Estimating Measurement Probability Distributions with Mixture Density Networks

Y. T. Chung, T. Williams, and Y. Sun, *Department of Computer Science and Engineering, University of South Florida*

***Abstract*—Since sensors are often impeded by noise and mobile robots depend on sensors to determine their position and orientation, the estimation of noise becomes an important issue. In order to increase the accuracy of sensors, we used mixture density network to calculate the probability distributions of the noise measurement for sensors in this paper. For distinct datasets, we built different networks and tested the training results to find the appropriate hyper-parameters for each mixture density network.**

I. Introduction

State estimation is the method of finding a mobile robot's state in an environment. Mobile robots rely on sensors to determine their position and orientation since they are not capable to directly approximate their states. Many different kinds of sensors are developed to help robots in localization such as GPS transceivers are used for determining where robots are on Earth and sonar sensors are used for determining the distance between robots and obstacles in an environment. However, sensor measurements are not reliable due to the effect of noises.

The motivations of noise measurements include a robot depends on its sensors to decide its state and the state of the environment [2-3]; sensors support an entity to locate over time to deal with tracking and localization problems [4-6]; the number of noise determines the performance of features match for feature matching problems [7-9]; robots are require to have expectation when performing certain actions like pouring water or cutting cheese [10-14]. Those motivations impel the fields like robotics and computer vision to estimate the amount of noise.

A robot's state and the state of an environment affect the amount of noises. Therefore, we need a mechanism to effectively compute the number of noise for robots. Conditional probabilistic models can be used for estimation of noise in sensor measurements. As a result, we used mixture density networks in our experiments to assess the amount of noise.

II. Mixture Density Network

When encountering the target data is multi-valued, the conditional averages provide little information of the target properties by reason of the average of many correct target values is not exactly the essential value. To derive a complete description of a data and predict the outputs corresponding to new input values, we need to model the conditional probability distribution of a target data and accommodating again to the input.

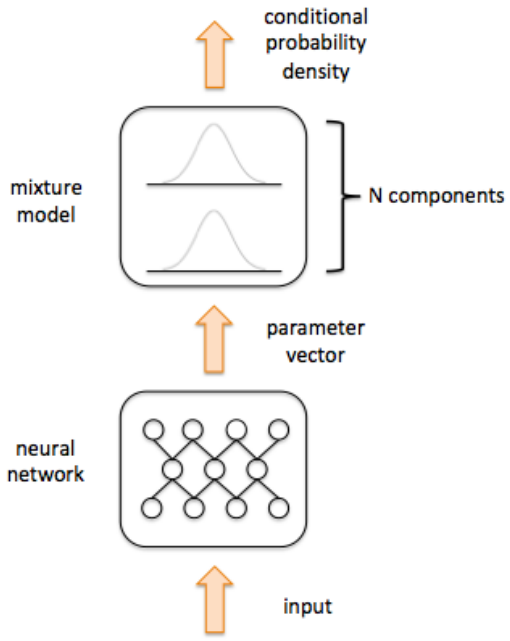


Figure 1. The Architecture of a Mixture Density Network

A mixture density network (MDN) is a feed-forward network that combining a conventional neural network with a mixture density model, which contains the conventional least-squares technique but replacing the Gaussian distribution with a mixture model. As a result, MDN is able to represent random conditional probability distributions and flexible to build general distribution functions. A linear combination of kernel functions for the probability density of the target data is

$$p(t|x) = \sum_{i=1}^{m} \alpha_i(x)\, \emptyset_i(t|x) \qquad (1)$$

where x is a set of input variables $\{x_1, x_2 \dots\}$ to a set of output variables $t = \{t_1, t_2 \dots\}$, and m is the number of components in the mixture layer. The parameters $\alpha_i(x)$ are mixing coefficients, which represent prior probabilities conditioned on x of the target t from the i^{th} component of the mixture. The kernel functions $\emptyset_i(t|x)$ are the conditional density of the target t for the i^{th} kernel, and the functions are Gaussian in the form

$$\emptyset_i(t|x) = \frac{1}{(2\pi)^{c/2}\,\sigma_i} \exp\left\{-\frac{\|t-\mu_i(x)\|^2}{2\sigma_i(x)^2}\right\} \qquad (2)$$

where the $\mu_i(x)$ stands for the center of the i^{th} kernel with components. The variance $\sigma_i(x)$ represents the distribution that components of the output are hypothesized they are independent statistically to each other, but the components of t are not hypothesized that they are statistically independent.

