International Journal of Engineering & Technology, *7 (4) (2018) 2964-2970*

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET
doi: 10.14419/ijet.v7i4.12472
Research paper

Robotic sensor data analysis using stream data mining techniques

Radhakrishnan Gopalapillai [1] *, Deepa Gupta [2], Sudarshan, T. S. B [3]

[1] Assistant Professor, Department of Computer Science & Engineering, Amrita School of Engineering, Bengaluru, Amrita Vishwa Vidyapeetham, India
[2] Associate Professor, Department of Mathematics, Amrita School of Engineering, Bengaluru, Amrita Vishwa Vidyapeetham, India
[3] Professor, Department of Computer Science & Engineering, PES University, Bengaluru, India
**Corresponding author E-mail: g_radhakrishnan@blr.amrita.edu*

Abstract

Many robotic applications deploy multiple robots and it is possible that more than one of those robots are operating in the same environment. Such situations demand grouping together of similar environments in real-time to perform actions in a coordinated way. The main challenge when robots sent huge amount of data is to process the data stream without storing them. In this work, an experimental setup is created to gather data from simulated robotic environments. The data collected are treated as continuously arriving time series data and they are com-pressed using summary data structures suitable for clustering. The robotic environments are clustered using techniques based on simple single pass K-means and StreamKM++ algorithms. The methods used to adapt these two algorithms for robotics data streams are discussed. The suitability of these techniques for robotic applications is analyzed and performances of the algorithms are compared.

Keywords: *Clustering Algorithms; Data Mining; Robotic Environment; Sensor Data; Stream Data.*

1. Introduction

Robots are used in many applications to do tasks that are normally performed by humans. They are particularly used to automate mundane tasks or to perform tasks that are risky or dangerous for a human to perform. For example, mobile robots are used in warehouses for efficient movement of materials. A key component of robotic applications is the mechanism to understand the environment in which the robot is deployed to perform particular task. In many robotic applications, multiple robots surveying the environment produce huge amount of sensor data which may be streamed to a central computer for processing. The data collected are typically multi-modal high dimensional data.

Stream data mining has developed as an independent research area that can be applied to any domain where data are continuously arriving. Advances in wireless communication led to the development of low-power sensors. It is utilized in numerous sensing automation tasks such as temperature monitoring, humidity and surveillance. Sensor networks are used in many application domains, such as object tracking, environment monitoring, disaster management as well as smart environments. In these applications, reliable monitoring is an essential requirement for the information that are continuously arriving from sensors and camera attached to unmanned vehicles flying in unknown territory.

Clustering techniques are widely utilized as part of machine learning applications to compress substantial amounts of high-dimensional information to summarized data that are valuable for a particular application. Streams often deliver elements rapidly and once the data is processed, it is not viable to store the data and hence not accessible for further processing. Stream data mining algorithms do incremental processing of data in real time as the data is not available for iterative reading. They are often executed in main memory, without access to secondary storage or with limited passes of data stored in secondary storage. These issues in stream data clustering are to be taken into account for effective clustering.

In many applications, it is important to analyze the data collected and transform it into usable information through stream data mining techniques so that the processed information can be used later for decision making. The aim of this work is to record useful information from robotic environments that produces streams of data and to cluster them using stream clustering algorithms. An exploratory robotic environment has been designed where robot is customized to move in a straight path with a consistent pace to record data using sensors attached to it. An arrangement of distinctive exploratory environments, which incorporates hot and cold objects of different shapes and sound sources kept around the robot path, is outlined. The data collected by the robot are then used to cluster the environments using stream data mining algorithms.

This paper is organized as follow: Section 2 explains past work related to robotic environment and stream data mining. Section 3 describes the overall architecture of the work focusing on design of robotic environment, data collection and stream clustering algorithms. This section discusses two stream data mining algorithms, simple single pass K-means [1] and streamKM++ [2] that have been adapted to cluster the data gathered. Experimental results obtained from the two algorithms are shown in Section 4. The results are analyzed and compared in terms of accuracy of clustering and computation time. Conclusions from the analysis and directions for future work are discussed in Section 5.

2. Related work

Data mining on time series data have been explored in the past [3]. Dynamic Time Warping (DTW) distance based clustering of time series data collected by mobile robots have shown good results [4]. Complete linkage clustering algorithm has been applied in that work and the accuracy of results obtained through this method fall in the range of 88 to 97%. Clustering of robotic scenarios using data gathered with simple infrared sensors have shown good accuracy when the scenarios are simple and distinct [5]. The clustering performed using agglomerative clustering techniques reported accuracy in the range of 73 to 98%. The work discussed in [6] explains how back propagation neural network (BPNN) has been used for classification of robotic environments using time series data gathered using sonar sensors. Another experiment [7] that is based on similar robotic environment summarizes the experimental setup for creating a virtual robotic environment. The robotic environments are then clustered using the data acquired from IR proximity sensors and thermal sensors. K-medoid clustering algorithms are used for clustering the sensor data and it is observed that the clustering accuracy is in the range of 75-100%. Clustering is done using traditional offline algorithms which requires multiple pass over the data. The experiments on robotic environment described earlier [3-7] focused on clustering the complete dataset that is already gathered using multiple passes of the data.

