Path Planning and Obstacle Avoidance in Unknown Dynamic Environments

Mithun Jacob
University of Pennsylvania
mithunj@seas.upenn.edu

***Abstract* – A laser-based navigation and obstacle avoidance algorithm for mobile robots has been developed. This method allows a mobile robot to navigate an environment with a large number of dynamic obstacles and static obstacles (convex as well as concave). Construction of an efficient testbed has been discussed as well.**

I. Introduction

This work considers the problem of motion planning for a mobile robot in a dynamic environment populated with static and dynamic obstacles where the objective is to reach a goal avoiding collision with all obstacles in accord with the robot's dynamics. There are several examples of such an environment: public places like malls, highways, and even in a domestic environment.

The problem of motion planning in dynamic environments is quite different from navigating static environments. Motion planning in static environments guarantees a solution if there exists one whereas motion planning in dynamic environments in intractable [1]. Hence, the objective of motion planning in dynamic environments is to avoid collisions minimizing motion time as much as possible.

A number of autonomous systems have been developed for handling dynamic environments such as household lawn mowing robots leading to robot lawn mower competitions [2]. Various approaches have been implemented to solve this problem, such as a practical implementation of an approach similar to the roadmap for 2 degree-of-freedom (DOF) manipulators using analytic representations of the configuration space obstacles [3].

Other approaches involve determining the solution by searching the position-time space as an extension of the configuration space [4]. In [5], determining the solution is a two-step process; first, a path avoiding all the static obstacles is determined and then the velocity profile along that is set to avoid dynamic obstacles. The theory of differential games has been used in planning for dynamic environments where solutions are computed in state space under dynamic constraints [6]. The usage of the velocity obstacle (VO) concept is also quite popular and [1] and involves representing dynamic obstacles by their VO.

This work describes the navigation and obstacle avoidance system for a mobile robot. This system is based on the data acquired from a laser scanner and assumes that all data is clean and the true position of the robot is known. As the robot navigates the environment (such as in Fig. 1), it continuously builds a map of discovered local obstacles by analyzing the laser range scanner data and differentiating between static and dynamic obstacles. Using this map, a global path to the goal from the current position is determined using the A* search algorithm and a potential field navigation function is used to stay on the global path while avoiding dynamic obstacles.

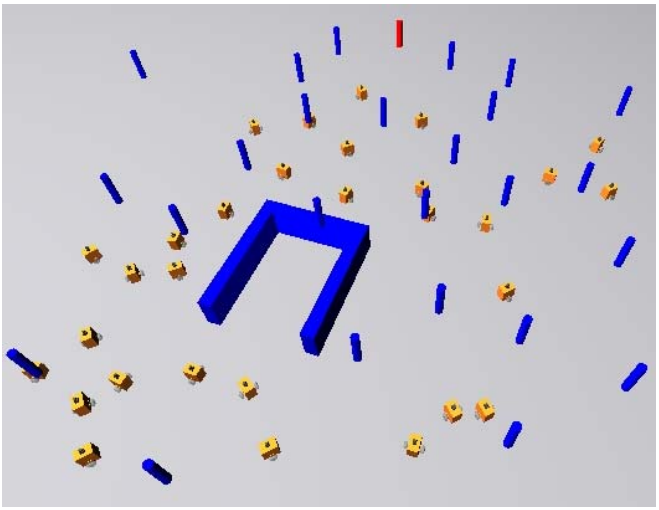

Fig. 1. Simulated Environment containing static and dynamic obstacles (10m x 10m with 30 robots and 1 static obstacle)

The remainder of the paper is structured as follows. Section 2 describes the construction of the simulated environment. Section 3 describes the technique used to interpret laser scanner data in order to distinguish between static and dynamic obstacles. Section 4 describes the method used to navigate the generated environment. Section 5 presents an example of this method at work and Section 6 discusses conclusions and future work.

II. Simulated Environment

The simulation testbed was designed to provide a large number of test environments of varying characteristics with minimal effort. Gazebo [7] was used to simulate the environment and test the developed navigation and obstacle avoidance algorithm.