The mixture model takes the parameters of mixing coefficients $\alpha_i(x)$, the means $\mu_i(x)$ and the variances $\sigma_i(x)$ as general functions of x by using the output of a conventional neural network as its input. With sufficient amounts of kernel functions and hidden units, a MDN can estimate conditional density $p(t|x)$ [1].

III. Dataset

In this section, we briefly describe the two types of dataset we used in our experiments:

A. Localization Dataset

We used simplified version of the UTIAS multi-robot cooperative localization dataset [15]. Various amounts of noise were added to the datasets with perfect measurements by random sampling. There are four values in each simplified dataset, which are (1) a range measurement, (2) a bearing measurement, (3) an x position, and (4) a y position. The x and y position represent the obstacle position relative to the robot's position and orientation. The range value represents the Euclidean distance and the bearing value represents the angle from a robot to an obstacle as shown in Fig. 2. The range units are in meters, while the bearing units are in radians.

B. Motion Dataset

Motion data are the data points show the number of force that mobile robots applied to a tool on a certain position and orientation. Each Motion dataset contains an input data and an output data. In each input data, there are nine features to describe the three-dimensional position of a tool: (1) x, (2) y, (3) z, (4) sin (yaw), (5) cos (yaw), (6) sin (pitch), (7) sin (pitch), (8) sin (roll), and (9) cos (roll). The first three features represent a three-dimensional position of the tool, and the rest of features represent the three-dimensional orientation of the tool. In the corresponding output data, there was one target feature, and it represents the most dominant force that the sensors detected during the motion. The x, y, and z units are in decimeters, while the other features' units are in sine and cosine space.

IV. Methodology

We implemented MDNs based on Christopher M. Bishop's paper [1] with Keras Deep Learning library in Python.

A. Mixture Density Network Helper Functions

The architecture of our helper functions is shown in Fig. 1, which has three layers and n components that varied on different dataset. Here we illustrate our each function:

- Add a univariate mixture layer to the end of a multi-layer perceptron. This layer will be the output layer of a mixture density network and provide a matrix containing the conditional mixture coefficients, means, and standard deviations. According to [1], the mixture coefficients $\alpha_i(x)$ need to satisfy the constraint

$$\sum_{i=1}^{m} \alpha_i(x) = 1 \tag{3}$$

$$\alpha_i = \frac{exp(z_i^{\alpha})}{\sum_{j=1}^{M} exp(z_j^{\alpha})} \tag{4}$$

Therefore, we use softmax function as its activation function. For variances σ_i, they are scale parameters; thus, we use the exponentials of the corresponding network outputs as

$$\sigma_i = exp(z_i^{\alpha}) \tag{5}$$

where z_i^{α} represent the outputs of network. This method can avoid pathological configurations in one or more of the variances become zero. As for centers μ_i, they represent location parameters; therefore, we use the network outputs directly

$$\mu_i = z_i^{\mu} \tag{6}$$

where z_i^{μ} represent the outputs of network.

- Separate the output matrix of the MDN into parameters, which are three arrays: a mixture coefficient matrix, a component mean matrix, and a component standard deviation matrix.
- Compute Gaussian kernel probability from each distribution. This computation is on the basis of (2) to get the individual probabilities of each component.
- Compute total probability vector from mixture coefficient matrix and the kernel probability matrix, which based on (1).
- Negative log likelihood loss is an error function by taking the negative logarithm of likelihood, which in the form

$$E = -\ln\left\{\sum_{i=1}^{m} \alpha_i(x)\, \emptyset_i(t|x)\right\} \tag{7}$$

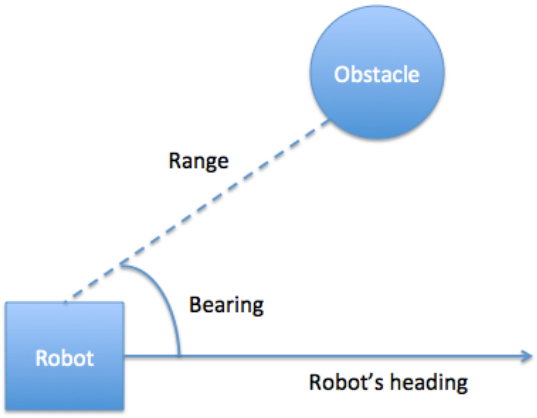


Figure 2. The Robot and an Obstacle

- Compute the mixture total mean and variance of entire distribution from mixture coefficients, means, and standard deviations. To calculate statistics mean, we use the conditional average of the target data according to [1], given by

