Festo Logistics Team Description Paper 2013

Slim Abdennadher, Mennat-Allah Saleh, Ahmed Badr, Ahmed Mustafa, Ahmed Safty, Kareem Yousry, Menna Nabil, Nourhan Aloush, Omar Fouda, and Sara Alaa

ArtSapience, German University in Cairo

1 Introduction

The GUC_Artsapience is a team from the German University in Cairo. We have participated in the Festo Hockey challenge in 2009 and in the Festo Logistics league since 2010. The team ranked second place in 2009 and first place in 2010. The Festo Simulation League problem is approached as an industrial simulation problem where the robots are considered as agents performing the task of delivering a set of products. The workflow is divided into 3 main components: planning, motion planning and computer vision. Planning is involved with strategy setting for the robots. The agents are equipped with a dynamic plan that allows them to formulate action steps to be taken during the match for reaching the final goal. The planning component is set using a centralized planning method which allows easy synchronization and collaboration of agents. Motion planning is concerned with moving the robots from one point in the arena to the other. This component is divided into static motion and dynamic motion. Static motion is responsible for moving the agent from one point in the field to the other with previous knowledge of locations of objects in the field. Dynamic motion is concerned with moving the robot from one point to the other while dynamically understanding and avoiding moving surroundings. Finally, the vision component is responsible for viewing the agent's world. The vision allows the robot to see and process the machine colors and signals as well as seeing and locating the pucks in the surrounding world.

2 Planner

The festo competition mimics an industrial situation in nature, and is hence, treated as such. The goal that is to be attained in such a simulation, is the production of as many products within a given time frame, whilst minimizing the cost of production, hence, optimizing the throughput of the entire process. Such a goal is to be achieved through the collaboration between multiple agents. Hence, the need for planning arises.

2.1 Approach

The approach pursued is multibody planning, a strategy, whereby the information collected by each agent is pooled centrally, contributing to a centralized world model, which is processed to advise the execution of the overall plan by each agent.

The planner dictates the actions to be carried out by the individual agents. The planner is of both, a dynamic and a domain-specific nature. The dynamicity of the planner implies that no specific sequence of actions is planned in advance. Instead, the planner is given a world model and a goal, and subsequently produces a set of condition-action rules. Should the current environment satisfy a condition, the corresponding action is executed.

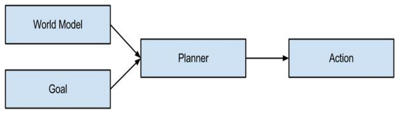

Fig. 1. World Model Representation

Furthermore, a domain-specific planner was opted for, due to its high efficiency compared to domain-independent and domain-configurable planners. Its specialized nature enabled the encoding of domain-specific problem solving techniques directly into the planner.

Our planner is utility-based, whose performance measure is not only concerned with the mere excerpt of whether goal is met or not, but also, to what degree is such a goal met efficiently, as opposed to a goal based planner. It further implements the HTN planning model (Hierarchical Tree Network planning model)[1]. In an HTN planner, the objective is described not as a set of goal states but instead as a collection of tasks that are to be carried out.

2.2 Environment and World Model

Our environment is a fully observable environment of a semi dynamic nature; since, even though, the environment itself isnt affected by the passage of time (except for machine turnoff) , the agents performance score does. Furthermore, it is of a deterministic nature, as the next state of the environment can be, almost, completely determined by the current state and the action executed by the agent.

The environment is represented as follows:

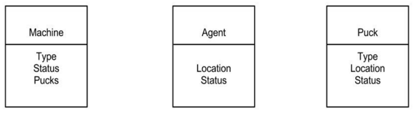

Fig. 2. Plan Execution

2.3 Match Phases

In the RoboCup Festo Logistics competition, the workflow is divided into two different phases: a discovery phase and the actual production phase.

