Behavioural cloning for robots in unstructured environments

M. Waleed Kadous  
ARC Centre of Excellence for Autonomous Systems  
School of Computer Science and Engineering  
University of New South Wales

Claude Sammut  
ARC Centre of Excellence for Autonomous Systems  
School of Computer Science and Engineering  
University of New South Wales

Raymond Ka-Man Sheh  
ARC Centre of Excellence for Autonomous Systems  
School of Computer Science and Engineering  
University of New South Wales

Editor:

Abstract

Behavioural cloning is an approach where an agent observes the actions of a human operator and then tries to generalise their behaviour. Successful applications include piloting aircraft in simulation (Sammut et al., 1992) and operating container cranes (Urbancic and Bratko, 1994). In this work, we examine the application of behavioural cloning to autonomous navigation of a robot in an unstructured environment.

In particular, we examine the traversal of the random stepfields introduced in the Robocup Rescue Robot League competition using tracked vehicles. One critical issue in behavioural cloning is representing the state of the environment in a manner amenable to machine learning. Our sensing equipment allows a faithful reproduction of the terrain. The representation we use is based on discussions with operators as to how they view the task; and involves breaking up the problem into the general ”lay of the land” around the robot, and the particular obstacles (both protrusions and holes) that must also be taken into account.

Keywords: Behavioural Cloning, Rescue Robotics, Modelling Terrain

1. Introduction

One commonly cited example where robots can be of benefit to humanity is urban search and rescue (USAR). Robots are deployed at a disaster site and autonomously search the area, co-ordinate with each other, deliver assistance to those in need and assist in rescuing survivors. While such a scenario remains unrealised, much research is being undertaken into technologies that may fulfill this goal. The RoboCup Rescue Robot League (RRL) is an annual competition that aims to provide both the motivation and the testing environment, for research related to USAR.

The robot rescue environment is highly unstructured. Because of this, approaches to control the robot have so far focused on tele-operation. However, in real rescue situations,
radio communication is unreliable, tethered robots have limited range and operators have limited attention. Clearly, therefore, autonomy is extremely desirable. But how can such autonomy be achieved? One promising technique is *behavioural cloning*, a process by which the actions of a human operator are recorded whilst the robot is being controlled. This behavioural trace is pre-processed and then input to a machine learning program that outputs control rules capable of driving the robot. Behavioural Cloning has already been demonstrated in tasks such as piloting aircraft (Sammut et al., 1992) and operating a container crane (Urbancic and Bratko, 1994). However, in those domains, the information required to control the systems was relatively easy to obtain. This is not the case for robots in an unstructured environment. In this paper, we propose methods for characterising sensor data in a way that makes machine learning, and behavioural cloning in particular, possible.

In what follows, we describe the rescue environment and the platforms we use for our experiments. We then give some background to Behavioural Cloning before discussing its application to robot control. We conclude with results from preliminary experiments.

2. The Rescue Robot Environment

The task of a rescue robot is to search a disaster site, attempting to identify as many victims as possible in a limited amount of time; gather information as to the state of the victims and map the environment. The annual RoboCup Rescue competition encourages developments in mobility, sensing, autonomy, mapping and human-robot interfaces. An arena is created in which competing teams must demonstrate the above skills. The competition also provides an excellent forum for comparing the effectiveness of different devices and techniques. Parts of the arena from RoboCup Rescue 2005, held in Osaka, are shown in Figure 1.

While most rescue robots are tele-operated, several teams are exploring autonomy. However, these autonomous robots have been confined to areas that might best be described as corridors, with loose paper as the only debris. Very little, if any, autonomy has been
demonstrated in the unstructured areas of the arena where the problem is not so much of navigation but local level traversal.

One particularly challenging element introduced in the 2005 competition was the random stepfield, shown in Figure 1. This is intended as an experimentally reproducible representation of the type of rubble one would find at a disaster site. The blocks are not securely fastened, so the terrain can change based on the actions of the robot. Clearly, analytical approaches for elements such as stepfield are likely to be very complex. Reinforcement learning has some potential but due to the high dimensionality of the state and action spaces, learning rates are slow.

Instead, our approach is to record the actions performed by a human operator who controls the robot as it traverses stepfields and to create a “clone” their behaviour. The intended result is a control policy that can take sensor input and autonomously control the robot to traverse stepfields in general. As with other branches of machine learning, developing a suitable representation is vital for effective performance. To this end, this paper presents preliminary work and early results from an approach for representing terrain, such as random stepfields, for use in behavioural cloning.

3. Background
To understand the proposed approach, behavioural cloning will first be presented followed by a brief survey of existing approaches to terrain representation. The hardware platforms under consideration will also be discussed.

