Segmentasi PSO Pararel GPGPU Untuk Pengenalan Citra Aksara Jawa

Oskar Ika Adi Nugroho, Dhany Faizal Racma
Sistem Informasi, Sekolah Tinggi Ilmu Komputer Yos Sudarso
Jalan SMP 5 Karang Klesem, Kab.Banyumas, Jawa Tengah 53144
e-mail: oskar@stikomyos.onmicrosoft.com, oskarikaadi@gmail.com

Abstrak

Dalam proses pengenalan karakter, diperlukan proses segmentasi citra. Tujuan dari dilakukan segmentasi citra adalah mengubah representasi dari suatu citra menjadi sesuatu yang lebih berarti dan mudah untuk dianalisis . Segmentasi citra dengan menggunakan metode clustering dapat digunakan dengan berbagai macam metode, salah satunya adalah Particle Swarm Optimization (PSO). Proses segmentasi citra menggunakan metode PSO membutuhkan waktu komputasi yang lama. Pada penelitian ini diterapkan metode parallel programing, dan arsitektur komputer yaitu GPU dengan support CUDA untuk mempercepat proses komputasi. Hasil akhir dari Penelitian ini dapat mengimplementasikan segmentasi citra digital dengan metode Particle Swarm Optimization pada Aksara Jawa dan mengimplementasikan metode backpropagation dari pelatihan jaringan syaraf tiruan untuk melakukan pengenalan karakter Aksara Jawa pada Aksara Jawa. GPU CUDA terbukti mempercepat proses segmentasi citra Aksara Jawa. Persentase rata rata akurasi hasil pengenalan tiap karakter untuk jenis font yang sama dengan yang dilatihkan mencapai 75%.

Kata kunci: *aksara jawa, segmentation, character recognition, PSO, backpropagation neural network, , GPGPU*

Abstract

In the process of character recognition, image segmentation process is required. The purpose of the image segmentation is to change the representation of an image into something more meaningful and easy to analyze. Image segmentation using clustering method can be used with various methods, one of them is Particle Swarm Optimization (PSO). The process of image segmentation using the PSO method takes a long time of computing. In this research applied parallel programming method, and computer architecture is GPU with CUDA support to speed up computation process. The end result of this research can implement digital image segmentation with Particle Swarm Optimization method in Java script and implement backpropagation method from artificial neural network training to perform character recognition of Java script in Java script. CUDA GPU proven to accelerate the process of image segmentation Java script. The average percentage of accuracy of each character's recognition for the same font type as the trained one reaches 75%

.***Keywords****: Javascript, segmentation, character recognition, PSO, backpropagation neural network, , GPGPU*

1. Pendahuluan

Indonesia adalah negara yang dimana terdapat beraneka suku dan budaya. Masing-masing suku dan budaya di Indonesia memiliki ciri khas, begitu juga dengan bentuk tulisan. Salah satu suku di Indonesia yang memiliki ciri khas dalam tulisan adalah suku Jawa yang hurufnya disebut dengan Aksara Jawa. Seiring cepatnya perkembangan teknologi informasi dan komunikasi. Banyak sekali tulisan-tulisan peninggalan nenek moyang yang hampir terlupakan oleh generasi masa kini.

Perkembangan dibidang pengolahan citra sering digunakan untuk riset dan pengembangan aplikasi dan teknologi. Banyak sekali metode dan algoritma yang diciptakan dan digunakan untuk mempermudah kinerja citra pada suatu obyek maupun media. Kemajuan pengolahan citra juga dapat digunakan dalam melestarikan ilmu dan budaya. Salah satu penerapannya dapat digunakan dalam melestarikan aksara jawa yang hingga kini hampir punah karena perkembang jaman.

OCR (*Optical Character Recognition*) merupakan salah satu area studi dalam bidang pengenalan pola [1].OCR merupakan solusi yang efektif untuk proses konversi dari dokumen cetak ke dalam bentuk dokumen digital [1]. Berdasarkan cara mendapatkan input , OCR dapat dikategorikan dalam dua bagian

yaitu *offline* dan *online recognition*. Pada *offline recognition*, *input* yang digunakan dipindai menggunakan *scanner* sebelum diproses. Sedangkan dalam *online recognition*, *input* langsung didapatkan pada saat *input* tersebut dituliskan diatas layar sentuh (*touch screen*). Kualitas input akan menentukan kinerja dari sistem OCR yang dibangun.

