Essential robotics swarm algorithms.

Implementation and simulation using ARGoS robot simulator.

Paweł Jakubowski [*]

Intermediate Project under instructions of Witold Paluszyński Ph.D.

Faculty of Electronics – Wrocław University of Technology, 11/17 Janiszewskiego St., 50-372 Wrocław

March, 2016

Abstract

This paper shows implementations of chosen algorithms that corresponds to essential swarm problems. Each of these algorithms was implemented in ARGoS robot simulator – a C++ framework developed within the Swarmanoid project. This work is preliminary research for master thesis and intention was to check if ARGoS simulator will be suitable and relatively easy to use in determining algorithm correctness and if it provide proper tools to perform different kind of experiments.

[*] This work is licensed under a Creative Commons Attribution 4.0 International License.

1 Introduction

Swarm robotics is still very fresh and challenging field of research. Designing and implementing algorithms for multi-agent robot systems is not a trivial task. Many factors have to be taken into account. Main difficulty is to deal with changing or unknown environment – in fact this is an area where swarm algorithms shows real advantage in comparison to traditional single-agent systems and centrally controlled multi-agent systems. In order to mimic swarms that exist in nature we want our systems to be distributed. Only then we can threat our agents as they are redundant – failure of one agent do not cause failure of whole system. What is more, such control allows system to be scalable (if implemented correctly) — this is general requirement for all swarm algorithms.

Although swarm robotics is relatively new field of research we can determine few essential problems that can be solved using swarm. First basic problem, which is also suitable for every robotics system, is movement with collision avoidance. Algorithm does not show any cooperation between agents, but it's simplicity makes it a good reference for future work. Implementation is described in Section 3. Detection and collision avoidance was based on ARGoS `Diffusion 1` example [1]. Second algorithm in this paper is flocking algorithm – i.e. implementation proposed in article [2]. This specific implementation focused on very simplified approach, so it could be used in memory-restricted robots such as nanobots. Flocking is used here as an aggregation technique and implementation is described in Section 4. Third algorithm is Particle Swarm Optimization (PSO) algorithm [4] — well known from optimization field of studies. In this paper only most basic version of PSO was implemented. Results of this implementation are available in Section 5

ARGoS simulator was chosen because it seemed to be very flexible and extensible. It was also important that source code was released under the terms of the MIT license (published as git repository `https://github.com/ilpincy/argos3`), have some documented usages [3, 5] and active support forum.

Development environment consist of Ubuntu 15.10 64 bit machine with i5-2410M CPU and 8GB RAM.

2 ARGoS simulator

According to project website [1] authors motivation to develop this simulator was absence of software that supports large-scale heterogeneous robot swarms. It is based on microscopic modeling – each robot has own physical model, what results in strong coupling of robot entity with it's sensors and actuators. This approach lead to more accurate simulation results. It also allows to simulate heterogeneous swarms, where robots have not same physical models, but this require to deep into unnecessary details if we want higher level of abstraction.

ARGoS's architecture consist of modules (showed on Fig. 1). Each module can be overwritten by user and plugged into simulation as shared libraries. Controller is main module that contain robot's algorithm.

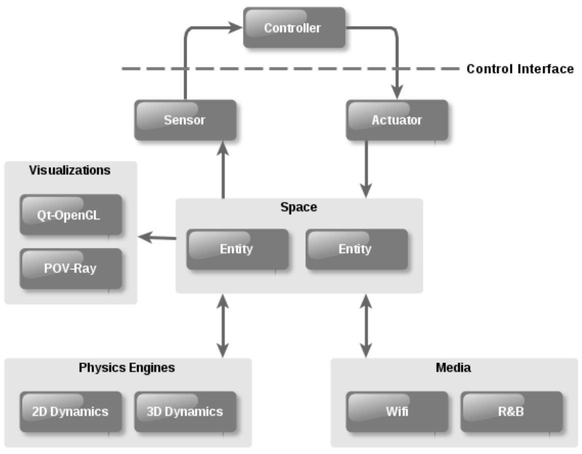


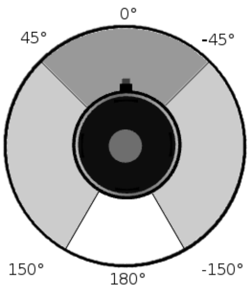
Figure 1: ARGoS modules [1]

Experiment configuration is stored as XML file, typically with extension `.argos`. Configuration determine information about all modules and experiment-specific settings like visualization type, length of experiment or position of cameras.