A. Gazebo

Gazebo is a multi-robot simulator capable of simulating a population of robots, sensors and objects in a three-dimensional world. It generates both realistic sensor feedback and physically plausible interactions between objects using an accurate simulation of rigid-body physics.

B. Generating the Environment

Environments or worlds in Gazebo composed of a model hierarchy where models define simulated devices, and specifies how models are physically attached to each other. Gazebo uses an XML file called a 'world file' to create the environment. The world generator works by processing arguments related to the required environment and writing an XML file specifying model parameters such as the number of dynamic obstacles, size of the map, size and location of static obstacles and distribution of the obstacles. Handlers are automatically generated for all models in the required environment and passed to the world controller.

C. Environment and Implementation details

A feasible environment must have static obstacles as well as a large number of dynamic obstacles randomly moving through the environment avoiding the static obstacles and each other. This is achieved by positioning each dynamic obstacle at random positions throughout the map ensuring that the distance between each obstacle is above a certain threshold. Then, an equal number of goal positions are defined such that these goal positions are also spaced at a threshold distance from each other as well as the initial positions of the dynamic obstacles.

Once the initial positions and goals have been defined for each dynamic obstacle, the simulation begins. Each dynamic obstacle makes its way to the goal using the Local Planner (described in Section 4) avoiding all the obstacles successfully. Once it has reached its goal position, it waits until the system has found a goal position not used by any other dynamic obstacle. The same procedure is repeated for all dynamic obstacles, thus ensuring that the simulation uses an environment containing a large number of dynamic obstacles continuously and smoothly moving around the map with near uniform coverage.

Fig.1 shows a sample environment (10mx10m) containing 30 dynamic obstacles and a single U-shaped obstacle. The dynamic obstacles are Scarab models, which are differentially driven mobile robots. The blue cylinders (which float high above the generated world) represent goal positions for the dynamic obstacles.

The main robot (represented in red in Fig. 2) is also a Scarab model with an on-board model of the Hokuyo URG Laser range finder. Since the URG laser range finder does not offer 360° coverage, the blind spot on the rear side of the robot has been the major cause of collisions. All the controller algorithms have been written in MATLAB and C++ and communication with the Gazebo server is possible through GazeboAPI [8] which is an interface between MATLAB and Gazebo.

III. Interpreting Laser Scanner data

In this method of laser-based navigation of an unknown environment, robust discernment between static and dynamic obstacles is critical. If a sufficient number of dynamic obstacles around the robot have been erroneously determined to be static, then the global path planner will determine that it's impossible to reach the goal. And in certain scenarios, if a static obstacle is read as a dynamic obstacle, then the Local Planner (which is a potential field navigation function) will be unable to efficiently proceed to the goal. These points are discussed in detail in Section 4.

In brief, the laser scanner data is interpreted by taking 2 successive scans with a brief delay in between. The two scans are compared, and a score matrix (S) is updated. S_{ij} holds the probability of the point (j,i) being a static obstacle. Once, S_{ij} exceeds a certain threshold λ, the coordinate is considered as a static obstacle henceforth and all entries in S below a certain threshold (generally $\lambda/2$) are reset to 0. By resetting those values to 0, the possibility of mistakenly considering a coordinate to be occupied by a static obstacle since many dynamic obstacles have crossed it is greatly reduced.

A. LaserMap structure and the Difference matrix

In order to accurately store the probability of a point on the map being a static obstacle, the entire map is uniformly discretized and the LaserMap matrix is used to hold information about the state of the obstacles around the robot. The laser scanner range data obtained from the scanner is converted into the global coordinates of each obstacle using the laser range and angle data and the current position of the robot. These coordinates are rounded off to match the degree of discretization and using this information, the LaserMap matrix is updated.

In order to determine which obstacles are static between two laser scans, the following algorithm is used:

1. Store the detected obstacles in the first scan and set the obstacle points in the LaserMap matrix as α and the other points as 0. Store this as LM1
2. Delay
3. Repeat Step 1, but set obstacle points as 2α and store as LM2.
4. Determine the difference matrix D = LM2 – LM1

The difference matrix D can be interpreted as follows. If D_{ij} is equal to 0, this implies that the point is probably occupied by a static obstacle and its probability on the score matrix is increased. If $D_{ij} = \alpha-1$, then the point is the current position of the dynamic obstacle.

