Journal of Computer Science 7 (5): 657-663, 2011
ISSN 1549-3636

A Novel Approach to Medical Image Segmentation

[1]Nandagopalan Shanmugam, [2]Adiga B Suryanarayana, [3]Sudarshan TSB
[4]Dhanalakshmi Chandrashekar and [4]Cholenally Nanjappa Manjunath
[1]Department of Computer Science and Engineering,
Amrita Vishwa Vidyapeetham, Amrita School of Engineering, Bangalore, India
[2]Tata Consultancy Services, Parallel Computing Division, Bangalore, India
[3]Department of Computer Science and Engineering,
Amrita Vishwa Vidyapeetham, Amrita School of Engineering, Bangalore, India
[4]Sri Jayadeva Institute of Cardiovascular Sciences and Research, Bangalore, India

Abstract: Problem statement: Segmentation is a vital aspect of medical imaging. It aids in the visualization of medical data and diagnostics of various diseases. Ultrasound image segmentation, in particular echocardiographic image segmentation, is required to identify the regions of interest such as Left Ventricle (LV) and other cardiac cavities. Existing methods do not address the drawbacks of speed and quality of segmentation. A faster method is required for effective, accurate and scalable clinical analysis and diagnosis. **Approach:** In this research, a novel approach is used to segment the 2D echo images of various views. A modified K-Means clustering algorithm, called "Fast SQL K-Means" is proposed using the power of SQL in DBMS environment. In K-Means, Euclidean distance computation is the most time consuming process. However, here it computed with a single database table and no joins. This method takes less than 10 sec to cluster an image size of 400×250 (100K pixels), whereas the running time of direct K-Means is around 900 sec. Since the entire processing is done with database, additional overhead of import and export of data is not required. The 2D echo images are acquired from the local Cardiology Hospital for conducting the experiments. **Results:** The proposed algorithm was tested by considering a number of echo images in apical four chamber, long-axis and short axis views. We have compared the direct K-Means implementation with the proposed algorithm by varying the data size from 10-100K and found that the results outperformed compared to the results of other authors. The pattern of the data and the number of clusters had almost no impact on the clustering time. **Conclusion:** An efficient and non-traditional model for echo image segmentation is presented by using the SQL. Fast algorithms are required for immediate analysis of echo images within ICUs, remote places, telemedicine. The challenge is that ultrasound images are prone to speckle noise, segmented echo images carry gaps in the cardiac regions which in turn causes difficulties in boundary tracing and selection of seed values for the K-Means. Future research can enhance the speed by partitioning the database tables and use of parallel SQL statements.

Key words: K-means, echocardiographic image, image segmentation, echo image, biomedical image segmentation, database tables, speckle noise

INTRODUCTION

Medical image segmentation subdivides an image into its constituent regions or objects. Automatic LV segmentation is a difficult task due to the relatively poor quality (speckle noise) of echocardiography images (Lim and Goh, 2009). Many researchers have proposed algorithms in the past for image segmentation tasks (active contour or snakes), but all of them consume extensive computation time and suitable for natural images. One of the main objectives of our work is to improve the computational efficiency of the segmentation process and at the same time enhance the quality of the output. In particular segmentation of medical images is of prime importance even in mammograms (Abdallah *et al*., 2008) to identify ROI. Similarly in echo images the ROI is Left Ventricle. The current work is automatic segmentation meaning no prior information is required. However, semiautomatic Seeded Region Growing (SRG) based image segmentation is also used in the literatures (Tamilselvi and Thangaraj, 2011).

