A Motion Constrained Dynamic Path Planning Algorithm for Multi-Agent Simulations

T. R. Wan
Department of EIMC,
School of Informatics,
University of Bradford,
Bradford, West Yorkshire, UK,
BD7 1DP
T.Wan@Bradford.ac.uk

H. Chen
Department of EIMC,
School of Informatics,
University of Bradford,
Bradford, West Yorkshire, UK,
BD7 1DP
H.Chen3@Bradford.ac.uk

R.A. Earnshaw
Department of EIMC,
School of Informatics,
University of Bradford,
Bradford, West Yorkshire, UK,
BD7 1DP
R.A.Earnshaw@Bradford.ac.uk

ABSTRACT

In this paper, we present a novel motion-orientated path planning algorithm for real-time navigation of mobile agents. The algorithm works well in dynamical and un-configured environments, and is able to produce a collision-free, time-optimal motion trajectory in order to find a navigation path. In addition to the motion constraint path planning, our approach can deal with the unknown obstacle-space terrains to moving agents. It therefore solves the drawbacks of traditional obstacle-space configuration methods. Multi-agent behaviour has been explored based on the algorithm. In the simulation a simple physically-based aircraft model has been developed, which is addressing the manoeuvring capabilities of the moving agents, while the moving agents' accelerations and velocities are always continuous and bounded. The generated motion path is constituted smoothly and has continuous curvature on the whole state space of the motion, thus satisfying the major requirement for the implementation of such strategies in real-time animation or in simulation applications in VR environments.

Keywords

Motion modelling, Constraint motion, path planning, trajectory generation, multi-agents.

1 INTRODUCTION

Path planning with motion modelling is an important and challenging task that has many applications in the fields of AI, virtual reality, autonomous agent simulation, and robotics. The various approaches reported have different criteria to be met, which result in a number of algorithms, and provide solutions to specific application problems. The basic task for the motion constraint path planning is to perform navigations from one place to another by co-ordination of planning, sensing and controlling whilst maintaining a smooth motion trajectory. Navigation may be decomposed into three sub-tasks: mapping and modelling the environment; path planning and generation; and path following and collision avoidance. Path-finding is properly the most popular and frustrating game AI problem in computer game industries [Ari01, Tat91]. Early work was concentrated on offline planners. These methods used the map of the environment to produce a configured path [Oom87, Lum90]. The common ground for this type of method is that the planner must have full information about the environment [Pad00, Lat91]. Most of the current successful approaches lead to some sort of graph search strategy [Jos97]. An approach called line intersection was proposed when the data consisted only of geometrical objects. The objects here were solid and all the space not occupied by an object was considered unobstructed. There was no variation in vehicle speed or other parameters. The idea was to construct the convex hull of all objects using vertices and connect all vertices with edges. These edges are then filtered to find the shortest valid path between source and destination along a series of edges using standard graph algorithms. These methods suffer from the problem of a rapid increase in computation time and memory for large and complex maps [Mon87, Hol92]. Another popular approach is called weighted graph [Woo97], which divides the search space into a number of discrete regions, called cells, and restricts movement from a particular space cell to its neighbour. Neighbouring cells are those that can be directly reached from a particular cell. A weight function is defined by the cost of the connection

Conference proceedings ISBN 80-903100-7-9
WSCG'2005, January 31-February 4, 2005
Plzen, Czech Republic.

between neighbour cells [Woo97]. A* is the most popular algorithm of this kind, which uses the weighted graph idea. Recently an approach called path planning algorithm D* [Ste95, Yah98] was reported, which resembled the A* algorithm for applications in partially known environments, and only achieved limited success [Yah98].