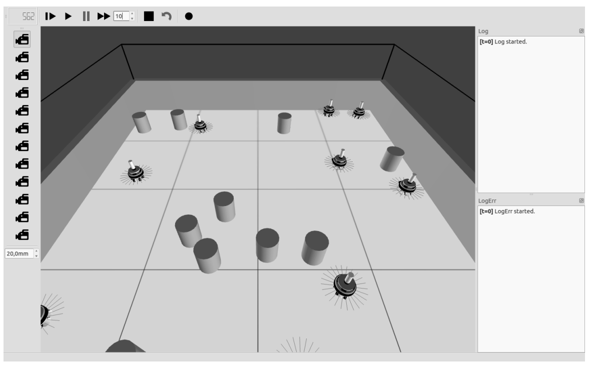
3 Movement with collision avoidance

Movement with collision avoidance is most basic task described in this paper. Whole algorithm was based on ARGoS `Diffusion 1` example [1]. For this implementation footbot robots were used (provided by ARGoS framework).

Algorithm itself was very easy – robot reads values from his proximity sensors and sum all vectors. As a result he knows center of mass of surrounded obstacles. When it detects obstacle it checks whether he is safe or he should turn. Robot consider obstacle not harmful if absolute weighted angle was more than 150 degrees or it was in front of him – between -45 and 45 degrees – but further than minimum safe distance. Robots angles are shown on Fig. 2a.



(a) Robot's angles



[t=0] Log started.

[t=0] LogErr started.

(b) ARGoS simulator with Qt visualization

This algorithm was used as sort of "Hello world" in ARGoS framework. Different setting of experiments was tested and all behaved properly (example on Fig. 2b). Settings that were tested:

- different arena sizes
- different obstacle sizes
- random placement of objects (obstacles and robots)
- passing parameters from XML to controller

It was sufficient to create controller class that implements above algorithm and compile it as shared library. Then properly written configuration allowed to run experiment.

4 Flocking

Algorithm that was base for implementing flocking behavior was introduced in article [2]. Main motivation that drives authors of that article was to design very simple algorithm that can be embedded into very limited robots. Short communication range, small sensor ranges, very limited computational power, little memory, imprecise locomotion — all of these factors was taken into account. Additionally robots do not possess any kind of global information.

Simplicity of this approach was reached by dividing robots sensor fields into sectors. For each sector threshold was assigned and different behavior was performed depending on which sector robot detect object or another robot. Zones and algorithm are shown on Fig. 3.

Implementation

Implementation was not so easy as for collision avoidance algorithm. This time goal was to properly implement algorithm and gather results from experiments. To be able to compare obtained results with those introduced in article [2] environment had to be as close as possible to original implementation. Robots used by authors of algorithm was not available in ARGoS library so there was a need to develop and plug custom robot entity class. It was done by copying e-puck robot implementation and modifying it to match description in article [2].

Number of proximity sensors was changed to 12 – 3 per each section. Maximum range of those sensors was increased to 5 robot-diameters (unit used by authors [2]). Additionally to mimic passive diodes virtual omidirectional camera from footbot was added and LED ring was lighten in red. That way robots can distinguish obstacles from other robots.