Using the method described above, an obstacle is determined to be static. Once this occurs, the D_{ij} for the points at which static obstacles have been confirmed to exist is set to infinity. This prevents a point from being analyzed again. Once the obstacles have been differentiated, the dynamic obstacles are passed to the local planner and the static obstacle map is updated for the global planner.

IV. Using the Local and Global Planner

A. Local Planner

Using the technique described in Section 2, the static and dynamic obstacles around the robot are determined. The updated static obstacle map is processed by the global planner (A*) and a global path is generated. The local planner continuously tries to reach each waypoint on the aforementioned path using the potential field navigation function defined below.

A. Local Planner

The Local Planner takes the current position of the robot, the goal (waypoint on global path) and the positions of all known dynamic obstacles within the laser scanner range. It then computes the potential at all 8 adjacent points around the robot using the following functions [9] to define the attractive and repulsive potential at a point.

The attractive potential at a point q is defined as follows:

$$U_{att}(q) = \frac{1}{2}\xi\rho_{goal}^2(q)$$

where ξ is a positive scaling factor and ρ_{goal} is the Euclidean distance between the goal and the current position of the robot. The repulsive potential at a point q is defined as follows:

$$U_{rep}(q) = \sum_{k=1}^{N-1} U_{CB_k}(q)$$

where

$$U_{CB_k}(q) = \begin{cases} 0 & if\ \ \rho_k(q) \le \rho_0 \\ \frac{1}{2}\eta\left(\frac{1}{\rho_k(q)} - \frac{1}{\rho_0}\right)^2 & if\ \ \rho_k(q) \le \rho_0 \end{cases}$$

where ρ_k denotes the Euclidean distance between the k^{th} dynamic obstacle in the area and η is a positive scaling factor. The degree-of-influence ρ_0 determines the extent of the force field about an obstacle.

The point around the robot with minimum potential q_{min} (the sum of the attractive and repulsive potential) is chosen and using feedback linearization, the angular and linear velocity of the robot is set.

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = q_{\min} - q_s$$

$$\begin{bmatrix} v \\ w \end{bmatrix} = \begin{bmatrix} \cos\theta & -L\sin\theta \\ \sin\theta & L\cos\theta \end{bmatrix}^{-1} \times \begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix}$$

where L is an offset length proportional to the wheel base dimension of the robot (0.15m for the Scarab models), θ is the roll Euler angle of the robot and q_s is the current position of the robot.

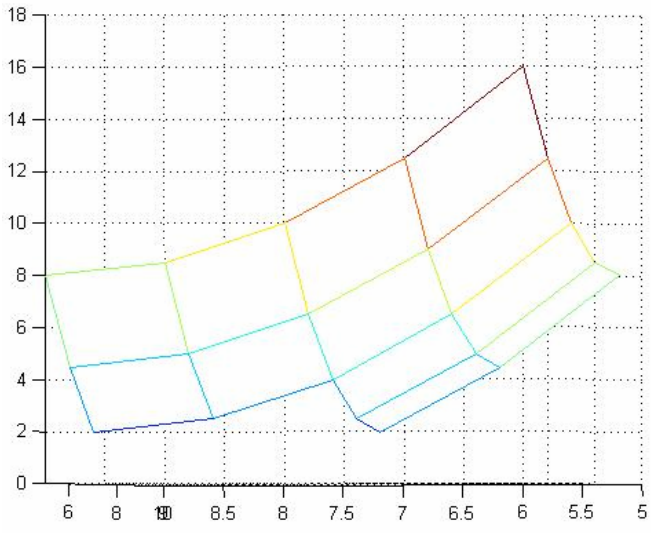

Fig. 2. Potential Field mesh around the robot

In the simulated environment, the dynamic obstacles navigate using the Local Planner alone. It computes the next best step by accounting for all the other N-1 dynamic obstacles in the map. By continuously setting the velocity vector at a high frequency, smooth motion was achieved without any collision. In a worst-case scenario, a dynamic obstacle would be rendered immobile if it lands in local minima.

