SENSOR FUSION USING MACHINE LEARNING TECHNIQUES

Project Report

By

Ruslan Masinjila (202488331)

Of

Group 24

Course: COMP-6936 Advanced Machine Learning

Instructor: Dr. Lourdes Peña-Castillo

Memorial University of Newfoundland
St. Johns

December 2, 2024

1. Introduction

Sensor fusion is the integration of homogenous or heterogeneous data and the associated errors/uncertainties from multiple sensors to produce more accurate and reliable information compared to the individual sensors. This technique is essential in various applications, including robotics, healthcare, and environmental sensing, where combining diverse data sources enhances system performance and decision-making. For example, in robotics, sensor fusion combines data from cameras, LiDAR, gyroscopes, and accelerometers for accurate and precise localization and navigation in dynamic environments. Current sensor fusion techniques include deterministic methods like the Kalman (KF), Extended Kalman (EKF), and Particle Filters (PF). These filters rely on mathematical models and assumptions about sensor noise and system dynamics. Probabilistic methods such as Bayesian networks also provide a framework for combining uncertain data. Despite their effectiveness in many scenarios, these approaches face limitations, such as sensitivity to model inaccuracies, difficulty handling large volumes of heterogeneous data, and challenges adapting to non-linear and complex environments. Additionally, filters such as the EKF are prone to producing inconsistent error estimates due to over-convergence or divergence of fused uncertainties.

Machine learning (ML) techniques offer promising solutions to overcome these limitations because they can learn complex patterns and relationships between sensor data without relying on explicit mathematical models. Techniques such as deep learning are increasingly used for sensor fusion. For instance, convolutional neural networks (CNNs) can process spatial data, such as depth data from various sources for robot localization and mapping. On the other hand, recurrent neural networks (RNNs) can analyze temporal sensor data for trajectory prediction. These ML-based approaches have demonstrated superior performance in non-linear and high-dimensional data scenarios. However, ML models often require extensive labeled data for training, which can be costly and time-consuming to acquire. Additionally, ensuring the real-time performance and interpretability of ML-based fusion systems is a significant challenge, particularly in safety-critical applications like autonomous driving.

1.1. Problem Statement

This project aims to study the feasibility of ML approaches in sensor fusion. Specifically, the project investigates the performance of Generalized Additive Models (GAM), Generative Adversarial Networks (GAN), and Convolutional Neural Networks (CNN) in the fusion of simulated range-bearing sensor data for robot localization.

1.1.1. Specific Objectives

The specific objectives of this project are as follows:

1. Generate a synthetic dataset simulating range-bearing sensor measurements with random uncertainty.
2. Divide the dataset into training and validation sets.
3. Train and evaluate the performance of Generalized Additive Models (GAM), Generative Adversarial Networks (GAN), and Convolutional Neural Networks (CNN) using 5-fold cross-validation on the training dataset.
4. Assess the performance of GAM, GAN, and CNN on the validation dataset.

2. Literature Review

Mobile robots must know their pose (position and orientation) to navigate and perform practical tasks. The problem of determining this pose relative to a global or local frame is called localization and is a critical component in providing autonomy to mobile robots. The basic idea behind most documented localization techniques involves the optimal combination of noisy and uncertain information from various robot sensors. Such sensors include proprioceptors such as wheel encoders and exteroceptors such as range-finders, sonars, depth cameras, radars, and LiDARS [1]. Generally, fusion of information from multiple sensors results in more accurate and confident pose estimation. Accuracy is the difference between the estimated and actual pose, whereas certainty is the difference between the estimated and actual or true error.

