Proceedings of the 2007 IEEE Symposium on
Computational Intelligence and Games (CIG 2007)

Snooker Robot Player - 20 years on

Kenneth H.L. Ho
Kenneth.hl.ho@gmail.com

Trevor Martin
University of Bristol, UK
trevor.martin@bris.ac.uk

Jim Baldwin
Jim.Baldwin@bris.ac.uk

Abstract—**This paper describes the Snooker Machine, an intelligent robotic system that was built between late 1985 and early 1988. The project was documented by the BBC over the course of 2 years. "The Snooker Machine" was broadcasted on BBC's territorial channel in the UK on the one hour Q.E.D. science programme of 16th March 1988.**

This paper summaries the technical details of the system. It consisted of a vision system, a fuzzy expert system and a robot manipulator. It outlines some of the difficulties that the Snooker Machine had to overcome in playing a game of snooker[1] against a human player. Given the recent interests in developing robotic systems to play pool[2], [3], [4], this paper looks back at some of those issues. It also outlines some computational intelligence approaches that may lead to solving some of the problems using today's technology.

Keywords: Computational Intelligence, Artificial Intelligence, Fuzzy System, Expert System, Computer Vision, Robotics, Games

I. Introduction

In the mid 1980s there were significant developments in utilizing information technology to automate complex tasks in business and in manufacturing industry. Computer Integrated Manufacturing (CIM) was beginning to be put into practice in factories. At the time, automated manufacturing was mainly based on pre-programmed inflexible tasks within fixed locations. There was a need for intelligent robots, capable of sensing, making decisions in order to handle irregular components and complex tasks.

The Snooker Robot Player project[5] was first envisioned and led by Prof Koorosh Khodabandehloo, who headed the Robotics and Manufacturing Systems Research Group at Bristol University. The project was joined by Prof Jim Baldwin and his AI group at Bristol. The idea was to use Snooker[1], a game of pool which was very poplar in the UK, to demonstrate the fundamental principles of artificial intelligence in robotics. It was believed that the basic technology for developing such an intelligent robot would be transferable to handle complex industrial tasks by adapting to an ever changing environment. An intelligent robot would be a robot which has to deal with uncertainty in sensing its environment, making complex decisions based on available information at the time, adapting to physical restrictions or its own limitations either through its own knowledge base or through sensing its environment.

II. Project Objectives and System Requirements

The main characteristics or requirement of an intelligent robotic system include the following:

1) The system must operate automatically, making decisions with the use of sensory data. An intelligent robot must be able to search for alternative solutions in case of difficulty.
2) Data is gathered using sensors and knowledge is updated by learning from previous actions.
3) An intelligent robotic system should be able to deal with uncertain situations and conflicting information. It should be able to derive a possible solution or to issue a further set of actions in order to resolve the situation.

One of the main objectives of the project was to push the frontier of intelligent robotic systems and to demonstrate that such a system could be transferred and be used in industrial applications. It was then decided that equipment and software packages used for the project had to be off-the-shelf. Tailor-made solutions were kept to a minimum.

The other main objective was to explore and to model uncertainty, imprecise definitions, vague human concepts and strategies in human decision making through the game of snooker. Snooker would give AI researchers a significant number of challenges in dealing with knowledge representation of an environment that did not have fixed predefined positions, e.g. a game of Chess. Human players could not easily articulate the reasons of their decisions while playing snooker. It would be even harder to represent that knowledge in a coherent and logical system.

III. System and Hardware

The main system consisted of three separate sub-systems:

1) a robotic arm, i.e. a mechanical manipulator.
2) a CCD camera which was mounted on the ceiling directly above the snooker table. The camera was attached to an industrial vision system.
3) a decision making expert system running on a separate workstation.

The choices of some of the following hardware were based on availability, affordability and suitability from our sponsors.

A. The Robotic Manipulator

In the beginning, we had tried to use a Puma robotic arm, a general purpose 6 axis mechanical manipulator. There were a few drawbacks. One was that the PUMA had very limited reach even on a small size table. The second drawback was the limited speed and power of the manipulator. Without re-programming its original controller, we found that it would be rather difficult for it to hit many shots. Finally it did not have the accuracy that we needed.

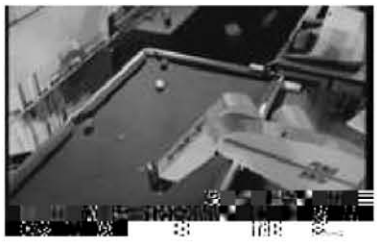

Fig. 1. Snooker Robot - using a PUMA robotic arm

Next we tried a much larger Redifussion robotic manipulator. It had more reach and better accuracy, but it lacked speed.

