ALMA MATER STUDIORUM – UNIVERSITY OF BOLOGNA
CAMPUS OF CESENA

Engineering and Architecture Faculty
Master's Degree in Engineering and IT

ROBOT EXPLORATION

Subject
INTELLIGENT ROBOTIC SYSTEMS

Presented by
GABRIELE GUERRINI

Accademic Year 2019 – 2020

Table of Contents

Chapter 1

Project goal

Few landmarks are scattered around the arena, and the main goal is to explore them all using swarm robotics techniques (non-deterministic transition between executed tasks, local sensing, low computational power of each robot etc.). Each landmark should be explored exactly once.
All robots are initially located in a base placed somewhere in the arena. The base is identified by a black spot on the ground, and a light source is placed above it.
A robot randomly starts exploring the arena and, when a landmark is detected, it may stop in order to create/join a cluster around such landmark. The probability to start exploration depends upon the time spent resting.
When a cluster is complete, the robots return to the base by executing phototaxis after a short phase of recognition (that would be the phase where the robots should actually explore and map the landmark surrounding area, but in this project it will be mocked for simplicity). The probability of the robot to join the cluster depends upon two parameters, i.e. the number of the robots that already joined the cluster and the time spent exploring.
A robot may leave a joined cluster randomly before it is completed depending on the two same aforementioned parameters.
A robot may quit exploration task randomly and return to the base. The probability to end such task is proportional to the spent exploration time.
In the end, few obstacles may be distributed in the arena, and so an obstacle avoidance activity must be executed along all the robot lifetime. Such task includes avoidance of any obstacle (walls and boxes as well as other robots and landmarks).

Chapter 2

State of the art

An overall of four different well-known tasks can be detected in the problem, namely:

- **Arena exploration**: the robots must explore the arena to find the landmarks.
- **Clustering**: the robots must create cluster near any landmark so it can be considered explored.
- **Phototaxis**: used to allow the robots to easily come back to the base.
- **Obstacle avoidance**: general behaviour to be executed to prevent any collision (between robots as well as between robots and obstacles/walls).

It follows a brief description of the state of the art for each task.

2.1 Arena exploration

The exploration of the area is a well-known task in the literature, and lots of algorithms and methodologies have been studied. However, when talking about swarm robotics, the main adopted method is the *random walk* approach. This is mainly due to the usage of off-the-shelf robots that own limited individual abilities (low processing power, local sensing etc.). Nevertheless, the usage of a swarm of robots for exploration tasks is still widely applied thanks to the advantages it entails respect to the usage of a single robot (flexibility, robustness, scalability).

There are few different common sense implementations of a random walk task [1], namely:

- Brownian motion
- Lèvy Flight
- Lèvy Taxis
- Correlated radom walk
- Ballistic motion

The main idea is to randomly choose a direction and go straight till a new one is selected.
All the aforementioned models share the same underlying mathematical model, and the different tuning of the parameters provides different behaviours and exploration capabilities. Only one exception can be identified, that is the ballistic motion, where the two parameters *step length* (μ) and *turning angle* (ρ) are not provided and/or limited.

Further from the above basic approaches, new solution have been probed to overcome the limits they impose. Indeed, there are two main problem that emerge, that is:

- Execution of repeated exploration of the same point.
- Exploration does not scale with arena widening. Ineed, just zona near the starting position will be quite well explored.

New approaches have been studied to overcome the aboe issues. The example reported in [2] improves the random walk by considering relative distribution of robots among the arena so that each zone can be equally explored by a even quantity of robots.

Once the robots are spread over the arena, different algorithm can be applied to map it (e.g., *GMapping* that produces a two-dimensional occupancy grid of the environment).

