A study on Artificial Potential Fields

Kadir Firat Uyanik
KOVAN Research Lab.
Dept. of Computer Eng.
Middle East Technical Univ.
Ankara, Turkey
kadir@ceng.metu.edu.tr

Abstract— **Having dealt with a very basic local sensing based path finding algorithm -tangent bug - in the first homework, we will focus on again a very simple reactive path planning method, *artificial potential fields(APF)*. APF is a reactive approach since the trajectories are not planned explicitly but obtained while executing actions by differentiating a function what is called *potential function*. A potential function is differentiable real-valued function $U : \mathbb{R}^m \rightarrow \mathbb{R}$. The output of a potential function can be taken as the energy and it's negative gradient as the net-force acting on the robot. In this report, two different potential functions are presented; attractive-repulsive potential function, and navigation function. These two functions are tested in a simulation environment with various configurations. Experiments are done on the robocup small-size league robot soccer platform that I have been started developing with the first homework. At the end, we see that both methods have some limitations and innate problems.**

I. INTRODUCTION

The main idea behind artificial potential field methods is finding a function that represents the energy of the system, and generating a force on the robot so that the energy of the system is minimized and reach it's minimum value, preferably only, at the goal position.

Therefore, moving to the goal position can be thought as moving robot from a *high-value* state to a low-value state by following a "downhill" path. Mathematically speaking, this is the the process of following negative gradient, or *gradient-descent*, i.e. $\dot{c}(t) = -\nabla U(c(t))$. The motion terminates as soon as the gradient vanishes, that is $\nabla U(q^*) = 0$ where q^* is called the critical value of U. This critical point is either maximum, minimum or saddle point as it shown in figure 1.Although the maximum is not that critical as soon as robot doesn't start movement from this point, but the local minimum points are the real concern while designing a potential function. Here, one assumption is that the saddle points -being unstable points- do not cause any problem due to the robot's inertia and other dynamics.

In this document, two main approaches, additive attractive/repulsive potential, and navigation potential functions are to be investigated. The rest of the article organized as follows: first, a brief mathematical background of potential functions are given, then the experimental framework is represented. Lastly, experimental results, pros and cons of the two methods are given in the discussion section, and it ends with conclusion.

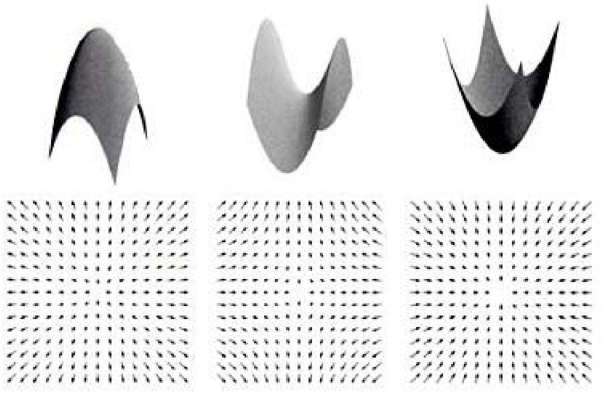
Fig. 1. From left to right: maximum, saddle, and minimum critical points are shown where figures at the top are the functions and the below are the gradient vectors of them.

II. POTENTIAL FUNCTIONS

Two different potential functions are examined in this report, additive attractive/repulsive potential functions and navigation potential functions.

A. Additive Attractive/Repulsive Potential Functions

The intuition behind the additive potential functions is attracting the robot to the goal position while repelling the robot from the obstacles by superposing these two effects into one resultant force applied on the robot. This potential function can be constructed as $U(q) = U_{att}(q) + U_{rep}(q)$, that is the sum of the attractive and repulsive potentials.

