterwards used as input for the discrete HMMs to determine which gesture is currently being performed. The recognized gesture is the one associated to the HMM which has the highest probability to have generated the observations. To train our HMMs, we use the Baum-Welch algorithm, and for the recognition we use the Forward algorithm. They are both implemented in the GRT[1] open library. Figure 5 illustrates our methodology.

4.1.4 Online gesture recognition in real-time

We want to *continuously* recognize gestures *in real-time* while the operator is working, performing the technical gestures one after the other naturally (i.e without any pause between successive gestures). To this end, we use a temporal sliding window of length T. Using the Forward algorithm, we compute the likelihood for each HMM to have produced the T last observations. To filter out transient errors, we finally output as recognized gesture class the one which has been the most recognized during the 10 last positions of the sliding temporal window.

To evaluate the performance of our real-time recognition system, we use the five standard metrics listed below:

$$\overline{R} = \frac{\#(gestures\ correctly\ recognized)}{\#(gestures\ performed)} \tag{1}$$

$$R_i = \frac{\#(gestures\ i\ correctly\ recognized)}{\#(gestures\ i\ performed)} \tag{2}$$

$$\overline{P} = \frac{\#(gestures\ correctly\ recognized)}{\#(gestures\ classified)} \tag{3}$$

$$P_i = \frac{\#(gestures\ i\ correctly\ recognized)}{\#(gestures\ classified\ i)} \tag{4}$$

$$F = 2\frac{\overline{P} \times \overline{R}}{\overline{P} + \overline{R}} \tag{5}$$

in which:

- $\#(gestures\ performed)$ represents the total number of gestures of all classes performed by the operators
- $\#(gestures\ i\ performed)$ represents the number of gestures of class i performed by the operators
- $\#(gestures\ correctly\ recognized)$ represents the number of gestures correctly recognized by our system.

[1] http://nickgillian.com/grt/

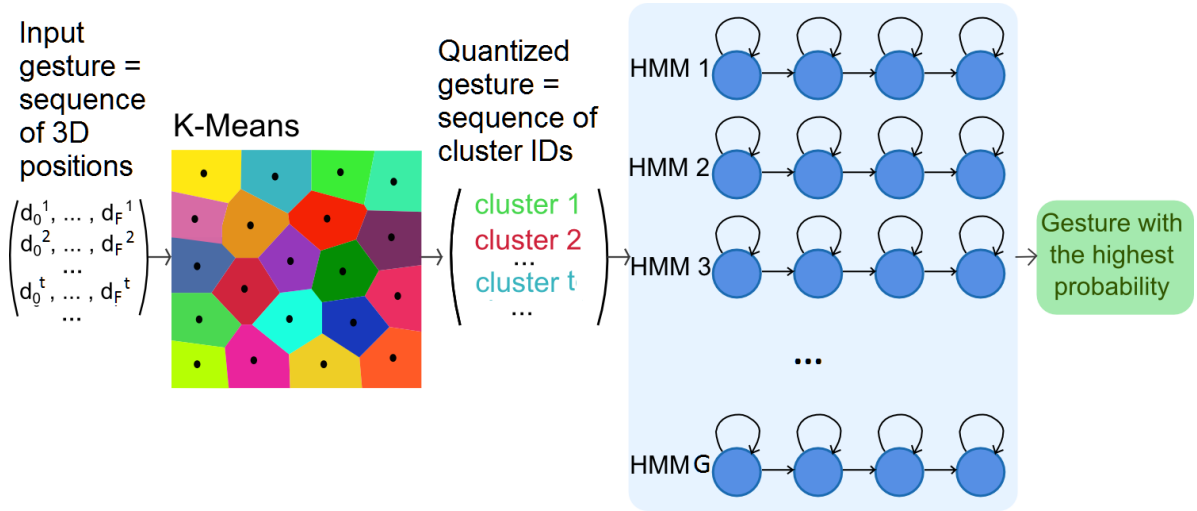

Fig. 5 Gesture recognition pipeline: input gesture (left) is a temporal sequence of feature vectors of same dimension F; each continuous-valued feature vector is quantized by K-means into a *discrete-valued "approximate posture" label* (middle); the obtained temporal sequence of successive posture labels is fed one after the other into G discrete HMM (1 per gesture class); for each time-step, our system outputs the most probable current gesture class, by selecting the HMM which has current maximum likelihood.

- #(*gestures i correctly recognized*) represents the number of gestures of class i correctly recognized
- #(*gestures classified i*) represents the number of gestures classified with the label i.
- #(*gestures classified*) is equal to the sum of all the value of #(*gestes classified i*) among all the classes

The *average recall* $\overline{R}$ provides a global information on the capacity of our system to detect the gestures. The values R_i detail the separate detection ability for each class of gestures. The values P_i indicate the accuracy of our system when it outputs the corresponding gesture class ID. The *average precision* $\overline{P}$ is the average of all precisions, P_i. Finally, the F-score F is the harmonic mean of average precision and average recall, and provides a global recognition performance index.

4.2 Evaluation criteria

To evaluate our system of gesture recognition, we use two criteria. The first one, called *jackknife*, estimates the future performance of our system for a new user, from whom no gesture was used to learn K-Means and train HMMs. Our second criterion, called 80%-20%, estimates performances of our system for users of whom example gestures are included in the training set. These two criteria are illustrated on Figure 6.

4.2.1 Jackknife

Our database contains recorded technical gesture examples from N operators. To evaluate our system for an unknown user, we train it with a database composed of gesture examples from $N-1$ operators, and estimate recognition rate on gesture examples performed *only* by the last operator (not included in training set). We test all possible combinations of $N-1$ operators for training and 1 operator for recognition estimation. This evaluation criterion is illustrated on Figure 6(a).

