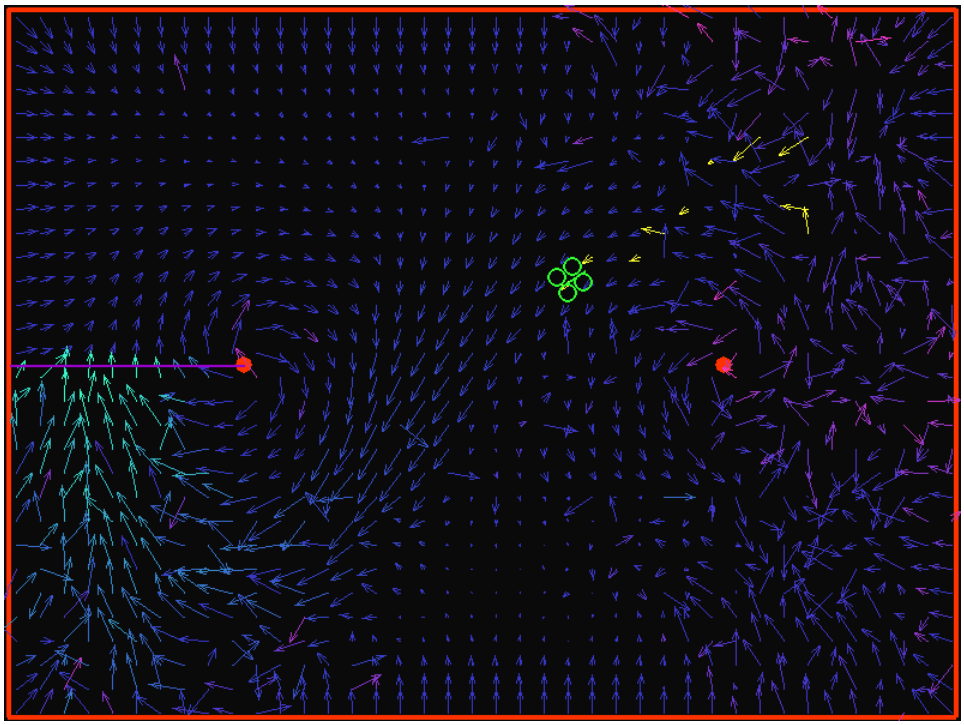
Optimizing Artificial Force Fields for Autonomous Drones in the Pylon Challenge using Reinforcement Learning



Martijn F.W. van der Veen

Optimizing Artificial Force Fields for Autonomous Drones in the Pylon Challenge using Reinforcement Learning

Martijn F.W. van der Veen
5964008

Bachelor thesis
Credits: 15 EC

Bacheloropleiding Kunstmatige Intelligentie

University of Amsterdam
Faculty of Science
Science Park 904
1098 XH Amsterdam

Supervisor
Dr. A. Visser

Intelligent Systems Lab Amsterdam
Faculty of Science
University of Amsterdam
Room C3.237
Science Park 904
NL 1098 XH Amsterdam

July 28th, 2010

Abstract

The idea of imaginary forces acting on a robot has gained a lot of interest from robotics researchers in the nineties as useful and easy-to-set-up paradigm for navigation without collisions. However, the force field essentially is not optimal, leaving room for improvements. A common used method for learning optimal policies is Reinforcement Learning. This thesis tries to apply reinforcement learning to force fields in a way to improve the initial force field. The framework is applied, with some additional improvements, to the "Figure-Eight" task using an aerial robot. Some interesting results follow from the (simulated) experiments, and the possibility of the proposed framework will be evaluated.

keywords:
reinforcement learning, potential field methods, force fields, figure-eight, drones, imav2011

Contents

1 Introduction

People seem to think that the goal of Artificial Intelligence is the creation of autonomous, intelligent robots. One could state that, at least in the field of robotics, it seems indeed to be the *ultimate* goal. Although a lot of progress has been made in the last couple of decades, the ultimate goal still appears far from reached. Different sub-problems ask for different algorithms, and 'the best' framework does not (yet) exists. In the course of time, a lot of algorithms have been invented and are investigated exhaustively, and some became part of the standard toolbox[1]. Since there still are a lot of challenges, existing ideas are being shaped and new algorithms are being proposed, and tested on both existing and new problems.

One challenge within robotics is the simple task of autonomous navigating towards a goal, without hitting obstacles. One of the proposed methods that try to solve this task uses artificial forces to guide the robot [5]. The robot is attracted by the goal, while obstacles spread a repelling force. The approach works, but does not give an optimal path. This leads to the question whether the forces, which look like a good starting point, could be improved by learning.

A well-known approach within robotics for learning is reinforcement learning. This semi-unsupervised learning methodology uses the intuitive notion of reward and punishment received from the environment to learn optimal actions for possible situations (i.e., locations within the environment). The objective of this thesis is to take the forces, or force field, as initial set up, and improve it using reinforcement learning. The improved force field should lead to better paths that cause the robot to move faster to the goal.

The proposed method will be tested in the often-used Figure-Eight (Pylon) challenge using an aerial robotics platform. The objective of the Figure-Eight challenge is to move in figure-eight's around two poles. The tests will be used to gain insight in the benefits and possibly disadvantages of the proposed method.

[1] As an example, the SLAM algorithm, for both mapping and localizing in unknown environments, is being used frequently; http://openslam.org

1.1 Motivation

Artificial force fields, often called Potential Field Methods (PFM) in the literature, are praised for their simplicity, elegance and efficiency [10]. They are simple to set up and give reasonable results with no collisions if the parameters are set up correctly. The force fields are also computationally highly efficient as they only require some simple vector calculus on local obstacles. Furthermore, the physical analogy of a free-flying object makes it more appropriate for free-flying (omni-directional) aerial robots than for the ground robots the method is mostly used for [4].

The figure-eight challenge is a commonly used to test robotic navigation algorithms. It has been used frequently for both wheel-driven and walking ground robots, and in the last couple of years it is used for testing the navigation skills of aerial robots too, principally Micro Air Vehicles (MAV's). For example, the figure-eight task is part of the yearly International Micro Air Vehicle (IMAV) Flight Competition[2]. Using the results of the research on improved force fields presented in this thesis, we will participate in this year's IMAV 'Pylon' indoor competition in September 2011, held in the Netherlands. A Parrot AR.Drone[3] Quad-copter will be used as hardware platform. The AR.Drone is one of the first affordable MAV's and comes with an Open Source API programming interface, making it an excellent tool for robotics research labs.

