MODEN: Obstacle-driven Elastic Network for Line-of-Sight Communication*

Francisco S. Melo and Manuela Veloso

Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213, USA
{fmelo,veloso}@cs.cmu.edu

Abstract. In this paper, we address the problem of connecting two distant communication nodes by deploying a number of mobile robots that act as gateways between the towers. To address this problem, we propose ODEN, a path-planning algorithm that relies on an *elastic network* to produce an obstacle-free path between the two towers. This algorithm builds over a previous elastic-network-based path-planning algorithm and overcomes some of the major limitations of this algorithm. We also propose an extension of ODEN to address the more complex situation in which multiple pairs of towers must be connected in a common environment. We illustrate the results obtained with both ODEN and its multi-path counterpart, MODEN, in several test-scenarios.

1 Introduction

In this paper, we consider the problem of connecting two distant communication nodes (henceforth referred as "towers") placed in such a way that no communication is possible between them. To overcome this difficulty, a number of mobile robots can be deployed in the environment. These robots possess the ability to communicate with the towers (when in range) and with each other. In the most general setting, we assume that the robots have only local knowledge about the configuration of the environment (*e.g.*, about the existence and position of obstacles) obtained by means of any local on-board sensors. The purpose of the robots is to navigate the environment until a configuration is reached in which the two towers are connected, as illustrated in Fig. 1(a).

In this paper, we model this problem as a *path planning* problem between the positions of the two towers and use an "elastic network" to model the movement of the gateway robots deployed in the environment. Our algorithm is an extension of a previous method proposed in [1], in which the path is represented as an elastic network initialized as a straight-line between the two target positions. The mobile

* This work was partially supported by the Information and Communications Technologies Institute, the Portuguese Fundação para a Ciência e a Tecnologia, under the Carnegie Mellon-Portugal Program. A preliminary version of this work was previously presented at 7th Portuguese Robotics Festival and Scientific Meeting, 2007.

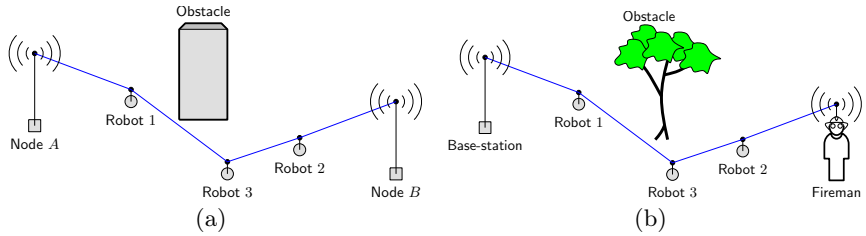


Fig. 1. Illustration of the typical scenario considered in the paper and a motivating example.

nodes in the network (the mobile robots in our scenario) are "attracted" toward free-space and "repelled" away from the obstacles, yielding an obstacle-free path.

The contributions of this paper are two-fold. On one hand, we improve on the original algorithm from [1] by proposing a new update rule that yields smoother and more efficient deployment of the robots, in a sense soon to be made clear. This leads to the ODEN algorithm, our first contribution. On the other hand, we propose an extension of ODEN to address the more complex situation in which multiple pairs of towers must be connected in a common environment. In this situation, and to minimize interference, it is desirable that the deployment of the robots is conducted in such a way as to minimize intersection between the different communication paths. This leads to MODEN, the "multi-path" version of ODEN and the second contribution of this paper.

To motivate this work, consider, for example, a forest fire situation in which firemen must maintain contact with a base-station. In order to do so, they can carry small, portable mobile units that they drop as they move toward their assigned position. These units will ensure connectivity between each fireman and the base-station by locally adjusting their position so as to improve connectivity and minimize interference (see Fig. 1(b). Aimed at such applications, the approach in this paper explicitly considers the existence of obstacles in the environment, driving this work away from other works addressing open space deployment and connectivity [2]. Furthermore, we are particularly interested in modeling scenarios in which the robots move in an unstructured (outdoor) environment. In these scenarios, signal obstruction can arise from obstacles such as groups of trees that can actually be *traversed* by the robot, but cause severe decrease in the ability of the robot to communicate with its neighbors. Also, in the setting considered herein, we are not concerned with radio-based *navigation* or *localization*, a topic of recent current research [3–7]. As will be apparent from our algorithm, the robots need only *minimal navigation capabilities*. In particular, the algorithm will not require them to use any localization or path planning algorithm, but simply execute very simple movement primitives (such as moving straight in a given direction). Finally, the problem considered herein is, in a sense, closely related to the problem addressed in [8]. The latter work proposes an algorithmic procedure to determines *when* the robots can move in order not to loose the overall connectivity of the network. In this paper we propose an algorithmic procedure to determine *how* the robots can move in order to improve the overall connectivity of the network.