2.2 Clustering

Clustering is one of the main tasks identified in most of the proposed taxonomies for swarm robotics. The goal is to gather objects (the robot themselves or tokens to be moved) in one or more points of the arena. The task can be used for multiple purposes: foraging activity (group food supplies), divide robots upon classes that execute different tasks, etc..
All the proposed clustering algorithms create different solutions upon one of three common approaches [10], that is:

- Force-oriented solutions (similar to motor schema principles).
- Probabilistic approach by using non-deterministic transition among states and tasks. Indeed, as stated in [13] [9], randomness resulted to be a crucial aspect for emerging behaviours in swarm robotics in general.
- Artificial evolution through genetic algorithms.

Independently from the ad hoc approach, the main idea is to apply *swarm intelligence* principles to guide robots behaviour, and so to create a result inspired to various animal species (insects, birds, fishes) or natural laws (e.g., the settling process of liquids of different densities, [11]).
Beyond this, few more articulated experiments have been executed trying to add further capabilities to robots (e.g., *spatial awareness* and exchange of *virtual tokens*, [12]).
In the end, notice that swarm intelligence principles can be applied to other domains as well. Indeed, lots of experiments have been made about the clustering of data (instead of robots) in order to overcome traditional clustering algorithms such as *K-means* by blending swarm intelligence with other domains (e.g., game theory, [9]).

2.3 Obstacle avoidance

The obstacle avoidance task has been widely studied, and so few different algorithms have been proposed. The input values can be retrieved with distance sensors (e.g., sonar, proximity sensors) and/or visual sensors. In this case, we are going to focus on techniques that exploit the first ones solely.

The most known technique is the *Artificial Potential Field (APF)* that is based upon the motor schema idea by creating two potential fields: the first is attractive and generated by the goal point so that the robot can aim toward it, the second one is repulsive and it is generated from obstacles. Obviously, even though this method suffers all the issues of the motor schema approach such as local minima, it finds shorter paths than other well-know algorithm such as the Bug one [3] [4].
Other solutions, even though rely on the same underlying mathematical model, contemplate different types of potential fields: tangential fields to circumnavigate obstacles, Gaussian potential fields [8].
In addition, we notice the *Vector Field Histogram* algorithm that produces a vector force as output (length, angle), but differently from others it exploits peculiar tools as histograms to represent the distribution of obstacles around

the robot. The algorithm provides for three stages so that the initial 2D histogram is collapsed to a polar one.
Moreover, as well as the classical approaches, novel solutions have been explored by combining genetic algorithms and neural networks (as reported in [6]).
In the end, complete navigation with obstacle avoidance have been experimented by using the *Particle Swarm Optimization* algorithm [4].

Chapter 3

Solution development

3.1 Behaviour design

The behavior of each robot is modeled using a non-deterministic finite state automata (*NFA*)(see figure 3.1). Such an architecture has been chosen for the robot since its behavior is not trivial and more tasks must be executed at different times. By using a FSA, it is easy to explicitly express activities, conditions and non-deterministic transitions. Moreover, each state can be developed as a separated module, and the most suitable solution can be implemented case by case.
The robot starts in the "resting" state. A non-deterministic transition makes the robot leave the base and start the exploration with a probability $P1$.
When exploring, the robot looks for any close landmark and, if it is the case, it joins the cluster around such landmark with a probability $P2$ and enters in "waiting for cluster to complete" state. Moreover, the robot may quit the exploration task at any moment and return to the base with a probability $P3$. In the latter casuistry, the robot enters in the "returning to base" state.
When a cluster is completed by reaching the required number of robots N, the robot enters in the "reconnaissance" state. Such a state conceptually correspond to the phase during which the robots in the cluster should explore the area near the landmark. Such a task is not executed in this project for simplicity, but few different algorithms have been created to let a set of robots to explore and map a given area.
The robot ends the reconnaissance task after t_1 seconds, and it returns to the base.
When returning to the base, the robot executes a phototaxis task. The robot enters in the "resting" state once it is detected that it successfully returned to the base.
In the end, the robot may leave the cluster before it is completed with a

probability $P4$. In this case, it enters in the "biased exploration" state, that is a state where the robot executes the regular exploration tasks but with inhibited stimuli from landmarks. After t_2 seconds it returns in the normal exploration behaviour. Such a intermediate state is required so that the robot may step away from the landmark (otherwise, even though it is a probabilistic transition, it would join the cluster again in most of the cases). TODO: required?