Given that motion factors must be taken into consideration, the main problem is to generate a smooth trajectory curve by joining two distinct configurations in the space with constraints using interpolated piecewise polynomials. Curves have been an object of mathematical study and also a tool for solving technical problems and applications. The construction of smooth curves is the trajectory generation for moving agent path planning and motion control, for example, the simulation of an autonomous car or aircraft in a virtual environment, and the robot motion trajectory control in the real world. A C^2 smooth curve is necessary for the agent or its control system to track either in a virtual environment or in the real world. Motion trajectory was discussed by L.E.Dubins [Dub57], who proposed the solution using straight segments connected with tangential circular arcs of minimum radius and proved that the shortest distance between two configurations was such a path. However it is important to note that the curvature along such a trajectory curve is discontinuous; the discontinuities occur at the transitions between segments and arcs. The non-continuous curvatures may result in difficulties in agent control. Continuous-Curvature Curve (3C) generation has become a key technique for on-going research in this area. A few types of splines have been proposed to solve this problem. Yamamoto et al [Yam99] studied the B-spline-based path planning for finding the time optimal trajectory; Tomas et al [Ber03] used the bezier curve in path planning, having considered minimizing the square of the arc-length derivative of curvature along the curve. Y. Kanayama and N.Miyake [Kan86] suggested that using clothoid curve would form a smooth path with continuous curvature, and the resulted curve curvature varies linearly along the path. Later, Y.Kanayama and Hartman, B. I [Kan89] proposed another solution using a cubic curve. A. Scheuer and Th. Fraichard [Sch96, Fra01] used clothoid curves in their vehicle control experiment. Bryan Nagy and Alonzo Kelly [Nag01] adopted cubic splines in their trajectory generation algorithm. However, previous work has mainly been focused on the static trajectory generation problem and on finding the solutions for 2D applications.

In this paper, we propose a new approach for motion modelling for autonomous moving agents in virtual environments. We improve the conventional A* method by developing a dynamical visible point detection system, which allows the system to update its optimised node system based on the viewed vision field, rather than the pre-configured environments, to find a smooth motion path in real-time. Our method is capable of dealing with dynamic unknown environments using motion constraints and is very efficient in terms of computational cost.

2 UNKNOWN ENVIRONMENTS

One of our objectives in the current work is to study and develop a path planning algorithm for autonomous agent navigation or exploration in unknown environments. The task can be divided into three parts, plan a main path according to the pre-information, keep tracking the difference between the map and the real environment, and then locally amend the pre-designed path. This strategy can efficiently use the available information and reduce the re-planning time. Navigation in an unknown environment is a more challenging topic; at the same time it is also the most promising technology that could be used for generic applications. For example, unmanned robots with navigation ability in unknown environments could achieve tasks in many dangerous places that humans would not wish to entry for safety or health reasons. Navigation is an important part of AI. Navigation in an unknown environment means no pre-information is available before the path-planning algorithm has been executed. The self-guided agent uses the sensor equipped to detect the surrounding environment and obtain the local information. It then uses the local information to generate the path to the destination. In our current work, a virtual environment has been built, which consists of a terrain with unknown configuration and obstacles. A hierarchical strategy has been developed in the current work for creating a navigation environment using raw terrain data.

3 MOTION DYNAMICS

Representing all the motion characteristics by analytical equations can be unpractical. A simplified motion model is considered in the current work. The moving agent model developed has six degrees of freedom and the dynamics of the model can be represented as a set of motion parameters in terms of mass, accelerations and steering angles as well as external force conditions, such as air resistance or ground frictions. The dynamics of a moving autonomous agent must follow the basic law of motion dynamics, which may be represented as a set of general ordinary deferential equations in the form:

$$\frac{d^2X}{dt^2} = f(X, \dot{X}, d) \quad (1)$$

where, X, $\dot{X}$ $\hat{I}$ $\hat{A}$ which is the motion state of the agents and its first derivative, and d is the motion

control input. We can recast the equation for our motion optimisation problem in the form:

$$\frac{d^2X}{dt^2} = \hat{f}(X, \dot{X}, d) + \Delta(d) \quad (2)$$

$$\Delta(d) = f(X, \dot{X}, d) - \hat{f}(X, \hat{\dot{X}}, d(\hat{a}, \hat{q})) \quad (3)$$

where, $\hat{\dot{X}}$, $\hat{a}$ and $\hat{q}$ are approximate values of motion velocity, acceleration and direction of motion (i.e. a steering angle) respectively, and the motion control d is a function of acceleration a and moving direction q. A desired or predicted motion state of the moving object is pre-estimated by a set of approximate functions according to the state of moving object and the environment conditions related to the surrounding obstacle–space. The actual motion track is then computed. The difference between the predicted motion and the actual motion will be used for estimating the control input to the motion system above.

4 PERCEPTION AND CONTROL ARCHITECTURE

In our system the "visual sensor" captures the information about the environment. It is a simple method to compute which parts of objects can be seen from the location of the agent. A virtual camera performs perspective projection, all points along a line pointing from the optical centre towards the location of the agent are projected to a single point. We assume all the obstacles are opaque. The mutual occlusion of objects and self-occlusion are analysed. The maximum detection distance and view angle are then calculated. All the obstacles out of the maximum detection distance or view angle are assumed to be invisible. If in one direction there is no obstacle within the maximum detection distance, we can use the point at the end of the detection distance as the flag in this direction.