Fig. 2. Using a Redifussion robotic arm

It was then decided that a tailor made cue was necessary. A pneumatic cue was designed and created to solve the problem. The cue had a linear actuator which was attached at the end effector of the robotic manipulator. This allowed the system to position accurately and by the use of the pneumatic actuator, it would generate enough power and speed to hit the cue ball.

Fig. 3. Potting the ball with a pneumatic cue

The new pneumatic cue was not without its own deficiency. Due to the fact that it was pneumatic in nature, it could only take open or close positions, there was no effective way to control the power of the shot. It also meant that it would be rather difficult to control the placement of the cue ball.

As for the large Redifussion robotic arm, it could only reach slightly more than a quarter of the area of a small size snooker table. In order to play a full game, another robot manipulator was needed. We eventually chose to use a gantry robot, an IBM 7565 Assembly Robot[6]. We also chose to use a much smaller 4 ft x 6 ft snooker table. This allowed the manipulator to reach most part (around 90Its accuracy and repeatability over the table were not uniform due to the limited of its mechanical joints and control algorithm.

Fig. 4. The IBM 7565 Assembly Robot and the new snooker table

With the new IBM robot, a new pneumatic cue was used. It was lighter, and smaller in order to fit with the new but much smaller end effector. Given that the table was smaller, the cue ball only needed to travel a much shorter distance. The snooker balls were also smaller and lighter than standard snooker balls. Therefore, we could use a much smaller pneumatic piston on the actuator for the cue.

Fig. 5. The pneumatic cue on the IBM 7565 Assembly Robot

The new pneumatic cue was designed with a tapered angle. It allowed the cue to be positioned behind the cue ball unobstructed even when the cue ball was very close to the cushion or other snooker balls.

B. The Vision System

The vision system consisted of an industrial Automatix AV4 vision system, once made by Robotic Vision Systems Inc. The system came with its own image processing algorithms and applications and they were managed by a high-level structured language called RAIL.

The vision system was connected to a CCD camera which was located directly above the snooker table. This allowed the system to visualize the whole table without any obstruction.

By the beginning of 1988, another CCD camera was added onto the vertical part of the manipulator. This second CCD camera allowed the system to provide a more accurate calculation for the location of the cue ball.

C. The Fuzzy Expert System/Hardware

The fuzzy expert system was entirely developed on a separate computer system. In the beginning it was running on an IBM PC XT[7] which had 640KB of memory and a 4.77MHz Intel 8088 8-bit processor. It had a monochrome

Hercules graphics card which allowed it to display line graphics in moderate resolution. The system ran on the IBM-DOS operating system. The expert system was programmed in Arity Prolog. The graphics was written in Borland's Turbo Pascal. The main problem of using the IBM PC XT was the limitation of its memory. Slow computational speed was another drawback. By 1987, the fuzzy expert system was moved to an IBM RT[8] workstation running IBM AIX operating system and the system was re-written using Fril, an advanced fuzzy programming language developed at the University of Bristol.

D. System Integration

Nearly all the components of the system were off-the-shelf industrial equipment, they were not designed to communicate or to interface with one another. The only common interface among the components was the RS-232 serial interface.

The final configuration of the system in early 1988 can be found in figure 6. The vision system would capture an image from the CCD camera. It would then process the image data and provide a set of x-y co-ordinates of the balls' positions on the table to the fuzzy expert system through the serial RS-232 interface running at 9600kbps. The fuzzy expert system would then make a decision based on the positions of all the balls on the table. It would then issue a command to the robotic manipulator. It consisted of the preferred cue location in x-y co-ordinates and a directional vector. The command was sent to the robot manipulator via the vision system. The communication between the vision system and the fuzzy expert system was bi-directional. However, the communication between the vision system and the robot manipulator was unidirectional. The vision system could only send commands to the robot manipulator, no feedback was required from the robot manipulator.