$$\sum_i \alpha_i(x)\,\mu_i(x) \tag{8}$$

As for the variance of the density function of the conditional average, given by

$$\sum_i \alpha_i(x)\left\{\sigma_i(x)^2 + \left\|\mu_i(x) - \sum_j \alpha_j(x)\mu_j(x)\right\|^2\right\} \tag{9}$$

- Compute the mean and standard deviation from the component with the maximum mixture coefficient.

B. Network Training and Hyper-Parameter

The networks and hyper-parameters varied from different type of dataset. For the localization dataset, two MDN were trained: the first one was estimating the (1) bearing measurement distribution $p(\rho|x,y)$and the second one was (2) bearing measurement distribution $p(\phi|x,y)$. For the motion dataset, only one MDN was trained for estimating the force distribution $p(f|x,y,z\ldots)$. The detailed information about each network is shown in Table I. As we can see, the first parameter inside the parentheses of layers is the number of units (neurons) and the second parameter is the activation function.

The data we used for training and evaluation were localization assessment 00 and 29 dataset, and force assessment 25 dataset. In addition, we divided each dataset into a training set, a validation set, and a testing set. The training set was 75% of data and we used 10-fold cross validation so as to find better hyper-parameters.

TABLE I. NETWORKS

Dataset	Hyper-parameters	
Local 00 (Range)	Layers = (5, activation="relu") (10, activation="tanh") (5, activation="relu")	Epochs = 30 Batches = 128 Learning rate = 0.001 Components = 1
Local 00 (Bearing)	Layers = (5, activation="relu") (10, activation="tanh") (5, activation="relu")	Epochs = 30 Batches = 128 Learning rate = 0.001 Components = 1
Local 29 (Range)	Layers = (5, activation="relu") (10, activation="tanh") (5, activation="relu")	Epochs = 30 Batches = 128 Learning rate = 0.001 Components = 1
Local 29 (Bearing)	Layers = (5, activation="relu") (10, activation="tanh") (5, activation="relu")	Epochs = 30 Batches = 128 Learning rate = 0.001 Components = 1
Motion 25	Layers = (5, activation="relu") (10, activation="tanh") (5, activation="tanh")	Epochs = 300 Batches = 64 Learning rate = 0.001 Components = 1

TABLE II. MEASUREMENTS OF MSE BASED ON NUMBER OF COMPONENTS

Network / Number of Components	Model Local 00 Range	Model Local 00 Bearing	Model Local 29 Range	Model Local 29 Bearing	Model Force 25
1	0.002	0.002	0.001	0.001	0.099
2	12.867	0.084	12.729	0.083	0.551
3	13.870	0.085	13.774	0.086	0.565

TABLE III. MEASUREMENTS OF MSE BASED ON LEARNING RATE

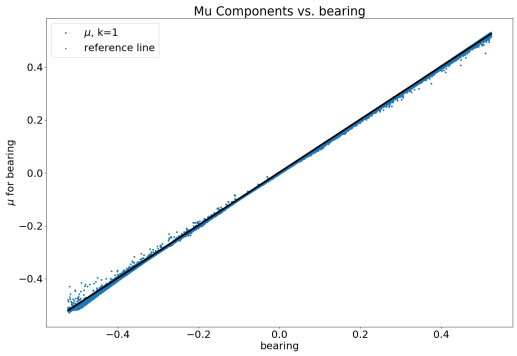
Network / Learning Rate	Model Local 00 Range	Model Local 00 Bearing	Model Local 29 Range	Model Local 29 Bearing	Model Force 25
0.1	2.259	0.084	2.263	0.083	0.462
0.01	0.003	0.003	2.273	0.001	0.112
0.001	0.002	0.002	0.001	0.001	0.099
0.0001	0.002	0.002	0.001	0.001	0.152