Corresponding Author: Nandagopalan Shanmugam, Department of Computer Science and Engineering,
Amrita Vishwa Vidyapeetham, Amrita School of Engineering, Bangalore, India

Our focus is on designing a simple, elegant yet robust algorithm that segments a cardiac image for extracting its clinically relevant features (Muda *et al*., 2010). For this purpose, K-Means clustering algorithm is selected which partitions a data set into several groups such that the intra cluster points are similar and the inter-cluster points are dissimilar. K-Means is ideally suitable for biomedical image segmentation since the number of clusters (k) is usually known for images of particular regions of human anatomy. Though K-Means has been shown to be effective in producing good clustering results, one of its main drawbacks is the poor time complexity: O(Inkd), where I is the number of iterations, n is the number of data points, k is the number of clusters and d is the number of dimensions (Nandagopalan et al., 2010a). Integrating K-Means algorithm and SQL has many advantages. The image data can easily be stored in relational DBMS and we can perform all computations faster in SQL. Since the resolution of the image is normally large, handling such huge data sets without the help of DBMS is a daunting task. However, with proper SQL join statements (Ossama, 2010; Ordonez and Pitchaimalai, 2010) it is possible to make it faster. Semantic learning based dominant foreground region can be extracted for CBIR applications as discussed in (Rajam and Valli, 2011).

The authors contribution in this research is that the conventional K-Means algorithm is implemented in SQL and achieved an upper bound of O (n log n). Since SQL is already designed with efficient algorithms it is obvious that the proposed design must produce accurate output. We have kept the table joins to minimum to avoid unnecessary time delay. This algorithm can be used both for 2D echo and color Doppler images. To achieve further enhancement in speed, for all table updates a novel idea of TRUNCATE-INSERT combination is used (Nandagopalan et al., 2010b).

K-Means algorithm was successfully used for biomedical image segmentation using adaptive techniques and morphological operations. Muda *et al*. have applied K-Means algorithm to Intrusion Detection Systems (IDS). Three algorithms were proposed by Carlos Ordonez using DBMS SQL and C++ and demonstrated how K-Means can be of practical importance for clustering large data sets (Jaradat *et al*., 2009). Another approach to speed up the (Patel and Sinha, 2010) K-Means was based on k-d tree structure.

MATERIALS AND METHODS

Standard K-Means clustering algorithm: Generally the input to K-Means algorithm is the number of clusters (k) and is decided by the user depending upon the problem domain. This algorithm works like this: first it randomly selects k of the objects, each of which initially represents a cluster mean. For each of the remaining objects, an object is assigned to the cluster to which is the most similar. This is done based on the Euclidean distance between the object and the cluster mean. It then computes the new mean for each cluster and the process iterates until the criterion function converges. The quality of clustering is determined by the following error function:

$$E=\sum_{i=1}^{k}\sum_{p\in C_i}|p-m_i|^2 \quad (1)$$

Where:

E = The sum of the square error for all objects in the data set D

p = The object

m_i = The mean of cluster C_i

Given an initial set of K-Means $m_1^{(1)},\ldots,m_k^{(1)}$ which may be specified randomly. Assign each observation to the cluster with the closest mean by:

$$S_i(t) = \{x_j : \| x_j - m_i^{(t)} \le \| x_j - mi_*^{(t)} \| \text{ for } i^* = 1,\ldots, k\} \quad (2)$$

Calculate the new means to be the centroid of the observations in the cluster:

$$m_i^{(t+1)}=\frac{1}{|S_i^{(t)}|}\sum_{x_j\in S_i^{(t)}} x_j \quad (3)$$

It is obvious that the conventional K-Means algorithm (Nandagopalan *et al*., 2010a) for clustering n data objects to C_j clusters is not efficient.

The input for K-Means is a data set D containing n points with d dimensions, $D = \{i_1, i_2, i_3, .., i_n\}$. For our case, we shall assume that k = 3, because typically an echo image is segmented into three regions, i.e. cardiac cavity (black region), near endocardium (white region) and the rest (gray region). The data set is the pixel values of the given image f(x, y) of size M × N, where f(x, y) is the gray scale value of a pixel at location (x, y). No spatial details of these pixels are taken into account for clustering.

