# STOCHASTIC MULTI-AGENT PATROLLING USING SOCIAL POTENTIAL FIELDS

Evgeny Shvets
Institute of Information Transmission Problems
Moscow, Bolshoy Karetny per. 19, build.1
Email: leednee@iitp.ru

**KEYWORDS**

Multi-agent Patrolling; Agent-based modeling.

**ABSTRACT**

In this paper we consider a task of decentralized, multi-agent patrolling of a continous outdoor terrain. We propose an algorithm that efficiently operates under the condition of low communication throughput and is robust to the failure of one or more patrolling agents. The solution is based on Social Potential Fields and is easily extensible to allow other types of behavior. We describe an agent-based simulation system and use the obtained results to show how the patrolling algorithm should be altered to be effective on different types of terrain. Several techniques to increase the efficiency of patrolling are provided.

## INTRODUCTION

Patrolling is an inherently multi-agent task of providing a uniform frequency coverage of an area and detecting intruders. Efficient patrolling algorithms are useful for military surveillance systems; they can also be used in civil applications, for example, automatic garbage collectors systems, and even in computer strategy games.

There is a significant volume of research on the multi-agent patrolling; however, the algorithms of patrolling vary significantly depending on the type of area to patrol, the number of patrolling agents and the additional requirements upon the algorithm.

In this paper we focus on the patrolling of an outdoor, continuous terrain. We put some additional requirements on the patrolling algorithm:

- Decentralization: single center that governs agent movement presents a security risk. If it is hacked or destroyed, the whole system becomes dysfunctional. It is much harder to compromise the system when each patrolling agent makes decisions of its own.
- Unpredictable movement of agents. When the agents move over regular paths, it is very easy for an intruder to predict their movement and plan the infiltration into the guarded object.
- Ability to operate efficiently when the communication with other agents is limited or shut down.
- Efficient behavior of the patrolling system when the number of agents changes dynamically. When new agents are added to reinforce the patrolling task, or some agents are broken or removed from patrolling, the remaining agents should alter their behavior so that the patrolling continues in an efficient manner.
- The solution should be easy to extend to allow other types of behavior, for example, surrounding of an intruder or a suspicious object.
- Since the system is decentralized, and each agent should make its own decisions, the algorithm should be computationally easy enough to run in the real-time on an average CPU.

Fitting these requirements is essential for a system of robots patrolling a military object, as it can be targeted for infiltration. Although patrolling is a well-studied task, we were unable to find an existing solution that satisfies these requirements, so we devise our own. In the following section we analyze the existing body of relevant research and identify what existing algorithms we can use as a basis for developing our own. Then we outline the structure of the algorithm used by each agent and divide the complex problem of patrolling into separate easy tasks. We consider different scenarios and solve the problems necessary to implement the algorithm. Finally, we formulate the final version of the algorithm, conclude the paper and present the directions of our future work.

## OVERVIEW OF EXISTING PAPERS

A very common technique used to simplify the patrolling task is skeletonization – representation of the continuous terrain as a graph. For that, the area is split into parts; nodes of the graph correspond to these parts, and nodes corresponding to the neighboring parts are connected with graph edges. Usually the area is split into parts in such a way that when an agent visits a node, it can observe every point of area corresponding to the node. An example of terrain that is easily represented as a graph is an office space: nodes correspond to the rooms, and edges correspond to the corridors and doors between them. The advantage of graph representation is that many mathematical results can be used to optimize the movement of agents over the graph.

Once the area is represented as a graph, there are a lot of ways to organize the patrolling. Simplest one to organize a cyclical movement of patrolling agents over the graph. Such patrolling algorithm provides very high frequency coverage; unfortunately, the agents move over the static paths, and their movement is very predictable; therefore, this solution does not fit our requirements. Other way is to divide the graph into several parts; each

Proceedings 29th European Conference on Modelling and Simulation ©ECMS Valeri M. Mladenov, Petia Georgieva, Grisha Spasov, Galidiya Petrova (Editors)
ISBN: 978-0-9932440-0-1 / ISBN: 978-0-9932440-1-8 (CD)

agent gets a part of the graph to patrol. The movement of agents along these parts can be cyclical as well (which makes it predictable); or each agent can use more complicated algorithms to patrol its part of the graph. One advantage of this method is that agents are always uniformly distributed among the patrolled area. Still, such patrolling is predictable in a sense that each agent does not leave its designated part. Also the method isn't robust to the failure of one or more of the agents and to the addition of new agents: the patrolling graph should be re-divided between the agents every time a agent is added or removed; and the graph may have a structure that can't be efficiently divided into the necessary number of parts.

A more suitable approach is not to determine the movement path of each agent in advance, but rather make each agent continuously re-estimate the situation on the patrolling area and make decisions on the fly. If the set of rules governing the behavior of agents is complex enough, it is almost impossible for an intruder to predict their movement. Several papers approach the patrolling problem this way: paper (Santana et. al. 2004) utilizes machine learning to organize the patrolling: agents change the patrolling algorithms dynamically in order to make the patrolling more efficient. Papers (Dias et. al. 2006), (Menezes et. al. 2006) and (Hwang et. al. 2009) propose "auction" or "market" patrolling algorithms, with which the system of agents decides which points of the map should be visited, and agents distribute these points between themselves according to different factors, such as their current positioning on the map. There are several drawbacks to these methods: first, they are complex, and second, it demands extensive communication and a reliable high-throughput radio channel. If the communication channel is hacked, the trespasser can obtain the goals of each agent and plan the intrusion. In case of a communication failure, the method becomes inefficient. Also, this approach is not easily extensible to provide other types of behavior such as surrounding the intruder when it is detected, or group patrol, when agents move as a single formation.

