Available online at www.sciencedirect.com

ScienceDirect

IFAC PapersOnline CONFERENCE PAPER ARCHIVE

IFAC PapersOnLine 51-13 (2018) 485–489

Non-linear system modeling using LSTM neural networks

Jesús Gonzalez, Wen Yu

Departamento de Control Automático
CINVESTAV-IPN (National Polytechnic Institute)
Mexico City, Mexico (yuw@ctrl.cinvestav.mx)

Abstract: Long-Short Term Memory (LSTM) is a type of Recurrent Neural Networks (RNN). It takes sequences of information and uses recurrent mechanisms and gate techniques. LSTM has many advantages over other feedforward and recurrent NNs in modeling of time series, such as audio and video. However, in non-linear system modeling normal LSTM does not work well(Wang, 2017). In this paper, we combine LSTM with NN, and use the advantages. The novel neural model consists of hierarchical recurrent networks and one multilayer perceptron. We design a special learning algorithm which uses backpropagation, and backpropagation through time methods. We use two non-linear system examples to compare our neural modeling with other well known methods. The results show that for the simulation model (only the test input is used and the past test output is not used), the modified LSTM model proposed in this paper is much better than the other existing neural models.

Keywords: neural networks, Black-Box identification, LSTM

1. INTRODUCTION

System modeling is very important to predict the future behavior of the dynamic system; to apply model based control and to understand the physical process. Systems modeling applications are found in all areas of science. Many systems cannot be represented by linear models. Non-linear models are valuable resources; however, it is not easy to obtain their model structure and parameters. A convenient method is Black-Box modeling, which only uses input and output data. The model structure (linear or non-linear) is no longer an issue.

Neural network (NN) is a Black-box modeling tool. Thanks to its characteristic of universal approximation and automatic learning NN has gained great strength in system modeling. The example in (Kumar, 2014) shows that the system modeling with NN can predict solar radiation. Although NNs show very good results for the modeling of non-linear systems, they still have problems that must be solved. There are many hyper-parameters that should be defined before they are applied, such as how many hidden layers and nodes are needed. The training processes are based on the Gradient Decent algorithm (Narendra,1991). These cause the local minima problems. A multilayer neural network can model continuos functions (or systems) provided the number of the hidden nodes is large enough. However, increasing the hidden neuron number has the over-fitting problem (Srivastav, 2014).

An alternative method is to increase the hidden layers instead of hidden nodes. It is the basic idea of the deep neural networks (Hinton, 2006). The theory and application show that it can also achieve a desired training accuracy, and avoid the above local minima and structure determination problems (Bengio, 2007). In recent years, deep learning has achieved impressive results in many difficult tasks. For example, the convolutional neural networks (LeCun, 1998) can extract high level features from images to improve the classification accuracy to 99.17%, and is higher than humans can achieve (Krizhevsky, 2012).

Recurrent Nneural Networks (RNN) can represent time series, audio, video, anything that is presented by means of data sequences. In the sequence data, the present values depend on their past values, for example, the sentence and the sound wave. One of the most import recurrent neural networks in deep learning is the Long-Short Term Memory network (LSTM) (Schmidhuber, 1997). It the output of the NN feeding back to the input which allows modeling data sequences or chains of information. Since the training algorithm of RNN needs the backpropogation through time this has a Vanishing gradient problem. LSTM uses several gated units to avoid this. LSTM has been widely applied in many areas, especially in time series modeling, including speech recognition, natural language processing and sequence prediction (Graves, 2013).

The behavior of non-linear systems can be also regarded as special time series. However, there are two correlation time series in system identification: input and output. Non-linear system modeling is to find the best conditional probability distribution of the output with respect to the input (Ljung, 1987). There are few articles addressed on LSTM for system identification. As well as the problems of online learning of LSTM and Vanishing gradient, LSTM cannot model the correlation of the input and output effectively. In order to take the great advantages of LSTM in time series modeling, the other identification techniques are combined with the LSTM. In (Hirose , 2017), the mini-batch training method is applied to LSTM. In (Wang, 2017), LSTM is the basic model in multiple models adap-

2405-8963
Peer review under responsibility of International Federation of Automatic Control.
10.1016/j.ifacol.2018.07.326

tive identification. However, the above LSTM cannot work well with the simulation model, which is used for multi-step prediction. They have to use the past test output.