Table 1: Matrices / DB Tables

Matrix	Size	Description
Data	n × d	Pixel data
Centroid	k × d	Cluster mean
Eucl	n × k	Euclidean distance
CVCD	n × d	Cluster assignment

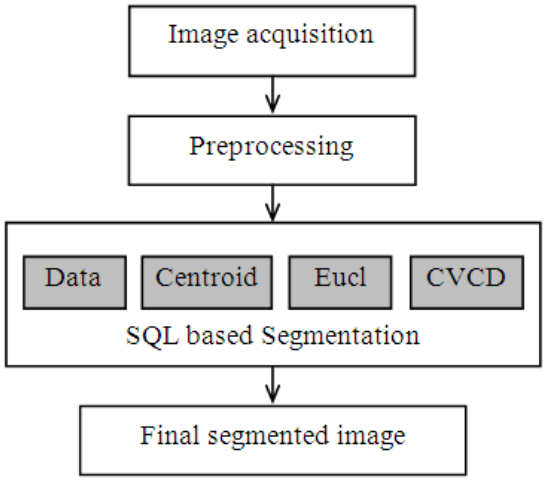

Fig. 1: Overview of Echo image segmentation

We use the matrices or tables as shown in Table 1 throughout the discussion of this study.

Each tuple in Data represents a pixel with its spatial co-ordinates and the gray scale intensity [0-255] value. This means that the number of dimensions is just one-intensity value of the pixel. Since k = 3, the Centroid table always contains 3 rows with the pixels being selected as centroids in each iteration. Next, in order to store the Euclidean distances, d1, d2 and d3, the table Eucl is used and each entry gives the distance of i^{th} pixel to the respective centroids in k clusters. Our algorithm uses Euclidean distance to find the nearest centroid to each pixel, i.e., the distance between C_j and D_i as shown in Eq. 4:

$$\text{Euclidean Distance} = \sqrt{\sum_{l=1}^{d} (D_{li} - C_{lj})^2} \quad (4)$$

The table CVCD stores the pixels and their assigned cluster number (j) during every iteration. At the end of the predefined iterations, the CVCD table would contain the desired segmented pixel data. We use the following subscripts in this study: i: 1..n: number of data points (pixels), j: 1..k: number of clusters and l: 1..d: number of dimensions.

Proposed algorithm: Fast SQL K-Means: The design follows almost the same conventional approach as given in (Patel and Sinha, 2006; Muda *et al*., 2010, Ordonez and Pitchaimalai, 2010), except that the procedure is implemented in SQL with its overview appearing in Fig. 1. The shaded boxes represent the name of the database tables being used for segmentation.

Image preprocessing: Generally, an echo image is acquired from an ultrasound and echocardiography system with a resolution of 800×564 (Philips machine is used for our experiment). Then, it is median filtered to remove the speckle noise (Lim and Goh, 2009). These images are of JPG format with 24bpp grayscale. This means the RGB values are same and taken as a one-dimension intensity value of each pixel, i.e., one byte. Now this filtered image is given as input to K-Means algorithm.

Relational DBMS tables: Following are the normalized database tables that are required for the clustering of pixel data:

Data (i, x, y, val), Centroid (j, x, y, val), Eucl(i, d1, d2, d3) and CVCD(i, j, x, y, val)

The Data table stores the pixel data in which i is the id and declared as primary key and also indexed. Next, to store the mean values of each cluster, a separate table Centroid is used. For this table j acts as the primary key which references i in Data table. The Euclidean distance is computed for each data point in Data with rows in Centroid and the computed value is stored in Eucl table for all clusters. Since k = 3, the Centroid table will have 3 tuples at any point of time. The table CVCD is to store the pixel and the assigned cluster number (1, or 2, or 3).

Algorithm: In order to speed up the processing, minimum number of tables and joins must be formulated in the SQL statements. Figure 2 shows the proposed algorithm for Fast SQL based K-Means clustering.

Steps 1-4 are initialization steps and do not depend on n. The for loop in line 5 iterates Q times and executes 3 INSERT statements. The first sub-step deletes the Centroid table data (using TRUNCATE) and inserts the newly computed mean of each cluster. Next step computes the distance between each pixel in Data table with the cluster centroid and insert them into Eucl table. Finally, the third step computes the minimum out of d1, d2, d3 and assigns the cluster id of each pixel and inserts them into CVCD. The final table SI is to quantize the pixels into 3 colors: 0, 150 and 255 for the final image.