Discovery Phase In this phase, the agents will traverse the machines in an order that maximizes the performance measure, through minimizing the time consumed. This is achieved through the division of the arena into three areas, each covered by one of the agents. Agents will communicate their findings to the centralized planner, contributing to the overall knowledge about the world model. The discovery phase holds an upper bound of nine and a lower bound of six machine visits, where the centralized planner would try to infer the remain- ing machine types through information currently available regarding the world model.

Production Phase Generally, goals are a set of products and late order prod- ucts. Each subgoal is given a priority based on the heuristic of how far it is from the goal state. In that context, a finished product assumes the highest priority whereas the raw material assumes the lowest. The plan is based upon partial-order planning, such that the order of execution of actions is flexible. The loosening of the execution order implies that there exists a set of possible plans for the same problem, each possessing a unique order of execution.

The planner starts off by the goals dictated through the referee box. These goals are broken down into primitive actions by the planner, which then assigns each action to an available agent. The agents act as a single body, collaborating to achieve the dictated goals.

Data processing is centralized, improving world synchronization, thus; avoiding the replication of the world model with its full details on the 3 agents. The processing happens on the shared world model to produce the best action to be taken at any time depending on the worlds current state and the subgoals/goals priorities. This approach also reduces the heavy communication dependency, by reducing the amount of communication by nearly a factor of 2, since each agent has to communicate changes only to the server.

3 Motion Planning

The planner decides where the robot should go, the construction of the path to this destination point and the actual physical relocation of the robot from one point to the next is the task of the motion planning component.

In its simplest form, the way between the robot and the destination point is free of any obstacles. In most cases however, the path will have some obstacles in it. These obstacles have been divided into two categories, static obstacles whose positions are already known to all the robots and dynamic obstacles that can move at random throughout the arena.

3.1 Static Avoidance

When moving from one point to another it is vital to make sure that the robot does not enter the machine's area. Therefore, when a robot detects a machine blocking its way it has to choose one of four alternative transitional points to head to first. These are the points to the left, right, front and back of the machine. In the situation clarified by diagram 3 what the algorithm does is that it calculates which of the two points, point 1 and point 3, is closer to the robots current position and then compares between this point and points 2 and 4 to find the point closest to the destination point. After arriving at this point, the robot then recalculates the path that it needs to follow to get to the original destination point. Again that path may contain obstacles and further obstacle avoidance would be required.

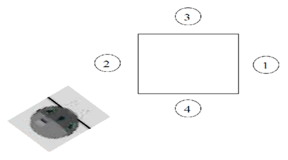

Fig. 3. Possible transitional points

3.2 Dynamic Avoidance

Dynamic avoidance is used to avoid collision between robots. The robot is equipped with nine infrared sensors all around its perimeter. These are used to continuously check if another robot is approaching too closely in which case, one of the robots moves sideways so as to avoid any collision. This is dictated by the Hierarchical Obstacle Avoidance (HOA) algorithm. The robots infrared sensors are divided into three different groups made up of three sensors each. For each of these groups, the HOA takes as input the voltage reading of each of the three sensors, this reading is then converted to distance and fuzzy logic is used to decide on the most appropriate velocities that the robot should take in the x-direction and y-direction as well as the angle of rotation. The HOA is implemented using Fuzzy Logic Control (FLC).

3.3 Fuzzy Logic

A basic skill that comes naturally to humans is the ability to understand descriptions that are rather vague. An example can be a distance described as close at between 0 and 2 meters, medium between 2 and 5 meters and far greater than 5 meters. If the distance is actually 4.99 humans immediately know the distance is closer to being far than it is medium. A computer code however would not be able to recognize this difference and would consider the distance to be in the medium range[2]. This is where Fuzzy Logic is applied. Using fuzzy sets, we can create a set representing a certain property, for example the distance near and assign a value range for this property. On arrival of an input value, these crisp values are then transformed to grades of membership functions indicating how much our input value belongs to a particular fuzzy set. Each input value will belong to a particular set with a certain probabilistic value (0.0-1.0) where value 1 indicates that the input belongs completely to the set and value 0 indicates that the input value does not belong to this set at all, this range value is called the firing strength and the process itself is known as fuzzifcation.