In this paper, the problem of non-linear modeling with the simulation mode is solved by the combination of LSTM and the feedforward NN. Although the novel LSTM has hierarchical and recurrent structure, by the backpropogation and backpropogation through time methods, we propose a simple training method for this neural model. We also use two non-linear system examples to compare our neural modeling with the others. The results show that the modified LSTM model proposed in this paper is much better than the other existed neural models.

2. NON-LINEAR SYSTEM MODELING WITH THE RECURRENT NEURAL NETWORKS LSTM

Unlike traditional neural networks, the recurrent neural networks LSTM is an extremely efficient tool when the information is sequential. The basic condition of LSTM modeling is that all inputs and outputs are independent of each other. However, the dynamic non-linear system has the following form

$$y(k) = \Phi\left[y\left(k-1\right), \cdots, y\left(k-n_y\right), u\left(k\right), \cdots, u\left(k-n_u\right)\right] \tag{1}$$

$\Phi\left(\cdot\right)$ is an unknown non-linear difference equation representing the plant dynamics, $u\left(k\right)$ and $y\left(k\right)$ are measurable scalar input and output, n_y and n_u are the last values of the output and input, respectively, to be considered for the system dynamics. This discrete-time system (1) can be identified by the following two types of models:

(1) Simulation model,

$$\hat{y}(k) = F[u\left(k\right), \cdots, u\left(k-n\right)] \tag{2}$$

where $\hat{y}(k)$ is the output of the model, $N\left[\cdot\right]$ is the model structure; for example it is a neural network. n is the regression order for the input $u\left(k\right)$. since n_u is unknown, $n \neq n_u$. In general form, (2) can be written as

$$\hat{y}(k) = N[\hat{y}(k-1), \cdots, \hat{y}(k-m), u\left(k\right), \cdots, u\left(k-n\right)] \tag{3}$$

where m is the regression order for the model output $\hat{y}\left(k\right)$, $m \neq n_y$. It is the parallel identification model (Narendra,1991).

(2) Prediction model,

$$\hat{y}(k) = N[y\left(k-1\right), \cdots, y\left(k-m\right), u\left(k\right), \cdots, u\left(k-n\right)] \tag{4}$$

where m is the regression order for the output $y\left(k\right)$, $m \neq n_y$. It is the series-parallel identification model as in (Narendra,1991)

In many modeling applications, such as multi-step prediction, the past test output $y\left(k-i\right)$ is not available and the prediction model (4) cannot be used. The static input-output mapping model (2) does not represent the dynamic relationships between $y(k)$ and $u\left(k\right)$. Usually we use a self-recurrent model (3) or recurrent model, such as LSTM.

Since connecting previous information to the present task depends on many factors, such as the gap length. LSTM can learn to use the past information, where the gap between the relevant information and the place is small. In theory, a RNN is absolutely capable of handling such "long-term dependencies." by picking parameters. In practice, it does not seem to be able to learn them. LSTM uses gas cell to remember them. The key to LSTMs is the cell state. The gate in LSTM is a way to permit information to enter. LSTM has three gates to protect and control the cell state: forget gate, input gate, and output gate.

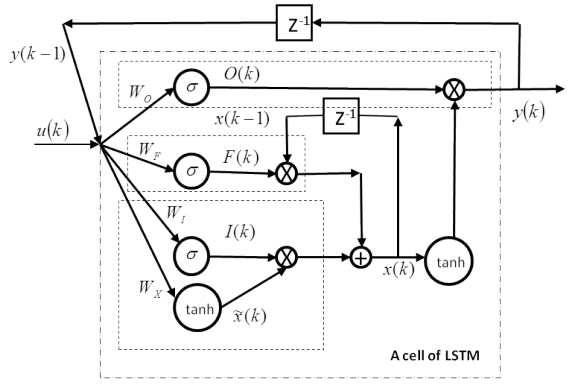

Fig. 1. The recurrent structure of LSTM

1) Forget gate:

$$F\left(k\right) = \sigma\left(W_F\left(k\right)\left[y\left(k-1\right), u\left(k\right)\right]\right)$$

where $u\left(k\right)$ is the input, $y\left(k-1\right)$ is the output, σ is a sigmoid function, $\sigma = 1$ represents "keep this", $\sigma = 0$ represents "get rid of this" $F\left(k\right)$ is the input to the cell state x_{k-1}, $W_F\left(k\right)$ is the weight.