TABLE IV. MEASUREMENTS OF MSE BASED ON BATCH SIZE

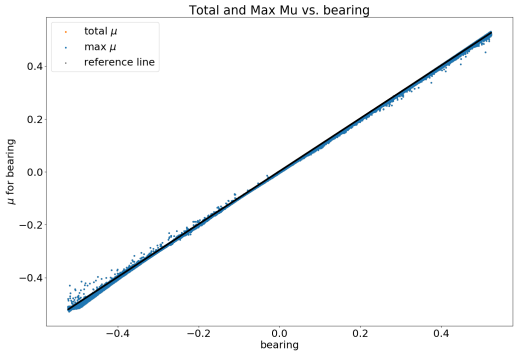
Network / Batch Size	Model Local 00 Range	Model Local 00 Bearing	Model Local 29 Range	Model Local 29 Bearing	Model Force 25
64	0.003	0.082	0.001	0.001	0.099
128	0.002	0.002	0.001	0.001	0.122
256	0.002	0.002	0.002	0.001	0.109

TABLE V. PERFORMANCE OF TRAINED MODELS ON TEST SETS

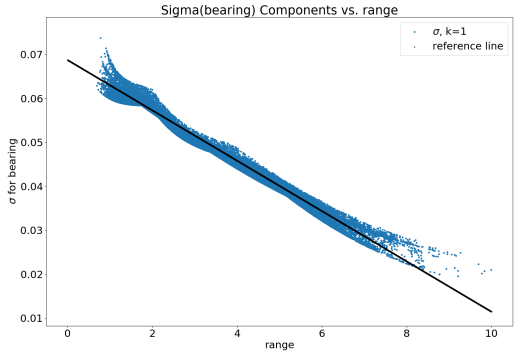
Network	MSE	Pearson's R^2
Model Local 00 Range	0.00247	0.99945
Model Local 00 Bearing	0.00252	0.98463
Model Local 29 Range	0.00113	0.99975
Model Local 29 Bearing	0.00077	0.99527
Model Force 25	0.09934	0.88256


Mu Components vs. bearing
μ, k=1
reference line
μ for bearing
bearing

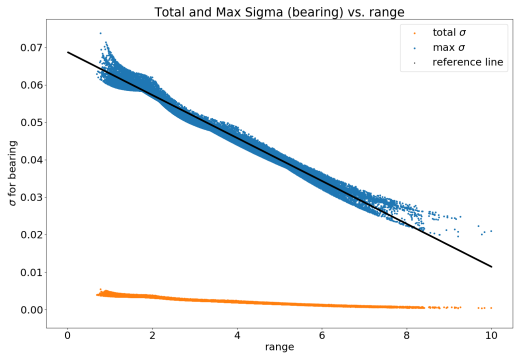
(a)


Total and Max Mu vs. bearing
total μ
max μ
reference line
μ for bearing
bearing

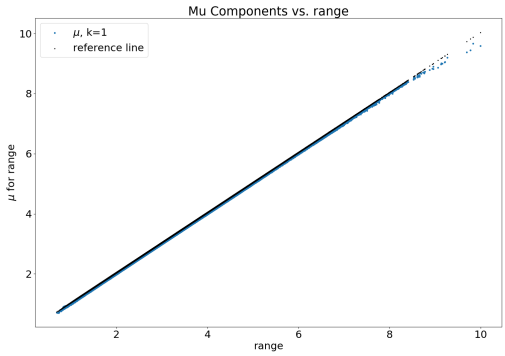
(b)


Sigma(bearing) Components vs. range
σ, k=1
reference line
σ for bearing
range

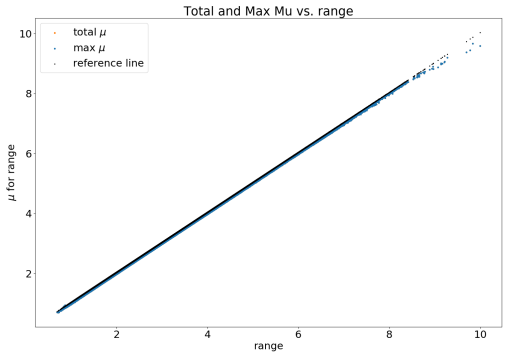
(c)


