Algoritmi paraleli de procesare a imaginilor

Carmen PUGHINEANU

***Abstract*—In this article we aim to use the power calculation offered by a cluster to process the images. We achieved and proved the use of the paralel algorithms to process the images. We studied and paralelised two clustering algorithm, namely the k-means algorithm and ISODATA. The experimental results compare the two algorithms from the point of view of the execution time, obtained by the algorithm implementation using the MPI standard.**

***Index Terms*—grayscale, cluster, segmentation, parallelization, threshold value**

I. Introducere

În prima parte a articolului imaginile color vor fi convertite în imagini grayscale, apoi se vor paraleliza [1] cei doi algoritmi, k-means şi ISODATA. În partea a doua are loc compararea timpilor de execuţie şi concluziile obţinute în urma segmentării imaginilor.

Segmentarea este procesul de clasificare [2] a pixelilor dintr-o imagine. Este unul dintre procedeele cele mai complexe în procesele de vizualizare.

Prin segmentarea imaginilor digitale obţinem partiţionarea acestora în mai multe segmente (seturi de pixeli) necesare pentru a localiza obiecte, a identifica zone ce sunt uniforme în raport cu un anumit criteriu.

Procesul de segmentare a unei imagini constă în partiţionarea completă a imaginii într-un ansamblu de mulţimi disjuncte, nevide şi conexe, ce satisfac un anumit criteriu [3].

Algoritmul k-means [4], model propus de MacQueen (1967) este considerat cel mai simplu algoritm de segmentare a imaginilor.

Algoritmul ISODATA propus este în principiu asemănător cu algoritmul k-means dar acesta permite ca numărul de clustere care urmează a se determina, să se modifice automat în timpul iteraţiei prin fuziunea grupurilor similare şi divizarea celor cu deviaţii standard mari [5].

Aceşti algoritmi lucrează cu un set mare de date şi necesită o putere de calcul destul de mare. Din acest motiv implementarea secvenţială a acestor algoritmi nu este foarte eficientă.

În acest articol se urmăreşte implementarea algoritmului k-means pe un cluster format din 9 calculatoare (master + 8 noduri). Sistemul de operare folosit pentru acest cluster este Fedora 12(GNU/Linux). Pentru realizarea CLUSTER-ului s-a utiliza două switch-uri, unul de 100MB(MPI) şi unul de 1GB(folosit de NFS).

Carmen PUGHINEANU, Profesor CNPR Piatra Neamţ

II. Transformarea imaginilor din RGB în imagini grayscale

Pentru a putea aplica algoritmii este nevoie de o fază de preprocesare şi anume transformarea imaginii color în imagini cu niveluri de gri.

Fişierele alese pentru crearea setului de date au format .bmp şi chiar dacă au o mărime mult mai mare decât altele, ele nu au pierderi de date, iar reprezentarea datelor este deosebit de simplă.

La o imagine pe 24 biţi, sunt necesari 3 octeţi pentru definirea unui pixel. Fiecare din cei trei octeţi asociaţi unui pixel codifică cantitatea de roşu, verde şi albastru (imaginea RGB - Red-Green-Blue) cu valori cuprinse între 0 şi 255.

III. Descrierea setului de date de intrare pentru algoritmul k-means si algoritmul ISODATA

Setul de date de intrare are o formă proprie în funcţie de structura fiecărui punct al imaginii. Fiecărui punct i se asociază un vector de trăsături caracteristice. Aceste trăsături sunt nivelul de gri (grayscale) al pixelului şi coordonatele sale în imagine.

Mulţimea vectorilor de caracteristici astfel obţinuţi (numărul lor este evident egal cu numărul de puncte din imagine), formează setul de date de intrare şi este partiţionată cu ajutorul algoritmului descris în acest articol.

Imaginea este astfel reprezentată printr-o matrice în care fiecare punct corespunde unui pixel. Pentru a realiza procesul de grupare a pixelilor este nevoie de o funcţie de similitudine.

Fiecărui cluster (fiecărei clase) i se atribuie o anumită etichetă şi toate punctele imaginii al căror vector de trăsături aparţine aceluiaşi cluster vor primi eticheta clusterului respectiv.

IV. Segmentarea imaginilor

Segmentarea imaginilor [6] este una din cele mai importante componente ale procesului de analiză şi interpretare a imaginilor. Erorile care apar în procesul de segmentare vor afecta toate componentele următoare.