The approach we choose to adopt is based on "Social Potential Fields", as described in (Reif & Wang 1999). Potential field method is used in robotics for tasks such as planning robot movement (Ge & Cui 2002), obstacle avoidance and movement in different formations (Bravi et. al. 2012). With this method, the virtual forces are introduced. They are exerted on the robot from different objects on the patrolling area: from the unobserved points of territory, from other robots, and from obstacles. These forces can be both repulsion and attraction. The movement of robot is governed by the superposition of these forces. By varying these forces, one can achieve complex behaviors of robots: for example, to surround the intruder, a force pulling the robots towards the intruder should be introduced. Repulsive forces between robots will allow robots to spread uniformly over the territory.

A simple way to organize the patrolling is to implement the following principle: "each robot moves to the territory, which was observed the longest time ago, and is close to this robot". It can be done by introduction of virtual forces of attraction between each robot and recently unobserved points of the map; the force of attraction between each robot and the point of the area is proportional to the time the point was unobserved and inversely proportional to the distance between the point and the robot. This approach is inspired by (Chu et. al. 2007), where a virtual pheromone is introduced, evaporating from the unobserved territory and attracting the patrolling agents.

Papers above demonstrate how to use potential field method on a graph. However we need to develop an algorithm for patrolling of an outdoor area, and such area cannot be easily skeletonized. It is impossible to split a big continuous outdoor area into a graph simple enough to analyze. Paper (Portugal & Rocha 2013) presents an overview and comparison of commonly used patrolling algorithms and states that patrolling algorithms that operate on a graph lose a lot of efficiency when the graph is big and/or highly-connected. Furthermore, splitting the territory into a graph implicitly implies that when a robot visits a node, it observes every part of the area corresponding to a graph, no matter which direction robot came from and which direction it is moving. In a real scenario, robot may have a directional field of view, and the area that it observes depends on the direction it is facing; also the points of area further away from the robot are observed with lower quality. When a big part of area is united into a single graph node, it is impossible to distinguish, which point of this area is observed with which quality. Therefore skelezotonization of territory is a noticeable simplification, and we do not use it.

There are not as many algorithms devoted to the patrol that do not rely on the skeletonization. Such algorithms may use ideas similar to ones used in patrolling over a skeletonized terrain; however, the algorithms for graph patrolling need to be modified to be usable on a continuous area. We were unable to find a solution for our problem in the existing body of research. Paper (F.Sempe & Drogoul 2003) provides relevant adaptive algorithms, but it does not specify them in enough details for the implementation. Paper (Bravi et.al. 2012) provides an interesting solution for group patrol, where patrolling agents move in a necessary formation, but it doesn't help with the patrolling task where each agent moves independently.

**Formal Criteria for Patrolling Algorithm efficiency**

Let us call the time since the point $T$ has last been observed (by any of the robots) the 'Fog of War" ($FoW$) in point $T$. To estimate the efficiency of the patrolling algorithm, papers (Portugal & Rocha 2013), (Chu et. al. 2007) propose to use the average FoW of the points of the map over the duration of patrolling. Sometimes additional requirements are introduced, such as "at each point of the patrol, each point of the map should be reachable by at least one robot in the preassigned time". In this paper we use other criteria. Let us denote $MFoW(t)$ the maximum $FoW$ over the map a the moment $t$. We use the average $MFoW$ over the time of patrolling to estimate its efficiency. Paper (Portugal & Rocha 2013) shows that average $MFoW$ : $FoW$ ratio is different for different patrolling algorithms, but for most of them it lies between 2 and 3. The rationale for choosing the $MFoW$ criteria is that (i) it is simple and (ii) it guarantees that an alien

object is found fast no matter where it is placed in the area. Using average $MFoW$ as a criteria lets us recognize and discard algorithms that visit some points of the map notably more frequent than others. Such algorithms are flawed: by observing the area, the intruder may notice that some parts of the territory are protected worse than others, and use that knowledge.

In this paper we introduce the normalized Fog of War $nFoW(t,T)$. It is calculated as $nFoW(t,T) = FoW(t,T)/MFoW(t)$. That is, the normalized fog of war at each point of territory takes values between 0 (just observed point) and 1 (the point that hasn't been observed for the longest time). In the rest of the paper, we use only the normalized fog of war but use the $FoW$ abbreviation.

## OVERVIEW OF THE PROPOSED ALGORITHM

In compliance with criteria above, the efficiency of a patrolling algorithm is determined by how often robots observe points with high $FoW$. The simple idea behind our algorithm is that each robot moves to the points of the map which have high $FoW$. In order for all the robots not to move to the same points, each robot prioritizes moving to the points that are closer to itself.