Berdasarkan latar belakang diatas penulis tertarik teknik Particle Swarm Optimization (PSO) untuk melakukan segmentasi citra Aksara Jawa memanfaatan parallel computing GPU CUDA dan pengenalan karakter Aksara Jawa dengan menggunakan metode Jaringan Syaraf Tiruan Propagasi Balik (backpropagation neural network).

2. Tinjauan Pustaka

Segmentasi Citra Teks Sastra Jawa, menggunakan metode profil proyeksi telah dilakukan [2] dan tingkat kesuksesan 86,78% mensegmentasi dokumen teks Sastra Jawa. Pengenalan pola huruf Jawa dengan metode Learning Vector Quantization (LVQ) dilakukan [3]. Presentase ketepatan yang diperoleh dari percobaan pada penelitian ini adalah 56,61 %. Pengenalan pola karakter huruf jawa dengan metode Backpropagation Neural Network dilakukan oleh [4] dan hasilnya rata-rata keakuratan *Backpropagation Neural Network* dalam mengenali pola karakter huruf Jawa adalah sebesar 99.563% untuk data sampel berupa data pelatihan, 61.359% untuk data sampel diluar data pelatihan, dan 75% untuk data sampel data pelatihan dan di luar data pelatihan.

Upaya melestarikan dan mengekstrak pengetahuan implisit naskah Jawa kuno oleh [5] berhasil mengevaluasi algoritma ekstraksi fitur antara lain arah jalur lokal, mesh dan pendekatan yang berbeda lainnya dalam hal tingkat klasifikasi Machine Support Vector (MSV). Algoritma Widiarti-Winarko [6] berhasil menerjemahkan dokumen kuno yang ditulis dalam karakter Jawa dengan hasil 62,96%, dan 75% menemukan kata yang benar dalam bahasa Jawa.

Particle Swarm Optimization telah digunakan untuk mengatasi berbagai masalah masalah pada kajian segmentasi dan pengenalan pola [7] [8] [9] [10] [11] [12] [13] [14] [15] [16] [17] [18] [19]. *Particle Swarm Optimization* bisa melakukan segmentasi citra dengan hasil yang memuaskan [20]. PSO memberikan kinerja yang menjanjikan dan perilaku yang stabil dalam mengenali angka tulisan tangan [21]. Penggabungan algoritma *Particle Swarm Optimization* dan *Backpropagation* telah dilakukan [22].Pengabungan algoritma *Particle Swarm Optimization* dan Backpropagation juga telah dilakukan [23]untuk pengenalan pola tulisan tangan.

Penggunaan teknologi GPU banyak diterapkan juga pada algoritma Particle Swarm Optimization [24] [25] [26]. Jika menggunakan NVIDIA CUDA untuk mempararelkan komputasi, PSO berjalan lebih cepat 170% pada GPU, daripada berjalan pada CPU dengan jumlah partikel 100. [27] Pada permasalahan konvergensi untuk optimalisasi fungsi *Rastrigin* dan fungsi *Ackley* di ruang pencarian 30-dimensi dengan menggunakan *General Purpose Graphics Processing Unit* (GPGPU) untuk pemrosesan paralel dari PSO, fungsi *Rastrigin* berjalan lebih cepat 16 kali dari komputer cluster dan untuk masalah *Ackley* itu lebih cepat 8 kali daripada komputer cluster. [28] Teknik yang didasarkan pada metode Sauvola-Pietkinen untuk mendeteksi tepi dengan kesinambungan menunjukkan bahwa algoritma PSO yang memanfaatkan teknik thresholding lokal baru, lebih baik daripada yang menggunakan metode Otsu [29]). Masalah estimasi parameter yang realistis di mana setiap prosesor melakukan perhitungan yang signifikan dengan metode PSO berhasil menunjukkan peningkatan percepatan ditandai dengan ukuran populasi [30] Implementasi clustering K-Means secara pararel telah dilakukan [31] menghasilkan peningkatan kinerja 13kali.

