$$\Delta x = \Delta d * \cos\left(\theta + \frac{\Delta\theta}{2}\right) \tag{2}$$

$$\Delta y = \Delta d * \sin\left(\theta + \frac{\Delta\theta}{2}\right) \tag{3}$$

The relative displacement Δd and orientation $\Delta\theta$ of the mobile platform between two steps are given as in Eqs. 5 and 4, respectively.

$$\Delta\theta = \frac{\Delta\text{dl} - \Delta\text{dr}}{B} \tag{4}$$

$$\Delta d = \frac{\Delta\text{dl} + \Delta\text{dr}}{2} \tag{5}$$

where Δdl and Δdr stand for the relative displacement of the left wheel and the right wheel of the mobile platform, respectively, which are measured by IR sensor used as encoders [14].

Δx, Δy, $\Delta\theta$	Mobile platform travelled position after Δt sampling interval
Δdl, Δdr	Travelled distances by left and right wheel, respectively
B	Distance between the two wheels of mobile platform
$\Delta\theta$	Heading angle of robot measured using IMU

If Δdl is not equal to Δdr, then

$$\begin{aligned} x_{i+1} &= x_i + \Delta x \\ y_{i+1} &= y_i + \Delta y \\ \theta_{i+1} &= \theta_i + \Delta\theta \end{aligned}$$

If Δdl is equal to Δdr, then

$$\begin{aligned} x_{i+1} &= x_i + \Delta x \\ y_{i+1} &= y_i + \Delta y \\ \theta_{i+1} &= \theta_i \end{aligned}$$

Trajectory of mobile platform is plotted using left, right wheel encoder and IMU. When mobile platform travels straight, Δdl and Δdr are equal and heading angle of mobile platform is only angle from IMU, and if it is not travelling straight, then heading angle of mobile platform is addition of angle in Eq. (4) and IMU.

3 Experimental Method

A number of experiments are performed using mobile platform to estimate the position and understand the error due to variation in distance 'L' of the IMU sensors from the wheels. Test environment used for experiments is shown in Fig. 5.

3.1 Effect of dc Motors on Mobile Platform Heading Angle

Experiments were carried out by varying distance between IMU and midpoint of axis joining left and right dc motors. Initially, distance is set to 1 cm and mobile platform heading angle θ is measured. Further distance L is increased by 1 cm up to 17 cm and heading angle θ is measured in each case. Experiments were also repeated by setting mobile platform pointing towards the North, south, east and west directions to study the effect of direction. The effect of variation in current of the dc motor was also studied but it was observed that there is no effect of dc motor changing current on mobile platform heading angle θ.

Heading angle θ is measured for each 1 cm distance and this step is repeated for 10 runs. From the above graph in east and west directions, as the distance between IMU and midpoint of axis joining left and right dc motors becomes greater than 8 cm, effect of positioning of dc motor starts reducing on heading angle θ and goes down to zero at distance greater than 12 cm. In north and south directions, as distance between IMU and midpoint of axis joining left and right dc motors becomes greater than 2 cm, effect of positioning of dc motor is completely vanished. It has been also