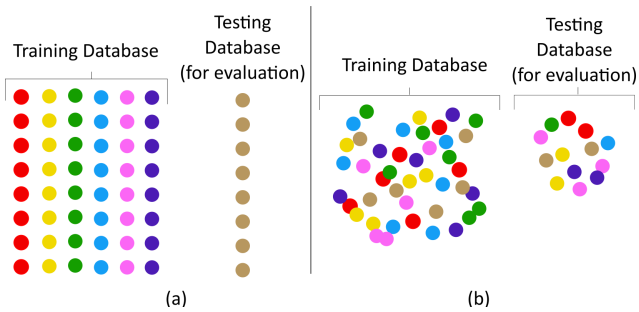

Fig. 6 Illustration of our two evaluation criteria. Each color represents an operator, and each dot a gesture example. (a): Jackknife, (b): 80%-20%

4.2.2 80% - 20%

For this evaluation criterion, we randomly divide our database in two parts. The first part is used for training and contains 80% of all gestures by all operators in our database. The second part is used for testing recognition, and is composed of the remaining 20% of our database. This evaluation criterion is illustrated on Figure 6(b). The main difference with the *jackknife* is that with the 80%-20% criterion, the system uses exam-

ples of gestures from the same operators in both training and testing databases, so it estimates recognition performance for "known" operators, i.e. included in the training set.

4.3 Use of inertial sensors *placed on tools*

To get more information on executed gestures, it can be interesting to equip the scene with additional sensors. In particular, valuable and complementary information can be obtained by *placing inertial sensors on the tools* that are used by the operator. Thus, as already reported in our previous work (Coupeté et al, 2016b), we have put an inertial sensor on the screwing-gun[2] . We use this additional data source with a "late-fusion" scheme: output of vision-based HMMs are first computed, and movement information from the tool is used only afterwards to deduce the final gesture classification result. We chose the "late-fusion" method because these data will only be used to distinguish one particular gesture class ("screwing") against another one ("assembling").

The screwing-gun is supposed to move only when the worker is using it to screw together two parts of motor hose. There is a conflict with the result of the HMMs classification in two cases:

- case 1: when gesture G4 ("screwing") is recognized, while the screwing gun does **not** move
- case 2: when a gesture which is not G4 is recognized, while the screwing gun did move

For the first case, if we suppose that the inertial sensor cannot be broken, it is not possible to screw without moving the screwing-gun. Therefore, if the likelihood of the HMM for the gesture "to screw" is above a threshold, we decide that this gesture has been executed, otherwise no gesture is recognized (zero output).

For the second case, it is possible that the screwing-gun moved without being used, if the worker wants to move it from one side of the table to another for example. In this case we also look at the output likelihood of the HMM matching with the gesture "to screw". If this likelihood is above a threshold, we replace the gesture previously recognized by "to screw", otherwise we keep the gesture associated to the HMM with the highest likelihood.

With this method, we want to make our system more robust by correcting confusion errors that can easily occur between rather similar gesture classes.

[2] Note that in modern factories, many tools such as screwing guns are actually connected to the assembly-line information system, so that binary information such as "moving" or "in use" can be readily available even without having to place inertial sensors on them.

4.4 Adaptation to a new user

As already unveiled in (Coupeté et al, 2016a), we studied the adaptation to a new user of our system. For this purpose, we experiment adaptation of the training dataset to this new user. We think that it is quite feasible in practice, and worth considering, when an operator learns to perform a new human-robot collaborative task, to record several repetitions of his gestures while he is experimenting his new task. We could have tried to apply an incremental learning algorithm: starting from the HMMs pre-trained on other users and fine-tune them with gesture examples from the new user. However, because in our case HMM training is rather fast, we decided to proceed by re-training from scratch on training database enhanced by addition of some gesture examples by the new user. As already reported in our previous work (Coupeté et al, 2016a), we also investigate the impact on recognition performance of the number of gesture examples from the new user added to original multi-users database. Figure 7 illustrates our methodology. Furthermore, in our final application, it could be possible to switch between user-specific gesture classifiers depending on the identity of current operator. We therefore perform and present here a new evaluation to compare with performance of classifiers trained only on other gesture examples of the same new user.

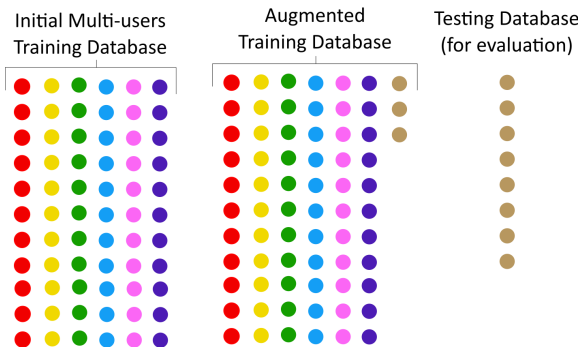

Fig. 7 Our method for adaptation to a new user: the initial multi-users training set is complemented by a few gesture examples of the new user, and retrained "from scratch". Evaluation is performed only on independant gesture examples of the same new user.

To evaluate this method, we add to the previous database a growing number of sets of gesture examples recorded from the new operator. One set is composed of one gesture of each class.

As for the *jackknife* criterion, we test all possible combinations of gesture examples from $N-1$ operators $+\ \epsilon$ gesture examples of the last operator to create

the training database, and remaining gestures examples by the last operator to create the test database. Also, to avoid potential bias due to varying size of training set, we take care of maintaining a constant number of gesture examples in our databases (training and test), whatever the added number of new user gestures' sets.