We assume that the patrolling robots continuously exchange information about their current positions. The positions of other robots are used to constantly reestimate the $FoW$ of the points of the map. This information can be sent even over a low-throughput channel. While performing the patrolling task, they do not share any other information. Note that even when the communication is completely shut down, robots can continue the patrolling with lowered efficiency.

In this paper we assume that the patrolling terrain is known and each robot has a map of it. Exploration and mapping of an area is a widely researched problem and falls outside of the scope if this paper. To represent the terrain as a map, we divide the territory into the squares of 10x10 cm and call each square a point of territory. Each of the points is either passable or impassable (an obstacle). Every part of every robot should be positioned in a passable cell at every point of the time.

Let $P(R,T)$ be the path from robot $R$ to the point $T$ of patrolling area. The path should bend around the obstacles of the terrain and isn't always a straight line. Let us denote the direction of the path at the point $R$ as $D$. That is, the direction $D$ of the path $P(R,T)$ is the direction in which the robot should move from its position $R$ in order to traverse through the path and reach point $T$. Let $L(R,T)$ be the length of the path from robot $R$ to the point $T$.

Here is the outline of the algorithm to be used by each of the robots:

1) Find the path from robot $R$ to each point $T$ of the area. Determine direction $D$ and length $L$ of the path.
2) Calculate forces exerted on the robot by each point $T$ of the map. The greater the $FoW(T)$ is and the lesser the $L(R,T)$ is, the greater is the force. The force is exerted in the direction $D$ of the shortest path between $R$ and $T$.
3) After all forces exerted on the robot by the points are found, determine where the robot should move based on these forces.

Therefore, to develop the algorithm we need to find the solutions to the following problems:

1) A simple but efficient pathfinding algorithm that finds paths from point $R$ to each point $T$ of the area.
2) Function $F(R,T)$ that determines the force exerted on robot by points of the area.
3) A method for choosing robot movement direction when all forces exerted on robot are known (henceforth called "direction decision method").
4) An additional algorithm for obstacle and mutual collisions avoidance while patrolling.

Each of these problems can be solved in several ways. Efficiency of the resulting algorithm depends both on the quality of the solutions for each problem and their compatibility. However it is nearly impossible to try every possible combination of these solutions. To ease the task we first consider the scenario where (i) patrolling robots are holonomic, that is, they can move in any direction and change their speeds instantly, and (ii) the terrain to patrol contains no obstacles. Then we increase the complexity of scenarios, adding obstacles to the terrain and considering nonholonomic robots. This way we can outline the algorithm using simple scenarios and refine it using the complex ones.

In the next section we consider following scenarios:

1) Patrolling of an area without obstacles using holonomic robots. In an area without obstacles, there is no need for a pathfinding algorithm: the path is always a straight line connecting $R$ and $T$. Therefore we can focus on choosing the optimal function $F$ and the direction decision method.
2) Patrolling of an area containing obstacles using holonomic robots. After the $F$ function is chosen, we can focus on the pathfinding algorithm.
3) Patrolling of an area containing obstacles using nonholonomic robots. After both $F$ and pathfinding algorithm are chosen, we adjust the algorithm to be able to operate with nonholonomic robots.
4) The final refinement of the algorithm to decrease the number of collisions between robots and with obstacles.

We consider these scenarios separately and sequentially. For each scenario, we pose the problem, provide an algorithm to solve it, and present corresponding numerical results.

## DIFFERENT PATROLLING SCENARIOS

### Patrolling of an area without obstacles using holonomic robots

This simple scenario is used specifically to determine the optimal function $F$. There are two reasons for that: firstly, the efficiency of a patrolling algorithm is especially easy to judge on a territory that has no obstacles: there

is only a few possible directions for robot movement in an obstacle-rich environment. For example, in a narrow corridor, robot can move only forward or backwards. In an open space the robot choses between many possible movement directions; therefore, patrolling of a territory without obstacles helps to highlight the weaknesses of a choice of $F$ function. Secondly, the lack of obstacles allows to use the simplest pathfinding algorithm (the path between two point is a straight line connecting them) and focus on optimizing the $F$ function.

We use two direction decision methods:

- Vector sum method. All forces are summed as vectors; robot moves in the direction of resulting force.
- Mode method. All possible directions of robot movement (360 degrees) are split into several sections. For each sector, the sum of modules of forces directed in that sector is found. The robot moves along the bisetrix of the sector with maximum sum.

In this paper we consider only the functions $F(R,T)$ that can be expressed as $F(R,T) = f(FoW(T)) \cdot g(L(R,T)))$, where $f$ is a function of normalized fog of war in point $T$ and $g$ is a function of length of the path between $R$ and $T$. We consider functions $f$ and $g$ presented in Table 1.