Probabilistic techniques such as the Kalman (KF), Extended Kalman (EKF), and Particle Filters (PF) have widely been used with significant success in sensor fusion for robot localization and mapping. Despite their success, filters such as the EKF are prone to converging to uncertain pose estimates due to their reliance on linearizing nonlinear models and assumptions about sensor noise and system dynamics, especially in decentralized systems. In other words, they are prone to produce optimistic (overconfident) or pessimistic (underconfident) pose estimates [2]. Estimates are optimistic when the estimated errors are smaller than actual ones. On the other hand, estimates are pessimistic when estimated errors are smaller than actual errors. If the estimated and actual errors are about the same, they are consistent. In recent years, Deep Learning (DL) has emerged as a leading approach for processing large datasets, characterized by three notable features [3]:

1. DL excels in nonlinear mapping, enabling the automatic creation of complex models that are otherwise challenging to design manually.
2. Its deep architectures allow for the generation of high-dimensional representations, offering a more comprehensive alternative to traditional feature extraction methods.
3. DL's versatility supports its application across diverse domains, including computer vision, autonomous driving, robotics, medical diagnostics, and industrial manufacturing.

These characteristics make deep learning (DL) a compelling technology for improving the performance of multi-sensor data fusion algorithms. For instance, Mahfouz et al. [4] introduced an innovative DL-based method that integrates a Kalman filter with both the acceleration data of a moving target and radio fingerprints of received signal strength indicators (RSSIs) from a wireless sensor network (WSN) to estimate the target's real-time position. The Kalman filter is refined using a kernel-based ridge regression iterative algorithm, resulting in high accuracy and robustness in position estimation. Similarly, Gao et al. [5] developed a DL-based solution utilizing an adaptive Kalman filter (AKF) embedded with an Elman neural network to effectively identify the position of interest.

Examples of learning models successfully used in data fusion include adapted versions of Generative Adversarial Networks (GAN) and Convolutional Neural Networks (CNN). Generative Adversarial Networks (GANs) are a type of machine learning model designed to generate realistic data by training two neural networks in a competitive setup: a generative model (G) that learns the data distribution and a discriminative model (D) that estimates the probability that a sample originates from the training data rather than from G [6]. On the other hand, Convolutional Neural Networks (CNNs) are a class of deep learning algorithms primarily used for processing and analyzing visual data, such as images and videos. They can automatically and adaptively learn spatial hierarchies of features from data, making them especially effective in tasks like image recognition, object detection, and image segmentation [7].

3. Methodology

3.1. Range-Bearing Sensor Model

A range-bearing sensor is a type of exteroceptor that measures the distance to that target (range), as well as the angle of the target relative to the sensor's frame of reference. For example, in **Figure 1**, the target at (**X_L, Y_L**) is at a distance **rho** and angle **phi** with respect to the range-bearing sensor mounted on robot **b**.

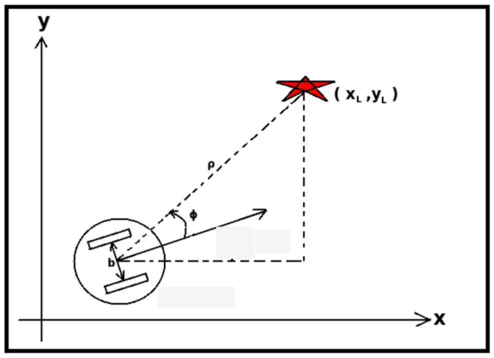

Figure 1: A Robot **b** measures the distance **rho** and bearing **phi** of the target at **(X_L, Y_L)**

In real-world settings, sensors must have some error associated with the measurements. Due to these errors, measurements are taken as intervals instead of discrete values. For example, typically, range-bearing measurements would look as follows:

(Range, Bearing) = (R, PHI) ± (ΔR, ΔPHI)

Consequently, the exact position of a target cannot be determined with certainty. Instead, its position is depicted as the area between two concentric circles bounded by 2ΔR and 2 ΔPHI (i.e., blue section), as illustrated in **Figure 2**.