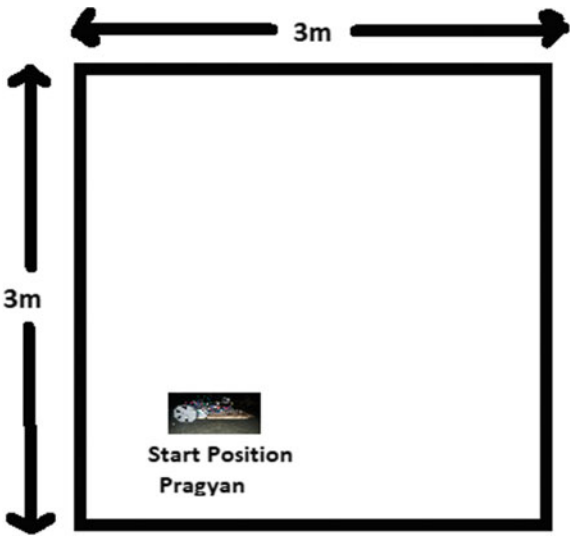


Fig. 5 Test environment

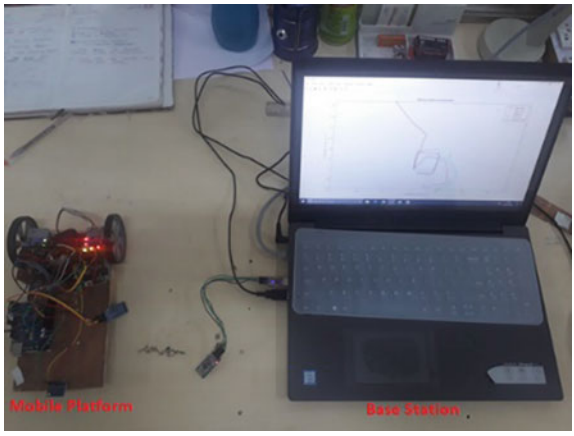


Fig. 6 Experimental set-up

found that, when current flowing through dc motor is changed, there is no effect on mobile platform heading angle θ. Same heading angle is obtained at different currents 0 mA, 45 mA, 50 mA and 55 mA, indicating driving current does not affect the IMU data.

3.2 Trajectory Estimation

The mobile platform Pragyan is programmed for following a square trajectory of 40 cm $\times$ 40 cm in the 3 m $\times$ 3 m environment. Left and right wheel encoder counts and robot heading angle θ are send to laptop using Bluetooth module HC-05. Experimental set-up is shown in Fig. 6. Throughout the travel, the position and heading angle data were recorded. MATLAB program is written to convert wheel encoder count and IMU angle into mobile platform position, and the position of mobile platform is plotted to obtain the trajectory as shown in Fig. 8. Experiment is performed to test the accuracy of mobile platform state and heading angle θ.

4 Result and Discussion

Effect of dc motor on IMU in the form of predicated heading angle is plotted for varying direction as well as distances. From the above graph in east and west directions, as distance between IMU and midpoint of axis joining left and right dc motors becomes greater than 8 cm, effect of positioning of dc motor on heading angle θ starts reducing and completely vanished at distance greater than 12 cm. In north and south directions, as distance between IMU and midpoint of axis joining left and right

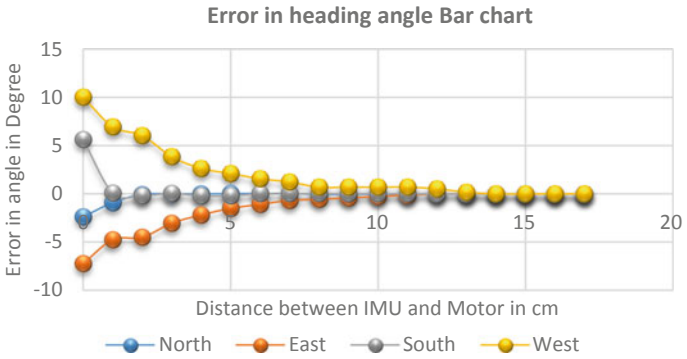


Fig. 7 Error in heading angle bar chart

dc motors becomes greater than 2 cm, effect of positioning of dc motor on heading angle is completely vanished.

The graph in Fig. 7 shows that effect of the dc motors on heading angle θ of mobile platform is affected by the distance between IMU and midpoint of axis joining left and right dc motors.

Experiments were also performed to test the effect of mobile platform state and heading angle θ on accuracy of the trajectory. For this purpose, the mobile platform was placed in the test environment and programmed to follow the square shape trajectory *A*, *B*, *C* and *D* with dimension 40 cm × 40 cm. The predicted trajectories are plotted for distance '*L*' varying as 0 cm, 5 cm, 10 cm and 15 cm are shown in Fig. 8.