Table 1: Functions $f$ and $g$ considered in the paper

| | |
|---|---|
| $f_1 = FoW(T)$ | $g_1 = \frac{1}{L(R,T)}$ |
| $f_2 = FoW^2(T)$ | $g_2 = \frac{1}{L(R,T)^2}$ |
| $f_3 = e^{\alpha \cdot FoW(T)}$ | $g_3 = e^{-\frac{l \cdot L(R,T)}{\beta}}$ |
| $f_4 = e^{\alpha \cdot FoW^2(T)}$ | $g_4 = e^{-\frac{L^2(R,T)}{\beta}}$ |
| $f_5 = e^{\alpha \cdot FoW^4(T)}$ | $g_5 = e^{-\frac{L^4(R,T)}{\beta l^2}}$ |

In the formulas presented in the table, $\alpha$ and $\beta$ are normalization coefficients for efficient use of floating point arithmetics, and $l$ is a typical linear size of the area to patrol:

$$\alpha = 40,$$
$$\beta = 4000,$$
$$l = 80\ m.$$

By choosing different $f$ and $g$ functions we can change how strongly $F$ is affected by the fog of war and by the path length. For example, if we choose $F = f_5 g_1$, the force exerted on robot relies heavily on FoW and in a much lesser degree on the path length. By trying every possible combination of functions, we can find the optimal one.

The simulation of this scenario takes place on a virtual terrain – a rectangle of 100m x 88 m size. The uneven size is chosen because patrolling on an even square could have regular behavior; most real terrains aren't ideal squares. We assume that robots move with the speed of $4.4m/s$ , or $15.8km/h$. This corresponds to the speed of real robots we plan to deploy the system on. Each robot is simulated as a circle object with a radius of 1 meter. Therefore, we assume that the minimum distance between the centers of two robots equals 2 meters; otherwise, a collision occurs. We use step-by-step simulation, with one step corresponding to $0.5$ seconds of virtual time. At each step, every robot estimates the situation on the area according to the algorithm described above and chooses the action. In a case of a collision happening, both robots cancel their movements for this step. To avoid "deadlocks", when two robots block each other movement for an extended period of time, and to introduce a probabilistic element to the patrolling, we postulate that with $10\%$ chance each robots moves in a random direction every step of the simulation. Since one simulation step is $0.5$ seconds, one random move does not change the robot position much; but it turned out that potential fields are sensitive, and $10\%$ chance of a random move is enough to make every simulation run different. Every simulation lasts for $10000$ iterations, or $5000$ seconds of virtual time.

In this scenario we use 3 robots for the patrolling. The results are presented at tables 2 and 3.

Table 2: Results of simulation with vector sum direction decision method, average $MFoW$, seconds

| | $g_1$ | $g_2$ | $g_3$ | $g_4$ | $g_5$ |
|---|---|---|---|---|---|
| $f_1$ | 1476.0 | 1306.9 | 136.1 | 199.2 | 763.4 |
| $f_2$ | 927.5 | 290.5 | 119.1 | 126.0 | 218.1 |
| $f_3$ | 260.3 | 143.9 | 101.5 | 80.7 | 111.1 |
| $f_4$ | 257.9 | 119.2 | 204.1 | 76.1 | 120.6 |
| $f_5$ | 303.4 | 125.0 | 243.5 | 94.3 | 125.6 |

Table 3: Results of simulation with mode direction decision method, average $MFoW$, seconds

| | $g_1$ | $g_2$ | $g_3$ | $g_4$ | $g_5$ |
|---|---|---|---|---|---|
| $f_1$ | 1211.3 | 545.6 | 132.7 | 144.3 | 394.2 |
| $f_2$ | 490.1 | 204.7 | 127.9 | 113.8 | 167.1 |
| $f_3$ | 167.0 | 133.0 | 98.5 | 90.0 | 125.8 |
| $f_4$ | 167.9 | 140.8 | 118.2 | 92.9 | 116.5 |
| $f_5$ | 168.5 | 142.3 | 192.8 | 106.7 | 128.3 |

As seen from the tables, the best combinations are $f_4 g_4$, $f_3 g_4$ and $f_4 g_3$. Vector sum movement decision method is slightly better for the patrolling of a terrain with no obstacles. However, its is shown below that on real terrains with obstacles mode method performs notably better. In the rest of the paper we use the following formula to calculate the force exerted on each of the robots:

$$F(R,T) = f_3(T) \cdot g_4(R,T) = e^{\alpha \cdot FoW(T)} \cdot e^{-\frac{L^2(R,T)}{\beta}} \quad (1)$$

**Patrolling of an area containing obstacles using holonomic robots.**

Presence of obstacles introduces the necessity for a pathfinding algorithm. Given the position $R$ of a robot, it is necessary to find a path to every point $T$ of the map, and the lengths $L$ and directions $D$ of these paths. The pathfinding algorithm should be computationally easy to run in real-time and continuously re-estimate the paths.

We tried several pathfinding algorithms, and only two of them performed good and fast enough. Below we describe and compare them.

*1. "The wave" (Lee pathfinding algorithm).* This method uses the representation of a patrolling area as a weighted graph. Nodes represent the points of territory (each point corresponds to a 10 x 10 cm square, as described above). Neighboring nodes are connected: nodes that are connected via a vertical or horizontal edge have a weight of 1; diagonals have a weight of 1.41. The distances to each point of the territory are found using Lee path connection algorithm (Rubin 1974). Note that with this method, all forces exerted on robot will have one of the 8 possible directions, based on what is the first step of the path found by Lee algorithm.