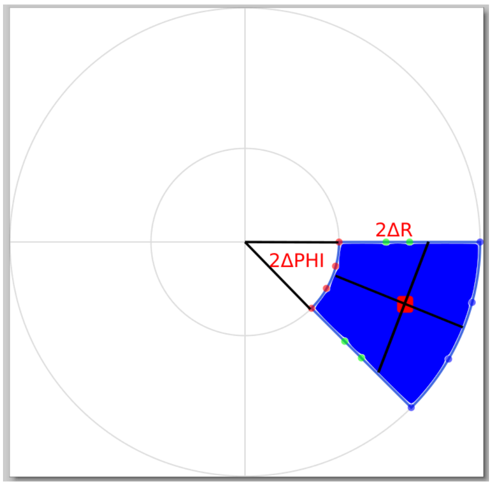

Figure 2: Uncertainty associated with Range-Bearing Measurements

From **Figure 2**, the equation of the perimeter around the uncertainty area (blue) associated with range-bearing measurements can be expressed in polar coordinates as follows:

$$E = R_1 <= R <= R_2 \text{ when } PHI = PHI_1$$
$$E = R_1 <= R <= R_2 \text{ when } PHI = PHI_1$$
$$E = PHI_1 <= PHI <= PHI_2 \text{ when } R = R_1$$
$$E = PHI_1 <= PHI <= PHI_2 \text{ when } R = R_2$$

Where: **(R_1 ,R_2)** and **(PHI_1 ,PHI_2)** are R's and PHI's low and upper bounds**,** respectively.

Because the Machine Learning models chosen for this project require actual data points in the form of cartesian (x,y) or polar (r, theta), the outer surface of the sensor model depicted in **Figure 2** was sampled into cartesian coordinates, as shown in **Table 1**.

Interval	**Number of (x,y) coordinates**
$R_1 <= R <= R_2$ when $PHI = PHI_1$	40
$R_1 <= R <= R_2$ when $PHI = PHI_1$	40
$PHI_1 <= PHI <= PHI_2$ when $R = R_1$	50
$PHI_1 <= PHI <= PHI_2$ when $R = R_2$	100
Complete loop	230

Table 1: Number of cartesian points along each side of the uncertainty area of the range-bearing sensor model

The sensor model was sampled using the Python script <1 sensor model.py>. **Figure 3** illustrates the sensor model's outer perimeter defined using 230 cartesian coordinates.

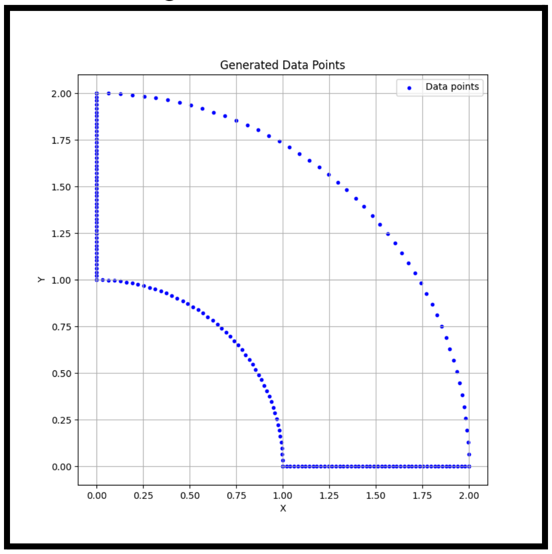

Figure 3: Base Range-Bearing sensor model defined using 230 cartesian points

3.2. Augmentation of Base Range-Bearing

The base range-bearing sensor model implemented in Section 3.1 was augmented several times to create 2000 different variations, each consisting of 230 cartesian coordinates. This augmentation was done using the Python script <2 data augmentor.py>. In other words, the cartesian points along the outer perimeter of the base sensor model were subjected to random rotation, scaling (resize), translation, and skewing.

Random rotations and translations were performed to simulate various locations and bearings of targets relative to a sensor while scaling and skewing were used to simulate different degrees of uncertainty and its spatial distribution.

