Learning Multirobot Joint Action Plans from Simultaneous Task Execution Demonstrations

Murilo Fernandes Martins
Dept. of Elec. and Electronic Engineering
Imperial College London
London, UK
murilo@ieee.org

Yiannis Demiris
Dept. of Elec. and Electronic Engineering
Imperial College London
London, UK
y.demiris@imperial.ac.uk

ABSTRACT

The central problem of designing intelligent robot systems which learn by demonstrations of desired behaviour has been largely studied within the field of robotics. Numerous architectures for action recognition and prediction of intent of a single teacher have been proposed. However, little work has been done addressing how a group of robots can learn by simultaneous demonstrations of multiple teachers.

This paper contributes a novel approach for learning multirobot joint action plans from unlabelled data. The robots firstly learn the demonstrated sequence of individual actions using the HAMMER architecture. Subsequently, the group behaviour is segmented over time and space by applying a spatio-temporal clustering algorithm.

The experimental results, in which humans teleoperated real robots during a search and rescue task deployment, successfully demonstrated the efficacy of combining action recognition at individual level with group behaviour segmentation, spotting the exact moment when robots must form coalitions to achieve the goal, thus yielding reasonable generation of multirobot joint action plans.

Categories and Subject Descriptors

I.2.9 [**Artificial Intelligence**]: Robotics

General Terms

Algorithms, Design, Experimentation

Keywords

Learning by Demonstration, Multirobot Systems, Spectral Clustering

1. INTRODUCTION

A substantial amount of studies in Multirobot Systems (MRS) addresses the potential applications of engaging multiple robots to collaboratively deploy complex tasks such as search and rescue, distributed mapping and exploration of unknown environments, as well as hazardous tasks and foraging – for an overview of the field, see [13]. Designing distributed intelligent systems, such as MRS, is a profitable technology which brings benefits such as flexibility, redundancy and robustness, among others.

Cite as: Learning Multirobot Joint Action Plans from Simultaneous Task Execution Demonstrations, M. F. Martins, Y. Demiris, *Proc. of 9th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2010)*, van der Hoek, Kaminka, Lespérance, Luck and Sen (eds.), May, 10–14, 2010, Toronto, Canada, pp. 931-938

Figure 1: The P3-AT mobile robots used in this paper, equipped with onboard computers, cameras, laser and sonar range sensors.

Similarly, a substantial amount of studies have proposed numerous approaches to robot Learning by Demonstration (LbD) – for a comprehensive review, see [1]. Equipping robots with the ability to understand the context in which they interact without the need of configuring or programming the robots is an extremely desired feature.

Regarding LbD, the methods which have been proposed are mostly focussed on a single teacher, single robot scenario. In [7], a single robot learnt a sequence of actions demonstrated by a single teacher. In [12], the authors presented an approach where a human acted both as a teacher and collaborator to a robot. The robot was able to match the predicted resultant state of the human's movements to the observed state of the environment based on its underlying capabilities. A supervised learning method was presented in [4] using gaussian mixture models, in which a four-legged robot was teleoperated during a navigation task.

Few studies addressed the prediction of intent in adversarial multiagent scenarios, such as the work of [3], in which group manoeuvres could be predicted based upon existing models of group formation. In the work of [5], multiple humanoid robots requested a teacher's demonstration when facing unfamiliar states. In [14], the problem of extracting group behaviour from observed coordinated manoeuvres of multiple agents along time was addressed by using a clustering algorithm. The method presented in [9] allowed a single robot to predict the intentions of 2 humans based on spatio-temporal relationships.

However, the challenge of designing an MRS system in which multiple robots learn group behaviour by observation

of multiple teachers concurrently executing a task was not addressed hitherto.

This paper presents a novel approach for LbD in MRS – the Multirobot Learning by Demonstration (MRLbD) – in which multiple robots are capable of learning task solution procedures (denominated as multirobot joint action plans) by observing the simultaneous execution of desired behaviour demonstrated by humans. This is achieved by firstly learning a demonstrated sequence of actions at single robot level, and subsequently applying a Spectral Clustering (SC) algorithm to segment the group behaviour. Lastly, a multirobot joint action plan is generated by combining the actions at single robot level with the segmented group behaviour, resulting in a sequence of individual actions for each robot, as well as joint actions that require coalition formation.

The remainder of this paper is organised as follows: Section 2 discusses the issues that must be addressed when designing an MRLbD system. In Section 3, the teleoperation platform, developed to allow remote control of real robots (pictured in Fig. 1) engaged in a realistic search and rescue activity, is detailed. This section also describes how the HAMMER architecture [7] and an implementation of the SC algorithm proposed by [14] were utilised to tackle the action recognition and group behaviour segmentation issues. Then, Section 4 describes the experimental tests carried out to demonstrate the multirobot plan generation, and Section 5 analyses the results obtained. Finally, Section 6 presents the conclusions and further work.