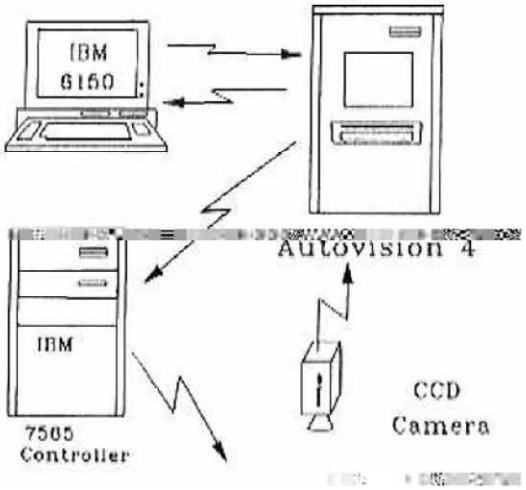

Fig. 6. Snooker Robot Player - system configuration

IV. Computer Vision - ball identification

The AV4 vision was programmed to identify twenty three coloured balls on the table using its own built-in image processing library. The problem was made easier under stable lighting condition as gray scale values remained constant. The basic algorithm for ball identification was as follows:

1) Determine the brightest and the darker gray values of the table, representing the upper and lower bounds of the background colour. This was used to locate balls outside the this band (see figure7 and 8).
2) Locate the objects and check whether the area of each object was compatible with the expected location of the balls.
3) Check whether the object had a circular shape.

Figures 7and 8 indicates how gray scale values were distributed around the playing area and how balls could be missed or mis-identified.

When a ball was identified, its location was obtained by recording the centre of the ball area in x and y pixel co-ordinates. These co-ordinates were used as reference point to obtain an average value of the ball colours by counting

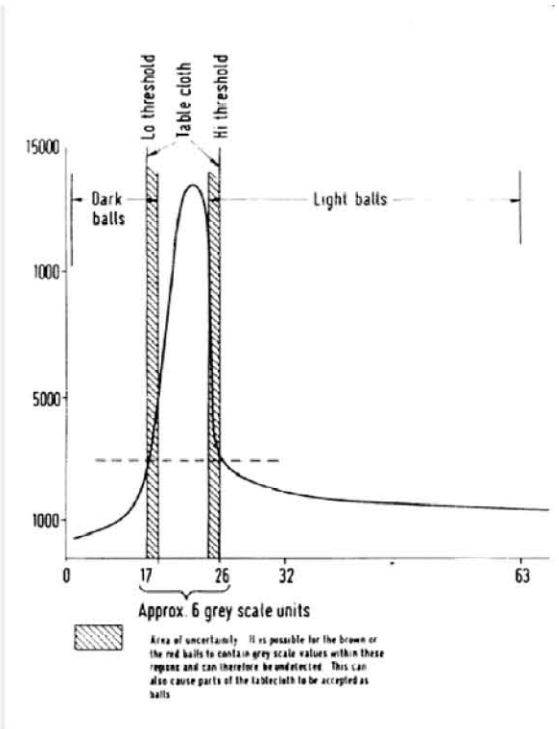

Fig. 7. Grey Scale Distribution

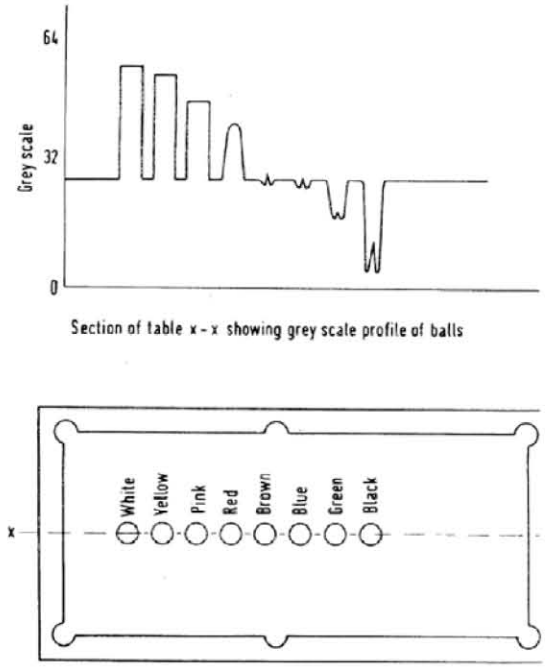

Fig. 8. Grey Scale Profiles of Coloured Balls

the gray scale value in a 7 by 7 pixel square around the ball's centre. The central 3 by 3 pixel area was ignored as this normally consisted of the direct reflection of the lighting from the celling. The gray scale level in this 3x3 area, if included in the calculation, would shift the average gray scale value of the ball's colour towards the gray scale value of light coloured balls (see figure 9).