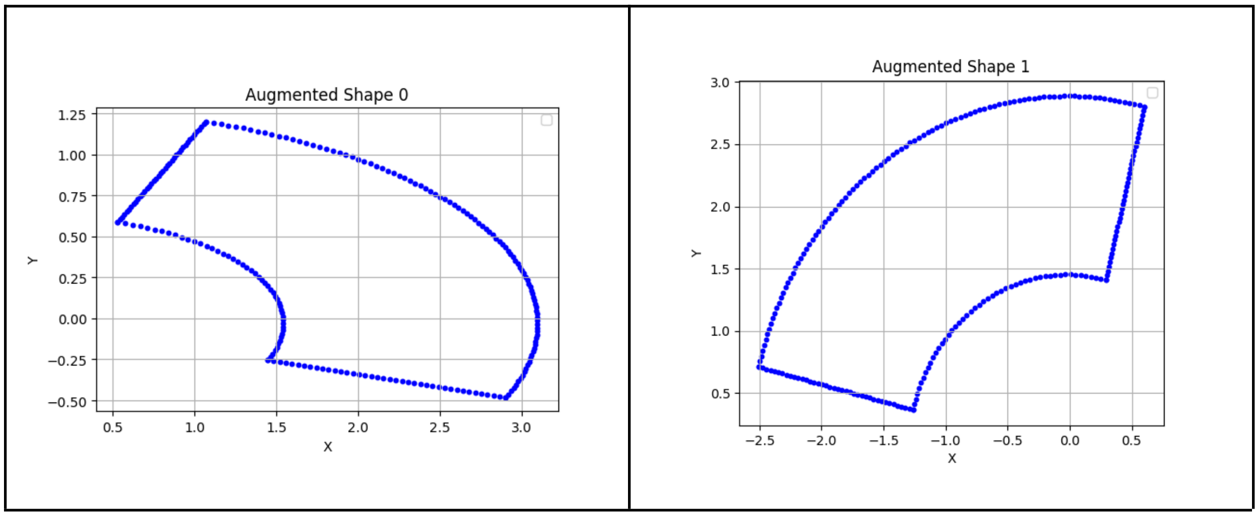

Figure 4: Examples of variations of sensor data obtained through Augmentation of the base Range-Bearing Sensor model.

Figure 4 depicts two of the 2000 variants of simulated sensor data acquired through augmentation of the base range-bearing sensor model. Each of the two examples is at a different location and has a different orientation, size, and degree of skewness.

3.3. Ground Truths Generation through Data Interpolation

Sensor fusion blends data from multiple sources to improve accuracy. For instance, the Extended Kalman Filter (EKF), discussed in Sections 1 and 2, integrates data from a robot's proprioceptive sensors, such as wheel encoders, with data from its exteroceptive sensors, like range-bearing sensors, to achieve more precise pose estimates. This fusion is handled probabilistically: robot poses are expressed as vectors (**mu**), while the corresponding uncertainties are modeled using square covariance matrices (**sigma**). Geometrically, these covariance matrices are either ellipses (2D) or ellipsoids (3D) around estimated robot poses. Thus, the fusion of covariance matrices by an Extended Kalman Filter (EKF) is analogous to the geometric blending of the shapes (ellipses or ellipsoids) in space. This can be done through intersection, union, or weighted interpolation of shapes. In this project, we will perform weighted interpolation between two shapes, assigning greater weight to the smaller shape due to its higher degree of certainty. As a result, the final shape will be a weighted hybrid of the two shapes.

Weighted interpolation is carried out in the script <3 data interpolator.py>, and the resulting interpolated data will serve as the ground truth (actual output) for the machine learning models. Before interpolation, the 2000 variants of simulated sensor data are split into two groups, each containing 1000 variants. Each variant from the first group is then geometrically fused with a variant from the second group, resulting in 1000 unique fusion pairs and their associated outputs. **Figure 5** provides examples of the fusion of simulated range-bearing sensor data pairs from the two groups.