The motion control for the agent moves through a field of obstacles to a goal, which includes finding and predicting an optimised path and controlling its motion parameters in order to follow this path. We use the information from the virtual vision sensor to identify the key obstacle points and edges, then create and add the obstacle nodes and path nodes to the vision system. One of the advantages of our approach is that the generated desired or predicted path is dynamic but it is not necessarily the ones the mobile agent must pass exactly at a given time and the actual motion track is therefore smoother in terms of curvature. The position errors between exact desired path nodes and the actual motion track are then used to modify the motion parameters. Another advantage over other methods is that our approach is quite robust with respect to errors and external disturbances. If both errors and the disturbances are within certain bounds, the algorithm can still work effectively. The architecture of the system is shown in Figure 1.

5 PATH-PLANNING ALGORITHM

The Path-Planning in our work can be stated as follows: given an arbitrary rigid polyhedral object, P, and polyhedral environment, find a continuous collision-free and smooth motion trajectory path taking P from some initial configuration to a desired goal configuration. For a path planning problem, the A* algorithm appears to be an obvious option, and it is so far the most widely used searching algorithm associate with heuristic search. A* explores paths from an initial state in a systematic manner whilst paying a little attention to the path finding cost. It is the optimal solution under certain conditions. The A* path planning algorithm will execute actions (sequence of actions from a state to another in the state space) after the shortest path has been found. However, it is basically an offline search algorithm and can not be used for unknown or dynamic environments. Our method uses dynamically allocated points through on-line searching, sensing and reasoning in the environment. At any location of the environment, we could find the points of visibility that are concerned with the co-ordination of the agent. The information perceived is analysed and the resultant path points are recorded and used to

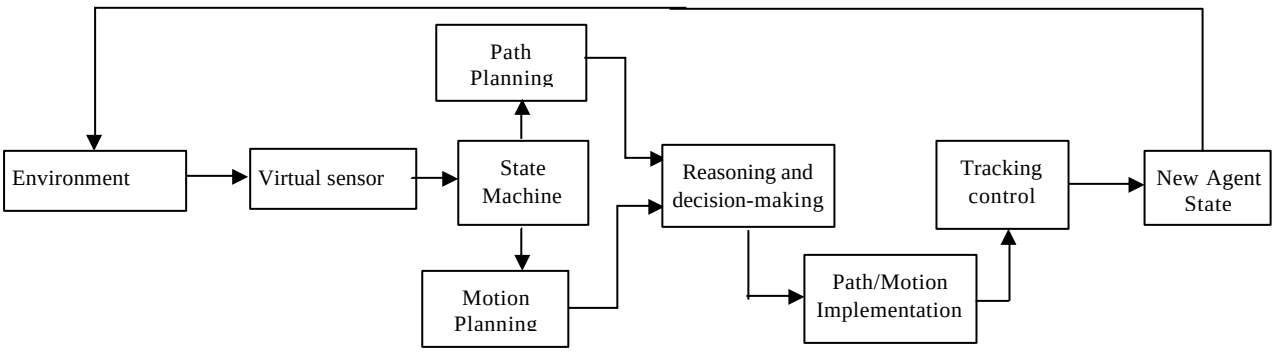


Figure 1: The architecture of the motion control system

construct memorised path nodes. The algorithm uses the angle to partition the obstacle-space, while keeping a safe margin, which allows the agent to plan its path and motion trajectory at any location in the environment. Penalty functions were introduced to make the state node with no obstacles a higher priority. It then chooses the lowest cost state as the local goal and keeps iterating until it reaches the destination goal.

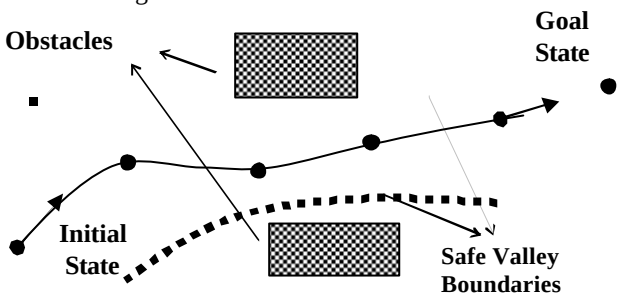


Figure 2: Creating a smooth motion path trajectory in a dynamically allocated safe-pass valley.