1) The Attractive Potential: The essential requirement in construction of the attractive potential is that U_{att} should be monotonically increasing with the current distance $d(q, q_{goal})$ of the robot to the goal, q_{goal}. Two common U_{att} functions are the ones having conical and/or quadratic forms. In the experiments, combinations of the two are used since the conical functions suffer from the discontinuity in the goal position, that is *division-by-zero problem* in computational sense.

$$U_{att}(q) = \begin{cases} \frac{1}{2}\xi d^2, & d \leqslant d^{\star}_{goal} \\ \xi d^{\star}_{goal} d - \frac{1}{2}(d^{\star}_{goal})^2, & d > d^{\star}_{goal} \end{cases} \quad (1)$$

where $d = d(q, q_{qoal})$, ξ is the attraction gain, and $d^{\star}_{goal}$ is the threshold after which quadratic function changes to the conical form.

The gradient of the function (1) is obtained as;

$$\nabla U_{att}(q) = \begin{cases} \xi(q - q_{goal}), & d(q, q_{goal}) \leqslant d^{\star}_{goal} \\ \xi d^{\star}_{goal} \frac{(q - q_{goal})}{d(q, q_{goal})}, & d(q, q_{goal}) > d^{\star}_{goal} \end{cases} \quad (2)$$

2) The Repulsive Potential: The behavior of a proper repulsive potential function should look like the behavior of a tightened spring, or the magnets getting closer to each other so that obstacle avoidance can be satisfied. In this study following formula is utilized,

$$U_{rep}(q) = \begin{cases} \frac{1}{2}\eta(\frac{1}{D(q)} - \frac{1}{Q^{\star}})^2, & D(q) \leqslant Q^{\star} \\ 0, & D(q) > Q^{\star} \end{cases} \quad (3)$$

Derivative of the repulsive potential function(3) is the repulsive force which is,

$$\nabla U_{rep}(q) = \begin{cases} \eta(\frac{1}{Q^{\star}} - \frac{1}{D(q)})\frac{1}{D^2(q)}\nabla D(q), & D(q) \leqslant Q^{\star} \\ 0, & D(q) > Q^{\star} \end{cases} \quad (4)$$

where $Q^{\star}$ is the distance threshold for an obstacle to create a repulsion effect on the robot, and η is the repulsion gain.

The gain parameters (i.e. η and ξ) and the threshold parameters(i.e. $Q^{\star}$ and $d^{\star}$) are set empirically.

The total repulsive potential field can be obtained by summing up the potentials caused by all of the obstacles,

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

and the movement is realized by following the negative gradient of the sum of attractive/repulsive potentials or simply the following vectorial sum;

$$-\nabla U(q) = -\nabla U_{att}(q) - \nabla U_{rep}(q)$$

In this study, the translation velocity assignment is simply done by utilizing $-\nabla U(q)$.

B. Navigation Potential Functions

Due to the local minima problem in the additive attractive/repulsive potential fields, many other local-minima free potential field methods are proposed such as wave-front planner [6][7]. However these methods have discretization problems and that's why they become computationally intractable for high dimensional and large configuration spaces. To solve the local minima problem, a special function to be constructed which has the only minimum at the goal position. These functions are called navigation functions, formally defined in [8][9].

A function ϕ:$Q_{free} \rightarrow [0, 1]$ is called a *navigation function* if it

- is smooth (or at least C^k for $k \geqslant 2$)
- has a unique minimum at q_{goal} in the connected component of the free space that contains q_{goal}.
- is uniformly maximal on the boundary of the free space, and
- is *Morse*[1]

[1]Detailed discussion can be found in [1].

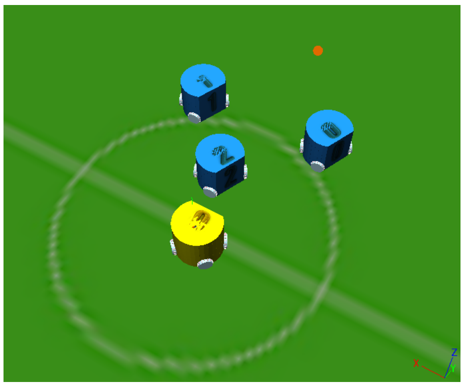

Fig. 3. A screenshot from the Webots simulator, while robot is tested in a three-obstacle between robot and the goal case.