The rest of the thesis is organized as follows. In the next section some related work will be discussed. In section 3, the used methodology is explained, followed by an explanation of the accomplished tests. The results will be shown in section 5 and discussed in section 6. The last section describes possible future research.

2 Related Work

2.1 Force Fields

The idea of imaginary forces that act on a robot has first been suggested by Andrews and Hogan [1]. They introduce the idea of an attracting (positive) force towards the target and repelling (negative) forces nearby obstacles.

[2] http://www.imav2011.org
[3] http://ardrone.parrot.com

The target is viewed as a point which exerts a constant force towards himself on all locations, while the obstacles exert a repelling force towards the robot inversely proportional to the distance between the obstacle and the robot.

The force field framework, later on called *Potential Field Methods* (PFM), was further investigated in the 90's by many researchers in the robotics area and became quite popular for a short time as obstacle avoidance method. For example, Koren and Borenstein [2] developed the *Virtual Force Field* (VFF) which will discretize the environment and uses a *certainty grid* to represent the certainty of finding an obstacle at each grid position. Obstacles can thus be unknown initially, but it is assumed that the location of both the robot and the goal is known.

Two forces are being calculated in PFM: the attractive force $\vec{F}_t$, and the total repelling force $\vec{F}_r$. The resultant force vector is the sum of both: $\vec{R} = \vec{F}_t + \vec{F}_t$. The direction determines the robot's steering-rate, which is a constant times the difference between the current and the desired direction. This is, of course, based on the idea of wheel-based robots; omni-directional vehicles could simply proceed their path in the desired direction. The new direction is thus determined by the strength (length of vector) of the repelling forces relative to the attracting force constant γ. Note that the length is not important and a constant speed is used.

The Virtual Force Field uses a *Active Window* of grid cells nearby to calculate the vectors $\vec{F}_r$ and $\vec{F}_t$ as:

$$\vec{F}_r = \sum_c F_c = \sum_c \frac{\delta}{d^n(c,r)} \frac{(\vec{P}_c - \vec{P}_r)}{d(c,r)}$$

and

$$\vec{F}_t = \gamma \frac{(\vec{P}_t - \vec{P}_r)}{d(t,r)}$$

where c stands for a cell, r for the robot and t for the target, δ is a value determined by a negative force constant, the certainty of an object being at cell c and the size of the robot, γ a target force constant, $\vec{P}$ the various positions, and $d(c,r)$ the distance between cell c and the robot. Thus, the attracting target force has constant size and points towards the target, and the repelling force is the sum of all cells possibly containing an obstacle, inversely proportional to the distances to the robot. In practice, n is often chosen to be 2.

More extensions have been proposed, such as the *Vector Field Histogram* by Borenstein which calculates directions with a high concentration of free

space, *Navigation Fields* which are potential fields that do not have local minima but are difficult to create, methods incorporating global path planning, and extensions to deal with motion dynamics at high speed movement [6]. This last extension uses a *dynamic window approach* where the size of the active window is changed according to current speed and direction, to take into account the inertia of both the rotation and the translation. For omni-directional robots such as quad-copters, the rotational inertia is not a problem, but the translation inertia could cause problems. However, when learning is involved, possibly wrong initial forces could be corrected. Another extension is the use of a variable attracting force determined by the distance, which could help avoiding the problem of local minima with can originate out of many obstacles near the goal [7].

The Potential Field Methods are reported to be successful in practice [3, 10, 4], both in simulation and real-world application. Some limitations are described too [10], which are discussed in section 6. Later on, more advanced methods were developed that are more proficient for small environments than Potential Field Methods, and the PFM became less used. For larger open areas such as in the "Figure-Eight" task, PFM are still useful. Anyway, the most important objective of the proposed methods is avoiding obstacles, leaving room for more efficient paths. Hence, it is worth examining an adjustment of the potential field method.

2.2 Reinforcement Learning

A well-known method for learning and improving *policies* is Reinforcement Learning [12, p.51-82]. RL uses the intuitive idea of reward and punishment (or negative reward) that follow on actions of an agent. In learning methods concerning supervision, the best action possible will be told to the agent after acting in a certain situation. In practical problems, the best action is often not known, but a certain value of the resulting *state* could be defined. In RL the *reward* of different actions for different states is being used to update either the *value function*, that determines the potential value of each state, or the *policy*, that maps possible observed states to the best known actions. The *reward function* thus defines the goal of the task being accomplished, without specifying exactly *how* the task will be accomplished. The goal of the Reinforcement Learning problem is to maximize the expected (average) total reward over time, called *return*, so to maximize the following function

[9]:

$$lim_{h\to\infty}E(\frac{1}{h}\sum_{t=0}^{h}r_t)$$

or, more often cited [11], to find the optimal policy π^*:

$$\pi^* = argmax_\pi[E(\sum_{t=0}^{\infty}\gamma^t r_t)]$$

where r_t is the immediate reward following the last action and determined by the current state, and γ is some discount parameter to make the infinite sum finite. The interaction with the environment enables the agent to learn optimal actions for each state by choosing the action that has the largest expected return.

Updating the agent's knowledge could be accomplished by using a method involving some form of temporal difference (TD) [12, p.133-157], which updates the value function, or a more exotic method such as an evolutionary algorithm (EA) [11], which updates the policy directly. TD techniques update the values or value function parameters based on the direct reward and the difference between the current and last potential state value, weighted with a learning parameter:

$$V(s_{t-1}) \leftarrow V(s_{t-1}) + \alpha[r_t + \gamma V(s_t) - V(s_{t-1})]$$

or

$$V(s_{t-1}) \leftarrow (1-\alpha)V(s_{t-1}) + \alpha(r_t + \gamma V(s_t))$$

with $V()$ the value of a state, r_t the current reward and α the learning parameter. The update is thus based on the difference of the value of the last visited state and the value of the current state, or a weighted combination of the old expected (long-term) return value and the current one based on current reward and the value of the next state. On the side of evolutionary algorithms, whole policies instead of values are being evaluated and 'evolve' to better policies.

To use the knowledge learned from experience, as well as gather new knowledge, a trade-off between *exploitation* and *exploration* has to be made. In practice, often some percentage of the actions is chosen random and the remaining actions are determined by the value function or policy learned so far.

2.3 Combining Potential Fields and Reinforcement Learning

The Potential Force Method basically is a gradient descent search method directed towards minimizing the potential function. The potential function is defined by summing over the forces from goal to each point, with a descending slope towards the goal and hyperbolic hills around obstacles.