*2. The region method.* With this method, the passable part of the patrolling terrain is split into several parts – so-called "regions", with each part being a convex figure. Robots can move between any two points within one region over a straight line. On the borders between each two neighboring regions a connecting point is chosen. All paths traversing from one region to another pass through the connection point between these two regions. With this algorithm, pathfindning becomes an easy problem: to find the path it is only necessary to find the correct sequence of connection points to move through; the resulting path is a polygonal line connecting them. However, to use this method it is necessary to split the area into regions in an efficient manner. The complexity of the splitting task is decided by the number and the shape of obstacles: region method is very effective when there is limited number of obstacles in the area, but it loses efficiency if there are a lot of obstacles and/or obstacles are of complex shape. Figure 1 shows an example of a patrolling area split into regions (dotted lines represent the borders between them).

In a general case, the length of the path found with region method can be found as:

$$L(R,T) = \rho(R,CP_1) + \rho(CP_1,CP_2) + ... + \rho(CP_n,T),$$

where $CP_i$ is the $i$-th connection point and $\rho(X,Y)$ is Euclidian distance between points $X$ and $Y$.

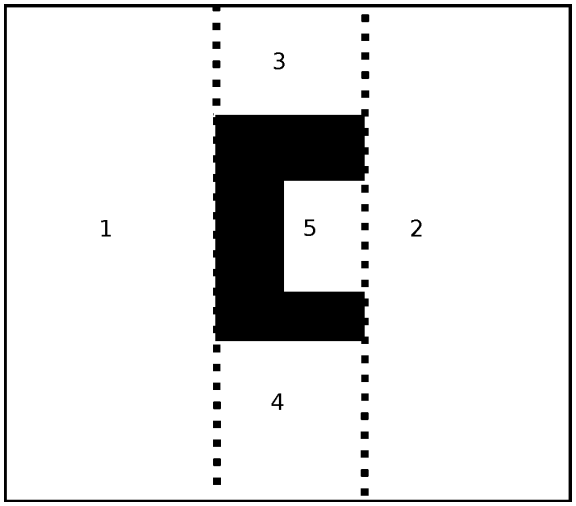


Figure 1: Patrolling Terrain, Split into 5 Regions

The main advantage of the region method is that it works significantly faster than the wave method, especially on large areas with small number of obstacles. The drawback is that areas with complex obstacles are hard to split into proper regions; and the efficiency of patrol depends heavily on the way the area is split. It is even harder to split a complex terrain into regions manually, therefore an algorithm should be developed and an automatic software splitting tool should be used. The region method is very computationally efficient, and developing a tool for region splitting is an important direction of future work. However, in this paper we do not propose such an algorithm and do not consider the problem of finding the optimal way to split the territory. In what follows we use the wave method, unless stated otherwise.

Table 4: Results of simulation for various pathfinding algorithms and decision methods (holonomic robots). Average $MFoW$, seconds

| Number of Robots | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| Wave + Mode | 326.9 | 186.8 | 123.8 | 110.9 | 81.6 |
| Wave + Vector Sum | 373.8 | 207.7 | 122.2 | 105.4 | 84.8 |
| Regions + Mode | 322.4 | 174.1 | 119.2 | 97.4 | 76.9 |
| Regions + Vector Sum | 289.7 | 153.0 | 101.0 | 78.7 | 71.3 |

For the simulation of this scenario, we use the terrain shown in the Fig. 1. Size of the terrain is 100m by 88m. Forces exerted on robot by points are calculated with the help of (1). We consider both mode and vector sum method and two pathfinding algorithms: wave method and region method. The results of simulation are presented in Table 4.

Important thing to note is that vector sum decision method becomes less and less efficient compared to mode vector when more obstacles are introduced into the area. Let's consider another terrain for the next simulation; it is shown in Figure 2. Size of the terrain is 118 by 118 meters; the results are presented in the Table 5.

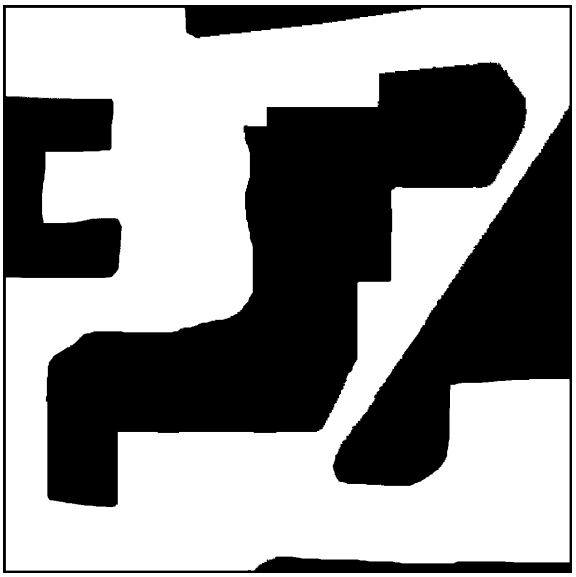

Figure 2: Patrolling Terrain with Additional Obstacles

Table 5: Results of simulation with wave pathfinding algorithm (holonomic robots). Average $MFoW$, seconds

| Number of Robots | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| Mode | 534.9 | 231.7 | 160.5 | 131.4 | 112.7 |
| Vector Sum | 513.5 | 264.4 | 201.0 | 164.5 | 122.7 |

At this map mode direction decision method is clearly preferable over vector sum. An example of why vector sum method becomes less efficient is simple: let's assume there are two areas with high $FoW$: to the front of the robot and to the right. With vector sum method, the robot will move to the right and to the front simultaneously, which will lead him to neither groups of points. What is more important, the robot may move into an obstacle. The higher number of obstacles, the more likely such a scenario becomes. We choose to use mode method because it is more reliable and stable in a wide variety of scenarios.

