Technologies for Human and Robot Teamwork

Isaac J. Sjoblom
University of Minnesota, Morris
Sjobl014@morris.umn.edu

ABSTRACT

This paper focuses on the technologies involved in the processes of teamwork between humans and robots in professional and domestic settings. Technologies include automated task planners and sensor networks. The technologies allow for greater levels of autonomous actions to directly inter-operate with human actions, minimizing intrusiveness.

Categories and Subject Descriptors

I.2 [**Computing Methodologies**]: Artificial Intelligence; C.2.1 [**Computer Systems Organization**]: Computer-Communication Networks—*Network Architecture and Design*

General Terms

Algorithms, Human Factors, Theory

Keywords

autonomous, robots, human, interaction, task planner

1. INTRODUCTION

Technology is progressing and the field of robotics is expanding farther into the home and work place. Robots are being used to vacuum carpets and mop floors, as well as aid in experiments and other job tasks. This can create just as many problems as are being solved. Problems involving noise and clutter can arise during an autonomous robot's work cycle causing stress and possibly danger to human inhabitants. There is also a question of how to optimize a robot's workload and function to be as autonomous as possible so that the humans involved are free to pursue their own chores and tasks instead of always having to bother with giving the robot tasks. This paper will look at several algorithms and systems that address the issues of human-robot teamwork and cohabitation. This paper applies to semi-autonomous and autonomous robots, that is, robots that operate free of direct human control. Semi-autonomous robots operate with human control but have autonomous operations such as stabilization and path finding.

UMM CSci Senior Seminar Conference Morris, MN.

1.1 Human-Robot Teamwork Examples

There are many systems already in place where autonomous robots are working side by side with people. There are also situations were people are working to implement autonomous systems. A few examples of these follow.

A great example of a system already in place is a robot from NASA, the Robonaut 2 (R2). This is a semi-autonomous robot designed for researchers on the international space station to observe how robots behave in space. The plans for the robot's future are to help spacewalkers maintain and build on the space station as well as assist in experiments [2].

Research is currently being done on the possibility of using autonomous robots in construction, removing humans from the more dangerous situations [7]. Such research has shown that there may be need to revise our current construction methods if autonomous robots are going to be involved in the heavy lifting.

An example of how robots might be used in the near future is in nuclear cleanup. Right now, most of the nuclear clean up in Japan is done by hand; however, autonomous robots are expected to provide a bigger role in the future. Nuclear clean up requires robots that are capable of many movements as well as requiring them to have sufficient shielding to prevent excessive damage in the robots normal work tasks [5].

These are just a few examples of current technologies and the goals that future robotics projects will be moving toward. There will also be many commercial applications that require design to be compatible with human actions. The great example of commercial robots available today are the robotic vacuum cleaners. Robotic vacuum cleaners run autonomously on a cycle, cleaning small messes and dusting the floors so that a human does not have to do this; however, these robots can generate a lot of noise and become a hazard to human occupants. We will discuss how to avoid these problems in a later section.

This paper will discuss human-aware planners, task planners, and sensor networks. We will see what these technologies are and how they can change the existing design of human-robot teamwork. We also examine how these technologies may be able to inter-operate with each other.

2. HUMAN-AWARE PLANNING PROGRAM DESIGN

In a world where robots are taking on the dull actions of the home and workplace, it is almost guaranteed that a robot will eventually get in a human's way. This may not only cause tension, stress and irritation, but could also be dangerous for the human. Robots on the other hand can have their work impeded by human interference. This reduces the robot's usefulness. If autonomous robots are to work side by side with humans then consideration must be made not only in the actions a robot takes, but also in its planning of future jobs. *Human-aware planning* is a term defined by M. Cirillo et al. that describes the observation of human behaviors in determining a robot's future plans [1]. Planning is done by a probabilistic approach to human action. The need for a probabilistic approach comes from the necessity of being aware that human actions are not absolutely known or even necessarily predictable. Using an adaptive system, a robot can continuously update its probabilities and give the best guess of a human's course of action.