An interesting idea using the gradient descent view of the PFM is proposed in [11], which is the only paper known to us that claims to combine reinforcement learning and potential force fields. The potential field model as stated in the last paragraph shares some features with the reinforcement learning model. Using these similarities they redefined a maze problem, originally stated in a reinforcement learning model, as a potential field problem:

- States with positive rewards become attractive sources
- States with negative rewards become repulsive sources
- Other states have no attractive or repulsive force

Using these rules, they rewrite the RL problem, in fact creating a path planning problem from a optimization problem. Applying the proposed transition in [11], their maze problem consisting of a 25x28 state grid becomes a grid of objects exerting attracting or repelling forces, used for direct reward. With the utility function representing the map given, they learn a policy to navigate through this map. The policy in their case consists of four discrete actions (North, West, South, East) for each grid point.

In this thesis a policy is learned with continuous actions; the drone can fly in any direction and the optimal policy will consist of the optimal directions. The path planning problem is thus being optimized by reinforcement learning. The proposed "Figure-Eight" task will use a big force field with omni-directional free-flight possibilities, instead of small corridors with discrete locations.

3 Methodology

The objective of this thesis is to start with an initial potential field, which is best seen as a force field in this methodology. The initial force field set up, which is a reasonable point to start concerning navigation tasks, will be followed by improvements of the force field using a method based on the

reinforcement learning method temporal difference. First will be explained the initial force field set up including necessary steps to prepare for the next step, which is the learning cycle, where experience in the environment containing the specified problem will make changes to the force field. Then this framework is viewed in context of the "Figure-Eight" problem, with some additional comments on how to incorporate the problem with the force fields.

3.1 Artificial Force Fields

Initial Force Field The Potential Field Method sets a force field for a particular problem. A robot navigating through the environment will 'perceive' forces from different sources, resulting in a total force on each location. To be able to learn a better force field a discretization step is applied where for a set of discrete locations L the corresponding total force is being calculated. The forces with their corresponding location are called *force vectors* and the set of force vectors L a discretized *force field*. Note that this transition is done only once, meaning that dynamically moving objects do not change the force vectors directly. However, reinforcement learning makes room for dynamically changing environments. Nonetheless, the method will work best for static environments.

The discretization could be a uniform distributed field, or more elaborate distributions such as coarse coding or a particle filter. With a particle filter, more particles could be set at locations often visited or with big differences in the value function (i.e., narrow corridors). In our implementation, a uniform distributed field is used.

Three force sources could contribute to the total force:

- **Obstacles** apply a force towards the locations inversely quadratic proportional with the distance, with a maximum range
- **Target** applies a force towards himself with constant strength for all locations
- **Bias** to incorporate important aspects of a specific task, such as preferred rotational direction around a particular object

All partially forces add together resulting in the force vectors. Appropriate settings and possibly added bias forces are problem-specific and some tweaking could be needed, although the exact values do not seem that important

(see also section 5).

Speed update The force vector being used to update the speed is chosen by nearest neighbour. Since a unique vector is being used each time, it is easier to update the *visited vectors* later on than it would have been with a interpolation step.

Wheel-driven vehicles need some time to change the direction of movement. However, the focus lies on aerial drone vehicles (MAV's), which have omni-directional fly capacities. The force vector could thus directly be applied to the drone without the need for turn instructions. For low speed the new speed will be almost identical to the force vector applied to the vehicle, while for high speed movements the inertia could play a role too and the new speed (vector) has a small component of the last speed (vector) too.

In practice, drones often have a maximum speed, preventing the drone from moving so fast that the force field has not enough power to steer the vehicle. Alternatively, an artificial maximum could be set. In addition it seems useful to set a minimum speed, to prevent the drone from coming to a standstill in areas with small force vectors. When the minimum speed is set equal to the maximum speed, the paths will be traversed with constant speed. This turns out to be a natural choice in big open areas, but will be less proficient in problems with small areas where less speed could be necessary.

If the parameters are set up correctly, the drone now moves towards the goal without hitting obstacles. An example task is shown in figure 1, which has a left-rotation bias around one obstacle. Shown in yellow are the vectors being selected by nearest neighbour during the run. As could be seen, the path does lead to the goal but is not very efficient.

3.2 Reinforcement Learning

Exploration A parameter ϵ is set that determines the amount of exploration. During an exploration step, instead of the actual nearest force vector a random force vector is created and used until a next force vector is reached.

Value Update The return function $r(\vec{L})$ returns for the current location $\vec{L}$ a positive value when a goal is reached, a negative value when hitting an obstacle and zero otherwise. To use the reward information for estimating

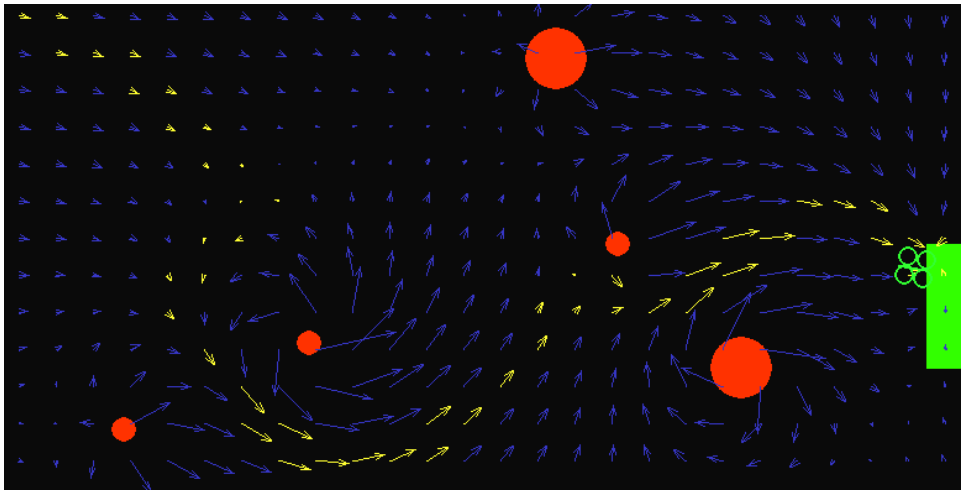


Figure 1: Potential Field, leading a drone towards a goal (simple simulation)

the expected long-term return, each vector is extended with a *(potential) value* V. This value is being used for the expected return value[4], and is necessary for temporal difference used by the vector update.