2. SYSTEM DESIGN ISSUES

The MRLbD architecture proposed in this paper is based upon a platform for robot teleoperation, which design was inspired by the work of [8] and [16], as well as the LbD architectures presented in [7], [9].

The design of any MRS encompasses common issues within this field of research. In particular, systems for robot teleoperation bring forth three central design issues, which are discussed in the following sections.

2.1 Human *vs.* robot-centred perception

Platforms for teleoperation usually provide restricted perception of the remote environment in which a robot is inserted. However, depending upon the application and the environment, the human can be strategically positioned in such a way that global, unrestricted observation is feasible.

The first design issue to be addressed is the human *vs.* robot-centred perception: should the human be allowed to observe the world with own senses; or should they have their perception restricted to robot-mediated data.

While the former statement results in a simplified system, the potential applications of MRS aforementioned inevitably fall into the latter. The teleoperation platform implemented in this work is therefore based upon having a restricted perception of the environment, providing the human with the same remote perception that the robot can acquire locally through its sensors (the human is "placed into the robots' perceptual shoes").

2.2 Observations of human behaviour

Another key issue in designing a teleoperation platform is related to how to define the commands that are sent to the robot. The human actions are not directly observable to the robots. Although the humans are "placed in the robots' perceptual shoes", a robot has access only to its teleoperator manoeuvre commands, rather than the human's intended actions, as illustrated in Fig. 2.

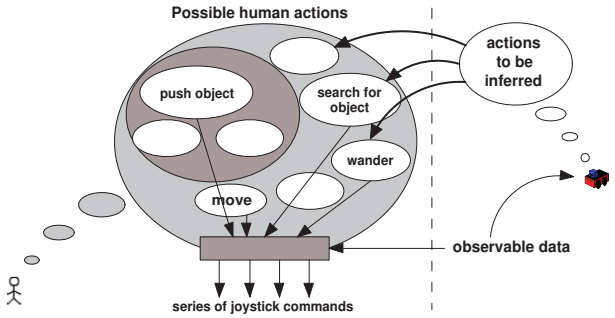


Figure 2: Human actions space *vs.* observable data diagram.

Two straightforward possibilities arise: send commands which represent the robot's underlying capabilities; or send control signals, such as motor commands to the robots.

The former assumption is coherent with most LbD methods, as the robot's primitive behaviours are used to match to actions observed by the robots. However, the human would be limited to few, inflexible handcrafted actions programmed into the robots, which are usually application specific, and also dependent upon the robot's design.

Conversely, the latter possibility allows humans to play with a full repertoire of actions, only restricted to environmental conditions. The teleoperation platform herein described makes use of this feature, even though matching motor commands to the robot's primitive behaviours is a more complex issue.

A key feature which strongly motivated this decision is due to the gain of flexibility, allowing the presented approach to be applied to distinct robots, such as unmanned aerial vehicles and wheeled-mobile robots, in a handful of potential applications in the field of MRS with little or no modification required.

2.3 Action recognition at single robot level

When addressing the problem of recognising observed actions, a mismatch between observed data and robot internal states might happen. This is a common issue known as the correspondence problem [10].

Even though the robot is able to passively observe the actions being performed by the human operator using its own sensors, mapping manoeuvre commands to robot primitive behaviours inevitably falls into the correspondence problem. Recognising actions from observed data when using a teleoperation platform becomes even more challenging, as the state of the environment is only partially observable. Thus, important variables may not be present at a particular observation of the state of the environment.

Furthermore, human actions have deliberative and reactive components. During task execution, a human deliberately actuates on the joystick in order to manoeuvre the teleoperated robot. In addition, sudden changes in the perceived environment (e.g., a moving obstacle appears in front of the robot) result in a reactive behaviour of the human, attempting to change the robot's course. Likewise, the human

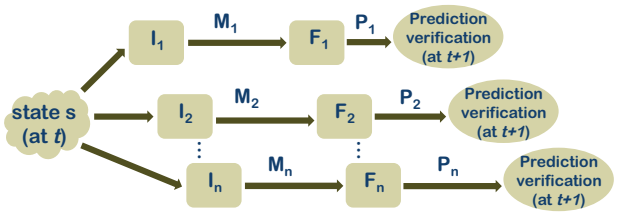


Figure 3: Diagramatic statement of the HAMMER architecture. Based on state s_t, multiple inverse models (I_1 to I_n) compute motor commands (M_1 to M_n), with which the corresponding forward models (F_1 to F_n) form predictions regarding the next state s_{t+1} (P_1 to P_n) which are verified at s_{t+1}.