2.1 Human-Aware Planning Program Design

According to M. Cirillo et al. [1], the needs that a human-aware task planner program should take into account are:

1. Support for several courses of action a human might take in a given time block
2. Estimated duration of any given task
3. Ability to specify *interaction constraints* (IC), defined as "formulas that determine how the robot should or should not interact with a human."
4. Support for situations involving partially completed tasks
5. Support for observing interference of human actions on a robot's given task

2.2 State Transitions, Moving from One Task to Another

The above list describes the functionality of a complete program. This paper will look more closely at the algorithms that take a list of robot actions and a list of human actions and produce a course of action that is the most probable to succeed with minimal human-robot collisions.

For the algorithm of such a program, there needs to be consideration for a few terms. Firstly s is the notation for a state, S is the notation for a set of all states. Actions are denoted a, a time duration $t_a \in \mathbb{R}$, and the transition function $Res_a : S \times S \to [0, 1]$. $Res_a(s, s')$ is the conditional probability for transitioning to state s' from a state s when action a is performed. For instance, from a given state s a robot may be able to select many actions, which leave the world in a new state. The Res_a function will determine which transition is most likely to succeed. A^H is the set of human actions and A^R is the set of robot actions. An agenda is a finite list of activities $(a_1^H, a_2^H, a_3^H, ..., a_n^H)$. These activities may be attempted to be ordered, but due to the probabilistic nature of human action from a robot's standpoint it is not truly ordered. A *situation* is a tuple (s, t_r, t_h, A^H). s is a state, t_R is the time that the last complete robot action ended, t_H is the time that the last complete human action ended, A^H is the remaining agenda for a human.

The piecewise equation to calculate Res_a can be found in Figure 1. Here in Equation 1 the first part of Res_a is a recursive function that generates a new situation s'' in the event that the next human action a_{nxt}^H ends before the current robot action a_{cur}^R leaving the world in a new state. Read the *when* clause as: the new action a' is the first human action of the remaining human agenda (a_{first}^H) and the remaining human agenda a'' is the rest of the human agenda excluding a' and the calculated time the next human action ends, t''_H is the sum of the time the last human action ended t_H and the estimated time for the next human action t_{nxt}^H and that t''_H is less than the time it takes a robot to finish its current and next tasks $t_R + t_a$. The probabilities of all possible outcomes of a_{nxt}^H are then summed up.

The second part of the equation is for the event that a robot's current action a_{cur}^R ends before the next human action a_{nxt}^H. You can read the *when* clause as such; when the next robot's action ends ($t'_r = t_r + t_{a_{nxt}^R}$) and the next human action has not started yet ($t'_H = t_H$) and for all the possible human actions that could occur next, the time the robot completes its next action is less than the time that the current human action ends plus the time it takes for the next human action to end, and the current human action is the human action in the new situation s' ($a'_H = a_H$)

An example of the Res_a function would be:
Parameters:
The time to clean: $t_{clean} = 5$
The probability of cleaning successfully is 1: $Res_{clean}(s_1, s_2) = 1$
The time of the next human action to end is 4: $t_{watchTV} = 4$,
The probability of the human successfully watching tv is 1: $Res_{watchTV}(s_1, s_2) = 1$
The remaining human agenda at the point where the robot starts its next action *clean* is: $\{watchTv, brushTeeth\}$

$$
\begin{aligned}
&Res_{clean}[(s_1, 5, 3, \{watchTv, brushTeeth\}),\\
&(s_3, 10, 7, \{brushTeeth\})]\\
=&Res_{watchTv}(s_1, s_2)\times\\
&Res_{clean}[(s_1, 5, 7, \{brushTeeth\}),\\
&(s_2, 10, 7, \{brushTeeth\})]\\
=&Res_{watchTv}(s_1, s_2)\\
&\times Res_{clean}(s_1, s_2) = 1.0
\end{aligned}
$$

Of course this is a very basic example. Not only are the probabilities for many actions not going to be 100%, a state may transition to several different possible states with associated probabilities for transition, and a state may be reached through many different previous states. An example of this could be that a human has a 30% probability to go brush his teeth, but a 70% probability to take out the garbage and smoke a cigarette. This would possibly give the robot enough time to clean the bathroom before the human goes in there.