2) Input gate: if the input $u\left(k\right)$ could be added into the hidden layer, then

$$I\left(k\right) = \sigma\left(W_I\left(k\right)\left[y\left(k-1\right), u\left(k\right)\right]\right)$$
$$\tilde{x}\left(k\right) = \tanh\left(W_X\left(k\right)\left[y\left(k-1\right), u\left(k\right)\right]\right)$$

where $\tilde{x}\left(k\right)$ and $I\left(k\right)$ are inner states, see Figure 1.

3) Output gate: the output is based on the cell state, we only output the parts we decided on

$$O\left(k\right) = \sigma\left(W_{O,k}\left[y\left(k-1\right), u\left(k\right)\right]\right)$$
$$y\left(k\right) = O\left(k\right)\tanh\left(x\left(k\right)\right)$$

where the sigmoid function σ decides what parts of the cell state go to output. The cell state is updated as

$$x\left(k\right) = F\left(k\right)x\left(k-1\right) + I\left(k\right)\tilde{x}\left(k\right)$$

here the old state $x\left(k-1\right)$ being multiplied by $F\left(k\right)$ means to forget the things we want, and add $I\left(k\right)\tilde{x}\left(k\right)$

$$\begin{aligned} x\left(k\right) &= \sigma\left(W_F\left(k\right)\left[\hat{y}\left(k-1\right), u\left(k\right)\right]\right)x\left(k-1\right) \\ &+\sigma\left(W_I\left(k\right)\left[\hat{y}\left(k-1\right), u\left(k\right)\right]\right)\tanh\left(W_X\left(k\right)\left[\hat{y}_{k-1}, u_k\right]\right) \\ \hat{y}\left(k\right) &= \sigma\left(W_O\left(k\right)\left[\hat{y}\left(k-1\right), u\left(k\right)\right]\right)\tanh\left(x\left(k\right)\right) \end{aligned} \tag{5}$$

where $x\left(k\right) = \text{rand}$.

The object of non-linear system modeling using LSTM is to update the weights W_F, W_I, W_X, and W_O, such that the output of the neural network (5) convergence to the system output $y\left(k\right)$ in (1)

$$\arg\min_{W_F, W_I, W_X, W_O}\left[\hat{y}\left(k\right) - y\left(k\right)\right]^2$$

From Figure 1 we can see that LSTM is the simulation model (2). When the past test output $y\left(k-i\right)$ is available, we can defined

$$U\left(k\right) = \left[y\left(k-1\right), \cdots, y\left(k-m\right), u\left(k\right), \cdots, u\left(k-n\right)\right]$$

then the LSTM is the prediction model (4). In the multi-step prediction case, for example we want to predict $y(k+10)$ using $y(k), y(k-1), \cdots u(k), u(k-1), \cdots$, we have to use the simulation model (2) or (3). Like most neural models, the LSTM (5) could not identify the non-linear system (1) with the simulation models very well (Wang, 2017).

In this paper, we combine the classical neural networks with LSTM. The novel neural model is shown in Figure 2. Here we use $p \times q$ LSTMs, which are connected in simple feedforward form. The final p LSTMs are fully connected to a multilayer perception.

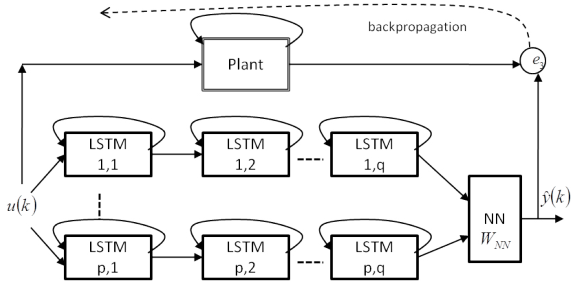

Fig. 2. LSTM based neural model

3. NEURAL MODEL TRAINING

All recurrent neural networks are composed of certain blocks or chained modules. The structure of these modules is formed by a single activation function, commonly tanh. LSTM logically show this same structure; however, the structure of each module is somewhat more complex. The training of recurrent neural networks is also done using backpropogation. However, since it takes into account not only the present state, but also all the previous states it is called Back Propogation Through Time (BPTT)

