

Autonomous Traversal of Rough Terrain Using Behavioural Cloning

M. Waleed Kadous, Claude Sammut, Raymond Ka-Man Sheh

ARC Centre of Excellence for Autonomous Systems

Computer Science and Engineering, University of New South Wales, Sydney, Australia

waleed@cse.unsw.edu.au, rsheh@cse.unsw.edu.au, claude@cse.unsw.edu.au

Abstract

Behavioural cloning is a method for acquiring skills by building generalised behaviours based on the observations of human performance. 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, by a tracked vehicle. One critical issue in behavioural cloning is representing the state of the environment in a manner amenable to machine learning. Our representation and behavioural cloning technique was evaluated by implementing it on our CASTER advanced mobility robot and training it to traverse a sequence of two NIST-specification stepfields. Despite being derived from only ten training runs, the cloned behaviour was able to successfully traverse the stepfield unaided 40% of the time and required only minor intervention the remaining 60% of the time.

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 distant, research towards this goal continues.

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, some degree of 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 [1] and operating a container crane [2]. 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 the environment in a way that makes behavioural cloning possible. We evaluate the proposed

approach on simulated rubble: a NIST specification stepfield. Our results show that even with minimal training examples (traversing approximately 20 metres of rubble), the learnt clone can perform well – completing autonomous runs over difficult terrain 40 per cent of the time and requiring simple interventions in the remaining 60 per cent. In one case it outperformed the human operator who trained it.

2 The Rescue Robot Environment

The annual Robocup Rescue Robot League (RRL) 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.

One particularly challenging element introduced in the 2005 competition was the random stepfield, which consists of 121 wooden blocks of varying heights (between 4.5cm and 36cm) arranged in a grid. 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.

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” of their behaviour. The intended result is a control

policy that can take sensor input and autonomously control the robot to traverse stepfields. As with other branches of machine learning, developing a suitable representation is vital for effective performance. We propose a representation that facilitates the application of standard supervised classification techniques.

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.

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.

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 can be processed to create a controller that mimics the behaviour of the expert. The controller is thus a "clone" of the expert, and can be constructed by using machine learning to predict the action based on the state.

This approach of *behavioural cloning* is a form of learning by demonstration [3] that tries to model the behaviour of a human operator rather than modelling the system. Consistently with behaviour-based robotics [4], it eschews development of an explicit model of the world: we simply model the users' actions based on the sensor input. Distinctively however, the behaviours are learnt by observation rather than by being programmed.

This approach was first done by [5]. The method was to record human operators as they controlled a simulated pole-balancer. Subsequent work was done by [1] to apply behavioural cloning to learning to fly an aircraft in a flight simulator and by [2] 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.

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. 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* [6]. 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?

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 [7] 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.

Other approaches that attempt to segment out interesting features from raw pointclouds include [8] where a statistical approach was used to classify sections of pointclouds taken from an autonomous vehicle in a forest environment and used to detect such obstacles as trees and wires.

Alternative representations can, instead, consist of occupancy grid based techniques, with some relying on map generation. [9] used a 3D scanning laser to detect solid obstacles hidden amongst sparse vegetation by maintaining an occupancy grid.

[10] also use an occupancy grid based technique, with stereo vision on a planetary rover used to build an occupancy grid map of the terrain around the robot. This is then analysed for terrain "goodness" by observing the roll, pitch and roughness of patches of ground comparable in size to the robot.

4 Approach

4.1 Platforms

For these experiments, we used the robot CASTER, which was the basis of our entry in the RRL Competition in 2005, where we came third out of 26 teams. One of the distinguishing features of our entry was the ability to generate 3D maps of the

arena. CASTER is built on a Yujin Robotics Robhaz DT3 base [11]. 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) and flipping or rolling the robot when it attempts to climb an obstacle that is too high.

The core of CASTER’s mapping and 3D sensing capabilities lies in the range imager [12] and a web camera. Rather than providing colour values for each pixel, the range imager provides distances. The addition of an accelerometer to measure tilt and roll enables the production of textured, level 3D reconstructions.

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 the simplest form of behavioural cloning, known as “situation-action” behavioural cloning. This consists of three stages: the gathering of training data, building of the clone and execution of the clone.

During training, at each step the situation is viewed by the operator. The operator selects an appropriate action which is then executed. The operator is presented with a new situation for the next step. This continues until the task is complete. During this phase, the situation, in the form of an image from the range imager, and the roll and pitch of the robot, are recorded along with the operator’s action taken in response to that situation.