2.3 Placing Human-Aware Boundaries on Robot Actions

Interaction Constraints (ICs) are logical functions that determine what a robot can do given a human activity. In the above example, a robot could have the IC to not clean while the human watches television. IC's are generally log-

$$Res_a[(s,t_R,t_H,A^H),(s',t'_R,t'_H,A^{H'})] = \begin{cases} \sum_{s''} Res_a(s,s'') \times Res_a[(s'',t_R,t''_H,A^{H''}),(s',t'_R,t'_H,A^{H'})] \\ \quad \text{when } a' = a^H_{first} \wedge A^{H''} = rest(A^H) \wedge t''_H = t_H + t_{a^H_{nxt}} \wedge t''_H \leq t_R + t_a \\ Res_a(s,s'') \\ \quad \text{when } t'_R = t_R + t_{a^R_{nxt}} \wedge t'_H = t_H \wedge \\ \qquad \forall a^H_{nxt}(a^H_{nxt} = first(A^H) \Rightarrow (t'_R < t_H + t^H_{nxt}) \wedge A^{H'} = A^H \end{cases}$$

Figure 1: Piece-wise Transition Function Res_a

ical constraints in the form of *always(not(human_is_in = 'TVRoom') and (robot_is_in = 'TVRoom')* If such an IC was in place, the robot would not clean the TV room until the human has finished watching TV and has exited the room.

Interaction Constraints and the Res_a function are enough to compute actions in an environment where there is a complete awareness of the current situation for the robot, including the full human agenda. For an algorithm that predicts human agendas and other unknown variables see [1]. A simplified algorithm may be as follows:

```
While robot agenda is not empty
   observe human
   compute Res for actions not in conflict with IC
   execute action with highest Res calculation
   remove the action from the robot agenda
```

A robot may determine violations of ICs based on *sensor networks* discussed in the sensor network section of this paper.

3. HUMAN ROBOT TEAMWORK PLANNER

So far we have discussed problems and solutions to human and robot cohabitation in the same environment. We will also mentioned ways that a robot may autonomously update its own heading and provide data back to a human worker. In this section we will discuss how robots and humans work side by side with each other and how the work load is effectively managed, particularly in environments where there are many unknown variables.

Many people have some experience with how human work load is divided out. A manager, who knows the overall goal as well as the employees under them, creates sub tasks for their employees based on their individual skills that move the team to the goal. This is similar for human and robot task planning, only here the manager may be a computer program instead of a person. The computer program is referred to as a *planner*.

Planners take into account some goal that the humans and robots working are trying to achieve. A common example of this is Urban Search and Rescue (USAR)[9]. In USAR small, fast robots are released in disaster zones looking for injured persons while the human rescuers then work to reach the injured persons found. The robots have the advantage of being able to get into places where debris would have to be cleared for a human to enter, this way less time is wasted clearing debris that does not lead to injured persons.

The biggest issue in planners that deal with these robot-human tasks, the environment in the situation, or at least a large part of it, tends to be unknown. Most of the time, there are no indigenous sensor networks so the planner can only be updated by individual robotic sensors, or from human reports. This is the biggest hurdle in designing planner systems for unknown environments. Environments can also change during a task. An example of this would be a ceiling falling down on a room a robot had just been in. The planner would not be aware of this and therefore would assign tasks to robots that may rely on the fact the ceiling has not fallen.

A common practice for the design of planner programs is to incorporate a rewards and punishments system. For instance, the planner has goals such as "find injured persons" and will give a numerical reward value to the robot if the robot completes this goal. Similarly failure to complete a goal will result in a punishment value. The robots then are designed to update their own programs effectively. The robots, for instance, will notice a door and know that a room is on the other side. Opening the door has the potential to deliver rewards. If the robot opens the door and finds no injured person, and reports one anyway, then the robot receives a punishment. Punishments and rewards will minimalize the errors in reporting and therefore make the robot more effective in robot-human teamwork.

