Karlsruher Institut für Technologie (KIT)
Institut für Prozessrechentechnik, Automation und Robotik (IPR)
der Fakultät für Informatik

Path Planning for Contact Based Safe Human-Robot Cooperation

Bachelorthesis
by
José Carlos González Dorado

State: October 2, 2012

Advisor:
Prof. Dr.-Ing. Heinz Wörn
Dipl.-Inform. Stephan Puls

Contents

Chapter 1

Introduction

1.1 Abstract

This work improves a human-robot cooperation system using the research framework MAROCO. There are already implemented many essential parts needed to provide this cooperation, but some of them work independently of each other. The main goal is to join all these parts to conform a system that allows contact based cooperation.

1.2 Motivation

Since the seventies, industry has been using robots to automate certain parts of their industrial processes and reduce their production costs. The automotive industry was the great launcher of this technology, expanding it and diversifying it in different ways, but very especially in robotic arms. Currently these robots are part of the daily routine in the big industries, accelerating and perfecting their production.

In this environment, robots have been replacing human workers and have brought a huge amount of advantages like more precision, more strength, no fatigue or the fact that they can be used in very dangerous tasks for humans. In industry they can be used to solder, mount, cut, mechanization, inspection, etc., but out of the industrial use the applications are also abundant. Among them it is worth mentioning their use in surgery and helpers for disabled people. Nowadays, research in this field is very active and each year robotics can solve more problems.

Non teleoperated robots normally need very systematic tasks to work well. In assembly lines, usually this is not a problem, but it limits the capabilities of this technology. Also, robots usually work separately from humans due to collision danger. The robot follows preprogrammed paths and behaviors and usually lacks of sensors to avoid collisions with a person.

Considering the meaning of cooperation, in the future it can arrive as far as humanoid robots freely interacting in the society, but nowadays, even in very "basic" levels like a robotic arm giving an object to a human, this cooperation has not been achieved in practice. Normally this cooperation consists only in disabling the robots or inducing them a "safe" working mode for a person that needs to manipulate something in the working area of the robots. It does not exist real contact based cooperation between human and robot. Research in this area opens a wide range of potentially exploitable applications.

The best way to perform this cooperation is when the human does not carry any special devices. This implies that sensors to detect the human pose are needed and that a computer processes the information from the sensors to determine what should be done by the robot to cooperate. A first step after detecting the human pose is avoiding the human at every time to continue working alone. However, the key point is to have a contact based cooperation with the human. For this, in addition to the human pose detection, it is important to anticipate its intentions with a proactive behavior, so that the robot can approach the human safely. The underlying complexity of a system with these characteristics, the added cost of the sensors, the potential risks that can appear and the lack of confidence of the people to work with these non teleoperated devices cause that contact based human-robot cooperation needs more research to handle the vast amount of possibilities that this young field has to offer.

1.3 Task description

The target of this project consists in perfecting a contact based cooperation system between a human and a robot. The robot is a Reis RV6L arm and its movements have to be planned to cooperate with a human safely. There is a 3D sensor installed at the ceiling to detect the human pose inside the working area as shown in Figure 1.1.

These devices are integrated into the research framework MAROCO [Puls et al. 2012]. The main modules to allow the cooperation are already implemented in the framework, but some of them work without connection with each other which currently prohibits contact based cooperation.

- Human pose reconstruction: It uses an Optical Flow Field that is computed using the XCLG method with multigrid solvers [Graf et al. 2010]. With the 3D sensor installed at the ceiling, this module is able to reconstruct the human pose and translate it into a 3D model inside the framework. It focuses in head, shoulders and arms position. Also, it can determine the direction where the person is looking. There is no need to modify this module because the human pose is translated directly to a 3D model. This model has all the necessary parameters like position, etc., so the method used to set it is totally transparent for this work.

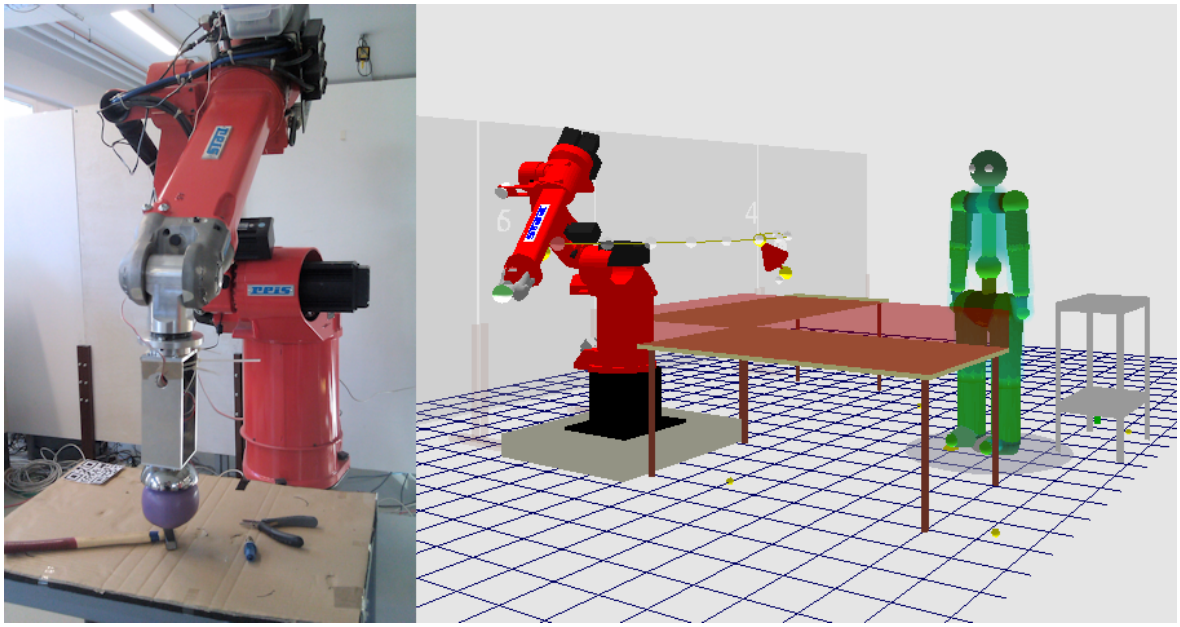


Figure 1.1: Real and simulated working area in MAROCO.

- Situation awareness: Using the provided data by the 3D sensor, this module is able to determine the human intentions and the situation, like start the cooperation, transmit orders, take a rest, etc. It makes use of Description Logics to represent knowledge [Puls et al. 2011]. Currently this system is not connected to any other module, so the detected situations have no effect in the perceived risk nor in the behavior of the robot. There is no need to modify this module because only its output is important to know how to do the cooperation. In addition, the situation awareness does not work in real time, so there will not be a detection of the situation for each cycle.

- Risk quantification: In the MAROCO framework there are different risk estimation techniques implemented. The best one uses a knowledge base built with two-threaded fuzzy logics [Graf et al. 2010b]. These logics allow positive and negative fuzzy rules into the knowledge base. Also it uses a custom hyperinference operator which is a trade-off in comparison to the typical strong and weak veto. In essence, this system takes into consideration the position of the robot and the human and establishes a risk value according to their distance and the relative speed between them. Part of the work of this project consists in modifying this module to calculate the risk depending on the output of the situation awareness system. The addition of new fuzzy rules is needed to give a useful meaning to the risk to allow contact based cooperation.

- Path planning: The path planning for the robotic arm is based on a preemptive

A* algorithm, allowing the program to work in real time [Graf et al. 2009]. If the path recalculation takes too much time, it pauses and it will be resumed in the next cycle. This module uses the risk estimation to move the arm without exceeding a certain risk threshold. If it is exceeded, the robot stops. The speed of the robot arm is always the same. Currently this system is effective avoiding collisions with the human. Modifications in this module to allow contact based cooperation are also part of this project.

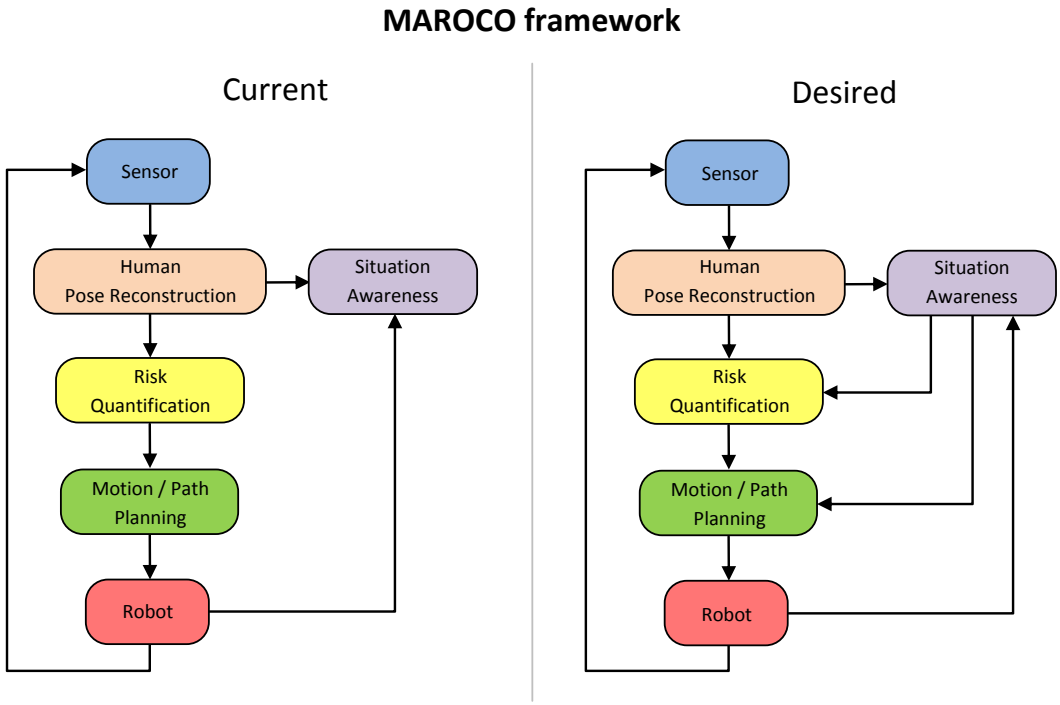


Figure 1.2: Current and desired flow among the modules of the MAROCO framework.

So, the main goal is to use all the information of these modules to achieve effective contact based human-robot cooperation. To do it is necessary to connect the situation awareness module with the risk quantification module (reducing the risk or increasing the risk threshold if the robot arm has to cooperate physically with the user, giving him an object, for example) and connect it with the path planning module to alter the path according to the situation. All these modifications have to be implemented in C++ for Windows in the research framework MAROCO. The solution has to work in real time and guarantee the safety at all costs.

Chapter 2

State-of-the-art

2.1 Human-robot cooperation

The word cooperation has a very wide meaning. In essence the main purpose of any autonomous robot is to cooperate with a human in some way to achieve a goal: directly through contact, or indirectly without human intervention during its operation. Almost all the current development is focalized on indirect cooperation because it is vastly easier to develop.

There are numerous familiar autonomous robots in service robotics and especially household robotics that prove that this field is interesting. Maybe the best example is Roomba, the popular robotic vacuum cleaner that is able to clean the floor without human intervention. Although its utility is out of doubt, its technological complexity is not too high because it has almost no concern about safety or contact cooperation. In fact many more complex prototypes like the dishwasher robot KAR [Mizuuchi et al. 2009] are not designed to work in a direct cooperative mode with a person. Only some prototypes like the Care-O-bot of the Fraunhofer Institute [Reiser et al. 2009] have achieved a certain level of contact based cooperation, in this case using a tray to rest objects and a tactile screen where the human can give him orders.

But even in the industrial area, where the situations should be some more predictable than at home, there are neither many contact based cooperation tries. Some of them use markers that the human must wear to ease its detection by the sensors [Sax 2004]. Its use undermines the concept of pure human-robot cooperation because the idea is that the human should interact with the robot without depending on any other tool or extern device.

A simple commercial markerless approximation has been developed by MRK Systeme [1]. Using tactile and capacitive proximity sensors and a soft surface, the robot arm stops when it detects a collision with the human or when it is about to occur. It is true that these types of solutions allow a person to interact in some way in the robot working area, but with the lack of a camera based vision system it cannot have a

situation awareness module to vary its behaviour nor a path planning algorithm to adapt itself to the human.

Another very specific solution without cameras is used in [Edsinger et al. 2007]. The hands of a humanoid robot are able to detect when a human has placed an object on them to grasp it and put it in another place. The system requires that the human gets used to manipulate the robot in this way to increase the grasping success. The communication with the robot is made in the contact phase, which prohibits the proactive part of the cooperation.

The most sophisticated systems use to include some sort of camera to determine the position of the human and its pose inside the working area. The company Pilz [2] commercializes a human-robot cooperation system based on safety fences and stereo cameras. This solution divides the working area in various 3D safety zones. However it is only able to detect if something strange is inside the predefined areas or not, no matter if it is a human worker, a machine or any other element. It is impossible to detect any meaningful situation because it cannot distinguish the human from other object types. Again, this system can only have reactive behaviours. Another similar work was developed by the Fraunhofer Institute. They use a time-of-flight camera in the robot working area [Winkler 2008] to handle three predefined region types and limit the maximal velocity of the robot, reducing the risk for the human.

In [Thiemermann 2005] the author uses CCD-cameras to follow the human hands through colour segmentation and uses a classic fuzzy logic system to determine the risk. Also, the maximum speed of the robot can be limited. The main problem is that it only takes the human hands into account, so this system could be already limited by design. Also, CCD-cameras are too sensitive to illumination conditions and probably are not the best type of sensor for a system that prioritizes safety.

But implement too many sensors can affect the costs negatively. More sensors can lead to more detection capacity and precision, potentially improving the general safety, but it has no sense if costs are too high. An example of this can be found in [Kulic 2005]. It uses a PUMA robot which includes a stereo colour vision system, an electrocardiograph and an electromyography. Independently of the success in the cooperative work that this robot could achieve, people probably find little profitable to spend all this effort integrating all these different sensors. The solution has to be as simple as possible to be useful.