**Patrolling of an area containing obstacles using nonholonomic robots.**

We use robots that have a minimum turning radius of 3.5 meters. In simulation, we put similar restraints on the robot movement. Parameters of the algortihms we use below are optimized for such turning radius as well. These optimal parameters are obtained through simulation.

After both $F$ function and pathfinding algorithm are chosen, we adjust the patrolling algorithm to be able to operate on nonholonomic robots. The algorithm described above cannot be directly used with nonholonomic robots because it sometimes requires the robot to change the movement direction and speed instantaneously: for example, when robot is moving to an unobserved point, and that point gets observed by another robot. Such instantaneous turn or change of speed is impossible for nonholonomic robots. To accommodate the algorithm for nonholonomic robots, we make the robot turn his wheels and move forward so that he positions himself in the necessary direction as soon as possible, instead of instantaneously changing the direction.

Another obvious problem is that paths built with the described pathfinding algorithms often cannot be precisely followed by nonholonomic robots. However, replacing the pathfindng algorithm with one that builds a correct and always passable path proved to be ineffective: accounting for robot nonholonomity increases the computational complexity of pathfinding algorithms dramatically (it is a problem because we need to find paths to every point of the map). The problem with paths built by the two proposed algorithms is only relevant for those points of the area close to the robot, as, with rare exceptions, even the nonholonomic robot can reach a distant point using the path close to the built one. Therefore using the pathfinding algorithms described above seems to be the best way to maintain the balance between calculational complexity and efficiency. We still need to provide some additional rules to the robot movement algorithm to compensate for the situations when this algorithm leads the robot to a so-called "dead end" – a position, where the robot is stuck, and the default patrolling algorithm cannot provide a robot with correct instructions for further movement. Dead ends usually happen in the corners of the obstacles.

We propose a simple but efficient algorithm to escape the dead end (henceforth "unstuck algorithm"):

1) When the robot detects it is stuck, it chooses the direction it is willing to be oriented in after it leaves the dead end. This is the direction in which the most forces are exerted on the robot at the moment when it detects it is stuck.
2) Robot moves back steering in a way that the angle between the desired orientation and robot orientation decreases.
3) If the robot detects that it has oriented in the necessary direction, the unstuck algorithm finishes its work.
4) If the robot encounters an obstacle behind and can no longer move backwards, it starts moving forward steering so that it orients toward the target direction. He does so until he orients in the necessary direction or encounters an obstacle.
5) Robot repeats steps 2-4 until it leaves the dead end.

The described algorithm is already functional; however with this algorithm robots spend a lot of time in dead ends. It is bad because (i) it lowers the efficiency of patrolling and (ii) real robots have problems changing the direction of movement; starting off from the standing position is undesirable. Below we propose two methods to decrease the number of dead end situations. To evaluate this number we measure the share of time each robot spends in the "unstuck" algorithm (so it measures not only how often robots get stuck, but also how long it takes them to escape from the dead end).

**Reducing the Time Spent in Dead Ends**

We propose two refinements to the algorithm to help robots reduce the number of dead ends situations.

First one is called "goal refinement". In the original algorithm, the direction decision method provides robot with a direction, and robot steers and moves in that direction. With this enhancement, the robot converts the direction into a goal point to move to ($g$ meters from robot along the chosen direction. Numerical results proved $g = 5.5$ meters to be most effective). The goal chosen by robot is then modified. The algorithm considers a square centered at the goal point, with a side of $R = 2$ meters. For each point $(x_t, y_t)$ in this square the function $\psi$ is calculated that describes how close to the obstacles the $(x_t, y_t)$ point is. The greater the function, the closer to the obstacles the point is:

$$\psi = \Sigma_{x\in(x_t-A,x_t+A);y\in(y_t-A,y_t+A)} \frac{obst(x,y)}{\rho^2},$$

where $obst(x, y)$ equals 0 if $(x, y)$ point is passable and 1 otherwise; $\rho$ is distance from $(x, y)$ to $(x_t, y_t)$ and $A$ determines the maximum distance to the obstacles that should be taken into the equation. Essentially, every nearby obstacle point gives a penalty to the point, by increasing $\psi$ function; the closer the obstacle is, the

bigger is the penalty. The point $(x_t, y_t)$ with the lowest value of $\psi$ is chosen as the new goal for the robots movement. With this algorithm, the robot moves roughly in the necessary direction, but "tries" to stay away from the obstacles.

The second approach is inspired by fuzzy logic. This approach is applicable only to the algorithm using mode direction decision method. After the forces are summed in each of the 8 sectors, the sum for each sector is multiplied by a coefficient $c_x$ based on the distance $R_c$ to the closest obstacle in a sector:

$$c_x = \begin{cases} 1, & R_c \geq R_0 \\ R_c/R_0, & R_c < R_0 \end{cases},$$

here $R_0$ is a constant equal to 5.5 meters.

Results of simulation for these algorithms are presented at Table 6 for $A = 10$ meters. As seen from the table, goal refinement is the superior method, but fuzzy logics approach also provides slight efficiency boost.