For a recurrent cell

$$x(k) = \phi[W(k)\, x(k-1)]$$

It can be unfolded into m steps

$$x(k-m) = \phi[W(k)\, x(k-m-1)]$$

If $y(k)$ is the desired output, $\hat{y}(k) = x(k)$, the instantaneous of the squared output errors is

$$J(k) = \frac{1}{2}[\hat{y}(k) - y(k)]^2 = e_o^2(k)$$

The gradient decent is

$$W(k+1) = W(k) - \eta \frac{\partial J(k)}{\partial W(k)}$$

where $\frac{\partial J(k)}{\partial W} = \frac{\partial J(k)}{\partial e_o(k)} \frac{\partial e_o(k)}{\partial y(k)} \frac{\partial y(k)}{\partial x(k)} \frac{\partial x(k)}{\partial p(k)} \frac{\partial p(k)}{\partial W} = e_o(k)\, \phi' x(k-1)$.
For m steps

$$W(k+1) = W(k) - \eta \sum_{i=1}^{m} e_i(k)\, x(k-i)$$

$$e_i(k) = e_{i-1}(k)\, W(k)\, \phi' = e_o(k) \Pi_{i=1}^{m} [W(k)\, \phi']$$

If $\|W(k)\, \phi'\| > 1$, it is gradient blow up. If $\|W(k)\, \phi'\| < 1$, it is gradient vanish. LSTM can avoid the gradient vanish problem by adding the gate cells, such that only when $\|W(k)\, \phi'\| \approx 1$, the layer is activated.

By the above BPTT for modeling, the weights of the LSTM in Figure 1 are updated by:

$$\begin{aligned}
W_O(k+1) &= W_O(k) - \eta \sum_{i=1}^{M} e_O(k)\, [\hat{y}(k-i), u(k-i)] \\
W_F(k+1) &= W_F(k) - \eta \sum_{i=1}^{M} e_F(k)\, [\hat{y}(k-i), u(k-i)] \\
W_I(k+1) &= W_I(k) - \eta \sum_{i=1}^{M} e_I(k)\, [\hat{y}(k-1), u(k)] \\
W_X(k+1) &= W_X(k) - \eta \sum_{i=1}^{M} e_X(k)\, [\hat{y}(k-1), u(k)]
\end{aligned} \tag{6}$$

where $e_O(k) = e_{O-1}(k)\, W_O(k)\, \sigma' \phi(x(k))$,

$$e_{O1}(k) = e_o \tag{7}$$

$e_F(k) = e_{F-1}(k)\, W_F(k)\, \sigma' x(k-1)$,
$e_I(k) = e_{I-1}(k)\, W_I(k)\, \sigma' \phi(W_X(k)\, [\hat{y}(k-1), u(k)])$,
$e_X(k) = e_{X-1}(k)\, W_X(k)\, \phi' \sigma(W_I(k)\, [\hat{y}(k-1), u(k)])$.

The learning law (6) is only one LSTM cell. The novel neural model is a hierarchical structure. The objective is to train the weights so that the error between the output of the neural model in Figure 2, and the output of the plant (1) is minimized. The performance index is defined as

$$J = \frac{1}{2} e_0^2 \quad e_o = \widehat{y} - y$$

The block NN is the classical feedforward neural network, the training law is the gradient descent

$$W_{NN}(k+1) = W_{NN}(k) - \eta \widehat{y}_{i,q}(k)\, e_o(k) \tag{8}$$

where $\eta > 0$ is the learning rate, $\widehat{y}_{i,q}$, $i = 1 \cdots p$, are the outputs of the last layer of the LSTMs.

For each LSTM, the training law is (6), the training error e_o (7) becomes

$$\begin{aligned}
e_{i,q} &= e_o(k)\, W_{NN} \\
e_{i,j-1} &= e_{O,i,j} \sigma' W_O
\end{aligned} \tag{9}$$

where $i = 1 \cdots p$, $j = 1 \cdots q$. The identification error $e_o(k)$ can be propagated to the other blocks. Since the interior structures of each LSTM are the same, we can use the same technique for each block's training. If we know the output error of each block, we can train this block. The training procedure is as follows.