Jacqueline Heinerman, Evert Haasdijk and A.E. Eiben introduced Context Recognition in Data Streams (CoRDS), a method that enables a robot to identify and recognize different situations in its environment [8]. CoRDS uses environment data collected using robot's sensors. .Sabarish B.A et.al. explored clustering of large volume of spatiotemporal data generated from GPS enabled devices such as smartphones, cars, sensors, and social media [9]. They clustered trajectories using hierarchical clustering method using Dynamic Time Warping (DTW) distance measure.

Jonathan A. Silva, et all [10] have done a survey of data structures and algorithms used for clustering data streams. According to them, important aspects of data stream clustering are data structure for statistical summary, number of user-defined parameters, cluster shape and type of clustering problem. Since stream data do not provide the entire dataset at initial point, stream clustering algorithms do the clustering incrementally. Barbara [11] explores the requirements that are needed for data stream clustering. The challenge is to design an algorithm that can take care of changes incrementally within the available memory and time.

In data stream clustering techniques, the clustering step uses an appropriate summary data structure to store statistical summaries and then a standard clustering algorithm is used to find clusters The structures used for summarizing data and the specific clustering techniques used may vary from one stream data mining algorithm to another. Most of the algorithms use K–means for clustering the data. One of the popular algorithms to mine large dataset is BIRCH [12] that builds a hierarchical data structure using a balanced tree to incrementally capture clustering features of incoming points. This algorithm focusses on performing the clustering within the constraints of available memory by minimizing the input data points required. Aggarwal et.al. developed Clustream algorithm to handle data streams that have evolving characteristics [13]. A Hierarchical algorithm called ODAC [14] maintains a tree-like hierarchy of clusters based on variables. It manages concept evolution using a combination of both divisive and agglomerative hierarchical clustering.

K-means algorithm is probably the best known algorithm for data clustering where the objects to be clustered have numerical attributes. The scalability issues in K-means algorithm when applied to stream data are addressed by Farnstrom et al. [1] who used compression-based techniques of Bradley et al. [15] to obtain a single-pass algorithm, but they have not mentioned any method to initialize K-means. K-means++ algorithm describes a better technique to fix initial cluster means [16]. Clustering of distributed data stream using STREAMLS [17] is an extension to Bradley's techniques that keeps the same goal but has restriction on use of available memory and buffer. Once the input buffer is filled, STREAMLS creates k interim clusters with the data in buffer and retains only centroids of the clusters formed weighted by the number of data in each cluster. This process is repeated with new points. Stream KM++ used for clustering stream data of very large dataset uses the concept of coreset tree to summarize the data that are arriving and then summarized data is clustered using K-means++ algorithm. Nair, PC et.al discusses an experimental study on the implementation of StreamKM++ to effectively cluster time series robotic image data with memory restrictions [18]. Their work required extraction of features from images as a pre-processing step.

The research work discussed in previous sections focused either on offline clustering of robotic environments or stream clustering on generic data. This work on focusses on adapting stream clustering techniques for robotic data to cluster robotic environment. The objective of experiments discussed in this paper is to cluster robotic environments using data collected by inexpensive sensors. Though the data set collected is small enough to fit in main memory, stream mining techniques employed in this work will be scalable for large datasets where multiple passes of the data is not feasible. This work adapts two stream data clustering algorithms – simple single pass k-means algorithm and streamKM++ algorithm and analyzes the suitability of adapted methods to cluster the data objects collected from indoor robotic environments specifically designed for the experiment. The details of the architecture used for the experiment is discussed in the next section.

3. Architecture

The overall architecture of the proposed work is shown in the Figure 1. First, information is gathered from experimental robotic environments using sensors attached to robots. These were indoor robotic environments that had objects of difference shapes and sizes. A wheeled robot mounted with different sensors was used to collect data about the environment from different locations. The data gathered at different was stored and later processed as if these data were coming as data stream. Since the study was on techniques for clustering environments, complexities involved in real time data acquisition and transmission was not focused. The information gathered is converted into a format that is useful for processing and the features are extracted and compressed. Then the stream data mining algorithms are applied on the extracted information. Though the data collection is done in an offline mode, techniques used here can be applied to stream data very well.

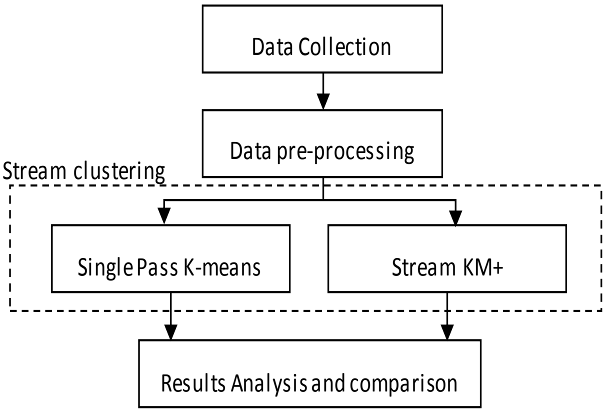


Fig. 1: High Level Architecture of the Clustering Process.