Table 6: Results of anti-collision algorithms. Nonholonomic robots

| | Obstacle Avoidance | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| Avg. $MFoW$, sec. | Pure | 601.0 | 343.7 | 248.4 | 191.8 | 187.2 |
| | Goal refinement | 493.0 | 290.2 | 221.2 | 167.5 | 136.0 |
| | Fuzzy logic | 494.1 | 344.9 | 219.8 | 171.0 | 142.5 |
| Time Spent in Dead Ends, % | Pure | 6.0 | 7.1 | 9.1 | 9.3 | 13.5 |
| | Goal refinement | 1.9 | 4.8 | 5.5 | 6.5 | 7.4 |
| | Fuzzy logic | 3.3 | 5.3 | 7.2 | 7.8 | 8.7 |

## OTHER ADDITIONS TO THE ALGORITHM

A significant inefficiency of the proposed algorithm manifests itself when robots "miss" the small unobserved areas. For example, the robot may be situated between a very small area that hasn't been observed for long time and a big area that hasn't been observed for an average amount of time. The robot will often move to observe the bigger area, when it would be better to move to the lesser area, as the $MFoW$ value is determined by that little unobserved area.

To fight this phenomena we introduce the "dilatation" mechanism. It is the modification of the $FoW$ in every point of the territory according to the formula:

$$FoW(x, y) = max(FoW(x', y')),$$
$$x' \in (x - d, x + d),\ y' \in (y - d, y + d),$$

that is, $FoW$ in every point of the map is worsened to match the highest $FoW$ in a window with $d$ side. Robots are less likely to "miss" small unobserved pieces, as with this technique small unobserved areas efficiently have increased size.

For the simulation, we considered the terrain shown at Figure 3. This terrain is a model of a real terrain on which we plan to deploy the simulation with real robots. Size of the terrain is 240 by 160 meters. Result for the simulation are presented in Table 7. As seen from the table, dilatation is a very efficient technique to increase the efficiency of patrolling. In the modifications proposed below, we use dilatation with $d = 5$ meters.

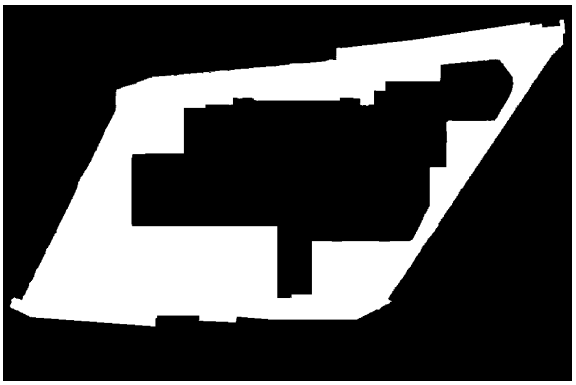

Figure 3: Patrolling Terrain

Table 7: Effect of dilatation mechanism on the efficiency of patrolling (nonholonomic robots). Average $MFoW$, sec

| | Dilatation wing $d$, m | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| Avg. $MFoW$, sec. | 239.5 | 252.0 | 222.7 | 203.0 | 173.3 | 172.7 | 181.0 | 187.5 |
| Time spent in dead ends, % | 6.1 | 4.8 | 5.9 | 5.7 | 6.6 | 6.5 | 7.3 | 7.1 |

The proposed algorithm has one other inefficiency: sometimes during the patrolling two robots start moving close to each other and along the similar path. This happens because forces exerted on them are roughly the same; therefore they move in the same direction, usually in the direction of a point with very high $FoW$. Such a phenomena noticeably lowers the quality of patrolling. To solve this problem we tried to introduce repulsive forces between robots; negative effect from such forces outweighed the positive effect: more often than not the the repulsive force from other robots shifted robot's trajectory in a non-optimal way; and on average the patrolling quality became worse. Due to the limited space we will not delve into details of this; but will instead outline two interesting solutions we found.

First one is called "Artificial Obstacles". With this solution, every robot creates a virtual obstacle with given radius $R_{obst}$ at the position of each other robot. As the result, paths produced by pathfinding algorithm do not cross the neighborhoods of other robots.

Second method is called "Semaphores". With this method we choose "critical areas" on the map, where the simultaneous presence of two or more robots may cause trouble: for example, a narrow corridor. Whenever a robot detects that other robot entered one of the critical areas, it renders the whole semaphore area impassable, in the same manner an artificial obstacle is created in the first method.

The results for these methods are presented at Table 8. In the table, we use the following abbreviations: GR –

Goal Refinement; S – Semaphores; AO – Artificial Obstacles.