The value corresponding to the last force vector $\vec{F}_{t-1}$ is updated each time a new force vector $\vec{F}_t$ is being used using TD:

$$V(\vec{F}_{p-1}) \leftarrow V(\vec{F}_{p-1}) + \alpha[r(\vec{L}_p) + \gamma V(\vec{F}_p) - V(\vec{F}_{p-1})]$$

or

$$V(\vec{F}_{p-1}) \leftarrow (1-\alpha)V(\vec{F}_{p-1}) + \alpha(r(\vec{F}_p) + \gamma V(\vec{F}_p))$$

where $\vec{F}_{p-1}$ stands for the last force vector being used (with $p-1$ the previous force vector position) and $\vec{L}_p$ for the current location.

To increase the learning speed, the value update rule could be used backwards for the h vectors in the history.

Vector Update The force vector will be updated as a weighted sum of the force vector and the speed vector that was actually flown[5].

[4] see section 7 on future research for an alternative approach

[5] If the inertia would become stronger on high speeds, one could calculate the force vector that would be needed to fly the speed vector and use the result, instead of the speed vector.

Paths that result in high returns were (obviously) better than paths that result in lower returns. Therefore speed vectors that increases the (potential) value V more should be weighted more than speed vectors that increase V less. Paths resulting in a negative return get a negative difference value, which effectively points the weighted speed vector to the opposite direction.

The vector update follows:

$$\vec{F}_{p-1} \leftarrow (1-\lambda)\vec{F}_{p-1} + \lambda\vec{S}_{p-1}$$

where $\vec{S}_{t-1}$ stands for the previous speed vector, and λ determined by α and the temporal difference:

$$\lambda = \alpha(V(\vec{F}_p) - V(\vec{F}_{p-1}))$$

It is important that the values do not exceed the boundaries -1 and 1 if there is no maximum speed set, since otherwise the force vectors could increase their length to high values not only due to long speed vectors, but also due to high TD and λ weights.

A history of the last H used force vectors could be remembered and used to update a couple of vectors back each time based on the temporal difference between the value of the current vector and the value of each vector h steps back in history divided by the number of vectors h, setting lambda to:

$$\lambda = \alpha\frac{V(\vec{F}_p) - V(\vec{F}_{p-h})}{h}$$

Using this history update faster learning could be obtained with less episodes needed for reasonable results.

3.3 Figure-Eight

The goal of the Figure-Eight task is to move in a Figure-Eight around two poles (figure 2). The task occurs in a static environment and is essentially a 2D problem, although the used method should work equally well in 3D since general vectors are being used in all formulas. We will discuss some difficulties concerning the use of the framework we talked over until now, and show the steps being accomplished to define the "Figure-Eight" task into the framework.

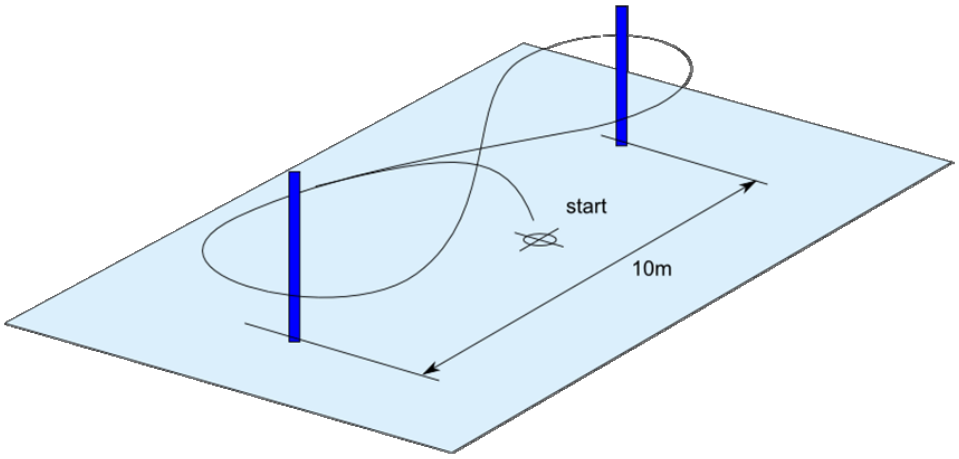


Figure 2: Figure-Eight pylon challenge

Force Fields One difficulties in using a potential field for the "Figure-Eight" task is the continue characteristic of the latter, which makes defining a clear target location difficult. Another one is the center of the field (between the two poles), which sometimes needs to be passed in one direction and other times in the other direction. Both difficulties are solved by introducing two different *stages*, one for 'left to right' and one for 'right to left'. Each stage has its own stage *transition* which is defined as passing the line through both poles at the outside of the corresponding pole (right or left). Each stage has its own force field too, enabling the robot to learn different forces for the same locations, depending on the current location of the target line.

Reinforcement Learning The reward function returns 1 for a state transition and -1 for a collision. The reinforcement learning occurs within stages and is restarted after a transition, as a result of which the values fall from almost 1 to possibly lower than 0 without getting problems with these large negative temporal differences. Repeated training runs[6] will then change the force fields by using the value and vector update rules.

[6]The number of runs needed for round time improvements depend on the number of force vectors, the parameter settings and the desired amount of improvement; for example, the Results section shows round times of 66% of the original time after 300 figure-eight's of training and conservative parameter values.

4 Practical Tests

4.1 Simple Simulation

A simple simulation has been used to view the influence of both different parameters for the algorithm and physics of the environment. Due to the simplicity, aspects of the environment could be changed very quick, although this advantage was at the expense of realistic physics. For example, the simulation updates the speed of the drone by adding the nearest speed vector to the current speed and decreasing the speed over time. Different setups have been tested, with different results. First will be shown a general goal finding task with initial force field, then we will switch to the "Figure-Eight" task for which different parameter values were tested on-the-fly and interesting results will be shown for reasonable (but not necessarily optimal) parameter combinations.

4.2 USARSim simulation

The USARSim environment, based on the Unreal Tournament engine, is a robot simulation environment and used extensively as research tool[7]. The Parrot AR.Drone is ported to USARSim by Nick Dijkshoorn and Carsten van Weelden. Comparison of the simulation and real AR.Drones shows rather good similarity [13]. The physics of the simulation are realistic for both the motions and the sensors. For example, the lighting is realistic enough to use the virtual camera on the AR.Drone as simulation of the real-world camera input.