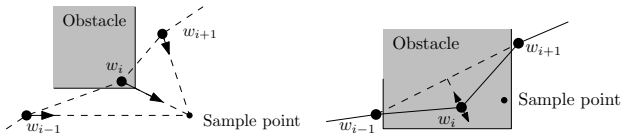


Fig. 2. Several updates of the closest PU, depending on the sampled point.

2 Path Planning Using an Elastic Network

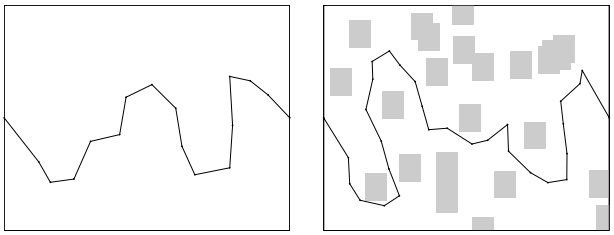
The algorithm described in [1] makes use of an elastic network of "processing units" (PU) linking the two target towers throughout the environment (henceforth abusively referred as the *configuration space* for the network). The network is initialized as a straight line between the two towers and we denote by $\mathcal{U}^t$ the set of all units in the network at iteration t. The algorithm proceeds at each iteration t by sampling a random point within a distance of r around the network, *i.e.*, a point in the "perceptual range" of some robot in the network. It then updates the position of the nearest node in the network (the best matching unit, BMU) according to the following heuristic:

- If the sample point is in free space, it "attracts" the BMU toward it (see Fig. 2(a));
- If the sample point is in occupied space, it "repels" the BMU away from it;
- If both the sample point and the BMU lie in occupied space, the BMU moves orthogonally to the line segment whose extreme points are BMU's neighboring units (Fig. 2(b)).

Once the BMU is updated, its immediate neighbors are updated accordingly. The points used in the updates are randomly sampled around the nodes in the network, in a region with radius r that is successively decreased to a final minimum value of r_F. The algorithm starts by "pulling the network" toward large regions of free space and then locally adjusts the network to the "details" of the environment.

It is worth noting that, in terms of the robots in the network, these "abstract" operations can be implemented in a remarkably simple way. At each iteration, one of the robots in the network uses its sensor information to move toward a randomly chosen position in free-space. This will typically cause the signal strength between the robot and its neighboring robots to change. The neighboring robots then adjust their positions so as to compensate for the change in the signal strength. We also note that, in practice, this operation can be performed by multiple robots simultaneously, as long as none of the robots are contiguous or share a neighbor.

Finally, every λ time instants, a new unit is added in between the two most distant units in the network until a maximum number of units is reached. This condition could easily be replaced by similar condition of adding a new unit (a new robot) whenever the distance between two nodes in configuration space was above a pre-specified threshold. However, the use of a temporal condition allows the initial updates to be more efficient.



(a) No obstacles. (b) 25 obstacles.

Fig. 3. Trajectories obtained by the original algorithm.

The term *elastic network* arises from the fact that each unit exerts some conservative tension on its neighboring units so as to maintain their relative positions. In other words, the updated PU is "pulled" by its neighbors and also "pulls" its neighboring units. More details on the algorithm can be found in the referred work [1]. This "elastic tension" between neighboring units can be seen as a local adjustment that each of the units performs whenever one of its neighbors moves in order to improve connectivity.

The method has several interesting properties. It is extremely simple to implement. It only requires a sampling mechanism around the network and the evaluation of the *attraction function*. This attraction function merely determines whether a point is in free-space or in occupied space and the update of a unit reduces to a few extremely simple operations that translate into very simple motion commands. Also, given its simplicity, it is a surprisingly efficient algorithm, being in fact able to find obstacle-free paths between the desired points with very little computational effort. The examples reported in [1] were able to determine obstacle-free paths in several complex environments in few thousand iterations. Furthermore, it does not require any search or any model of the environment, but just a sampling function able to determine whether a point lies in free-space or not. Finally, the method is *local* in that each unit is updated considering only its immediate neighborhood. This is a very interesting feature of the algorithm for the particular set of problems we are interested in. In our setting, each unit is actually an independent robot that must adjust its individual position communicating only with its neighboring units and sampling the surrounding space, but the method still provides a way of determining a *global* obstacle-free network.