Also, some authors employ a certain type of markers that are not necessarily artificial, like for example the skin colour, with all the disadvantages that this carries with it. This is the case of the VooDoo system [Lösch et al. 2009] that tries to reproduce the human pose through hand skin colour detection. Although this solution does not care about the occlusions between the sensors and the objective, it has a very slow response time so it is not adequate for a safety critical industrial robotic cell.

In [Heinrich et al. 2008] the authors use a system that identifies the pixels that belong to the robot, some foreground objects and the background. It also has a path planning

module that uses the wave propagation algorithm, however this system does not try to reproduce human kinematics so it is restricted to the avoidance of obstacles.

There are only a few works that deal with contact based cooperation, and many of them lack important parts like a situation awareness module or even a path planner. Also, only a few of them try to work with human kinematics to improve their chances of success. Some works use pseudo-markers like the hand colour and others use insecure type of sensors. Most of the works that have been developed are designed for a limited cooperation and do not try to provide a more global solution to the problem.

2.2 Path planning

There are many works over path planners applied to robotic arms. The path planner is responsible for planning a path for the robot between two positions, taking certain optimizations into account. It can be designed for two different types of environments. Dynamic environments are always changing and have some unknown variables, like objects in unknown positions or with unpredictable movements (a human in the case of this project). On the other hand, fixed environments are always stable and perfectly known.

Recently, probabilistic roadmaps (PRM) methods have been in the scope of the researchers in this area. PRMs are appropriated to be used in stable environments because they are based in the idea that the robot will be working in the same environment during large periods of time. A good example of this method appears in [Amato et al. 1998]. They use an offline phase to preprocess a roadmap in the configuration space of the robot using randomization. Each node in this graph is a collision free configuration. After that, the path is found in an online phase connecting the initial and final points with the roadmap and finding a course in it between these two connection points. Preprocessing requires high execution times but it does not matter because is done in an offline phase and the resulting representation will be used to find paths very quickly.

However PRMs are not effective in dynamic environments because in these situations it is not possible to extrapolate past solutions to the present, so Dynamic roadmaps (DRM) have been developed [Leven et al. 2002]. These methods are mainly based on PRMs but they also rely in a precomputed workspace mapping for fast invalidation of blocked roadmap parts. This provide a quick response to environment variations. This work affirms that its response time is less than a second, but according to [Kunz et al. 2010] an algorithm should work at 180ms at the most to consider that is real time, because that is the average reaction time of a person. To speed up the original DRM, they identify possible bottlenecks and apply some very specialized optimizations to get it run in less than 100ms per cycle.

But out of the roadmap solutions, there have been a number of path planning ap-

proaches in dynamic environments. The Ariadne's Clew algorithm [Mazer et al. 1998] operates generating landmarks in an exploration phase and connecting them to a existing network in the search phase. It can be tuned by changing the search phase or using different criteria to select candidate landmarks. In [Kindel et al. 2000] follow the idea of expanding a network from the start and the goal positions, making it grow until both positions are connected. Also, in [Vallejo et al. 1999] an adaptable approach that uses multiple local planners is described. At run time, characteristics of the problem are used to determine which combination of local planners is the best at each moment.

Chapter 3

Background

This chapter shows the fundamentals of the work. It is divided in two main sections: the risk quantification part and the path planner part. These modules have received the most of the modifications and its knowledge is essential to understand the following sections of this work.

3.1 Risk quantification

To allow the robot moving with a human in its working area it is necessary to control the risk of each movement in every instant. Several solutions can be done to get a risk value to evaluate the motion safety, but the best method implemented in MAROCO is the one that use a fuzzy logics system [Graf et al. 2010b]. All the changes in the risk quantification part have been done in this solution.

3.1.1 Fuzzy logic

The fuzzy systems allow working with graduated variables in which the limits of each degree are not exactly defined.

Unlike the traditional logic, where there can be only conclusions which are fully true or false, in the fuzzy logic is allowed a certain range of truth between 0 and 1 for each grade [Zadeh et al. 1996]. There can be partial truths. In this way it is possible to have a value considered true and false at the same time, but with different truth levels each one. This kind of logic is also many-valued. A numeric variable can be described in linguistic terms to use more than two grades. For example, temperature can be graded as *cold*, *warm* and *hot*. These values can be controlled by specific functions.

In Figure 3.1 the black vertical bar represents a certain temperature. It can be considered as no hot, lightly warm or very cold. This process of converting a crisp temperature

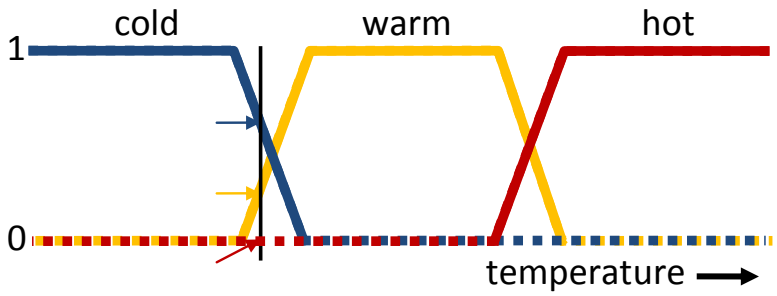


Figure 3.1: Fuzzy variable temperature example.

into membership values for each grade of a fuzzy variable is called fuzzification.

Fuzzy logics are based in heuristic rules with the form IF (antecedent) THEN (consequence), where the antecedent and the consequence are also fuzzy variables.

The inference methods for these rules must be simple, fast and efficient. The results of these methods are a final area, which is the result of a group of overlapped areas among them (each area is the result of an inference rule). To choose a certain output among all of these fuzzy premises, the most used method is the centroid, in which the final output is the centre of gravity of the resulting total area. This process is called deffuzification (Figure 3.2).

These rules of a fuzzy system can be formulated by experts or learned, using neuronal networks in this case to make the future decisions better.

Input data are normally collected by sensors, which measure the input variables of a system. The inference engine sometimes is based in fuzzy chips, which are increasing their processing capacity exponentially each year.

3.1.2 Double threaded

But in this work, the risk quantification module is a double threaded fuzzy system. It means that it has two rule groups: positives and negatives. This allows reducing the amount of rules and simplifies the rule set. Positive rules are the classical ones and are always in any fuzzy system. An example of positive rule could be:

- `IF Temperature = Medium`
  `THEN Fan_speed = Low`

And this could be a negative rule.

- `IF Temperature = Very_cold`
  `THEN Fan_speed = High OR Medium UNWANTED`

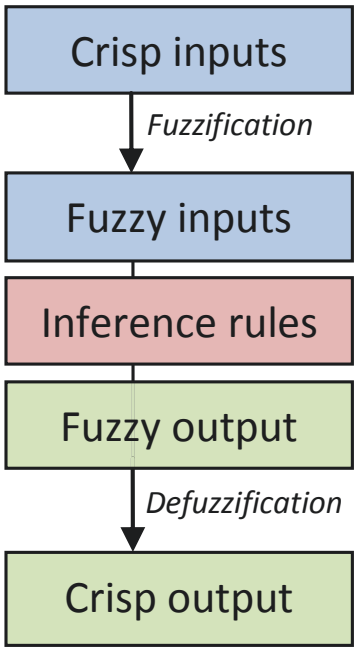


Figure 3.2: Fuzzy system flow.

To take both kinds of rules into account a new hyperinference step is used to limit the fuzzy output values depending on the negative inference rules. There are several ways to do this. The veto operators represent the core of the hyperinference. The classical ones are the strong veto and the weak veto. The strong veto operator is formally defined in the Equation 3.1.

$$\mu(u) = \begin{cases} \mu^{+}(u), \text{ if } \mu^{-}(u) = 0 \\ 0, \text{ otherwise.} \end{cases} \tag{3.1}$$

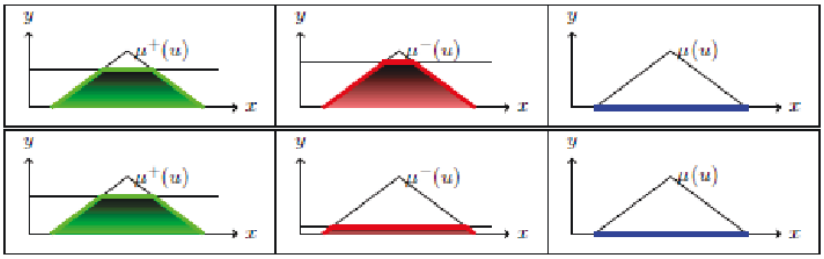


Figure 3.3: Response characteristic of the strong veto operator.

The strong veto operator does not respond to the area under the activated positive rule (Figure 3.3). For this reason, the area under the negative rule is weighted too much,

which means that when the area is not small enough a veto will be generated.

One great advantage of the two threaded fuzzy logics lies in its great flexibility, but it is ignored with the strong veto operator, showing a suboptimal performance when is connected to a path planning module or controlling the robots velocity.

$$\mu(u) = \begin{cases} \mu^+(u), \text{ if } \mu^+(u) \geq \mu^-(u) \\ 0, \text{ otherwise.} \end{cases} \tag{3.2}$$

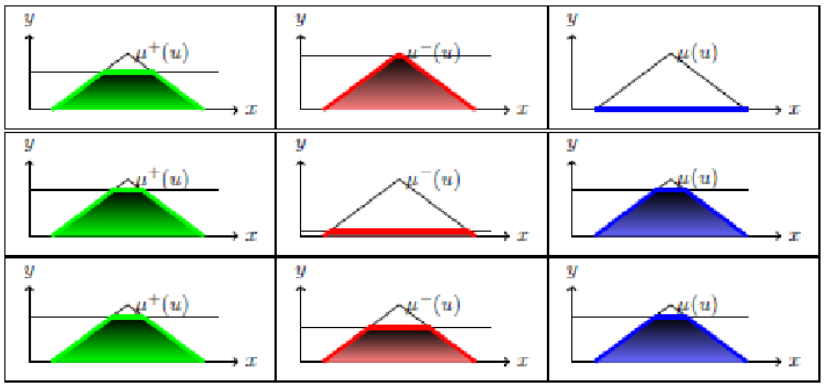


Figure 3.4: Response characteristic of the weak veto operator.

The weak veto is another classical operator which behaves differently. The area under the positive rule in Figure 3.4 is smaller in comparison to the area under the negative rule. For this reason, a veto is applied, which is desired for this special case. The positive rule generates a greater area compared to the negative rule, independent to the difference of both areas. The output is equivalent to the area under the positive rule for all possible cases. This means that the negative rule does not have any influence on the output for this special case, which is really not desired.

$$\mu(u) = \begin{cases} \mu^+(u) - \beta^-(u), \text{ if } \mu^+(u) > \mu^-(u) \\ 0, \text{ otherwise.} \end{cases} \tag{3.3}$$

To solve this suboptimal approximations, the fuzzy system used in this work implements a novel hyperinference veto, which is formally described in Equation 3.3. As is shown in Figure 3.5, this veto subdivides the area under μ^- into three parts. At first, the cut of the curve is determined according to the output of the activated negative rule. Then, an orthogonal line is generated (bottom row). This defines three parts of the area under the operator. The outer area elements are identical due to the symmetric characteristic of the operator and described by β^-. The adequate output of the veto operator is then generated by $\mu^+ - \beta^-$.

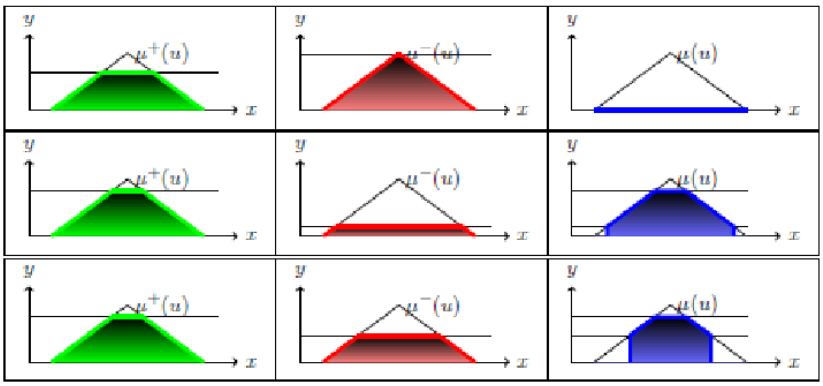


Figure 3.5: Response characteristic of the novel veto operator.

3.2 Path Planner

The task of the path planner is to find a path for the robot to move it from a starting point to another. In this case, the path planner must allow the system working in real time, so a method to pause and resume the path search is a good idea. There are many different systems that try to solve this problem, normally in the configuration space of the robot (the rotation angles of each degree of freedom). A* is an algorithm widely used for its performance and accuracy.

3.2.1 A* algorithm

The family of the informed algorithms against the uninformed ones or by brute force are those that have an extra information about the structure of the studied object, which they exploit to reach their final objective faster, with a minimum cost path from the start point until the end point.

Informed search use the specific knowledge beyond the problem definition itself, which can find solutions more efficiently than a non informed strategy, very inefficient in the majority of the cases.

The problem of some informed search algorithms in complex structures, like the greedy algorithm, is that they are guided exclusively by the heuristic function, which could not indicate the path with lowest cost, or just by the real cost to move from one node to another (like the hill climbing algorithm), so sometimes is needed to do a higher cost movement to reach the solution. Because of that, a good informed search algorithm should take both factors into account, the heuristic value of the nodes and the real cost of the path.

The most widely known form of the Best-First search is called A* (A star search) [Hart et al. 1968]. It evaluates the nodes combining $g(n)$, the cost to reach that node, and

$h^*(n)$, the estimated cost to the objective node: $f^*(n) = g(n) + h^*(n)$.

Having $g(n)$ that is the cost of the path from the initial node to the node n, and $h^*(n)$ that is the estimated cost of the cheapest path from n to the objective, $f^*(n)$ is the cheapest cost of the solution through n.

In this way, to find the cheapest solution, is reasonable to try first with the node with the lowest value of $g(n) + h^*(n)$. If the heuristic value of $h^*(n)$ satisfies certain conditions, A* search is complete and optimum.