Segmentarea unei imagini digitale pe nivele de gri în funcţie de nivelul de gri şi de coordonatele pixelilor din imagine presupune o partiţie a imaginii în mulţimi conexe, astfel încât fiecare mulţime are toţi pixelii din imagine cu nivel de gri apropiat.

Pixelii din imagine sunt asociaţi uneia sau alteia dintre mulţimi în funcţie de proprietatea lor de similitudine. În urma alocării tuturor pixelilor din imagine la o mulţime sau alta,

rezultă segmentarea imaginii în numărul de mulțimi, obținându-se o partiție a imaginii.

V. MASURI ALE SIMILARITATII DINTRE DOUA PUNCTE

Spre deosebire de clasificările clasice, care au un scop bine stabilit şi criterii clare de grupare, clasificările automate au drept unic criteriu similaritatea dintre pixelii ce urmează a fi grupați. Există o varietate de metode ce pot fi utilizate pentru măsurarea similitudinii.

În data clustering cea mai folosită metodă de măsurare a similarității o reprezintă distanța Euclidiană dată de următoarea formulă:

$$d(x, y) = \|x - y\| = \sqrt{\sum_{i=1}^{3} (x_i - y_i)^2} \qquad (1)$$

unde: $x=(x_1,x_2,x_3)$ şi $y=(y_1,y_2,y_3)$ sunt două puncte date în coordonatele carteziene.

Practic, o clasificare automată realizează o grupare optimă de puncte sub raportul similarității dintre ei.

VI. ALGORITMUL K-MEANS

Algoritmul k-means (MacQueen, 1967) este unul din cei mai populari algoritmii de partiționare şi urmăreşte identificarea a *k* grupe (clustere) distincte, astfel încât datele din fiecare clasă să fie suficient de similare [7].

Fiecare clasă va avea un reprezentant şi va fi considerat centrul clasei. Aceste centre se definesc aleatoriu şi pe la o distanță cât mai mare una de alta, deoarece alegerea acestor puncte poate influența foarte mult rezultatul dat de algoritm. Următorul pas o reprezintă determinarea apartenenței fiecărui punct la un cluster. Zonele din imagine cu aceeaşi similaritate pot să nu fie conexe, deoarece punctele imaginii depind de trăsăturile utilizate, şi deci orice segmentare prin clustering va fi urmată de etichetarea imaginii segmentate, pentru identificarea obiectelor individuale. Când punctele nu îşi schimbă clusterul din care fac parte atunci gruparea este terminată. În caz contrar trebuie să se recalculeze centrele pentru fiecare cluster în funcție de punctele care fac parte din el. După calcularea acestor centre se realizează din nou determinarea apartenenței fiecărui punct la un cluster. Astfel este generată o buclă. La fiecare iterație din buclă, clusterele îşi schimbă centrul. Ieşirea din buclă se poate realiza în momentul în care clusterele nu îşi mai schimbă centrele. Setul de date de intrare ce urmează a fi prelucrat este mare. Dacă setul de date ce conține *n* puncte ($x_1,x_2,\ldots x_n$), în care fiecare punct este considerat ca un vector de observații de dimensiune *d (d=3)*, atunci clusterizarea k-means presupune împărțirea acestui set în *k* partiții (clase) disjuncte şi nenule ($k<n$) $S=\{S_1,S_2,\ldots S_k\}$ astfel încât să se minimizeze (suma pătratelor distanțelor din interiorul clusterului) funcția de similaritate:

$$E = \arg\min_{s} \sum_{i=1}^{k} \sum_{x_j \in S_i} \left\| x_j - c_i \right\|^2 \qquad (2)$$

unde: c_i este media lui S_i.

Algoritmul secvențial de grupare a *n* puncte în *k* clase este:

```
Aleator se stabilesc k puncte (pixeli) ca
centrele inițiale a claselor
repetă
  Pentru fiecare punct se caută cea mai
apropriată clasă
  Dacă punctele şi-au schimbat clasa din
care face parte
    Recalculează centrele claselor în
    funcție de punctele care fac parte din
    fiecare clasă
Până când punctele care şi-au schimbat
clasa este mai mic decât o valoare
prestabilită
```

Algoritmul serial k-means are timpul de execuție egal cu $O(R_s kn)$ unde *k* este numărul dorit de clustere (grupe sau clase) şi R_s este numărul de iterații.