In this thesis, the USARSim environment is used mostly for more realistic physics (i.e., inertia) compared to the simple simulation. At the sensor part, a virtual sensor detects collisions and another sensor presents the location (ground truth); however the camera is currently not being used. Using the camera it would be possible to detect the poles and other obstacles using SLAM (using for instance disparity maps [8]) or other landmark-based localization or detection systems.

A virtual 3D model of a gym containing two pylons is built and used as environment for the simulated drone. A screenshot of the gym is show in Figure 3.

[7] http://sourceforge.net/projects/usarsim/

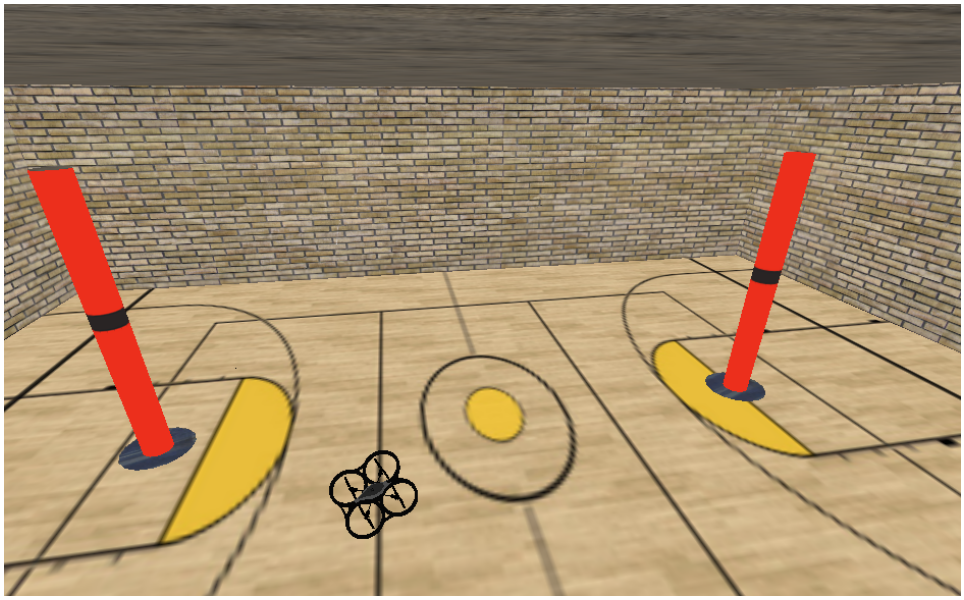


Figure 3: 3D model of a gym with two pylons (USARSim)

4.3 IMAV2011 & AR.Drone

Based on the USARSim simulation it is expected to be relatively easy to switch to a real gym and use the force fields learned so far in simulation to guide a real AR.Drone. A non-trivial step is however the localization [13], since the simulation uses the absolute location (*Ground Truth*) which is not present in real-life. This transition will be necessary for participation to the IMAV2011 competition, but is not discussed in this thesis.

5 Results

The results of force fields are best shown in figures. Interesting aspects will be discussed in the next section.

5.1 Simple Simulation

A classical example of a field with one goal position and some obstacles was shown in figure 1 (section 3). Five circle-shaped obstacles were created with two different radii. All have a (quadratic inversely proportional) repelling force, some have a rotational bias. The goal has a attractive force which is of constant length. The drone is released on random locations not containing an object. In the initial force field shown in the figure, one hundred randomly chosen locations (not inside objects) result in the drone finding the goal. Changing the strength of the different force setting does not really matter for this result as long as no local minima are being created. These could only occur if two repelling forces and the goal-directed force sum to a vector pointing, which could be the case depending on the relative strength of the attracting and repelling forces, as well as the density of the objects.

For the "Figure-Eight" task two circle-shaped obstacles are created (the pylons) and a rectangular-shaped obstacle is used for the walls. Repelling forces are added, as well as a rotational bias for both pylons (one clockwise and one anti-clockwise) to prevent the drone from passing the pylon at the wrong side (flying "Figure-Zeros"). Apparently no attracting force is needed and the drone starts flying figure-eights as soon as it is released somewhere in the force field. Shown in figure 4 is the initial force field with the yellow colored force vectors used in the first run.

After the initial force field is set, the drone is released at the center of the figure-eight. The drone then flies continuously figure-eights. When an object is hit, the drone is restored at it's original starting point with zero speed.

A typical result of the reinforcement learning algorithm applied to the initial force field is shown in figures 5(a) and 5(b). For the learning part, $\gamma = 0.6$, $\alpha = 0.4$, $\epsilon = 0.3$, $H = 10$, 1200 force vectors are used and 250 stage transitions were made (making 125 figure-eights). A constant speed caused the vectors getting equal length. Following the vectors shows a shorter and thus improved path nearby the pylons[8], while still containing trained vectors

[8] At the time these results were saved, the simple simulation did not incorporated the

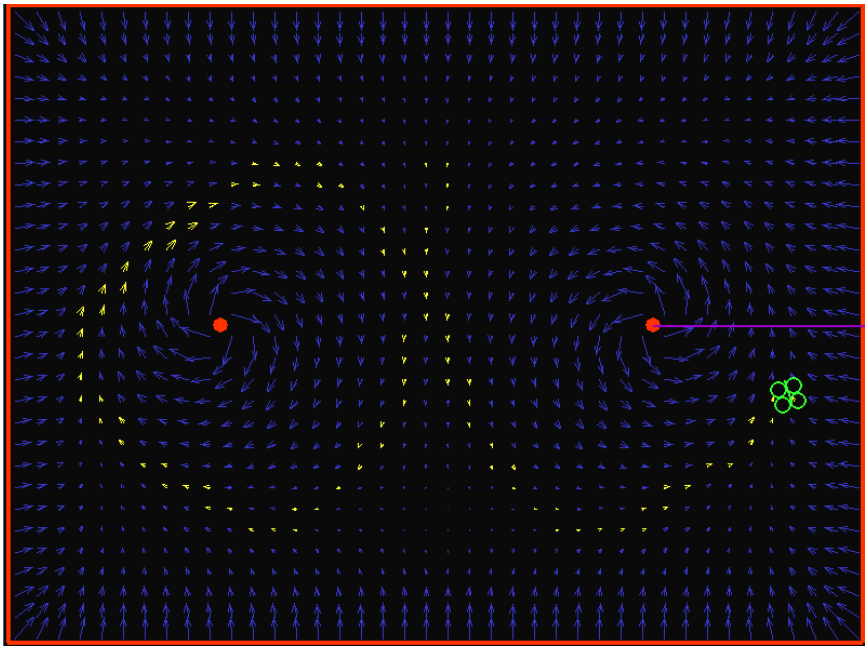


Figure 4: Figure-Eight initial force field (simple simulation)

in wider paths which were reached due to exploration.