The blue square indicates the ground truth, and in case of 0 cm distance, the predicted trajectory matches ground truth till *B* and then is totally erroneous. When the distance between IMU and midpoint of axis joining left and right dc motors is set to 15 cm on mobile platform then resultant trajectory is close to ground truth.

As the mobile platform travels through test environment, experimental coordinates are recorded. The end position experimental coordinates are compared with end position ground truth coordinates, as shown in Table 1 and graph is plotted root-mean-square error (RMSE)/cm versus distance between IMU and midpoint of axis joining left and right dc motors.

The root-mean-square error is calculated using ground truth coordinates (Xr, Yr) and experimental coordinates (Xe, Ye).

It can be seen that as the distance between IMU and midpoint of axis joining left and right dc motors increases, RMSE decreases, which is shown graphically in Fig. 9.

The method mentioned in [10] is complicated and the method used here is much simpler. When mobile platform travelled square-shaped trajectory, the total distance travelled by it is 160 cm and RMSE is 11.80 cm at '*L*' greater than 12 cm. Mobile platform travelled trajectory is compared with ground truth, and positioning error is 7.375%.

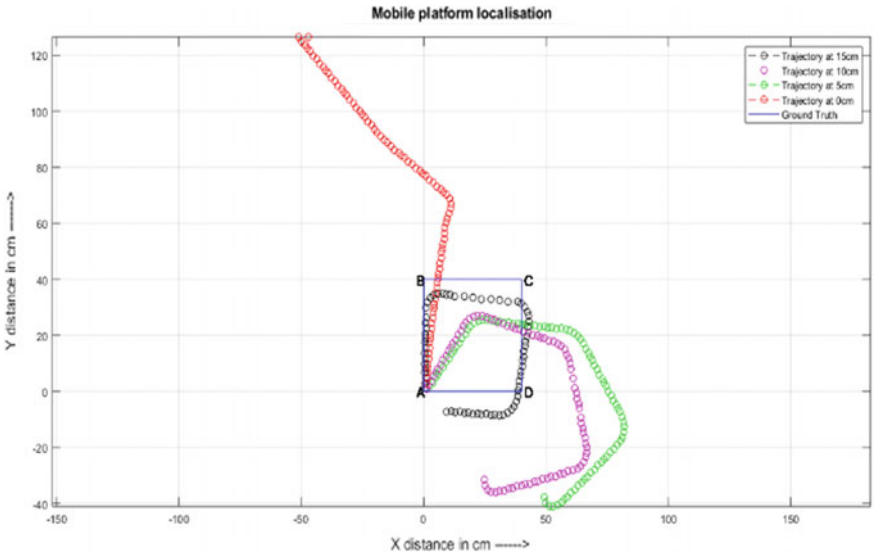
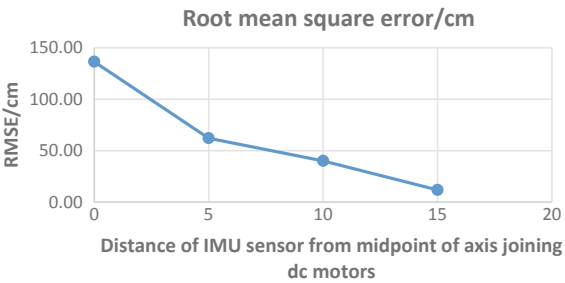


Fig. 8 Experimental mobile platform travel path

Table 1 Root-mean-square error analysis of mobile platform travel path

S. No.	Ground truth end position coordinates		Experimental end position coordinates		Distance L in cm	RMSE/cm
	Yr	Xr	Ye	Xe		
1	0	0	126.47	51.08	0	136.40
2	0	0	−37.7	49.30	5	62.06
3	0	0	−31.51	24.76	10	40.07
4	0	0	−7.27	9.30	15	11.80