Another common practice in the design of planners is to incorporate a cost system. Costs may also be given to robots to perform task optimization. The robot has finite resources, commonly linked to battery life. For instance, moving forward for x amount of time may incur a cost y giving the robot reason to do so only if it leads to potential rewards.

3.1 Planner Design Combination

The two above sections deal with human-aware task planning and open world teamwork planners separately. We can infer that a system may be designed with both of these implementations and benefit from both equally. A simple example of how this might work out is, a planner in a USAR scenario could make sure that robots have the interaction constraint to not search rooms that human USAR agents are currently in. There are many other examples that could benefit by including both systems in one program. Unfortunately little has been done in the melding of these new technologies. So little can be proven right now as to how effective the combinations could be.

4. SENSOR NETWORKS AND ENVIRONMENTAL INFORMATION

Situations which require multiple autonomous robots, or robots in corrosive environments, may benefit from a sensor network. The benefits of a sensor network include provision of more accurate data (such as precision data and zoning data) as well as limiting the number of sensors on an individual robot so that costs can be cut in some cases. In situations where a robot can be expected to fail eventually, such as in a corrosive environment, stationary sensors may

have better shielding than a mobile platform.

Sensor networks may provide more accurate information than individual robot sensors. Robotic positioning on a global level, where the robot is expected to be moving around, provides relational data for the robot and its boundaries. The alternative is to have individual sensors on each robot. Individual sensors on a robot provide very limited positional data. In a sensor network the robot may ask the network its position and make a much more informed decision on its next move. Sensors on a robot will only give the robot a report of its own position in an unknown world. Robotic sensor network inquiries will be discussed in a later section.

Another mentioned benefit of sensor networks is accurate zoning data. Zones are areas of an environment, usually categorized as dangerous and safe. For example, a robot may want to avoid areas of high contamination when cleaning a spill. A sensor network would be able to show the robot exactly where the high levels of contamination exist as well as the boundaries of that zone. Similarly, a robot may want to work in a specific area that, at the time, is being occupied by a gathering of people. A sensor network would be able to let the robot know where concentrations of people are located so the robot would not work there. The sensor network could then inform the robot when the group has moved or dispersed and that it is safe to work there now.

The ultimate goal in the design of a sensor network is to provide robots with enough data to remain as autonomous as possible. This frees up human efforts to focus on non-robotic tasks.

4.1 Sensor Network, Danger Field Calculations

In autonomous robotics applications involving danger zones, it is important to have a system that can continuously update and discover the safest path for a robot to its goal. The simplest example of a network that could achieve this is a network whose sensor nodes are evenly distributed. A robot's path would then be defined as, for instance, from node A to node T. Nodes can sense when they are in a danger zone and communicate this to its surrounding nodes, which in turn communicate this to their surrounding nodes. The information passed between nodes is the source of the danger, and how many node jumps it is from the danger node. That said, every time a node passes along the information it increments the distance (in number of nodes) the danger is by 1. Nodes with limited memory, would only store the information about the closest source of danger. The algorithm is as follows:

pot is the value for potential danger. The higher the *pot* value the less danger there is.
for all sensors s_i in the network do
  $pot_i = 0, jumps_j = \infty$ for any danger j
    if s_i senses danger then
      $jumps_i = 0$
      $SendMessage(i, 0)$ - i is source and 0 is number of jumps
    if receive $(j, jumps)$ then
      if $jumps_j > jumps + 1$ then
        $jumps_j = jumps + 1$
        $SendMessage(j, jumps_j)$
  for all recieved message m do
    Compute pot^m of m using $pot^m = \frac{1}{jumps_m{}^2}$
    Compute pot at s_i using all
      $pot^m, pot_i = pot_i + pot^m$