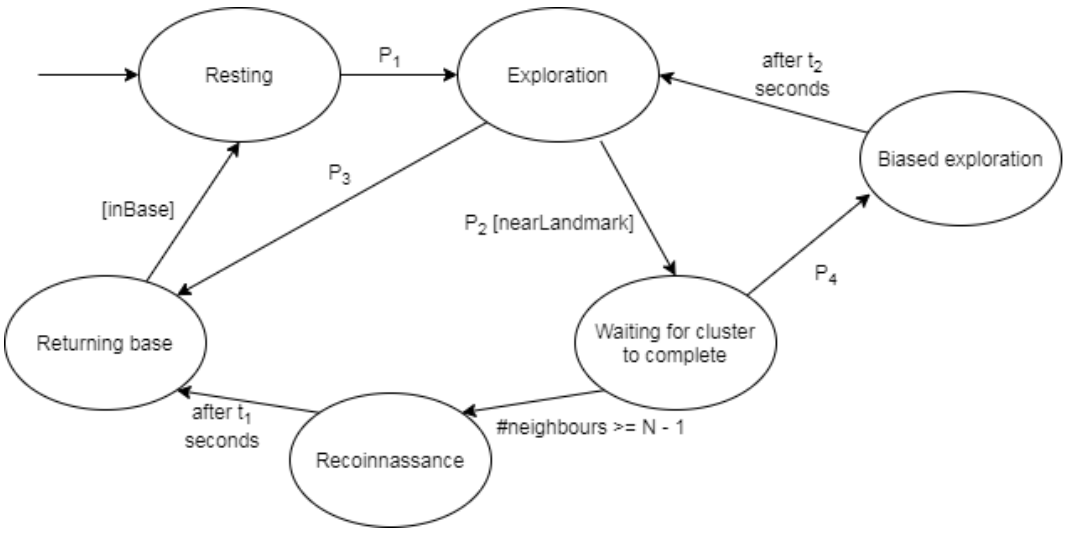

Figure 3.1: *NFA representing the behaviour of the robot. P_i represent a non-deterministic transition that happens with such probability.*

The implementation of the FSA is executed by using a "states" table to store the code be executed in each state. The variable "current_state" represents the current active state, while transition are executed by checking simple"if-then-else" conditions.
In the end, a variable "t" stores the time spent by the robot in the current state.

3.1.1 Motor schema design

When a state requires motion, its design has been developed using a motor schema approach and, once the total force suffered by the robot has been calculated, the following formula is used to switch from the translational-angular model to differential one (i.e., find the velocity of each wheel out from the direction and length of the total suffered force):

$$\begin{bmatrix} v_l \\ v_r \end{bmatrix} = \begin{bmatrix} 1 & -L/2 \\ 1 & L/2 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \tag{3.0}$$

where:

v_l = velocity of the left wheel
v_r = velocity of the right wheel
L = distance between the two wheels
v = translational velocity
ω = angular velocity

3.2 Resting

The robot just checks if a transition to the exploration state should occur, and so no particular architectural style has been adopted for this module. The following function has been used to model the probability:

$$p = tanh((t - Shift)/Patience) + 1 \tag{3.1}$$

The idea is to create a probability that is directly proportional to the time spent in the current state using a non linear relation. Moreover, the probability should grow very slowly at the beginning so that the robot will remain in the resting state for a while.
The function $tanh(x)$ is the basic function used to represent such relation[1] (figure 3.2[2]). The $Shift$ value is constant (500) so we can take the slice of the function having up concavity while dealing with positive time values. The $Patience$ value is a parameter used to tone all values down. In the end, the +1 factor let the function have positive values.
Table 3.1 reports few reference values of the function.

t	p
0	0.013
10	0.015
50	0.022
100	0.036
500	1

Table 3.1: *Few example values of the function used to model dependency with elapsed time using a non linear relation.*

[1] A reasonable alternative would be the exponential function, but it requires the tuning of few parameters so that the "tail" of the function take reasonable values

[2] The figure illustrates the regular *tanh* function instead of the one exposed in formula 3.1 since the latter one, despite having a "similar shape", can hardly be graphically shown.

