CS26410 Differences Between Simulation and Reality

Samuel Jackson
slj11@aber.ac.uk

March 14, 2013

There are many differences between the deployment of a robotic controller in a simulator and on a real robot. In the real world, there are many variables which must be taken into account which are outside the control of, or cannot be represented in a simulated environment.

A simulated environment offers an unrealistic description of the world which raises many issues when developing autonomous robot platforms. The major difference between a simulation and reality is that there are limitations to the accuracy and precision that robotic sensors and actuators can work too as well as the complexity and uncertainty of the real world.

One of the key differences between a simulation and actual deployment on a robot is the precision of the sensor equipment. In a simulation, a perfect and precise environment can be modelled where the accuracy and precision of sensors are entirely repeatable, reliable and measurable to a greater degree of accuracy than the real world. In reality, sensor equipment such sonar all have limitations where by the measurement becomes weakened due to attenuation, crosstalk and interference not present in the simulator.

Furthermore, errors can occur in the reading a sensor makes of the environment due to a variety of factors. For example, the Pioneer robots used in the ISL use sonar to sense their environment. These sonar arrays can misinterpret what is actually happening due to reflection of (or lack of) the sonar pulse in a complex environment. This could be caused by surfaces which do not reflect sonar well or by the wave bouncing off at an awkward angle (such as in a corner), or even missing an obstacle entirely (such as a low curb below the sensors field of view) and returning a reading with is not a true representation of the environment.

The degree of accuracy that a sensor is able to read too is also a key difference between a simulation and reality. In a simulator it is easy enough to get a reading of the desired accuracy, however in reality a robot is limited to the accuracy that the sensor can measure too. A good example of this with the Pioneers is that they cannot measure their yaw to anything more accurate than $1\,^{\circ}$.

Another common cause of sensor error is interference. This is an issue that is particularly prevalent when using sonar. An example of this using the Pioneers is that sensors may interpret a sound wave from another sonar device as its own. This is because a sonar sensor takes the first response it can interpret as its own wave returning and has no way of differentiating between one wave and another, giving a false reading of the range. This problem is illustrated particularly well when multiple agents are involved as they can accidentally read the sound waves from another robot, causing them to believe an obstacle is closer than it really is.

When working with a mobile platform such as a Pioneer robot, there are even more variables that come into play. These come from the accuracy and precision that a robot can move to. As an example, Pioneer robots turn and move by counting 'ticks' as their wheels rotate. In a simulator we can turn to a given angle or move to a given point with almost perfect mathematical accuracy. However in reality, we cannot make perfect movements but can only move to within a small error of the desired point. When turning a physical robot, we are not measuring the angle that it has turned, but simply the amount a wheel has rotated, which can be quite inaccurate, depending on the environment and sensitivity and fineness of the tick encoders.

Actuator inaccuracy can be demonstrated by turning a robot at different speeds. Increasing the speed at which a Pioneer can turn also increases the amount of error in the measurement of how much the robot has turned. This means that the robot can think it has turned 90 ° when it has actually turned more. Another problem occurs if the speed is set too low. In a simulator, a robot can turn at any speed, but in reality, the minimum speed is limited by friction. Set the speed too low and the force exerted on the wheel by the motors won't be enough overcome friction and move or turn the robot.

While it is relatively easy to get within a small amount of error of a single point, problems persist due to compound errors in movement causing the robot to steadily loose track of where it thinks it is in its internal coordinate system. As an example, imagine we wish to turn a robot by 90 ° and move forward 5 metres. But our robot can only turn to within 2 ° of the desired angle, meaning that we can undershoot or overshoot the desired angle by as much as two degrees. This error will be compounded as we will no longer be moving straight forward 5 metres at 90 °, but will be moving straight forward at a rate slightly off the desired angle. The more moves made, the more errors become compounded. In the real world, this problem is intensified by external influences such as the tire pressure of one wheel being lower than another or an uneven terrain.

It is also important to remember that robots in the real world cannot perform tasks instantaneously like in a simulator. Imagine a robot moving a speed that sees an obstacle ahead of it. A control signal is sent to set its speed to zero and immediately stop moving. In the real world however, a robot cannot simply go from a moving at speed to a state of rest. It must first decelerate. Taking too long to decelerate will result in the robot overshooting and probably crashing into the obstacle. Another point to bear in mind is that if a robot is controller remotely (such as with the pioneers), then network latency may also play a part in the robot's reaction time.

As a simulation is a representation of an almost perfect environment for a robot, it need never encounter or accommodate for the unexpected obstacles and dimensions present in the real world. A good example of this is would be a robot encountering a shallow hill. It is shallow enough so that the robot does not interpret it as an obstacle to be avoided and therefore reads it as an open space. The difficulty however, comes with the fact that a robot requires more power to climb a hill and therefore will not travel as far as expected when on flat ground. This is hard to account for in a simulator because a simulation only deals with the model of the robot in two dimensions rather than the three.

Issues such as these can attempt to be accounted for by using the integral and derivative parts of a proportional-integral-derivative (PID) controller, where the integral and derivative of a measurement can be used to increase or dampen the force exerted by measuring previous and predicting future error respectively. The Integral can solve the problem of a robot not being able to overcome the force of friction by giving it an extra 'kick' if it has fallen short of the target. Conversely, the derivative can dampen the speed if the current speed predicts an overshoot. All combinations of the three components of a PID controller can also be applied depending on the robot and the type of expected environment.

Localization provides further challenges to a robot in the real world. In a simulator, there is very little chance of a robot losing its way within its fixed coordinate system. In the real world, localization can become a major issue. It is not just compound errors which can cause a robot to lose track of where it is in the world. Consider what would happen if a robot were to be picked up and moved. Any internal concept of where the robot was would be lost as the robot's internal position and yaw would not be updated according to it being moved. The robot would therefore "think" it was in one place when it is really in another. This problem is not so much of an issue in a reactive controller model, but when environment mapping and behavioural controllers start being applied the issue becomes extremely important.

To prevent a robot from becoming out of sync with its internal view of the world, we can use techniques such as Monte Carlo localization or Kalman filters to correlate the location of the robot in the real world within its internal world representation. Alternatively, uncertainty about moves within the environment can be limited to ensure we always end up where we want to be, even with errors causing uncertainty through robust techniques such as fine motion planning.

An additional consideration when dealing with the real world is adaptation to a changing environment. A simplistic example would be trying to map a room with a human or another robot moving about inside. A naive controller would probably misinterpret the moving object as an obstacle and add it as a wall in the map, making navigation more complex. On the other hand, a controller cannot simply ignore moving obstacles all together as they are still obstacles to be avoided. A robot operating in a multi agent environment needs to be prepared for collision avoidance.

In conclusion, there are many different issues that can occur between the simulation of a robot and its deployment in the real world. The primary cause is that a simulation is a simplification of the real world and as with any simplification, you end up ignoring some factors of the world as the only true model of the world is itself. Robots need to be able to plan and reason in away that can react to the unexpected.

References

Owen, J. (2010). *How to Use Player/Stage*, 2nd edition.

Russell, S. J. and Norvig, P. (2003). *Artificial Intelligence: A Modern Approach.* Pearson Education, 2 edition.