3.4 Fuzzy Logic Control

Fuzzy control systems are based on fuzzy logic; they analyze an analog input variable ranging in values between 0 and 1 to provide an output fuzzy strength value also ranging between 0 and 1. In Fuzzy Control, a set of rules is established in a form similar to IF-THEN statements with the IF part called the antecedent and the THEN part the consequent. Fuzzy control relies heavily on membership functions, usually triangular or trapezoid shaped. The control receives the output from each rule defining its membership to each class and based on all these outputs, a single output is calculated.

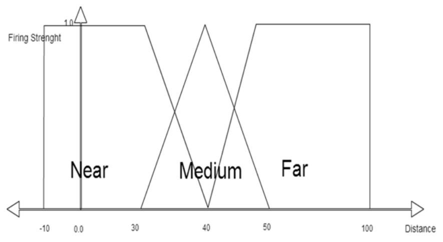

Fig. 4. Member function of the left front sensor distance.

FLC is used to provide the robot with a set of rules to identify the minimum distance that must be kept obstacle free around the robot. Fuzzy logic control is also used when the robot is moving towards a destination point. Instead of moving at a fixed speed, FLC can be used to handle the robot's velocity. A set of rules is used to provide a robot with a velocity to be used depending on where the destination point is in regard to the robots current position. This means that if a robot still has a long distance to travel it can move at a higher velocity and only slow down when approaching the destination until a complete stop at the destination point.

4 Vision

The Vision Component represents the main input to the system that keeps track of the environment around the robot. The vision task can be separated into two main challenges, the first one is to detect the colors of machines inside the arena. The second challenge is to detect the pucks and give an output of the number of pucks seen by the robot. In addition, telling which puck is the nearest one to the robot and calibrating the distance and angle that the robot should move with in order to fetch the puck from the shortest path.

4.1 Machine Detection

Machine Detection should support a set of different color combinations of lighting LEDs. Therefore, three modications are performed on each frame before detection.

Frame Brightness Since the area of interest is detecting which LED is turned on, the focus should be on the bright parts of the image to detect their color. This is done by decreasing the brightness of the whole image, ending up with an image where every pixel is almost black except the direct light parts.

Gamma Correction Gamma correction is the process used to approximate the brightness measurements performed by the camera[3]. It helps to deliver true colors and suitable luminance values in the image. In our model the gamma value 3 is chosen to be multiplied by the image pixels which gives more focus to the light parts and make their color appear more obvious with accurate colored value.

Colors Separation The last step is Colors Separation of the image into three different copies, one for yellow pixels, one for red and the third for green pixels where those pixels in each frame are then enclosed together into contours with smooth boundaries and detected as a lighted LED according to its area.

4.2 Puck Detection

The second challenge is Puck Detection; it follows the same concept of detection using colors. However, Gamma correction is not used as there is no luminance value this time unlike the machines.

Colors Separation The image is separated into three channels R, G and B. The red channel is converted to binary values, which is done by setting the value of each pixel to one if it exceeds a pre-set threshold, otherwise it is set to zero. Then, according to the neighboring red pixels-area, a puck is detected.

Distance of Puck The distance between the camera and the puck is calculated using the Camera Calibration. Before the match starts the robot's Camera is calibrated through detecting a chessboard pattern that is placed at a fixed distance and generates the inverse homo-graphic matrix that can be multiplied by the location of any pixel in the image to give its correspondent 3-D coordinates in real world. So by applying the same method, the distance of any 3-D object in the image could be calculated during the match.