The (potential) values for the force vectors are shown as the vector color, with colors ranging from light blue for expected return 1 to light purple for expected return -1.

To test whether there actually is improvement in round time another simulation is run with the (more common) parameter values $\gamma = 0.9$ and $\alpha = 0.2$ (resulting in less extreme vector improvements per round). During learning the time is measured between two stage transitions with a total of 600 transitions (thus 300 figure-eight's). After each one hundred stages of learning, the update algorithm is paused and a test of 40 stages (20 figure-eight's) without exploration is executed. For each of seven tests the mean duration of a stage is calculated. The stage durations during learning are shown in figure 6 (the test runs are thus not shown); the means with the standard deviation of the 40 test runs (which where thus independent of the learning trials) are shown in table 5.1.

size of the AR.Drone, therefore the force field vectors allow for paths at small distances from the pylons

The algorithm seems to have a relatively big improvement at the start. The peeks in figure 6 at the very beginning are caused by the AR.Drone flying an extra circle around the 'current' pole, which seems to stop after 25 trials. After one hundred trials learning seems to progress slowly during the rest of the simulation.

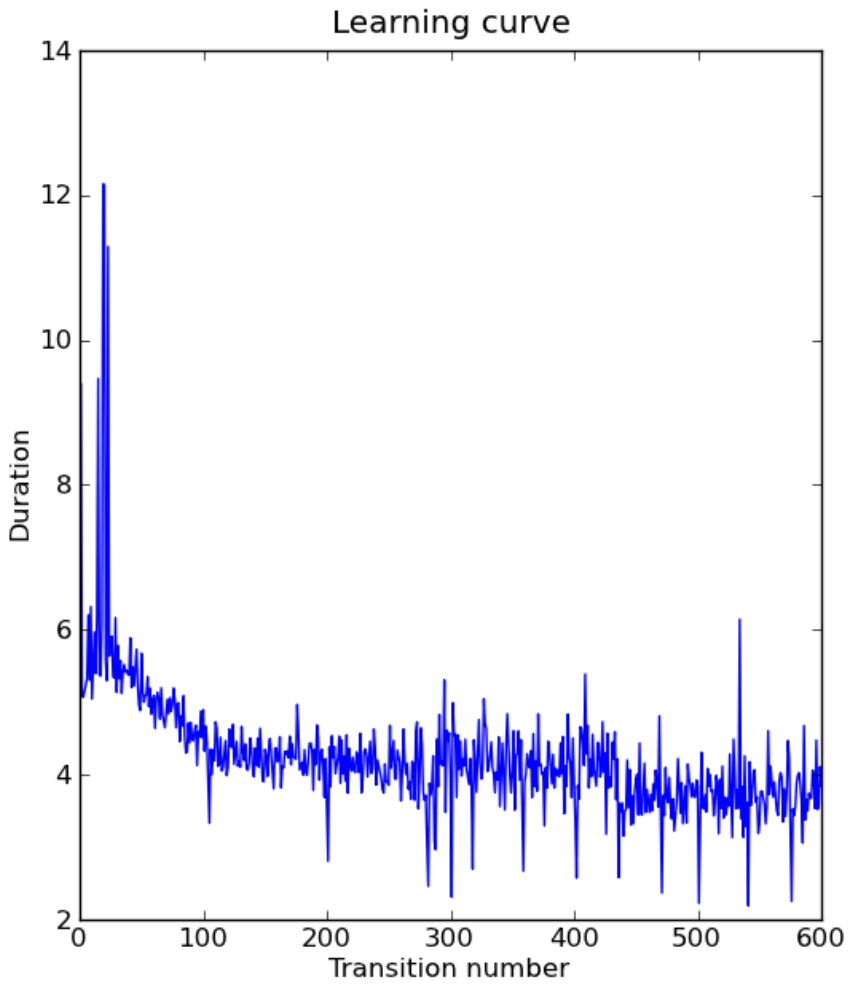


Figure 6: Learning curve (durations of stages during learning)

Training stages	Duration (mean)	Standard deviation
0	4.330	0.0090
100	3.759	0.0261
200	3.442	0.0080
300	3.525	0.1545
400	3.013	0.0528
500	2.985	0.0723
600	2.876	0.1049

5.2 USARSim Simulation

In addition to the initial force field that is set in the simple simulation, an attracting force towards the 'other pole' is set to prevent the drone from circle movements around a pole. In the simple simulation this was not necessary due to a stronger inertia effect, which was too high compared to the USARSim physics.

For the USARSim simulation shown in figure 7 no constant speed is set and $H = 3$, thus updating the last three vectors only, after just 40 stage transitions. This results in a less smoother field with nevertheless interesting parts. One of the interesting structures that originates is the force field on the stage transition line ('finish line'), shown in figure 8. Passing the transition seems to be save somewhere left from the center, where the probability of hitting the wall or the pole is small. Therefore all the (strong and positive rewarded) force vectors in front of the transition point in roughly the same direction, which may be interpret as a point on the optimal path. Repeated runs result in similar force fields after 40 stage transitions.

Some vectors are updated and some are almost untouched. The values are not yet settled and some temporal differences cause vectors to adapt very fast to the current speed. Before locations where the robot will get a negative reward due to a collision (i.e., next to a wall or a pylon), strong vectors originate trying to change the path of the robot.

6 Conclusion & Discussion

The initial force fields set by adding a repelling force for the obstacles, an attracting force towards the goal and possibly some bias forces do quite a good job in leading a robot to a goal without collisions. The precise

parameter values do not seem to have much impact on the goal-finding ability of the robot. However, the path is not optimal and improving the initial field could (theoretically) solve some well-known limitations[10] of the Potential field method:

- **Local minima problem** and the inability to **navigate between narrowly spaced obstacles**, which could both be solved by using exploration to find out the best way out and updating the force vectors
- **Inertia problem** which could be solved by negative reward, causing the possible passed negative vectors (due to inertia) to increase their strength

Reinforcement learning is an intuitive way to solve learning problems where setting up a reward function is easy, whereas the optimal solution is not yet known. Therefore, we tried to incorporate the potential field method and reinforcement learning into a framework for navigation tasks.

The improved force fields show some interesting characteristics, which we will discuss now.