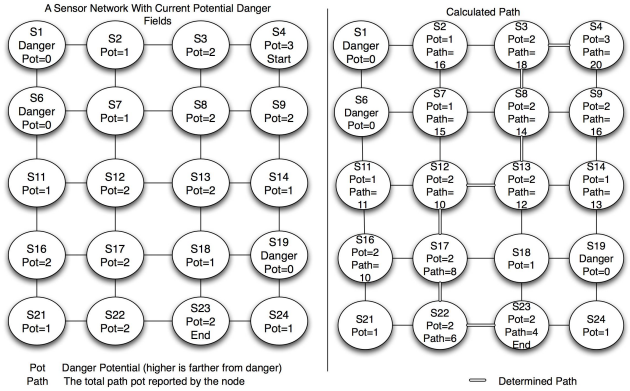

Figure 2: A sensor network updating potential danger field values for each sensor and mapping a path.

This makes the algorithm to determine the safest path between two nodes fairly easy.

A robot that wants to go from node A to node T would query node T as to how far from danger it is. Node T then transmits a message to all surrounding nodes giving an initial value of the potential danger field as 0. The nodes receiving this message add their own potential to the incoming value and broadcast the message from there with the new potential. A node will most likely receive many messages, it selects the message with the smallest reported potential and remembers the sender. The message gets back to the initial node A with the danger potential field values of A's surrounding nodes. In environments where danger can spread rapidly, the robot may run this query every time it moves to a new node, giving its updated position as the initial position. This way a robot always has the safest path between it and its goal. A depiction of this can be seen in figure 2 The robot may become trapped in this system though if the danger area spreads behind it and in front of it.

The design for most sensor networks is decentralization. A sensor network should be able to operate if any sensor node goes down at any time, which means that every node should have some idea of the state of the world. This provides a more robust system.

4.2 Sensor Networks, Active Sensor Networks

Sensor networks do not have to be stationary. *Active sensor networks* (ASNs) are sensor networks whose nodes are on mobile platforms. ASNs may be thought of as a network of individual robots who share their sensory data with similar robots working on the same task. This of course defeats the shielding benefits of stationary networks, but still holds the relational position data benefits. However, the relation is in respect to the position of the other robots instead of a fixed environment. Each node in an ASN shares a common view of the environment; where zones are and where the objective is. Similar to the stationary sensor networks, individual nodes only propagate information to adjacent nodes in an algorithm similar to the one outlined in section 4.1. The main goal of ASN design is to have the network comprised of some minimum number of nodes, the base number of nodes that is required to complete the task, but still be expandable with more nodes in the system without having

to perform any updates externally. The benefits of adding nodes to an ASN is to increase the speed and accuracy of data collected [6].

4.3 Human Roles in Sensor Networks

Teamwork with autonomous robots in a sensor network gives the human operator a different role than simply being a robot pilot. Human roles are now more observational and managerial. Such tasks a human would do is:

- Tell a robot what to do.
- Tell a system what operating mode to switch to.
- Determine what the task success case is.
- Give necessary information about the environment the robots may lack.
- Retrieve data from the network [6].

Designs of modern ASNs have the human operator retrieving information from a single given node. That is so that a time lag or resources spent communicating with the node does not extend to the system as a whole. This is opposed to querying an entire network, causing lag and spending resources to obtain information from all nodes. Also, if the operator had to query each node, the state of the world may have changed between querying the first node and the last node. Having each node share a common view of the environment will fix this problem. This will allow an operator to query any single node and get all the information they need. With a decentralized system, the default node an operator queries could change if that node goes down as every node should have the same propagated information about the system.

We can see how having an ASN would benefit the Urban Search and Rescue teams mentioned in the previous section. An ASN does not rely on an indigenous sensor network to be in place. Receiving information that is gathered from every robot gives a planner more current information about the state of the world since it can query a robot (node) actively seeking out injured persons while collecting environmental data.