As noted earlier, in order to deal with unknown or dynamic environments, a suitable path planning algorithm must be a real-time one. The agent executes actions before finding the global solution. So it is an exploration in an unknown environment or a dynamic environment. A number of heuristic functions were defined, which could provide the least cost path estimated from the current state to a goal state and the actual cost estimated from the initial state to the current path state. The actions defined will return a list of possible solutions in the state space. Assuming that an action is deterministic, the agent might have access to an admissible heuristic function, which estimates the distance from the current to a goal state. The objective of this consideration is to reach the goal with shortest distance with a motion constrained minimum cost. For example, the agent is required to go through a complex obstacle block to reach a destination in a reasonable time whilst maintaining smooth motion characteristics. Figure 2 shows an example of generating a smooth path in the range of a safe valley identified by the agent's perception and reasoning.

6 BUILDING UP A DYNAMIC TREE FOR PATH FINDING

A dynamic path tree must be created for the searching process. The planning algorithm searches in a state space for the least expensive path from a start state to a goal state by examining the neighbouring or adjacent states of particular states (the states are formed according to the map representation). By repeatedly examining the promising unexplored location area, the algorithm will reach an end if a configuration is the final goal. Otherwise, it takes notes of all that location's neighbours for further exploration.

In order to avoid the agent being halted in a dead-end, we assume the path or actions executed are reversible. So if our real-time path planning reaches a dead-end state, where no goal state is reachable, the agent could seek to reverse it actions. It should be noted that no algorithm can avoid dead-ends in all state spaces. In cases where the agent reaches a dead-end, it must find a way back by its own reasoning according to the information available. According to the requirements of motion dynamics, it can not simply travel back to the state it previously visited. Instead a number of extra actions must be executed, and an extra set of states must be added, in order to return the motion back to the state. After that it should follow the path in a reverse direction until it finds a branch node, which was executed before. The algorithm does not make the agent follow the action path tree in a reverse order, because of the motion constraints. We must guarantee the motion is smooth or at least C1 continuity. Once the agent reverts to the nearest branch path node, the sequence of the path node state will be pruned from the path tree. The searching algorithm uses two data structures to record the path information, one is a list called Close to record the passed states, and the other is called a tree called PathTree to record the path map. At the start, Closed is empty, and PathTree has only the starting state. In each iteration, the algorithm removes the most promising neighbour

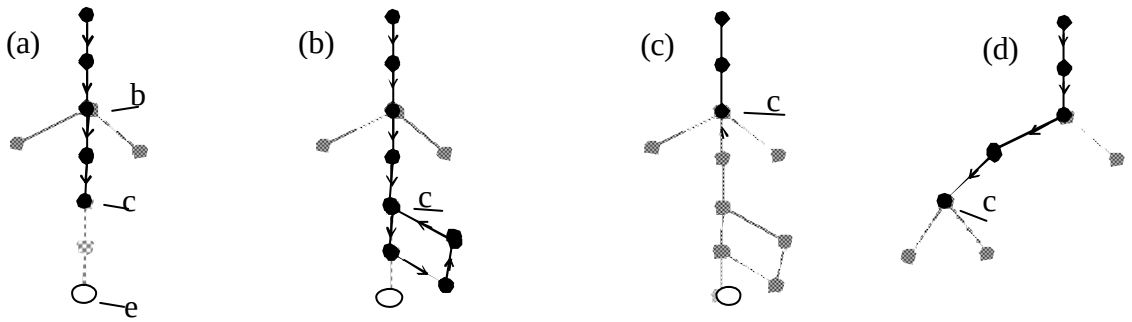


Figure 3: An illustrating example of building up a dynamical tree
b: Branch node; c: Current node; O: Dead-end.

state for examination. If the state is not a goal, all the other neighbouring states are sorted. If they are already in Closed, they are ignored. Otherwise they are kept in temporary locations. The algorithm will then perform a procedure called Merging to reduce the useless neighbour states. If some states could survive, then the current state is recorded as a split node, and the neighbours are recorded as crosses in PathTree. If no neighbour state is available, the PathTree will trace back to the nearest split node, and switch to a new cross. If all the crosses in the PathTree have been tested before the goal is reached, it concludes that there is no path to the goal from the specified start configuration. Figure 3 shows an illustrating example, in which an agent performed an on-line path search: It reached a dead-end in (a), and found a way back, (b), reached a branch node (c), and pruned the path sequence unsuccessful and planed the path in another route (d).