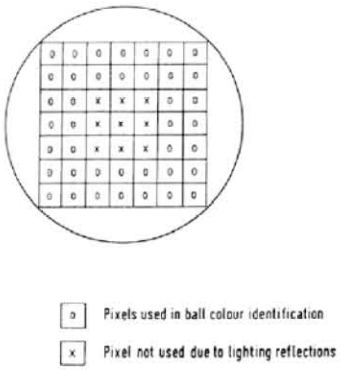

Fig. 9. Pixels used in Calculation of Ball Colours

The average gray scale value of the remaining pixels was assumed to represent the colour of the ball. This procedure was repeated for all the balls on the table and a hierarchy of ball gray scale values was obtained. The normal order of bright to dark balls was known to be white, yellow, pink, red, brown, blue, green and black (see figure 8). Therefore, from the hierarchy of gray scale values, the first 3 values were given the labels white, yellow and pink. The last 4 values were given the labels brown, blue, green and black. The remaining balls were given the label red.

The main drawback of this technique was that it was very sensitive to the ambient light variation. Calibration of the vision system was done during the boot up process and sometimes a re-calibration was needed.

Markers were also positioned on the side cushions of the snooker table for the vision system to identify the boundary of the table and to be used for calibration.

V. Modeling Uncertainty - Fuzzy Expert System

Knowledge-based approach was used to represent high-level human concepts, knowledge and strategies in playing a game of snooker. Historically the first expert system that manages uncertainty was developed by Buchannan and Shortcliffe, the medical expert system MYCIN[9]. The main drawbacks in most systems that need to manage uncertainty are the rules of combination of evidence. MYCIN used a set of ad hoc rules to derive its solutions.

For the Snooker Machine, we employed a much more rigorous approach which was also developed at the University of Bristol by Baldwin and his associates. A support logic programming[10] paradigm was used for the representation of knowledge relationships (i.e. heuristic rules) with fuzzy and probabilistic uncertainties. This provided us a methodology which allowed the integration of different sources of uncertain evidence using a calculus which is a generalization of probabilistic reasoning.

In this section a brief introduction of Fril and Support Logic programming is given before we describe how we apply them to our fuzzy expert system.

A. Support Logic and Fril[11]

The language Fril[11] embodies all the feature of Prolog[12] and in addition to this, Fril can handle knowledge bases containing uncertainties. The syntax of Fril is very similar to the syntax of Micro-Prolog[13] and differs from the more popular Edinburgh Prolog[12].

In Fril a clause or rule has the form

((head)(body))(support pairs)

e.g.

$((a)(b_1)(b_2)...(b_n)) : (S_n S_p)$

where a is an atom and $b_1, b_2, ..., b_n$ are conjunctions of literals. S_n, S_p are called the necessary and possible support of a given $b_1, b_2, ..., b_n$ are true. If the body of a is empty, then a is a unit clause, i.e.

$$((a)) : (S_n S_p)$$

The inference under uncertainty is called support logic programming. In the support logic programming system, Prolog type statements are not necessarily true but can be supported to a certain degree by an additional support pair. Each support pair shows the support for and against the truth of the clause. A support pair in Fril is defined as an interval between 0 and 1 in which the unknown probability of the clause lies. A statement or clause with a support

pair of (1 1) represents the fact that the statement is true or has a probability of 1, whereas a support pair of (0 0) represents false or probability of 0. If nothing is known about a particular statement or fact, then the appropriate pair would be (0 1) indicating that the probability of the fact being true lies between 0 and 1.

The calculus of support logic is based on Baldwin's Mass Assignment Theory. Details of the calculus can be found in [10] and [14]. A summary of it is given here.

Support logic conjunction, disjunction and negation can be summarized as follows: Given two facts a and b

$$((a)) : (S_n^a S_p^a)$$

$$((b)) : (S_n^b S_p^b)$$

and assuming that a and b are independent, the supports for conjunction and disjunction of a, b are

$$(and\ (a)(b)) : (S_n^a \cdot S_n^b S_p^a \cdot S_p^b)$$

$$(or\ (a)(b)) : ((S_n^a + S_n^b - S_n^a \cdot S_n^b)(S_p^a + S_p^b - S_p^a \cdot S_p^b))$$

In addition Fril provides an inference mechanism to combine a set of rules, this is, compound statements made up of propositions joined together with logic operators. The inference rules is based on the theorem of total probability.

$$\begin{aligned} Pr(head) &= Pr(head|body) \cdot Pr(body) \\ &\quad + Pr(head|not\ body) \cdot Pr(not\ body) \end{aligned}$$