Telah dilakukan implementasi dari algoritma backpropagation pada CUDA. Implementasi diuji dengan dua set patokan data standar dan hasilnya menunjukkan bahwa algoritma pelatihan paralel berjalan 63 kali lebih cepat. [32] Hasil komputasi algoritma pelatihan untuk jaringan saraf yang mengandung operasi matriks matematika, sangat baik jika jika menggunakan GPU [33].Teknik untuk pelatihan jaringan saraf yang rumit pada GPU (tersedia dalam *PC low end*) memungkinkan pelatihan yang dilakukan dengan waktu utilisasi CPU yang minimal. [34]. Penelitian [35] pada *neural network convolutional* menggunakan GPGPU menghasilkan emulasi yang lebih buruk daripada program yang ditulis native untuk CPU. Paralelisasi telah dilakukan pada algoritma pelatihan jaringan syaraf pada arsitektur heterogen [36]. [37] melakukan penelitian menggunakan GPU pada model perilaku pasar keuangan menggunakan *Evolving Neural Networks*. CUDA meningkatkan performansi hingga 80% dibandingkan dengan implementasi pada CPU pada Simulasi paralel neural network [38].

3. Metode Penelitian

Metode yang akan digunakan dalam penelitian ini terdiri dari langkah-langkah berikut:

1. Pengumpulan Data

Pengumpulan data untuk penelitian ini dilakukan studi pustaka atau literatur yaitu pengumpulan data dengan membaca buku-buku referensi yang terkait dengan penelitian ini. Studi pustaka antara lain mencari jurnal-jurnal tentang *Particle Swarm Optimization (PSO)*, Jaringan Syaraf Tiruan, mempelajari *CUDA* dan *Qt*, mencari *typefont* aksara jawa.

2. Rancang Bangun Perangkat Lunak Pengenalan Aksara Jawa terdiri dari empat tahap utama :
    a. Analisis sistem Aplikasi. Pada tahap ini akan dilakukan analisa kebutuhan perangkat lunak yang akan dikembangkan.
    b. Desain sistem Aplikasi. Pada tahap ini akan dilakukan perancangan model perangkat lunak yang akan dikembangkan.
    c. Pengkodean sistem Aplikasi. Pada tahap ini akan dilakukan proses penulisan program untuk merealisasikan rancangan sistem dengan menggunakan bahasa pemrograman.
    d. Pengujian sistem Aplikasi. Pada tahap ini akan dilakukan proses pengujian fungsionalitas sistem Aplikasi yang telah dikembangkan.

Sistem yang diusulkan mengandung tiga subsistem. Subsistem prapengolahan, subsistem ekstraksi fitur dan subsistem pengenalan karakter. Pada subsistem prapengolahan, sistem akan melakukan segmentasi pada citra menggunakan metode *Particle Swarm Optimization*. Pada subsistem ekstraksi fitur, setiap karakter Aksara Jawa akan dikonversi menjadi nilai vektornya. Pada subsistem pengenalan karakter, akan digunakan metode jaringan syaraf tiruan backpropagation untuk mengenali karakter Aksara Jawa.