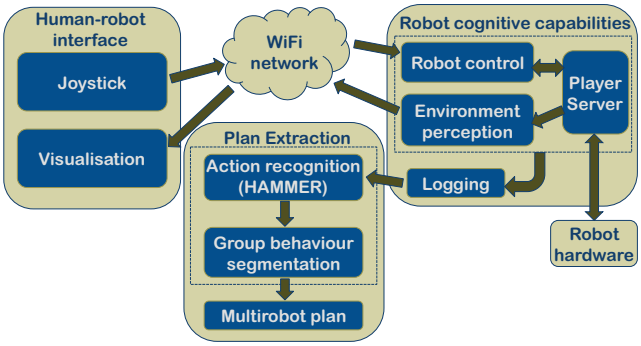


Figure 4: Overview of the teleoperation platform developed in this paper.

may perform certain actions sequentially or simultaneously resulting in a combination of actions, while the robot has access to the joystick commands only.

In order to recognise actions from observed data and manoeuvre commands, this paper makes use of the Hierarchical Attentive Multiple Models for Execution and Recognition (HAMMER) architecture [7], which has been proven to work very well when applied to distinct robot scenarios. HAMMER is based upon the concepts of multiple hierarchically connected inverse-forward models. In this architecture, an inverse model has as inputs the observed state of the environment and the target goal(s), and its outputs are the motor commands required to achieve or maintain the target goal(s). On the other hand, forward models have as inputs the observed state and motor commands, and the output is a prediction of the next state of the environment. As illustrated in Fig. 3, each inverse-forward pair results in a hypothesis by simulating the execution of a primitive behaviour, and then the predicted state is compared to the observed state to compute a confidence value. This value represents how correct that hypothesis is, thus determining which robot primitive behaviour would result in the most similar outcome to the observed action.

3. SYSTEM IMPLEMENTATION

The MRLbD approach proposed in this paper is demonstrated using the aforementioned platform for robot teleoperation, which consists in a client/server software written in C++ to control the P3-AT robots (Fig. 1) utilised in the experiments, as well as an implementation of the HAMMER architecture for action recognition and a Matlab implementation of the SC algorithm similar to the one presented in [14]. An overview of the teleoperation platform can be seen in Fig. 4.

The server software comprises the robot cognitive capabilities and resides on the robot's onboard computer. The server is responsible for acquiring the sensor data and sending motor commands to the robot, whereas the client software runs on a remote computer and serves as the interface between the human operator and the robot.

3.1 The robot cognitive capabilities

Within the *Robot cognitive capabilities* block, the server communicates with the robot hardware by using the well-known robot control interface *Player* [6], which is a network server that works as a hardware abstraction layer to interface with a variety of robotic hardware.

Initially, the internal odometry sensors are read. This data provides the current robot's pose, which is updated as the robot moves around and used as the ground truth pose for calculating objects' pose and building the 2D map of the environment. Odometry sensors are known for inherently adding incremental errors and hence lead to inaccurate pose estimations; but nevertheless, it is shown later on in Section 5 that this inaccuracy was immaterial to the results.

The image captured ($320x240$ pixels, coloured) from the robot's camera (at 30 frames per second) is compressed using the JPEG algorithm and sent to the client software over a TCP/IP connection using the Wi-Fi network.

Additionally, the image is also used to recognise objects based upon a *known objects database*, using the approach presented in [15]. This algorithm consists in detecting the pose (Cartesian coordinates in the 3D space, plus rotation on the respective axes) of unique markers. The *known objects database* comprises a set of unique markers and the object that each marker is attached to, and also offset values to compute the pose of the object based upon the detected marker's pose.

A short memory algorithm, based upon confidence levels, was also implemented to enhance the object recognition; the object's pose is tracked for approximately 3 seconds after it has last been seen. This approach was found extremely useful during the experiments, as the computer vision algorithm cannot detect markers from distances greater than 2 metres and occlusion is likely to happen in real applications.

The Sick LMS-200 laser range scanner provides millimetre-accuracy distance measurements (from up to 80 metres), ranging from 0 degrees (right-hand side of the robot) to 180 degrees (left-hand side). In addition, 16 sonar range sensors, placed in a ring configuration on the robot, retrieve moderately accurate distance measurements from 0.1 to 5 metres and a 30-degree field of view each. Despite the lack of precision, the sonar sensors play a fundamental role in the overall outcome of the teleoperation platform: as the human operator has limited perception of the environment, particular manoeuvres (mainly when reversing the robot) may be potentially dangerous and result in a collision. Thus, obstacle avoidance is achieved by using an implementation based upon the well-known algorithm VFH (Vector Field Histogram) [2]. However, the human operator is able to inhibit the sonar readings as desired, feature which is useful when pushing objects, passing through narrow gaps and

doorways.

In addition, joystick inputs are constantly received from the client and translated into motor commands (translational and rotational speeds), which are then sent to the robot through the Player interface.

Lastly, all the data manipulated by the server is incrementally stored in a log file every 0.5 seconds. The log file comprises a series of observations made by the robot during task execution, which are composed of the following elements: time stamps in the Unix time format; robot's pose based on odometry data; list of pose of objects recognised and their unique identification; laser and sonar range sensor readings; and finally, joystick inputs and motor commands.

3.2 The human-robot interface

The client software constitutes the human-robot interface, in which the visualisation module displays to the human operator the sensor data which is received from the server. This data comprises the robot's onboard camera, battery level and Wi-Fi signal strength, sonar and laser range scanners, and the robot's pose based upon odometry sensors.