Therefore, if we have

$$((head)(body)) : (S_n^{(h|b)} S_p^{(h|b)})$$

$$((head)(not\ body)) : (S_n^{(h|\neg b)} S_p^{(h|\neg b)})$$

$$((body)) : (S_n^b S_p^b)$$

then using the theorem of total probability, the necessary and possible supports for $head$ to be true are

$$S_n^h = \begin{cases} S_n^{(h|b)} \cdot S_p^b + S_n^{(h|\neg b)} \cdot (1 - S_p^b) & : \quad S_n^{(h|b)} \leq S_n^{(h|\neg b)} \\ S_n^{(h|b)} \cdot S_n^b + S_n^{(h|\neg b)} \cdot (1 - S_n^b) & : \quad S_n^{(h|b)} > S_n^{(h|\neg b)} \end{cases}$$

$$S_p^h = \begin{cases} S_p^{(h|b)} \cdot S_p^b + S_p^{(h|\neg b)} \cdot (1 - S_p^b) & : \quad S_p^{(h|b)} \leq S_p^{(h|\neg b)} \\ S_p^{(h|b)} \cdot S_p^b + S_p^{(h|\neg b)} \cdot (1 - S_n^b) & : \quad S_p^{(h|b)} > S_p^{(h|\neg b)} \end{cases}$$

In Prolog, if a clause does not exist in the knowledge base, the negation of the clause is assumed to be true. In support logic, if a clause does not exist in the knowledge base, it is assumed to have a support of (0 1), i.e. totally uncertain.

Fril also provides additional calculus to combine supports from different perspectives. This corresponds to a predicate with multiple clauses from which supports can be derived differently from different proof paths from an instantiated goal. Fril provides two choices of reasoning mechanism for multiple perspectives:

- the intersection law - it combines supports from different proof paths by finding the intersection of all the probability intervals, provided that the probability intervals overlap each other.
- Dempster-Shafer renormalization law[15] - Dempster's law assumes that the multiple support clauses correspond to independent viewpoints which can contain conflicting evidence. This law assigns supports for joint occurrence of different sources of evidence and redistributes the estimated conflict by a process of renormalization regardless of whether there is any foundation for such conflict. This law is not based on the theory of probability and it is not consistent with the Mass Assignment Theory.

B. Playing Snooker with Fuzzy Rules

The fuzzy expert system for the Snooker Machine was first implemented on the IBM PC XT using SLOP (Support Logic Programming Language)[5] [16] which was a predecessor of Fril. SLOP's inference engine was written in Prolog, so it was rather slow and limited. The system was re-programmed in Fril when it was migrated to the IBM 6150 RT system.

The fuzzy expert system consists of 3 parts:

1) Mathematical primitives - functions which calculate distances, angles between balls and pockets.
2) Mathematical definitions - consists of equations defining the path of the cue ball or the path of an object ball.
3) Fuzzy expert rule - consists of predicates that define vague imprecise human concepts, e.g. an easy shot. It also includes rules that collect evidence from the situation and making a decision through Support Logic, e.g. whether to play a break or to play safety[1] instead.

C. Definition of a Fuzzy Easy Shot

The concept of an easy shot can have the following dependencies (see Fig 10):

- the angle between the cue ball path and the target ball path,
- the distance between the cue ball and the target ball C
- the angle between the target ball path and the pocket P.
- the distance between the target ball C and the pocket P.

However, it is not all so easy to define an "Easy Shot" by using those parameters. By using Fril and Support Logic, one can define such a concept separately and let Fril combine the evidence together.

A "easy shot" can be defined as follows in Fril:

```
((easy_shot C X Y P)
                (possible_shot C X Y P)
                (straight C X Y P)) (0.7  1)

((easy_shot C X Y P)
                (possible_shot C X Y P)
                (ball-close C X Y P)) (0.6  1)

((easy_shot C X Y P)
                (possible_shot C X Y P)
                (full-pocket C X Y P)) (0.6  1)
```

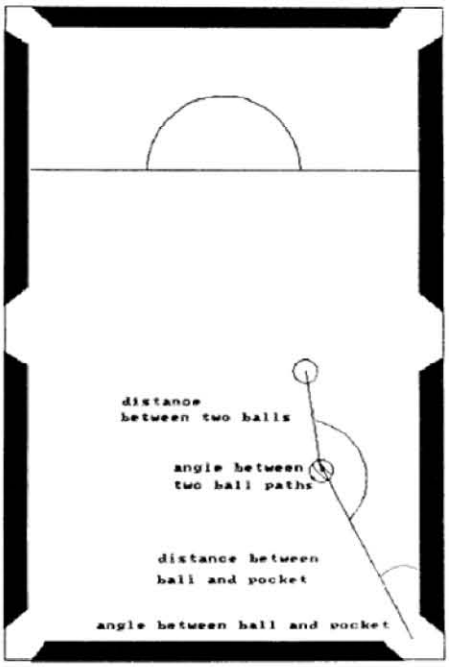

Fig. 10. Defining a Fuzzy Easy Shot

```
((easy_shot C X Y P)
                    (possible_shot C X Y P)
                    (near-pocket C X Y P)) (0.9  1)
```

Possible_shot was defined as the path of the cue ball the the target ball of colour C, positioned at X-Y is clear and the path between the ball and the pocket is not blocked.