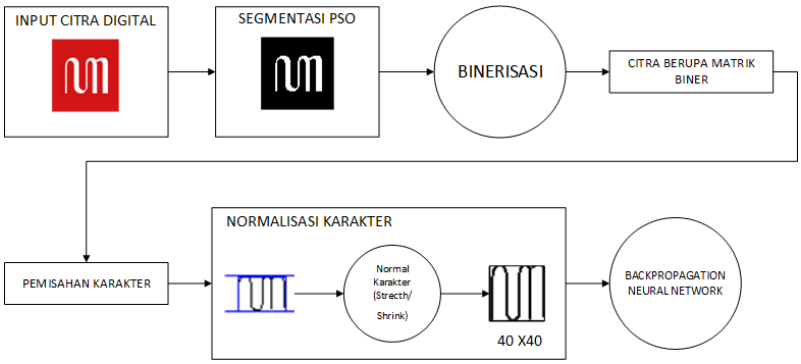
3.1. Proses

Rangkaian proses yang akan dilakukan oleh aplikasi ini adalah baca gambar, segmentasi dengan PSO, binerisasi, *grayscaling*, *thresholding*, pemisahan karakter, normalisasi hasil segmentasi, ekstrasi ciri (*encode input*) dan *backpropagation neural network*. Proses *backpropagation neural network* sendiri membutuhkan proses pelatihan (*training*) agar *output* yang dihasilkan benar dan akurat.

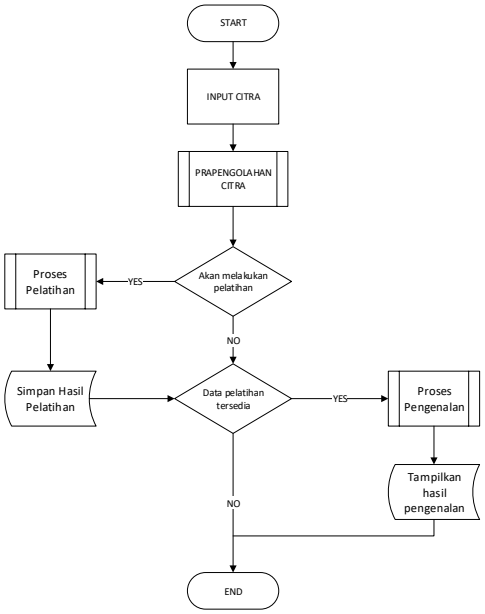
Jadi secara garis besar dapat dikatakan bahwa alur aplikasi ini dimulai dari diinputkannya sebuah gambar dari sebuah gambar teks dengan latar belakang berwarna. Di mana gambar tersebut akan melalui serangkaian manipulasi citra dalam proses *image preprocessing* diantaranya proses segmentasi PSO, *grayscaling*, *thresholding*, pemisahan karakter, normalisasi dan ekstrasi ciri yang akan mengubah gambar tersebut menjadi serangkaian nilai-nilai yang berguna untuk proses utama berikutnya yaitu : proses pelatihan (*training process*) ataupun proses pengenalan (*recognition process*).

Output dari proses pelatihan adalah sekumpulan nilai bobot jaringan dan vektor pola stabil yang disimpan ke dalam sebuah file pelatihan, file pelatihan tersebut dapat dibuka kembali bila akan diupdate ataupun digunakan dalam proses pengenalan. Sedangkan output dari proses pengenalan adalah konversi dari citra yang berisi angka-angka bipolar menjadi huruf-huruf atau angka yang dapat disimpan ke dalam file teks.

Dalam hal ini perlu diperhatikan bahwa sebelum melakukan proses pengenalan, terlebih dahulu dilakukan pengecekan apakah sudah ada data pelatihan (training) yang dapat dijadikan acuan dalam melakukan pengenalan. Bila tidak, maka proses pengenalan tidak bisa dilakukan.



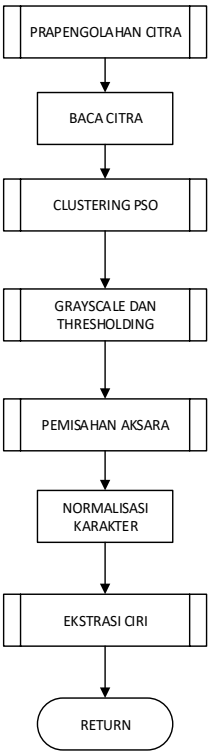
Gambar 1. Alur Proses Pengenalan Karakter



Gambar 2. *Flowchart* Proses Keseluruhan

3.2. Proses Prapengolahan

Proses ini merupakan proses yang pertama kali dijalankan bila user hendak melakukan proses pelatihan maupun proses pengenalan. Pertama-tama, citra dibuka akan dibaca. Kemudian dilakukan proses segmentasi *Particle Swarm Optimization Clustering*. Langkah selanjutnya adalah *grayscaling* dan *thresholding* hingga menjadi gambar hitam-putih untuk memudahkan dalam pengolahan data selanjutnya. Proses selanjutnya adalah mencari baris demi baris yang ada pada gambar dan memisahkankan tiap karakter yang ada pada citra hasil prapengolahan tersebut. Proses pengambilan tiap-tiap karakter yang ada pada gambar ini sangatlah penting karena tanpa akurasi input yang baik, maka proses pengenalan (*recognition*) tidak akan berfungsi dengan baik.