Fig. 9 RMSE error analysis



5 Conclusion

This paper has outlined the experiments related to the accuracy of localization, using wheel encoder and IMU data. Estimation of mobile platform heading angle based on IMU data and travel distance measured via wheel encoder is described. Experiments were carried out by setting a mobile platform pointing towards the north, south, east and west directions to check the effect of position of IMU along the axis. The experiments indicate that the heading angle θ of mobile platform is affected by the distance between IMU and midpoint of axis joining left and right dc motors as well as the direction of movement. The error in position estimation is observed to be more in east and west direction, if IMU is placed at distance less than 12 cm. If IMU is placed at distance greater than 12 cm positioning error in all directions goes down to zero. It can be seen that as the distance between IMU and midpoint of axis joining left and right dc motors increases, RMSE decreases. It has been found that, when current flowing through dc motor is changed, there is no effect on the mobile platform heading angle. To understand the effect of error in heading angle on predicted trajectory, Pragyan was placed in the test environment and programmed to follow the square-shaped trajectory. The results clearly indicate the errors in trajectory prediction though the robot follows a square path.

Overall the error in heading angle could be attributed to dc motor. In future work, error generated due to placing of IMU near to the dc motors can be corrected mathematically. Further work on use extended Kalman filter (EKF) to remove error due to wheel slippages to obtain the more accurate trajectory of mobile platform is underway.

References

1. Baumgartner ET, Aghazarian H, Trebi Ollenu A (2001) Rover localization result for the FIDO rover. California Institute of Technology, Pasadena
2. Lee T, Shin J, Cho D (2009) Position estimation for mobile robot using in-plane 3-axis IMU and active beacon. In: IEEE international symposium on industrial electronics (ISlE 2009), Seoul Olympic Parktel, Seoul, Korea, July 5–8, 2009
3. Choi JM, Lee SJ, Won MC (2011) Self-learning navigation algorithm for vision-based mobile robots using machine learning algorithms. J Mech Sci Technol 25(1):247–254
4. Shen J, Tick D, Gans N (2010) Localization through fusion of discrete and continuous epipolar geometry with wheel and IMU odometry. Preprint submitted to 2011 American control conference. Received Sept 27, 2010
5. Baumgartner ET, Aghazarian H, Trebi-Ollennu A (2001) Rover localization results for the FIDO rover. In: Proceedings volume 4571, Sensor fusion and decentralized control in robotic systems IV, Intelligent systems and advanced manufacturing, Boston, MA, United States
6. Borenstein J, Feng L (1996) Measurement and correction of systematic odometry errors in mobile robots. IEEE Trans Rob Autom 12(6):869–880
7. Moon WS, Cho BS, Jang JW, Baek KR (2010) A multirobot positioning system using a multicode ultrasonic sensor network and a Kalman filter. Int J Control Autom Syst 8(6):1349–1355
8. Barshan B, Durrant-Whyte HF (1995) Inertial navigation systems for mobile robots. IEEE Trans Robot Autom 11(3):328–342

9. Brossard M, Bonnabel S (2019) Learning wheel odometry and IMU errors for localization. In: International conference on robotics and automation (ICRA), May 2019, Montreal, Canada
10. Cho B-S, Moon W, Seo W-J, Baek K-R (2011) A dead reckoning localization system for mobile robots using inertial sensors and wheel revolution encoding. J Mech Sci Technol 25(11):2907–2917
11. Yi J, Zhang J, Song D, Jayasuriya S. IMU-based localization and slip estimation for skid-steered mobile robots. https://www.researchgate.net/publication/221065982
12. Dellaert F, Fox D, Burgard W, Thrun S (1999) Monte Carlo localization for mobile robots. Carnegie Mellon University, Pittsburgh, Institute of Computer Science III, University of Bonn, Bonn
13. El-Diasty M (2014) An accurate heading solution using MEMS-based gyroscope and magnetometer integrated system (preliminary results). In: ISPRS annals of the photogrammetry, remote sensing and spatial information sciences, vol II-2, 2014 ISPRS Technical Commission II symposium, 6–8 Oct 2014, Toronto, Canada
14. http://www.cs.cmu.edu/~rasc/Download/AMRobots5.pdf