VII. ALGORITMUL ISODATA

Un rol important în procesul de segmentare îl constituie alegerea inițială a numărului de clase. O alegere greşită determină erorile de supra-segmentare sau sub-segmentare. Supra-segmentarea se poate corecta prin reunirea ulterioara a obiectelor deja determinate si este deci preferabila alegerea unui *k* mai mare decât necesar.

Inițial se doreşte segmentarea imaginii în k clustere, dar algoritmul permite ca numărul de clustere să se modifice automat în timpul iterației prin fuziunea grupurilor similare şi divizarea celor cu deviații standard mari.

Parametrii utilizați în acest algoritm sunt [8]:

1 - K = numărul de clustere dorit;
2 - I = numărul maxim de iterații permise;
3 - P = numărul maxim de perechi de cluster care se pot uni (fuziona);
4 - Θ_N = o valoare de prag pe care o poate avea un număr minim de probe din fiecare cluster (utilizată pentru divizare);
5 - Θ_S = o valoare de prag pentru deviația standard (utilizată pentru operația de separare);
6 - Θ_C = o valoare de prag pentru distanțe perechi (folosite pentru operația de îmbinare).

Paşii algoritmului

1. alegerea arbitrară a celor k centre, M_1, M_2, ..., M_k
2. fiecare probă N se atribuie celui mai apropiat cluster:

$X \sim w_j$ dacă

$$D_L(X, M_j) = \max\{D_L(X, M_i), i = 1,2,...,k\}$$

3. se îndepărtează clusterele cu mai puțin de Θ_N membri
4. se recalculează fiecare centru:

$$M_j = \frac{1}{N_j} \sum_{X \sim w_j} X, \; j = 1,...,k$$

5. se calculează distanţa medie D_j între probele din clusterul w_j şi centrul corespunzător:

$$D_j = \frac{1}{N_j} \sum_{X \sim w_j} D_L(X, M_j), j = 1,...,k$$

6. Se calculează distanţa medie generală a probelor din centrele clusterelor lor respective

$$M_j = \frac{1}{N} \sum_{j=1}^{k} N_j D_j$$

7. Dacă sunt prea puţine clustere se merge la pasul 8;
Altfel
dacă sunt prea multe clustere se merge la pasul 11;
altfel se merge la pasul 14.

Paşii 8 – 10 sunt pentru operaţia de separare, Paşii 11–13 sunt pentru operaţia de fuziune

8. Primul pas pentru separare.
Se găseşte vectorul deviaţiilor standard $\sum | j = \left[\sigma_1^{(j)},...,\sigma_n^{(j)}\right]$ pentru fiecare cluster

$$\sigma_i^{(j)} = \sqrt{\frac{1}{N_j} \sum_{X \sim w_j} (x_i - m_i^{(j)})^2}, i = 1,...,n, j = 1,...,k$$

9. se află componenta maximă notată cu $\sigma_{\max}^{(j)}$ pentru orice $j=1,\ldots,k$

10. Dacă pentru fiecare $\sigma_{\max}^{(j)}$ ($j=1,\ldots,k$) tot ce urmează este adevărat:

- $\sigma_{\max}^{(j)} > \theta_s$
- $D_j > D$,
- $N_j > 2\theta_N$

atunci M_j este divizat în două noi centre de cluster M_j^+ şi M_j^- prin adăugarea lui $\pm\delta$ la componenta lui M_j corespunzând lui $\sigma_{\max}^{(j)}$ unde δ poate fi $\alpha\sigma_{\max}^{(j)}$ pentru vreun $\alpha > 0$. Apoi se şterge M_j şi $k \leftarrow k+1$ şi apoi se merge la pasul 2
altfel se merge la pasul 14.

11. Primul pas spre fuziune. Se calculează distanţele, în perechi, $D_{i\ j}$ între fiecare două clustere: $D_{ij} = D_L(M_i, M_j)$, (pentru toţi $i \neq j$) şi se aranjează aceste k(k-1)/2 distanţe în ordine ascendentă.

12. Se găsesc cei mai mici D_{ij} nu mai mulţi ca P, care sunt de asemenea mai mici decât θ_C şi se menţin în ordine ascendentă: $D_{i1\ j1} \leq D_{i2\ j2} \leq \ldots \leq D_{ip\ jp}$.