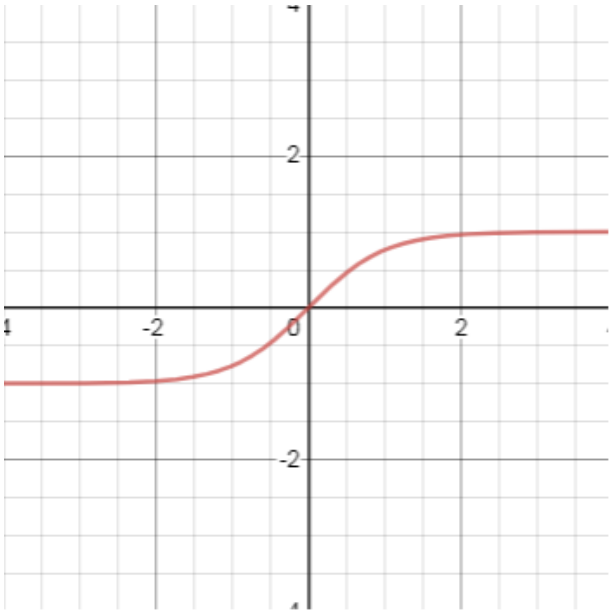

Figure 3.2: *Graph of the tanh function in range [-4;4].*

3.3 Exploration

The robot executes a ballistic motion. A single potential field is created since the obstacle avoidance feature is implicitly given from the random walk itself. The direction of such potential field is randomly chosen each time an obstacle is detected, while its strength is given by a constant value "CRUISE_VELOCITY".

When exploring, the robot has a fixed probability to quit such task. Such parameter has been tuned with a very low vale (0.001) in order to avoid too much segmented behaviours where the robots keep on fluctuating just between the two states of "exploration" and "returning to base".

3.4 Clustering

A robot may join/create a cluster when close to a landmark with a probability described by the following function:

$$p = \frac{(tanh((t - Shift)/Patience) + 1) * tanh(n/NeighborInfluenceLimiter)}{(NeighborInfluenceLimiter * PatienceEnhancer)} \tag{3.2}$$

The probability is proportional to number of robots already in the cluster (n) and to the time spent exploring (t) both. The same formula in 3.1 has been used the time dependence term. The parameter $NeighborInfluenceLimiter$, as the name hints, limits the influence of number of neighbors in the final result.
The denominator is a normalization term so reasonable values can be obtained, and it can be tuned by the parameter $PatienceEnhancer$ that makes the robot wait longer as it is increased.
Figure 3.3 reports the graph of the function in formula 3.2.