However, it also presents several inconveniences. First of all, the algorithm will often spend a lot of iterations updating units that need not be updated. This happens, for example, in an obstacle-free environment or any environment where there is a lot of free space, where most of the units will already start away from any obstacle. Furthermore, because sample points in free-space keep attracting the network to free-space, this may lead to peculiar trajectories as depicted in Fig. 3. Finally, the algorithm was designed to run for a fixed (usually large) number of iterations and takes generally this number of iterations to terminate before a path is produced. In the next section we propose a modification of the algorithm that, while maintaining the simplicity of the original algorithm and its functional principle, alleviates the reported inconveniences.

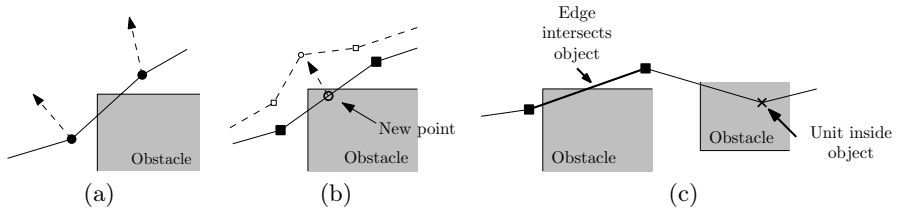


Fig. 4. Situations addressed by the new update mechanism. (a) Situation where further updating is desirable. (b) The extra point solves the problem. (c) Two different classes of "updatable" units.

3 The ODEN Algorithm

In this section we describe two fundamental changes to the algorithm in the previous section: a new update mechanism for the units and an additional condition to introduce new units in the network. These modifications yield the ODEN algorithm,[1] and readily overcome the inconveniences pointed out in the previous section. We also present the results obtained with ODEN that show the introduced modifications to actually improve the overall efficiency and performance over the original algorithm.

3.1 The Update Mechanism

Recall that the two main inconveniences reported in the previous section are the many iterations spent updating units that need not be updated and the fact that the *whole network* is attracted toward free-space, this leading to undesirable trajectories. This undesirable behavior is mainly due to continuous updating of units that are already in free space. To overcome such phenomenon we introduce a simple modification to the algorithm: we partition the set $\mathcal{U}^t$ into two sets, $\mathcal{U}_U^t$ and U_F^t. The set $\mathcal{U}_U^t$ contains the "updatable" units and the set $\mathcal{U}_F^t$ contains the "fixed" units. In a first approach, we consider the set U_F^t as the subset of $\mathcal{U}^t$ lying in free-space. Therefore, we update only the units that lie inside objects.

There is a purpose in continuously updating the different PUs even when they are in free space. Consider the situation depicted in Fig. 4(a). In this situation there is an obstacle between the two depicted units that may hamper the corresponding communication channel. In the original algorithm, the two units depicted would be continuously updated, eventually pulling their common edge out of the obstacle. If we consider the set $\mathcal{U}_F^t$ as containing all units in free space, none of the two units depicted will ever be updated again, and the path thus obtained will not be obstacle-free, even if all units may eventually lie in free space. To overcome this situation, we consider a second class of updatable units: those whose contiguous edges intersect any of the obstacles. In Fig. 4(c) we illustrate the two classes of updatable units.

One final remark to refer that, in order to determine if a given edge intersects any object, we sample several random points along the edge and test whether they lie in free space. The number of points depends on the actual length of the edge. This method generally produces quite reliable results even without

[1] Obstacle-Driven Elastic Network.

sampling too many points. In the case of actual robots, the same information can generally be determined from the link between the two robots.

3.2 Introducing New Units

Recall that in the original algorithm a new unit is added to the network every λ iterations up to a maximum pre-determined number of units. The addition of such extra units, as well as a decrease in the sampling radius around the network, allows the network to converge to smoother trajectories and forces the updates to consider increasingly *local* data around each unit (see [1] for complete details).

Considering the modified version of the algorithm described insofar, it is possible and indeed likely that all units reach free-space before λ iterations have occurred. This implies that the algorithm will "stall" until a new unit is introduced. As such, we include an additional condition in the algorithm, and a new unit is added to the network if any of two conditions is verified:

- Whenever λ iterations have occurred; or
- Whenever $\mathcal{U}_U^t$ is empty.

If the algorithm introduces a new unit in the network before λ iterations have occurred, the sampling radius is, nevertheless, decreased accordingly.