The image is decompressed and displayed in a dedicated window, while a second window shows a sketch of the robot in the centre, as well as sonar and laser data. A screenshot of the human-robot interface can be seen in Fig. 5.

Furthermore, line segments are extracted from the laser data (using an implementation of the Split-and-Merge algorithm described in [11]) and displayed on top of the raw laser data. These line segments are mostly red-coloured, apart from line segments which width coincides with a known object's width; these segments become blue-coloured. This colour differentiation is not yet used by the robots, but it is meant to be used as an attention mechanism for the operator which highlights probable objects of interest based upon their shape.

Also, in case another robot is recognised, then a blue-coloured ellipse boundary representing the observed robot's pose is displayed; a green-coloured circle border is printed otherwise. Note that the size of these shapes are scaled according to the real size of the objects.

In addition, an image on the top-right side of the main window displays the map of the environment, which is incrementally built using laser and odometry data. This visualisation was found substantially useful for location awareness when navigating throughout the corridor and rooms.

In the meantime, the human actively teleoperates the robot by controlling translational and rotational speeds via joystick inputs.

At the end of a task execution, the log files stored by both robots are sent to the *Plan Extraction* block. Within this block, the actions performed by the humans at single robot level are recognised by using the HAMMER architecture, and group behaviour segmentation is achieved by applying the SC algorithm, as described in the following sections.

3.3 The HAMMER architecture

The implementation of the HAMMER architecture for this paper makes use of five inverse-forward model pairs, defining the robots' underlying capabilities. These pairs are: *Idle* (the robot does not make any movement), *Search* (the robot explores the environment looking for a particular type of object), *Approach* (the robot approaches a specific object once it is found), *Push* (the robot moves a particular object

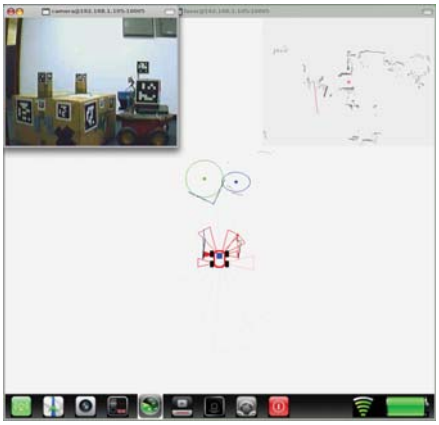


Figure 5: A screenshot of the human-robot interface; camera is shown on top-left, while the 2D map can be seen on top-right. A box (green circle) and a robot (blue ellipse) were also recognised and displayed on the main window.

to a determined area known *a priori*) and *Dock* (the robot navigates back to its base). In this implementation, the state of the environment is defined by the spatio-temporal relationship between the robot and the objects it recognises at each iteration.

The inverse models are hand-coded primitive behaviours which compute a navigation path using the clamped cubic splines algorithm, based upon the current observed state of the environment and the desired target. As an example of the *Approach* inverse model, given a partial observation of the environment where the robot recognises an object, the inverse model outputs motor commands such that causes the robot to move towards the object.

Similarly, the forward models were hand-coded procedures based upon the differential drive kinematics model of the robot. Given the motor commands, the forward models output a prediction of the next state for the correspondent inverse models. Recalling the example describing the *Approach* inverse model, the correspondent forward model, based on the motor commands, predicts that the robot should be closer to the object at the next state.

It is worth noticing that the inverse models are independent of each other. While each inverse model has its internal conditions which must be satisfied (e.g., to push an object, the robot must recognise and be close enough to the object), there are no interdependence between inverse models. This means that an inverse model does not influence or inhibit others (e.g., there is no rule defining that *Push* must be preceded by *Approach*). Thus, forward models can formulate a hypothesis by simulating the execution of the inverse models.

This is a key advantage of the HAMMER architecture, in contrast with other approaches, such as [12], in which the mapping from observations to primitive behaviours relies on task-dependent, static preconditions for a given behaviour.

3.4 Group behaviour segmentation

In most of the intelligent systems involving multiple entities (e.g., virtual agents, robots, humans, cars and so forth), the data obtained is often represented in the spatial domain.

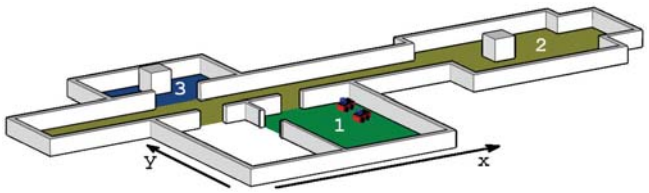


Figure 6: Map of the real environment where experiments were performed: Green area (1) indicates the robots' base, while beige (2) shows the area where the box is randomly located at the start; the blue (3) area denotes the desired final location of the box. X and Y axes are also annotated.