Gambar 3. *Flowchart* Prapengolahan Citra

3.3. *Particle Swarm Optimization Clustering*

Metode yang digunakan dalam melakukan segmentasi pada citra digital pada perangkat lunak ini adalah metode clustering. Algoritma clustering yang digunakan adalah *Particle Swarm Optimization* (PSO). Menurut [7] dalam konteks clustering, suatu partikel tunggal merepresentasikan centroid cluster sebanyak N_c. Sehingga untuk setiap partikel X_1, dapat direpresentasikan sebagai berikut :

$$X_1 = (m_{i1}, \dots\dots, m_{ij}, \dots\dots m_{iN_c})$$

dimana m_{ij} merupakan *centroid cluster* ke-*j* pada partikel ke-*I*, pada cluster C_{ij}. Dari representasi artikel di atas, maka dapat disimpulkan bahwa kumpulan (*swarm*) dari *p centroid* merepresentasikan sejumlah *clustering*, sebanyak jumlah partikel, pada data tertentu.

Fungsi *fitness* dari suatu partikel dapat dihitung dengan menggunakan quantization error yang dinyatakan sebagai berikut :

$$f = \frac{\sum_{j=1}^{N_c} \frac{\sum_{\forall Z_p \in Z} d(Z_p, m_j)}{|Z|}}{N_c}$$

dimana, *Z* merupakan kumpulan data yang di *cluster*, dan *d* merupakan fungsi jarak *euclidean* yang dinyatakan dalam persamaan berikut :

$$d(Z_p, m_j) = \sqrt{\sum_{i=1}^{d_m} (Z_{pi} - m_{ji})^2}$$

dimana d_m merupakan dimensi dari data yang di *cluster*.

3.4. Proses *Grayscaling* dan *Thresholding*

Pada proses *grayscaling* ini citra input yang berwarna dapat diubah menjadi gambar yang tediri dari warna putih dan gradiasi warna hitam dengan menggunakan representasi warna RGB. Pengubahan gambar ke dalam bentuk *grayscale* ini dilakukan dengan mengambil nilai *pixel* dari suatu gambar *input* yang kemudian dihitung dengan persamaan yang ada yaitu :

$$Gray = 0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B$$

Proses *thresholding* dilakukan dengan cara memeriksa apakah nilai intensitas dari sebuah *pixel* berada di bawah atau di atas sebuah nilai *intensity threshold* yang telah ditentukan. Apabila nilai *pixel* tersebut berada di atas batas nilai yang telah ditentukan, maka *pixel* tersebut akan diubah menjadi putih yang berarti bahwa *pixel* tersebut merupakan *background*, dan sebaliknya bila *pixel* tersebut berada di bawah batas nilai yang ditentukan maka *pixel* tersebut akan diubah menjadi berwarna hitam yang berarti dianggap sebuah karakter.

Citra *input* yang berformat *bitmap* akan diubah ke dalam sebuah matrik yang berukuran sesuai dengan ukuran *pixel* dari *input* image tersebut. Selanjutnya akan dilakukan perubahan ke dalam vektor matrik biner yang hanya bernilai 0 atau 1 pada setiap *pixel*-nya. Setelah didapatkan nilai *grayscale* kemudian nilai pada setiap pixel hasil *grayscale* akan dilakukan proses *thresholding* yang akan menyebabkan setiap *pixel* hanya bernilai 0 atau 1.