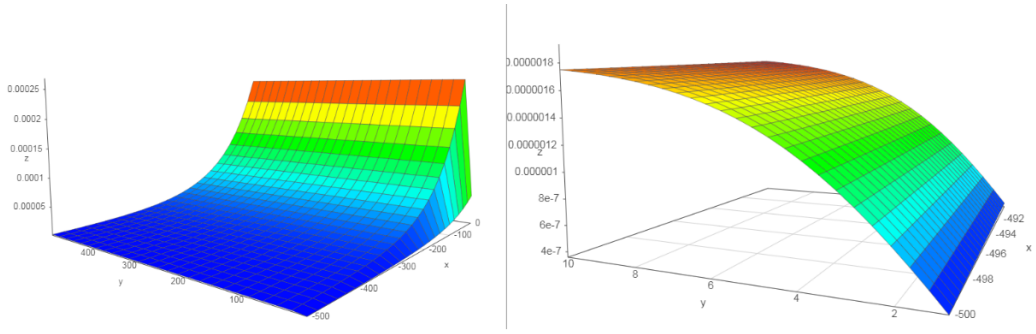

Figure 3.3: *x-axis = time, y-axis = number of neighbors, z-axis = probability. Graph of the plot in domain [-500;0], [1;500] showing the total trend of the function (left) so the time relation can be shown well. On the right, a zoom in in range [-500; -492], [1:10] is executed to show the dependence upon neighbors (indeed, the number of robots in the cluster will never be about the previous order of magnitude (10^2) , and so low values, that cannot be appreciated in the left graph, are the the ones of interest.*

Similarly, the probability for a robot to leave a cluster before it is completed is modeled using the following function:

$$p = \frac{(tanh((t - Shift)/Patience) + 1)/tanh(n/NeighborInfluenceLimiter)}{(NeighborInfluenceLimiter * PatienceEnhancer)} \tag{3.3}$$

Formula 3.2 and 3.3 are exactly the same function except for a sign (division instead of multiplication of the two terms that model relation with time and number of neighbors). Indeed, the robot increases its probability to leave as time passes (direct proportionality) but, at the same time, its probability to leave is decreased as the number of perceived neighbors increases (inverse proportionality)(the more robots, the more probability to complete the cluster soon is the underlying reason that may lay under such behavior).
Figure 3.4 reports the graph of the function in 3.3.

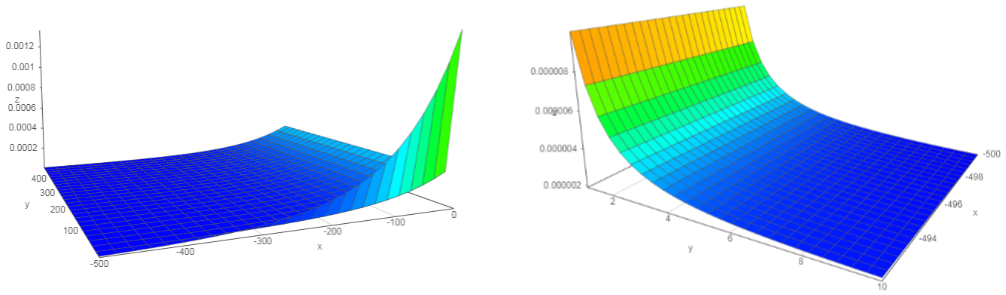

Figure 3.4: *Similal reasonings of figure 3.3. The probability slowly increases as time passes (x-axis) and rapridly decreases as numbe of neighbors increases (y-axis).*

3.5 Returning to base

This state must mixes phototaxis and obstacle avoidance, and so a motor schema approach has been used to easily blend such two tasks.

The phototaxis capability is created by using an attractive potential field produced by the light source. Hence, the resulting force, whose direction guides to robot toward the light source, has a value inversely proportional to the perceived light.

The obstacle avoidance capability is created by using a tangential field around any obstacle so the robot can circumnavigate boxes or other robots. By exploiting proximity sensors, a perpendicular force is created($\pi/2$ from the sensor orientation, and proportional to the proximity). A possible issue is given from the boundary walls that may create local minima in some rare cases. However, such problem may be fixed by introducing a third potential field: a perpendicular one coming from the walls. Moreover, when travelling to base, the robot should not slip into any wall in most of the cases, and so the issue is not of primary importance for the time being.

Chapter 4

Performance evaluation

Few different arena layouts are used by changing the position of the landmarks. We consider three different arrangements by placing the landmarks:

- At the four cardinal points
- At the four corners of the arena
- Aligned on a same side of the arena

The three configurations are reported in figure ?.