Steady Model for Classification of Handwritten Digit Recognition

Anujay Ghosh, Aruna Pavate, Vidit Gholam, Gauri Shenoy and Shefali Mahadik

Abstract Handwritten digit recognition plays an important role not only in computer vision but also in pattern recognition. Handwritten digit recognition is the competence of a machine to receive, calculate and decipher a human handwritten input from sources such as handwritten manuscripts, especially created before the advent of a digital revolution and digital images.This work implements the system to read the handwritten digits with a custom novel method identical to the amalgamation of different techniques, including principal component analysis, support vector machine and *K*-nearest neighbours to recognize and classify handwritten digits into their respective labels. PCA algorithm finds out the best linear combinations of the original features so that the variance along the new feature is maximum. Recognition of characters is done using KNN nonparametric machine learning algorithm, and SVM lowers the generalization error of the overall classifier. The proposed work does the analysis on digit data set having a total of 70,000 image samples. The performance of the system is analysed using different measurement metrics like precision, recall, f1 score and support, and the recognition of the patterns in the images shows the result with classification accuracy of 97%.

Keywords *K*-nearest neighbour · Support vector machine · Principal component analysis · Classification · Handwritten digit

1 Introduction

The human capability to examine and categorize objects and scenes is a very useful skill, researchers have tried to implement this through machine learning algorithm in many domains including Education, Sports, Transportation, Oil and Gas, Financial Services, Marketing and Sales, Government, health care and in many safety-critical applications like fingerprint recognition, facial recognition and many more [1]. Handwriting digit recognition is one of the major applications in machine learning applied

A. Ghosh (✉) · A. Pavate · V. Gholam · G. Shenoy · S. Mahadik
Atharva College of Engineering, Mumbai University, Mumbai 400095, India

R. Sharma et al. (eds.), *Innovation in Electrical Power Engineering, Communication, and Computing Technology*, Lecture Notes in Electrical Engineering 630,
https://doi.org/10.1007/978-981-15-2305-2_32

in many wide ranges of real-life applications such as signature identification and verification, zip code recognition in postal mail categorization, form processing, handwritten digit verification in bank, fraud detection etc. Handwritten digit recognition plays a crucial role in optical character recognition (OCR) and in pattern recognition [2]. There are many devices such as smart phones and tablets that can take handwriting as an input to a touch screen via a finger or using an electronic stylus. This allows user to quickly transfer the text to the devices which helps especially for the selective individuals who are not well versed with input devices such as keyboards to write text faster rather than typing slowly through input devices. Recognition of such text is very hard even by humans. Thus, a system that supports an automatic recognition of text would be very helpful in many applications.

1.1 Need of the System

Handwritten digit recognition system is developed to improve the accuracy of the existing solutions to achieve higher accuracy and reliable performance. Over the last decades, many machine learning algorithms made use of impressive handwritten digit recognition techniques such as baseline linear classifier, baseline nearest neighbour classifier, pairwise linear classifier, radial basis network, large fully connected Multilayer neural network, tangent distance classifier, optimal margin classifier [3], support vector machine (SVM) [4–8], CNN [5], fuzzy [9] neural network [7, 10–14], PCA [6, 15], CNN-SVM classifier [2, 16], KNN [17], recurrent neural network (RNN) [18] and DNN classifiers [19].

However, there are still some challenges that need to be solved. As handwritten characters are different in writing style, stroke thickness, deformation, rotation, etc., it is difficult to recognize [17, 18, 20, 21]. The main challenge in handwriting recognition system is to classify a handwritten digit based on black and white images. Furthermore, to meet the industry need, accuracy and robustness to the variation in writing style of the individual must be high.

1.2 Scope of the System

The digital world's advent began a mere century or two ago, but scriptures and books after books have been handwritten by human scholars from the beginning of mankind. Accepting the digital world first begins with the task of integrating the scripts that came into existence before the rise of computers and technology. Thus, this conversion and integration must begin with the most common values in the world that transcend different languages as well numbers.