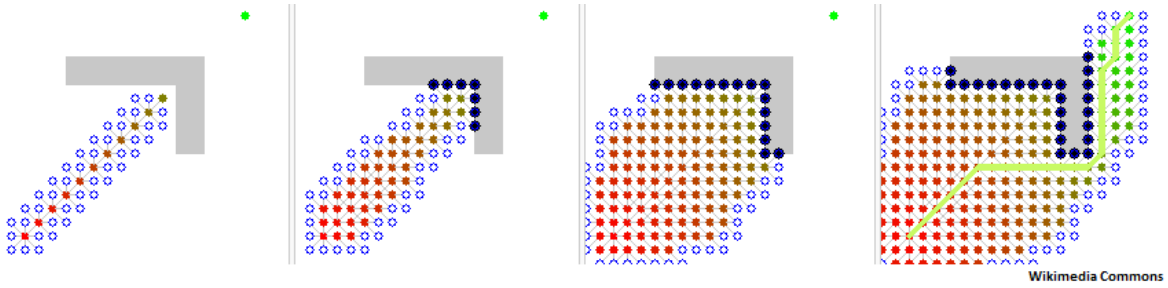


Figure 3.6: A* finding path in a robot motion planning problem.

In Figure 3.6, A* finds a path in a robot motion planning problem. The empty circles represent the nodes in the open set, and the filled ones are in the closed set. The colour on each closed node indicates the distance from the start (greener means farther). A* starts moving in a straight line in the direction of the goal, then when hitting the obstacle, it explores alternative routes through the nodes from the open set.

The optimality of A* is easy to analyze if it is compared with the tree search. In this case, A* is optimal if $h^*(n)$ is an admissible heuristic, which means that $h^*(n)$ never overestimates the cost to reach an objective. Admissible heuristics are optimistic by nature, because they think that the cost to solve the problem is lower than in reality. $g(n)$ is the exact cost to reach n, so as an immediate consequence $g(n)$ never underestimates the real cost of a solution through n.

An obvious example of an admissible heuristic is the distance in a straight line h^* that is used to go to the destination of a travel. The distance in a straight line is admissible because the shortest path between two points is a straight line, so the straight line cannot be an underestimation.

The performance of the heuristic search algorithms depends on the heuristic function quality. Good heuristics can be constructed sometimes relaxing the definition of the problem, by precalculated solution costs for sub problems in a data base model, or by learning from the experience with these classes of problems.

Also, while the admissibility criterion guarantees an optimal solution path, it also means that A* must examine all equally meritorious paths to find the optimal path. It is possible to speed up the search at the expense of optimality by relaxing the admissibility criterion [Pearl 1984]. By bounding this it is possible to guarantee that

the solution path is no worse than $1 + \epsilon$ times the optimal solution path. This new guarantee is referred to as ϵ - admissible (Figure 3.7).

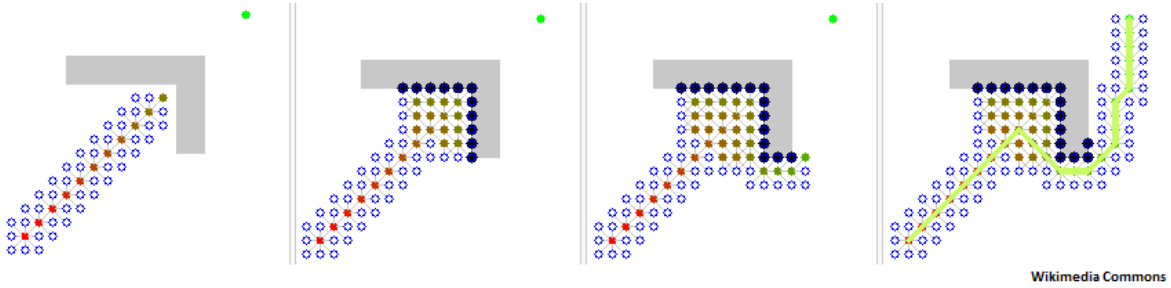


Figure 3.7: A* with a bounded relaxation ($\epsilon = 5$).

The formalized algorithm for A* is in Algorithm 1. Algorithm 2 recreates the optimal path using the information of the nodes.

Algorithm 1: A*

```
Input: start node
Input: goal node
Output: solution path
closedset ← the empty set                          // Nodes already evaluated
openset ← start                                    // Tentative nodes to be evaluated
came_from ← the empty map                          // Map of navigated nodes
g_score[start] ← 0                                 // Cost from start along best known path
f_score[start] ← g_score[start] + heuristic_cost(start, goal)
while openset is not empty do
    current ← the node in openset having the lowest f_score[] value
    if current = goal then
        return reconstruct_path(came_from, goal)
    remove current from openset
    add current to closedset
    foreach neighbor in neighbor_nodes(current) do
        if neighbor in closedset then
            continue
        tentative_g_score ← g_score[current] + dist_between(current,neighbor)
        if neighbor not in openset or tentative_g_score < g_score[neighbor] then
            if neighbor not in openset then
                add neighbor to openset
            came_from[neighbor] ← current
            g_score[neighbor] ← tentative_g_score
            f_score[neighbor] ← g_score[neighbor] + heuristic_cost(neighbor, goal)
return failure
```

Algorithm 2: reconstruct_path

```
Input: came_from
Input: current_node
Output: path
if came_from[current_node] is set then
    p ← reconstruct_path(came_from, came_from[current_node])
    return (p + current_node)
else
    return current_node
```

3.2.2 Current Path Planner

The path planner is divided in an initial offline phase and an subsequent online phase [Graf et al. 2009]. The offline phase takes place at the beginning of the process. During this, a graph in the robot configuration space is constructed. Some discrete positions in the configuration space will serve as vertices of the graph. Vertices are connected between them using a k-nearest neighbour algorithm. After that, a path between the start node and the goal node is found using the A* search algorithm.

During the online phase the path is traversed. In each execution step, the path planner creates the new intermediate configuration of the robot by the interpolation of the path found. Later, this is sent to the robot for its execution. The planner reactivity is done in this phase too.

It is needed to do constant checks to evaluate if there are imminent collisions, and if there are, a new path replan is triggered. It is not safe to check only the next interpolated configuration, although it is not necessary to interpolate the full path each time to check every possible collision. The look-ahead function checks only some of the next nodes. These collision tests are done discretely along the full path, so the discretization can cause the missing of an obstacle that is too small for the discrete steps (Figure 3.9). To avoid this problem a collision is detected if the obstacle is into a safety clearance zone. Due to the human pose modelization, the size of the obstacle is known and the safety clearance can be set empirically.

The online path planning uses the graphs constructed in the offline phase. To not exceed the time thresholds and allow the system working in real-time, is needed to modify the default behaviour of A* to take this restriction into account, because the original algorithm uses as much time as is needed to find the optimal path. The modification is the pre-emptive A* algorithm. If the search takes more time than a predefined threshold, it pauses and can be resumed in the next program cycle. The A* algorithm classifies the graph nodes in two separated lists: the open list and the closed list. It is necessary to save these lists in order to continue the searching (Algorithm 3).

Due to the online phase, the person can move freely and without restrictions. As a result, the search space can be very limited, because the new freed configurations are not considered but the new colliding ones are discarded. The search can be slow. The resulting path is pruned to not adjusting perfectly to the actual search space. In these cases, the search is aborted after a predefined timeout and restarted again.

The Euclidean distance is used as a metric to calculate the heuristic value and the cost of each configuration. This is done for each node during its parent node expansion. Also, this is extended to a risk check in each evaluated node. In this way, configurations that have a wide separation distance can be rejected because are considered unsafe. When the person is approximating fast, it is possible that the current robot pose is considered risky, in this case, it is senseless to do a path replanning because any movement could mean danger for the person. The robot continues in its pose until the

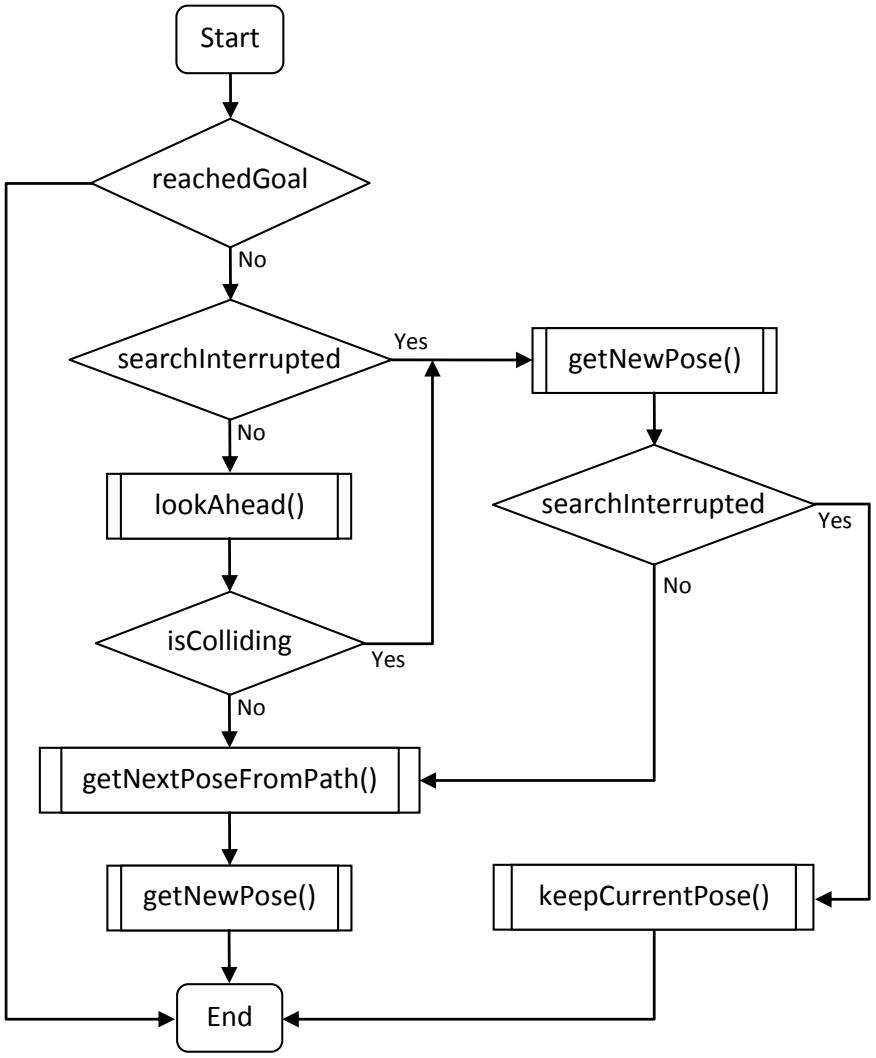


Figure 3.8: Path traversing with pre-emptive search.

human is out the risk area.

The search space is restricted to three dimensions, that are the main degrees of freedom, which are the first three joints. The rest of the configuration space dimensions do not have a major impact in the ability to avoid collisions between the robot and the human.

In addition, the first configuration space dimension can be extended cyclically. Doing this, the robot can move more than 360 degrees around the joint of its base. The opposite side of the human is always the safest in this case.

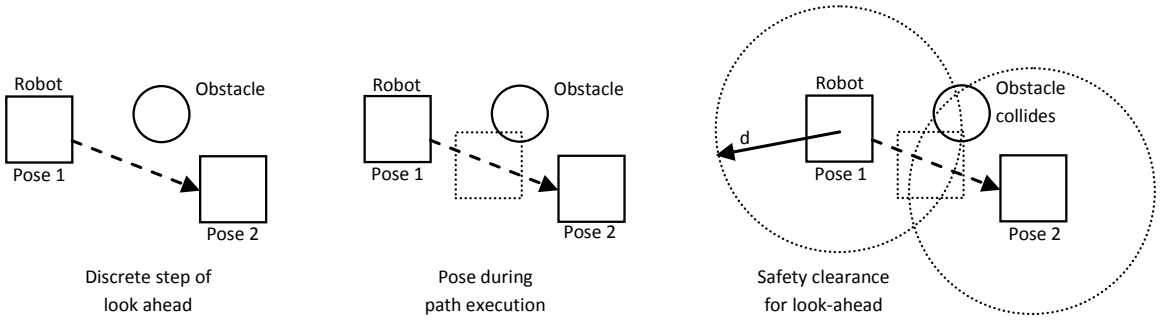


Figure 3.9: Look-ahead with the safety clearance.

Algorithm 3: preemptiveAStar

```
Input: start_node
Input: goal_node
Input: isNewSearch
Output: hasFoundGoal
Data: closedset
Data: openset
Data: node
if isNewSearch then
    init(openset, closedset, start)
else
    load(openset, closedset)
startTime ← Now()
hasFoundGoal ← false
while openset is not empty do
    if Now() - startTime > threshold then
        save(openset, closedset)
        break
    node ← best (openset)
    if node = goal then
        hasFoundGoal ← true
        break
    expandNode(openset, cloasedset, node)
return hasFoundGoal
```

Chapter 4

Implementation

The work is divided in three main parts. The first one consists in a modification of the risk fuzzy logic system to get a reduced risk value when the robot has to cooperate with the human. The second part is the changes and improvements to the path planning module to plan the robot behaviour according to certain situations provided by the situation awareness module. The last part consists in the addition of a velocity control to the path planner to allow reaching some dangerous nodes that otherwise were inaccessible.

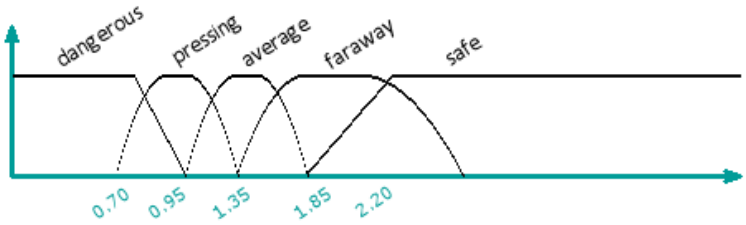
4.1 Risk quantification

The original fuzzy system risk quantification [Graf et al. 2010b] demonstrated that it is useful avoiding the human, but it cannot manage situations that require physical cooperation. The path planner takes the risk (a value between 0 and 1) and evaluates if a node is safe or not to go through. Giving an object to a human, for example, raises the risk value in the original risk quantification when the distance gets near to the human. The path planner interprets that it is impossible to traverse through these nodes because the risk is too high, stopping the robot motion or replanning its path. It is needed to modify this system to have a meaningful risk that allows doing all the necessary actions for a cooperative work, always guaranteeing the safety.