Algorithm FKM_SQL(I, Q)
// Input: Echo image, I and # of iterations: Q
// Output: Segmented Image, S
1. DeleteTableData() // TRUNCATE table
2. Load Image pixel values to Data table
3. Centroid ← Rand(Data table)
4. Initialize Eucl and CVCD tables.
5. for i ← 1 to Q do
   - INSERT INTO Centroid ← Average($CVCD_{Group\ By\ j}$) // average of each cluster, j = 1..k
   - INSERT INTO Eucl ← Euclidean distance ($Data_i$ * $Centroid_j$), for i = 1..n and j = 1..k
   - INSERT INTO CVCD ← Min($Eucl_i$(d1, d2, d3))
6. For each cluster data in CVCD, assign a special value [0, 150, 255] and insert into SI.
7. Create S from SI
8. Return S
9. **end** FKM-SQL.

Fig. 2: Algorithm for fast SQL K-means

Since all the tables are indexed, the time taken by the three insert statements in the for loop of step5 can be shown to be O(n log n). Further, it does not have any impact on k.

Delete table data: Before populating the database tables, we must delete all the existing rows. The fastest way to do this in Oracle 10g is by executing the following statements:

```
TRUNCATE TABLE CVCD;
TRUNCATE TABLE EUCL;
TRUNCATE TABLE CENTROID;
TRUNCATE TABLE DATA;
```

Note that TRUNCATE is faster than DELETE, because the former does not store the deleted tuples in rollback segments. Next, loading the image data into Data table can be done as follows:

```
INSERT INTO Data (i, x, y, val) VALUES (:i, :x, :y, :val)
```

where, *i*, :x, :y, :val are the pixel id, pixel coordinates and the intensity vale arrays. This method of parameterized insertion is more efficient. In order to initialize the Centroid table, we simply use random number generator and pick 3 pixels from Data table. However, this must be done using the dynamic SQL:

```
INSERT INTO Centroid
    (SELECT 1, x, y, val FROM Data WHERE i = "j1");
INSERT INTO Centroid
    (SELECT 2, x, y, val FROM Data WHERE i = "j2");
INSERT INTO Centroid
    (SELECT 3, x, y, val FROM Data WHERE i = "j3");
```

where, j1, j2 and j3 are the three random numbers generated using the host language (C#.NET). The other tables are initialized in a similar way and are shown below:

```
INSERT INTO CVCD (i, j, val)
 ( SELECT v1.i,
    Case          when d1 <= d2 and d1 <= d3 then 1
         when d2 <= d3 and d2 <= d1 then 2
         when d3 <= d2 and d3 <= d1 then 3
    end as j, v1.val
  FROM Eucl v2, Data v1
  WHERE v2.i = v1.i);
```

It is easy to notice that the only input to this algorithm is Q-number of iterations. Next, *k* is set to its default value as 3. It is experimentally verified that a maximum of 6-8 iterations is sufficient to get good segmentation results. Euclidean distance, d is calculated with a single SQL statement for all pixels w. r. t the centroids without join (Nandagopalan *et al*., 2010b).

Update of database tables: We must update all the tables, except the Data table to cluster the data points. Depending upon the pixel data distribution and initial seed values, certain clusters may contain NULL values. This would cause incorrect join operations. To overcome this problem, left-outer join and NVL are appearing in the queries. Hence, our queries are carefully designed for any eventualities and this exhibits the robustness of the proposed design. The sub-steps of statement 5in Fig. 2 can be accomplished using the following SQL statements.

Update Centroid table: Before performing the insert operation, it is mandatory to delete the rows with TRUNCATE statement:

```
UPDATE Centroid c3
SET  j, val ) =
    (SELECT c1.j, c2.val FROM Centroid c1,
    (SELECT j, Avg(val) as val
        FROM CVCD GROUP BY j ) c2
     WHERE c1.j = c2.j(+) AND c1.j = c3.j );
```

Since the number of rows of this table is always 3, there is no need to use INSERT statement and instead it can be updated directly.