First of all it seems the (potential) values of the vectors are plausible and, after more than a hundred figure-eights, cause a smooth landscape of hills on state transitions and valleys around walls. The values could be updates even faster by using the history of visited vectors and corresponding speed vectors and immediate reward values.

However, before the vectors update to a smooth vector field, hundreds of trials must be made, making the availability of a simulation environment a pre. Some irregularities arise as long as the values are not settled to a stable landscape. Using a constant speed ends in vectors with equal length, while a variable speed gives vectors with variable length; some longer and thus causing areas with fast moving and areas with slow moving.

The duration tests of figure-eight flights during learning as shown in figure 6 show a slowly descending curve. The tests executed between the learning sessions show a descending trend too with multiple times the standard deviation in decrease each next test. The only increase in duration has a standard deviation that is more than the increase. Thus, it seems plausible to state that the learning algorithm as presented in this thesis indeed improves the path of a drone in the "Figure-Eight" task.

The USARSim result seems to show similar initial results, suggesting that the physics in the simple simulation were reasonable except for the inertia

strength. Although experiments with a lot more trials are only shown in both figures and statistically for the simple simulation (which could be accelerated), it is likely, but not guaranteed, to converge to a smooth vector field with better paths as happened in the simple simulation.

7 Future Research

Future Research could focus on two different aspects.

Localization The force field contains force vectors on different locations and needs to know which force vector to use at each moment. In the simulations we used the absolute position, but in real life we don't know this position that certain. A transition has to be made to this uncertain world. For localization it may be enough to use a map of the floor. Since the whole level is in a static, known environment, it is reasonable to presume it possible to present a floor map to the robot. Alternatively this map could be made by the robot by stitching together floor snapshots. In [13] it is shown that a reasonable visual floor map could be created using the AR.Drone if enough texture is present. Another possibility is the use of an algorithm based on SLAM, with the poles as obvious landmarks. Scale is not relevant, as long as the mapping from the map to the vectors is possible.

Another option is using features from sensors like cameras or sonar as abstract input for the state representation, which replaces the location representation. For example, the width of the nearest pole could be enough to both set a reasonable initial force and learn a better force vector identified by that feature.

Mathematical improvements It is yet not known if the proposed method will converge mathematically to the optimal solution ('policy'). Further mathematical work could be done to prove (or disprove) this.

At the moment, each force vector contains an additional value. Both a value update and a vector update exist. It would be mathematically advantageous to incorporate these updates into one update on one data structure. For example, when using a constant speed, the norm of the vector could be used as value, or as the temporal difference. Further research could develop one

coherent update, that may look like or may not look like the potential force fields where we started this journey.

References

[1] J.R. Andrews and N. Hogan. Impedance control as a framework for implementing obstacle avoidance in a manipulator. MIT, Dept. of Mechanical Engineering; American Control Conference, 1983.

[2] J. Borenstein and Y. Koren. Real-time obstacle avoidance for fast mobile robots. *IEEE Transactions On Systems, Man and Cybernetics*, 1989.

[3] R.A. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, 1986.

[4] Wolfram Burgard, Armin B. Cremers, Dieter Fox, Dirk Hahnel, Gerhard Lakemeyer, Dirk Schulz, Walter Steiner, and Sebastian Thrun. Experiences with an interactive museum tour-guide robot. *Artificial Intelligence*, 1999.

[5] H.M. Choset et al. *Principles oof Robot Motion: theory, algorithms, and implementation.* MIT press, 2005.

[6] Dieter Foxy, Wolfram Burgardy, and Sebastian Thrun. The dynamic window approach to collision avoidance. *Robotics & Automation Magazine*, 1997.

[7] S.S. Ge and Y.J. Cui. New potential functions for mobile robot path planning. *IEEE Transactions On Systems, Man and Cybernetics*, 2000.

[8] R. Jurriaans. Optical flow based obstacle avoidance for real world autonomous aerial navigation tasks. Bachelor Thesis, 2011.

[9] L.P. Kaebling, M.L. Littman, and A.W. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 1996.

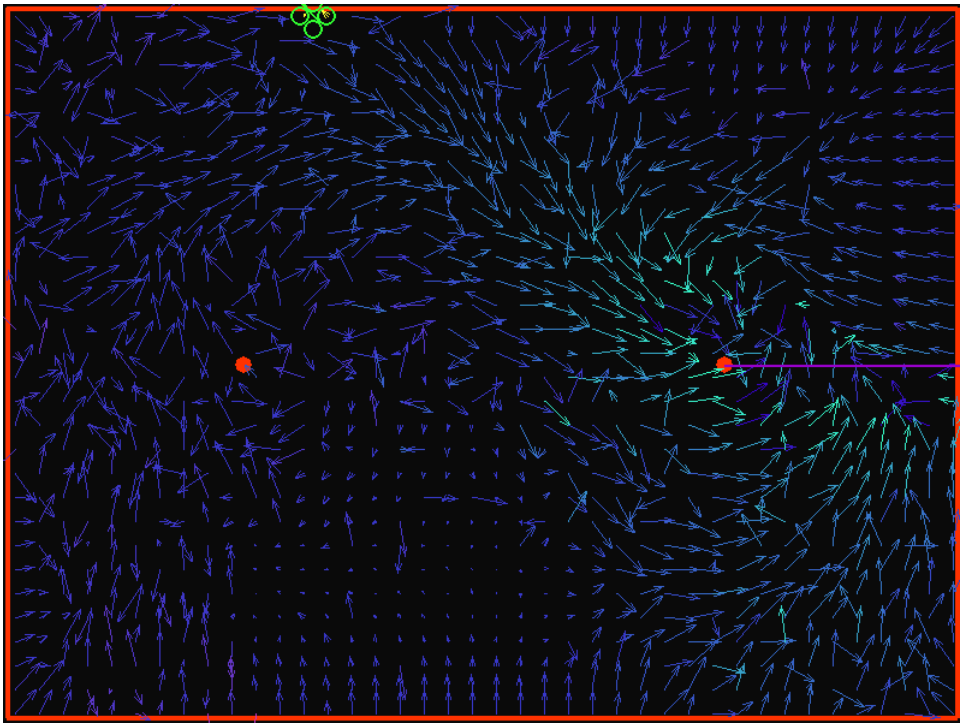
[10] Y. Koren and J. Borenstein. Potential field methods and their inherent limitations for mobile robot navigation. In *Robotics and Automation*, 1991.