7 MOTION TRAJECTORY CURVE

Trajectory generation is important for the motion constraint path planning, because the moving agent will have to adjust or control its motion state to follow the trajectory reasonably closely whilst maintaining good motion characteristics. The clothoid curve is very useful because its curvature varies linearly along the arc. Kanayama [Kan89] proposed to use this curve for motion trajectory design. It is now the most commonly used curve type for highways and railroad design [Esv01]. It is chosen for generating a smooth path since it satisfies all the requirements for agent motion control tasks and modelling. The Clothoid curve is an intrinsic spline, it can not be expressed in a close form, and this is the biggest disadvantage and results in calculation difficulty. However, its curvature varies linearly along the arc, and the curve can be constructed from its curvature. On the other hand, since the parameter t is proportion to the length of the arc, it can be used directly as a trajectory. The derivative of the curvature of clothoid curve is a constant which is identical with the Bang-bang control theory in aiming at giving solution to the optimal-control problem. The clothoid curve is defined as following form:

$$Cv(s) = k * s + Cv_0 \quad \ldots (3)$$

where s is the arc length, $Cv(s)$ is the curvature and k is a constant. The direction of the tangent vector is the integration of the curvature and expressed by

$$q(s) = \int_0^s (k * s + Cv_0) ds = \frac{1}{2} * k * s^2 + Cv_0 * s + q_0 \quad \ldots (4)$$

In our work, three dynamically allocated points in a 3D space are identified at each state for each basic curve element configuration. Unlike a conventional approach, we do not restrict the curve as a symmetric pair since it may encounter difficulties in a global configuration. The global trajectory is made up of a set of local curve pieces and we should not only guarantee the smooth agent movement along local curve, but also the smooth transition between these curve elements. Our approach is to use an un-symmetric clothoid curve element plus two extra dynamic control points S_2 and S_4 to offer the more flexible and powerful solution to the trajectory generation problem. In practice, a symmetric pattern cannot always be guaranteed to be the optimal trajectory to follow, and the un-symmetric pattern is the general situation and offers more flexibility for the control process. In order to achieve a successful smooth connection at the joints between curve elements, keeping in mind that each destination state is also the new initial state for the next move step, the direction of the motion trajectory at the current destination location should be the same as at the next new initial agent location. The curvatures at the both locations should also be the same for smooth curvature transition. To meet these requirements, two transition points are introduced which are determined by the motion kinematical states of the agent to produce a smooth transition between curve elements.

As shown in Figure 4, the agent starts at S_1. S_3 S_6 are the points perceived, and S_2 , S_4 are the two points added to control the agent movement to make a smooth transition between adjacent clothoid curve elements, which are derived from the agent motion requirements that satisfy the agent kinematical conditions and the equations (3) and (4). The first local curve element ends at the destination point S_4, which is also the new initial position of the next clothoid curve element. At S_4, the curvature is decreased to zero and the direction is from S_3 to S_6, The trajectory generation will keep going as long as subsequent path points are supplied.

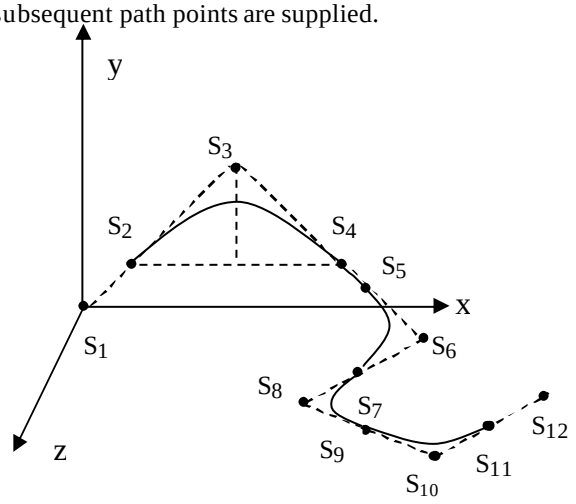


Figure4: An example of the global configuration

8 GROUP BEHAVIOUR

Our multi-agent path planning algorithm in unknown environments is developed based on the single agent path planning environments. In the multi-agent environments, introducing more than one agent simply will lead to poor performance if they are just simply added. Therefore the agent path planning algorithms must be changed to introduce elements for joint activities. The observation and prediction about other agents will be included in order to operate effectively and in a timely manner. Our method is to plan actions jointly via coordination and communication. In a simulation, the agent also acts as an obstacle or part of the environment, although they are not static most of the time. The path planning problem and strategy for joint activities is best constructed according to the goals specified for the location and motion states of each agent. The algorithm is summarised as follows:

Agent A: Initialise agents A, which including agents' state location, behaviours etc.
Goals: Assign specific goals to A, including a final goal and sub-goals
Actions: Move, how to move, perceiving environment, recognise other agents and the environment. Predict other's intention etc. Approaching other agents
Agent B:
…
Plan and strategy: Coordination, communication, competition and collaboration etc.

The algorithm has been implemented in our simulation and a basic exploration test was conducted at this stage.

9 SIMULATION RESULTS

A number of simulations have been conducted in our work for evaluating the motion trajectory generation algorithm. An auto-pilot-aircraft was created as an intelligent agent and it has the ability to perceive the visual information about the environment when it is in a navigation. Sensors have been created and attached to the aircraft.

Figure 5 show three instances of a path planning simulation. using the algorithm discussed above. Figure 5a shows that the moving agent moves to the entrance of an unknown alley, where there are three potential paths that have been created and a split node is created and added to the path tree. Figure 5b shows the agent moving back to the nearest split node. Figure 5c shows that the moving agent's successful navigation through the alley.

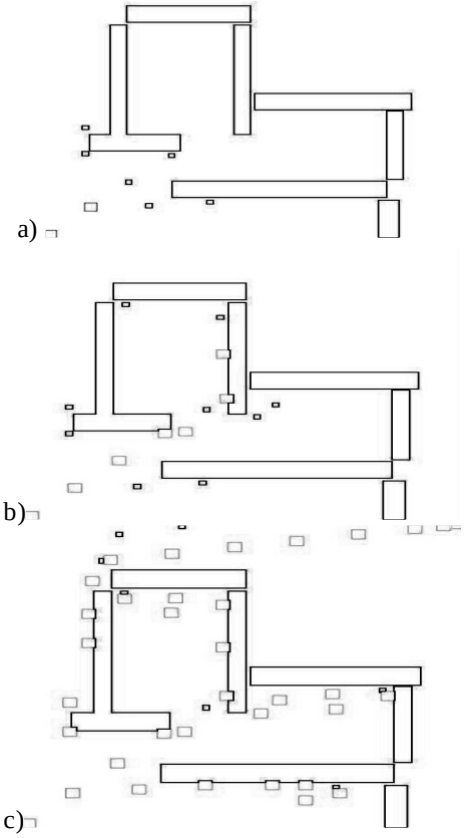


Figure 5: A simulation test; a. 1 merge paths and search optimal path b. exit from blind alley c. successful navigation through obstacle block

Figures 6 and 7 show two simulation experiments. The experiments were designed for an autonomous moving agent to navigate through complex obstacle environments with different motion characteristics. The simulations were quite successful and the results are satisfactory. Figure 8 shows a screen capture of real-time aircraft simulation in a virtual environment, in which the aircraft acted as an autonomous agent who used visual information to detect obstacles in order to find a path to conduct an obstacle-free navigation.

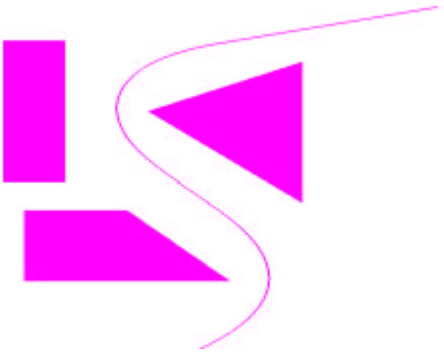


Figure 6 Navigation Simulation Case 1.

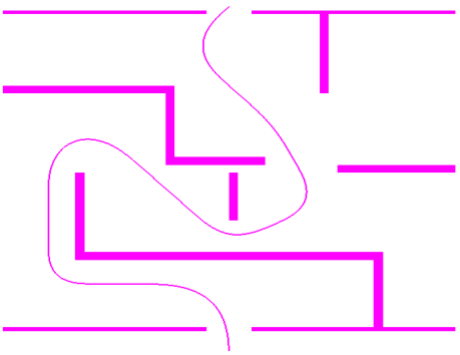


Figure 7 Navigation Simulation Case 2

Figure 9 show two instances of path planning simulation involving two agents. The final goal for one agent is market as g and the goal for the other agent is actually the first agent's configuration. Therefore, the two agents perform a chasing simulation. Each agent is a part of the environment. Figure 10 shows a 3D path planning simulation involving two aircrafts in a virtual environment.