We note that the introduction of new units also alleviates the problem described in Fig. 4(a). In fact, in many situations such as the one in Fig. 4(a), a new point will actually be added between the two points in Fig. 4(a), leading to a solution like the one on Fig. 4(b).

3.3 The ODEN algorithm

We refer to the modified version of the algorithm as ODEN, standing for obstacle-driven elastic network. ODEN is summarized in Algorithm 1, where α is a randomly chosen number such that $-\beta \leq \alpha \leq \beta$ and F is the attraction function.[2]. The parameters $t_{\max}$, λ, β, η_0 and η_1 are common to the algorithm in [1]. The first two parameters represent, respectively, the maximum number of iterations for the algorithm and the number of iterations between two insertions of a new point. The last three parameters define the "update rates" and "elastic coefficient" used in the updates of the various components of the algorithm. The input parameters x_I and x_F represent the positions of the two towers and N_0 and $N_{\max}$ represent the initial and final number of nodes in the network.

3.4 Experimental Results with ODEN

We tested ODEN in several random environments with different degrees of complexity. In all results displayed, the algorithm was run with the same parameters as those reported in [1].[3]

Figure 5 presents the results obtained in the same environments as those in Fig. 3, illustrating how ODEN is able to overcome the inconveniences of

[2] The attraction function is just an indicator function for an obstacle, taking the value of 1 if x is in the free-space and -1 otherwise.

[3] In particular, we use $N_0 = 10$, $N_{\max} = 100$, $\beta = 0.0025$, $\eta_0 = 0.05$, $\eta_1 = 0.01$, $r_I = 2$, $r_F = 0.7$ and $t_{\max} = 40,000$.

Algorithm 1 The ODEN algorithm.

Require: $x_I, x_F, N_0, N_{\max}, r_I, r_F$;

Set $w_0 = x_I$ and

$$w_i = x_I + i \cdot \frac{x_F - x_I}{N_0 - 1}, \quad i = 1, \ldots, N_0 - 1.$$

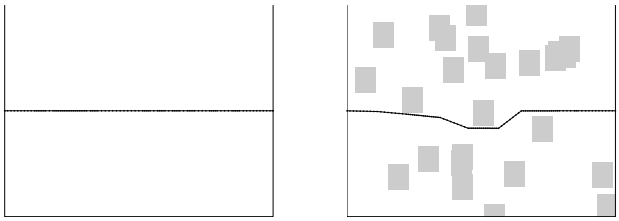
Set $t = 1$, $L_{ins} = 0$, $N = N_0$;
Compute $\mathcal{U}_U^t$;
if $\mathcal{U}_U^t = \emptyset$ **then**
  Set $t = L_{ins} + \lambda$ and goto 17;
end if
Randomly choose $w_i \in \mathcal{U}_U^t$;
Set $r = r_I \cdot \left(\frac{r_F}{r_I}\right)^{t/t_{\max}}$;
Randomly choose $x \in X$ such that $\|x - w_i\| < r$;
Set $w_j = \arg\min_{w \in \mathcal{U}_U^t} \|x - w\|$;
if $F(x) > 0$ **then**
  $w_j \leftarrow w_j + \eta_0(x - w_j) + \beta(w_{j-1} + w_{j+1} - 2w_j)$;
  $w_{j\pm1} \leftarrow w_{j\pm1} + \eta_1(x - w_{j\pm1})$;
else
  $w_j \leftarrow w_j + \alpha \frac{(w_{j+1} + w_{j-1})^\perp}{\|w_{j+1} + w_{j-1}\|}$;
end if
if $t - L_{ins} \geq \lambda$ **then**
  $L_{ins} = t$;
  Insert w_{new} such that $w_{new} = \frac{w_k + w_{k+1}}{2}$, where

$$\|w_k - w_{k+1}\| = \max_{w_i \in \mathcal{U}_U^t} \|w_i - w_{i+1}\|.$$

  $N = N + 1$;
  Update $\mathcal{U}_U^t$;
end if
if $t < t_{\max}$ and $(I \neq \emptyset \vee N < N_{\max})$ **then**
  Set $t = t + 1$ and goto 7;
else
  Exit;
end if