3.1 Behavioural cloning
Experts are people who know what they are doing but don’t necessarily know what they are talking about. By that, we mean that by the time a task becomes second nature to a person, most decision making is subconscious and therefore not accessible to introspection. It is not possible for experts to explain precisely what they do, because there is no way for them to know. Thus, any explanation is usually an after-the-fact justification for why certain actions were taken, rather than an accurate representation of what the person was thinking at the time of the decision.

However, it is possible to construct a model of an expert’s skill by recording the actions of the expert, along with the corresponding state of the system at the time of each action. This was first done by Michie et al. (1990). The method was to record human operators as they controlled a simulated pole-balancer. The log file contained the action, apply a force left or right, along with the position, angle and velocity and angular velocity of the pole. These five variables become the input to a learning program. The result was a decision tree that could be used to control the pole and cart, using a strategy similar to that of the operator.

Subsequent work was done by Sammut et al. (1992) to apply behavioural cloning to learning to fly an aircraft in a flight simulator and by Urbancic and Bratko (1994) to control a container crane. In these cases, it was relatively easy to define the inputs to learning. For example, to learn to pilot an aircraft, we recorded each time the pilot moved the joystick or adjusted throttle or flaps. We also recorded the state of the simulation, corresponding to observing the instruments of the aircraft at the time the action was taken.
For each possible action, adjusting elevators, ailerons, throttle, flaps, we built a decision tree using Quinlan’s C4.5 (Quinlan, 1993). At the end of the learning process we had a decision tree for each action. Thus, the control loop consisted of sampling the state of the simulation, running each decision tree on the state variables and applying the output to the corresponding control variable.

It was beneficial to break the flight into sub-tasks, for example, take-off, climb, turn, etc. However, each stage still consisted of situation action-rules that were specific to a particular mission. These proved to be brittle, in the sense that changes in conditions, such as heavy turbulence, could seriously affect performance. To solve this problem Isaac and Sammut (2003) modified the approach to be more goal-directed. First, the system is taught to perform task-independent manoeuvres. Thus, the right-hand turn procedure can now be given a target heading and turn rate. Once we have a library of manoeuvres, we then teach the system how to achieve a goal setting by applying the appropriate manoeuvre. The control loop now consists of two stages. After sampling the state of the system, determine the goals for this stage of the flight. For example, during the approach to the landing, the goal rules will tell us the desired descent rate. We then invoke the action rules to adjust the controls to achieve the target descent rate. Goal-directed behavioural clones have proved to be considerably more robust than situation-action clones.

Representing the state of the world, when piloting an aircraft, is relatively easy since the instruments in the cockpit provide sufficient information to control the system. Behavioural cloning for a ground vehicle is also straightforward when the vehicle operates in a structured environment. D’Este et al. (2003) demonstrated behavioural cloning on a wheeled robot where the primary input was from a laser range-finder. The environment consisted of a flat floor and walls. However, controlling a ground vehicle in an unstructured environment is much more complex since it must negotiate uneven terrain with different surface textures. The challenge in modelling the operator’s behaviour in the case of our rescue robot has little to do with the learning algorithm. The greatest difficulty is in finding a representation of the environment that provides the appropriate information for learning. By appropriate, we mean that the information must be sufficient to be able to learn control rules but it must also be concise enough not to swamp the learning algorithm with too much data.

Before moving on to describe our representation, it is worthwhile contrasting our approach to behavioural cloning with apprentice learning (Ng et al., 2003). In the latter case, the method is to automatically build a model of the plant from observations of the behaviour of the system and then apply reinforcement learning using the model. Once a control policy has been found, in simulation, it is the applied to the physical plant. To learn to fly a helicopter, a human operator flew the radio-controlled aircraft. Data were collected and used to build the simulator. The goal of the reinforcement learning is to obtain an optimal policy. The key difference between behavioural cloning and apprentice learning is that in the latter, we learn a model of the system, while in the former, we learn a model of the operator. The reasoning is that since we have data of a human controlling the system and the human has already learned a good control policy, why not learn directly from the human?
3.2 Terrain representation

Several groups have addressed the issue of 3D terrain representation in the context of robot control, although most have done so for the application of autonomous road vehicles. Representations such as those used in Talukder et al. (2002) process pointclouds sensed at a particular instant in time and do not make use of an ongoing map. Obstacles are segmented based on their deviation from the driving surface, which need not be flat but is considered to be easily traversable. The obstacles are characterised by parameters such as average slope, bounding box, colour and position relative to the robot. These parameters are used to predict the effect of objects, such as rocks and bushes, on the vehicle.