(1) Calculate the output of each LSTM by (5). Some outputs of the model should be the inputs of the next level.
(2) Calculate the error for each block starting from the last block. Then we back propagate the identification errors by (7).and (9)
(3) Train the weights for each block independently by (8) and (6).

4. COMPARISONS

One way to model and understand a system is to be able to predict its behavior in a very short time, taking into account what has happened during the last step. This is known as one-step prediction. Most of neural models work well for the one-step prediction with the model (4). In this paper we challenge the multi-step prediction problem; we only use the input $u(k)$ as in Figure 2. In this session, we

use two examples to compare with the classical Multilayer Perceptrons (MLP).

4.1 Wiener-Hammerstein system

This example describes a benchmark for non-linear modeling using the SYSID 2009 Wiener-Hammerstein data set (Schoukens, 2009). There are 188,000 pair samples that will be divided in two parts: 80% for training and 20% for testing the model.

We first use the prediction model (4) with $n_y = 4$, $n_u = 5$. The recursive input vector is

$$[y(k-1)\cdots y(k-4)\ u(k)\cdots u(k-5)]^T .$$

The classical MLP has two hidden layers, each hidden layer has 10 nodes. The LSTM model in Figure 2 uses $p = 2$, $q = 1$. The Mean Squared Error (MSE) of the testing for MLP it is 6.45×10^{-4};for LSTM it is 1.21×10^{-4}. Both of them work well.

Then we use the simulation model (4) with $n_y = 0$, $n_u = 10$. The recursive input vector is $[u(k)\cdots u(k-10)]^T$. The MLP has the same structure as above and the MSE of the testing is 2.43. The MSE of the testing with LSTM is 7.17×10^{-3}. So LSTM proposed in this paper can model the Wiener-Hammerstein system with the simulation model, see Figure 3, while MLP cannot. If we increase the dimension of the input, we found that when the input vector reaches $[u(k)\cdots u(k-30)]$, the MSE is 5.15×10^{-4}. Then the accuracy cannot be improved much.

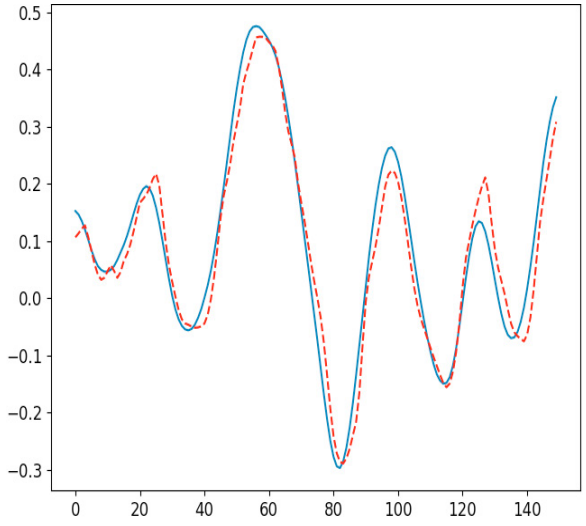

Fig. 3. Test result with LSTM and the input $[u_{(k)}, \ldots, u_{(k-10)}]$

It is necessary to review different network architectures of our LSTM. The results with different p and q in Figure 2, are shown in Table 1.

Table 1, Different structure of LSTM ($\times 10^{-4}$)

Layers	**Training**	**Testing**
$p = 1, q = 1$	4.84	5.15
$p = 3, q = 1$	2.66	2.98
$p = 4, q = 1$	2.86	3.17
$p = 50, q = 1$	2.94	3.15
$p = 80, q = 1$	2.26	2.53
$p = 150, q = 1$	2.30	2.53
$p = 4, q = 4$	6.72	6.95
$p = 8, q = 8$	5.28	5.72
$p = 4, q = 20$	5.62	6.32
$p = 8, q = 40$	5.22	5.72

4.2 A non-linear system

We use the benchmark model of (Narendra,1991)

$$y(k+1) = \frac{y(k)}{1+y^2(k)} + u^3(k)$$

with the input as $u(k) = A\sin(\frac{\pi k}{50}) + B\sin(\frac{\pi k}{20})$. In training phase, $A = B = 1$, in testing phase $A = 0.9, B = 1.1$. The conditions are the same as the above. The testing results are shown in Figure 4.