[11] XIE Li-juan, XIE Guang-rong, CHEN Huan-wen, and LI Xiao-li. A novel artificial potential field-based reinforcement learning for mobile

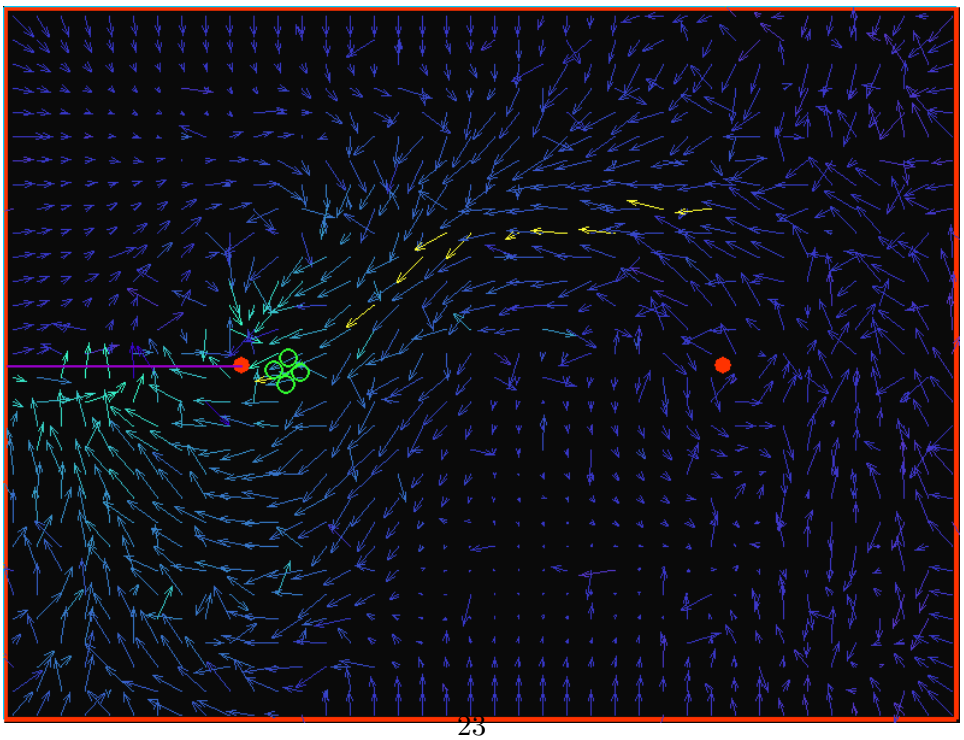
robotics in ambient intelligence. *International Journal of Robotics and Automation*, 2009.

[12] R. S. Sutton and A. Barto. *Reinforcement Learning: An introduction.* Cambridge: MIT Press, 1998.

[13] A. Visser, N. Dijkshoorn, M. van der Veen, and R. Jurriaans. Closing the gap between simulation and reality in the sensor and motion models of an autonomous ar.drone. Accepted for publication at the IMAV2011 conference, 2011.



(a) Left to Right



(b) Right to Left

Figure 5: Figure-Eight after 250 stage transitions (simple simulation)

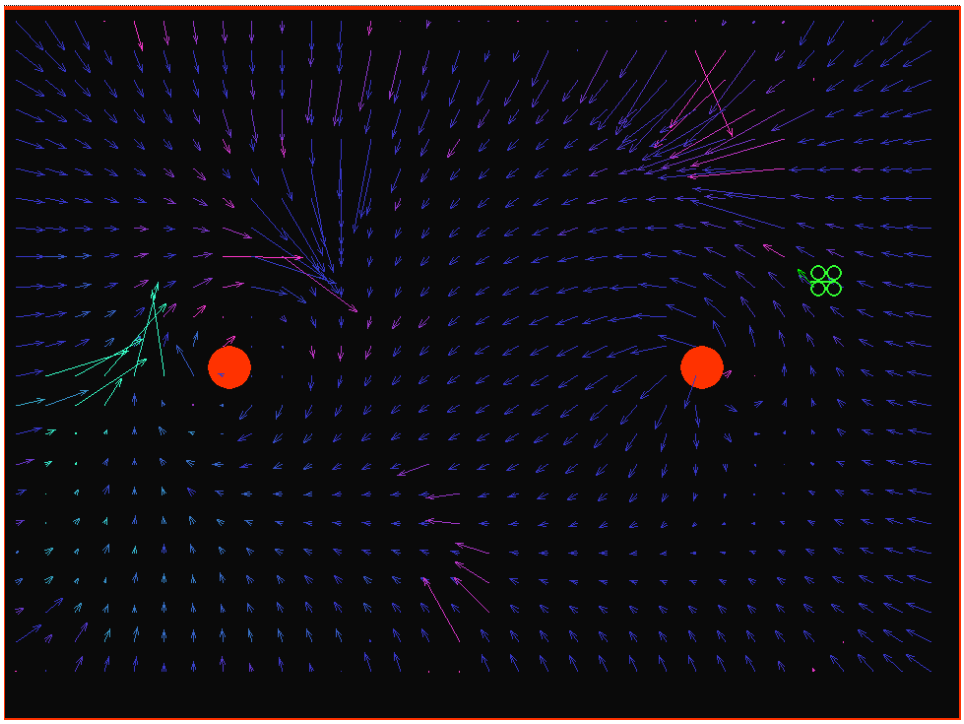


Figure 7: USARSim result

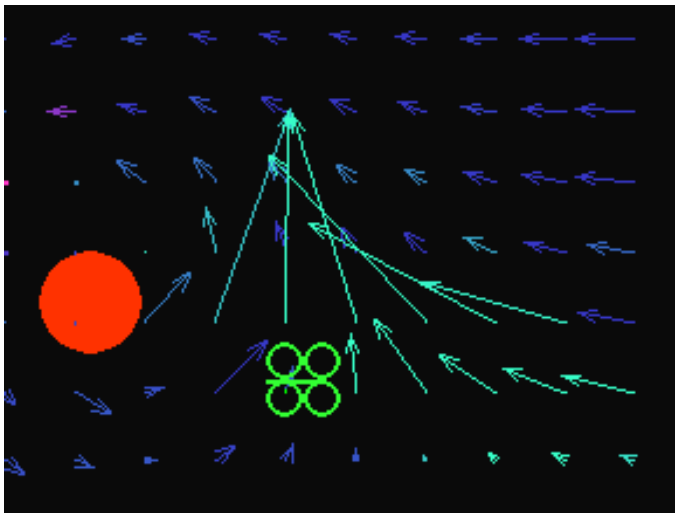


Figure 8: USARSim result, stage transition area