B. Global Planner

This scenario is not devastating to the simulated environment since dynamic obstacles are not allowed to enter the U-shaped obstacle (as shown in Fig. 1) and since the local minima is created by the dynamic obstacles, there is a high chance it will dissipate. But the main robot navigating the environment will be stuck in the concave obstacle without any chance of recovery.

This scenario is taken care of by the global planner (A*). Once an obstacle has been determined to be static by the technique described in Section 3, the static obstacle map is updated, and a new global path to the goal is immediately generated by the global planner. Then, the Local Planner takes over and tries to stay on the global path generated by the global planner by setting the waypoints on the global path as its local goals.

The only information the global planner has is the current position of the robot, the goal position and the known map of static obstacles. It computes the shortest path after discretizing the whole environment and returns the global path in terms of coordinates. When the robot first starts to navigate the environment, the global planner returns a straight-line path to the goal. But once it starts to map the environment and detects static obstacles in its path, the global planner updates the global path.

An interesting point to be noted about the Laser Scanner Interpreter is how it perceives dynamic obstacles staying too long in a certain positions (oscillating about a certain point, perhaps) as a static obstacle. This can work to our benefit it

GetNext(qs,qg,dyn_obs)

Returns the velocity vector towards the point with the least minimum potential using the Local Planner using feedback linearization.

AStar(qs,qg,stat_obs)

Returns a global path to the goal bypassing all static obstacles after uniformly discretizing the map .

LaserScannerInterpreter()

Updates the static obstacle map (stat_obs), and returns a vector of points occupied by dynamic obstacles (dyn_obs). This is achieved by taking successive scans with a small delay in between and comparing the state of the obstacles around the robot in order to differentiate between static and dynamic obstacles.

Main()

```
qs=initial position of the robot
qg=goal position of the robot
stat_obs=NULL

WHILE norm(qs-qg) <= 0.05m //MainLoop

    path=Astar(qs,qg,stat_obs)

        FOR every waypoint in the global path

            qs =current position of robot
            qgl=current waypoint

            WHILE norm(qs-qgl) <= 0.02

                Update dyn_obs and stat_obs with
                LaserScannerInterpreter

                IF new static obstacles are found
                   Break to MainLoop

                Set the velocity vector of the robot
                using GetNext()

             END

        END

END
```

Fig.3. Algorithm

relieves the load from the Local Planner, since the global planner returns a path which skirts around the dynamic obstacle.

Fig.3 shows the algorithm used to navigate the unknown dynamic environment.

V. Examples

Using the world generator described in Section 2, several worlds were created and tested. This section shows an environment of size 8m x 8m with 20 robots. The red robot navigates the environment, and the yellow robots serve as dynamic obstacles. The floating green and red cylinders denote the start and goal positions of the red robot respectively and as mentioned before, the floating blue cylinders denote the goals of the dynamic obstacles.

In the following figures, windows marked as Figure 2 display the red robot as a red spot and blue points as the dynamic obstacles around it. Figure 1 shows the current static obstacle map and Figure 10 shows the global path as computed by the global planner. The gui-cam window shows the Gazebo simulation.

Fig. 4 shows the robot at the mouth of the concave obstacle with an incomplete map of the static obstacle. The global planner still returns a straight line path to the goal and has also designated a dynamic obstacle as static since it is oscillating about a certain point.

Fig 5 shows the updated static obstacle map as well as the new global path circumventing the U-shaped obstacle. If the local planner alone was used, the robot would've been stuck in the local minima.

Fig. 6 shows a sequence of images displaying the dynamic obstacle avoidance algorithm at work. As observed in the first row of images, the red robot veers to the right as it detected the circled robot. This action brings it in the proximity of the robot circled in the second and third row of images. The rest of the motion can be observed the remaining rows of images.

V. Conclusions and Future Work

In this paper, we have discussed a system architecture which integrates mapping with motion planning. The laser scanner data is interpreted to distinguish between static and dynamic obstacles and this data is used by the global and local planners to navigate the robot through the unknown dynamic environment.