3.5. Pemisahan Karakter

Tujuan pemisahan karakter adalah memisahkan tiap karakter dari baris teks. Pada proses pemisahan karakter dilakukan dengan memetakan jumlah titik hitam setiap baris pada gambar ke sumbu *y* (*Y-Mapping)* dan setiap baris karakter hasil pemetaan tersebut dipetakan lagi ke sumbu *x* (*X-Mapping).*

3.6. Normalisasi Karakter

Karena data yang dihasilkan dari proses pemisahan karakter dapat berbeda-beda dimensinya. Oleh karena itu untuk mendapatkan data yang seragam, akurat dan konsisten dari setiap sampel, data gambar hasil segmentasi tersebut akan dinormalisasikan (*stretch/shrink)* dengan ukuran 40 x 40 *pixel*.

Dengan demikian jumlah area yang ada pada setiap sampel akan bersesuaian dengan jumlah *neuron input* jaringan syaraf tiruan yang akan digunakan.

3.7. Ektraksi Ciri

Setelah didapatkan informasi matrik hasil proses normalisasi setiap karakter dalam dimensi matrik 40 x 40, selanjutnya adalah mengektraksi setiap *pixel* dari citra ke dalam sebuah vektor hal ini dilakukan agar dapat dihasilkan kumpulan data yang seragam pada setiap sampel yang akan diamati. Ciri-ciri citra adalah *pixel-pixel* yang memiliki nilai 1 dan 0. *Pixel* bernilai 0 adalah *pixel* berwarna hitam atau objek, sebaliknya *pixel* bernilai 1 adalah *pixel* berwarna putih atau *background*. Proses ekstraksi bertujuan untuk menangkap ciri tertentu dari citra input.

3.8. Pelatihan (*Training Process*)

Pada proses pelatihan ini, jaringan akan dilatih berdasarkan vektor *input* gambar huruf dan angka yang sudah disiapkan sebelumya untuk proses pelatihan. Dalam proses ini huruf dan angka dilatihkan satu per satu sesuai dengan *input* vektornya. Adapun *output* dari modul ini adalah file data informasi jaringan saraf tiruan yang berisi bobot koneksi antar *neuron* dari semua pola vektor yang terbentuk dan vektor pola yang dipanggil secara stabil (konvergen).

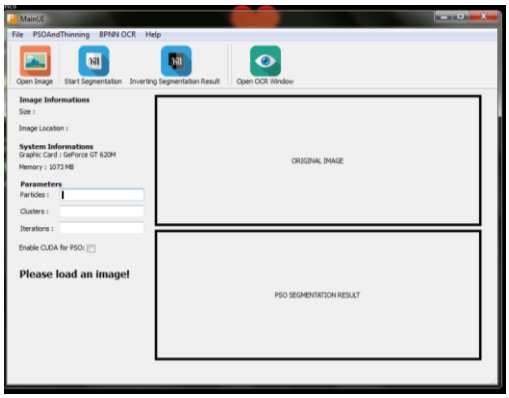
3.9. *Backpropagation Neural Network*

Metode *backpropagation* sering juga disebut dengan *generalized delta rule*. *Backpropagation* adalah metode turunan gradien (*gradient descent method*) untuk meminimalkan *total square error* dari *output* yang dikeluarkan oleh jaringan. Metode ini memiliki dasar matematis yang kuat, obyektif dan algoritma ini mendapatkan bentuk persamaan dan nilai koefisien dalam formula dengan meminimalkan jumlah kuadrat *galat error* melalui model yang dikembangkan (*training set*).

4. Hasil dan Pembahasan

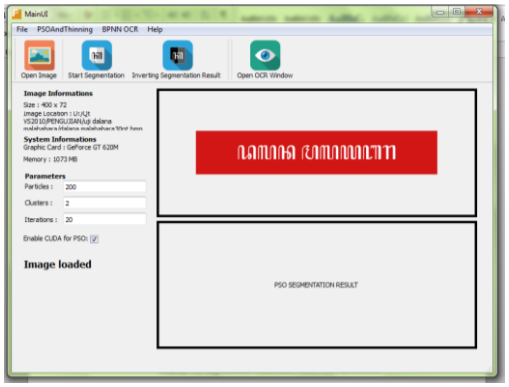
4.1. Implementasi Antarmuka Proses PSO

Pada saat aplikasi PENAWA dijalankan, maka kan tertampil antarmuka seperti pada gambar 4. Informasi mengenai GPU yang digunakan oleh pengguna akan tertampil jika pengguna menggunakan GPU yang telah mendukung NVIDIA CUDA.