Total and Max Sigma (bearing) vs. range
total σ
max σ
reference line
σ for bearing
range

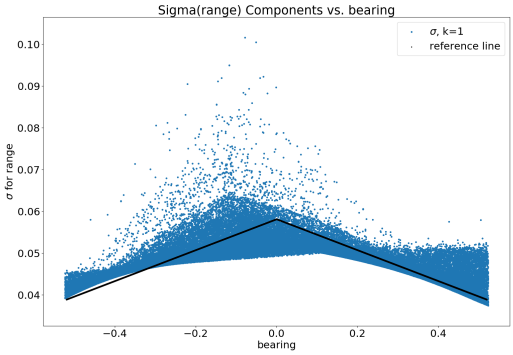
(d)


Mu Components vs. range
μ, k=1
reference line
μ for range
range

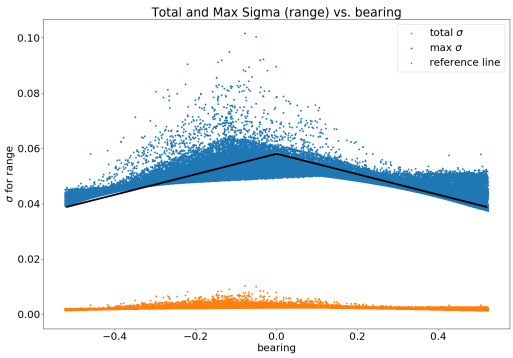
(e)


Total and Max Mu vs. range
total μ
max μ
reference line
μ for range
range

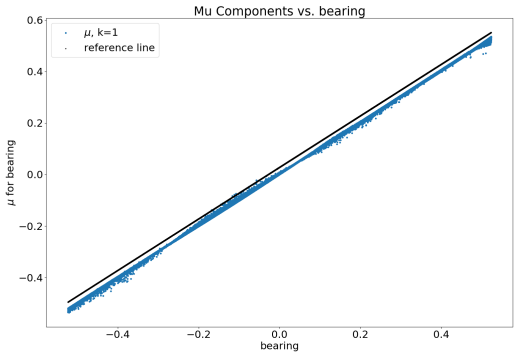
(f)


Sigma(range) Components vs. bearing
σ, k=1
reference line
σ for range
bearing

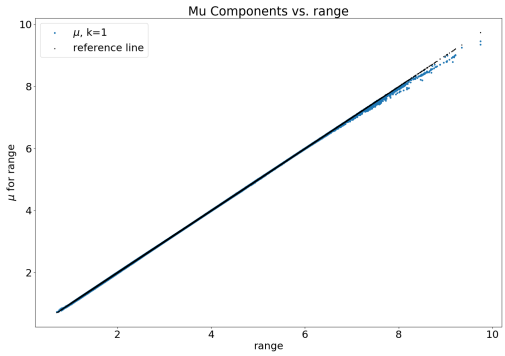
(g)


Total and Max Sigma (range) vs. bearing
total σ
max σ
reference line
σ for range
bearing

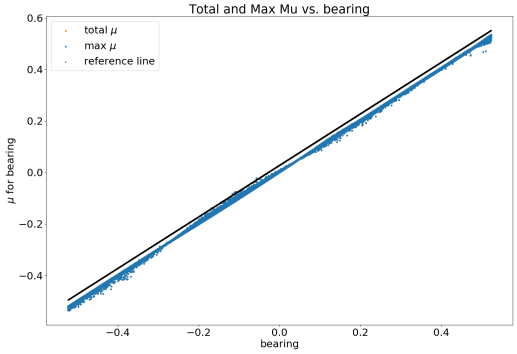
(h)



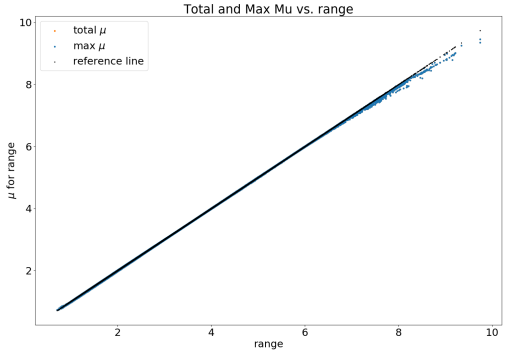
(i)