The clone is built by first extracting features from the recorded situations, in this case the range images, to form a vector that numerically represents the situation, amenable to machine learning. The corresponding actions become class labels and the problem treated as a supervised classification task. Standard machine learning algorithms can then be applied to produce a classifier that can determine the class – in this case an action – corresponding to the feature vector from an unseen situation.

Finally, the clone is executed by sensing the situation, extracting the representative feature vector from the sensed data, again the range image, applying the classifier built in the previous stage to determine the appropriate action, then performing the action which is then executed. This process repeats for each time step until the task is complete.

4.2 Representation of the Problem

We represent the actions CASTER can perform as one of those typical of a vehicle: forward left, forward, forward right, spin left, spin right reverse left, reverse and reverse right. In practice, this combination of actions was found to be sufficient for a human operator to effectively control the robot.

Representing the situation consists of two components: representing the state of the robot and representing the terrain around the robot. The state of the robot in this case involves the roll and pitch.

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 types of 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.

The terrain is captured using the range imager, which is pointed at the terrain immediately in front of the robot. After processing, this generates a cloud of points that lie ahead of the robot as shown in figure 1.

The pointcloud obtained from the range imager is converted into a height map, where the vertical height above ground level z is a dependant variable in x and y . To represent the terrain, a plane is fitted to the points using multivariate linear regression. This gives an equation of the form $z = ax + by + c$. Thus, a indicates the tilt of the terrain around the robot’s roll axis and b the tilt around the robot’s pitch axis. This gives a general characterisation of the terrain and the way in which it is leaning. The values a , b and c become features in the situation description.

Once this plane is fitted, holes and protrusions can be detected. The space in front of the robot is divided into a 3×3 grid. Each of the grids is then divided into a 10×10 subgrid. Within each subgrid square, the average z value is computed amongst all points bounded by the x and y limits of that subgrid square. The subgrid square which has the maximum absolute difference between the

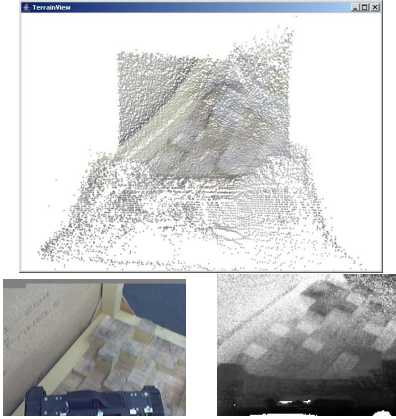


Figure 1: A pointcloud generated by the range imager (top) and the corresponding colour and range images (bottom).

average z value and the z value predicted by the plane of best fit for the centre of the subgrid square is found. In a sense, this tries to find the biggest obstacle (hole or protrusion) in that grid square. The x and y indices of the subgrid square with the maximum absolute difference become features for each grid square. This captures information about where the biggest obstacle within each grid square is. For example, if there is a large protrusion on the right hand side of grid square $(0, 1)$, the x index would be 9, whereas if the obstacle was on the outer edge of the stepfield, the x index of the biggest obstacle would be 0. In addition, the sign and magnitude of the obstacle is used as a feature. Thus, for each of the 9 grid squares, 3 features are generated.

Situation-action clones are essentially Markovian or reactive: they make the decision as to their actions based solely on the current state. This means that the robot has no concept of “getting stuck”, and it will continually repeat the same action in the same situation, no matter how many times this has occurred. In order to prevent this, features were added that attempted to capture some information about change over the last interval. The technique is based on comparing consecutive frames from the range imager and measuring an average change per pixel. This is more robust to changes in lighting and slight movement of the robot than the visual camera, since fine texture changes will produce large average differences between pixels. This difference was also thresholded at approximately 1.5cm/pixel and a count generated of the last time this threshold was exceeded – this count was used as an indication of the number of frames during which the robot had been “stuck” for. This allows a distinction in the situation space between the actions taken by the operator when the robot first becomes stuck and the actions taken to extricate

the robot, even if the situation has not otherwise changed, while maintaining the simplicity of the situation-action approach.

Thus, in total, the situation is represented by 3 features representing the general situation of the terrain, 27 features representing obstacles in different grids in front of the robot, 2 features representing the tilt of the robot and 2 features representing how long and how much the robot has moved since the last action. In total, 34 features are used.

5 Evaluation

5.1 Experimental Setup

To evaluate the approach, CASTER was modified for the purposes of the experiment. Rather than the camera head being mounted on a pan-tilt unit as usual, it was mounted on the end of a robot arm and pointed downwards to maximise resolution of the sensors over the area of interest. A photo of the configuration is shown in figure 2. The view from the robot arm captures the front part of the robot as well as the area immediate to the sides and in front of the robot as shown in figure 1. The cameras sit at an angle of approximately 15 degrees from straight down and is approximately 1 metre above the ground.