As in the case of attractive/repulsive potential functions, navigation functions are used to move the robot from an arbitrary configuration to the goal configuration in the negative gradient direction of the potential at that instant. The navigation potential functions can be constructed as follows:

$$\phi = \frac{(d(q, q_{goal}))^2}{(\lambda\beta(q) + d(q, q_{goal})^{2\kappa})^{\frac{1}{\kappa}}} \quad (5)$$

where $\beta(q)$ is a repulsive-like function calculated as

$$\beta_i(q) = (d(q, q_i))^2 - {r_i}^2$$

for each obstacle having a radius of r_i. The critical point in choosing $B_i(q)$ is that $QO_i = q \mid \beta_i(q) \leqslant 0$; this means $\beta_i(q)$ is negative inside an obstacle and it is positive outside in the free space, which requires

$$\beta_0(q) = (d(q, q_0))^2 - {r_0}^2$$

where q_0 is the center and r_0 is the radius of the sphere world.

In this report, it is assumed that the configuration space is bounded by a sphere centered at q_0 and the other obstacles are also spherical centered at q_i.

The λ in the equation (5) is necessary for bounding the navigation function so that it has 0 at the goal and 1 on the boundary of any obstacle.

The κ on the other hand, has an effect of making the navigation function has the form of a bowl near to the goal. Increasing κ causes the other critical points(local minimum etc.) shift toward the obstacles and resulting in a negligible repulsive behavior compared to the overwhelming influence of the attractive field. The effect of increasing κ is shown in the figure2, for the case of having three obstacle in front of the robot and a goal position beyond the obstacles as it is illustrated in the figure 3.

Once a "proper" κ value is decided, robot's configuration or -in this study- velocity is updated with the negative gradient of the navigation potential(equation (6)) function similar to the what's been done in the attractive/repulsive potential functions.

$$-\nabla\phi = -\left(\frac{2d\nabla d[(\lambda\beta(q)+d^{2\kappa})^{\frac{2}{\kappa}})] - \frac{1}{\kappa}(\lambda\beta q + d^{2\kappa})^{\frac{1}{\kappa}-1}(\lambda\nabla\beta(q) + 2\kappa d^{2\kappa-1}\nabla d)}{(\lambda\beta(q)+d^{2\kappa})^{\frac{2}{\kappa}}}\right) \tag{6}$$

where $d = d(q, q_{goal})$ and $\nabla\beta(q) = \sum_{i=0}^{N}\left(\nabla\beta_i(q)\prod_{j=0,j\neq i}^{N}\beta_i(q)\right)$. $\nabla\beta_i(q) = 2(q-q_i)$ for $i > 0$, $\nabla\beta_0(q) = -2(q-q_0)$.