The future work involves correcting some of the common causes of failure. Some of them are:

1. Sometimes the degree of influence proves to be too large and the robot refuses to navigate through an area with a large number of dynamic obstacles. Although this is advantageous at times, it is extremely time-consuming as the robot waits for an opening until it can slip around the robots. Dynamically decreasing the degree-of-influence based on the number of dynamic obstacles around the robot at that given moment in time is a possibility which will be explored.

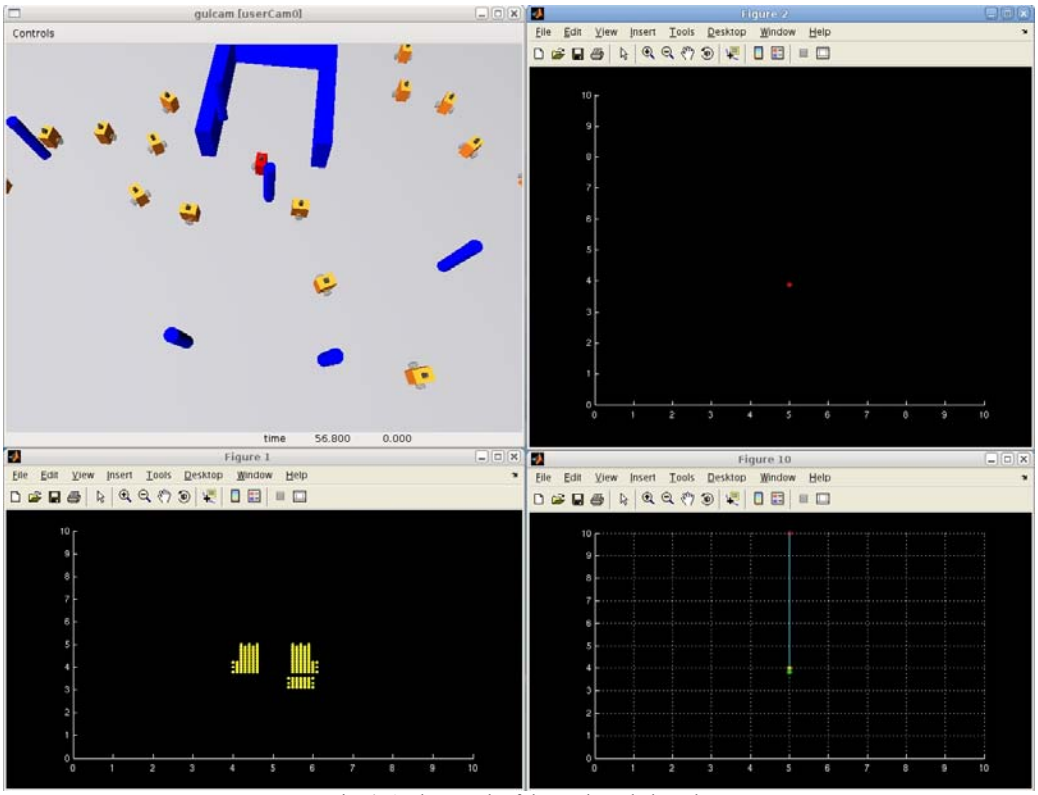

Fig. 4. At the mouth of the U-shaped obstacle

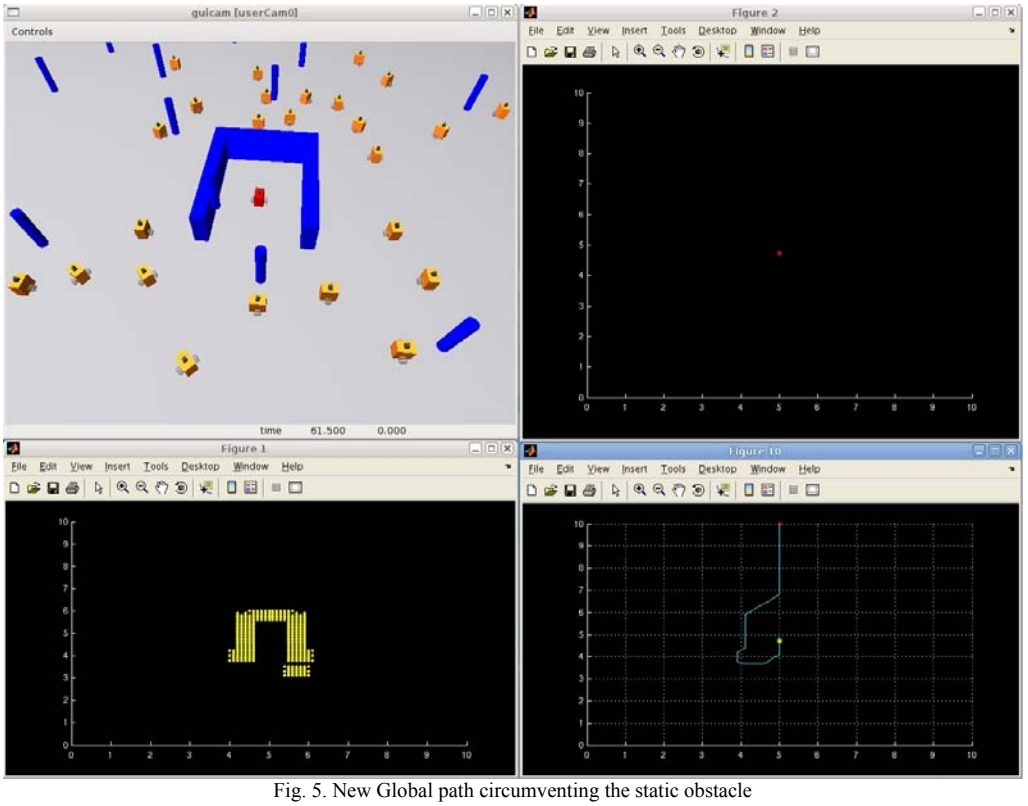

Fig. 5. New Global path circumventing the static obstacle

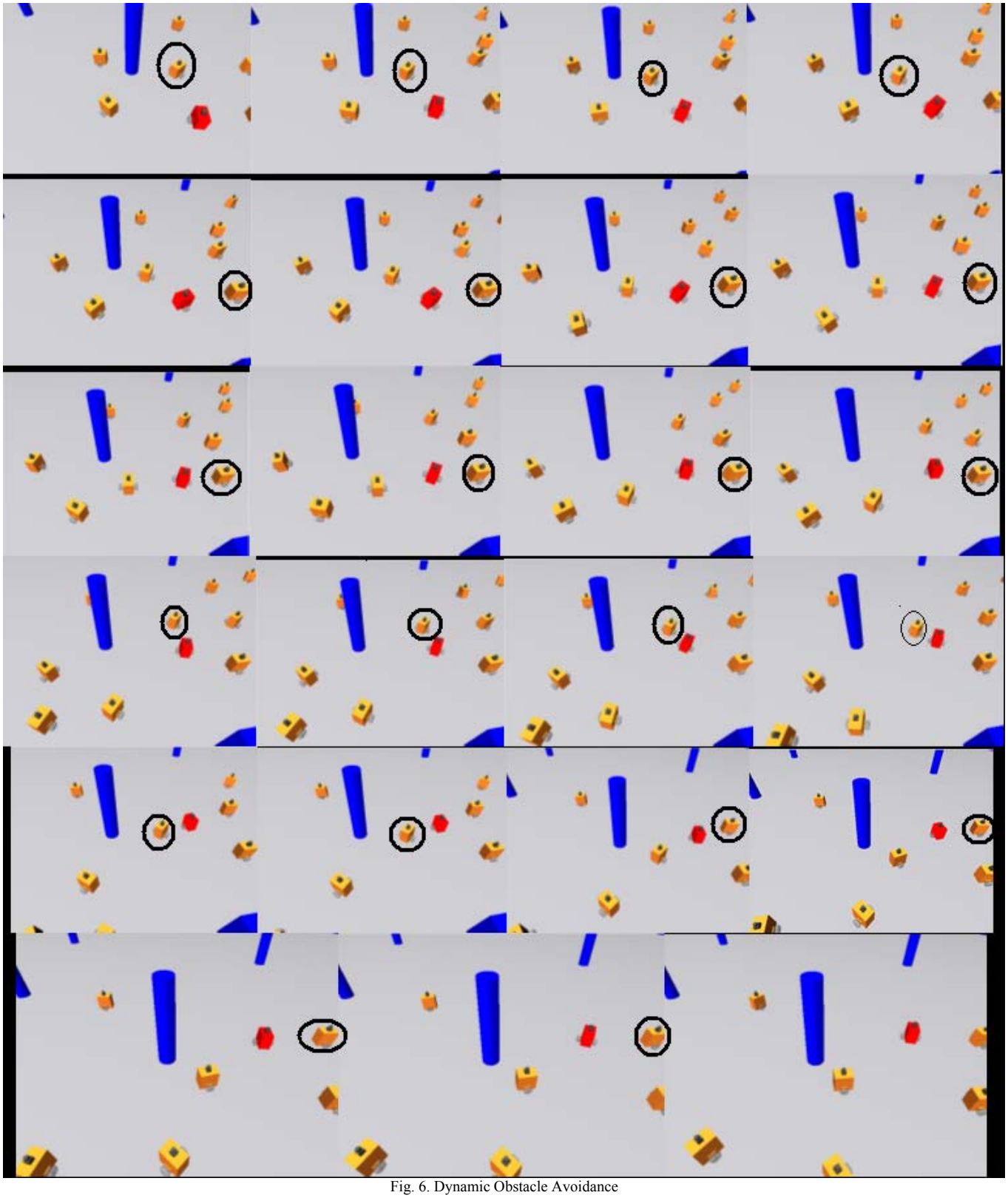

Fig. 6. Dynamic Obstacle Avoidance

2. Due to the blind spot of the laser scanner, some rear-end collisions occur. This can be rectified by introducing a second laser scanner.