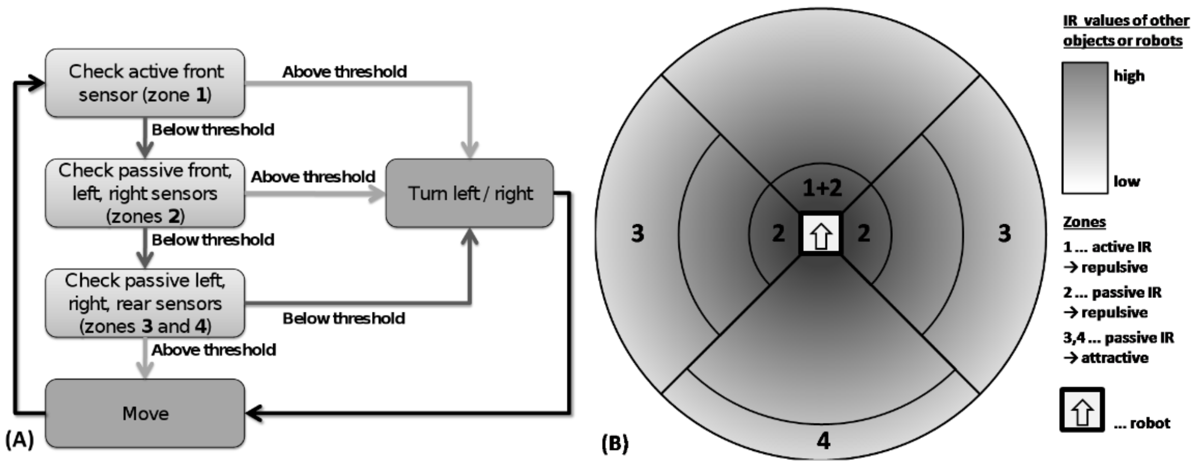

Check active front sensor (zone 1)
Above threshold
Below threshold
Check passive front, left, right sensors (zones 2)
Above threshold
Turn left / right
Below threshold
Check passive left, right, rear sensors (zones 3 and 4)
Below threshold
Above threshold
Move
(A)
(B)
1+2
2
2
3
3
4
IR values of other objects or robots
high
low
Zones
1 ... active IR → repulsive
2 ... passive IR → repulsive
3,4 ... passive IR → attractive
... robot

Figure 3: A: Schematic of minimalist flocking algorithm. B: Robot sections and visualization of threshold levels. [2]

ARGoS do not take path of shared library with robots as opposite to controllers or `loop_functions`. However it loads on start all libraries that are located in one of paths defined by `ARGOS_PLUGIN_PATH` environment variable. It was sufficient to overwrite this variable to point directory with custom robot library and it was loaded like build-in robot types.

Performing experiments

This algorithm was really good test for ARGoS, because it required to perform experiment thousand times to collect statistical data. Only then obtained results could be compared with original results described in article [2].

Turns out that ARGoS can run without any visualization module – so it only compute environment without overhead of visualizing and rendering it. Thanks to that experiments could be performed very fast even though experiment time limit was set very high – 10 000 seconds. Remembering that each control step was called 60 times per second it gives big amount of data that have to be calculated.

Time limit was not goal in this experiment. It should end (successfully) if at least 60% or 80% of robots are aggregated in same flock. For this purpose loop-function module has do be developed (Algorithm 1). It calculates flocks aggregation level so it can determine if any of those flocks contains desired number of robots.

Algorithm 1 Aggregation calculator

```
1: procedure CALCULATE
2:     Set of flocks $F = \varnothing$
3:     for each robot $r$ do
4:         if $r$ is not assigned to any flock then
5:             Add new flock $f$ to set $F$
6:             $f = f \cup r$
7:         for each robot $o$ do
8:             if $o \neq r$ and distance from $r$ to $o < 1$ robot dimension then
9:                 $f = f \cup o$
       return max $f * 100/$ robots count
```

To help with changing parameters of experiments simple bash script was written that used `sed` program to change those parameters.

Results

Results are not quite the same as presented in article [2]. Implemented swarm has bigger tendency to drive along walls and flocks that had more that 2 robots are nearly impossible to move. Only partial results for Density-neutral Efficiency was taken. Only 5 and 10 robots swarms was able to finish task before timeout (Fig. 4). Same as in article [2] results presented first, second and third quantile of obtained results for 1000 experiment runs.

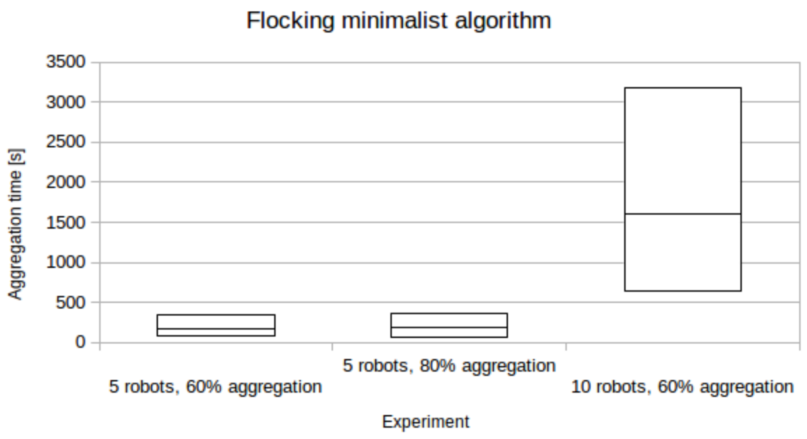


Figure 4: Results obtained for implemented flocking algorithm

5 Particle Swarm Optimization

Particle Swarm Optimization algorithm is one of best known distributed optimization algorithms. It is used to find optimal solution within problem environment. It was designed as optimization algorithm that operates in multi-dimension set. It could be also successfully used to swarm robots search problem in typical 2D space. One variation of this algorithm was described in article [4].