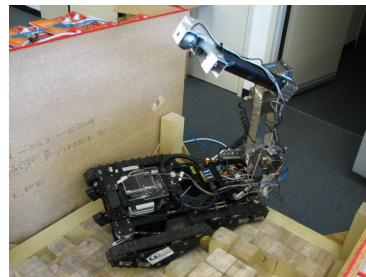


Figure 2: CASTER traversing the stepfield. Note the position of the arm.

To test the techniques, a sequence of two stepfields of official NIST construction were prepared. They were configured to be similar in difficulty to the Robocup 2005 competition stepfields.

A user interface was developed that allowed the operator to look at the scene from the top-down perspective. The operator was instructed to drive the robot over the stepfield as quickly as possible. We chose an experienced operator for these experiments who had approximately 50 hours of driving experience.

Training runs, during which a human operated the robot, were started with the robot approximately lined up with the center of the stepfield. The operator, based only on the scene observed through the range imager and colour camera, chose the direction to travel in. The robot would proceed

along this path for one second. The operator would then have to make a choice again. This constituted a single step of the operation. The experiment would terminate when the operator reported that only saw carpet on the screen, which would occur when the robot had completed traversing the two stepfields.

For each step, the images seen by both the camera and the range imager, the pitch and the roll of the robot and the action performed by the operator were recorded.

The stepfield was traversed 8 times in this manner to gather sufficient data for training (one run was not completed due to technical difficulties mid-way through, but the data up until the point was recorded). A typical run, including set up and run time, would take approximately 15 minutes. Two additional datasets were also collected that involved the robot being placed in a number of unusual and potentially dangerous situations (such as very close to an obstacle or at an unusual angle) and the operator manoeuvring the robot back to a safe state. Other research in behavioural cloning has shown that such “difficult problem” training can be an advantage. Since a clone may not be identical in its behaviour to the expert, it may end up in situations that were not similar to the training data were it collected from only “normal” runs. Actions of the operator in less frequent but still important “difficult situations” can therefore improve performance significantly in such tasks.

Once the data was collected, feature extraction was conducted as discussed. This was then processed using J48 from Weka [13] to produce a decision tree. In order to increase the amount of data for training, the data was mirrored along the front-to-rear axis of the robot. For each training example, the terrain data, the roll and the action were mirrored. If the action had no direction element (e.g. forward), it was set to the same value, while, for example, a forward-left became a forward-right. This served to increase the number of examples, while simultaneously ensuring that the robot had no conceptual bias to veer to either the left or the right.

The robot, as for the training cases, was started approximately in line with the center of the stepfield. The robot was then allowed to continue operating until an observer decided it was irrevocably stuck, or it reached the other side of the sequence of stepfields. The operator was not involved in the operation of the robot except to start the robot and, when necessary, determine that the robot was stuck and stop the robot. When the robot became stuck, an “intervention” was recorded and

Table 1: Performance Operator (Op), Behavioural Clone and Always Forward (AF)

Approach	Runs	Steps	Intervns	Success
Op	7	75.7	0.14	0.86
BC	5	71.8	0.6	0.40
AF	5	45.5	2.0	0.00

the robot moved the minimal amount required such that it could proceed safely.

In order to provide a baseline for comparison, this method was also compared with a default “always forward” action. This is because empirically it was observed that 67 percent of actions taken by the operator were to drive forward.

5.2 Results

A total of 707 actions over the eight normal and two “difficult” runs were collected. 67 per cent of these actions were instructions for the robot to drive forward (and hence our baseline “always forward” action seems to be well-founded). In comparison to the massive situation space (34 dimensions, many of them continuous) and the complexity of the task, this is a very small training set.

Table 1 shows a comparisons of the different techniques. The obvious measure of success is the number of times that the stepfield was successfully traversed. As can be seen below, the human operator was able to traverse the stepfield 86 per cent of the time – the task is sufficiently difficult that even the human operator got the robot irrevocably stuck at one point. The clone was able to complete the traversal 40 per cent of the time. It required only three interventions over the 5 runs and each of these interventions only required minor repositioning of the robot.

The “always forward” controller failed to complete any traversals, and required constant interventions. It required an average of 2.0 interventions per run and many of these were due to the robot becoming dangerously close to rolling as it collided with the sidewalls of the stepfield. Clearly this approach is impractical, and significantly worse than behavioural cloning.

The clone appears to have acquired some of the characteristics of the human operator.