5 Experiments and results

In this section, we present our results. In a first part, 5.1, we explain how we recorded the gestures to create our dataset of examples. In a second part, 5.2, we present our study to choose optimal set of features describing technical gestures. Afterwards, we provide and justify our choices of parameters (part 5.3), and then present our *online* gesture recognition results (part 5.4). Finally, we provide and analyze our gesture recognition performances after adapting our system to a new user by training set modification (part 5.5); those results are also compared with performance attainable when training our system with gestures from only the new user (part 5.5.2).

5.1 Data acquisition protocol

To have a sufficiently large dataset for testing our method, we recorded 13 "naïve" operators (among which 2 women and 11 men) aged from 25 to 60 years old (47 years old on average). Each operator has executed between 20 and 25 assembly tasks. For each assembly, between 7 and 8 successive gestures are performed by the operator. Note that operators did not have any prior knowledge or experience on the task: they were only shown how to assemble pieces together, and told that the robot would handle pieces to them, but absolutely no instruction was given to them on detailed way to execute the technical gestures; this implies that operators were actually performing the assembly task for the first time during recording, thus increasing variability even between cycles executed by same operator.

5.2 Comparison of feature sets

As explained in 4.1.2, instead of using directly as features all the body posture information that our depth-image processing algorithm provides, we try and evaluate five different sets of features (see Table 1 for their definitions and Figure 4 for their illustrations).

Table 2 shows the results obtained with these different sets of features, which is a new finding not investigated in our previous works. We can observe that

Table 2 Rates of correct gesture recognition obtained depending on set of features. Recognition on isolated gestures, *jackknife* criterion.

15 samples head location hands locations	**7 samples head location hands locations**	**3 samples head location hands locations**	**head location hands locations**	**hands location**
65%	**70%**	**72%**	**74%**	**79%**

best result of correct gesture recognition, 79%, are obtained when we use *only the two hands 3D locations*. When we add information which are not directly linked with the *effective* part of gestures, the recognition rates decrease. Indeed, samples from the shortest paths and head location provide information about the operator's posture, but they can vary significantly from one operator to another, and even between several executions of the same gesture by the same user. Furthermore, this finding is coherent with the results of (Chen et al, 2015) in which very good recognition rates using only hand movements as features are reported, for gestures of an operator in a set-up similar to ours.

For the rest of presented results, we use as features only the set of two hands 3D locations.

5.3 Gesture classification algorithm parameters

As described part 4.1.3, we use a combination of K-Means and discrete HMMs to learn and recognize technical gestures. Both these algorithms have parameters that we must determine: the number of clusters for K-Means, and the number of hidden states for the HMMs. We have tested different combinations of parameters (which we had not reported in our previous publications) to determine and choose the values providing the best results. These tests are conducted on isolated gestures, i.e. gestures which are already segmented, and using jackknife criterion. Results are presented in Table 3.

We can observe that, when the number K of K-means clusters increases, the rate of correct gesture recognition gets clearly better, until K reaches 20 or 25. This was somewhat expected because more clusters implies a finer discretized description of hands postures, allowing a better distinction between different classes of gestures; and conversely when quantization is sufficiently fine, further increase of K cannot bring more improvement. As highlighted in Table 3, the recognition rate reaches a maximum of 82% of correct recognitions

Table 3 Correct gesture recognition rates as a function of number K of K-means clusters, and number S of HMM states.

		Number S of HMM states					
		5	**7**	**10**	**12**	**15**	**20**
Number K of K-means clusters	**10**	74%	75%	73%	72%	74%	73%
	15	76%	78%	78%	79%	78%	79%
	20	77%	80%	77%	78%	79%	78%
	25	76%	77%	79%	**82%**	81%	80%
	30	77%	78%	78%	80%	80%	79%

for $K = 25$ clusters and using HMMs with $S = 12$ hidden states. For the rest of this study, we use 25 clusters for K-means and 12 hidden states for HMMs as parameters of our gesture classification algorithm.

5.4 Online recognition perfomances

We need to recognize gestures *while they are performed by the operator*, and ideally even before they are finished.

Figure 8 illustrates two examples of output by our online continuous gesture recognition, each one during a complete assembly task executed by an operator. The blue line and the colors on the background represent the ground truth, i.e. the gestures which are currently performed by the operator. The red line represents the *real-time* output of our system.

On top of Figure 8, during the execution of the first gesture 1 "take a motor hose part in the robot right claw", from 0 to 2 seconds, our system is still recognizing the previous gesture 5, "to put the assembled piece in the box", but finally recognizes gesture 1 roughly 0.5 seconds before the end of its execution. All the following gestures are correctly recognized before they end (most of the time between 0.5 and 2 seconds in advance). During this assembly, our system wrongly recognizes gesture 3, "to join two parts of the motor hose". During this time the operator has his two hands in front of him, while waiting for the robot to bring him the next motor piece. This posture is similar to the one observed during execution of gesture 3, this is why we can observe this mistake.

On bottom of Figure 8, one can also observe that our system correctly recognizes technical gestures performed by the operator. In this case, it can be noticed that our system sometimes outputs zero instead of a gesture class ID. This occurs when current gesture is not well-enough recognized, and our system returns a zero value, rather than risking to output a false recognition.

In the two next sections, we present our results of online gesture recognition evaluated with our two criteria, jackknife and 80%-20%.

5.4.1 Jackknife

We first evaluate with jackknife criterion the result of our system continuously recognizing gestures in real-time. The performances, depending on duration of the temporal sliding window, are presented in Table 4 for recall rates and in Table 5 for precision rates.