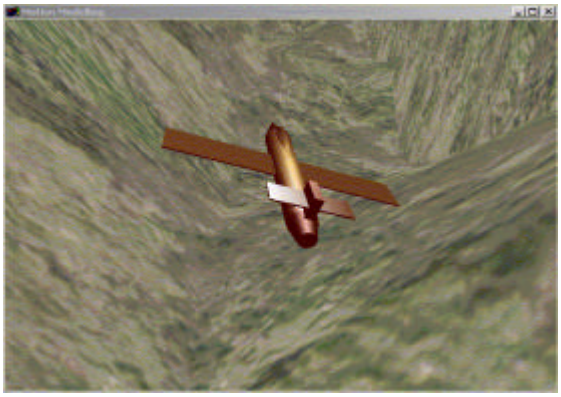
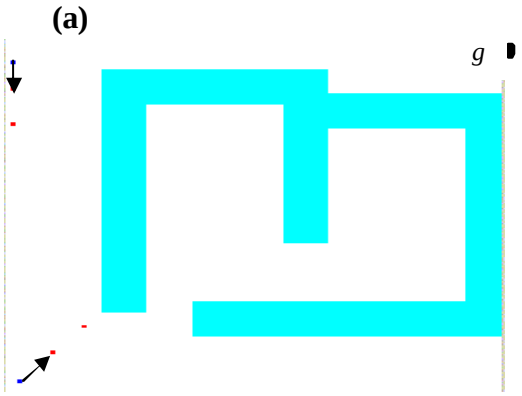


Figure.8 A aircraft simulation in a 3D environment



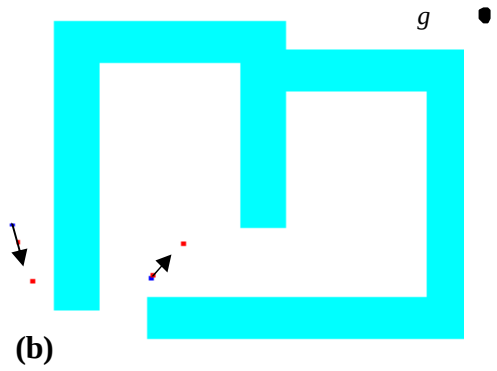


Figure.9 Two agents chasing in a 2D environment
g: final goal

Figure.10 Two aircraft chasing training simulation

The algorithm has been implemented in C++ and tested on a 1000 MHz Pentium processor PC with 256M memory and Matrox G450 graphics card (360MHz, 32M memory). The experiments of indoor scene simulation shown in Figure 7 were using a 500*500 units' space. The time cost of trajectory generation, the accurate trajectory length and the average end position error, using different space curves, are shown in Tablet 1.

9 CONCLUSIONS

A novel motion constraint path planning approach for real-time navigation of agents is proposed in this paper. The algorithm works well in dynamical and un-configured environments, and is able to produce a collision-free, time-optimal smooth motion trajectory. Multi-agents behaviour has been explored based on the algorithm. A simple physically-based

Tablet.1. Time cost of trajectory generation, trajectory length and average end position error of an indoor scene

	Trajectory Length (unit) 35000 steps	Total Time cost (ms)	Average end position error (unit)
Bezier Curve	1131.405	240	0.0
Clothoid curve	1049.296	280	0.0445
Cubic Spline	1033.071	280	0.0768

aircraft model has been developed, which is addressing the manoeuvring capabilities of the moving agents, while the moving agents' accelerations and velocities are always continuous and bounded. The generated motion path is constituted smoothly and has continuous curvature in the whole state space of the motion thus satisfying the major requirements for the implementation of such strategies in real-time navigation. The clothoid curve has been chosen as the basis for the motion trajectory generation. A 3D aircraft simulation has been conducted and the result is quite promising. The simulation result is quite satisfactory. The next step for our research is to refine the algorithm and look at path planning with more complex group behaviours in simulated environments.