Update Eucl table: For this table, the TRUNCATE must be used prior to insert, because the size of this table is equal to the resolution of the image (normally huge):

```
INSERT INTO Eucl (i, d1, d2, d3)
    (SELECT i, sqrt(power((e2.val - c1.val), 2)) as d1,
          sqrt(power((e2.val - c2.val), 2)) as d2,
     sqrt(power((e2.val - c3.val), 2)) as d3
     FROM Data e2,
         (SELECT j, x, y, NVL(val, 0) as val
           FROM Centroid WHERE j = 1) c1,
         (SELECT j, x, y, NVL(val, 0) as val
           FROM Centroid WHERE j = 2) c2,
         (SELECT j, x, y, NVL(val, 0) as val
           FROM Centroid WHERE j = 3) c3
 );
```

The advantage of SQL is evident here; the distance calculation for all clusters and all data points are computed with a single query; thus obtaining fast running time.

Update CVCD table: Finally, the cluster assignment to each pixel is done by finding the least distance in Eucl table and update CVCD table accordingly:

```
INSERT INTO CVCD (i, j, val)
 (SELECT v1.i,
    Case          when d1 <= d2 and d1 <= d3 then 1
         when d2 <= d3 and d2 <= d1 then 2
          when d3 <= d2 and d3 <= d1 then 3
    end as j, v1.val
  FROM Eucl v2, Data v1
  WHERE v2.i = v1.i);
```

With a Case statement, this study can easily be done as shown above.

Next, in order to create an image for display we assign 0 (black) to all pixels in cluster 1, assign 150 (gray) to all pixels in cluster 2 and 255 (white) to all pixels in cluster 3. This task is also carried out by the following SQL statement:

```
INSERT INTO SI (i, j, val)
    ( SELECT i, j, DECODE
         (j,        1, 0,
                    2, 150,
                    3, 255) val
      FROM CVCD
    );
```

Now, with the data in CVCD table the image can be constructed and returned to the calling routine. The sample segmentation of images is shown in the results section. Since the number of dimensions is just 1 and the data size is small, K-Means algorithm converges very fast.

Experimental setup: To evaluate the performance of the proposed algorithm, a number of images are used. For the developmental work, following tools are helpful: Visual Studio 2010, ODAC (Oracle Data Access Components (ODAC) on C#.NET Framework, Oracle 10g DBMS running on Dell T3400 Desktop with Intel Core2 Duo Processor @ 2.33GHz with 2GB RAM and 150GB HDD.

RESULTS

Following approach is used to demonstrate the improved efficiency of the proposed algorithm:

- 2D echo images of different views but of same size (i.e., n remains constant)
- A single image at different resolutions (10K pixels to 100 K pixels - varying n)
- A data size of 100 K pixels with k = 5 (varying d)

These real test data are executed on two algorithms: Conventional and Fast SQL K-Means. The running times are also compared with the earlier results. Figure 3 shows five 2D echo images of different views with resolution 400×250 (n = 100 K) and their segmented outputs.

The objective of the next experiment is to compute the execution time for 10 echo images by running conventional and our algorithm. As per Table 2, it is evident that the conventional K-Means is significantly slow compared to the SQL implementation.

These timings are calculated only for the statements inside the for loop (with Q = 4) and does not include the set up time or other times required for the clustering process.

Another surprising inference obtained from this analysis is that, when k = 5, there is no significant change in the running time for the same data size. This clearly indicates that the SQL statements are independent of the number of clusters. The table data is plotted as a bar chart and shown in Fig. 4.