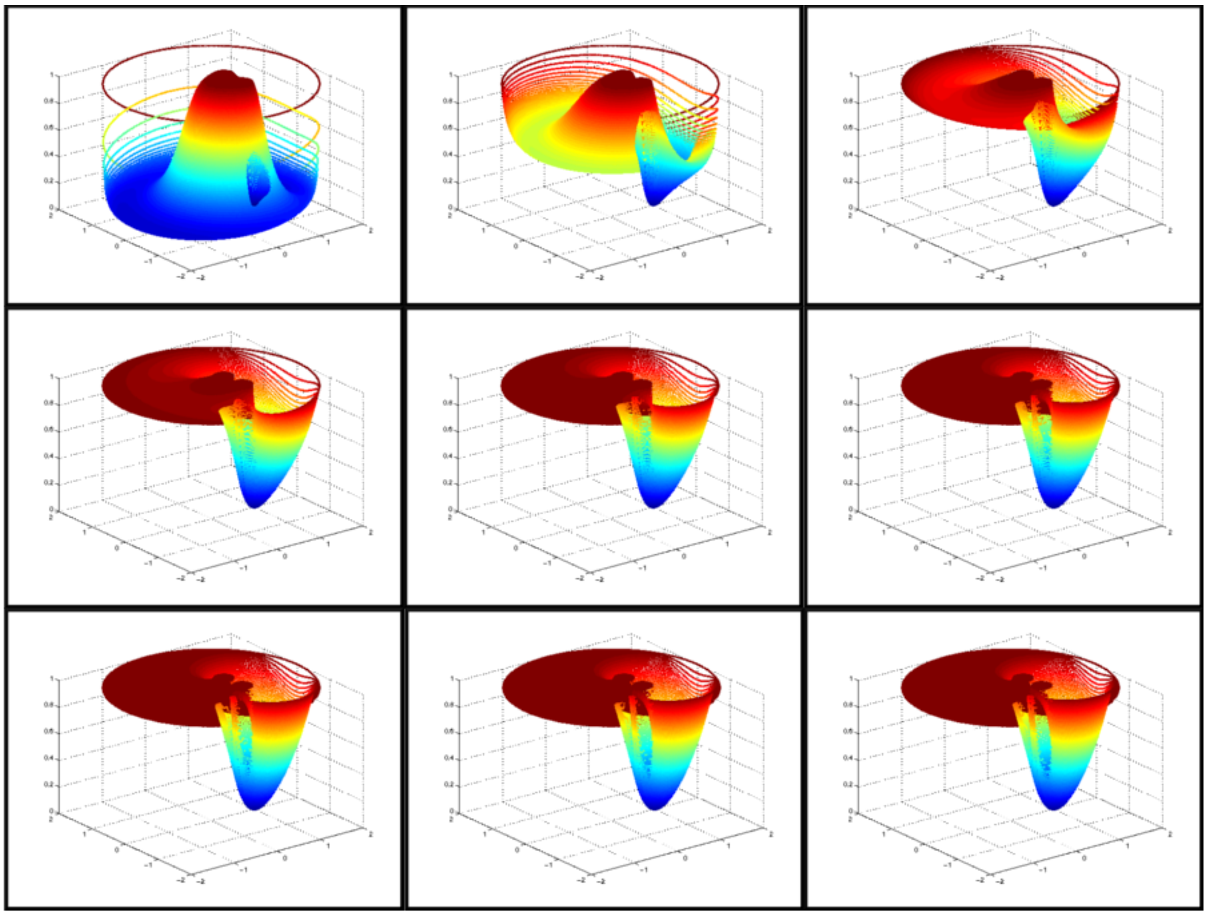

Fig. 2. From left to right and top to bottom, κ is changed between 1 and 9, and navigation potential function is calculated for the sphere-world by sampling the world for the specific configurations(viz. 2D positions) varying in polar coordinates $r = [0.01, 2.0]$ and $\theta = [0.01, 2\pi]$ with the discrete sampling steps of 0.01. As it is clearly shown that κ doesn't effect the navigation potential considerably after a certain value, which is $\kappa = 5$ in this case.

III. Experimental Framework

In this study, I have used *Webots*[5] robotics simulator by adding ROS (Robot Operating System,[4]) support to the framework that I have been developing for RoboCup Small Size League robot soccer competition. The robot design is done in VRML (Virtual reality modeling language) which has four asymmetrically distributed omniwheels enabling robot to translate and rotate at the same time while following a linear path.

Below is the kinematic equation for the robot that is used navigate robot as if it is a point robot but having a specific radius:

$$\begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{bmatrix} = \begin{bmatrix} sin(\alpha) & -cos(\alpha) & -r \\ sin(\beta) & cos(\beta) & -r \\ -sin(\beta) & cos(\beta) & -r \\ -sin(\alpha) & -cos(\alpha) & -r \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ \omega \end{bmatrix}$$

Robots wheels are specially designed so that they do not cause so much trouble while moving sideways. Each wheel has 15 rollers and this is pretty much enough to realize smooth movements.