For each arena arena, different configurations are probed by tuning the following parameters, namely:

- Number of robots: 20, 30, 40
- Expected cluster size: 1, 3, 5
- Number of obstacles (boxes): 0, 10, 20

The system is evaluated by checking how many landmarks are explored as time passes. A simulation is concluded when time reaches a maximum allowed value (3000). For example, if we have the following values:

[100, 400, 800, 3000]

The first landmark has been explored at time 100, etc. The last one has not been explored during the simulation, and so its value is set to 3000.
We define *goodness* a quantitative metric that estimates how well the system behaves so that we can compare different combinations of parameters in a quantitative way. A given time t is associated to the number of landmarks

that had already been explored in such time. Hence, starting from the four exploration times, we can model the trend of the system along the simulation using a time series. Figure ? reports a clarifier example.

Table 4.1 reports the *minimum guaranteed level of service* (i.e. worst values among all the executed runs) and the goodness of each casuistry.

Robots	Cluster size	Obstacles	Time				Goodness
20	1	0	161	258	343	1081	0.84
20	1	10	300	329	1705	1931	0.64
20	1	20	266	370	863	2771	0.64
20	3	0	559	1231	2730	3000	0.37
20	3	10	3000	3000	3000	3000	0.0
20	3	20	3000	3000	3000	3000	0.0
20	5	0	3000	3000	3000	3000	0.0
20	5	10	3000	3000	3000	3000	0.0
20	5	20	3000	3000	3000	3000	0.0
30	1	0	115	372	414	1032	0.83
30	1	10	234	418	548	2112	0.72
30	1	20	317	623	994	3000	0.58
30	3	0	869	1165	2401	3000	0.33
30	3	10	1449	1962	2981	3000	0.21
30	3	20	1461	2667	2990	3000	0.15
30	5	0	3000	3000	3000	3000	0.0
30	5	10	3000	3000	3000	3000	0.0
30	5	20	3000	3000	3000	3000	0.0
40	1	0	175	236	367	862	0.86
40	1	10	268	410	420	1022	0.82
40	1	20	307	580	1018	2069	0.66
40	3	0	833	834	970	2566	0.53
40	3	10	1542	1633	2394	2453	0.33
40	3	20	1534	2815	3000	3000	0.13
40	5	0	3000	3000	3000	3000	0.0
40	5	10	3000	3000	3000	3000	0.0
40	5	20	3000	3000	3000	3000	0.0

Table 4.1: *Evaluation of the system on 27 proposed combinations. For each casuistry, it is reported the moment of exploration of each landmark. Every value is the worst value among the registered ones. Moreover, the goodness of each combination is reported too.*

Chapter 5

Conclusions

Bibliography

[1] https://link.springer.com/chapter/10.1007/978-3-030-25332-5_19

[2] https://www.hindawi.com/journals/jr/2019/6914212/

[3] https://arxiv.org/ftp/arxiv/papers/1306/1306.1144.pdf

[4] https://www.researchgate.net/profile/Dr_Anish_Pandey/publication/317101750_Mobile_Robot_Navigation_and_Obstacle_Avoidance_Techniques_A_Review/links/59266dad458515e3d45393b3/Mobile-Robot-Navigation-and-Obstacle-Avoidance-Techniques-A-Review.pdf

[5] https://www.hindawi.com/journals/jat/2018/5041401/

[6] http://anderslyhnechristensen.com/pubs/alchristensen_alifex_holeavoidanceandphototaxis_2006.pdf

[7] https://www.hindawi.com/journals/jat/2018/5041401/

[8] https://www.hindawi.com/journals/jat/2018/5041401/

[9] https://www.sciencedirect.com/science/article/pii/S0004370220300047

[10] https://www.sciencedirect.com/science/article/pii/S0925231215010486

[11] https://link.springer.com/chapter/10.1007/978-3-319-21407-8_2

[12] http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.310.7632&rep=rep1&type=pdf

[13] https://www.sciencedirect.com/science/article/pii/S0004370220300047#br0140