For recall, best results are obtained with medium duration of temporal sliding windows: 1 second and 1.5 seconds. For both window lengthes, we have a $\overline{R}$ score of 77%. With the window duration of 0.5 second the $\overline{R}$ score is lower, 65%. This window is not long enough to contain sufficient information to correctly recognize the technical gestures. For the longest sliding window, 2 seconds, we obtain a $\overline{R}$ score of 74%. With this duration, short gestures can be drowned with other information, inhibiting their correct recognition.

Table 4 Recall for *online continuous* recognition of technical gestures using data from the depth-camera and from inertial sensor on the screwing-gun. Evaluation criterion: jackknife, number of states: 12, number of clusters: 25

Length of temporal sliding window	**Recall**					
	$\overline{R}$	R_1	R_2	R_3	R_4	R_5
0.5 s	**65%**	54%	62%	38%	94%	83%
1 s	**77%**	68%	79%	60%	95%	87%
1.5 s	**77%**	67%	75%	64%	95%	84%
2 s	**74%**	65%	64%	64%	95%	82%

We observe for precision a trend similar to the one obtained for recall. Better results are obtained with longer temporal sliding windows, and the best one is obtained with a window duration of 1 second, reaching a $\overline{P}$ score of 84%.

Table 5 Precision for *online continuous* recognition of technical gestures using data from the depth-camera and from inertial sensor on the screwing-gun. Evaluation criterion: jackknife, number of states: 12, number of clusters: 25

Length of temporal sliding window	**Precision**					
	$\overline{P}$	P_1	P_2	P_3	P_4	P_5
0.5 s	**79%**	57%	81%	71%	92%	80%
1 s	**84%**	68%	87%	79%	92%	86%
1.5 s	**83%**	67%	87%	76%	92%	84%
2 s	**83%**	53%	76%	77%	98%	90%

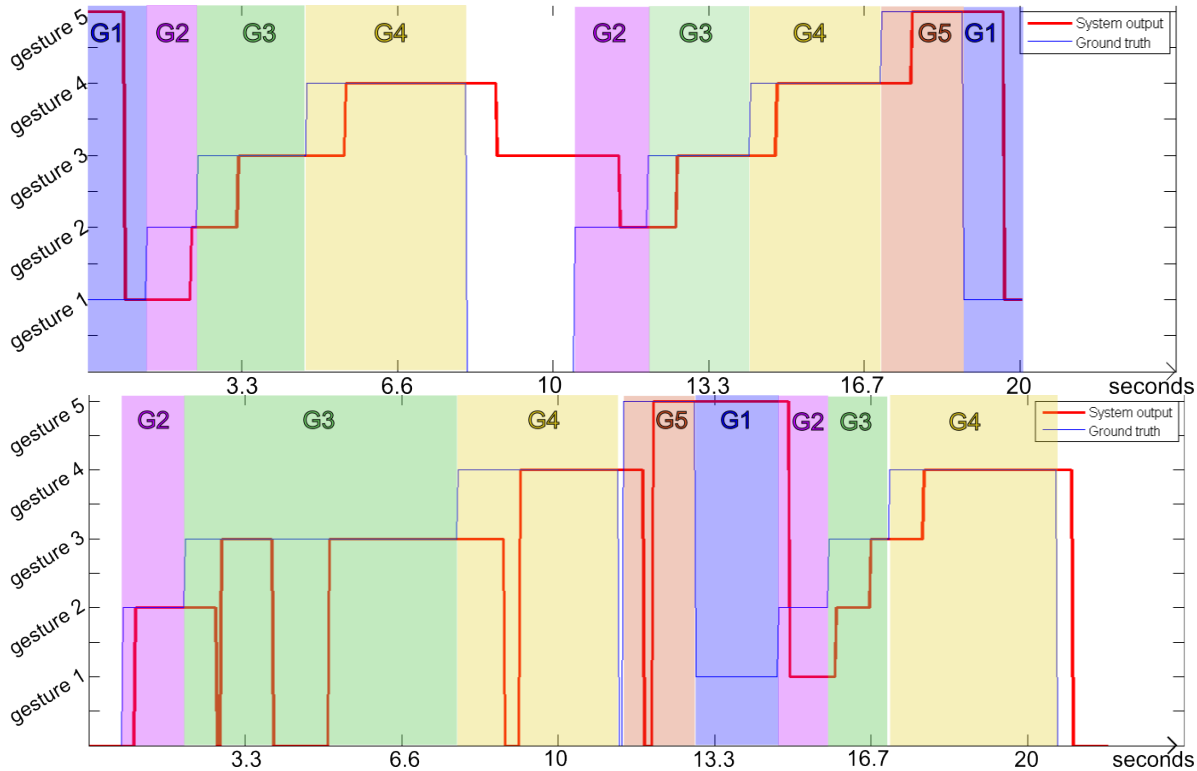

Fig. 8 Examples of online continuous gesture recognition. Blue line and colors on the background: ground truth, Red line: our system output. Note that, in order to avoid false recognitions, our system sometimes ouputs '0' (as can be seen on second example) instead of a gesture class when it is unsure about the type of currently executed gesture.

5.4.2 80% - 20%

We also evaluate our system using the 80%-20% criterion. The results are presented in Tables 6 and 7 for recall and precision respectively.

These results follow a trend similar to that observed with *jackknife* criterion. The best results of recall are obtained for medium-sized temporal sliding windows, with a duration of 1 second and 1.5 seconds. For these windows, we reach a recall of 85% and a precision of 82%. As expected, we can observe that recall is higher when using the 80%-20% criterion than with *jackknife*. Indeed, with the 80%-20% criterion, the system already *"knows"* the operator, i.e. some of *his* gesture examples were used to train the HMMs. These results motivated us to explore a method to adapt our system to a new user by modifying the training set.