This framework is intended to be as modular as possible so that newly developed path planning algorithms can be inserted or replaced with the existing ones with almost no effort whatsoever.

As it is shown in figure 4, there are multiple processes that are dedicated to different tasks such as providing global information, deciding on the goal positions, planning paths for a given current-goal configuration pair, and transmitting action commands to the robots. This modular architecture helps us to replace simulated environment with the real robotic system without changing the algorithmic content of the system.

Below are the descriptions of the nodes in the ROS network.

- **ssl_sim_vision**, Small-size-league(SSL) simulated vi-

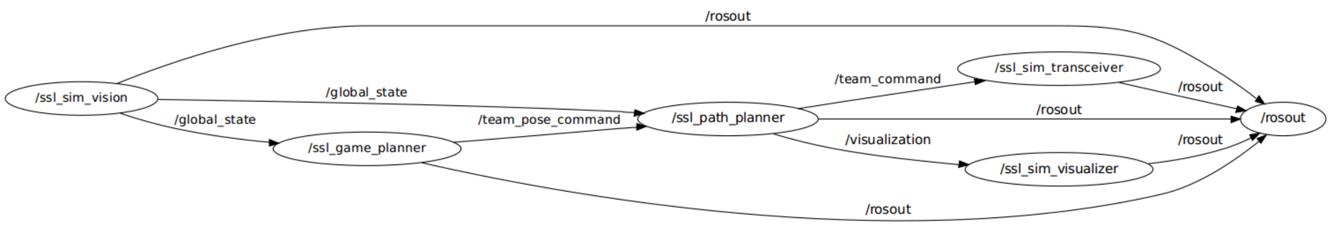

Fig. 4. This graph is auto-generated by ROS *rxgraph* utility. It shows how the messages are transmitted through which node(=process)

sion module. It publishes global pose of the robots that are inside and just-outside of the field with proper state flags (inside-field, outside-field, out-of-fov etc.) to the global_state topic. It is also a supervisor node in the webots environment.

- **ssl_game_planner**, SSL game planner. Currently it publishes predefined target poses for all robots, but it is planned to be a hybrid/deliberative robot controller in which there might be several modules -nodes- later on. It subscribes to the global_state topic and publishes the target poses of the robots to the team_pose_command topic.
- **ssl_path_planner**, SSL path planner. In this study, this node runs attractive/repulsive potential fields or navigation potential function based path planners. It subscribes to the team_pose_command and global_state topics, and publishes to the visualization and team_command topics.
- **ssl_sim_transceiver**, SSL simulated transceiver. It is a bridge between the robots and offboard computing units. Subscribes to the team_command topic and *emits* messages to the simulated robots through webots' communication interface.
- **ssl_sim_visualizer**, SSL simulator visualizer. It helps to visualize the path taken by the robots and the commands given to them.

IV. DISCUSSION

The two potential functions mentioned in this report, namely additive attractive/repulsive potential function and navigation potential function are tested in three different scenarios, one-obstacle direct-free path, three-obstacle indirect-free path, two-obstacle direct-free path. Please see the section VI for the links to the videos representing each experiment case.

A. Case: One-obstacle direct-free path

In this experiment, we would like to see if the robot can traverse around a single obstacle to reach the goal position by following a preferably linear path. Normally, since the obtacle is not entirely in the path of the robot, we would expect that the robot almost-directly goes to the target position. But it is shown that without doing a visibility check basic additive potential fields may cause perturbances in the path or even oscillations depending on how the attraction and repulsive gains are set. Navigation functions, on the other hand, may give a better path if κ is properly chosen. Since there is not a theoretical way to select best κ, selection process may require serious intuition and effort.