Alternative representations can, instead, consist of occupancy grid based techniques, with some relying on map generation. Wellington and Stentz (2004) used a 3D scanning laser to detect solid obstacles hidden amongst sparse vegetation by maintaining an occupancy grid. The use of multiple hits/non-hits per cell, taken from several viewpoints as the robot moves around, allows solid obstacles, which are always hits, to be distinguished from sparse obstacles (which are often non-hits).

3.3 Platforms

Two robot platforms are being used for this work. Our first robot, dubbed CASTER, was the basis of our entry in the RRL Competition in 2005, where we came 3rd out of 26 teams. One of our distinguishing abilities was our ability to generate 3D maps of the arena. Figure 2(a) shows an annotated views of the robot. CASTER is built on a Yujin Robotics Robhaz DT3 base (Robotics, 2005). The robot has two pairs of differentially driven rubber tracks for locomotion. The robot is articulated in the center, allowing it to follow terrain such as stairs. The DT3 robot base can move in highly unpredictable ways on unstructured terrain due to its length, suspension and skid-steering properties. Two main concerns are getting stuck on an obstacle (e.g. an object lodging under the body of the robot), or flipping the robot when it attempts to climb an obstacle that is too high.
CASTER is fitted with a wide variety of sensors. These include a pan-tilt unit with a USB web camera, a CSEM SwissRanger 2 time-of-flight range imager, a Thermovision A10 thermal camera and a stereo microphone. Additional sensors include three auxiliary colour cameras and an accelerometer-based tilt meter. The core of CASTER’s mapping and 3D sensing capabilities lies in the range imager (Oggier et al., 2004). Rather than providing colour values for each pixel, this imager provides distances. One limitation, however, is that it only has a field of view of approximately 45 degrees. The pan-tilt unit can be used to overcome this limitation. The range imager is co-located with a colour camera on the highly accurate pan-tilt unit. The addition of an accelerometer to measure tilt and roll enables the production of textured, level 3D reconstructions. This process will be described in section 3.4.

Figure 2 shows Redback, currently under construction. This robot is designed to be lightweight, low cost and flexible, and is based on the MGA Tarantula toy vehicle. It has four treads mounted on two pairs of movable flippers that allow the robot to configure its treads in a variety of ways to overcome different obstacles. This provides a total of four degrees of freedom and enables the traversal of many types of obstacles, at the expense of complex control.

Despite being very lightweight, this platform can carry a variety of equipment, including a small laser scanner on a rotating mount, an omnidirectional camera and a small computer capable of processing these data. Additional sensors include position encoders on the flippers, a magnetometer and an accelerometer for determining pose. This robot must make full use of its high dimensional action space in order to perform effectively.

3.4 3D mapping

CASTER is able to produce dense, textured 3D maps using the range imager and colour camera. Simultaneous captures from these three sensors are combined with measurements from the pan-tilt unit and accelerometer to form a set of data called a “snap”. This may be represented as a cloud of coloured points with known locations in 3D space relative to the robot and horizon. By moving the pan-tilt unit around whilst keeping the robot base stationary, multiple “snaps” may be taken and directly registered, using the pan-tilt unit’s measurements, into a local area map known as a “scan”. An example of a scan is shown in Figure 3.

An alternative sensor package compatible with the techniques in this paper is also being developed, based on a rotating Hokuyo URG-04LX miniature laser scanner. Combined with an omnidirectional camera, this package will also be able to produce 3D textured pointclouds and will be able to trade off resolution for scanning speed.

4. Approach

Initially the goal of our work is to develop a system that will be able to safely and autonomously drive in a straight line over obstacles such as stepfields. Subsequent work will then examine more complex issues, particularly turning on the stepfield. Note that while simplified, driving in a straight line over a stepfield is a highly non-trivial process as the terrain will cause the robot to veer from side to side and at times it may be necessary to
aim off-center to avoid obstacles or to ensure that the robot’s tracks hook onto desired parts of the terrain.

Our approach is to use behavioural cloning. A representation is developed to characterise the situation and actions, after which training can take place. Our main problem is in processing the sensor values to transform them into a feature vector suitable for learning. A training instance consist the feature vector representation of a situation, labelled by the operator’s action in that situation. The learnt concept, which predicts the action given the situation, is called a behavioural clone. This clone is then used for controlling the robot. Previous work has shown that decision trees work well; particularly because they are fast to both learn and execute in real time. We also plan on using the Weka software package (Witten and Frank, 1999) to explore alternatives.

4.1 Action space representation

CASTER only has two degrees of freedom – the left and right motor speeds. Hence the action space is small. While it could be represented as a continuous action space problem, to begin with we plan to represent it as nine possible actions: full forward, full back, left turn forward, right turn forward, left turn back, right turn back, right spin, left spin and stay still. In practice, this combination of actions is sufficient for a human operator to effectively control the robot. In addition, Redback will have the following four actions: rotate front flippers forward and backward and rotate rear flippers forward and backward.

