Virtual World Modeling for Software Robot using Neural Networks

Piperakis Romanos†, Suguru Saito*, Takahashi Hiroki†, Nakajima Masayuki†

*Graduate School of Computer Science and Engineering
†Faculty of Precision and Intelligence Laboratory,
Tokyo Institute of Technology

Ookayama Campus: 2-12-1 Ookayama, Meguro-ku, Tokyo
Nagatsuta Campus: 4259 Nagatsuta-cho, Midori-ku

E-mail: romax@img.cs.titech.ac.jp, suguru@pi.titech.ac.jp,
{rocky, nakajima}@img.cs.titech.ac.jp

Abstract: A software robot in the scope of this paper is regarded as an autonomous intelligent agent situated in a virtual reality environment. As such, it is of fundamental importance to be able to create an internal representation or model of its surroundings. This model must gather the relevant to the robot scene information in a compact representation, which will subsequently be used to accommodate the robot's functional requirements. The model is also required to be incremental, in order to be able to grow and adapt to environmental information as it becomes available, and it must allow for incomplete or fuzzy knowledge to describe parts of the scene that have not been available to the robot's sensors. We propose a neural network based model along with a training technique that succeeds in meeting these requirements.
1. Introduction
The choice of an environment model is of great importance to the successful and efficient operation of a software robot. In this paper a software robot is taken as an autonomous intelligent agent situated in a virtual reality environment. This means that the model we are interested in creating is the topological description of the virtual world the agent dwells in. If there is going to be any benefit from the use of such a model, it is required that that model should possess certain well-defined properties. Probably the most important property of all is compactness. The robot will often need to manipulate the model in order to carry out its functional repertoire and thus it is necessary for that model to be small in size even if it is detailed. Naturally this requirement renders polygon representations of the scene particularly unappealing. Of course there are many proposed compact geometrical models, such as [1], [2] and [3] to name a few but another equally important property they fail to accommodate is being incremental. By this we mean that information from the robot’s sensors must be integrated into the model as soon as it becomes available rather than creating a new model every time. This implies that the model should be adaptable, in other words plastic to changes and that the mechanism that is responsible for the models adaptation should be able to append new information to the existing, preferably without significant impact on the model’s size. Finally the fact that the environment topology is explored through the robot’s sensors and is not known a priori, suggests that there will often be parts of the virtual scene that the robot has not yet experienced and thus doesn’t know about. The model should therefore be able to cope and support this ambiguity. In other words it must be able to deal with fuzzy or incomplete information and to generalize by giving a probabilistic measure of the geometry it doesn’t have enough information about.

2. Neural network architecture
The notions of adaptability, fuzziness and generalization all point towards a connectionist implementation of the model [7]. In this paper we have chosen to use probably the simplest and most well known connectionist architecture, which is a multi-layer, feed-forward, back-propagation neural network [6] shown in figure 1.

![Multi-Layered, Feed-Forward, Back-Propagation neural network architecture](image_url)

Figure 1 Multi-Layered, Feed-Forward, Back-Propagation neural network architecture

The inputs to this network are the virtual world coordinates of the 3D scene in which the agent is moving and the squares of these coordinates. Even linear combinations of these terms can give rise to many different kinds of geometric shapes such as planes, parabolas, ellipsoids etc which combined at the output layer serve as building blocks for
the composition of the scene. The output is a real number in the \([0,1]\) domain and is usually interpreted as the probability of the point tested, of belonging to a solid scene object. Consequently a value of 1 signifies that the point belongs to an object and a value of 0 that the point is in empty space. Intermediate values can suggest fuzziness, a degree of doubt or a probabilistic measure of proximity to an object. Clearly this type of representation is volumetric since it describes every point in space and not only the surface bounds of the objects. Experiments conducted with this type of network have lead to the conclusion that the compactness of such a geometric representation is immense. See [5] for details.

3. Software robot sensors
Since the focus of the current work is not that of computer vision or image processing, the software robot implemented was equipped with a z-buffer sensor. This type of sensor enabled the robot to acquire information only from the visible part of the scene for every given viewpoint and orientation. These z-buffer values where subsequently converted to depth information by inverting the perspective transform. Thus the robot at each position and orientation inside the virtual world, had access to a partial surface-type view of the scene.

4. Training process
The training process comprises of two steps. First the points that were read from the robot sensor are processed to create several internal and external surfaces to the objects contained in the current view. This serves twofold. It drastically reduces the amount of training points used for the current viewing volume and it assists the neural network in generalizing beyond what is explicitly trained. In figure 2 we can see a prototype object used for training and in figure 3 a set of external surfaces generated for two oppositely placed viewpoints.

Figure 2 Prototype object used for training

Figure 3 External surfaces generated by the training process to reduce training set and enhance generalization

The distance of the extra surfaces is taken to increase exponentially as we move from the object towards the viewpoint and decrease exponentially as we move away from it. This means that the internal surfaces are much closer together than the external and thus the
The probability of points of the internal surface not actually belonging to the object is minimized while at the same time achieving the maximum generalization outside the object. The training set consists only of points belonging to these surfaces and not the entire viewing volume. During the training sessions points residing on the external surfaces are trained with an output value of 0, points on the surface of the object with a value of 0.5 and points on the internal surfaces with a value of 1. This is in accordance to the interpretation of the output mentioned above. The initial weights and thresholds are taken to be random and the training set points are randomised before being presented to the network for training. The average error of a typical training session follows the graph of figure 4.

Figure 4 Graph of the average error by the trained epochs for a typical training session

It must also be noted that the training algorithm used for all the examples in this paper is the generalized delta rule with momentum.

5. Experimental results
Various 3D objects and simple 3D scenes have successfully been trained using the methodology described in the previous section. In figure 5 we can see the prototype object used for training, reconstructed from the neural network model. The red points are points where the output of the neural network was greater than 0.5 or put another way, the probability of object existence was higher than 50%. In the same manner on figure 6 we can see a zoomed area of the reconstructed model where we can observe that incomplete training can cause artifacts to arise near the actual surface of the object.

Figure 5 Reconstructed object from the neural network model of the prototype hand object. The red points are points with output value greater than 0.5.

Figure 6 Close-up on the trained object reveals artifacts due to incomplete training.

In a different example shown in figures 7 and 8, where more training was
allowed to take place, we can observe that the artefacts disappear.

![Figure 7: Zoomed portion of another prototype object](image1)

![Figure 8: Reconstructed object using the neural network model trained with 3 viewpoints from the prototype object](image2)

6. Conclusion
The experimental results prove that using neural networks for modeling 3D objects can be both compact and accurate. Furthermore, such a representation can deal with ambiguity and can integrate geometric information in the same model as it becomes available. Its adaptation rule is simple and well understood [9] and as an analytical function it possesses properties that can be useful in many applications for tasks ranging from navigation [4] to matching [8]. Its main drawback is the convergence speed but different learning algorithms or starting the learning from a neural network that already represents an approximating polyhedron of the scene can possibly help to overcome that. Due to these properties such a model can be argued to be a very good candidate for modeling the virtual environment of a software robot or intelligent agent in general.

7. References
4. E. Piperakis, I. Kumazawa, Affine Transformations of 3D Objects Represented with Neural Networks, 3rd International Conference on 3D Digital Imaging and Modeling, 2001 (accepted)
5. E. Piperakis, I. Kumazawa, R. Piperakis, 3D Object and Light Representation with Neural Networks, Neurocomputing, Elsevier, 2001 (submitted)