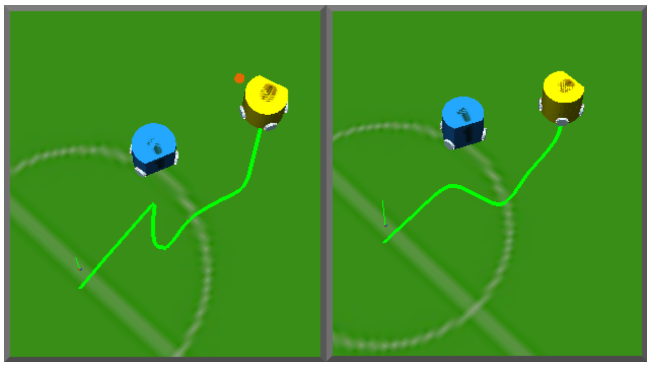

Fig. 5. Left: additive att/rep potential function, right: navigation potential function.

B. Case: Three-obstacle indirect-free path

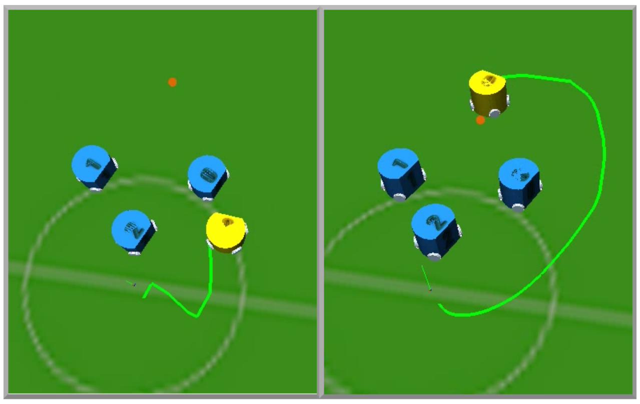

Fig. 6. Left: additive att/rep potential function, right: navigation potential function.

As it is shown in figure 8, basic att/rep potential function method suffers from the local minima problem and robot gets stuck at some point. On the other hand, navigation function finds a path to the goal, though far from being optimal. Figure 7 shows how the path length varies with different κ values.

C. Case: Two-obstacle direct-free path

Although this case is similar to first case, it shows the severity of the additive att/rep potential fields which even

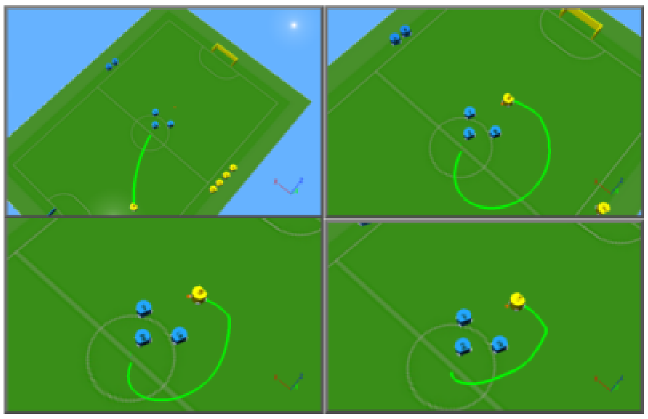

Fig. 7. From top left to the bottom right κ values are 3,4,5 and 7. The time it takes to reach to the goal is t_{κ_3} =no value, $t_{\kappa_4} = 62s$, $t_{\kappa_5} = 53s$ and $t_{\kappa_7} = 45s$ in simulation time. That's why κ is chosen to be 7 in the experiments. Please notice that the viewpoints of these figures are different than each other, careful inspection will reveal the fact that as κ values decrease, the length of the path increases since the shape of the bowl=like potential function becomes more spreaded-out.

may not result a complete solution, although the only thing that the robot should do is going just forward.

Navigation function, on the other hand, could find a path between the obstacles, though the movement was a little oscillatory. One reason of the oscillation is the delay between the communication nodes(=processes) since the PC that I've used for the experiments was a standard laptop.

The overall results for the three experiment cases are given in the table I. Although additive potential functions are lack of completeness, in most of the cases, they provide shorter paths than the paths navigation functions return.