This data is usually collected in large scale (e.g., hours of video images from CCTV cameras), which increases the difficulty in extracting the desired information. In such cases, techniques for clustering the data into smaller groups are usually utilised.

As a large amount of data, comprising spatial and temporal information of multiple robots and objects, is obtained from the demonstrations of desired group behaviour, this paper makes use of the SC algorithm presented in [14].

In [14], the authors extended the SC to the time domain by redefining the affinity measure in order to incorporate the time component. In addition, a temporal sample number N_T is defined to counter the problem of varying sampling frequency, which is advised to be much smaller than the number of observations contained in the log files. The affinity measure proposed by [14] makes use of two distinct scaling parameters: σ_S for spatial normalised cuts and σ_T for temporal normalised cuts, in contrast with traditional clustering algorithms. These parameters play an important role in the SC algorithm, balacing spatial and temporal cuts, thus favouring either spatial or temporal segmentation.

The parameter values adopted in the implementation of the SC algorithm for this paper were $N_T = 100$, $N_S = 8$ and $\sigma_T = 0.1$. These values were thoroughly determined based upon the results shown in [14] and in such a way that neither spatial not temporal segmentations would be prioritised, therefore preserving the generality of the algorithm when applied to different contexts and tasks.

Also, the 2 events added to the SC algorithm were carefully defined based upon the known objects recognised by the robots during the experiments, as a means of facilitating the group behaviour segmentation. The event SEESBOX occurred every time the robot recognised a box within its field of view, while the event SEESROBOT occurred when the robot recognised its group mate. The importance value of both events was set to 1. No other events were taken into account in order to ensure that the algorithm would not be tailored to the task performed.

3.5 Multirobot plan generation

In this paper the output of the SC algorithm, which represents the group behaviour segmented over time and space, is combined with the recognised actions at single robot level in order to build multirobot action plans. This work hence moves a step further from the work of [14], where the output of the SC was not used in any subsequent stage, but only displayed on graphs for human analysis.

Firstly, the log files of each robot are processed by the HAMMER architecture. Based upon the prediction P of each forward model, a coherence verification is made. This verification returns a binary similarity value (either 0 or 1), which describes whether the correspondent inverse would result in a similar observed state. At this state, the prediction P of more than one inverse-forward model pair may be coherent with the observed state. As an example, a robot might be moving towards a particular object, but its movement could also be towards its base, causing both *Approach* and *Dock* inverse-forward model pairs to output coherent predictions. This issue is drastically minimised by providing the robot a spatio-temporal awareness at group level, using the SC algorithm.

Each cluster obtained from the SC algorithm encompasses a segment of the group behaviour. However, a case in which only one robot composes a cluster might happen. A sequence of group actions is generated for each robot independently. At first, it is assumed that the robot is performing a group action, regardless which action it might be. The execution of this group action terminates when a transition between clusters is observed, hence causing an action transition. The next group action then starts, and this procedure takes place until the last segmented cluster, representing the end of the task execution. As a result, the group action sequence of both robots over time is properly determined.

Possessing the group action sequence over time for each robot, as well as the actions at single robot level recognised by the HAMMER architecture, a multirobot joint action plan can then be generated. This plan comprises a sequence of actions at single robot level, with specific times to start and end action execution, that each robot should perform along time as an attempt to achieve the same goals of the task execution formerly demonstrated.

Actions at single robot level are defined based upon the forward model predictions and start/end times, which are obtained from the sequence of group actions. For every cluster in which a particular robot must execute an action at single level, the confidence level C of the inverse-forward model of index i is updated according to the following rule:

$$C_i(t) = \begin{cases} C_i(t-1) + 1, & \text{if prediction is coherent} \\ C_i(t-1), & \text{otherwise} \end{cases} \quad (1)$$

After computing the confidence level of all inverse-forward model pairs, the action with highest confidence level is selected as the action to be performed within that cluster. The effectiveness of the proposed approach to generate multirobot joint action plans is evaluated through the experimental tests hereafter described.

4. EXPERIMENTAL TESTS

The experiments were performed in a realistic scenario illustrating a search and rescue task, where two mobile robots P3-AT were utilised. Three different people conducted 4 trials of experiments in turns of 2 people per time. A training session with the duration of 10 minutes took place before the experiments in order for the participants to familiarise with the human-robot interface for teleoperation. They were also given the same instructions for the task execution.

The task consisted in searching for a box which location was initially unknown and then moving the box to a predefined area, subsequently returning to the base, where the robots were originally located, as illustrated in Fig. 6.

The participants were appropriately accommodated in different rooms, which were carefully chosen for the purpose of avoiding visual contact and verbal communication. More importantly, none of the participants was able to visualise the robots or any part of the environment in which the task was being executed (for reasons previously discussed in Section 2).

In addition, to elicit the execution of a truly joint task, requiring a tightly coupled collaboration between robots, the size and weight of the box were rigourously defined so a robot would not be able to push a box by itself, thus requiring precisely coordinated manoeuvres of both robots to move the box around and deliver to the desired destination.