4.2 Situation space representation

Representing the situation consists of two components: representing the state of the robot and representing the terrain around the robot. For both robots, the pitch and roll are measured by accelerometers. These are important attributes, since these can have a particular impact over whether the robot is likely to flip or not. For Redback, it is also important to represent the state of the flippers. As the terrain is expressed relative to the robot, no other robot state variables are required.
To determine the appropriate representation of the terrain, we spoke to human drivers about how they drove the DT3. Strategies for human control tended to focus on two features of the stepfield directly in front of the robot, to a distance of about one robot length. The general layout of the terrain, such as flat, hill or valley would determine overall strategy. Particular obstacles such as blocks or holes that deviate from this general shape would then determine particular actions.

Our representation only considers the rectangular area in front of the robot to a distance of one robot length ahead and one robot width to either side. The learning and control process is incremental so this focus area moves with the robot. Data are taken either from a live feed from the 3D range sensor or, if one is not available, by estimating the robot’s position in a previous scan and recalculating the focus area. The result of this process is a point cloud of the ground, cropped to the focus area described above. This point cloud is then filtered for points such as walls and random noise that don’t form part of the terrain. Suitable filters range from simply ignoring points above the height of the robot to the use of height histograms and statistical thresholds such as picking the minimum in a bimodal distribution or removing the tail of the distribution.

This pointcloud is converted into a height map, where the vertical height above ground level $z$ is a dependant variable in $x$ and $y$, which lie in the ground plane and point towards the right and front of the robot respectively. The height map is broken up into a 3x3 grid along the $x$ and $y$ dimensions and 2D linear regression applied to each portion to obtain 9 planes of best fit, of the form $z = ax + by + c$. The three parameters $a$, $b$ and $c$ become attributes for each portion, giving a total of 27 attributes that characterise the underlying shape of the terrain. Each portion of the height map is examined for points that deviate significantly from the plane of best fit, these are represented in terms of its position $(x, y)$ and height $z$ above the plane of best fit. These may be either holes or peaks.

In total, therefore, the situation space is represented using a total of 56 numerical attributes for CASTER and 58 for Redback.

5. Preliminary results

To study the feasibility of the representation, we considered two random stepfields: one using the USARSim rescue site simulator, and one with data from the 2005 rescue competition. Note that the simulator also simulates the SwissRanger sensor, and thus has some of the same weaknesses, for example, in some cases the tops of blocks are not visible. The results are shown in Figure 4. In each case, to assist the reader, the original image is shown, then the extracted 3D data rotated approximately 30 degrees, then finally a graphical representation of the terrain, with the surfaces representing the planes of best fit in each grid square, and the red lines representing the biggest obstacles in the grid. Fitting the planes was done using Weka (Witten and Frank, 1999). The dot represents the point on the plane of best fit and extends to the obstacle itself.

While not perfect, the results do show promise. For example, in the USARSim data, the protruding block low on the left hand side is picked up as an obstacle (in fact, because it

1. At the competition in Osaka, there were 3 DT3’s that had different sensing, but very similar bases. Thus the strategies for attacking the stepfields tended to be similar.
Behavioural cloning for robots in unstructured environments

Figure 4: Preliminary results with (top row) simulator data and (bottom row) data from Robocup in Osaka. Left is an image of the scene, middle is the the SwissRanger data, and right is a visual representation of the terrain, showing the surfaces as well as the highest obstacles.

In the real data, we can see that the dark grey area off the stepfield on the middle right of the image is picked up as a negative obstacle; and the stepfield post in the middle left of the image is picked up as an obstacle the middle grid square. A driver might therefore choose to drive midway between these two obstacles.

At the time of writing, we are still developing the representation and have not yet conducted machine learning trials. Given that there are 58 attributes, many examples may be needed before the clones are effective. However, the data can be gathered passively from practice and competition runs with no intervention required by the user. In a standard 10-minute run, we expect to collect several thousand actions.

6. Conclusions and Future work

This work is still at an early stage and considerable future work lies ahead. Although not trivial, driving forward over a stepfield is one of the simpler operations possible. Practical experience has shown shown that changing direction on a stepfield is a significantly more difficult task, but one that is necessary for effective navigation. Even more challenging behaviours include choosing suitable actions for Redback to climb over more complex obstacles such as boxes the same height as the robot. Goal-directed behavioural cloning may be appropriate for such cases, where the operator indicates a particular goal to the system and then demonstrates to the robot how that goal can be achieved. This goal may then be used by higher level programs such as planners.
References