Table 2: Comparing Conventional K-Means with Fast SQL K-Means (Q = 4)

Image	Running time (sec) k = 3 Direct K-means	Running time (sec) k = 3 Fast SQL K-means	Running time (sec) k = 5 Fast SQL K-means
Img1	951.3	12.2	12.3
Img2	902.5	10.3	12.4
Img3	920.4	8.8	12.5
Img4	962.2	11.1	15.0
Img5	945.6	12.0	14.7
Img6	916.9	9.2	15.9
Img7	930.2	11.1	17.1
Img8	917.6	12.0	16.8
Img9	919.2	11.9	16.4
Img10	953.0	9.8	14.4

Table 3: Comparing Direct K-Means with Fast K-Means for various values of n (Q = 4)

n (pixels)	Running time (sec) Conventional K-Means	Running time (sec) Fast SQL K-Means
10000	31.7	0.9
20000	125.1	1.8
30000	214.5	2.7
40000	352.4	3.1
50000	461.8	4.4
60000	552.1	5.1
70000	663.2	6.3
80000	759.5	7.9
90000	842.7	10.6
100000	951.3	10.9

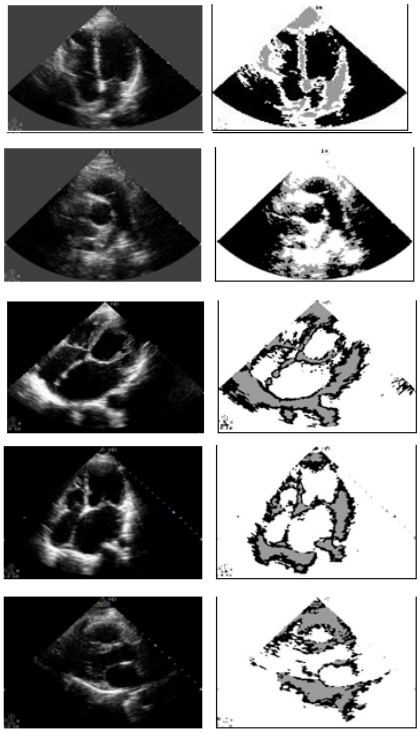

Fig. 3: Input images and their segmented images

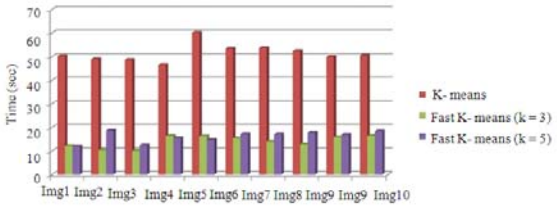

Fig. 4: Comparing Conventional K-Means with Fast SQL K-Means implementation for k = 3 and k = 5 with Q = 4.

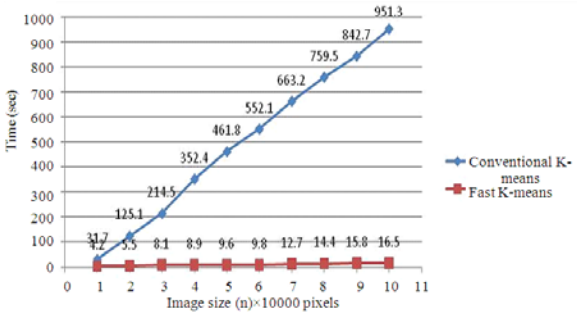

Fig. 5: Plot of running time (Direct K-Means Vs Fast SQL K-Means)

To understand the relative efficiency of this algorithm under more practical circumstances, two algorithms were executed by varying the data size from 10 K pixels to 100 K pixels. For this experiment second image of Fig. 3 was selected as input under different resolutions. The results appear in Table 3 and Fig. 5.

DISCUSSION

The proposed Fast SQL K-Means is faster than conventional and SQL K-Means by a factor of 90 and 10 respectively. No significant change in running time when k increases. We can compare the results of these experiments with that of other authors and is given below:

- Proposed Fast SQL K-Means: n = 100K, k = 3, Q = 4
- T = 8.8 s (best value)
- Carlos Ordonez's (Ordonez and Pitchaimalai, 2010) method 4 CPUs, 40 AMPs: n = 100K, k = 4,
- d = 8, Q = 4, OptKM = 17.0 s, IncrKM = 44 s
- Java based (Velmurugan and Santhanam, 2010): n = 1000,
- T = 7.6 s

- Dataset: 'Israel', k = 8, d = 4, T = 140.4 s

The above arguments reveal that our algorithm is fast to a large extent than others and also the quality of clustering is good for visual interpretation. It is also useful for ventricle border tracing and to extract clinically relevant features.