5. FUTURE WORK

Research in the field of autonomous, and semi-autonomous, robotics and human interaction is just beginning. Future work involves the melding of these systems together for more direct application purposes. Other fields that may be linked to the topic in this paper have emerged such as cloud computing systems for robots. Research into cloud computing involves making smaller robots with longer battery life that pass on the processing and data storage loads to stationary bases [3].

Another related field, human-robot interaction, has an emerging research area which is the ethical conduct of robots [8]. Here researchers are developing algorithms to have a robot think for itself, weighing consequences of inaction or action often involving how a robot speaks to a human. An example of this may be a robot giving medication to a patient, where the patient indicates that they do not want to take their medication at that time. The question then arises weather to have the robot walk away or must the robot insist that the person take their medication. In the future it is hoped that a robot will be able to accurately determine the danger to the human if they do not heed the robots request and when the robot has the option of peacefully walking away without putting the human in danger through inaction. The reason for ethical conduct is partially for the reassurance that robotic servants are a good thing for those that are less than convinced.

Task planners are designed today with assumptions that the environment is stable and known. There is need for a planner that will take into account an open ended world where the environment is constantly changing and be able to adapt to these changes. Research has shown that the best way to accomplish this, as of now, is to take a more general approach to problems. The problem then becomes giving specific tasks to autonomous robots who do not operate on general goals, but only very specific linear instructions [9].

6. CONCLUSIONS

In summary human-aware planning makes a robot more efficient by working in areas where human actions will not interfere. Human-aware planning also makes robot cohabitation less dangerous and stressful for human occupants and workers. Task planning provides an efficient, calculated way in which robot and human teamwork efforts may produce more effective results. Sensor networks and active sensor networks allow robots to effectively communicate with each other and with a human operator or software planner.

Currently, many human-robot teamwork solutions do not incorporate these technologies. Sensor networks are being used, but not as effectively as they can be as outlined in this paper. We will be seeing more industrial, and hopefully commercial, robot planning software and systems in the future.

In situations where there is one human and one robot, a human-aware task planner described in section 2.1 performs its tasks finding close to optimal solutions. More has to be done to take into account tasks with an unknown or flexible time duration. More also has to be done to increase performance in situations where there are multiple people and robots. We can see that sensor networks may be implemented to aid a planner in increasing performance [1].

Sensor networks have been shown to guide robots and humans along the safest path in a dangerous environment. The metric used to determine success in these networks is the time it takes the process to update nodes when danger spreads or the environment changes in some way. Current algorithms that are designed to optimally scan for new obstacles and danger zones do show promise [4].

Acknowledgments

Thanks to Elena Machkasova and Chris Olson for proof reading my paper and giving me excellent feedback.

7. REFERENCES

[1] M. Cirillo, L. Karlsson, and A. Saffiotti. Human-aware task planning: An application to mobile robots. *ACM Trans. Intell. Syst. Technol.*, 1:15:1–15:26, December 2010.

[2] B. Dunbar. Robonaut 2. *NASA*, 2011.

[3] E. Guizzo. Robots with their heads in the clouds. *Discovery News*, March 2011.

[4] Q. Li and D. Rus. Navigation protocols in sensor networks. *ACM Trans. Sen. Netw.*, 1:3–35, August 2005.

[5] D. Lyons. Send in the robots to japan's nuclear meltdown. *The Daily Beast*, March 2011.

[6] A. Makarenko, T. Kaupp, and H. Durrant-Whyt. Scalable human-robot interactions in active sensor networks. *IEEE CS and IEEE ComSo*, November 2003.

[7] T. Staedter. Robotic quadcopters build a tower autonomously. *Discovery News*, Jan 2011.

[8] T. Staedter. Robot makes ethical decisions. *Discovery News*, November 2010.

[9] K. Talamadupula, J. Benton, S. Kambhampati, P. Schermerhorn, and M. Scheutz. Planning for human-robot teaming in open worlds. *ACM Trans. Intell. Syst. Technol.*, 1:14:1–14:24, December 2010.