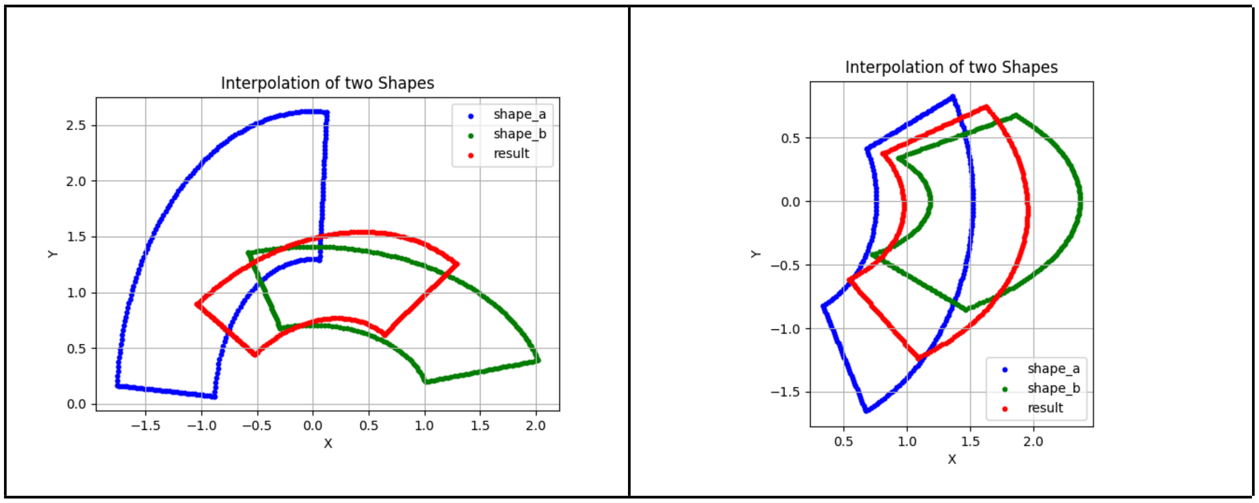

Figure 5: Fusion of range-bearing sensor data using weighted Interpolation

3.4. Machine Learning Models

3.4.1. Structure of the Data

The dataset created in Section 3.3 contains 1000 rows. Each input and output column contains cartesian coordinates of a shape (x,y), as depicted in **Table 2.**

Row	**Input 1**		**Input 2**		**Result**	
	x	**y**	**x**	**y**	**x**	**y**
1	**x1,x2,...x230**	**y1,y2,...y230**	**x1,x2,...x230**	**y1,y2,...y230**	**x1,x2,...x230**	**y1,y2,...y230**
2	**x1,x2,...x230**	**y1,y2,...y230**	**x1,x2,...x230**	**y1,y2,...y230**	**x1,x2,...x230**	**y1,y2,...y230**
…	**…**	**…**	**…**	**…**	**…**	**…**
1000	**x1,x2,...x230**	**y1,y2,...y230**	**x1,x2,...x230**	**y1,y2,...y230**	**x1,x2,...x230**	**y1,y2,...y230**

Table 2: Initial Data Structure for Machine Learning

From **Table 2**, the data structure is **1000 by 2 by 230**. The first 900 rows were used for training ML models, and the remaining 100 were used for inference.

Before training, the data was reshaped into **2 by (230 by 900)** or **2 by 207000**, as shown in **Table 3**.

Row	**Input 1**		**Input 2**		**Result**	
	x	**y**	**x**	**y**	**x**	**y**
1	**x1**	**y1**	**x1**	**y1**	**x1**	**y1**
2	**x2**	**y2**	**x2**	**y2**	**x2**	**y2**
…	**…**	**…**	**…**	**…**	**…**	**…**
207000	**x207000**	**y207000**	**x207000**	**y207000**	**x207000**	**y207000**

Table 3: Data Structure Used for Training ML Models

3.4.2. Training AND Evaluation Of Machine Learning Models

After restructuring the data, training the ML is relatively straightforward. For GAM, two models were trained, one for estimating the x coordinate and the other for estimating the y coordinate of the result. GAM was implemented in the Python script <4 GAM.py> using PyGAM. On the other hand, GAN was implemented in <5 GAN.py> using two fully connected feed-forward neural networks, one for the generator and the other for the discriminator. Lastly, the CNN was implemented in <6 CNN.py>.