5.4.3 Recognition delays

As can be seen on Figure 8 by comparing change instants of blue and red lines, our algorithm performs *early* recognition of gestures, in the sense that the operator's action is often correctly classified BEFORE the gesture is finished, and even sometimes a rather short time after it is initiated. We also have quantitatively

Table 6 Recall for *online continuous* recognition of technical gestures using data from the depth-camera and from inertial sensor on the screwing-gun. Evaluation criterion: 80%-20%, number of states: 12, number of clusters: 25

Length of temporal sliding window	Recall					
	$\overline{R}$	R_1	R_2	R_3	R_4	R_5
0.5 s	**80%**	38%	85%	76%	95%	77%
1 s	**85%**	44%	87%	87%	96%	80%
1.5 s	**85%**	55%	86%	80%	95%	86%
2 s	**81%**	64%	88%	80%	95%	81%

Table 7 Precision for *online continuous* recognition of technical gestures using data from the depth-camera and from inertial sensor on the screwing-gun. Evaluation criterion: 80%-20%, number of states: 12, number of clusters: 25

Length of temporal sliding window	Precision					
	$\overline{P}$	P_1	P_2	P_3	P_4	P_5
0.5 s	**74%**	68%	95%	55%	92%	89%
1 s	**80%**	70%	94%	75%	92%	89%
1.5 s	**82%**	70%	91%	77%	92%	89%
2 s	**73%**	67%	77%	76%	92%	89%

evaluated these delays between initiation of an action and the instant when it is correctly recognized by our system. As shown in Table 8, this delay is typically between 1 and 1.5 seconds, to be compared with gestures durations which vary between 1.5 s and 3 s; it is interesting to note that for gesture classes G3 and G4, recognition occurs on average respectively 0.9 s and 0.6 s *before* the end of the gesture.

Table 8 Time delay between gesture initiation and its recognition (averages and standard deviations, in seconds), compared to gesture average duration (in seconds)

Gesture class	mean gesture duration	*mean recognition delay*	*St.Dev.*
G1	1 s	1.1 s	0.4 s
G2	1.1 s	1.3 s	0.6 s
G3	2.5 s	1.6 s	1.2 s
G4	2 s	1.4 s	0.6 s
G5	1.7 s	1.6 s	0.6 s

5.4.4 Comparison with Dynamic Time Warping (DTW)

In order to assess if our particular recognition method (K-means+discrete-HMMs) has an important contribution to our final results, we have also conducted tests using DTW (Dynamic Time Warping) instead. As mentioned in section 2.2.3, DTW is a very commonly used technique for gesture recognition, so it provides a useful baseline result. As can be seen in Table 9 the recognition performance is much lower with DTW than with K-means+HMMs. This can be explained by the quite large intra-class variability of gesture execution in our application, because DTW models each gesture class by one single template, which makes it less suitable for our technical gestures. This hypothesis is confirmed by the fact that the drop of recognition rate for a new user (Jackknife criteria) compared to a "known" user (80%-20% criteria) is much higher with DTW (-10%) than with K-means+discrete-HMMs (-3%), which means our recognition method is clearly more robust to gestural variability.

5.5 Adaptation of gesture recognition to a new user

5.5.1 Adaptation of training dataset

We can observe on results presented above that our rates of correct gesture recognition are better when the system "knows" the user, i.e. was trained with a dataset

Table 9 Comparison of gesture recognition performance between DTW and K-means+discrete-HMMs): average F-score, average Precision and average Recall

Algo (evaluation criteria)	$\overline{F}$	$\overline{P}$	$\overline{R}$
DTW (Jacknife)	**39%**	**47%**	**34%**
DTW (80%-20%)	**50%**	**59%**	**44%**
K-means+HMMs (Jacknife)	**80%**	**83%**	**77%**
K-means+HMMs (80%-20%)	**83%**	**82%**	**85%**

containing at least some gesture examples performed by him. Indeed, we obtain better results with the 80%-20% criterion than with *jackknife*. This observation motivated us to adapt the training database to a new user, as explained in part 4.4.

Our approach consists in adding one or several sets of gesture examples executed by the new operator to the training database. For the comparison to be fair, we randomly remove gestures from the original multi-operators dataset when we add gestures by the new operator, in order to maintain the same size of training and testing datasets for all tests.

The results (using a 1 second long temporal sliding window) are compared on Table 10. Both recall and precision increase to reach 89%, when 15 sets of gesture examples have been added. Table data are plotted on top-left of Figure 9, showing recall precision and F-score as a function of the number of sets added in the training base.

Table 10 Precision and recall of technical gestures for online recognition, after an adaption of the training base with an increasing added number of gesture examples from the new user. Data from the depth-camera and inertial sensor on the screwing-gun.