4.1.1 Considerations of the original fuzzy system

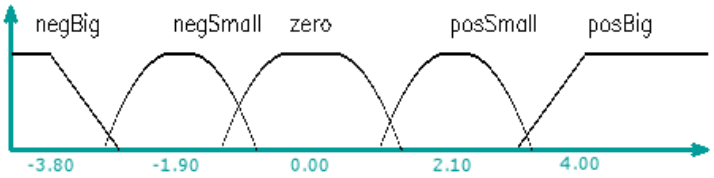
The technical report of this system [Czapiewski et al. 2010] has just a little information about the nature of the implemented fuzzy variables and sometimes does not match exactly with the provided graphs. These are the current graphs of the fuzzy variables based in the display of the MAROCO framework.

Distance:



Represents the minimum distance between the robot and the human. Is measured in meters.

Velocity:

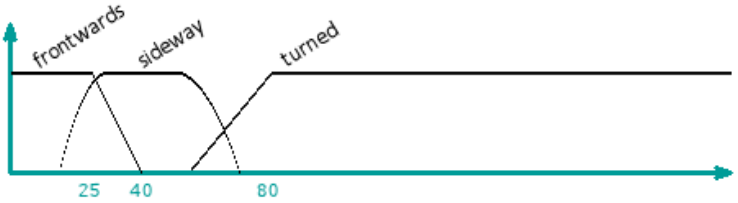


Represents the relative velocity between the robot TCP and the human. It is measured by deriving the distance to the human with respect to time in each cycle. It assumes that the time lapse between each cycle is 1/20 of a second and does not consider the angular velocities of the robot axis.

Head orientation:

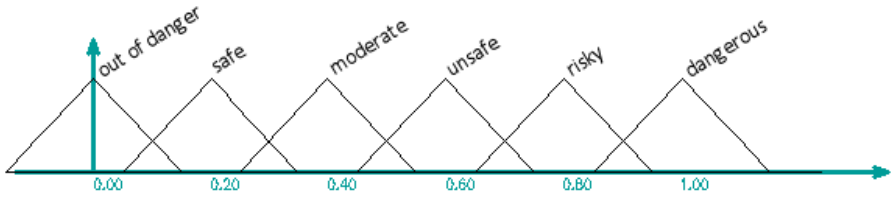
Measured in degrees between the human head and the robot. Useful to determine if the human is looking to the robot or not.

Body orientation:



Measured in degrees between the human body and the robot.

Risk:



Is the fuzzy output value of the risk, from 0 to 1.

The inference rules are well documented except three. This positive rule is omitted:

- `Rule 19: IF Head_orientation = 'frontwards' AND Distance = 'faraway' OR 'average' OR 'pressing' THEN Risk = 'outOfDanger' AND 'safe'.`

This rule is the only one that applies two different risk values. The other different negative two rules are the 16 and 18. Both have `'unsafe'` as an additional unwanted risk value.

So the positive rules are from 0 to 14 and the number 19. The negative ones are from 15 to 18. For this work it has been assumed that this system works well and it has been tried to modify it as little as possible to achieve the new functionality.

4.1.2 New fuzzy variables and rules

To make the robot working cooperatively, it is needed to analyze which parts of the original system are useful or not. The previous fuzzy system works well avoiding the human if the robot has to move without physical cooperation. This behaviour will continue being necessary in some cases, so for these moments it is interesting to maintain the same functionality that has been tested and gave good results.

In a first moment, the addition of two more degrees to the relative velocity fuzzy variable to control the cooperative work was considered, but finally this was discarded because the values *posSmall*, *zero* and *negSmall* are slow enough to allow cooperation. Also this is a relative velocity measure, so it is not convenient to have too fine granularity on it because the human also moves and varies this value. If the relative speed varies too fast, the risk could be constantly fluctuating and neither the path planner could have time to finish its calculations nor the robot to traverse the path before a forced replan.

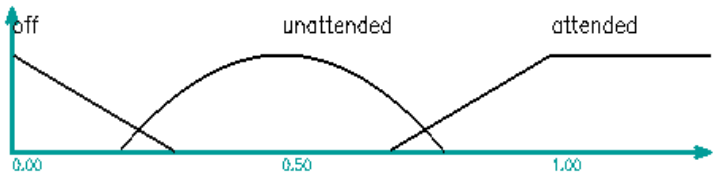
The original negative rules are interesting because they could be used to manage a certain cooperative behaviour, but the current ones are not enough to control it com-

pletely. There are more data to be considered to achieve the cooperation, so it was needed to program two new fuzzy input variables.

- Variable to indicate if a cooperative behaviour is needed or not: with it, the risk can be reduced only if the robot has to cooperate. The situation awareness module and the path planner will be responsible to modify its crisp value.
- Situation awareness delay time: the situation awareness module does not work as fast as the rest of the framework, so its output about the current expectation can be outdated. When the last detection is too old, the risk should raise.

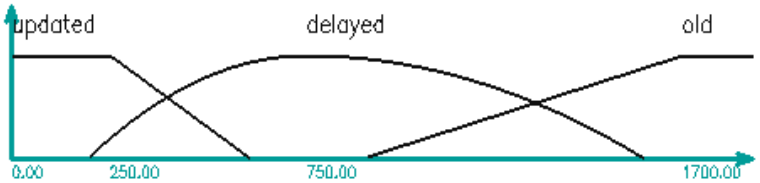
As will be shown in the Experiments section, these additional two fuzzy variables are enough to reduce the final risk depending on the current situation.

Cooperation mode:



Off indicates that cooperation with the human is not needed.
Unattended means that the robot can cooperate with the human without need to be looking directly at it (leave a beverage near a human that is reading something).
Attended means that the human must look to the robot to cooperate (give an object to a human)

Situation delay:



The milliseconds that have been passed since the last output of the situation awareness module.

To make the fuzzy system to consider these variables it is necessary to add new inference rules and/or modify the original ones. To do this, two preliminary ideas were developed before reach the third and definitive one. In each idea evolution, the original code has

been more respected and the risk has been modified in a more indirect and abstract way. The identifier numbers of the new rules are from 20 and on.

Discarded idea 1: Modification of the original rules.

It consists in adding new positive rules and modifying slightly the original ones to allow the cooperative use. It was necessary to add the condition that when the cooperation mode is off, the original rules behave as before. This condition was added to the rules 7, 8, 9, 12, 13.

Excepting the previous condition, the new rules coincide in everything with the original ones but they reduce the risk level when the cooperative mode is on.

- `Rule 07b: ...THEN Risk = 'safe'`
- `Rule 08b: ...THEN Risk = 'safe'`
- `Rule 09b: ...THEN Risk = 'moderate'`
- `Rule 12b: ...THEN Risk = 'safe'`

The rule 13 was divided in two. One for the case when the relative velocity is *negBig* so that the risk continues being *dangerous*. The other was for the case when the velocity is *negBig* or *negSmall* to reduce the risk to *safe*.

Technically this solution was correct and in the practice worked well, although it only took into account the cooperation mode fuzzy variable. However it forced to use too many rules (six more just for this functionality). It would be necessary to add even more rules and change the original ones in order to expand this proposal to control the situation delay. With this idea implemented, the average speed of the MAROCO was not affected too much but at that time, a better second idea was planned.

Discarded idea 2: Addition of new negative rules.

The second idea was very convenient because it takes advantage of the double threaded fuzzy system to economize rules and maintain the original ones unmodified. It consisted in adding negative rules to limit the risk if the cooperative mode was on. Each new negative rule had conditions to control the cooperation mode; if the human was looking or not, etc. With just two rules was possible to achieve all the previous functionality in a much more indirect way, just by limiting the risk.

However, this idea had also some problems. Sometimes the applied veto of the hyperinference did not prioritize the cooperation mode variable and, in consequence, the risk sometimes was too high to allow an efficient cooperation. At this point the last solution was planned, the double output.

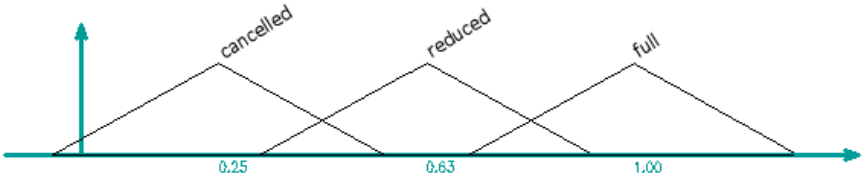
Current solution: The double output.

The previous ideas modified the original risk value and even the original rules. Adding future conditions could ruin the balance achieved by the original author and disadvantage the functionality. The best solution was to leave the original risk value as it was designed.

In fact, the basic modification in the fuzzy system needed to allow the cooperation is a risk reduction. Taking this premise, it is possible to develop a simple parallel fuzzy system to generate another output variable, a risk reduction factor. This solution was the chosen one. The reduction factor is a value between 0 and 1 that is multiplied to the original risk. A lower multiplier value leads to a lower risk value. If the cooperative behaviour rules are activated, the multiplier approaches to 0.25, reducing the original risk.

The minimum value of the multiplier has been set in 0.25 to avoid cancelling the risk completely and maintain always some information about the original risk system. If there is no need to cooperate, the value will be 1 and the original risk will be fully applied. This method is totally independent of the original fuzzy system, maintaining it unmodified and working only in the reduction factor.

Risk reduction:



The fuzzy output risk reduction value, from 0 to 1.

Three positive new rules had to be implemented in the parallel fuzzy system:

- `Rule 20: IF Cooperation_mode = 'unattended' THEN Risk_factor = 'cancelled'`
- `Rule 21: IF Cooperation_mode = 'attended' AND Head_orientation = 'frontwards' THEN Risk_factor = 'cancelled'`
- `Rule 22: IF Situation_delay = 'delayed' THEN Risk_factor = 'full'`

And three negative rules:

- `Rule 23: IF Situation_delay = 'old' THEN Risk_factor = 'cancelled' OR 'reduced' UNWANTED`

- `Rule 24: IF Cooperation_mode = ‘off’ THEN Risk_factor = ‘cancelled’ OR ‘reduced’ UNWANTED`
- `Rule 25: IF Velocity = ‘negBig‘ OR ‘posBig’ THEN Risk_factor = ‘cancelled’ OR ‘reduced’ UNWANTED`

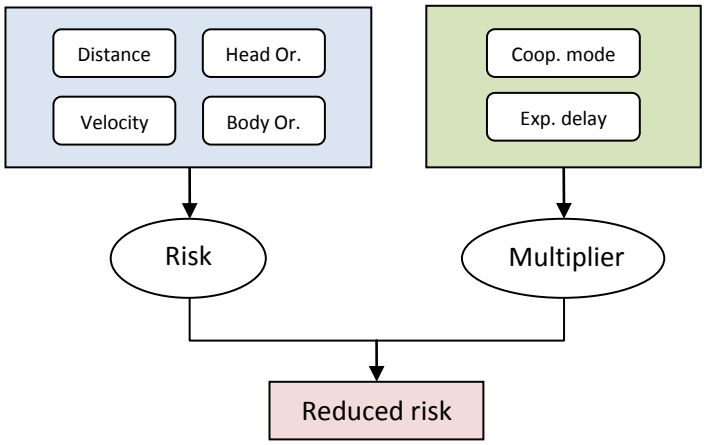


Figure 4.1: Flow diagram of the double output risk fuzzy system.

The diagram of the Figure 4.1 represents the fuzzy variables and how the two outputs are calculated. Having the risk and the risk multiplier factor, the reduced risk is obtained by multiplying both values. A* and the motion planner will use the reduced risk value. With all of these modifications, the fuzzy engine was completed, but there is one more improvement to make it working well.

As can be seen, the risk reduction depends on the cooperation factor, the relative speed, and the expectation delay. In the experiments, the cooperation factor and the relative speed worked well, but the expectation delay needed more development.

4.1.3 The expectation delay processing

Discarded idea: Instant expectation delay.

The first way to get the expectation delay was to measure the age of the last prediction it in each cycle and send it to the fuzzy risk quantification module. Although it seems reasonable, this solution has a very big problem as is shown in Figure 4.2.

The Figure 4.2 shows how the risk multiplier factor is affected by the latency of the situation awareness module (the expectation delay). If the last situation prediction becomes too old (more than 250ms), the risk is less reduced by the risk multiplier factor. This Figure shows the risk multiplier changing too fast because the delay takes the immediate value, not the average.

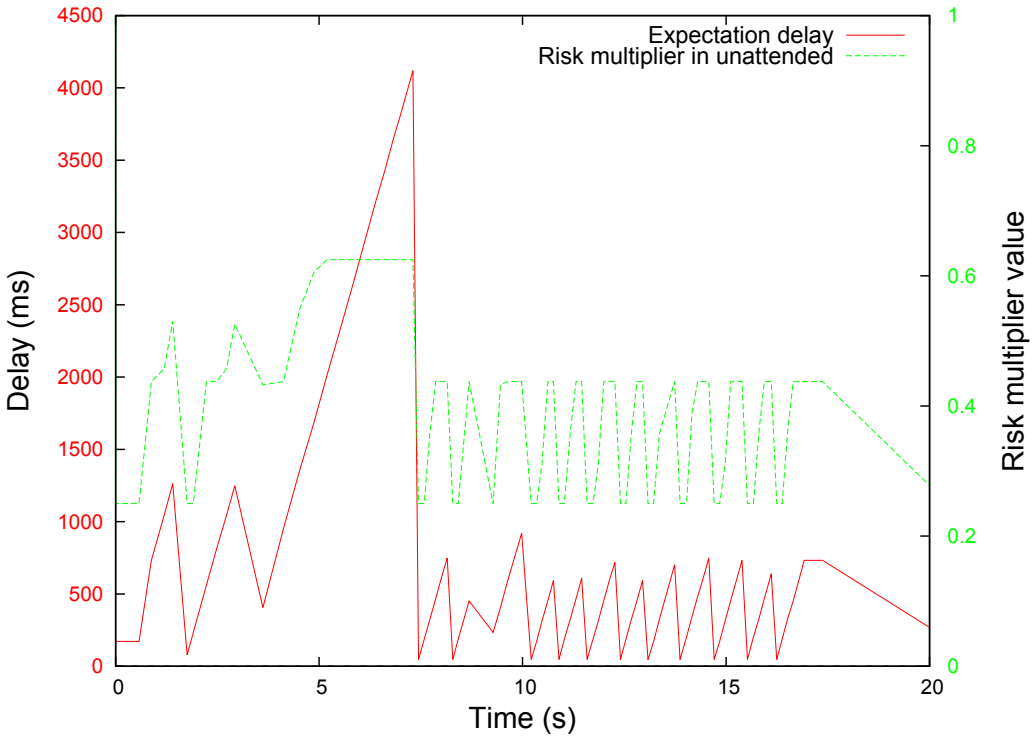


Figure 4.2: Risk multiplier depending on the instant expectation delay.