3.1. Design of experimental scenarios

The experimental setup is done to simulate different robotic environments. In this work, a set of seven different indoor robotic environments are created and data collected using sensors attached to robots exploring these environments or scenarios. The goal was to create a few scenarios which look different from each other.

Each robotic environment designed has objects of different shapes such as spherical, rectangular, cone or pyramid placed at different locations. The positions of the objects in a particular scenario are altered to create a new scenario. In order to make the data multimodal, a few of these objects are designed for having different thermal profile. Couple of audio sources are also added to the environment though they are not used in this study.

Each of the scenarios typically contain 32 objects - 24 non-thermal objects, 4 objects that are hot and 4 objects that are cold placed in a square area of 210cm by 210cm. The cold and hot objects have been kept at a constant temperature between 15°c and 70°c. The ambient temperature during experiment was close to 30°c. The scenarios fall into three distinct types. The first type of scenarios is uniformly dense; i.e. all objects are uniformly distributed throughout the environment. Another set of scenarios have more objects in one quadrant and very few objects in another quadrant. The third type of scenarios have more concentration of thermal objects in one quadrant. A photograph of one of the environments without thermal objects is shown in Figure 2.

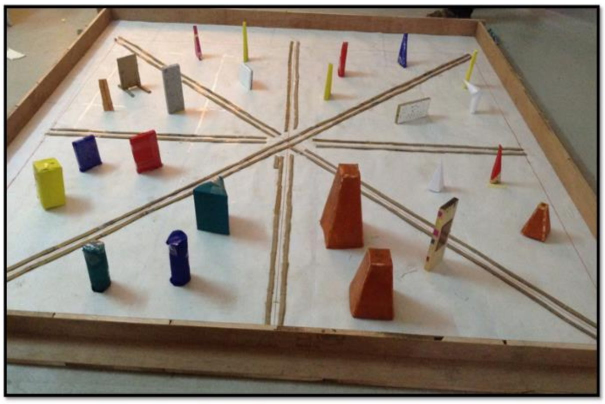


Fig. 2: Photograph of an Experimental Environment.

The experimental environment shown schematically in Figure 3 represents one of the scenarios where objects are distributed evenly throughout the environment. The thick horizontal and diagonal lines seen in the diagram are the straight paths used by the robot to explore the environment.

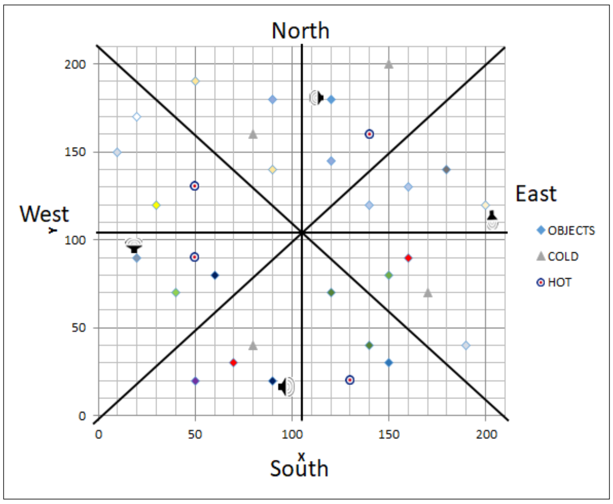


Fig. 3: Environment Where Objects are Evenly Distributed.

Figure 4 shows a scenario where objects are not distributed uniformly. In this scenario, there are more objects in the south-east quadrant and north-west quadrant. Scenario shown in Figure 5 has objects kept closer to diagonal path compared to horizontal and vertical paths.

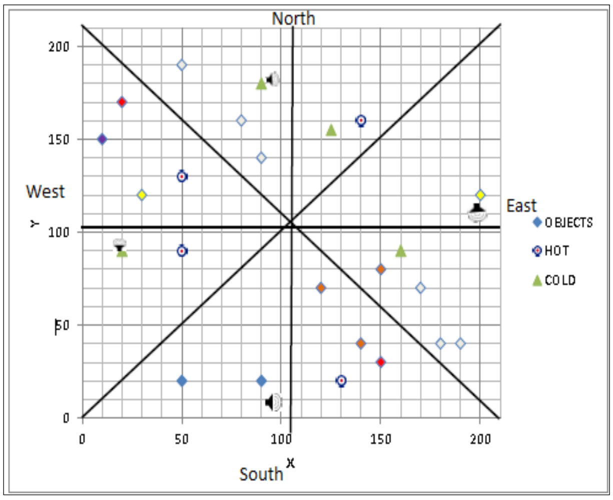


Fig. 4: Scenario with Uneven Distribution of Objects.