CONCLUSION

The main task of this research is the echo image segmentation and for this an efficient implementation of K-Means clustering algorithm, called Fast SQL K-Means is implemented. Traditional K-Means algorithm presents scalability problems with increasing number of clusters or number of points. Its performance graphs exhibit nonlinear behavior. The SQL based algorithm does not require extensive set up and also take extra time in the segmentation process, because the patient data is already available in the database.

The algorithm has been implemented on C#.NET framework. We have demonstrated the practical efficiency of this algorithm both theoretically, through a data sensitive analysis and empirically, through experiments on both synthetically generated and real data sets like live patient.

REFERENCES

Muda, Z., W. Yassin, M.N. Sulaiman and N.I. Udzir, 2010. A K-means and naive bayes learning approach for better intrusion detection. Inform. Technol. J., 10: 648-655. http://docsdrive.com/pdfs/ansinet/itj/2011/648-655.pdf

Jaradat, A., R. Salleh and A. Abid, 2009. Imitating K-means to enhance data selection. J. Applied Sci., 9: 3569-3574.

Nandagopalan, S., B.S. Adiga, C. Dhanalakshmi and N. Deepak, 2010a. A fast k-means algorithm for the segmentation of echocardiographic images using DBMS-SQL. Proceedings of the 2nd International Conference on Computer and Automation Engineering, Feb. 26-28, IEEE Xplore, Singapore, pp: 162-166. DOI: 10.1109/ICCAE.2010.5451438

Nandagopalan, S., B.S. Adiga, C. Dhanalakshmi and N. Deepak, 2010b. Color doppler echocardiographic image analysis via shape and texture features. Proceedings of the International Conference on Bioinformatics and Biomedical Technology, Apr. 16-18, IEEE Xplore, Chengdu, pp: 139-143. DOI: 10.1109/ICBBT.2010.5478992

Lim, C.T. and J.C.H. Goh, 2009. Speckle reduction of echocardiograms via wavelet shrinkage of ultrasonic RF signals. Proceedings of the 13th International Conference on Biomedical Engineering, Dec. 3-6, Springer, Singapore, pp: 395-398. ISBN: 3540928405

Ossama, K.M., 2010. Increasing database performance through optimizing structure query language join statement. J. Comput. Sci., 6: 585-590. DOI: 10.3844/jcssp.2010.585.590

Patel, B.C. and D.G.R. Sinha, 2010. An adaptive k-means clustering algorithm for breast image segmentation. Int. J. Comput. Appli., 10: 35-38. DOI: 10.5120/1467-1982

Velmurugan, T. and T. Santhanam, 2010. Computational complexity between k-means and k-medoids clustering algorithms for normal and uniform distributions of data points. J. Comput. Sci. 6: 363-368. DOI: 10.3844/jcssp.2010.363.368

Abdallah, M.H., A.A. AbuBaker, R.S. Qahwaji and M.H. Saleh, 2008. Efficient technique to detect the region of interests in mammogram images. J. Comput. Sci., 4: 652-662. DOI: 10.3844/jcssp.2008.652.662

Rajam, I.F. and S. Valli, 2011. SRBIR: Semantic region based image retrieval by extracting the dominant region and semantic learning. J. Comput. Sci., 7: 400-408. DOI: 10.3844/jcssp.2011.400.408

Tamilselvi, P.R. and P. Thangaraj, 2011. Computer aided diagnosis system for stone detection and early detection of kidney stones. J. Comput. Sci., 7: 250-254. DOI: 10.3844/jcssp.2011.250.254

Ordonez, C. and S.K. Pitchaimalai, 2010. Bayesian classifiers programmed. IEEE Trans. SQL. Knowl. Data Eng., 22: 139-144. DOI: 10.1109/TKDE.2009.127