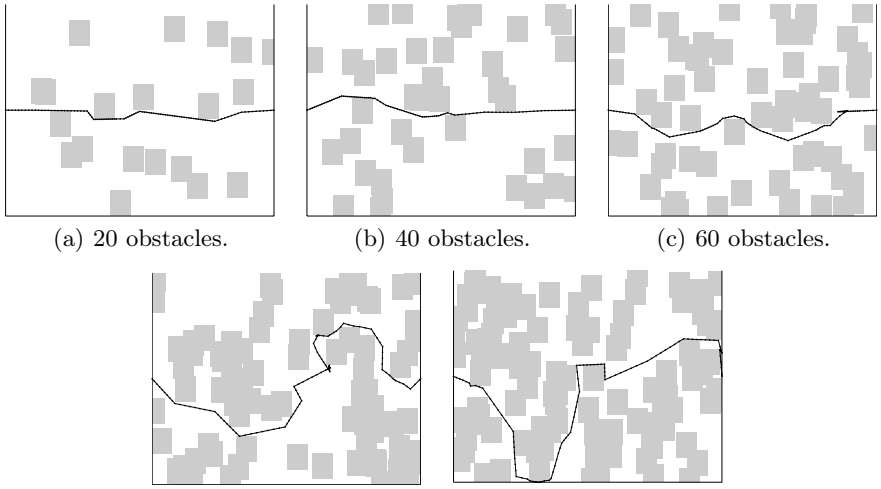
the original algorithm reported in the previous section. Figure 6 presents the results obtained in several random environments ranging from 20 to 100 random obstacles. It is worth noting that even if the algorithm does not take into account the minimization of the length of the path, the obtained paths often exhibit small perturbations from the initial straight path. Note also that the intersection effect in Fig. 4(a) is not observed in the trajectories, even if in many situations the path is close to these.

Finally, Fig. 7 presents the results obtained in environments with non-random obstacles. The distributions of the obstacles in these environments can be seen



(a) No obstacles. (b) 25 obstacles.

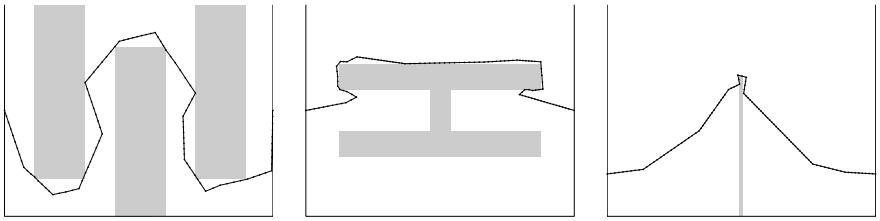
Fig. 5. Trajectories obtained with ODEN in the environments of Fig. 3, corresponding to 100% and approximately 77% of free-space.



(a) 20 obstacles. (b) 40 obstacles. (c) 60 obstacles.

(d) 80 obstacles. (e) 100 obstacles.

Fig. 6. Sample network outlines obtained with ODEN in five random environments with diverse number of obstacles, respectively corresponding to approximately 82%, 64%, 46%, 28% and 10% of free-space.



(a) (b) (c)

Fig. 7. Trajectory obtained with ODEN in environments with non-random obstacles.

as possible worst-case situations for the algorithm. Notice that the algorithm is, in fact, able to deliver the expected collision-free paths.

We note that, in the first set of environments, the algorithm is generally able to deliver a solution *and stop* within few iterations. This is in clear contrast with the original method, which would run for a predefined number of iterations. Also, ODEN generally performs better on more sparse environments than in en-

vironments with little free space. This is easily explained if we consider that the algorithm proceeds by sampling the space around the network and pulling the network toward free-space and away from the obstacles. If few sample points lie in free space, more iterations will be required for the algorithm to find a proper path. Finally, "thin" obstacles may be hard to sample and, therefore, hard to avoid. We produce one further experiment to test the performance of the algorithm when a single thin, long obstacle is found in the environment. The result is presented in Fig. 7(c). Even though the environment has a single obstacle and is, in general, a sparse environment, the algorithm took a considerable number of iterations (≈ 720) before a solution was found.

4 MODEN: ODEN in Multi-path Domains

We now address the more complex problem of connecting n pairs of towers in a common environment. Given the positions in configuration space for all towers in the set, we are interested in determining *individual paths* joining the two towers in each of the n pairs, so that the paths are non-intersecting. The requirement for non-intersecting paths is mainly to avoid undesirable interference between the corresponding communication channels. We henceforth refer to the nodes between the pair of towers i as the *network* EN_i and refer to the positions of the corresponding towers as the "initial" and "final" points of EN_i (with no particular convention as to which is the initial and which is the final).