The problem is to categorize handwritten digits into ten distinct classes with accuracy as high as possible. The digit ranges from zero (0) to nine (9). In this work, we utilized the support vector machines (SVMs), principal component analysis

(PCA) and *K*-nearest neighbour (KNN) techniques, by compounding to form a novel method to solve the problem. The experiment applied on digit data set [22, 23] is taken from the well-known Modified National Institute of Standards and Technology (MNIST) data set [23].

2 Related Work

For developing handwritten digit recognition, the literature presents a number of researches that have made use of machine learning techniques. Among them, a few techniques related to the work have been presented below.

Matan et al. developed a neural network architecture for recognizing handwritten digits in a real world. This network has 1% error rate with about 7% reject rate on handwritten zip code digits provided by the US portal service [24]. Jitendra Malik et al. developed simple and an easy approach for finding out the resemblance between shapes and utilized it for object recognition. The proposed approach was tested on COIL data set, silhouette, trademarks and handwritten digits [21].

S. M. Shamim et al. presented an approach to offline handwritten digit recognition. The main problem is the capability to develop a cost-effective algorithmic program that can acknowledge handwritten digits and which is submitted by users by the way of a scanner, tablet and other digital devices [14].

Caiyun Ma et al. proposed an approach based on specific feature extraction and deep neural network on MNIST database. The proposed work is compared with SOM() [6] and P-SVM [25], and the result shows the proposed algorithm with accuracy 94.2% with 24 dimensions and showed that the deep analysis is more beneficial than traditional in terms of visualization of features [19]. Anuj Dutt et al. compared the results of some of the most widely used machine learning algorithms like SVM, KNN and RFC 4 and with deep learning algorithms like multilayer CNN using Keras with Theano and TensorFlow. The result showed the accuracy of 98.70% using CNN (Keras + Theano) as compared to 97.91% using SVM, 96.67% using KNN, 96.89% using RFC and the lowest error rate 1.28% using convolution neural network [26]. Chayaporn Kaensar presented comparative analysis using three different algorithms like neural network, support vector machine and *K*-nearest neighbour. The analysis of the presented work demonstrates that the SVM is the best classifier with 96.93% accuracy with more time required for training as compared to neural network and *K*-nearest neighbour [7].

Mohd Razif Shamsuddin et al. presented handwritten digit recognition on MNIST data set. In this work, four different methods (logistic regression, random forest, extra trees classifier and convolution neural network) were applied on normalized MNIST data set and binary data set. The analysis result shows that the convolution neural network gives the system validation with the best result 99.4% on normalized data set and 92.4% on binary data set using extra trees algorithm. The analysis shows that the system works better on normalized data set [27]. Saeed AL-Mansoori proposed multilayer perceptron (MLP) neural network to solve the problem of the handwritten

digit recognition. The system performance is observed on MNIST data set by altering the number of hidden layers and the number of iterations, and the result showed the overall training accuracy of 99.32% and testing accuracy of 100% [8].

Cheng-Lin Liu et al. presented handwritten digit recognition on binary and grey images using eight different classifiers like KNN, MLP, PC, RBF, LVQ, DLQDF, SVC-poly and SVC-RBF tested on three different data sets CENPARMI, CEDAR and MNIST. The presented work is concluded as SVC-RBF gives the highest accuracy among all the algorithms, but this algorithm is extremely expensive in memory space and computation [28]. In addition to the above, other important works include research on local similarity [29], prototype generation techniques [30], handwriting verification [31], trajectory and velocity modelling [5] and feature extraction [15].

3 Materials and Methods

The work is implemented and tested in the following system requirements: Intel i 3 or later processor, minimum 2 GB RAM, minimum 2 GB graphics processing unit, operating system (Windows 7 and above), Anaconda Python 3.7. All the algorithms tried using scikit-learn Python library, version 0.17.1.