	Number of sets of gesture added								
	1	**2**	**3**	**4**	**5**	**7**	**10**	**12**	**15**
$\overline{P}$ (%)	84	84	86	85	86	85	86	86	**89**
$\overline{R}$ (%)	84	84	87	87	86	87	88	88	**89**
F (%)	84	84	86	86	86	88	86	87	**89**

The maximum improvement is quite large (+12% of recall and +5% of precision, compared to the initial jackknife results in Tables 4 and 5). Interestingly, it appears that even when adding only a small number ($\leq$ 5) of gesture sets, the improvement is significant (+9% recall and +3% precision). It can also be seen on curves plotted on top-right of Figure 9 that the first 5 added sets bring significantly more improvement by set. The recall and precision continue to increase when more gesture sets are added, but the impact of each set

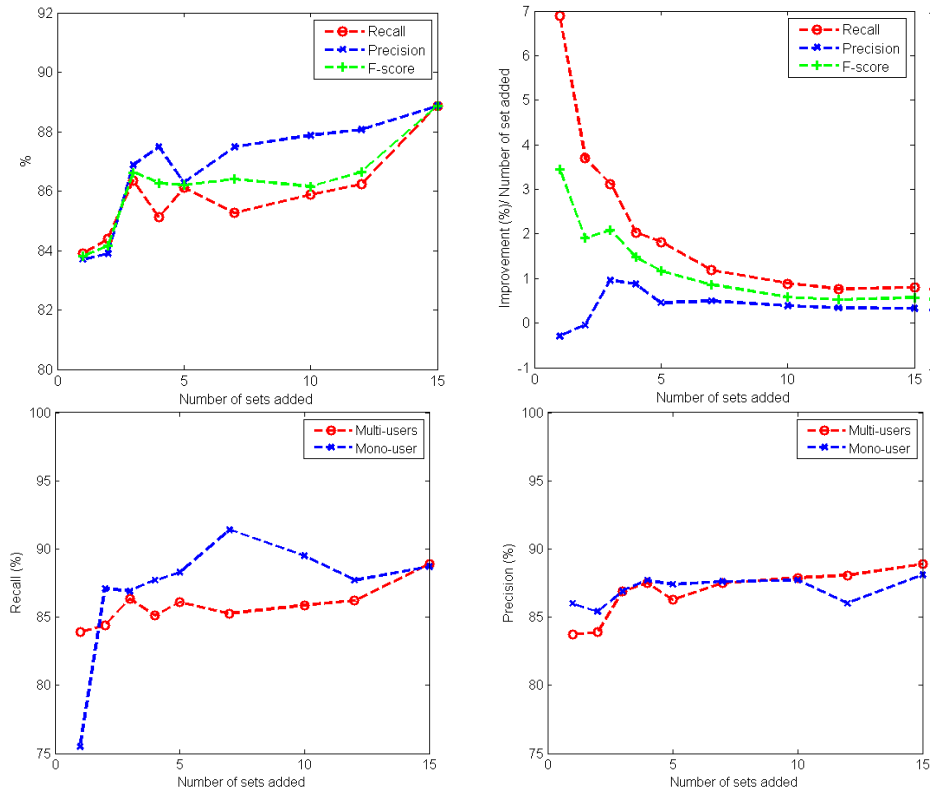

Fig. 9 Adaptation of gesture recognition to new user. On top, results obtained by training on multi-users database enriched by addition of some gesture examples from a new user (left: recall, precision and F-score as a function of the number of gesture sets added; right: improvement contribution brought by each new added set). On bottom, comparison of recall (on left) and precision (on right) between the performances attained by adding gestures example sets by new user to multi-users training database (red lines), and the result of training *with gesture examples ONLY by the new user* (blue lines).

decreases and converges around 1% of improvement for each new set added.

These observations show that adding to the training dataset a relatively small number of gesture examples from a new user can, after full retraining, significantly improves gesture recognition performances for this new user. Using operator-specific personalized gesture classifiers is therefore desirable, and easily feasible by retraining from scratch after very slight augmentation of an initial multi-users training database.

5.5.2 Comparison with training on mono-user datasets

Since we test a system adapted to a new user by modifying the training base, one can wonder what would be the performances of a system trained on gesture examples recorded *only* from this new user. We therefore also evaluate gesture recognition results, in case our system is trained and tested on databases composed only of gesture examples from the same operator. We conduct this evaluation, which is a new study compared to our previous work, for an increasing number of gestures in training set, the size of the test database being constant. Results are shown in Table 11. Not surprinsingly, recognition performances strongly increase when the number of gesture sets grows, particularly from 1 set (F-score = 80%) to 7 sets (F-score = 89%). Improvement is much slower for further addition of gesture sets.

Table 11 Precision, recall and F-score of technical gestures *online* recognition with *mono-user* training and testing databases. Data from the depth-camera and inertial sensor on the screwing-gun.

	Number of gestures sets in training								
	1	**2**	**3**	**4**	**5**	**7**	**10**	**12**	**15**
$\overline{P}$ (%)	86	85	87	88	87	88	88	86	**88**
$\overline{R}$ (%)	75	87	87	88	88	**91**	89	88	89
F (%)	80	86	87	88	88	89	89	87	88

Plots on bottom of Figure 9 compare recognition performances attained by using mono-user training dataset with those presented in 5.5.1, where a multi-users training database was enriched with a few gesture examples by the test user. For recall (graph on the bottom-left), results of the mono-user-trained system are glob-

ally better than those obtained from the multi-users adapted system; however recall rates become quite similar for higher number of gesture sets. For precision (bottom-right graph), results for the mono-user system and the multi-user adapted system are globally very close. However, *above ten sets, the precision from the system based on a multi-users adapted training base are better than those obtained with the mono-user system.*

These results of precision and recall suggest that a multi-users adapted system can be more robust than a mono-user system, which is a new finding. This could be explained by the fact that a multi-user adapted training set contains a greater quantity of ways to perform the same gesture than a same-size mono-user database. Hence the learnt HMMs are more general, allowing a better precision during recognition.

6 Conclusions and perspectives

In this study, we presented an experiment *on a real prototype* in which we continuously recognize, online and in real-time, technical gestures performed by operators on an assembly-line. This study highlights the feasibility to recognize technical gestures in such context *using only non-intrusing sensors.*

We use a depth-camera with a top-view to minimize possible occlusions on the collaborative task. We choose the gesture classes to be recognized so as to optimize coordination between the robot and operator.