Given the initial and final points in configuration space for each of the n networks, we can apply the ODEN algorithm to determine the path describing each network individually, yielding n distinct paths, $EN_1, \ldots, EN_n$. However, this approach does not take into consideration the existence of the other networks in the same environment and will probably lead to intersecting paths. A simple idea to overcome such difficulty is to consider each PU in the path of other networks as the center of some obstacle when running ODEN for each network EN_i.[4] The size of the obstacle can be adjusted as a parameter. This strategy turns the other networks's path into "obstacles" and ODEN should thus produce individual trajectories that do not intersect, if possible.

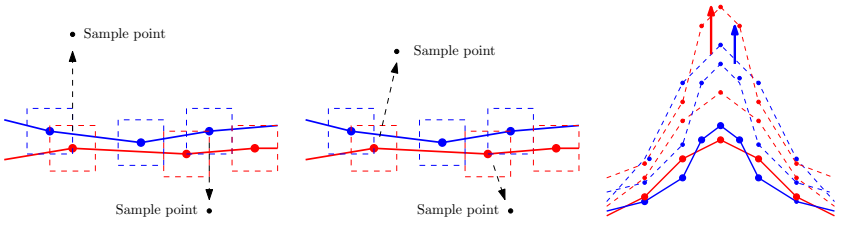


Fig. 8. Situations that may lead to intersection. (a) Different networks sample on opposite sides. (b) Same network samples on opposite sides of neighboring network. (c) Two networks keep "chasing" each other.

[4] The consideration of the other networks as an obstacle can also be implemented using the connectivity information between the robots in the networks.

Algorithm 2 The MODEN algorithm.

Require: $\{x_I^1, \ldots, x_I^n\}, \{x_F^1, \ldots, x_F^n\}, N_0, N_{\max}, r_I, r_F$;

1: Initialize X_{obs} with the obstacles from the environment;

$$w_0 = x_I;$$
$$w_i = x_I + i \cdot \frac{x_F - x_I}{N_0 - 1}, \quad i = 1, \ldots, N_0 - 1.$$

2: **for all** i=1,...,n **do**
3: $P_i = ODEN(x_I^i, x_F^i, N_0, N_{\max}, r_I, r_F)$;
4: Append P_i to X_{obs} (the extreme units in the network are not considered);
5: **end for**

However, if ODEN updates all trajectories simultaneously, this will lead to intersecting paths. Consider the situation depicted in Fig. 8(a). If the samples used to update each path continue to be sampled in opposite sides of the other path, the final paths will unavoidably intersect. Furthermore, situations may occur in which the paths keep "chasing" each other, as they keep sampling free points in one direction while trying to avoid the other path (Fig. 8(c)).

To minimize such phenomenon, we introduce an ordering among the networks and run the ODEN algorithm sequentially according to that order. According to this ordering, it is possible to refer to each of the networks in the set as $EN_1, \ldots, EN_n$, where EN_i stands for the ith network in the given ordering. We now successively apply the ODEN algorithm to each of the networks $EN_1, \ldots, EN_n$ individually: we determine the path for network EN_1 using the ODEN algorithm and considering only the original obstacles in the environment. Then, for each other robot EN_i, $i = 2, \ldots, n$, we determine the corresponding path using the same ODEN algorithm, but considering as obstacles the paths of the networks $EN_1, \ldots, EN_{i-1}$ besides the natural obstacles in the environment. Finally, to alleviate the intersections arising from cross-sampling (Fig. 8(b)), we include a small adaptive bias in the random sampling process.

4.1 The MODEN algorithm

MODEN is summarized in Algorithm 2, where $\{x_I^1, \ldots, x_I^n\}$ and $\{x_F^1, \ldots, x_F^n\}$ denote the sets of initial and final points for the n networks. P_i is the path generated for EN_i. As for the bias in sampling, recall that ODEN uses points sampled in a region of radius r around the network to update the units in the network. A sample point x is given by

$$x = w_i + r_x \angle \theta_x,$$

where w_i is a unit in the network, r_x is a random number between 0 and r, and θ_x is a random angle between 0 and 2π. Let $I_1, \ldots, I_k$ be a uniform partition of the interval $I = [0, 2\pi]$. A uniform sampling procedure chooses an angle θ_x in I_i with probability $p_i = 1/k$. Suppose that the chosen angle at some iteration was α_x and the corresponding point x is in free space. Then, the probabilities p_i are