It is true that with a faster computer the expectation delay can be always *updated*, but this is a nonsense because if it never moves from *updated* there is no need to use this variable in a parallel fuzzy system. Also it is absurd that A* depends on a risk value calculated using an instant delay value because the path which it will plan will take much longer to be traversed that this amount of milliseconds. The instant delay will change completely while the robot is moving to the goal node.

Current solution: Average expectation delay.

The current form to calculate the expectation delay is to do the average of the previous 10 maximums. This allows a much more stable risk reduction value with more meaning. Additionally, A* can construct its paths using this value because the expectation delay tends to be always the same for a certain computer. The value of 10 was deduced empirically in the Experiments section.

There is only one exception: sometimes the situation awareness takes too much time to do a detection and this delay is not taken into account to do the average because it has not reached its maximum yet. So, if the instant delay is higher than the average value, the instant value is passed to the risk quantification module. This method allows minimizing the expectation delay variations and also considers always the most dangerous value of the delay. The graph of this solution is in the Figure 5.3.

These are the all the modifications in the risk quantification system. In the following sections will be explained how the speed is considered and limited according to the cooperation mode. This makes intensive use of the described risk quantification module. Also, there are more changes involving the file `CRiskQuantification.cpp` but all of them are part of the Velocity Control section.

4.2 Task planner and target paths for cooperation

In the original system, the robot path was always the same, just moving in one direction and another (see Figure 5.4). In this work, this behaviour has been considered the default task, the main job that the robot has to do. But the situation awareness module has several outputs, and some of them are very interesting for the cooperation purposes. Especially the expectation output is the most useful. In this work, two of its values have been considered, requiring different cooperation levels: *Get Beverage* and *Get Object*.

- *Get Beverage*: When this expectation is detected, the human normally is sat on a chair, near a table. He can be reading something, or just doing some other operation, not looking at the robot. The robot has to leave on the table a glass of beverage, for example water, and near to the human. When the beverage has been leaved, the robot should continue doing its job, which is the default task.

- *Get Object*: When this expectation is detected, the human wants that the robot gives him an object or tool. The robot has to give this object to the human, doing a very deep movement inside the human working area while the human is looking at it. When the human gets the object gripped by the robot, the arm has to return to its default motion.

It is clear that the cooperation mode is different in the two situations, in one the human do not have to be looking but the other requires it. That is why the cooperation mode has three fuzzy values. In fact, the current fuzzy risk system supports cooperation values between *off* and *unattended*, or between *unattended* and *attended*, reducing the risk in a more soft way. This can be useful for future works which want to program the behaviour for other situations that require different cooperation levels than 0, 0.5 and 1, respectively. But this is out of the scope of this work.

In both cases the required task cannot be done in the original MAROCO system. The risk constraint is too strong and A* will never find a path safe enough. With the updated risk quantification module it is necessary to program these tasks to cooperate with the human. This work is focused on the motion planning, so the programmed behaviour is just a simulation. It does not consider the real object positions, just limits to the robot and human using the distance and the risk as a guide to evaluate

nodes. But in despite of everything, some very basic intelligence has been developed too. These are the programmed tasks for each expectation:

- Behaviour for *Get Beverage*: (Figure 5.5) The robot aborts any previous task and moves to simulate the beverage gripping in a static point situated on a table. This point is always the same. After that, the robot moves to the other table and leave the beverage in a static point, near the human. This movement is done with the *unattended* cooperation mode. The points are always the same, but it is sufficient to see if the movements to do the cooperation continue being forbidden or not. After that, the robot returns to the default task with the cooperation mode off.
- Behaviour for *Get Object*: (Figure 5.6) As in the other task, the robot aborts any previous task and moves to simulate the object gripping in a static point situated on a table. The next step is to give the object to the human, moving the arm in a fixed point of the space, comfortable and safe for the human to grip it. This movement is done with the *attended* cooperation mode, so the human should be looking at the robot. When the object is gripped by the human, the robot returns to its default task, without cooperation mode. The robot and MAROCO have not any form to guess if the human has gripped an object, so this sensor is simulated by pressing a key (this is the only "manual" action that has to be done in the final system).

The programmed tasks take also into account if the robot has already gripped the beverage or the object. If the robot has already gripped the required object, goes directly to leave it or give it depending on the task. On the other hand, if the robot has an object that is not required, it moves to the gripping point to put it again in the same place where the robot gripped it. These two situations can occur if the task is aborted or restored again during its execution.

At this point, the programmed system has many tools to allow the cooperation, but has a very big handicap too: the speed is always the same. With this problem in mind, before this point there have been some tries to control it, but all of them are going to be explained in the next section for clarity reasons.

4.3 Velocity control

The robot speed is a critical value in any contact based cooperative system between a human and a robot. It is determinant for the global safety and the original MAROCO system does not consider any change in it. The speed is always the maximum allowed in the configuration of the MAROCO and in the `AStar.cpp` file.

The lack of a velocity control leads to missing useful paths that could be traversed at lower speeds. It is true that the risk quantification module takes the relative speed into

account, but there is no way to reduce it or guess the maximum allowed value for a node. Also, the measured speed is the relative between the robot and the human, which means that the robot can be moving fast if the human is moving with it. Before explaining how these problems have been solved, it is important to remark some peculiarities of the original path planner.

4.3.1 Considerations over the original path planner

First of all, A* and the motion planner works independently. A* designs the path depending on the environment in the moment in which the path is being planned. The motion planner follows the path planned by A* and triggers a path replanning if the current Look Ahead function considers that the rest of the path is not safe enough. This works well, but if the human moves away and there is a safer environment, the path will be the one previously planned by A* in every case, even if with the new environment there are faster and more optimized paths that A* could find.

Second, in the path planner implemented in the used MAROCO version, the Look Ahead function does not work well. The purpose of this function is to check if a certain number of nodes after the current one are safe or not. To do that, it evaluates a safety clearance for each node considering the distance and the risk. It works perfectly for the distance, but there are problems if the risk is considered.

The motion planner asks for a replan if the Look Ahead function detects that a future node exceeds the distance threshold or the risk threshold. The original system sometimes freezes because the planned path is generated always inside the safety clearance of the Look Ahead. This can occur because there are nodes with a "controlled approach" flag activated. Between them the path is not generated by A*, is just a straight line. This path section could be prohibited by A* and, in consequence, by the Look Ahead function. In this case the motion planner asks constantly for a replan, and the replanned path is always the same, in a loop.

But even without nodes with this flag activated, in the original system the path sometimes is replanned without apparent reason. This was tested experimentally with a static human, always in the same position. With this situation, the risk in the following nodes should be the same that A* calculated, because there was not any change. The risk factor of the safety clearance had to be disabled to allow an effective motion. In despite of everything, with the new additions to the system, the robot can move with safe paths and replan the path if the human moves too near.

Taking the risk into account in the Look Ahead can be very good, but some factors considered in the risk evaluation can vary very quickly (head orientation and expectation delay for example) causing unnecesary path replans and undermining a successful cooperation. As a first approach, considering just the distance in the Look Ahead function is sufficient as is showed in the Experiments section.

4.3.2 Obtaining the instant velocity of the TCP

The ISO10218-1 rules indicate that a robot working in "reduced speed mode" cannot exceed the speed of 250mm/s in its TCP. To comply with these rules, the main problem is that the risk quantification system only uses the relative velocity between the robot and the human, and there is no place in the MAROCO framework to get this value.

The calculations to obtain this velocity are not trivial. One way is to check the angular velocity of every degree of freedom and use trigonometry to obtain the instant speed in m/s. This solution is valid, although is difficult to program it.

The author of the fuzzy risk system calculates the relative velocity by deriving the distance between the robot and the human with respect to time in each cycle. This is a very easy calculation and a similar method can be used to calculate the instant speed because the MAROCO can give the coordinates of the TCP. With two points it is possible to calculate the distance which divided by the elapsed time gives the instant velocity. Also, in the risk quantification system is assumed that the time elapsed between each cycle is 1/20 of a second. This makes sense because the risk will be considered in A* too, and the time between each cycle have to be extrapolated into future nodes. The calculation of the instant velocity uses also this method.

So to calculate the instant velocity is necessary to have the previous and the current node, get the distance between them and divide it by the 1/20s approximation, as is done with the relative velocity in the risk quantification module.

4.3.3 Limit the maximum speed in a cooperative mode

Once the instant speed has been obtained, the maximum velocity of the robot can be limited. Following the ISO rules, this speed cannot be higher than 250mm/s in a "reduced speed mode". In this work, the *unattended* and *attended* cooperation modes have been considered "reduced speed modes".

Discarded idea: Instant velocity as a fuzzy variable.

A first approach to solve this problem was to consider the instant velocity in the risk quantification module and later process the obtained risk to calculate the maximum speed. So it was needed to program a new fuzzy variable.

Robot velocity:

The linear velocity of the robot TCP in meters per second.

One negative rule can achieve all the required functionality by limiting the risk to 'dangerous'. The target here is to increase the base risk at maximum if the speed is higher than the mentioned in the ISO rules. This rule has to be implemented in the original fuzzy system because the new small parallel system can only do risk reductions.

- `Rule: IF Cooperation_mode = 'unattended' OR 'attended'`
  `AND Instant_velocity = 'medium' OR 'fast'`
  `THEN Risk_factor = 'outOfDanger' OR 'safe'`
  `OR 'moderate' OR 'unsafe' OR 'risky' UNWANTED`

The problem with this solution is that is too complex, and sometimes the hyperinference veto does not give the sufficient priority to the negative rule, causing that the risk have not the maximum value when the velocity exceeds 250mm/s in a cooperative mode. The current solution is much simpler.

Current solution: Set the speed to 250mm/s.

If the speed is higher than 250mm/s when a cooperation mode is activated, the target speed is set to this maximum allowed value. The risk quantification module does not need more considerations about the TCP speed, so consider new fuzzy variables and rules just complicate the system. Also, with the current solution the base risk value remains unaltered.

To do this, it is important to remark that the instant velocity is calculated in m/s, but the motion planner works with angular velocity measured in degrees/s. If, for example, the speed in m/s has to be reduced by a half (k = 2), then the angular speed have to be reduced also by a half as shows the Equation 4.1.

$$v = \frac{2\pi r\omega}{360} \longrightarrow \frac{v}{k} = \frac{2\pi r\frac{\omega}{k}}{360} \tag{4.1}$$

With this method the reduced speed motion of the robot complies with the ISO rules.

4.3.4 Calculate the maximum node speed

The robot is capable to limit its maximum speed according to the cooperation mode. This simple functionality allows to do a much safer cooperation, and A* can find many more paths near the human. But there are more paths that A* can obtain if the speed is more reduced. In a cooperation mode, a speed of 250mm/s can be too high for some nodes but if it is slower, the robot can move along these nodes safely. Even without cooperation mode, the speed can be reduced to find a better path. This dynamic speed control can limit the risk much more, allowing to find new potential paths.

A* gives to the motion planner a vector of nodes that the robot has to cross to reach the final goal. The developed solution consists in the addition of a new attribute to

each node which can take values from 0 to 1 to represent the speed reduction factor. It works like the risk multiplier; the speed is multiplied by this factor and reduced in consequence. The motion planner multiplies the target velocity by the speed reduction value indicated in the current node. The 250mm/s limit is also considered in this speed reduction factor.

The speed values of these nodes are calculated when A* checks the risk for them. If the risk is over the threshold, it repeats the calculation at a lower speed until it is about to reach 0. If the risk is always over the threshold, the node is closed. But if the risk is reduced under the threshold, the calculated speed reduction is associated to this node.

The experiments showed that sometimes only one node needs to reduce its speed. The robot has a maximum deceleration value, so sometimes between node and node the robot did not have time to decrease its velocity. To solve this problem, an interpolation function for the speed multiplier values was developed. The lowest speed should be in the node to be interpolated, increasing linearly in the previous and next nodes. So if after doing the interpolation the resulting speed value is smaller than in the node to be interpolated, the new value is not applied for that previous or next node.

Also, A* must know that a node with a slower speed is worse than other equal at full speed. To take this into consideration the node score system has been modified. In the original A*, the score of a node is the sum of the previous node score plus its own score. To take the speed into account, the own score of a node is increased depending on the speed reduction value until is almost the double if the speed is near 0.

With all these implementations, the MAROCO framework now has its modules joined together and have all the basic tools to achieve a contact based cooperation.

4.4 Other modifications

There have been programmed two other interesting modifications that are not directly related with the cooperation system.

- There is a new MFC event to recognize the joystick buttons. This is useful to connect a remote controller and simulate that the human grips the object given by the robot. This function has been fully tested with a Super Nintendo gamepad connected as the first joystick.
- There is another modified behaviour in the fuzzy risk module. In the original code, every time that A* or the motion planner called to it, all the visualizations of the fuzzy variables, rules and outputs were updated, causing a huge impact in the global performance. Now the visualizations are only updated when the risk quantification module is called from the motion planner, and all the windows are disabled by default.

Chapter 5

Experimental Analysis

This chapter presents different experiments to test and calibrate every part of the implemented system. The idea here is to show groups of isolated tests to check each functionality independently. There are many different modules that can affect to these experiments and much information that can be gathered, so it is necessary to have an environment as much stable as possible to have meaningful and measurable results.

There are four groups of experiments:

- Risk quantification modifications, especially the risk reduction factor.
- Path planning and tasks of the robot depending on each situation. Also check how the system reacts to changes in the environment.
- Velocity control, checking the effects in the planned path and the risk.
- The full system in a real situation, checking its behaviour and comparing the execution speed with the old system.