3. A major drawback to using A* as a global planner is the lack of scaling when the number of static obstacles is increased. This is easily solved by using another global planner like D* Lite [10].

Other future work includes utilizing the world generator to test a large number of environments and supplement the results with an empirical guarantee as well as enabling the robot to handle environments with a higher density of dynamic obstacles.

Acknowledgement

I would like to thank Dr. Maxim Likhachev for his advice and support of this work. Special thanks go to Dr. Vijay Kumar, Dr. Peng Cheng, Jon Fink and the rest of the class who all contributed to providing valuable feedback about this work.

References

[1] P. Fiorini and Z. Shiller, "Heuristic Motion Planning in Dynamic Environments Using Velocity Obstacles", IEEE Robotic & Automation, San Diego, California, USA, 8-May-1994.
[2] ION. ION Autonomous Lawnmower Competition.Internet: www.automow.com, 2006.
[3] Z. Shiller and R.Y. Gwo, "Collision-free path planning of articulated manipulators", ASME Journal of Mechanical Design, December 1993.
[4] M, Erdnrann and T. Lozano-Perez, "On multiple moving objects", IEEE International Conference on Robotics and Automation, pages 1419-1424, San Francisco,CA, April 7-10 1986.
[5] K. Kant and S.W. Zucker, "Towards efficient trajectory planning: the path-velocity", The International Journal of Robotics Research, 5(3):72- 89, Fall 1986.
[6] D.W. Johnson and E.G. Gilbert, "Minimum time robot path planning in the presence of obstacles", 24th Conf. on Decision and Control, Lauderdale, FL, December 1985.
[7] Gazebo 3D Simulator. Internet: http://playerstage.sourceforge.net, 2008.
[8] GazeboAPI-Nathan Michael. Internet: www.seas.upenn.edu/~nmichael, 2008
[9] Jean-Claude Latombe, "Robot Motion Planning", Kluwer Academic Publishers, Boston, MA, 1991.
[10] S. Koenig, and M. Likhachev, "Improved fast replanning for robot navigation in unknown terrain.", Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), 2002.