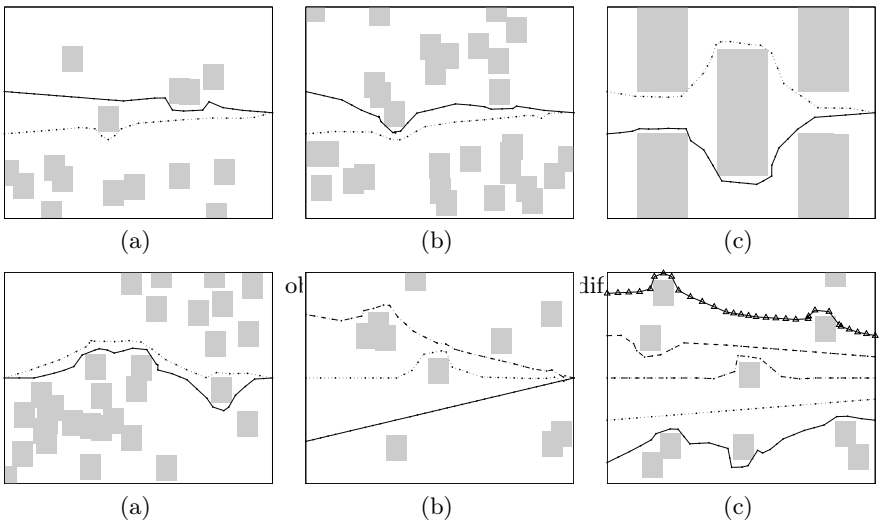


Fig. 10. Sample trajectories obtained for problems with coincident endpoints and with multiple networks (3 and 5).

biased toward the direction α_x according to

$$p_i = \begin{cases} p_i + \epsilon(1 - p_i) & \text{if } \alpha_x \in I_i; \\ (1 - \epsilon)p_i & \text{otherwise} \end{cases}$$

and then normalized to yield $\sum_i p_i = 1$.

4.2 Results using the multi-path algorithm

We now present the results of MODEN in different scenarios. In Fig. 9 we present the trajectories obtained in environments with 15 and 30 random obstacles and 5 non-random environments, where the networks joint different initial positions to a common final position. We then applied the algorithm to the extreme situation where two paths must joint the same initial and final positions. Notice that this implies that the paths for both networks are coincident in the initial iteration. Figures 10(a) depicts the results obtained in environments with 30 obstacles. In Fig. 10(b) and 10(c) we depict the results obtained when 3 and 5 networks are considered.

As in Section 3, the algorithm was run with the same parameters as those reported in [1]. The dimension of the "virtual obstacles" around each PU is $\frac{1}{3}$ of the size of the actual random obstacles.

5 Discussion

From our results, we claim ODEN to be an simple but efficient algorithm, being able to provide obstacle-free paths even in environments with a large percentage of occupied area. In a practical application, ODEN simply requires a function that samples a point in free-space around each robot in the network. This information can easily be retrieved from sensorial data and makes this method simple

to implement. On the other hand, the sampling mechanism supporting ODEN implies that the algorithm is less effective in environments with little free-space or with very thin obstacles. In the former case, it will be hard to sample points in free-space to pull the network away from the obstacles. In the latter case, it will be hard to sample the obstacles so as to drive the network away from them.

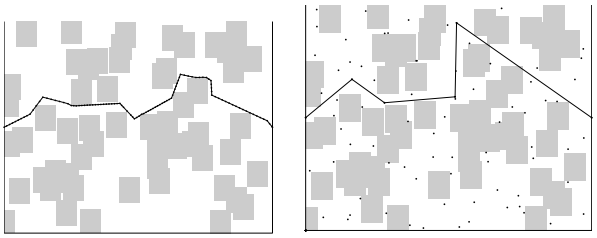
MODEN extends the simple principle behind ODEN to multi-path scenarios, while remaining a local method driven by obstacles. MODEN attests the applicability of ODEN in more complex problems and suitably illustrates the effectiveness of ODEN's underlying working principle. Nevertheless, and in spite of the encouraging results, we should remark that if the initial straight-line path of the several networks intersects, it is generally not possible to ensure a non-intersecting solution. And it may also happen that the path for the first network renders the task of finding an obstacle and intersection-free path infeasible.

As future work, it would be interesting to explore the use of richer attraction functions F. If the attraction function F is more than an "obstacle-indicator", the algorithm may use the extra information to drive the updates in a more informed fashion and thus improve its performance. In particular, the indicator function can provide information about signal strength, not only pulling each node toward free-space but inclusively drive each node/robot toward directions in which the signal strength is improved the most. This would bring our approach closer to potential-field/vector-field methods [9].