Angle of Puck To calculate the angle of the puck, a semicircle is drawn with a radius that allow it to cover the important pixels of the image. Then, for each degree of the 180 degrees a line is drawn connecting the origin of the image and a point on the border of the semicircle, as shown in the figure below.

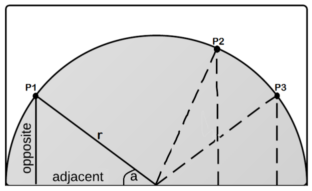

Fig. 5. The figure shows how the coordinates of the points on the semicircle boarder are calculated.

Using Bresenhams algorithm, each pixel is assigned to one of the 180 lines, where this line represents the angle of the pixel inside the image.
Another concept is also implemented, where distance and angle between the puck and the center of the robot are measured according to the equations discussed in the figure below.

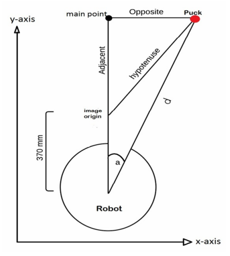

Fig. 6. The figure explains how the distance (d) and angle (a) between the puck and the center of robot is calculated.

$$Opposite = \sqrt{hypotenuse^2 - adjacent^2} \tag{1}$$

$$\angle a = \tan^{-1}(\frac{opposite}{adjacent + 370}) \tag{2}$$

$$d = \sqrt{opposite^2 + (adjacent + 370)^2} \tag{3}$$

5 Integration

The components were integrated using the technique shown in figure 7. The integration phase starts with the agent; which acts as the main link between the server and both Vision and Motion Planning components. First of all, the server sends a command about what needs to be done by each one of the three robots to their corresponding agent; who decides whether it needs to check anything in the surrounding world using vision or not. If yes, it will ask the vision component about the machines state or the pucks position, and according to the output it gets from the vision component, it sends a command to the motion planning component telling the robot exactly which machine to go to, in addition to telling it whether a puck needs to be fetched from the store or from another machine first. It also sends the Motion planning component the states of machines according to the vision output. After the robot has arrived to the desired location, motion planning returns back to the agent who subsequently updates the server with the robot's new location and state. Sometimes the agent needs to go to a certain location first, and then checks the surroundings to decide what to do. Therefore it starts by calling motion planning first to move the robot to the desired location followed by calling the vision component to check the machine state. So Eventually the agent controls everything the robot does whether by giving the robot some input from the sever regarding what the robot should do or by taking some output updates from the vision or motion planning components and sending these updates back to the server to decide what to do next.

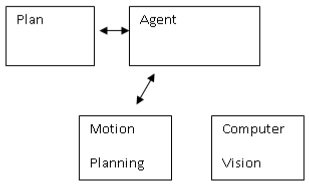

Fig. 7. Integration hierarchy overview

6 Conclusion

The problem was approached as an industrial simulation problem. It was considered as a multi-agent collaboration problem with the goal of producing the outputs decided by the referee box. The task was divided on a planner which had a centralized plan and communicated with the field agents to execute the plan and provide feedback to the centralized planner. In order to allow for the agent movement across the arena, a motion planning component was introduced. The component relied on both static and dynamic motion to move the robot whilst avoiding obstacles in the arena. Finally, a computer vision component was added to ensure that the robot can view its world; it was mainly responsible for puck and machine detection. All components were integrated using a centralized class agent.

References

1. M. Lekavy and P. Navrat: Expressivity of STRIPS-Like and HTN-Like Planning. Lecture Notes in Artificial Intelligence, Vol. 4496 Agent and multi-agent Systems. Technologies and applications. 1st KES International Symposium, KES-AMSTA 2007, Wroclaw, Poland, May/June 2007. - Germany, Springer-Verlag Berlin Heidelberg, 2007. pp. 121-130
2. Mat Buckland: Programming Game AI by Example, 2004
3. Andreas Siebert: Differential Invariants under Gamma Correction Computing Research Repository - CORR , vol. cs.CV/0003, 2000