As has been explained in the Implementation section, in this phase some variables have been calibrated and changed from the original MAROCO version. The following values are used in all the tests and the final system:

- The variable risk_threshold now is 0.8 (originally 0.95). This allows to the system to be much more reactive and dynamic to risk variations.
- The variable dist_threshold now is 0.4 (originally 0.05). This allows to A* to design paths avoiding the human with a much more reasonable safety margin.
- The maximum speed is increased to 200. In the practice, the robot normally will not reach this velocity because its acceleration takes time, but increases the freedom of the speed control system.

Also it was necessary to modify the original system to compare it with the new one. As has been explained in the Implementation section, the original system tends to freeze because sometimes A* always creates paths inside the Look Ahead risk safety range. To solve this problem, the path re-planning is triggered only when the human intersects the planned path or enters in its safety distance margin.

5.1 Risk quantification

These tests show the behaviour of the modified fuzzy risk system. All of them have these characteristics in common unless another thing is indicated:

- The task of the robot arm is to move itself about 180 degrees crossing the human action area. This is its default task in MAROCO.
- The human is very near to the robot, does not move, and does not look at it.
- The expectation delay is set to 0 to have a static environment.
- The cooperation mode is changed manually for each test to see how the risk quantification system reacts in the same environment.

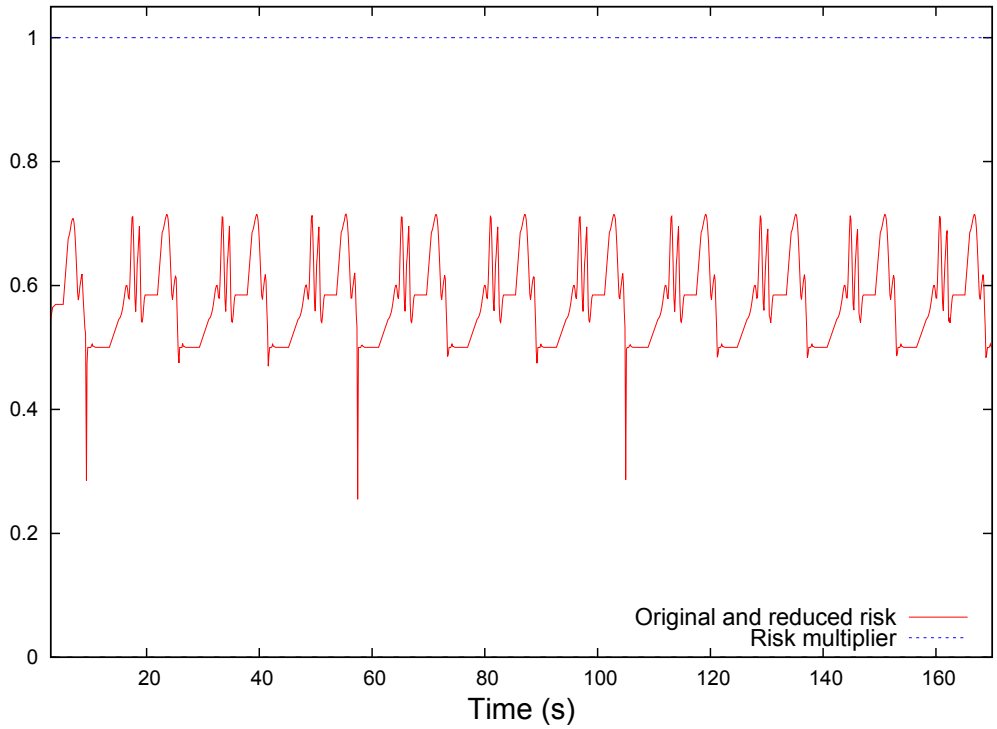


Figure 5.1: Risk without cooperative mode.

When the cooperative mode is *off* (Figure 5.1), the risk is exactly equal to the original system. The path taken can be a little different because of the new velocity considerations, but in this mode the fuzzy risk module behaves the same way as before. As can be seen, the final risk is always under the threshold along the full path.

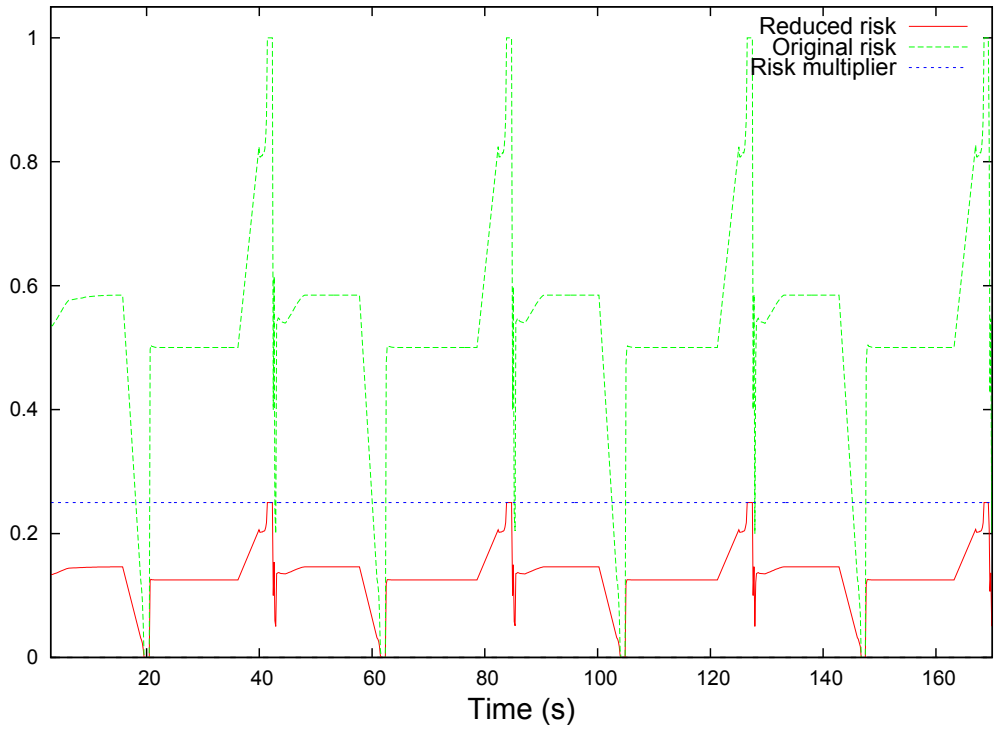


Figure 5.2: Risk with *unattended* cooperative mode.

If the cooperative mode is set *unattended* (Figure 5.2), the original risk fluctuates much more because the robot is nearer to the human. In this mode, the risk multiplier reduces the final risk value, maintaining it under the risk threshold and allowing the robot to approximate much more to the human. The risk multiplier value is constant because the environment, including the situation awareness delay, is static for these experiments. Without the risk reduction the final risk exceeds the allowed threshold and the cooperation cannot be done.

In this situation can be seen that when the robot is moving in the farthest end of the path (relative to the human) the original risk is next to 0 and when it moves in the nearest end, the original risk approaches to 1. This is because in both ends of the target path there are nodes with the *controlled approach* flag activated. The path between these nodes is not generated by A*, they are linked with a straight line. These parts of the path are not optimized taking the distance and risk into account, so the current risk fluctuates in consequence when the robot traverses these path sections.

The longer periods in the graph are also interesting. This is because when there is a cooperative mode on, the maximum speed of the TCP is always reduced at 250mm/s

as is indicated by the ISO10218-1 rules.

When the cooperation mode is set to *attended*, the risk is reduced only when the human is looking at the robot. Depending on the deviation of the human head with the robot, the risk is more reduced or less. If the human is looking directly to the robot, the risk reduction behaves exactly like in *unattended* mode. In this case, the base risk is also reduced in respect to the previous experiments. Of course, if the human is looking at the robot the situation is less risky in the original fuzzy system too.

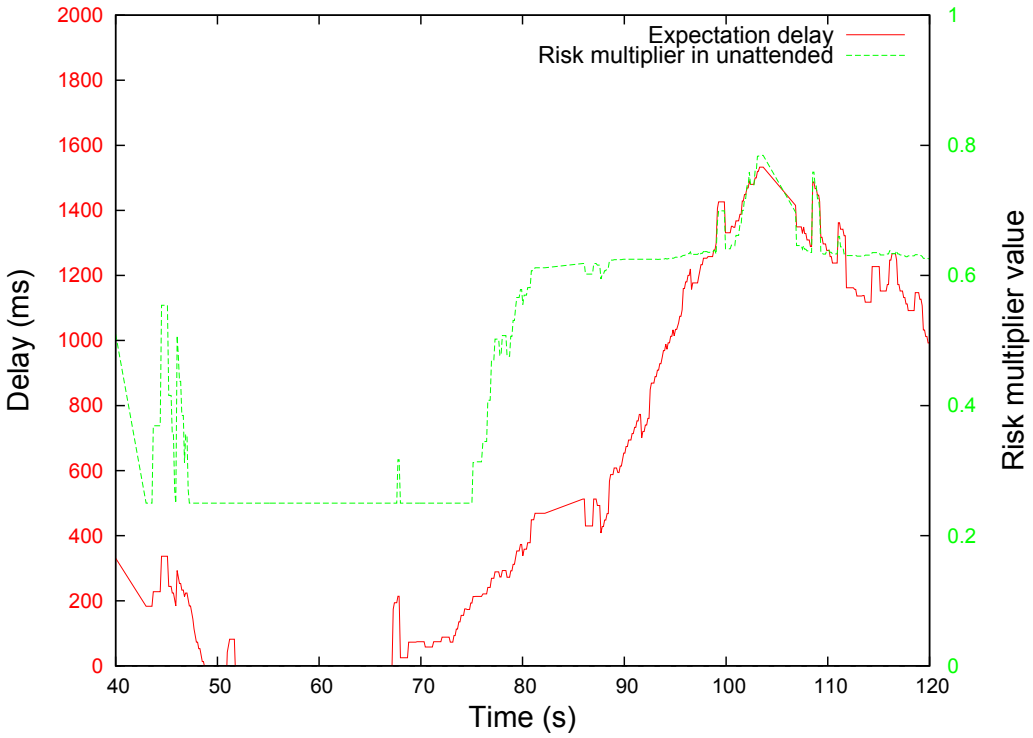


Figure 5.3: Risk multiplier depending on the average expectation delay.

In Figure 5.3 the expectation delay is changed manually to ilustrate the dynamism of the fuzzy risk multiplier value in the *unattended* cooperation mode. When the expectation delay exceeds a certain threshold, the risk is less reduced. As can be seen, the minimum multiplier value is 0.25 when there is an updated situation detection. As it gets older, the risk multiplier increases dynamically to avoid reducing the final risk too much.

This shows that the inclusion of the situation awareness delay restriction modifies the final risk value and, potentially, the planned path. So it is important to execute the framework in a fast machine to get always updated expectations and have paths as optimized as possible.

5.2 Path planner

In the previous section the target path was always the default of the MAROCO, but in this one the path is changed depending on the current task. The next experiments check the correct functions of the path planner and if the system allows a contact based cooperation with the human in a controlled environment. The expectation is changed manually, forcing the robot to change the task and behave according to the situation.

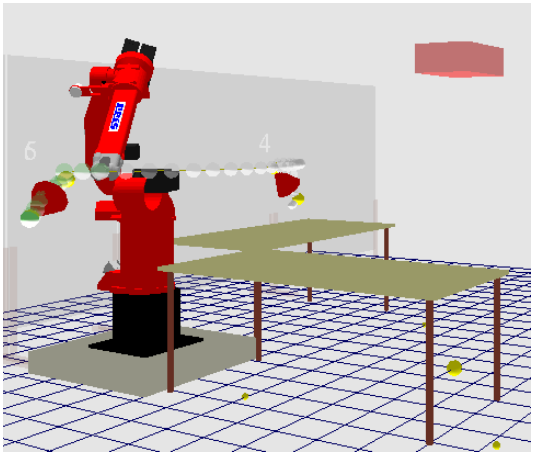


Figure 5.4: Target path for the default MAROCO task.

The Figure 5.4 shows the original path of the MAROCO. This is the default task, so when any other task is finished the robot continues doing this one until it receives a new expectation that forces him to change it. The robot TCP moves about 180 degrees crossing the human working area in one direction and the opposite, in a loop. At both ends of the path there are *controlled approach* nodes in which the path between them is not generated by A*. They are linked just by a straight line.

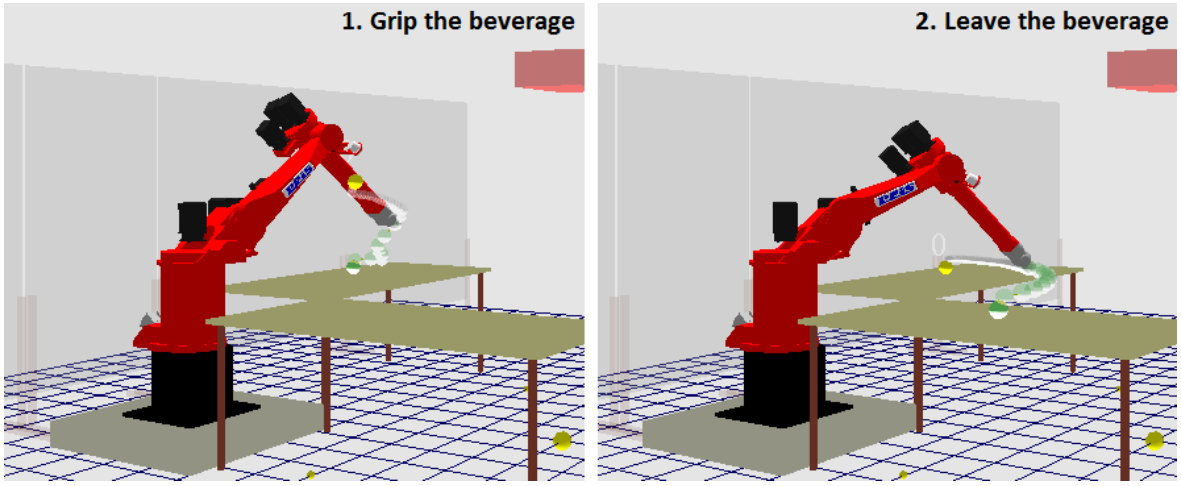


Figure 5.5: Target path corresponding to the *Get Beverage* expectation.