For the fuzzy definition of the easy_shot, the 1st Fril clause means that there is a subjective probability of 0.7 to 1 if it is a possible shot and if it is a straight shot, i.e. the angle between the cue ball path and the target ball C to pocket P is small.

2nd Fril clause says that there is a subjective probability of 0.6 to 1 if it is a possible shot and that the cue ball and the target ball C are close to each others.

Similarly the 3rd clause means that there is a 0.6 to 1 chance that it is an easy shot if it is a possible shot and one can also see a full pocket. The final clause means that there is a much higher probability, 0.9 to 1, to be an easy shot if it is a possible shot and that it is near the pocket.

By using Fril's calculus, the fuzzy expert system would combine all the evidence collected from those 4 clauses, i.e. different proof paths and derived a final support for whether the shot is an "easy shot". The early SLOP and Fril implementation used Dempster Shafer rule to combine the supports from the 4 clauses.

D. Fuzzy Concept of Ball Closeby - a recursive definition

In this definition, any two snooker ball, which are less than 55 millimeters apart, are said to be "closeby" together, while any balls which are more than 3 meters apart are treated as not close together.

Suppose we move the two close balls a little further apart, say 220 millimeters, the two balls may still be close together bu the support for them to be close will be less. Similarly if we move the two balls which are "not-close", a little closer to each other (e.g. 220 mm), the two balls are still not-close together with some support.

```
((Ball_closeby C X Y P)
                    (cue_ball Xc Yc)
                    (distance D Xc Yc X Y)
                    (ball_close D))

((ball_close D)(closeby D)):(1 1)

((ball_close D)(NOT (closeby D)):(0 0)

((ball_close D)(not-close D)):(0 0)

((ball_close D)(NOT (not-close D)):(1 1)

((closeby D)(lessthan D 55))
((closeby D)
          (sum S D 220)
          (closeby S))(0.9   1)

((not-close D)(largerthan D 3000))
((not-close D)
           (sum D S 220)
           (not-close S))(0.9   1)
```

E. Snooker, strategy and knowledge representation

Snooker is a very demanding and complex game. Although we had found a way to model uncertainty and defining some fuzzy concepts, it was a far cry from modeling knowledge of a human snooker player. During the course of the project, we had the opportunity to talk with the then World Champion Mr Steve Davis. Mr Davis was able to give us extremely valuable insight and knowledge about how a professional snooker player would make the decision. However, some of the concept of a pack of balls, lose reds, etc, were not easy to define and represent. In order to capture the knowledge of how experts make their decisions, we had to represent those concepts in a way that is similar to an Opening Library of a Chess program. In order to code a map of the table into the computer, we decided to represented the table in a grid. A pack could then be defined as more then 3 or 4 balls within a fuzzy area within the grid. Similarly experts' knowledge could then be represented in this fashion.

By the end of 1987, we had only built a small set of rules within the library and they were mainly for the opening shots. In order to make the machine to play like a human, one has to control the placement of the cue ball. Given that we did not have the capability to control the cue ball with the pneumatic cue, it was hard to assess the efficiency of those expert library rules.

VI. Performance Against a Human Snooker Player

By early 1988 a game between the Snooker Machine and a human snooker player , Mr Ted Lowe - a famous BBC commentator, was staged and was filmed by the BBC as the finale of the Q.E.D science programme. Robots and machines were starting to rival humans in terms of intelligence and in dealing with a changing environment.

Snooker Machine performed reasonably well given its hardware and software limitations at the time. The system was able to have minor breaks and was able to keep up with the score of the human player. The system was also able to clear all the colour balls achieving a 27 points break on the table in one of its practice runs.

The system required a couple of minutes to process the image data and then another couple of minutes to make a decision through the Fuzzy Expert System before the robotic manipulator could move into positions to take a shot. On average it took something around 3 to 5 mins processing time for each shot.

The Snooker Machine was surprisingly good and accurate in potting balls even when some of them were not "easy shot". We estimated that the system would probably miss a pot in every 3 to 4 shots, i.e. if a cue ball and a ball were put on the table randomly, the Snooker Machine could probably pot the ball into a pocket 66-75% of the time, although no statistics or actual recording of its performance was done at the time. The main problem was the accuracy of estimating the locations of the balls via the vision system.