Table 8: Variations of Patrolling Mechanisms(holonomic robots)

| | | $MFoW$ , sec | | | Time in dead ends, % | | |
|---|---|---|---|---|---|---|---|
| | | G.R. | S. | A.O. | G.R. | S. | A.O. |
| Number of robots | 1 | 632.5 | 632.5 | 632.5 | 5.0 | 5.0 | 5.0 |
| | 2 | 354.3 | 340.8 | 317.7 | 4.9 | 6.0 | 5.9 |
| | 3 | 250.8 | 249.9 | 233.3 | 5.4 | 6.5 | 6.5 |
| | 4 | 194.8 | 196.4 | 172.7 | 5.9 | 6.7 | 6.5 |
| | 5 | 158.2 | 153.3 | 154.6 | 5.9 | 6.9 | 6.8 |

As seen from the results, these methods make the patrolling more efficient; but also increase the number of dead end situations. The reason for such results is that robots avoid each other more efficiently; hence, the $MFoW$ is lower on average; however, to avoid each other robots have to do some additional maneuvering, and in some cases the maneuvers are complex and clearly inefficient (i.e. they fall into the "cycle" repeating the same mutual positions for several times until they manage to move along). These situations are rare, and we believe that a more elaborate versions of these two mechanisms can lower the $MFoW$ metric without increasing the number of dead end situations; developing such algorithms is a possible direction of our future work.

## FINAL ALGORITHM AND INTEGRATION WITH OTHER TYPES OF BEHAVIOUR

Final algorithm we propose:

1) The area is split into "points" – small areas of 10 x 10 cm.
2) Robot estimates the $FoW$ at each point of the map (by continuously updating $FoW$ near the location of itself and each other robot).
3) Robot applies dilatation mechanism to the resulting $FoW$ with $d = 5$ m.
4) Robot calculates paths to every point of the map using Lee pathfinding algorithm.
5) Robot calculates virtual forces exerted on it with (1).
6) Robot decides in which direction it should move, using mode direction decision method.
7) Robot chooses the goal $5.5$ meters along the chosen direction and modifies the goal with goal refinement mechanism.
8) Robot moves towards the goal.

Whenever the algorithm leads robot to a collision, the unstuck algorithm is activated.

We designed this algorithm to be effective in different scenarios. In each particular scenario, it is possible to slightly increase the efficiency of the algorithm; however, the proposed algorithm provides most stable results across the variety of scenarios.

With this method, it is very easy to implement other types of behavior using Potential Fields method. For example, if we want the robots to surround an intruder, we create the attraction force exerted on every robot from the intruder; we nullify forces exerted on robots by observed territory and introduce relatively weak forces repelling robots from each other, so that they surround the intruder.

It is possible to simulate the scenario where field of view of robots isn't circular. For example, field of view may be a $90$ degrees sector. Only thing we need to do to account for that is update the $FoW$ not in every point of territory around the robot, but rather only in the sector in front of it. We can also simulate different qualities of observation: we decrease $FoW$ weakly for the the points further from the robot: i.e. we decrease it not to $0$, but rather to some greater value.

## CONCLUSIONS

We proposed a distributed, efficient method for multi-agent patrolling of an outdoor area based on the Social Potential Fields. We provided guidelines on how to tweak the method to obtain optimal behavior in different scenarios and proposed several efficient techniques to enhance a patrolling algorithm.

The main direction of upcoming research is deploying the developed system on real robots and organizing the patrolling robot team. Regarding the algorithms, we will work on pathfindings algorithms for nonholonomic robots and integration other types of behavior into the system. An automatic tool for splitting the terrain into regions could enhance the quality of patrolling.

This research was conducted at the IITP RAS and supported by the Russian Scientific Fund grant #14-50-00150.

## REFERENCES

Bravi, A., Corradi, P., Schlachter, F. & Menciassi, A. (2012), Self deployment and coordination of an assembling swarm of robots, Physicomimetics, pp. 129 – 144.

Chu, H., Glad, A. & Simonin, O. (2007), Swarm approaches for the patrolling problem, information propagation vs. pheromone evaporation, Proceedings of the 19th IEEE International Conference on Tools with Artificial Intelligence - Volume 01, pp. 442 – 449.

Dias, M., Zlot, R., Kalra, N. & Stentz, A. (2006), Market-based multirobot coordination: A survey and analysis, Proceedings of the IEEE, 94(7), pp. 1257 –1270.

F.Sempe & Drogoul, A. (2003), Adaptive patrol for a group of robots, Intelligent Robots and Systems, pp. 2865–2869.

Ge, S. & Cui, Y. (2002), Dynamic motion planning for mobile robots using potential field method, Autonomous Robots Volume 13, Issue 3, pp. 207 – 222.

Hwang, K., Lin, J. & Huang, H. (2009), Negotiator agents for the patrolling task, ICCAS-SICE, pp. 4359 – 4363.

Menezes, T., Tedesco, P. & Ramalho, G. (2006), Negotiator agents for the patrolling task, Advances in Artificial Intelligence - IBERAMIA-SBIA 2006, Springer Berlin Heidelberg, pp. 48 – 57.

Portugal, D. & Rocha, R. P. (2013), Multi-robot patrolling algorithms: examining performance and scalability, Advanced Robotics 27.5, pp. 325–336.

Reif, J. & Wang, H. (1999), Social potential fields: A distributed behavioral control for autonomous robots, Robotics and Autonomous Systems, Volume 27, Issue 3, pp. 171 – 194.

Rubin, F. (1974), The lee path connection algorithm, Computers, IEEE Transactions on 100.9, pp. 907 – 914.

Santana, H., Corruble, V. & Ratitch, B. (2004), Multi-agent patrolling with reinforcement learning, Proceedings of the Third International Joint Conference on 97Autonomous Agents and Multiagent Systems-Volume 3. IEEE Computer Society, pp. 1122 – 1129.