The target path for the *Get Beverage* expectation is shown in the Figure 5.5. In it, the robot simulates to grip the beverage in one table, and leaves the gripped beverage in the other. This movement was impossible to do in the original MAROCO because the risk was always too high. In the current system, when this expectation is detected, the cooperation mode is set to unattended when the robot has to move near the human, reducing the risk and allowing the cooperation even if the person is not looking at the robot. After leaving the beverage, the robot continues doing the default task.

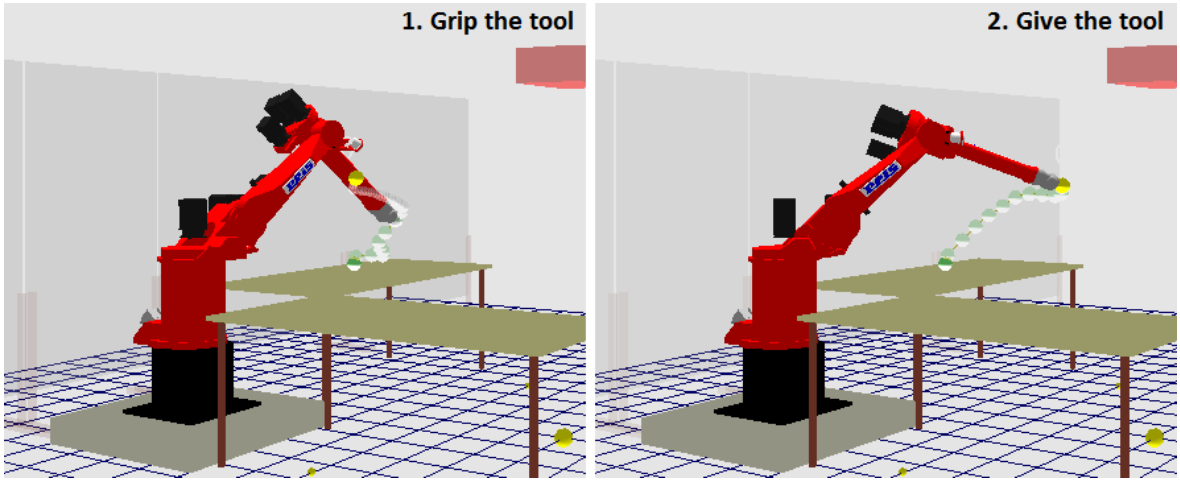


Figure 5.6: Target path corresponding to the *Get Object* expectation.

The other task is triggered when the expectation is *Get Object* (Figure 5.6). It simulates the gripping of an object in one table and moves it to a point very deep in the human area to give him the the object. This movement is the most difficult to do by the original system, because it lacks of a velocity control system and a risk reduction method. In the current system the cooperation mode changes to *attended* when the robot has to give the object to the human. The human normally has to be looking to the robot in order to allow this movement, but even if he is not looking, the movement sometimes can be done if the speed is low enough and the human is not too near of the final point.

This means that not only the risk reduction value allows the cooperation. With the dynamic velocity control A* can reach some nodes that were forbidden in the original system just by reducing the speed.

The robot returns to the default task when the human grips the tool. This is simulated in MAROCO by pressing the Z key of the keyboard or the button 1 of the connected joystick 1. Also, if a different expectation comes, the robot leaves the beverage or the tool in its original position if it had one of them gripped. If the robot had already gripped the required object, it moves directly to leave the beverage or give the tool.

The path replanning is triggered when the distance to the human is lower than the distance threshold of the Look Ahead function. The rest of the functions are like in the original system. It is important to say that the path planned by A* sometimes

passes through the tables in the original MAROCO too.

5.3 Velocity control

In this section, the main objective is to evaluate how the velocity control works and how it is influenced by the distance to the human. The cooperation mode is changed manually to see the effects in the speed. The default path is selected again for all the experiments.

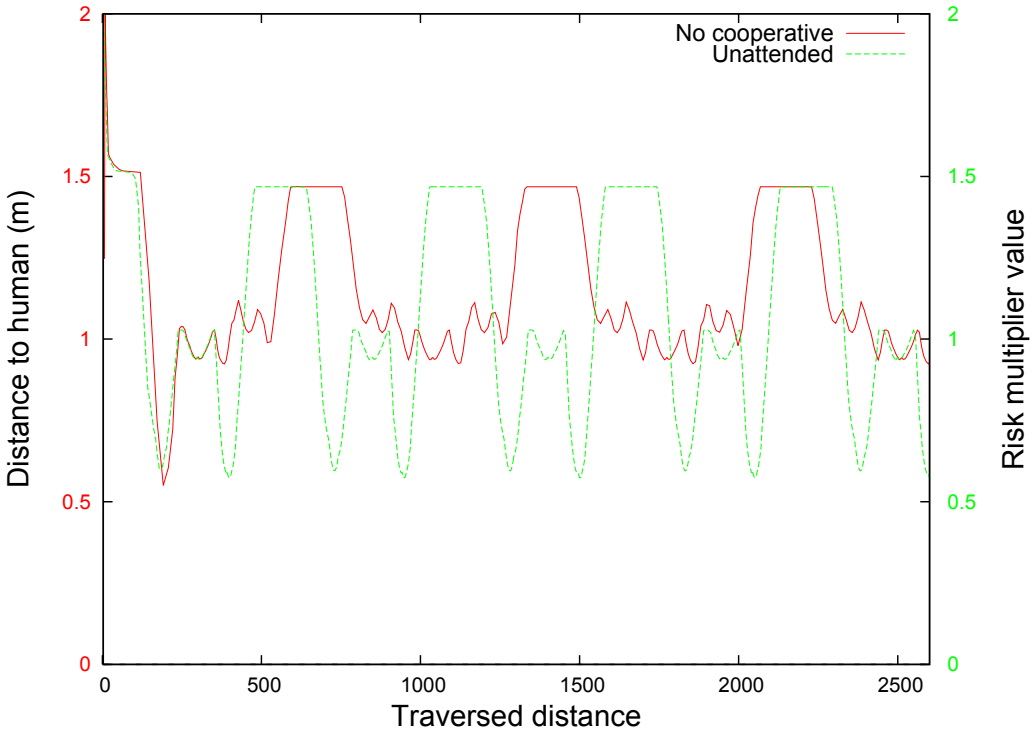


Figure 5.7: Distance to human with unattended and no cooperative mode.

The differences in the distance to the human depending on the cooperation mode can be seen clearly in the Figure 5.7. There are two different planned paths for the same task. Without cooperative mode the average distance is higher than in the unattended mode. The periods in the unattended mode are shorter, which means that the robot finish its path traversing less distance. This shows how the risk can influence in the planned path length.

The Figure 5.8 shows how the speed is reduced to allow nearer approximations to the human. When the distance is low, the velocity is reduced to pass through these nodes. The speed variations along the nearest path sections are very important because they

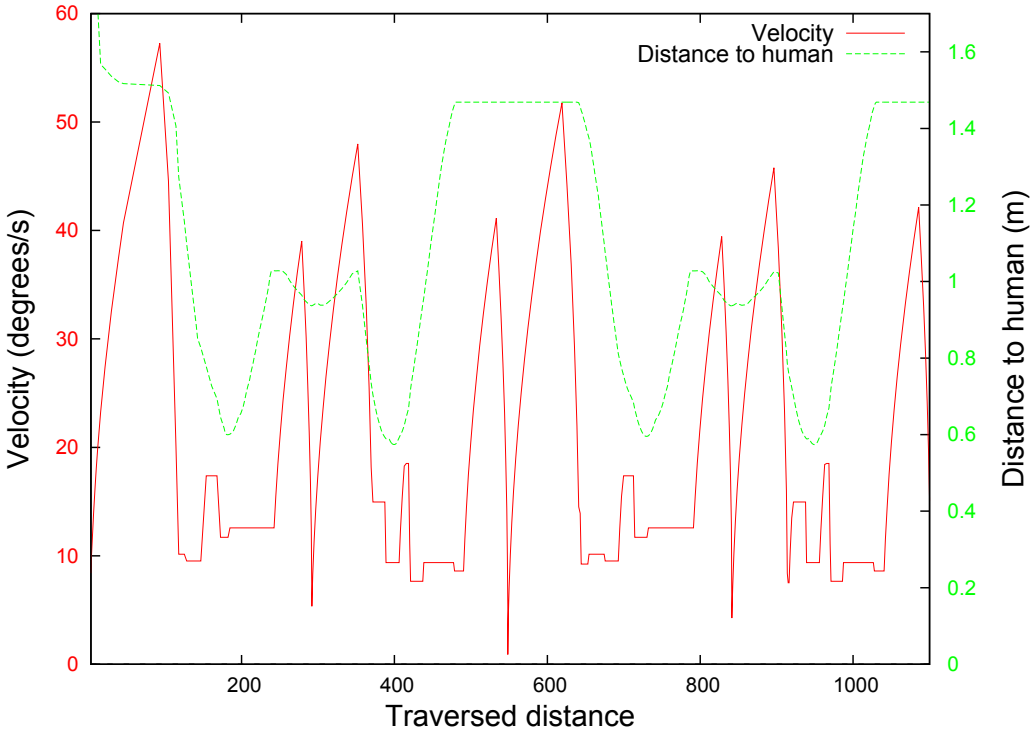


Figure 5.8: Robot velocity and distance to human variations with unattended mode.

demonstrate that the velocity control is working and A* takes nearer nodes although the robot has to reduce its speed.

The apparently fast decelerations in the robot speed means that it stops at the end of the path and starts replanning another. This graph uses the total traversed distance instead of the elapsed time for comparison reasons, so the plots are reasonable.

When there is not cooperation mode, the risk is equal to the original system, but the new one plans a little different path even with the same final risk value. The Figure 5.9 shows these small differences. The speed taken in some nodes is slightly inferior because A* considers a trade-off between distance and speed, and sometimes a reduced speed node is preferred. This implies that the velocity control is independent of the risk value, being more probable that A* can find a successful path to the goal node.

In the figure 5.10, the cooperation mode is changed manually to see how the path is altered. In the left image, the unattended mode reduces the risk, allowing approximating much more to the human. In the right image the risk is the original and the path tries to maintain the safety distance.

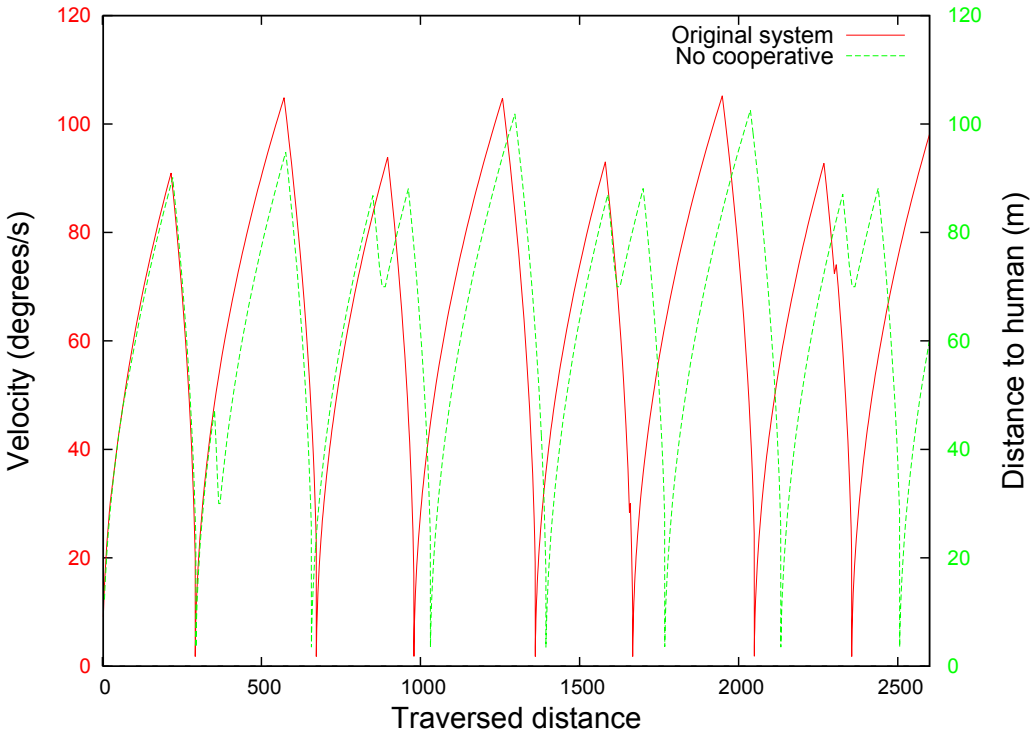


Figure 5.9: Velocity without cooperative mode and in the original system.

5.4 Full system

The final test set executes the program as is should be, without static environment and without changing any variable artificially. The human moves in the robot working area and interacts with it while the robot tries to do its default task. Now the system depends on the detections of the situation awareness module to change the task and the cooperation mode automatically. The only manual action is to press the key to simulate that the human has gripped the object given by the robot.

The test of the full system shows that the robot can cooperate with the human, guessing its intentions and working proactively, although there are yet some problems. The programmed behaviour for the *Get Beverage* expectation works well, but the robot sometimes moves too near to the human to leave the beverage in the place. The main problem is that the replanning takes some time, and (although the robot can stop its motion) if the human is not looking and does a fast movement, he can hit the robot arm. This can be solved just by adding an intermediate target node in the corner of the tables to avoid going round the human, but this is a trivial problem and the developed solution was better to test the speed control.