We had also simplified the decision making algorithm and used a much simpler algorithm to speed up the computational process, e.g. pot the red closest to a pocket. Due to its surprisingly good accuracy in potting balls, we did not find a degradation on its performance even when we switched to the simple algorithm instead of the more complicated Fuzzy Expert System.

The human snooker player was also handicapped in several ways. Due to the structure of the gantry robot manipulator, the human player could not position comfortably on some of the shots where the structure would block the player. Secondly the table and balls were much smaller and lighter than the actual balls, it was much more difficult for a human player to position and control the cue ball.

VII. What We've Learned

The Snooker Machine was surprisingly accurate in potting balls. Given that if one can keep potting balls into the pockets, it would be rather difficult for the other player to win the game.

We therefore concluded that the most efficient way to improve the performance of the Snooker Machine was to increase the accuracy of detecting the location of the cue balls and all the other balls on the table.

In a way the best computational solution for beating a human snooker player might not be to model how a human player plays snooker. Instead it would be best to utilize the computational power, the accuracy of vision system in identifying and locating the balls on the table and the accuracy and repeatability of the robotic manipulator in hitting the cue ball.

This would be similar to a Chess playing programme, instead of modeling human knowledge and behaviour in playing chess, Deep Blue was able to beat the human World Champion by searching for the positions of each chess piece exhaustively with raw computational power.

VIII. 20 years on - what can we do differently

Twenty years on, science and technology have been improving in a phenomenal pace. In 1986 a PC was an IBM AT running at 16MHz. Currently PC has dual core processor running at 2GHz or more. Memory at the time was within 1MB range, while most machines now feature 1GB of memory. Both memory and processor speed have improved 1,000 times in performance. CCD cameras have also improved tremendously. Most current mobile phones would have a camera, a processor and memory that rival what we had in making the Snooker Machine.

In this section we shall look back at some of the problems that we faced and see how we can do it differently with today's technology.

A. Computer vision - ball identification

As we have mentioned earlier we had encountered problems in correctly identifying the colour of the balls due to the sensitivity of ambient lighting condition. We solved that by controlling the environment with blinds and powerful lighting. The problem with glare from reflection of the lighting on the celling was solved through making some assumptions.

With today's technology, we could have solved those problems with colour computer vision. Given that the current technology has a much better resolution, we would like to think that colour vision will also solve the problem of identifying the cushion of the table without specific markers.

B. Computer vision - Ball localization

There were problems with ball localization due to shadows of the ball and glare of the snooker balls due to the reflection of the lighting. The accuracy of each shot depended on the correct localization of the cue ball and the other snooker balls. If we were to do that today, we would probably use multiple cameras to eliminate the problem with shadows. It would also be possible to do it with stereo vision.

As in our case, we were able to use another camera mounted on the robotic manipulator to get a closer look at the snooker balls on the table. We were able to improve the performance of finding the centre of the ball although it would take a much longer time in computation.

With today's technology in digital camera, we would be able to put a small digital camera at the end of the cue tip. It would also be able to track the motion of the robotic manipulator in real time and to make sure that the cue will hit the centre (or the preferred area) of the cue ball.

C. Pneumatic cue

As we have mentioned earlier, the pneumatic cue can only be on or off, i.e. we could not control the speed of the cue and with that the placement of the cue ball. At the time there was plans to adjust placement of the pneumatic cue behind the cue ball. Therefore, it would have some control of the distance of the cue could have travel before hitting the ball, thus controlling the speed of the cue ball. It would have been possible to do some mathematical calculation for that. Alternatively we could have used neural network or fuzzy system approach to find the correct correlation between the distance behind the cue ball and the distance that the cue ball would have traveled.

Given the limitation of the pneumatic cue, it may worth to consider an alternate design for the cue. One suggestion would be to use mulitple cues. The manipulator would change its end effector depending on the circumstances of the environment. For example, a pneumatic cue for long shots or opening shots when accuracy of controlling the cue ball would not be of prime concern. In other circumstances an electric motor driven mechanical cue could be used for precision shots or shots that need fine control of the cue ball.

D. Coverage of the Snooker table

We had problems in getting the robotic manipulator to cover 100% of the table. By around the end of 1988, a much larger gantry robot was installed and a small 6 axis manipulator was attached to the end effector of the large gantry robot. The small 6 axis manipulator would be able to give a much finer and more accurate control for the Snooker machine while the large gantry robot would allow the whole machine to cover a full size snooker table.

It would also be possible to mount a 6 axis manipulator on top of a mobile platform in order to provide 100% coverage of the table.