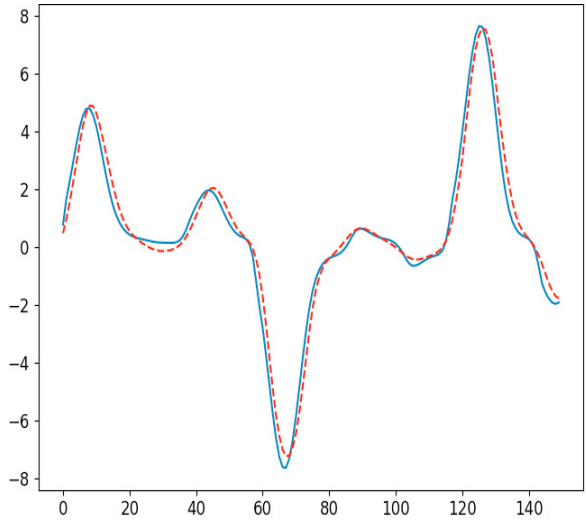

Fig. 4. Testing results of LSTM with input $[u_{(k-1)}, \ldots, u_{(k-3)}]$

4.3 Conclusion

In this paper, we successfully model the non-linear systems using a novel LSTM, which can only use the input $u(k)$, which is important for simulating systems and analyzing their behavior. It can also be concluded that by using the training algorithms, the testing results are satisfied. Furthermore, using the concept of recurrent neural networks, LSTM reduces the effort even more.

References

Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle (2007). Greedy layer-wise training of deep networks, *Advances in Neural Information Processing Systems (NIPS'06)*, pp. 153-160.

A.Graves, A.Mohamed, G.Hinton, (2013). Speech Recognition with Deep Recurrent Neural Networks.*arXiv: 1303.5778*

N.Hirose and R.Tajima, (2017). Modeling of Rolling Friction by Recurrent Neural Network using LSTM, *2017 IEEE International Conference on Robotics and Automation (ICRA),* Singapore, May 29 - June 3, 6471-6475

G.E.Hinton, S. Osindero, Y.W. Teh (2006) A fast learning algorithm for deep belief nets, *Neural Computation*, 18 (7).

N.Hirose and R.Tajima (2017). Modeling of Rolling Friction by Recurrent Neural Network using LSTM, *2017 IEEE International Conference on Robotics and Automation (ICRA),* Singapore, 6471-6476.

S.Hochreiter and J.Schmidhuber (1997). Long short-term memory. *Neural computation,* 9(8) : 1735-1780.

A.Kumar and Y.S.Chandel (2014). Solar radiation prediction using Artificial Neural Network techniques: A review, *Renewable and Sustainable Energy Reviews*,Volume 33, Pages 772-781

P.Kingma and J.Ba, (2014). Adam: A Method for Stochastic Optimization. *arXiv:1412.6980 [cs.LG]*, December

K. S. Narendra and K. Parthasarathy (1991). Gradient methods for optimization of dynamical systems containing neural networks, *IEEE Transactions od Neural Networks*, pp. 252-262

A. Krizhevsky, I. Sutskever, G.E. Hinton, ImageNet Classification with Deep Convolutional Neural Networks, *NIPS*. 2012.

Y. LeCun, L. Bottou, Y. Bengio, P. Haffne, Gradient based learning applied to document recognition, *Proceeding of the IEEE*, 1998.

L.Ljung (1987) *System Identification-Theory for User*, Prentice Hall, Englewood Cliffs, NJ 07632.

O.Nelles (2013). *Nonlinear system identification: from classical approaches to neural networks and fuzzy models*. Springer Science & Business Media.

J. Schoukens, J. Schoukens, and L. Ljung, (2009). Wiener-Hammerstein benchmark, *15th IFAC Symposium on System Identification*, Saint-Malo, France.

N.Srivastava, G.Hinton, A.Krizhevsky, I.Sutskever, R. Salakhutdinov (2014). Dropout: a simple way to prevent neural networks from overfitting. *Journal of machine learning research*, 15 (1), 1929-1958.

Y.Wang, (2017), A New Concept using LSTM Neural Networks for Dynamic System Identification, *2017 American Control Conference*, Seattle, USA, 5324-5329

N.Hirose and R.Tajima (2017). Modeling of Rolling Friction by Recurrent Neural Network using LSTM, *2017 IEEE International Conference on Robotics and Automation (ICRA),* Singapore, 6471-6476