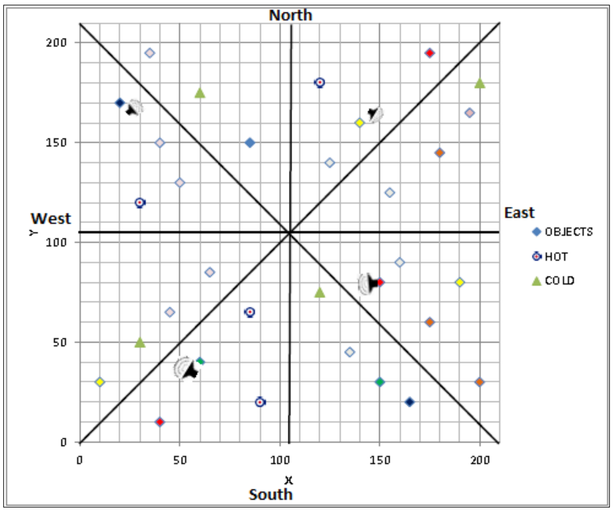


Fig. 5: Scenario with Objects Closer to Diagonal.

The robot is programmed to move in a single straight line path at constant speed to explore the environment. Robot moves in straight paths from east to west, north to south, west south corner to north east corner or north west corner to east south corner and vice versa. Robot conducted 1000 trails or explorations along each of the predefined paths in a single environment. For example, in one such exploration, robot will start from south west corner of the environment and travel towards the north east corner. The readings of on-board sensors like long range IR sensors, PIR thermal imaging sensor of resolution 4 x 4 and microphones are taken every 200 milliseconds. The data collection process was repeated in other six environments.

3.2. Characteristics of data

The robot is mounted with two thermal sensors, four IR distance sensors and audio sensors. Figure 6 shows the approximate locations where the IR and thermal sensors are mounted on the robot.

The thermal sensor is a low cost 4 x 4 pixels thermal camera which captures the thermal profile of the environment. The reading of the thermal sensor is calibrated to give temperatures in degree centigrade. Four attributes are collected from IR sensors and 16 attributes are collected from each of the two thermal sensors mounted on the right and left side. Thus the data gathered at a particular instance have 36 attributes.

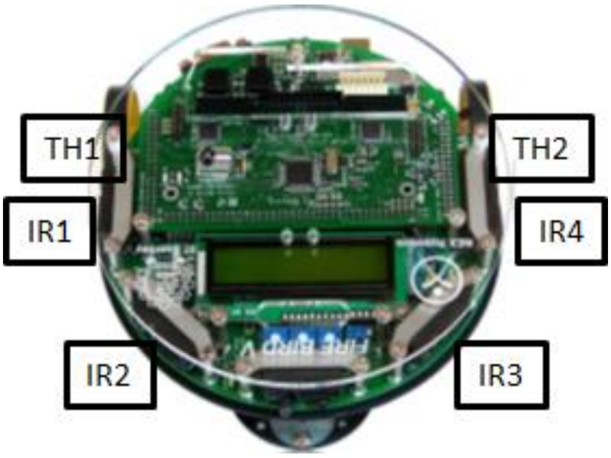
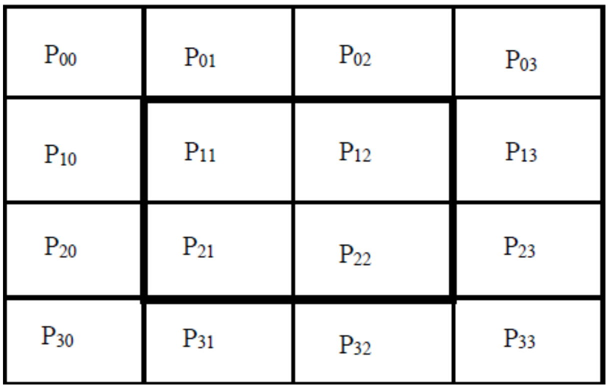


Fig. 6: Sensors Mounted on the Robot.

3.3. Data pre-processing

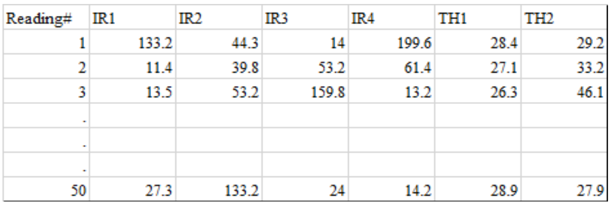
In order to reduce the amount of data passed to the clustering algorithm, the thermal image consisting of 16 pixels has been converted to a single attribute by taking the average of the pixel values of the central region of 2x2 pixels as shown in Figure 7. The pixels are marked as P00 to P33. The average values of pixels P11, P12, P21 and P22 are taken and the values of other pixels are ignored. Hence the data for each snapshot of the environment consisted of four distance attributes and two thermal attributes.



P_{00}	P_{01}	P_{02}	P_{03}
P_{10}	P_{11}	P_{12}	P_{13}
P_{20}	P_{21}	P_{22}	P_{23}
P_{30}	P_{31}	P_{32}	P_{33}

Fig. 7: Pixel Matrix of Thermal Sensor.