4. Results.

The performance of each model is illustrated in **Table 4**.

Model	5 - Fold Cross Validation MSE	Inference MSE on 100 inputs
GAM	0.2348 ± 0.0014	0.1592
GAN	1.1782 ± 0.1847	1.0475
CNN	0.0727 ± 0.0013	0.0964

Table 4: Performance of GAM, GAN, and CNN on Fusion of simulated Range-Sensor Data

The results in **Table 4** show that the CNN achieves the highest performance, followed by the GAM, while the GAN performs the least effectively. It is important to note that the ML models were trained using a single set of hyperparameters without an exhaustive hyperparameter search. This limitation may have introduced potential biases in the comparison of the models.

5. Conclusion

This project explored the feasibility of machine-learning approaches for sensor fusion in robot localization using simulated range-bearing sensor data. Generalized Additive Models (GAM), Generative Adversarial Networks (GAN), and Convolutional Neural Networks (CNN) were trained and evaluated on a synthetic dataset. The results demonstrated that CNN outperformed both GAM and GAN in terms of Mean Squared Error (MSE) across cross-validation and inference tests, highlighting its superior ability to handle the nonlinear and high-dimensional data inherent in sensor fusion tasks. The GAM model achieved moderate performance, showing its capability to capture some of the nonlinear relationships in the data but falling short compared to CNN. On the other hand, GAN showed the weakest performance, which may be attributed to its complex dynamics and tendency to collapse. This work underscores the potential of deep learning, particularly CNNs, in advancing sensor fusion applications. However, it also highlights key challenges, including the importance of hyperparameter tuning and the need for balanced model comparisons. Future research should focus on optimizing hyperparameters, exploring more robust architectures, and choosing more suitable models for fusion.

REFERENCES

[1] P. K. Panigrahi and S. K. Bisoy, “Localization strategies for autonomous mobile robots: A review,” *Journal of King Saud University - Computer and Information Sciences*, vol. 34, no. 8, Mar. 2021, doi: https://doi.org/10.1016/j.jksuci.2021.02.015.

[2] G. Huang, A. I. Mourikis, and S. I. Roumeliotis, “Analysis and improvement of the consistency of extended Kalman filter based SLAM,” May 2008, doi: https://doi.org/10.1109/robot.2008.4543252.

[3] Q. Tang, J. Liang, and F. Zhu, “A Comparative Review on Multi-modal Sensors Fusion based on Deep Learning,” pp. 109165–109165, Jul. 2023, doi: https://doi.org/10.1016/j.sigpro.2023.109165.

[4] S. Mahfouz, F. Mourad-Chehade, P. Honeine, J. Farah, and H. Snoussi, “Target Tracking Using Machine Learning and Kalman Filter in Wireless Sensor Networks,” *IEEE Sensors Journal*, vol. 14, no. 10, pp. 3715–3725, Oct. 2014, doi: https://doi.org/10.1109/jsen.2014.2332098.

[5] X. Gao, D. You, and S. Katayama, “Seam Tracking Monitoring Based on Adaptive Kalman Filter Embedded Elman Neural Network During High-Power Fiber Laser Welding,” *IEEE Transactions on Industrial Electronics*, vol. 59, no. 11, pp. 4315–4325, Nov. 2012, doi: https://doi.org/10.1109/tie.2012.2193854.

[6] I. J. Goodfellow *et al.*, “Generative Adversarial Networks,” *arXiv.org*, Jun. 10, 2014. https://arxiv.org/abs/1406.2661

[7] [1]K. O’Shea and R. Nash, “An Introduction to Convolutional Neural Networks,” *arXiv:1511.08458 [cs]*, vol. 2, Dec. 2015, Available: https://arxiv.org/abs/1511.08458