5.3 Empirical Observations

Several interesting and anomalous behaviours were observed and noted. Firstly, the best run (in terms of having zero interventions and minimal steps) was actually performed by one of the clones – it completed the stepfield in 48 steps with zero interventions, while the best human did so in 53 steps. The operator’s comments on seeing it perform was

that “it drove like I would have!” Secondly, the robot would also do things that to a human would appear “strange.” One of such strange actions is that the robot would sometimes drive forward, then drive backward, then drive forward again. It must be remembered that the clones being built here are purely reactive or Markovian. While the SR differencing technique helped prevent it from repeating actions when stuck, it did not help the robot getting stuck in a loop between two actions.

6 Conclusions and Future work

The experiments reported here show that behavioural cloning is a promising approach to building a controller for locomotion over rough terrain. For simplicity, these initial experiments adopted the simple situation-action model of behavioural cloning.

Previous application of behavioural cloning in piloting aircraft have demonstrated that generalisation can be improved by decomposing the controller into two components. The first determines the appropriate values for “goal variables” and the second determines the actions required to achieve those goal values. Future experiments with locomotion will investigate methods for problem decomposition that will achieve similar improvements in robustness.

Clearly one possibility is to study the efficacy of different learning algorithms such as support vector machines and naive Bayes as possible clones. It is also worth noting that having to use a propositional learner because of the noisy and numerical data that exists in this problem domain imposes some difficult limitations, such as the fixed 3×3 grid representation of the world. Obviously driving in this domain would benefit from using a relational representation of the obstacles and their spatial relations to one another, and thus relational learning could be useful. Unfortunately, relational learning techniques are not usually capable of handling noise and numerical data of this kind.

Finally, it is not clear whether cloning for one particular stepfield generalises to other terrains and stepfields. We therefore plan to collect training data from multiple stepfields and then test on new unseen stepfields.

7 References

[1] C. Sammut, S. Hurst, D. Kedzier, and D. Michie, “Learning to fly”, *Proceedings 9th International Conference on Machine Learning*, Aberdeen pp 385-393 (1992).

[2] T. Urbancic, I. Bratko, “Reconstructing human skill with machine learning”, *Proceedings*

of the 11th European Conference on Artificial Intelligence, Amsterdam pp 498-502 (1994).

[3] C.G. Atkeson, S. Schaal, “Robot learning from demonstration”, *Proceedings Fourteenth International Conference on Machine Learning*, Nashville, pp 12-20 (1997).

[4] R. Brooks, “A robust layered control system for a mobile robot”, *IEEE Journal of Robotics and Automation*, 2(1) pp 14-23 (1986).

[5] D. Michie, M. Bain, J.E. Hayes-Michie, “Cognitive models from subcognitive skills”, in M. Grimble, S. McGhee, and P. Mowforth eds, *Knowledge-base Systems in Industrial Control*, Peter Peregrinus pp 71-99 (1990).

[6] A.Y. Ng, H.J. Kim, M.I. Jordan, S. Sastry, “Autonomous helicopter flight via reinforcement learning”, *Advances in Neural Information Processing Systems 16*, Vancouver (2003).

[7] A. Talukder, R. Manduchi, A. Rankin, L. Matthies, “Fast and reliable obstacle detection and segmentation for cross-country navigation”, *Proceedings IEEE Intelligent Vehicles*, Versailles pp 610-618 (2002).

[8] N. Vandapel, D. Huber, A. Kapuria, M. Hebert, “Natural terrain classification using 3D ladar data”, in *Proceedings IEEE International Conference on Robotics and Automation 2004*, New Orleans pp 5117-5122 (2004).

[9] C. Wellington, A. Stentz, “Online adaptive rough-terrain navigation in vegetation”, In *Proceedings IEEE International Conference on Robotics and Automation 2004*, Versailles pp 96-101 (2004).

[10] S. Singh, R. Simmons, T. Smith, A. Stentz, V. Verma, A. Yahja, K. Schwehr, “Recent progress in local and global traversability for planetary rovers”, In *Proceedings IEEE International Conference on Robotics and Automation 2000*, San Francisco, pp 1194-1200 (2000).

[11] Yujin Robotics, Robhaz DT-3 Robot, http://www.robhaz.com/about_dt3_main.asp, Visited 18/08/2005.

[12] T. Oggier, M. Lehmann, R. Kaufmann, M. Schweizer, M. Richter, P. Metzler, G. Lang, F. Lustenberger, N. Blanc, “An all-solid-state optical range camera for 3D real-time imaging with sub-centimeter depth resolution (SwissRanger)”, *Optical Design and Engineering. Proceedings of the SPIE*, 5249, pp 534-545 (2004).

[13] I.H. Witten, E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*, Morgan Kaufmann (1999).