A* normally does a good job planning the path but sometimes the path has strange turns and loops which clearly are not the best way to do it, but the path always respects

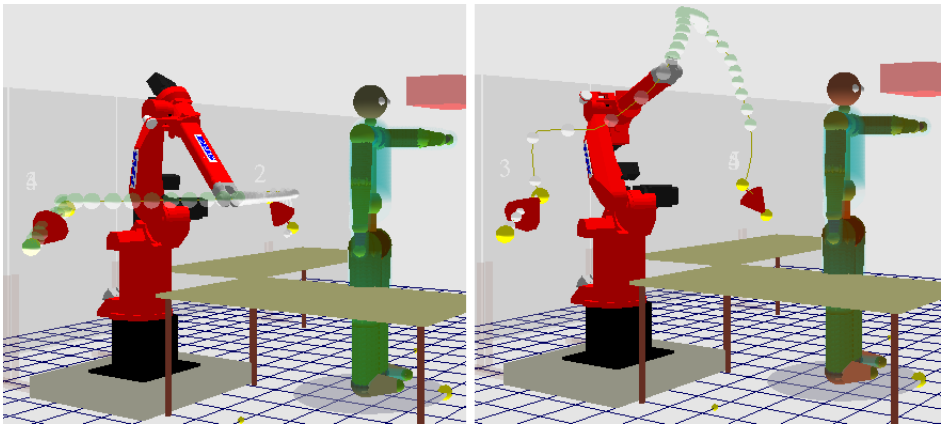


Figure 5.10: Path planned in unattended mode and without cooperation.

the risk and distance constraints. This path should be improved in future works to be straighter. Also, A* sometimes makes paths through the tables. This should be solved in future works too.

The behaviour for the *Get Object* expectation works very well. The robot arm approaches slowly to the human and stops in the programmed point. Sometimes, if the human approaches before the robot has reached the goal node, the Look Ahead function forces a path replan, and cannot find any.

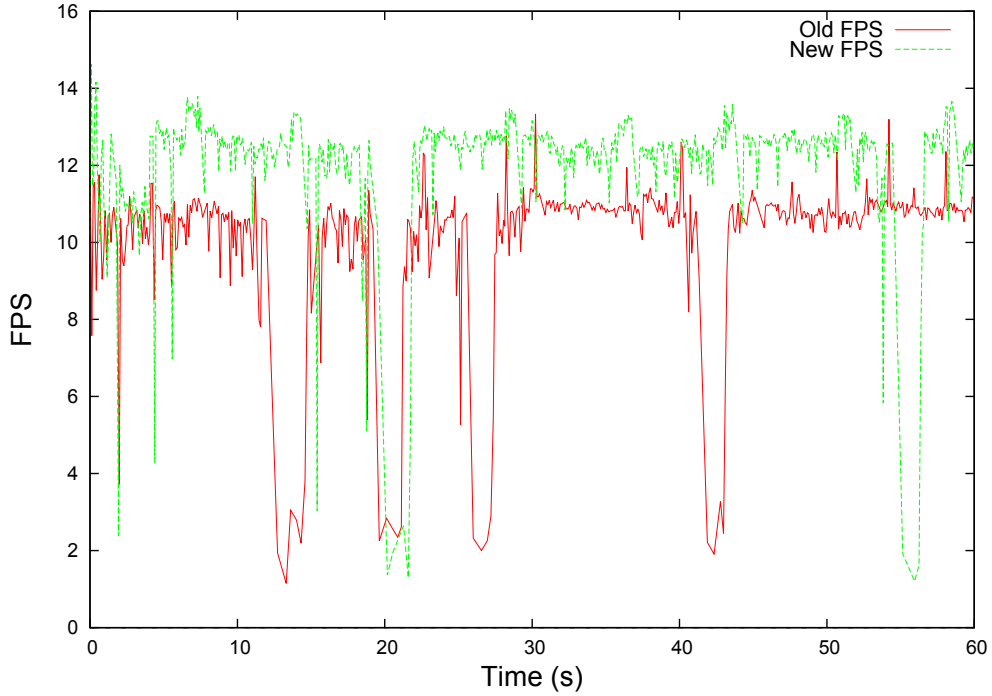


Figure 5.11: Frames per second in the original and the new implemented system.

This forces to the human to go back and allow the robot to finish its movement, because he cannot grip the object given by the robot until it has reached the goal node. This behaviour should be improved too.

The frames per second are similar than in the original system (Figure 5.11). The new program is even one frame per second faster but is a tiny difference. The sudden drops mean a path replan. In this case, there are more in the old system. The windows with graphs of the fuzzy variables, rules and fuzzy outputs slow down the system too much, so all the debug graphic windows except the main OpenGL have been disabled in the new and the original system. The most important added delay in the new system is in the path generation part, because the fuzzy risk system is executed many times for a node to determine the minimum acceptable velocity. This delay is not too big and this method does the job without making important changes in the original code.

In essence, the original system has many tools to cooperate but they were not joined together. Now, with the new added functions and adjusts, every part contributes to have an initial approximation of cooperation that can be extended and improved in the future.

Chapter 6

Summary

This section contains a summary of all the work that has been done, the final conclusions and the proposed future work.

6.1 Conclusions

In this work a system to allow a robot cooperating with a human has been improved. The risk quantification module needed changes because the original one was not designed to do potentially dangerous movements. This limitation prohibited the cooperation. A double output fuzzy system was designed. These two outputs are:

- The original base risk, maintaining the idea and all the meaning of the original fuzzy system that proved to be safe.
- A risk reduction factor, which multiplied by the original risk gives a reduced value more suitable for cooperation purposes.

The path of the robot changes according to the expectation received. A behaviour was programmed for two different expectations, *Get Beverage* and *Get Object*. The expectation is not detected in real time, so the risk reduction factor takes into account the delay of each detection of the situation awareness module.

The path planner was also modified. The original planner moves the robot always at the same speed, preventing the robot to pass through many nodes because it was very risky. A speed control was developed to achieve two main targets:

- Move the TCP under the speed of 250mm/s in reduced speed mode (when the robot has to cooperate with the human) to comply with the ISO rules.
- A dynamic speed control to reduce the speed to allow the path planner moving the robot through nodes that were inaccessible before at higher speeds.

The system is capable to decide (with the situation awareness module) when to leave a beverage on a table very near to the human, or when the robot has to get a tool and give it to the human. Now it is capable to plan the task according to this expectation and its movements considering safety constraints.

6.2 Future work

A very interesting possibility in this work is to improve the cooperation by giving the object to the human exactly in front of him, instead of in a fixed point in space. This can be interesting in the case of the beverage too, leaving it near the human instead of in a defined place in the table.

Also, currently the cooperation mode fuzzy variable takes always three values. 0, 0.5 and 1 for *off*, *unattended*, and *attended* respectively. But the system works also with values different of these ones, always between 0 and 1, giving different levels of cooperation. But also, this means that the cooperation mode could be interpolated along the path traversing. For example, currently after giving an object to the human, the cooperation mode returns to 0 immediately, causing a sudden fast robot movement and a forced path to avoid the human. If the cooperation mode is interpolated, the risk reduction change could be softer and the resulting path more natural.

The last proposed work is to have a path planner that tries to find better paths every time. Currently the path is built in one moment, taking the environment configuration. If the human moves out, for example, the robot will move to the goal using the already calculated path because it is only replanned when there is some danger detected in the planned path. With the speed control, this proposed improvement is now more interesting due to the fact that the robot will conserve the planned velocity too.

Bibliography

[1] http://www.mrk-systeme.de/e_produkte_interaction.html Last visited on 2012-08-25. (Cited on page 5)

[2] https://shop.pilz.com/eshop/cat/en/DE/00014000337042/SafetyEYE-Safe-camera-system Last visited on 2012-08-25. (Cited on page 6)

[Amato et al. 1998] N. M. Amato, O. B. Bayazit, L. K. Dale, C. Jones and D. Vallejo:*OBPRM: An obstacle-based PRM for 3D workspaces.* In Proceedings of Workshop on Algorithmic Foundations of Robotics, 1998, pp. 155-168. (Cited on page 7)

[Czapiewski et al. 2010] P. Czapiewski and J. Graf: *Entwurf und Integration eines Fuzzy-Logik Systems für die sichere Mensch-Roboter-Kooperation.* Interner Bericht, Fassung vom 5. Februar 2010. (Cited on page 20)

[Edsinger et al. 2007] A. Edsinger and C.C. Kemp: *Human-Robot Interaction for Cooperative Manipulation: Handing Objects to One Another.* Robot and Human interactive Communication 2007. The 16th IEEE International Symposium pp. 1167-1172. (Cited on page 6)

[Graf et al. 2009] J. Graf, S. Puls and H. Wörn: *Incorporating Novel Path Planning Method into Cognitive Vision System for Safe Human-Robot Interaction.* Proceedings of IARIA Computation World: Cognitive 2009, Athen, Greece, pp. 443-447. (Cited on page 4 und 17)

[Graf et al. 2010] J. Graf, F. Dittrich and H. Wörn: *High Performance Optical Flow Serves Bayesian Filtering for Safe Human-Robot Cooperation.* Proceedings of Int. Symposium of Robotics (ISR) and VDI/VDE Robotik 2010, München, Germany, 8 pages. (Cited on page 2)

[Graf et al. 2010b] J. Graf, P. Czapiewski and H. Wörn: *Evaluating Risk Estimation Methods and Path Planning for Safe Human-Robot Cooperation.* In Proceedings of ISR/ROBOTIK. 2010. Institute for Process Control and Robotics (IPR), Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany. (Cited on page 3, 9 und 20)

[Hart et al. 1968] P. E. Hart, N. J. Nilsson and B. Raphael: *A Formal Basis for the Heuristic Determination of Minimum Cost Paths.* IEEE Transactions on Systems Science and Cybernetics SSC4, pp. 100-107. (Cited on page 13)

[Heinrich et al. 2008] D. Heinrich, M. Fischer and T. Gecks: *Multi-Camera Collision*

Detection Between Known and Unknown Objects. Proceedings of 2nd ACM/IEEE International Conference on Distributed Smart Cameras (ICDSC), 2008, Stanford, USA. (Cited on page 6)

[Kindel et al. 2000] R. Kindel, D. Hsu, J.C. Latombe and S. Rock: *Kinodynamic motion planning amidst moving obstacles*. In Proceedings of IEEE Conference on Robotics and Automation, 2000, pp. 537-543. (Cited on page 8)

[Kulic 2005] D. Kulic: *Safety for Human-Robot Interaction*. Dissertation, Faculty of Graduate Studies, Mechnical Engineering, University of British Columbia, December 2005. (Cited on page 6)

[Kunz et al. 2010] T. Kunz, U. Reiser, M. Stilman and A. Verl: *Real-Time Path Planning for a Robot Arm in Changing Environments*. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'10), Oct. 2010 (Cited on page 7)

[Leven et al. 2002] P. Leven and S. Hutchinson: *A Framework for Real-time Path Planning in Changing Environments*. The International Journal of Robotics Research, Vol. 21, 2002, pp. 999-1030. (Cited on page 7)

[Lösch et al. 2009] M. Lösch, S. Gärtner, S. Knoop, S.R. Schmidt-Rohr and R. Dillmann: *A human body model initialization approach made real-time capable through heuristic constraints*. Proceedings of the 14th International Conference on Advanced Robotics (ICAR), 2009, Munich, Germany. (Cited on page 6)

[Mazer et al. 1998] E. Mazer, J.M. Ahuactzin and P. Bessière: *The Ariadne's clew algorithm*. Journal of Artificial Intelligence Research 9-295-316, 1998. (Cited on page 8)

[Mizuuchi et al. 2009] I. Mizuuchi, J. Fujimoto, K. Yamamoto, Y. Sodeyama, N. Muramatsu, S. Ohta, K. Hongo, T. Hirose and M. Inaba: *A Kitchen Assistant Robot with a Variety of Sensors Embedded in the Hand to Clear Away Dishes*. First International Symposium on Quality of Life Technology, 2009. (Cited on page 5)

[Pearl 1984] J. Pearl: *Heuristics: intelligent search strategies for computer problem solving*. Addison-Wesley Longman Publishing Co., Inc. (Cited on page 14)

[Puls et al. 2011] S. Puls, J. Graf and H. Wörn: *Design and Evaluation of Description Logics based Recognition and Understanding of Situations and Activities for Safe Human-Robot Cooperation*. International Journal on Advances in Intelligent Systems, vol 4 no 3 & 4, 2011. (Cited on page 3)

[Puls et al. 2012] S. Puls, J. Graf and H. Wörn: *Cognitive Robotics in Industrial Environments*. Maurtua Iñaki (Ed.): Cognitive Robotics in Industrial Environments (2012) pp. 213-234. (Cited on page 2)

[Reiser et al. 2009] U. Reiser, C. Connette, J. Fischer, J. Kubacki, A. Bubeck, F. Weisshardt, T. Jacobs, C. Parlitz, M. Hägele and A. Verl: *Care-O-bot® 3 - Creating a product vision for service robot applications*. The 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, October 11-15, 2009 St. Louis, USA. (Cited on page 5)

[Sax 2004] Carl Sax: *Using Robotics to Enhance Logistics Solutions: An Interdisciplinary Approach to IKF.* M. Jamshidi, A Ollero, L. Foulloy, M Reuter, A. Kamrani and H. Yutaka (Eds.), World Automation Congress Volume 4, 2004 Seville. (Cited on page 5)

[Thiemermann 2005] S. Thiemermann: *Direkte Mensch-Roboter-Kooperation in der Kleinteilmontage mit einem SCARA-Roboter.* Dissertation, Fakultät für Maschinenbau der Universität Stuttgart, 2005. (Cited on page 6)

[Vallejo et al. 1999] D. Vallejo, C. Jones, N.M. Amato: *An adaptive framework for 'single shot' motion planning.* Technical Report TR99-024, Department of Computer Science, Texas A&M University, College Station, TX, October 1999. (Cited on page 8)

[Winkler 2008] B. Winkler: *Konzept zur Sicheren Mensch-Roboter-Kooperation auf Basis von Schnellen 3-D Time-of-Flight Sensoren.* A. Verl und M. Hägele (Eds.), VDI/VDE Gesellschaft für Meß- und Automatisierungtechnik (GMA), Düsseldorf, Deutsche Gesellschaft für Robotik: Robotik 2008, pp. 147-151. Tagung München. (Cited on page 6)

[Zadeh et al. 1996] L. A. Zadeh, G. J. Klir and B. Yuan: *Fuzzy Sets, Fuzzy Logic, Fuzzy Systems: Selected Papers.* World Scientific, 1996. (Cited on page 9)