11 REFERENCES

[Ari01] Okan Arikan and Stephen Chenney and {D. A.} Forsyth, "Efficient Multi-Agent Path Planning", Proceedings of the 2001 Eurographics Workshop on Animation and Simulation , Sep, 2001

[Bem96] A. Bemporad, A. De Luca, G. Oriolo, "Local incremental planning for a car-like robot navigating among obstacles" in Proc. of the 1996 IEEE Int. Conf. on Robotics and Automation, Minneapolis, USA, 1996

[Ber03] Tomas Berglund, Hakan Jonsson and Inge Soderkvist, "An Obstacle-Avoiding Minimum Variation B-Spline Problem," International Conference on Geometric Modelling and Graphic July 16-18, 2003.

[Dub57] L. E. Dubins, "On Curves of Minimal Length with a Constraint on Average Curvature and with Prescribed Initial and Terminal Position and Tangents," American Journal of mathematics, vol. 79, pp.497-516, 1957.

[Esv01] C. Esveld, "Modern Railway Track" , Second Edition, Published by MRT-Productions, a subsidiary of ECS, ISBN 90-800324-3-3, 2001.

[Fra01] Th. Fraichard and J. M. Ahuactzin, "Smooth Path Planning for Cars," IEEE Int. Conf. On Robotics and Automation May 21-26, 2001.

Hol92] P. D. Holmes and E.R.A. Jungert, "Symbolic and geometric connectivity graph methods for route planning in digitized maps", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol, 14, no.5, 1992, pp549-565.

[Jos97] F. M. Josson, "An optimal pathfinder for vehicles in real-world digital terrain maps", the Royal institute of Science, School of Engineering Physics, Stockholm, Sweden. MSc thesis, 1997.

[Kan86] Y. J. Kanayama and N. Miyake, "Trajectory Generation for mobile Robots," Robotic Research, vol. 3, Cambridge, MA: MIT Press, 1986, pp.333-340.

[Kan89] Kanayama, Y and Hartman, B. I, "Smooth local path planning for autonomous vehicles," Robotics and Automation, 1989, vol. 3, pp. 1265-1270.

[Lat91] Latombe, J.-C., "Robot Motion Planning", Kluwer Academic Publis hers, 1991, ISBN 0792391292.

[Lum90] V. J. Lumelsky, S. Mukhopadhyay, and K. Sun. Dynamic path planning in sensor-based terrain acquisition. IEEE Transactions on Robotics and Automation, Vol 6, No 4, 1990.

[Mon87] M. Montgomery et al., "Navigation algorithm for a nested hierarchical system of robot path planning among polyhedral obstacles", Proceedings IEEE International conference on Robotics and Automation, pp. 1616-1622, 1987.

[Nag01] Bryan Nagy and Alonzo Kelly, "Trajectory Generation for Car-Like Robots Using Cubic Curvature Polynomials," in Field and Service Robots 2001, Helsinki, Finland June 11, 2001.

[Oom87] B. J. Oommen, S. S. Iyengar, N. S. V. Rao, and R. L. Kashyap. Robot navigation in unknown terrain using learned visibility graphs. Part i:The disjoint convex obstacle case. IEEE Journal of Robotics and Automation, Vol RA-3 No.6 December, 1987.

[Pad00] D. Padmanabhan, "Optimal 2-D Path Planning", AME 598C Project Report, Spring 2000.

[Sch96] A. Scheuer and Th. Fraichard, "Planning Continuous-Curvature Paths for car-Like Vehicles," IEEE-RSJ Int. Conf. On Intelligent Robots and Systems, November 4-8, 1996. vol. 3, pp. 1304-1311.

[Ste95] Anthony Stentz and martial Hebert, "A Complete Navigation System for Goal Acquisition in Unknown Environment", In Autonomous Robots, Volume, Number 2, August 1995.

[Tat91] S. R. Tate. ``Arithmetic Circuit Complexity and Motion Planning", Ph. D. Dissertation, Duke University, 1991.

[Woo97] S. M. Woodcock (editor). "Artificial Intelligence in Games", 1997,

[Yah98] Alex Yahja, Anthony Stentz, Sanjiv Singh, and Barry L. Brumitt, "Framed-Quadtree Path Planning for Mobile Robots Operating in Sparse Environments", In Proceedings, IEEE Conference on Robotics and Automation, (ICRA), Leuven, Belgium, May 1998.

[Yam99] M. Yamamo to, M. Iwamura, and A. Mohri, "Quasic-Time-Optimal Motion Planning of Mobile Platforms in the Presence of Obstacles," Int. Conf. on Robotics and Automation, pp. 739-744, 1999.