5. RESULTS AND DISCUSSION

The 4 trials were conducted following the procedure detailed in Section 4. Trial 1 contains 502 observations and trial 2 consists of 534, while trial 3 comprises 729 and trial 4 has 425. The log files containing the observations made by the robots engaged in the task execution were processed using the SC algorithm and the HAMMER architecture to generate the multirobot joint actions plans.

To facilitate the understanding of the group behaviour segmentation obtained from the SC, Fig. 11 illustrates where a few sampled points of the trajectories are located on the map. X and Y axes represent the 2D Cartesian coordinates of the robots, while Z axis represents time. Different colours symbolise different clusters.

In trials 1, 2 and 3, the box was originally located in the right-hand side foyer (area 2 shown in Fig. 6). Based upon what was witnessed during the experimental tests and a visual analysis of the Figs. 7(a) (trial 1), 8(a) (trial 2) and 9(a) (trial 3), the demonstrated task executions can be described as follows.

Starting from the base, the robots navigated across the corridor in order to locate the box. At that stage, the robots were concurrently deploying individual actions. Shortly after recognising the box, each robot was manoeuvred to a position so that it could push the box towards the destination (area 3 in Fig. 6). As a consequence of different exploration strategies (one could choose to turn left, instead of right, when leaving the base), the robots did not find the box simultaneously. Thus, one of the robots had to wait until the other was properly positioned to start moving the box. This outcome can be easily spotted in Figs. 8(a) and 9(a).

Immediately after the operators acknowledged that both robots were ready to start moving the box, the deployment of a tightly coordinated joint action started. It was found by the operators considerably hard to manoeuvre the robots while moving the box, demanding different periods of time, requiring correction manoeuvres as the box deviated from the ideal path, and thus resulting in different trajectories; nevertheless the box was successfully delivered to the desired area in all trials. Subsequently, the robots navigated back to the base, indicating the end of the task execution.

A straightforward interpretation of the Figs. 7(a), 8(a), 9(a) and 10(a) is the realisation of how stochastic the execution of the same task can be when using real robots, even though similar initial states and same goal were preserved.

From a thorough analysis of the results obtained, it is worth highlighting two main issues. Firstly, as discussed previously, the robots' pose estimation based on odometry sensors resulted in incremental shift errors along the Y axis

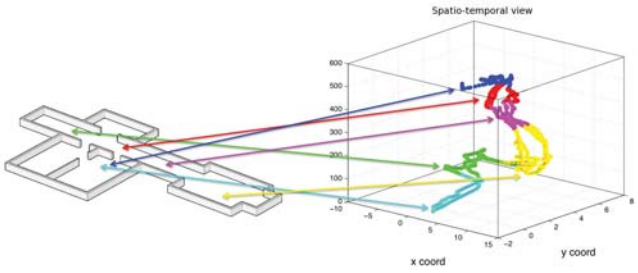


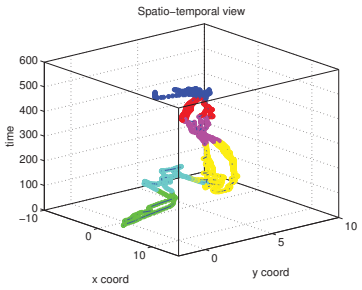
Figure 11: An illustration of the location of the robots at particular points within the trajectories over time, aiming for facilitating the comprehension of the segmented group behaviour. X and Y are the 2D Cartesian coordinates of the robots, while Z represents time.

on one of the robot's trajectory, which can easily be noticed in Fig. 10(a). Secondly, as a result of inaccurate pose estimations, the position of the box computed by each robot did not match when plotting its trajectory along time. Despite the different trajectories, this issue only marginally affected the plan generation results.

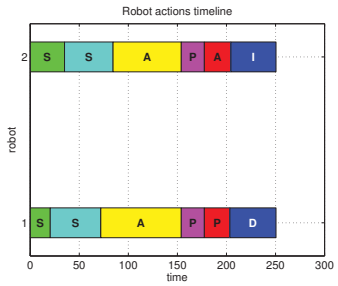
Robot joint action plans were generated according to the procedure previously described in Section 3.5. These plans are represented by text files comprising the total number of clusters, number of robots involved in the task execution, total time demanded to accomplish with the task, and a sequence of robot actions at single level over time. For the purpose of facilitating the comprehension, bar graphs showing robot actions at single level over time were appropriately generated from the original plan files (Figs. 7(b), 8(b), 9(b) and 10(b)). The characters within the bars represent the robot action, where "A" denotes *Approach*, "P" indicates *Push*, "S" represents *Search*, "D" stands for *Dock* and finally "I" denotes *Idle*. The colours were deliberately chosen to match to cluster colours, and thus ease comparison between the bar graphs and 3D graphs.