The complete set of data attributes collected is considered as a single object. Robot takes about 10 seconds to explore the environment in each direction. Since the sensor readings are taken every 200 milliseconds, each run of the robot captures 50 snapshots of the environment consisting of 6 attributes per snapshot. Typical data collected from the environment during a single trial are shown in Figure 8.



Reading#	IR1	IR2	IR3	IR4	TH1	TH2
1	133.2	44.3	14	199.6	28.4	29.2
2	11.4	39.8	53.2	61.4	27.1	33.2
3	13.5	53.2	159.8	13.2	26.3	46.1
.						
.						
.						
50	27.3	133.2	24	14.2	28.9	27.9

Fig. 8: Data Corresponding To a Single Trial.

The data collected from an environment is stored as the attributes of a single object for the purpose of clustering. Hence each environment is represented by a single object of 300 attributes where each attribute is numerical.

3.4. Data stream clustering method

As discussed earlier, the experiment consists of 7 different scenarios. The dataset created has 7000 data objects with each data object representing one of the seven scenarios. Each object can be considered as a labelled object with object label being the scenario from which that object is created. This label is removed during the clustering process. However, it has been used to calculate the accuracy of clustering obtained. The objects are taken uniformly at random from the scenarios. The objective of the clustering is to group together similar scenarios into seven clusters corresponding the seven experimental environments created.

In this experiment, two algorithms, simple single pass k-means and streamKM++ have been adapted for robotic data. Simple single pass k-means is an efficient algorithm that gives results comparable to other data stream clustering algorithms. StreamKM++ is reported to give better clustering results at the cost of computational efficiency. A brief description of the simple single pass k-means and streamKM++ algorithms and how they have been applied to cluster robotic data is discussed in the next section.

3.4.1. Simple single pass k-means algorithm

The simple single pass k-means algorithm [1] is an extension of the single pass k-means algorithm [12] which is used for clustering stream data. It uses primary and secondary compression techniques and then merges both to obtain the result. The simple single pass k-means need very less computation compared to all other stream data clustering algorithm. The k-means algorithm initializes the cluster means and performs clustering by assigning each point to the nearest cluster mean. k-means algorithm depends strongly on the initial set of centers. The algorithm uses a buffer area to store partial unprocessed data read from the data stream. Each data element in the buffer is clustered using standard k-means algorithm. Here Euclidean distance has been used to calculate the distance between two data objects. Once the clusters are formed with partial data, each cluster is stored as a new representative object and all the elements in the cluster are discarded. The simple single pass k-means algorithm used in this work is given in Figure 9.

```
Algorithm 1: Simple single pass k-means
1)  Read the data from the input stream until buffer is full
2)  Initialize the weight (w) of each data to 1.
3)  Select the k cluster means using modified random selection method.
4)  Assign each of data to the nearest cluster mean.
5)  Recalculate the cluster mean, taking into account the weight of each
    point.
6)  If there is change in cluster mean, go to step 3.
7)  If no more data in the stream, go to step 12
8)  Empty the buffer.
9)  Represent each cluster as a single data object and insert each repre-
    sentative object in to the buffer.
10) Fill the remaining space in the buffer by reading next set of stream
    data.
11) Go to step 3.
12) Output the cluster centers.
```

Fig. 9: Simple Single Pass K-Means Algorithm.

The initial cluster mean is chosen randomly, in most of the case, but if it is randomly chosen there may be chance of selecting more than one cluster mean from the same cluster that leads to more number of iteration while performing K-means which reduces the performance of the algorithm. In order to achieve better result, initial set of cluster centers are chosen using modified random selection method as follow.

i) Select a data point randomly from the available dataset as first cluster mean.
ii) Subsequent cluster mean is selected by computing distance of each of point to the mean already selected. The minimum distance to any of the cluster already selected is taken as the distance of object.
iii) The object that has maximum distance is selected as the cluster mean.
iv) Repeat this procedure until K cluster means are obtained.
v) An attribute vector and a weight w represent each object in a cluster. The sufficient statistics of a cluster are Sum, the

weighted sum of object attributes and N, the number of objects in the cluster. When an object Xi is added to a cluster Cj, sufficient statistics of the cluster are updated using the formulae given in equations (1) and (2).

$$\mathrm{Sum}^{(Ci)} = \mathrm{Sum}^{(Ci)} + WI*X_i \qquad (1)$$

$$N^{(Ci)} = N^{(C1)} + 1 \qquad (2)$$

Once the clusters are formed with objects in the buffer, a representative object is created for each cluster and buffer is cleared. The representative objects will have corresponding cluster means as their attributes. The weight of a representative object is the number of objects in the cluster.