3.1 Data Set

The proposed system was implemented and tested using MNIST data set (Modified National Institute of Standards and Technology database). The MNIST data set contains handwritten digits having 60,000 examples in the training set and 10,000 examples in the test set. The MNIST data set was associated with MNIST data set which is the super-set of MNIST. The size of the image is 28×28 pixels $= 748$ pixels. There are close to 60,000 images in the combined data set that can be used for training and judging the system. The data set contains the input and likelihood that the image belongs to different classes (i.e. the machine-encoded digits, 0–9) [22, 23].

3.2 Methods

Figure 1 shows that proposed approach is an association of PCA, KNN and SVM algorithms to improve the classification accuracy. The PCA algorithm helps to reduce the number of attributes which contribute more towards classification. The first step is to load the data set and abstract the feature columns with target columns. The size of the data set is rather large (60,000 samples with 784 features); thus, extraction of features from the original large dimensional features of the data is done using PCA in the initial stage.

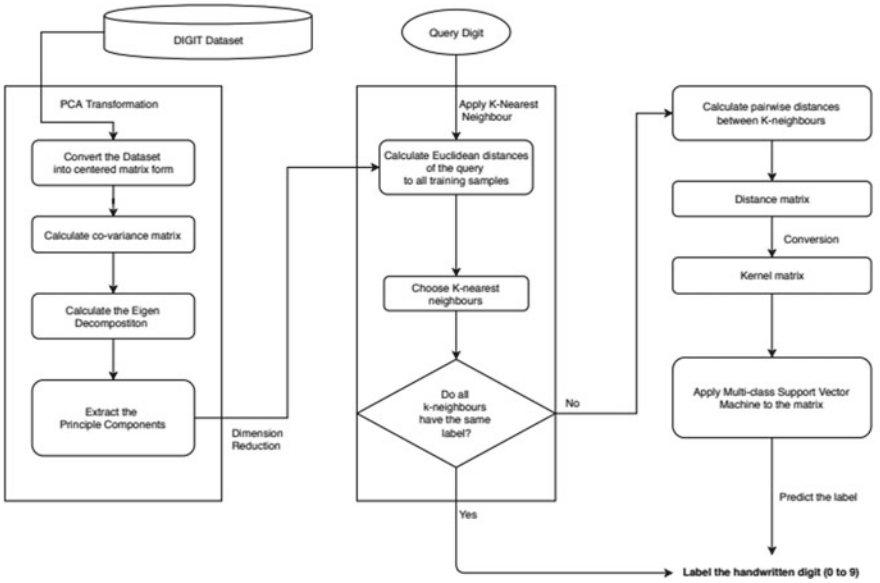


Fig. 1 Working of the proposed system for handwritten digit recognition

The first 60 features can explain approximately 97% of total substance (in terms of total variance retained), which fulfil to be typical of the information in the original data set as shown in Fig. 2. Thus, the first 60 principal components are implemented as the extracted features. The data is then split into training and testing sets. The simple implementation of SVM-KNN goes as follows: the KNN model is created and fit to the training set values, which trains the KNN classifier. For a query, it is necessary to compute the Euclidean distances of the query to all the training samples and pick the *K*-nearest neighbours. The general value of Euclidean distance (d) is calculated using Eq. 1.

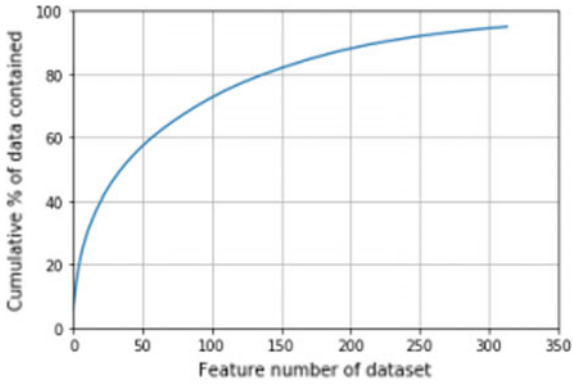


Fig. 2 Amount of data versus component number first 314 principal components as the extracted features using PCA