While Fig. 7(b) (trial 1) suggests that the first 3 clusters could potentially represent joint actions (because the actions of both robots belong to the same cluster and are approximately synchronised in time), Figs. 8(b) (trial 2) and 9(b) (trial 3) indicate that these actions are independent. Moreover, the actions *Search* and *Approach* are intrinsically independent, as a robot can execute these actions without the help of its group mates. Besides the first 3 trials, Fig. 10(b) interestingly indicates that the first 3 clusters are independent actions. However, the cyan-coloured cluster would possibly not exist if the robots' pose were appropriately estimated; but nevertheless, the action sequence is plausible.

When comparing trials 2 and 3 (Figs. 8(b) and 9(b)), the first 2 clusters are nearly identical. In regards to the actions at single robot level, while in trial 2 the second action was selected as *Push*, in trial 3 the action selected was *Idle*. Recalling that the action selected is the one with the highest confidence level, in trial 2 the action *Approach* was probably not selected for robot 2 because it was already too close to the box once it recognised it; the action *Push* was selected instead. Similarly, in trial 3, robot 2 was required to wait until robot 1 approached the box for a period of time long enough to cause the action *Idle* to have the highest confi-

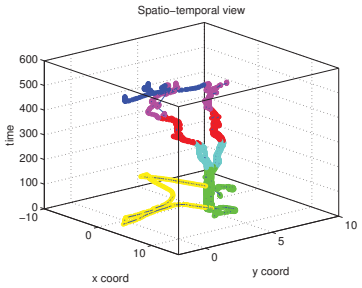


(a) 3D view of robots' spatio-temporal behaviour segmentation

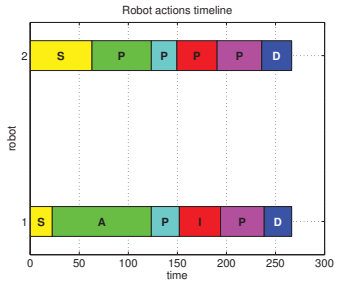


(b) Extracted plan of recognised robot actions along time

Figure 7: Results obtained from trial 1 (colours and characters illustrate the same cluster).

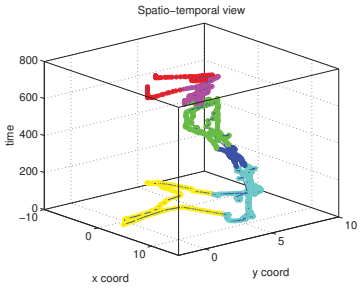


(a) 3D view of robots' spatio-temporal behaviour segmentation

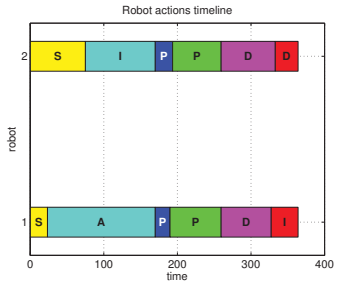


(b) Extracted plan of recognised robot actions along time

Figure 8: Results obtained from trial 2 (colours and characters illustrate the same cluster).

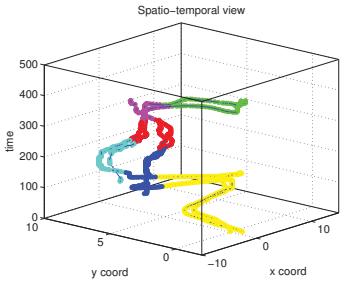


(a) 3D view of robots' spatio-temporal behaviour segmentation

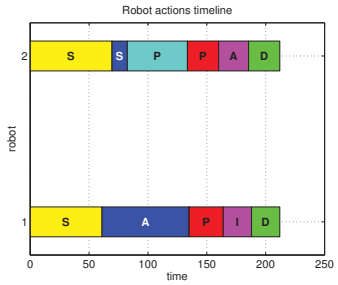


(b) Extracted plan of recognised robot actions along time

Figure 9: Results obtained from trial 3 (colours and characters illustrate the same cluster).



(a) 3D view of robots' spatio-temporal behaviour segmentation



(b) Extracted plan of recognised robot actions along time

Figure 10: Results obtained from trial 4 (colours and characters illustrate the same cluster).

dence level within cyan-coloured cluster.

On the other hand, the action *Push* was accurately selected when both robots started moving the box in all the trials. This markedly indicates the tightly coupled and synchronised nature of joint actions (in this experiments, the action *Push*), which is, perhaps, the most important aspect of the results obtained, corroborating the multirobot joint action plan generation approach proposed in this paper.

It is also noticeable that, even though the desired sequence of actions at single robot level were not provided, the generated plans suggested a reasonable sequence, usually starting with *Search*, followed *Approach* or *Push* (this one often tightly synchronised in time), terminating with *Dock*, with the action *Idle* being interleaved when synchronisation between robots was required (e.g., Fig. 9(b)).

6. CONCLUSION AND FUTURE WORK

This paper consolidates the use of SC in multirobot scenarios and advances on the challenging topic of multirobot plan generation from human-demonstrated task executions, proposing the novel MRLbD approach.

