Report 2: Dynamical approach to obstacle avoidance

Introduction

The problem uses the same environment as Problem 1 and poses the same challenge - navigating from starting point A to target point B while avoiding any obstacles on the way. However, in the present task the solution should incorporate a dynamical system which completely governs the behavior of the robot. In other words only the local information from the robot should be used in order to control the heading direction and the speed.

The first step will be implementing a dynamical system with a single attractor - the target location. It will navigate the robot from the starting location to the target location. In order to complete the task we should add obstacle contributions to our dynamical system which will "push away" the robot from obstacles. Additionally, concepts like odometry and interaction with sensors are reused in the same manner as in the algorithmic solution.

Methods

A dynamical system is described by a variable(s) changing over time. The variable that we are interested in is the heading direction of the robot, therefore we need to define our dynamical system over the possible heading directions. The development of the dynamical system over time will give us the heading direction of the robot in specific moments of time which is exactly what we need for navigation. Additionally, using the change in the heading direction we can calculate the speed of the wheels at a specific moment of time.

Dynamical systems express the change of a variables over time through differential equations. The general problem we need to solve is the construction of a differential equation which will produce the desired behavior. Currently we only want a successful navigation to a target, therefore we need to introduce a force that "attracts" the agent to the target and keeps it from orienting away from it. In order to achieve this we should create an attractor in the direction of the target and a repellor in the opposite direction.

Attractors and repellors. The function of the attractors is to pull the system towards them opposed to the repellors which are pushing it away. Attractors and repellors are fixed points in their simplest form - they are zero crossings of the rate of change. Figure 1 illustrates an attractor graphically. The presented system has a zero crossing with negative slope at x_0. If the system is in state x_1 then the change of the system $\dot{x}$ will return a positive value and thus push x(f) towards the zero crossing. If the system is in state x_2, $\dot{x}$ with invoke a negative value and thus push the system back to a smaller value towards x_0. If the system is in x_0, $\dot{x}$ remains unchanged and thus there will be no further changes unless external forces are present. Because of its property to "attract" the system, the zero crossing with negative slope is called an attractor and it is a stable state.

Figure 1 pictures a repellor. As seen in the graphic a repellor is a positive slope zero crossing of the rate of change. Due to the positive slope whenever the system is at value bigger than the x_0 the retrieved values will push the system even further away from the repellor. If the system is at a value smaller than x_0 negative values are returned and thus the system is driven again further away from the repellor. If the system has started in x_0 then it will stay there in any other case it will be always repelled from x_0.

Constructing a dynamical system. As discussed our system should be defined over the heading direction of the robot thus we need a differential equation that generates an attractor in the direction of the target and a repellor in the opposite direction. Hence, the attractor needs to be in direction of the target ψ. We need to turn the robot in direction ψ knowing its current orientation ϕ. Both ϕ and ψ are defined relative to the zero direction in the global coordinate system. Knowing this we can construct the following dynamical system:

$$\dot{\phi} = -\lambda.sin(\phi - \psi) \tag{1}$$

$\lambda > 0$ is the power of attraction. The sine function is more appropriate than a linear one since it is periodic and does not produce increasingly large values even for large angles. Figure 3 shows how a sine dynamic looks like. The attractor will be the heading direction leading to the target and repellor will be located in exactly the opposite direction. Notice that there will be a periodical repetition of the attractors and the repellors.

Calculating speed of the wheels. Having the rate of change of the heading direction we can calculate the current speed needed to complete the estimated change of orientation. From the current change of the heading direction we can calculate the length of the arc corresponding to this angle(for both wheels). This length is essentially the distance that the wheel needs to traverse in order to reach the desired heading direction. In order to finally obtain the speed we should divide by the time needed to traverse this distance. In our case the time will correspond to one iteration of a while() loop which can be obtained through a timer. Alternatively, it is a possibility to provide a hardcoded time value which however may cause inaccuracies.

$$d_r = \Delta\phi.r_w; d_l = -\Delta\phi.r_w \tag{2}$$

$$v_r = \frac{d_r}{\Delta t}; v_l = \frac{d_l}{\Delta t} \tag{3}$$

r_w is the distance between the wheels divided by 2. If we directly feed the obtained speed vector (v_r, v_l) to the robot the resulting behavior will be orienting in one place towards to target(in place rotation). In order to make the robot move

forward while orienting we need to add a constant value to the speed of the right and the left wheel.

```
kSetSpeed(h, v_l+S_CONST, v_r+S_CONST);
```

Avoiding obstacles. So far the robot is able to reach a target location from a given start location and orientation. In order to enable the agent to avoid obstacles we need to introduce a force that is repelling it from any encountered obstacles. Currently $\Delta\phi$ is determined only by the attractor dynamics. Now we need to consider the change of the heading direction as a combination of influences - being attracted to the target and being repelled from obstacles:

$$\dot{\phi} = -\lambda.sin(\phi - \psi) + \sum_i f_{obs,i}(\phi) \tag{4}$$

However, an obvious issue arises with this formulation - the sensors do not return a discrete set of obstacles but proximity estimations in different directions. We can overcome this by saying that each sensor corresponds to an obstacle, this way we obtain a set of 8 obstacles.