Only simplest version of this algorithm was implemented in this paper. Neighborhood was assumed to be global and for collision avoidance algorithm from Section 3 was used. All parameters are same as in article [4] – that also include arena size and type of robots (e-puck). Control steps was called 10 times per second and synchronization occurred every 100 steps.

Arena was 8 × 8m and contained 20 uniformly distributed obstacles – 10 cylinders 0.2 × 0.25m and 10 boxes 0.1 × 0.6 × 0.3m. On Fig. 6 are shown result of one experiment for 10 robots.

During implementation one problem showed up. Even though ARGoS treats light as normal entity it do not allow to randomly distribute this objects. It was a problem because in chosen implementation light source was used as a target for swarm.

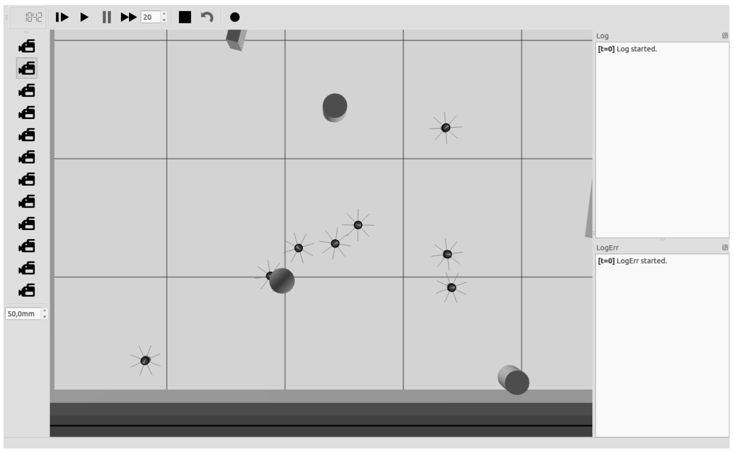


Figure 5: PSO algorithm experiment. Light source was used as a target.

6 Conclusions

Not all implementation provided satisfying results. In flocking one can expect to obtain similar results as those introduced in article [2]. Possible mistake was to introduce more IR sensors (3 for each section instead of 1) and passive sensors approximation in the form of omidirectional camera.

All in all ARGoS proved itself to be flexible swarm simulator that allow to implement and test essential swarm algorithms. Possibility to replacing each module of framework and clear design was very important

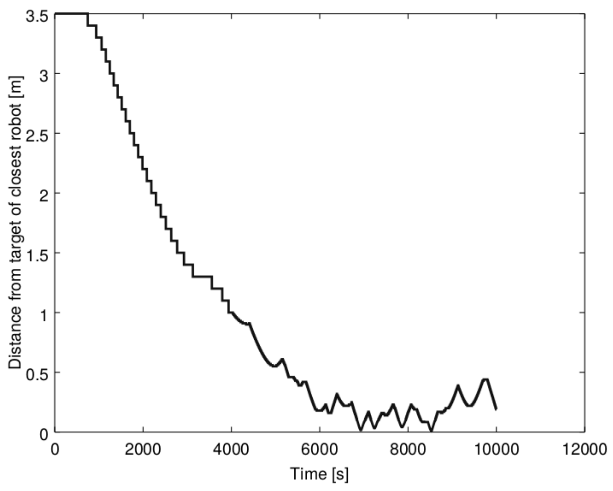


Figure 6: PSO algorithm results for 10 robots.

during algorithms implementation. It was also relatively easy to perform multiple — time consuming — experiments without additional overhead and in reasonable time.

References

[1] The ARGoS website. `http://www.argos-sim.info`. Accessed: 2016-02-05.

[2] C. Moeslinger, T. Schmickl, K. Crailsheim. A minimalist flocking algorithm for swarm robots.

[3] C. Pinciroli, V. Trianni, R. O'Grady, G. Pini, A. Brutschy, M. Brambilla, N. Mathews, E. Ferrante, G. Di Caro, F. Ducatelle, M. Birattari, L. M. Gambardella, M. Dorigo. ARGoS: a modular, parallel, multi-engine simulator for multi-robot systems. *Swarm Intelligence*, 6(4):271–295, 2012.

[4] J. Pugh, A. Martinoli. Inspiring and modeling multi-robot search with particle swarm optimization, 2007.

[5] T. Stirling, ames Roberts, ean Christophe Zufferey, D. Floreano. Indoor navigation with a swarm of flying robots. *Proceedings of the 2012 IEEE International Conference on Robotics and Automation*, 2012.