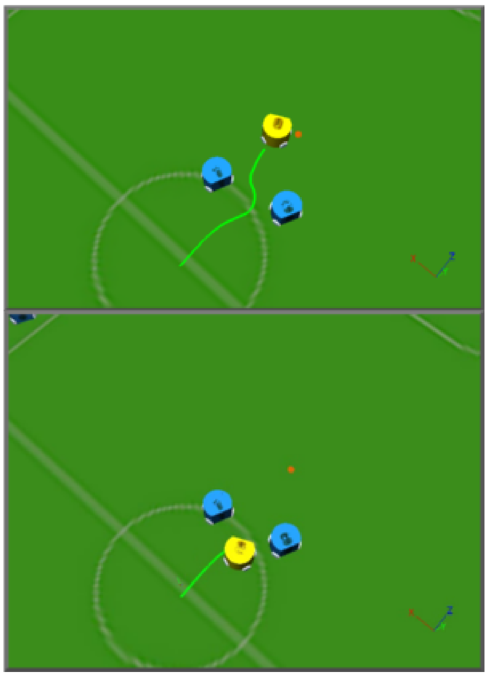

Fig. 8. Top: additive att/rep potential function, bottom: navigation potential function.

TABLE I

OVERALL RESULTS

	Additive		Navigation	
Experiment	time(sec.)	path(m)	time(sec.)	path(m)
Case 1	2.5	1.25	2.5	1.26
Case 2	∞	∞	45.52	1.97
Case 3	∞	∞	1.5	1.09

To deal with local minimum problem in additive functions one idea would be applying random forces on the robot once it is detected that the robot stops without reaching to the goal position (viz. getting stuck at a local minimum). Similar method is proposed in [3], called *random walk*.

Another solution to the local minima problem would be using repulsive forces in a different form. For instance, if the robot is imagined as a fluid or gas particle, at least in the sphere world, robot will be able to pass through obstacles standing between the robot and the goal position. However, this method will not be successful in the case of confronting narrow passages, causing collision to the obstacles. Another problem this method will cause is the oscillatory movements in locally cluttered environments in which there might be paths that are longer in the sense of euclidean distance but shorter in time since robot can reach higher speeds in obstacle-free paths.

One solution to the oscillatory movement problem would be grouping the obstacles together and having virtual obstacles which may contain narrow passages, and the obstacles that are sufficiently close to each other as it is shown in figure 9 and 10. This method with the visibility testing is proposed in [2], and I have successfully implemented this method(given in the algorithm 1), and tested in the simulation environment in one of my earlier studies. Although tests are done in highly cluttered environments, generated paths were close to the optimal paths and no local minima problem is observed.

Algorithm 1. Obstacle Grouping

```
while preObstList.size() > 0 do
  closestObst ⇐ getClosestObst()
  if isVisible(closestObst)=TRUE then
    tempObst ⇐ closestObst
    repeat
      if isLinkable(tempObst,preObstList[i])=TRUE then
        linkObsts(tempObst, preObstList[i])
        tempObst ⇐ tempObst→Neighbor
      end if
    until preObstList.size()= 0
  end if
  virtualObst ⇐ closestObst
  groupObsts(virtualObst)
  postList.push_back(virtualObst)
end while
```

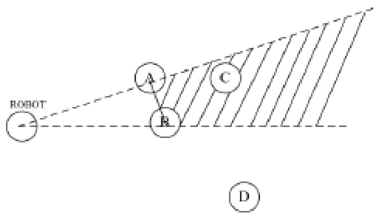

Fig. 9. Closest obstacle and its neighbor obstacle are linked to each other and nwe virtual obstacle makes obstacle C invisible to the robot (adapted from[**?**])