It was demonstrated in this paper that clustering a series of group manoeuvres in the spatio-temporal domain results in a reasonable and realistic group behaviour segmentation. In addition, a novel approach to learn multirobot joint action plans – the MRLbD – using SC combined with the HAMMER architecture was presented, which yielded to striking results. These plans can be used by a group of robots to autonomously attempt to replicate the task executed, preserving the spatio-temporal task execution procedure and seeking the same goals.

Work is already in progress in order to solve the problem of inaccurate robot pose estimation using odometry sensors. The use of visual landmarks, as well as an implementation of the well-known Monte Carlo Localisation algorithm are being incorporated into the teleoperation platform for robot localisation and environment mapping.

The experiments reported in this paper demonstrate that the MRLbD approach can learn sensible multirobot joint action plans that can potentially result in autonomous behaviour similar to the human-demonstrated task execution. This is the first stage to a number of useful potential applications, including learning to program a group of robots by demonstration, post-event analysis and debriefing for the RoboCup or other multirobot tasks involving learning over extended period of time, and for computational architectures that require generating response to the actions of a group of agents, such as adversarial domains. Future work will explore such applications.

7. ACKNOWLEDGEMENTS

The support of the CAPES Foundation, Brazil, under the project No. 5031/06–0, is gratefully acknowledged.

8. REFERENCES

[1] B. D. Argall, S. Chernova, M. Veloso, and B. Browning. A survey of robot learning from demonstration. *Robotics and Auton. Systems*, 57(5):469–483, 2009.

[2] J. Borenstein and Y. Koren. The vector field histogram-fast obstacle avoidance for mobile robots. *Robotics and Automation, IEEE Trans. on*, 7(3):278–288, Jun 1991.

[3] S. Butler and Y. Demiris. Predicting the movements of robot teams using generative models. *Distributed Auton. Robotics Systems*, 8:533–542, 2009.

[4] S. Chernova and M. Veloso. Confidence-based policy learning from demonstration using gaussian mixture models. In *AAMAS '07: Proc. of the 6th Int. Joint Conf. on Auton. Agents and Multiagent Systems*, pages 1–8, New York, USA, 2007. ACM.

[5] S. Chernova and M. Veloso. Teaching multi-robot coordination using demonstration of communication and state sharing. In *AAMAS '08: Proc. of the 7th Int. Joint Conf. on Auton. Agents and Multiagent Systems*, pages 1183–1186, Richland, SC, 2008.

[6] T. H. J. Collett and B. A. Macdonald. Player 2.0: Toward a practical robot programming framework. In *in Proc. of the Australasian Conf. on Robotics and Automation (ACRA 2005)*, 2005.

[7] Y. Demiris and B. Khadhouri. Hierarchical attentive multiple models for execution and recognition of actions. *Robotics and Auton. Systems*, 54(5):361–369, 2006.

[8] T. Fong, C. Thorpe, and C. Baur. Advanced interfaces for vehicle teleoperation: Collaborative control, sensor fusion displays, and remote driving tools. *Auton. Robots*, 11(1):77–85, 2001.

[9] R. Kelley, A. Tavakkoli, C. King, M. Nicolescu, M. Nicolescu, and G. Bebis. Understanding human intentions via hidden markov models in autonomous mobile robots. In *HRI '08: Proc. of the 3rd ACM/IEEE Int. Conf. on Human Robot Interaction*, pages 367–374, New York, USA, 2008. ACM.

[10] C. L. Nehaniv and K. Dautenhahn. *The correspondence problem*, pages 41–61. MIT Press, Cambridge, MA, USA, 2002.

[11] V. Nguyen, S. Gächter, A. Martinelli, N. Tomatis, and R. Siegwart. A comparison of line extraction algorithms using 2d range data for indoor mobile robotics. *Auton. Robots*, 23(2):97–111, 2007.

[12] M. N. Nicolescu and M. J. Mataric. Natural methods for robot task learning: Instructive demonstrations, generalization and practice. In *In Proc. of the 2nd Int. Joint Conf. on Auton. Agents and Multiagent Systems*, pages 241–248, 2003.

[13] L. E. Parker. Distributed intelligence: Overview of the field and its application in multi-robot systems. *Journal of Physical Agents*, 2(1):5–14, March 2008. Special issue on Multi-Robot Systems.

[14] B. Takacs and Y. Demiris. Balancing spectral clustering for segmenting spatio-temporal observations of multi-agent systems. In *8th IEEE Int. Conf. on Data Mining (ICDM '08)*, pages 580–587, 2008.

[15] D. Wagner and D. Schmalstieg. Artoolkitplus for pose tracking on mobile devices. *Proc. of the 12th Computer Vision Winter Workshop 2007 (CVWW'07)*, February 2007.

[16] J. Wang and M. Lewis. Human control for cooperating robot teams. In *HRI '07: Proc. of the ACM/IEEE Int. Conf. on Human Robot Interaction*, pages 9–16, New York, USA, 2007. ACM.