13. Se execută fuzionare în perechi: pentru $l = 1,\ldots,P$, se execută următoarele:
Dacă nici unul din M_{i1}, M_{j1} nu au fost folosite în această iteraţie, atunci se fuzionează pentru a forma un nou centru:

$$M = \frac{1}{N_{il} + N_{jl}} \left[N_{il} M_{il} + N_{jl} M_{jl}\right]$$

se şterge M_{i1} şi M_{j1}, şi $k \leftarrow k-1$. Megeţi la pasul 2.

14. Algoritmul se termină dacă numărul maxim de iteraţii I este atins. Altfel se merge la Pasul 2.

VIII. Implementarea paralelă a algoritmilor K-means şi ISOADATA

Procedura de clasificare în cazul paralel se derulează astfel: se pleacă de la un aranjament arbitrar al punctelor într-un număr prestabilit de grupe, după care punctele sunt mutate dintr-o grupă în alta în vederea minimizării varianţei din interiorul grupelor şi deci maximizării varianţei dintre grupe. Numărul de mutări poate fi, de asemenea, specificat [9].

Ideea de bază este a definii *k* centri de greutate, unul pentru fiecare cluster, atât cât este posibil mai îndepărtate unele de altele. Următorul pas este să luăm fiecare punct de-a lungul setului datelor de intrare şi să-l asociem celui mai apropiat centru de greutate. Când nu există puncte nedecise, primul pas este complet, iar o grupare iniţială s-a făcut. În acest moment trebuie să recalculăm *k*, noi centre de greutate ale clusterilor rezultaţi din pasul anterior.

În funcţie de anumite valori de prag, în algoritmul ISODATA există posibilitatea ca centrul unui cluster să fie divizat în alte două noi centre, sau ca aceste centre să fuzioneze pentru a forma un nou centru, astfel numărul de clustere creşte sau scade cu 1.

În continuare se vor partiţiona calculele care vor fi efectuate prin asocierea fiecărei operaţii cu datele cu care operează. Această partiţionare produce un număr de task-uri, fiecare task conţinând date şi un set de operaţii pentru ele. Pot exista cazuri în care o operaţie trebuie efectuată asupra datelor din task-uri diferite. În aceste cazuri este necesară realizarea comunicaţiei între aceste task-uri.

Pentru cazul de faţă se va folosi strategia realizării aceluiaşi set de calcule pe seturi diferite de date. Astfel datele sunt împărţite în părţi egale, numărul acestora fiind egal cu numărul de procese care se vor executa în paralel.

Implementarea acestor algoritmi se realizează cu ajutorul standardului MPI.

IX. Rezultate experimentale

Imaginea în formatul RGB din figura 1, este transformată în imagini grayscale, figura 2:

Fig. 1. Imaginea în format RGB.

Fig.2. Imaginea grayscale.

Urmează prelucrarea imaginii din figura 2 cu algoritmul k-means paralelizat. Imaginea a fost segmentată în 5 clase şi a fost extrasă imaginea ce face parte din clasa numărul 1, figura 3:

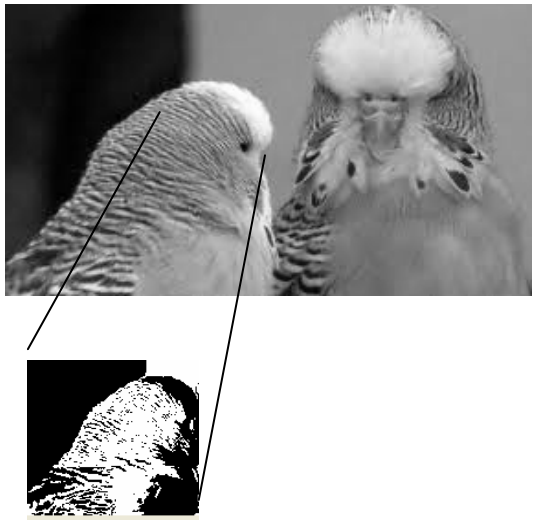


Fig. 3. Extragerea zonei (clasa 1) de interes din imaginea din fig. 2.