We propose an algorithm for estimating upper-body posture (especially hands positions) using geodesic distances between upper-body pixels and the head's top. We highlight that features directly linked to the *effective part* of gestures (hands movements) lead to better recognition results than using user's upper-body global posture.

We show that our system can recognize technical gestures in real-time, even for users not included in training database examples. We also propose a method to adapt gesture classification to a new user, by moderate enrichment of the training set. We reach 91% recall and 88% precision during online multi-users gesture recognition.

Furthermore, we highlight that training on a dataset adapted to a new user by addition of rather few gesture examples can lead to better precision of gesture recognition than learning a totally user-specific classifier for each operator, trained with only his own example gestures. This could be due to the fact that a multi-users database includes more variability of gestures' execution, leading to more robustness.

As for perspectives, we currently work on handling parasite gestures, which can be performed by operators while they are working, but are not technical gestures. We also plan to investigate use of different lengthes of temporal sliding windows for each gesture class, to take into account their unequal average durations. It could also be interesting in a future work to analyze if there could be a relation between situation of actual hands positions 6D vector within clusters along the gesture trajectory, and success or failure of the gesture recognition by the HMMs.

Finally, since our gesture recognition methodology (choice of feature set, classification pipeline, adaptation to new user) is rather general, it could be used in application contexts other than manufacturing assembly-lines, for example in assistance and service collaborative robotics.

Acknowledgements This research benefited from the support of the Chair 'PSA Peugeot Citroën Robotics and Virtual Reality', led by MINES ParisTech and supported by PEUGEOT S.A. The partners of the Chair cannot be held accountable for the content of this paper, which engages the authors´ responsibility only.

References

Aarno D, Kragic D (2008) Motion intention recognition in robotassisted applications. Robotics and Autonomous Systems 56(8):692–705

Bannat A, Bautze T, Beetz M, Blume J, Diepold K, Ertelt C, Geiger F, Gmeiner T, Gyger T, Knoll A, Lau C, Lenz C, Ostgathe M, Reinhart G, Roesel W, Ruehr T, Schuboe A, Shea K, Stork genannt Wersborg I, Stork S, Tekouo W, Wallhoff F, Wiesbeck M, Zaeh MF (2011) Artificial Cognition in Production Systems. IEEE Transactions on Automation Science and Engineering 8(1):148–174

Biswas KK, Basu SK (2011) Gesture recognition using microsoft kinect. In: The 5th International Conference on Automation, Robotics and Applications, IEEE, pp 100–103

Bregonzio M, Gong S, Xiang T (2009) Recognising action as clouds of space-time interest points. In: 2009 IEEE Conference on Computer Vision and Pattern Recognition, IEEE, pp 1948–1955

Bulling A, Blanke U, Schiele B (2014) A tutorial on human activity recognition using body-worn inertial sensors. ACM Computing Surveys 46(3):1–33

Calinon S, Billard A (2004) Stochastic Gesture Production and Recognition Model for a Humanoid Robot. In: Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on, pp 2769–2774

Chen C, Jafari R, Kehtarnavaz N (2016) Fusion of depth, skeleton, and inertial data for human action recognition. In: 2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, pp 2712–2716

Chen CP, Chen YT, Lee PH, Tsai YP, Lei S (2011) Real-time hand tracking on depth images. In: 2011 Visual Communications and Image Processing (VCIP), IEEE, pp 1–4

Chen F, Zhong Q, Cannella F, Sekiyama K, Fukuda T (2015) Hand gesture modeling and recognition for human and robot interactive assembly using hidden markov models. International Journal of Advanced Robotic Systems 12(4):48

Chen L, Wei H, Ferryman J (2013) A survey of human motion analysis using depth imagery

Cherubini A, Passama R, Crosnier A, Lasnier A, Fraisse P (2016) Collaborative manufacturing with physical human–robot interaction. Robotics and Computer-Integrated Manufacturing pp 1–13

Corrales Ramón JA, García Gómez GJ, Torres Medina F, Perdereau V (2012) Cooperative tasks between humans and robots in industrial environments. InTech

Coupeté E, Manitsaris S, Moutarde F (2014) Real-time recognition of human gestures for collaborative robots on assembly-line. In: 3rd International Digital Human Modeling Symposium (DHM2014), Tokyo, Japan, p 7 p.

Coupeté E, Moutarde F, Manitsaris S (2015) Gesture Recognition Using a Depth Camera for Human Robot Collaboration on Assembly Lines. Procedia Manufacturing 3:518–525

Coupeté E, Moutarde F, Manitsaris S (2016a) A User-Adaptive Gesture Recognition System Applied to Human-Robot Collaboration in Factories. In: 3rd International Symposium On Movement and Computing (MOCO'16), Thessalonique, Greece

Coupeté E, Moutarde F, Manitsaris S, Hugues O (2016b) Recognition of Technical Gestures for Human-Robot Collaboration in Factories. In: The Ninth International Conference on Advances in Computer-Human Interactions, Venise, Italy

Dijkstra EW (1959) A note on two problems in connexion with graphs. Numerische Mathematik 1(1):269–271

Dollar P, Rabaud V, Cottrell G, Belongie S (2005) Behavior Recognition via Sparse Spatio-Temporal Features. In: 2005 IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance, IEEE, pp 65–72

Dong L, Wu J, Chen X (2007) A Body Activity Tracking System using Wearable Accelerometers. 2007 IEEE International Conference on Multimedia and Expo pp 1011–1014