E. Knowledge-based Approach

Do we need intelligent algorithms if given two balls (a cue ball and another ball) on the table randomly, the machine can pot the ball 99% of the time? If the objective is to play a competitive game of snooker, it would be interesting to see whether a simple decision algorithm would be sufficient. On the other hand, if we were to create a machine to play like a human snooker player, i.e. to pass a Turing Test, then the modeling of human behaviour and their decision would be very important.

If we were to model human decision in a snooker match today, we would probably collect data from a snooker match using a video camera or even from televised footage. It would be possible to locate the relative locations of each snooker ball on the table through computer vision. A data mining or machine learning algorithm might be able to condense and generalize those data into a set of rules used in our Fuzzy Expert System.

In the end we had concluded that the Snooker Machine would not need more sophisticated expert system rules until the system could find a way to control the cue ball. It would have a dramatic improvement on its performance in the research in sensing and control.

IX. Conclusions

In conclusion the Snooker Machine was very successful as an intelligent robotic system in playing snooker. It had also fulfilled some of its objectives in transferring some of the knowledge to other domains[17] [11].

Acknowledgments

The authors would also like to take the responsibility of any error or inaccuracy in this paper. All the materials presented are the recollection of the authors. The project was led by Prof K. Kohdabendaloo. Ian Rennell worked on the vision system and he integrated that with various robotic manipulators. Prof Richard Gregory contributed much of his time in this project. We also had tremendous help and support from our undergraduates. Many thanks to our chief technician Mr Terry Gorman for setting up all the equipment. Externally we had supports from the BBC, Lang Pneumatics, Automatix UK Ltd, IBM Bristol, and IBM Havant.

References

[1] Wikipedia, "Snooker," 2006. [Online]. Available: http://en.wikipedia.org/wiki/Snooker

[2] W. Leckie and M. Greenspan, "Pool physics simulation by event prediction 1: Motion transitions," *International Computer Gaming Association Journal*, vol. 28, no. 4, pp. 214–222, Dec, 2005.

[3] M. Greenspan, "Homepage of michael a. greenspan," 2006. [Online]. Available: http://post.queensu.ca/ greensm/

[4] J. Ghan, T. Radzevicius, W. Robertson, A. Thornton, and B. Cazzolato, "Pool playing robot," 2002, University of Adelaide, Department of Mechanical Engineering. [Online]. Available: www.mecheng.adelaide.edu.au/robotics_novell/projects/2002/Pool/final.pdf

[5] K. Khodabandehloo, I. Rennell, and K. Ho, "Robots with artificial vision and intelligence," in *Advanced Robotics Programme (ARASA) International Conference on Nuclear Robotic Technologies and Applications Present and Future, University of Lancaster, UK.* ARASA, 1997.

[6] IBM_Archives, "IBM 7565," 2006. [Online]. Available: http://www-03.ibm.com/ibm/history/exhibits/robotics/robotics_3.html

[7] Wikipedia, "IBM personal computer XT," 2006. [Online]. Available: http://en.wikipedia.org/wiki/IBM_PC_XT

[8] ——, "IBM 6150 RT," 2006. [Online]. Available: http://en.wikipedia.org/wiki/IBM_6150_RT

[9] B. Buchanan and E. Shortcliffe, *Rule-based Expert Systems*. Addiso-Wesley, 1984.

[10] J. Baldwin, "Fuzzy and probablistic uncertainties," in *Encyclopedia of Artificial Intelligence*, S. Shapiro, Ed. 2nd Edition, John Wiley, 1992, pp. 528–537.

[11] J. Baldwin, T. Martin, and B. Pilsworth, *Fril - Fuzzy and Evidential Reasoning in Artificial Intelligence.* Research Studies Press/John Wiley & Sons, 1995.

[12] W. Clocksin and C. Mellish, *Programming in Prolog.* Springer-Verlag, 1984.

[13] K. Clark and F. McCabe, *Micro-Prolog: Programming in Logic.* Prentice Hall, 1983.

[14] J. Baldwin, "A calculus for mass assignments in evidential reasoning," in *Advances in the Dempster-Shafer Theory of Evidence*, R. Yager, M. Fedrizzi, and J. Kacprzyk, Eds. John Wiley & Sons, 1994, pp. 513–531.

[15] G. Shafer, *A Mathematical Theory of Evidence.* Princeton University Press, 1976.

[16] J. Baldwin, "Support logic programming," *International Journal of Intelligent Systems*, vol. 1, pp. 73–104, 1986.

[17] K. Khodabandehloo, "Robotic handling and packaging of poultry products," *Robotica*, vol. 8, pp. 285–297, 1990.