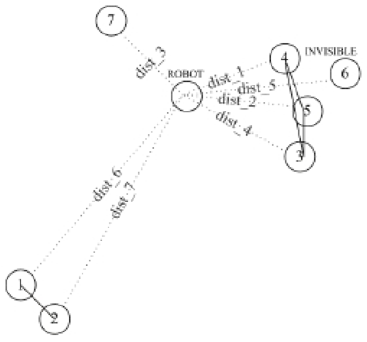

Fig. 10. Obstacle 3, 4 and 5 are linked to each other.They form a virtual obstacle (adapted from [2])

V. CONCLUSIONS AND FUTURE WORK

The main focus of this study was to compare simple additive attractive/repulsive potential function with the navigation potential functions(for various κ values). The results show that it is difficult to find a proper κ value for navigation potential functions, and simple additive attractive/repulsive potential functions suffer from the problem of getting stuck in local minima. In order to use these methods in a real-life application, like a robocup small size league robot soccer domain, it is better to utilize other functionalities such as visibility and path-free checks so that irrelevant obstacles can be discarded while calculating the potentials. It is shown that grouping the obstacles might result in shorter paths considering the time it takes to the goal position and it provides a better representation of the environment rather than handling every obstacle seperately.

VI. ACKNOWLEDGEMENT

All the source code; ssl_sim_vision, ssl_game_planner, ssl_path_planner, ssl_sim_visualization, ssl_sim_transceiver can be found in this repository.

This document could also be downloaded in pdf format,here

Here is the list of videos:

- Case: One-obstacle direct-free path
  - Additive attractive/repulsive potential function `http://www.youtube.com/watch?v=ra-4UXbUaz0`.
  - Navigation potential function `http://www.youtube.com/watch?v=QWG0X2xIJ-Y`
- Case: Three-obstacle indirect-free path
  - Additive attractive/repulsive potential function `http://www.youtube.com/watch?v=XreCjhkEJA8`.
  - Navigation potential function `http://www.youtube.com/watch?v=AaW1E8bh9kg`
- Case: Two-obstacle direct-free path
  - Additive attractive/repulsive potential function `http://www.youtube.com/watch?v=hGoSAFj7SL4`.
  - Navigation potential function `http://www.youtube.com/watch?v=k8oT6fFCZaM`
- Obstacle grouping algorithm `http://www.youtube.com/watch?v=0rp-VhkjYd0`

REFERENCES

[1] Principles of Robot Motion by Howie Choset etal., MIT Press, 2005, ISBN: 0-262-03327-5

[2] B. Zhang,W.Chen, M. Fei, An optimized method for path planning based on artificial potential field, Sixth International Conference on Intelligent Systems Design and Applications (ISDA'06) Volume 3 (2006)

[3] J. Barraquand and J.-C. Latombe. Robot motion planning: A distributed representation approach. Internat. J. Robot. Res., 10(6):628649, 1991

[4] Quigley M., Conley K., Gerkey B., Faust J., Foote T. B., Leibs J., Wheeler R., and Ng A. Y. (2009). Ros: an open-source robot operating system. n International Conference on Robotics and Automation, ser. Open-Source Software workshop.

[5] Michel, O. Webots: Professional Mobile Robot Simulation, International Journal of Advanced Robotic Systems, Vol. 1, Num. 1, pages 39-42, 2004

[6] R. Jarvis. Coliision free trajectory planning using distance transforms.*Mech. Eng. Trans. of the IE Aus*, 1985.

[7] J. Barraquand, B. Langlois, and J.C.Latombe. Numerical potential field techniques for robot planning.*IEEE Transactions on Man and Cybernetics*, 22(2):224-241, Mar/Apr 1992.

[8] Koditschek, Daniel E., and Rimon, Elon: Robot navigation functions on manifolds with boundary, Advances in Applied Mathematics 11(4), volume 11, 412442, 1990.

[9] Rimon, Elon and Koditschek, Daniele; Exact robot navigation using artificial potential functions IEEE Transactions on Robotics and Automation. Vol. 8, no. 5, pp. 501-518. Oct. 1992