3.4.2. Stream-KM++ algorithm

StreamKM++ algorithm uses a coreset tree to merge and reduce the data objects. The data objects in the data stream are represented as O_1, O_2,…,O_n. A structure called bucket is used to store the summarized data from the robotic data stream. Buckets are labelled as B_0, B_1, … B_k. Each bucket can store m data objects. At any point of time, bucket B_0 can store objects numbering between zero to m whereas other buckets contain either zero or exactly m objects. The number of buckets required store n data objects from the robotic stream is of the order of O(log n). The i^{th} bucket B_i contains summarized information for 2^{i-1}m points from the data stream

The Merge and reduce process is given in Algorithm 2 shown in Figure 10. The merge and reduce step uses the coreset tree construction process described in Algorithm 3, shown in Figure 11 to generate representative points. Every node in the coreset tree contains a representative point q, a set of objects S, and the cost of the node. The cost of a node is computed as the sum of squared distance of all objects in the node to its representative point. When the algorithm starts, corset tree contains just one node, i.e., the root node that has all objects to be merged and reduced.

```
Algorithm 2: Merge and Reduce
1)  Let $B_0$, $B_1$, $B_k$ be initially empty buckets
2)  While $B_0$ is not full
3)  insert points to $B_0$
4)  k = 1
5)  Create an empty bucket S and move contents of $B_0$ to S
6)  Empty $B_0$
7)  While $B_k$ is not empty
8)  Merge & reduce S and $B_k$ using Coreset Tree Construction & store
    in S
9)  Empty $B_k$
10) k = k+1
11) Move data from S to $B_k$
12) If data is available in data stream, goto step 2
```

Fig. 10: Merge and Reduce Algorithm.

```
Algorithm 3: Coreset Tree Construction
1)  Create a root node with 2m objects to be merged. Let the node to be
    split is denoted as cnode. Let representative object of a node i is de-
    noted as $q_i$.
2)  Select a representative object $q_{root}$ from root node at random.
3)  cnode = root
4)  $q_{Cnode}$ = $q_{root}$
5)  $q_{new}$ = select(cnode,$q_{cnode}$)
6)  Create a left child for cnode with $q_{cnode}$ as rep. point
7)  Create a right child for cnode with $q_{new}$ as rep. point.
8)  Rearrange points of cnode to left and right children
9)  Update cost of node as cost of left child +cost of right child
10) If the number of leaf nodes in the tree is less than m
11)   current = root
12)   While current is not leaf
13)   current=select_node(current)
14)   cnode=current
15)   Goto step 5
16) Form coreset with representative object of all leaf nodes
17) Wieght(qi) = number of points in the ith leaf node
```

Fig. 11: Corset tree construction algorithm.

Tree construction is achieved in a top-down approach by splitting nodes. Let the current node to be split is denoted as cnode and let qcnode be the representative point. A procedure select(cnode,qcnode) chooses a new representative point from cnode randomly with probability proportional to the sum of squares of Euclidian distances of all objects in cnode to qcnode. Procedure select_node(current) selects one of the children nodes of the current node to reach a leaf node based on PPC where PPC is the Probability Proportional to Cost of each of the child node.

Experimental results and analysis

In this section, the experimental results for the clustering of data streams are discussed. The simple single pass K-means and streamKM++ algorithm results are analyzed to find their effectiveness in clustering robotic environment using sensor data gathered. Each algorithm uses its own specific structure to store stream data that is converted into some data structure and a summarizing procedure to reduce the stream data that is already processed.

As discussed in section 3, data collected pertained to seven robotic scenarios. The robot has made 1000 explorations in each direction within each scenario. Hence there are 7000 data objects corresponding to the trails made in each direction. If these 7000 objects are clustered to form seven clusters, each cluster should correspond to one of the seven robotic environments. Each cluster should ideally contain 1000 objects related to the particular environment associated with that cluster. The objects allocated to each cluster are then checked to see whether they are assigned to the right cluster. The confusion matrix shown in Table 1 captures the result of clustering of the data taken during the exploration of environments from north to south direction using simple single pass algorithm. The seven environments are marked as S1 to S7 in the confusion matrix.

The confusion matrix shows that five of the clusters had 100% correct memberships. Objects belonging to scenario 3 got assigned to two different clusters - clusters 3 and 6. 119 objects belonging to scenario 3 were incorrectly assigned to cluster 6. Hence the overall accuracy of clustering was computed as the ratio of 6881 to 7000.

Table 1: Confusion Matrix of Clustering of Trials in North to South Direction Using Simple Single Pass Algorithm

		Predicted Scene (cluster)						
		S1	S2	S3	S4	S5	S6	S7
Actual Scene	S1	1000	0	0	0	0	0	0
	S2	0	1000	0	0	0	0	0
	S3	0	0	881	0	0	119	0
	S4	0	0	0	1000	0	0	0
	S5	0	0	0	0	1000	0	0
	S6	0	0	0	0	0	1000	0
	S7	0	0	0	0	0	0	1000

One of the parameters that affect the accuracy of clustering and space requirement is the buffer/bucket size used in the algorithm. The effect of buffer size on clustering accuracy is explored by varying the buffer size. In this experiment, bucket sizes varying from 100 to 700 objects have been used. Figure 12 shows the results of simple single pass K-means and streamKM++ algorithms with different buffer/bucket sizes.