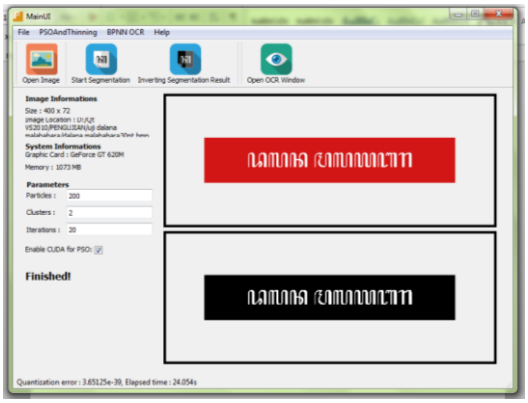
Gambar 4. Implementasi Antarmuka Proses PSO PENAWA 1

Untuk memulai proses segmentasi, pertama tama, pengguna diminta untuk memilih citra yang akan diproses, dengan cara memilih menu File, kemudian Load Image atau menekan tombol Open Image pada pojok kiri atas antar muka. Citra yang terpilih akan tampil pada bagian kanan atas antar muka (*Original Image*) seperti pada gambar 5.



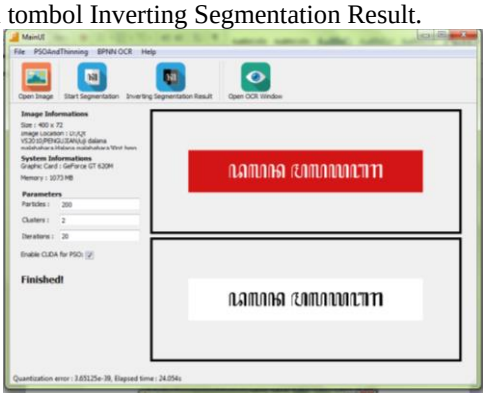
Gambar 5 Implementasi Antarmuka Proses PSO PENAWA 2

Untuk memulai segmentasi citra, user diminta untuk mengisi parameter parameter yang diperlukan, yaitu jumlah partikel pada *textbox Particle*, jumlah cluster pada *textbox Clusters*, dan iterasi maksimal pada *textbox Iteration*, kemudian memilih apakah proses dilakukan dengan menggunakan CUDA dengan memilih pada *checkbox Enable CUDA*. Kemudian *user* menekan tombol *Start Segmentation* untuk memulai proses. Hasil segmentasi citra akan tertampil pada panel Image *PSO Segmentation Result*, dan waktu pemrosesan akan tertampil di bagian kiri bawah. Setelah proses selesai, maka akan tertampil seperti pada gambar 6.



Gambar 6. Implementasi Antarmuka Proses PSO PENAWA 3

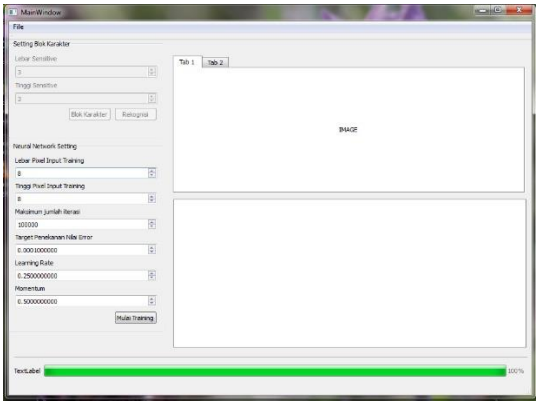
Setelah proses segmentasi selesai dilakukan, *user* bisa melakukan proses *inverting* citra hasil segmentasi dengan menekan tombol Inverting Segmentation Result.