Dragan AD, Bauman S, Forlizzi J, Srinivasa SS (2015) Effects of Robot Motion on Human-Robot Collaboration. In: Proceedings of the Tenth Annual ACM/IEEE International Conference on Human-Robot Interaction - HRI '15, ACM Press, New York, New York, USA, pp 51–58

Hägele M, Schaaf W, Helms E (2002) Robot Assistants at Manual Workplaces: Effective Co-operation and Safety Aspects. Proceedings of the 33rd ISR (International Symposium on Robotics) 7-11

Hamester D, Jirak D, Wermter S (2013) Improved estimation of hand postures using depth images. In: 2013 16th International Conference on Advanced Robotics (ICAR), pp 1–6

Hoffman G, Breazeal C (2007) Effects of anticipatory action on human-robot teamwork efficiency, fluency, and perception of team. In: Proceeding of the ACM/IEEE international conference on Human-robot interaction - HRI '07, ACM Press, New York, New York, USA, p 1

Joo SI, Weon SH, Choi HI (2014) Real-time depth-based hand detection and tracking. The Scientific World Journal

Junker H, Amft O, Lukowicz P, Tröster G (2008) Gesture spotting with body-worn inertial sensors to detect user activities. Pattern Recognition 41(6):2010–2024

Ke Y, Sukthankar R, Hebert M (2007) Spatio-temporal Shape and Flow Correlation for Action Recognition. In: 2007 IEEE Conference on Computer Vision and Pattern Recognition, IEEE, pp 1–8

Laptev I, Lindeberg T (2003) Space-time interest points. Proceedings Ninth IEEE International Conference on Computer Vision 1:432–439

Lenz C, Nair S, Rickert M, Knoll A, Rosel W, Gast J, Bannat A, Wallhoff F (2008) Joint-action for humans and industrial robots for assembly tasks. In: RO-MAN 2008 - The 17th IEEE International Symposium on Robot and Human Interactive Communication, IEEE, pp 130–135

Liu J, Zhong L, Wickramasuriya J, Vasudevan V (2009) uwave: Accelerometer-based personalized gesture recognition and its applications. Pervasive and Mobile Computing 5(6):657 – 675

Luo J, Wang W, Qi H (2013) Group Sparsity and Geometry Constrained Dictionary Learning for Action Recognition from Depth Maps. In: The IEEE International Conference on Computer Vision (ICCV), pp 1809–1816

Migniot C, Ababsa F (2013) 3d human tracking from depth cue in a buying behavior analysis context. In: 15th International Conference on Computer Analysis

of Images and Patterns (CAIP 2013), pp 482–489

Oikonomopoulos A, Patras I, Pantic M (2005) Spatiotemporal salient points for visual recognition of human actions. IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics) 36(3):710–719

Reyes M, Domínguez G, Escalera S (2011) Featureweighting in dynamic timewarping for gesture recognition in depth data. In: 2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops), pp 1182–1188

Rickert M, Foster ME, Giuliani M, By T, Panin G, Knoll A (2007) Integrating Language, Vision and Action for Human Robot Dialog Systems. In: Universal Access in Human-Computer Interaction. Ambient Interaction, Springer Berlin Heidelberg, Berlin, Heidelberg, pp 987–995

Schrempf OC, Hanebeck UD, Schmid AJ, Worn H (2005) A novel approach to proactive human-robot cooperation. In: ROMAN 2005. IEEE International Workshop on Robot and Human Interactive Communication, 2005., IEEE, pp 555–560

Schuldt C, Laptev I, Caputo B (2004) Recognizing human actions: a local SVM approach. In: Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004., IEEE, vol 3, pp 32–36 Vol.3

Schwarz LA, Mkhitaryan A, Mateus D, Navab N (2012) Human skeleton tracking from depth data using geodesic distances and optical flow. Image and Vision Computing 30(3):217–226

Sempena S, Maulidevi NU, Aryan PR (2011) Human action recognition using dynamic time warping. In: Electrical Engineering and Informatics (ICEEI), 2011 International Conference on, IEEE, pp 1–5

Shi J, Jimmerson G, Pearson T, Menassa R (2012) Levels of human and robot collaboration for automotive manufacturing. In: Proceedings of the Workshop on Performance Metrics for Intelligent Systems - PerMIS '12, ACM Press, New York, New York, USA, p 95

Shotton J, Sharp T, Kipman A, Fitzgibbon A, Finocchio M, Blake A, Cook M, Moore R (2011) Real-time Human Pose Recognition in Parts from Single Depth Images. In: Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition, IEEE Computer Society, Washington, DC, USA, CVPR '11, pp 1297–1304

Wang H, Ullah MM, Klaser A, Laptev I, Schmid C (2009) Evaluation of local spatio-temporal features for action recognition. In: Procedings of the British Machine Vision Conference 2009, British Machine Vision Association, pp 124.1–124.11

Wang P, Li W, Gao Z, Zhang J, Tang C, Ogunbona PO (2016) Action Recognition From Depth Maps Using Deep Convolutional Neural Networks. IEEE Transactions on Human-Machine Systems 46(4):498–509

Xia L, Chen CC, Aggarwal JK (2012) View invariant human action recognition using histograms of 3D joints. In: 2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, IEEE, pp 20–27

Yamato J, Ohya J, Ishii K (1992) Recognizing human action in time-sequential images using hidden Markov model. In: Proceedings 1992 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, IEEE Comput. Soc. Press, pp 379–385

Zhang H, Parker LE (2011) 4-Dimensional Local Spatio-Temporal Features for Human Activity Recognition. In: 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp 2044—-2049

Zhu HM, Pun CM (2012) Real-time Hand Gesture Recognition from Depth Image Sequences. 2012 Ninth International Conference on Computer Graphics, Imaging and Visualization pp 49–52