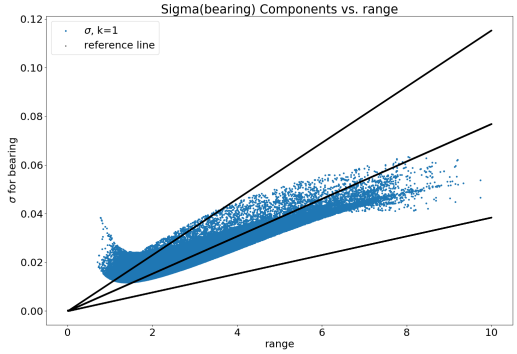
(m)



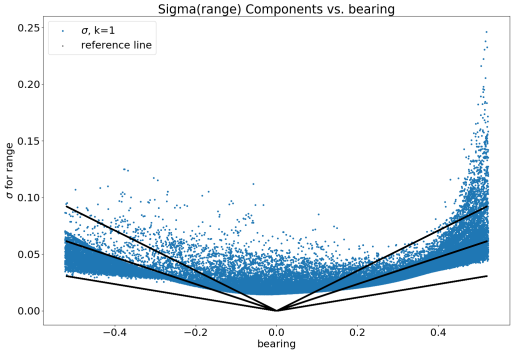
(j)



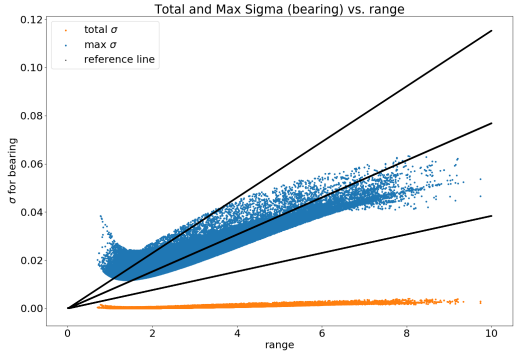
(n)



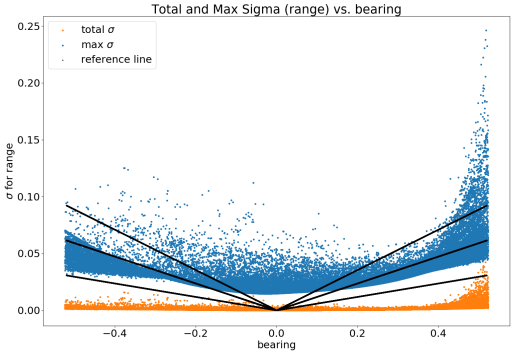
(k)



(o)



(l)



(p)

Figure 3. Graphical Results

V. Evaluation and Results

To evaluate the performance of our neural network, mean squared error (MSE) and Pearson's R^2 were used to contrast the MDN total mean with the target measurement values. In our experiments, there was no significant difference between the numbers of layers or units in each layer used for our model. As a result, we used three layers for every model and five units for first and third layers, and ten units for second layers. Moreover, we found that increasing the number of epochs would not affect the result after a certain value. Thus, we set 30 epochs for local models and 300 epochs for force model.

Table II, III, and IV demonstrate how the number of component, learning rate, and the size of batch influence MSE score of our models. For number of component, we found that using one component can effectively reduces MSE score. Moreover, with the learning rate 0.001, the performance was generally well even though there is only slight difference among learning rate 0.01, 0.001, and 0.0001. The effect on size of batch was even slightly. As we can see in Table 5, all of the MSE score from local datasets are below 0.01 and Pearson's R score is close to 1, which shows encouraging results. The MSE score of model force 25 is around 0.1 and Pearson's R score is around 0.88.

VI. Discussion

Fig. 3 shows all the measurement results for local datasets. The black lines in the graphs are the ground truth. (a), (b), (c), and (d) are the bearing measurement for 00 dataset; (e), (f), (g), and (h) are the range measurement for 00 dataset. (i), (j), (k), and (l) are the bearing measurement for 29 dataset; (m), (n), (o), and (p) are the range measurement for 29 dataset. Since we used only one component for every MDN model, the alpha for each model is staying stable at one. Even though there is a small bias from ground truth line in both (i) and (j), the plots of (a), (b), (e), (f), (m), and (n) align well with the ground truth. Accordingly, those MDN models had good results in the assessment of noises.