Aplicarea algoritmului ISODATA paralelizat pentru setul de intrare adică imaginea din figura 2, segmentează pixelii tot în 5 clase, fiind extrasă imaginea care face parte din clasa 0, figura 4:

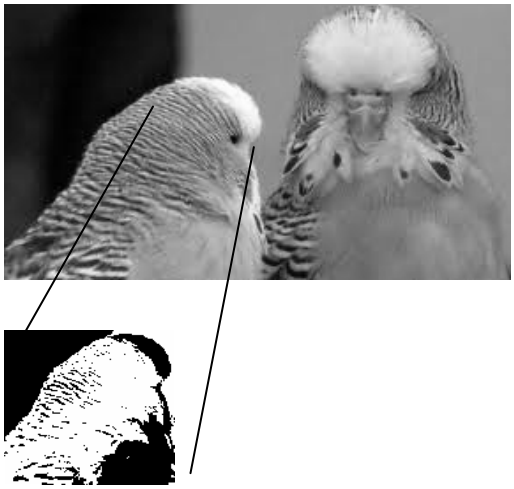
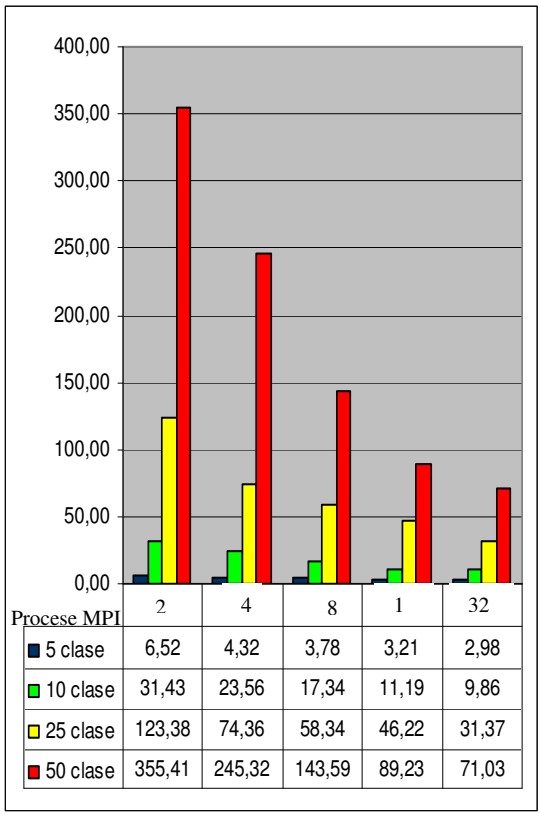


Fig.4. Extragerea zonei (clasa 1) de interes din imaginea din fig. 2.

Prin compararea zonelor de interes extrase, cu cei doi algoritmi paralelizaţi k-means şi ISODATA folosind imaginea din figura 2 observăm că, gruparea pixelilor cu algoritmul ISODATA paralelizat oferă rezultate mult mai bune.

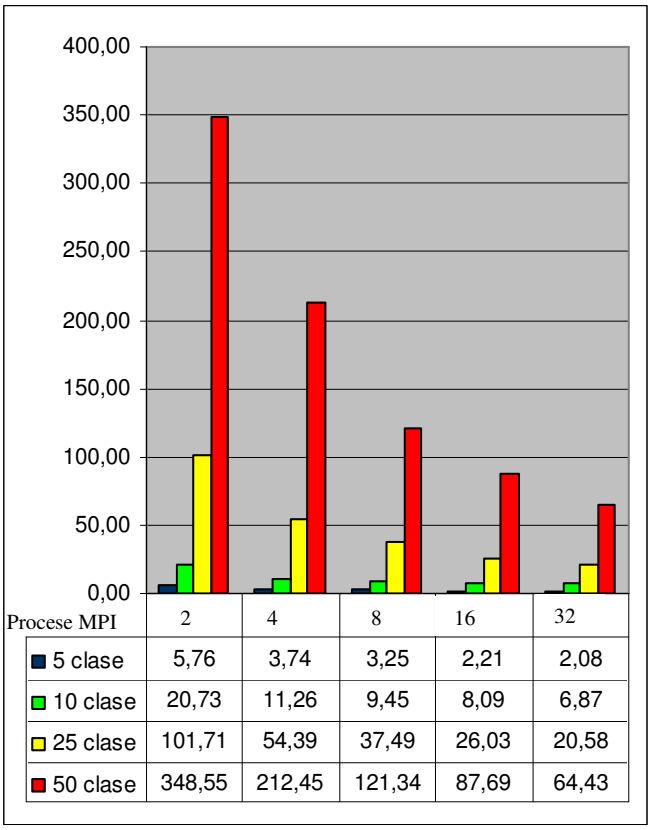
Acum se vor compara timpii de execuţie obţinuţi cu cei doi algoritmi k-means şi ISODATA paraleli.