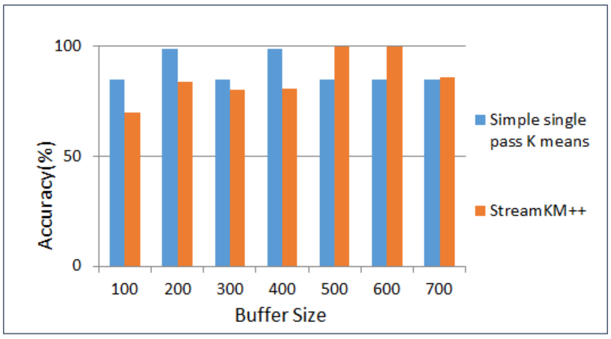


Fig. 12: Accuracy of Algorithms for Different Bucket Sizes for Path1 – North South.

The average accuracy of clustering with data collected from north to south and south to north directions of robotic environment is shown in the figure. It is observed that the accuracy of simple single pass K-means is higher for smaller buffer sizes compared to streamKM++. The computation time taken by the algorithms are shown in Figure 13. It is observed that simple single pass algorithm takes less computation time compared to streamKM++ algorithm. The computation time taken by simple single pass K-means is close to one fifth of the time taken by streamKM++.

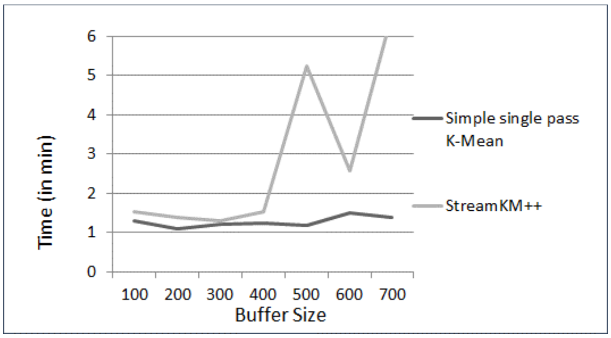


Fig. 13: Computation Time of Algorithms for Path1 – North South.

Similarly, Figure 14 shows the average results of clustering of trials conducted in west to east and east to west directions. It is observed that the streamKM++ produces better results

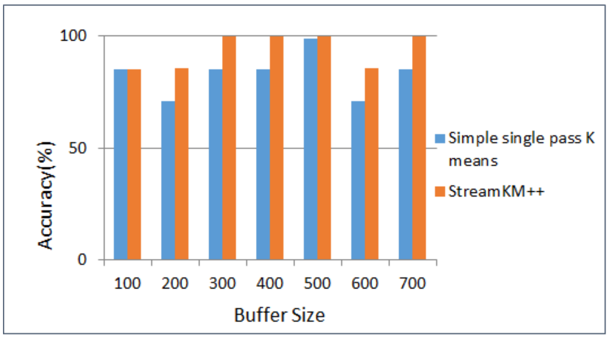


Fig. 14: Accuracy of Algorithms for Different Bucket Sizes for Path2 – East West.

Figure 15 shows the respective computation time and it is clearly seen that simple single pass K-means takes less time compared to streamKM++ for varying buffer size as in the previous case.

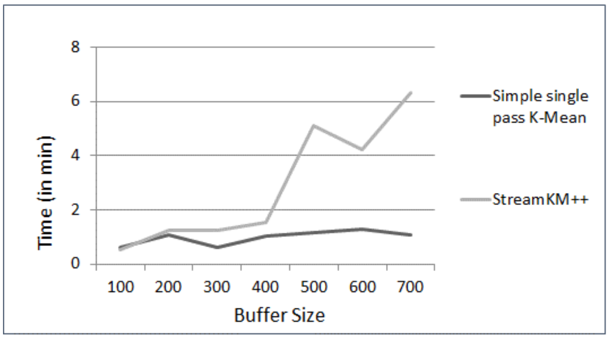


Fig. 15: Computation Time of Algorithms for Path2 – East West.

When bucket size of streamKM++ is increased, the time taken to process the data has not shown significant differences with data taken in both directions. It is observed that the algorithm for the sensor data clustering becomes less efficient as the bucket size increases even though there is an increase in accuracy up to bucket size of 500. Hence a bucket size of 500 seems to be good for the data considered. StreamKM++ algorithm is more efficient when the bucket size and number of buckets needed to process stream data is initially calculated based on the size of dataset. Since the data from the sensor is arriving like a stream, the size of dataset is unpredictable. Simple single pass K-means algorithm produces less accurate clusters but uses less computation time compared to streamKM++ algorithm. Selection of buffer size beyond 500 has not shown improvement in the accuracy of clustering for sensor data obtained from the robotic environment in most of the cases.

4. Conclusion and future work

This work summarizes the design of robotic environments, a data collection setup to collect data about the environment, and clustering of environments based on similarity. The data collected have been processed and the data has been read as if a stream of data is arriving. The robotic environments have been clustered using methods based on simple single pass K means and streamKM++ algorithms. The result obtained by simple single pass K means showed accuracy in the range of 71% for a bucket size of 100 to 100% for a bucket size of 500. Accuracy of single pass K means algorithm is less than the accuracy given by streamKM++ algorithm for bucket sizes less than 500. However, simple single pass algorithm takes lesser computation time. StreamKM++ algorithm gives better clustering results at the cost of higher computation time. The accuracy of clustering varies with bucket size used in these algorithms. More experiments are needed to arrive at an empirical relationship between bucket size and the expected number of clusters and data set size.