For (c) and (d), the plots also align with ground truth. As for (k) and (l), we can see the noises increase as the range increases as we anticipated that the closer the range is, the less the noise is. In (o) and (p), although we expected there is no noise when the bearing measurement is 0, we can see the plots follow the ground truth line that there are more noises when the radians of bearing are larger. The ground truth lines in graphs (g) and (h) are opposite to (o) and (p), (g) and (h) show the better results than (o) and (p), yet, there are slightly more noises when the radians of bearing are larger.

VII. Conclusion

In this paper, we presented an implementation of MDNs according to the paper from Christopher M. Bishop's to estimate the amount of noise for a robot's sensors. With localization data, we trained two MDN models: one was the range measurement and one was the bearing measurement. As for motion data, we trained one MDN model for the force measurement. The testing results have shown that the MSE scores for all localization dataset are under 0.003 and the Pearson's R scores are above 0.98. Even though the MSE score and the Pearson's R score from the MDN model of motion dataset are relatively higher than the other four MDN models, which are 0.099 and 0.88, the result is still encouraged.

References

[1] C. M. Bishop, "Mixture density networks," Aston University, 1994.

[2] H. M. Choset, "Principles of robot motion: theory, algorithms, and implementation," MIT press, 2005.

[3] T. Williams and Y. Sun, "Learning state-dependent sensor measurement models for localization," Intelligent Robots and Systems (IROS) 2018 IEEE/RSJ International Conference on IEEE, 2018.

[4] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: part I," IEEE robotics and automation magazine, vol. 13, no. 2, pp. 99-110, 2006.

[5] B. Lin, Y. Sun, X. Qian, D. Goldgof, R. Gitlin, and Y. You, "Video-based 3D reconstruction, laparoscope localization and deformation recovery for abdominal minimally invasive surgery: a survey," The International Journal of Medical Robotics and Computer Assisted Surgery, vol. 12, no. 2, pp. 158-178, 2016.

[6] B. Lin, A. Johnson, X. Qian, J. Sanchez, and Y. Sun, "Simultaneous tracking, 3D reconstruction and deforming point detection for stereoscope guided surgery," in Augmented Reality Environments for Medical Imaging and Computer-Assisted Interventions. Springer, pp. 35-44, 2013.

[7] B. Lin, Y. Sun, and X. Qian, "Thin plate spline feature point matching for organ surfaces in minimally invasive surgery imaging," Medical Imaging 2013: Image-Guided Procedures, Robotic Interventions, and Modeling, vol. 8671. International Society for Optics and Photonics, pp. 867112, 2013.

[8] B. Lin, Y. Sun, J. E. Sanchez, and X. Qian, "Efficient vessel feature detection for endoscopic image analysis," IEEE Transactions on Biomedical Engineering, vol. 62, no. 4, pp. 1141-1150.

[9] B. Lin, Y. Sun, J. E. Sanchez, and X. Qian, "Vesselness based feature extraction for endoscopic image analysis," in Biomedical Imaging (ISBI), 2014 IEEE 11th International Symposium, IEEE, pp. 1295-1298, 2014.

[10] Y. Huang, M. Bianchi, M. Liarokapis, and Y. Sun, "Recent data sets on object manipulation: A survey," Big data, vol. 4, no. 4, pp. 197-216, 2016.

[11] Y. Huang and Y. Sun, "A dataset of daily interactive manipulation," International Journal of Robotics Research, pp. 1-6, 2018.

[12] Y. Huang and Y. Sun, "Learning to pour," arXiv preprint arXiv: 1705.09021, 2017.

[13] Y. Lin and Y. Sun, "Grasp planning to maximize task coverage," The International Journal of Robotics Research, vol. 34, no. 9, pp. 1195-1210, 2015.

[14] Y. Lin, S. Ren, M. Clevenger, and Y. Sun, "Learning grasping force from demonstration," Robotics and Automation (ICRA), 2012 IEEE International Conference, IEEE, pp. 1526-1531, 2012.

[15] K. Y. Leung, Y. Halpern, T. D. Barfoot, and H. H. Liu, "The UTIAS Multi-Robot Cooperative Localization and Mapping Dataset," International Journal of Robotics Research, vol. 30, no. 8, pp. 969-974, 2011.