It is important to point out that, due to the local nature of the methods, neither ODEN or MODEN are optimization methods. This means that the network is adjusted *locally* to reach an obstacle free path. Since it departs from a straight-line condition, it is expectable that it may be able to reach a network configuration that is "close" to the straight-line path. However, and as other probabilistic path-planning methods (see, for example, Fig. 11), the configuration attained will generally not be optimal in any particular sense.

Another interesting discussion arises from the consideration of ODEN and MODEN as *path planning* algorithms, in which each network corresponds to the actual path of *one* robot. In this context, it is important to note that most path-planners are *global planners* in that they use information from the whole network which may not be practical (or even feasible) in the scenarios considered in this paper. Also, they are typically off-line methods, thus less suited for on-line implementation as intended in the applications envisioned here. On the other hand, and from a multi-robot path planning perspective, the requirement of non-intersecting paths in MODEN is not such a usual approach, as more elaborate ways exist to coordinate multiple robots wandering in a common environment (see, for example, the approaches in [10–12]). Nevertheless, non-intersecting paths alleviate the need for any knowledge of the robot dynamics/communication capabilities. The paths generated by the algorithm would be immediately usable by the robots, without considering any coordination or synchronization mechanism to prevent on-path collisions.

We note that, as with ODEN, environments with little free space also cause difficulties to probabilistic path planning methods such as PRMs [13]. Also, the



(a) ODEN. (b) PRM.

Fig. 11. Trajectories obtained with ODEN and PRM.

fact that ODEN uses all nodes in defining its path implies that, in general it will be able to locally adjust to the obstacles, producing potentially shorter paths. Figure 11 compares the paths obtained with ODEN and PRM in an environment with 60 random obstacles. A more extensive comparison of the performance ODEN againsta that of other state-of-the-art planning methods is still necessary to further understand the applicability of ODEN in more general scenarios than those considered here.

References

1. Moreno, J., Castro, M.: Heuristic algorithm for robot path planning based on a growing elastic net. In: 12th Port. Conf. Artificial Intelligence. (2005) 447–454
2. Ludwig, L., Gini, M.: Robotic swarm dispersion using wireless intensity signals. In: Int. Symp. Distributed Autonomous Robotic Systems. (2006) 135–144
3. Howard, A., Siddiqi, S., Sukhatme, G.: An experimental study of localization using wireless ethernet. In: 4th Int. Conf. Field and Service Robotics. (2003)
4. Peng, C., Shen, G., Han, Z., Zhang, Y., Li, Y., Tan., K.: A beepbeep ranging system on mobile phones. In: Conf. Embedded Networked Sensor Systems. (2007)
5. Poduri, S., Sukhatme, G.: Constrained coverage for mobile sensor networks. In: IEEE Int. Conf. Robotics and Automation. (2004) 165–171
6. Priyantha, N., Chakraborty, A., Balakrishnan, H.: The CRICKET location-support system. In: 6th ACM Conf. Mobile Computing and Networking. (2000) 32–43
7. Ziparo, V., Kleiner, A., Nebel, B., Nardi, D.: RFID-based exploration for large robot teams. In: IEEE Int. Conf. Robotics and Automation. (2007) 4606–4613
8. Reich, J., Misra, V., Rubenstein, D., Zussman, G.: Spreadable connected autonomic networks (SCAN). TR CUCS-016-08, CS Dept., Columbia Univ. (2008)
9. Barraquand, J., Langlois, B., Latombe, J.: Numerical potential field techniques for robot path planning. IEEE Trans. Systems, Man, and Cybernetics **22**(2) (1992) 224–241
10. Bennewitz, M., Burgard, W., Thrun, S.: Optimizing schedules for prioritized path planning of multi-robot systems. In: IEEE Int. Conf. on Robotics and Automation. (2001) 27–276
11. Švestka, P., Overmars, M.: Coordinated path planning for multiple robots. Robotics and Autonomous Systems **23** (1998) 125–152
12. Clark, C., Rock, S., Latombe, J.: Dynamic networks for motion planning in multi-robot space systems. In: Proc. Int. Symp. Artificial Intelligence, Robotics and Automation in Space. (2003)
13. Kavraki, L., Švestka, P., Latombe, J., Overmars, M.: Probabilistic roadmaps for path planning in high-dimensional configuration spaces. IEEE Trans. Robotics and Automation **RA-12**(4) (1996) 566–580