The future work includes the clustering of sensor stream data from a real world application and clustering the data that is arriving from different direction of the robotic environment.

References

[1] Farnstrom, F. Lewis, J. In addition, Elkan, C. Scalability for Clustering Algorithms Revisited, SIGKDD Exploration Newslett. 2, 1, 2000, pp. 51–57.

[2] Ackermann,M. Martens, R. Raupach, M. Swierkot, C. Lammersen, K. and Sohler, C. 2012. StreamKM++: A clustering algorithm for data streams. ACM J. Exper. Algor.17, 1. https://doi.org/10.1145/2133803.2184450.

[3] Nair, B B. Kumar, P.K.S. Sakthivel, N.R. Vipin, U, Clustering stock price time series data to generate stock trading recommendations: An empirical study", Expert Systems with Applications, Vol. 70, pp. 20-36, March 2017 https://doi.org/10.1016/j.eswa.2016.11.002.

[4] Radhakrishnan, G. Gupta, D. Abhishek, R. Ajith, A. Sudarshan, T.S.B. Analysis of multimodal time series data of robotic environment. Proceedings of 12th International Conference on Intelligent Systems Design and Applications (ISDA), Kochi, India, pp. 734-739, 2012.

[5] Radhakrishnan, G. Gupta, D. TSudarshan,T.S.B.,Experimentation And Analysis Of Time Series Data For Rescue Robotics, Proceedings of 2nd International Symposium On Intelligent Informatics (Ist'13), Mysore, India, pp.443-453, 2013.

[6] Gopalapillai, R. Vidhya, J. Gupta, D. Sudarshan, T.S.B. Classification of robotic data using artificial neural network, Proccedings of IEEE Recent Advances Intelligent Computational Systems (RAICS), Trivandrum, India, pp.333-337, 2013.

[7] Mishra, S. Radhakrishnan, G. Gupta, D. Sudarshan T.S.B., Acquisition and Analysis of Robotic Data Using Machine Learning Techniques, Computational Intelligence in Data Mining - Volume 3 Smart Innovation, Systems and Technologies Volume 33, 2015, pp 489-498.

[8] Jacqueline Heinerman, Evert Haasdijk and A.E. Eiben, Unsupervised identification and recognition of situations for high-dimensional sensori-motor streams, Neurocomputing, Vol. 262, 1 November 2017, pp. 90-107. https://doi.org/10.1016/j.neucom.2017.02.090.

[9] Sabarish B.A, Karthi R, Gireeshkumar T.B, Clustering of trajectory data using hierarchical approaches, Lecture Notes in Computational Vision and Biomechanics, Vol. 28, 2018, pp. 215-226 https://doi.org/10.1007/978-3-319-71767-8_18.

[10] Jonathan A. Silva, Elaine R. Faria. Data Stream Clustering: A Survey, ACM Computing Surveys, 2013 Vol. 46, No. 1, Article 13.

[11] Barbara. Requirements of Clustering Data Streams, SIGKDD Explorations (2002) 3(2):23-27. https://doi.org/10.1145/507515.507519.

[12] Zhang Et Al. Birch: An Efficient Data Clustering Method for Very Large Databases. ACM SIGMOD: (1996) 103-110.

[13] Aggarwal, C.C. Han, J. Wang, J. In addition, Yu, P.S. A framework for clustering evolving data streams. In VLDB 2003, Proceedings of 29th International Conference on Very Large Data Bases, pages 81–92.

[14] Rodrigues, P. P., Gama, J., And Pedroso, J. P,.Hierarchical Clustering Of Time-Series Data Streams. IEEE Trans 2008, Knowl. Data Engin 20, 5, 615 –627.

[15] Bradley, P. S., Fayyad, U. M., And Reina, C, Scaling clustering algorithms to large databases. Proceedings of the fourth International Conference on Knowledge Discovery and Data Mining (KDD'98). 1998.

[16] Arthur, D. and Vassilvitskii, S. (2007). K-means++: the advantages of careful seeding. Proceedings of the eighteenth annual ACM-SIAM symposium on discrete algorithms. Society for Industrial and Applied Mathematics Philadelphia, PA, USA. pp. 1027–1035.

[17] L. Callaghan, N. Mishra, A. Meyerson, S. Guha, and R. Motwani, Streaming-Data Algorithms for High-Quality Clustering, Proceedings of IEEE International Conference on Data Engineering, 2001, pp. 685-694.

[18] Priyanka C.Nair, Radhakrishnan G, Deepa Gupta, Sudarshan TSB, Clustering of Robotic Environment using Image Data Stream, Proceedings of the IEEE International Conference on Communication Control and Intelligent System (CCIS-2015), Mathura, India, 2015, pp. 208-213.