Pentru setul de intrare adică imaginea din figura 2, s-a realizat gruparea pixelilor în 10, 25, 50, 100 clase. Pentru fiecare grupare aplicaţia a fost executată folosind 2, 4, 8, 16, 32 procese MPI. Rezultatele testelor sunt date în graficele de mai jos:



Procese MPI	2	4	8	1	32
5 clase	6,52	4,32	3,78	3,21	2,98
10 clase	31,43	23,56	17,34	11,19	9,86
25 clase	123,38	74,36	58,34	46,22	31,37
50 clase	355,41	245,32	143,59	89,23	71,03

Fig. 5. Rezultatele experimentale ale algoritmului k-means având setul de date de intrare imaginea din fig.2.



Procese MPI	2	4	8	16	32
5 clase	5,76	3,74	3,25	2,21	2,08
10 clase	20,73	11,26	9,45	8,09	6,87
25 clase	101,71	54,39	37,49	26,03	20,58
50 clase	348,55	212,45	121,34	87,69	64,43

Fig. 6. rezultatele experimentale ale algoritmului ISODATA având setul de date de intrare imaginea din fig.2.

Din aceste rezultate se poate observa că prin creşterea numărului de procese MPI se realizează o scădere semnificativă a timpul de execuţie. De asemenea se observă că timpul de execuţie creşte dacă şi numărul de clase în care sunt grupaţi pixelii creşte. Timpul de execuţie obţinut cu algoritmul ISODATA este mai bun faţă de algoritmul k-means.

X. CONCLUZII

În cadrul acestui articol au fost prezentaţi doi algoritmi paraleli pentru segmentarea imaginilor.

S-a demonstrat că algoritmul paralel ISODATA utilizat în segmentarea imaginii este mult mai bun faţă de algoritmul paralel k-means, regiunea extrasă cu ISODATA este mult mai fidelă cu regiunea din imagine originală. Mai mult timpul de execuţie în algoritmul ISODATA este mai mic faţă de timpul de execuţie în algoritmul k-means.

S-a mai demonstrat că în ambii algoritmi timpul de execuţie scade atunci când numărul de procese creşte şi creşte atunci când numărul de clase creşte.

REFERINŢE

[1] [1] Zhonghui Feng, Bing Zhou, Junyi Shen, A parallel hierarchical clustering algorithm for PCs cluster system, Neurocomputing, Volume 70, Issues 4-6, January 2007, Pages 809-818

[2] [2] Michael W. Berry, Malu Castellanos , Survey of Text Mining II: Clustering, Classification, and Retrieval, Springer, 2008, ISBN 1848000456, 9781848000452

[3] [3] V. Neagoe, O. Stanasila "Teoria recunoaşterii formelor", Editura Academiei, 1992

[4] [4] Kanungo, T.; Mount, D.M.; Netanyahu, N.S.; Piatko, C.D.; Silverman, R.; Wu, A.Y.; An efficient k-means clustering algorithm: analysis and implementation, Pattern Analysis and Machine Intelligence, IEEE Transactions on Volume 24, Issue 7, July 2002 Page(s):881 – 892

[5] [5] R. Vancea, Ştefan Holban, Dan Ciubotariu, "Recunoaşterea formelor.

[6] [6] Pentiuc Ştefan-Gheorghe: Aplicaţii ale recunoaşterii formelor în diagnosticul automat, Editura Tehnică, Bucureşti, 1997.

[7] [7] T.M.Cover, P.E.Hart, Nearest Neighbor Pattern Classification, IEEE Trans. on Information Theory, IT-13, 1, pag. 21-26, (1967)

[8] [8]http://fourier.eng.hmc.edu/e161/lectures/classification/node13.html

[9] [9] Kanungo, T.; Mount, D.M.; Netanyahu, N.S.; Piatko, C.D.; Silverman, R.; Wu, A.Y.; An efficient k-means clustering algorithm: analysis and implementation, Pattern Analysis and Machine Intelligence, IEEE Transactions on Volume 24, Issue 7, July 2002 Page(s):881 – 892 .