The obstacle contribution is based on force-lets and thus the contribution of every individual linear function is weighted by a Gaussian with width σ, centered around the obstacle direction. This way we restrict the angular range of the original linear contribution making it local rather than global. Such a force-let is shown on Figure 4 and is defined in the following way:

$$f_{obs,i}(\phi) = \lambda_{obs,i}.(\phi - \psi_{obs,i}).e^{-\frac{(\phi - \psi_{obs,i})^2}{2\sigma^2}} \tag{5}$$

$\lambda_{obs,i}$ is a weighting function based on the distance to the obstacle - we want only close obstacles to have impact on the heading direction. $\lambda_{obs,i}$ is defined as follows:

$$\lambda_{obs,i} = \beta_1.e^{-\frac{d_i(t)}{\beta_2}} \tag{6}$$

β_1 and β_2 are positive constants and d_i is the distance to obstacle i at the current time t.

Results

The described approach produces a smooth and responsive behavior in contrast to the algorithmic solution. This is clearly visible when comparing the trajectories generated by the algorithmic and the dynamical approach. Figure 5 shows the trajectory generated by the algorithmic solution and Figure 6 displays a trajectory created by the dynamical approach. Starting location and target location are identical in both cases. The dynamical approach seems to produce a behavior which is more similar to the natural dynamics. It is important to mention the smooth change in rotation speed - faster rotation is observed when the robot is orientated far away from the target and the more the robot orients towards the target the slower the rotation gets.

Discussion

Constants influencing the system. As you read through the equations you will notice the presence of some constants which were mostly not discussed in detail so far. However, the proper choice of these constants may have a profound effect on the system. λ_{attr} was said to be the strength of attraction in the attractor contribution. As such this constant has also effect on the speed of the robot and thus some values may cause such rapid turns that the direction of the target may be passed. β_1 and β_2 are part of the weighting function of the obstacle contribution. β_1 is the strength of repelling(again related to wheel speed) and β_2 determines the distance at which the agent will initiate avoidance of the obstacle.

Bifurcations. Consider a situation which includes multiple obstacle force-lets contributing to the rate of change of the heading direction. Suppose that there are two obstacles with moderately large distance in between as shown in Figure 7. In this case the system has two repellors(the obstacles), one attractor laying between the repellors and two more attractor arising on the outskirts. It is so because the force-lets describing the repelling contribution of the obstacles have no or insignificant overlap causing the emergence of an attractor between them. The outskirt attractors correspond to avoiding the obstacles with a sharp turn. In other words all attractors correspond to orientation paths which prevent collision with the obstacles as visible from the image. Now, consider the situation in which the obstacles are close together, leaving not enough space for the robot to proceed between them as shown in Figure 8. In this case the force-lets representing the obstacles have a great overlap which leads to the disappearing of the middle attractor and leaves only the attractors on the outskirts. In this case the two obstacles are basically perceived as a single obstacle. Figure 9 illustrates the stability plotted over the distance between the obstacles thus clearly shows the discussed changes in the fixed points. Such a disappearance/emergence of attractors is called bifurcation and is essentially an instability which causes switch in the state of the system and emergence of behavior.

Figures

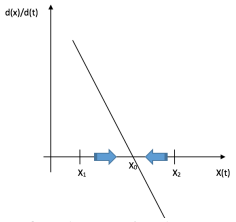

Figure 1. Attractor of a dynamical system

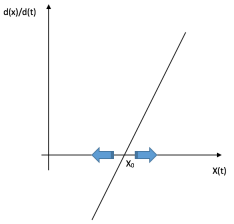

Figure 2. Repellor of a dynamical system

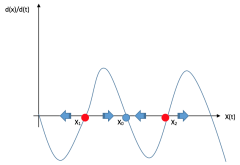

Figure 3. Sine dynamics

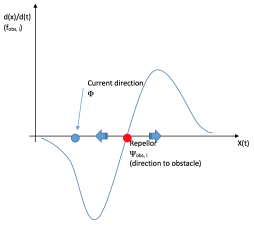

Figure 4. Single obstacle force-let

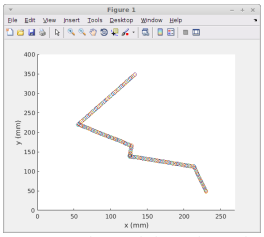

Figure 5. Trajectory produced by the algorithmic solution. Starting location - A3, target location - B2

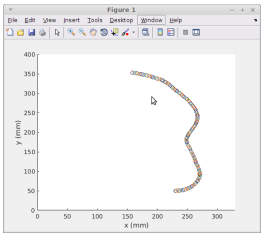

Figure 6. Trajectory produced by the dynamical solution. Starting location - A3, target location - B2

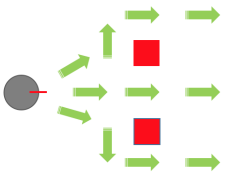

Figure 7. A robot heading towards two obstacles.

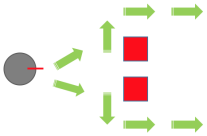

Figure 8. A robot heading towards two obstacles close to each other.

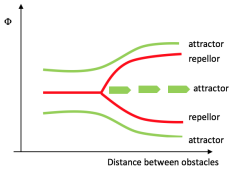

Figure 9. A pitchfork bifurcation.