Gambar 7. Implementasi Antarmuka Proses PSO PENAWA 4

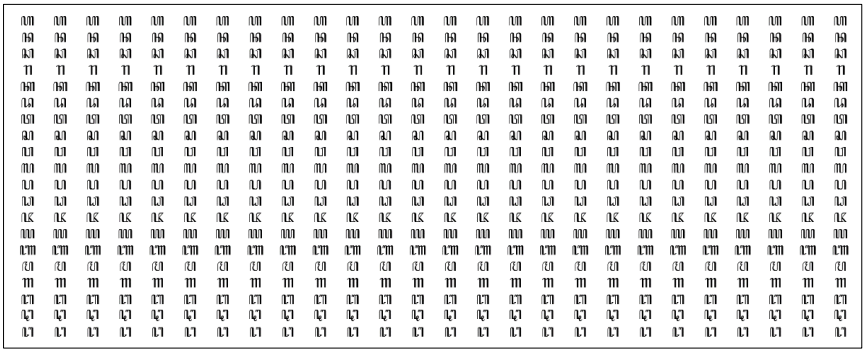
4.2. Implementasi Antarmuka Proses Pengenalan Karakter

Langkah selanjutnya adalah melakukan pengenalan citra dengan menekan tombol Open OCR Window. Akan muncul antar muka baru, antar muka ini digunakan sebagai antar muka pengenalan karakter dari perangkat lunak.



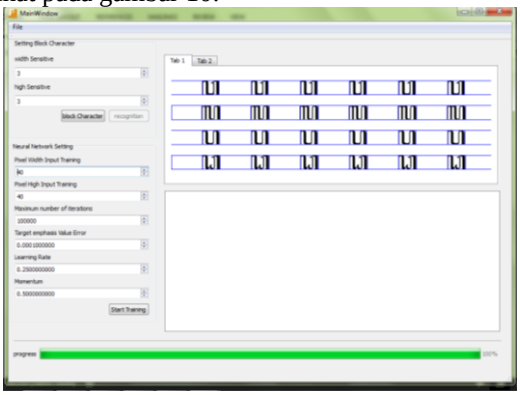
Gambar 8. Implementasi Antarmuka Proses Pengenalan Karakter 1

Sebelum dapat melakukan pengenalan pola aksara jawa, dilakukan proses pelatihan (*training*). Pada proses ini sistem melakukan proses pelatihan di mana keduapuluh citra aksara jawa dasar (*Aksara nglegéna*) dilatihkan tiap karakter. Jenis *font* karakter aksara jawa yang dilatihkan, yaitu *Hanacaraka* 33pt yang seperti pada gambar 9



Gambar 9. Contoh Citra *Training*

Pertama tama, *user* diminta untuk memilih citra yang akan diproses untuk *training*, dengan cara memilih menu File, kemudian Open Image. Citra yang terpilih akan tampil pada bagian kanan atas antar muka. Aplikasi akan secara otomatis memisahkan obyek karakter per baris, dengan cara mencari batas atas perbaris karakter citra dan kanan kiri karakter serta mengepaskan kembali batas atas dan batas bawah karakter seperti terlihat pada gambar 10.

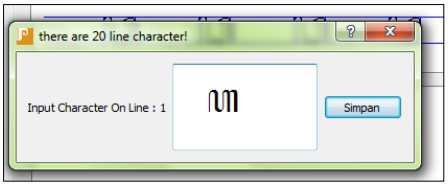


Gambar 10. Implementasi Antarmuka Proses Pengenalan Karakter 2

Untuk memulai proses *training*, *user* diminta memasukkan parameter parameter yang diperlukan. Parameter parameter tersebut adalah lebar *pixel* input training pada *textbox* Pixel Width Input Training, tinggi pixel input training pada *textbox* Pixel High Input Training, maksimum jumlah iterasi pada *textbox* Maximum number of iterations, target penekanan nilai error pada *textbox* Target

emphasis Value Error, *learning rate* pada *textbox* Learning Rate, momentum pada *textbox* Momentum. Setelah semua parameter dimasukkan, proses selanjutnya adalah menekan tombol Start Training.

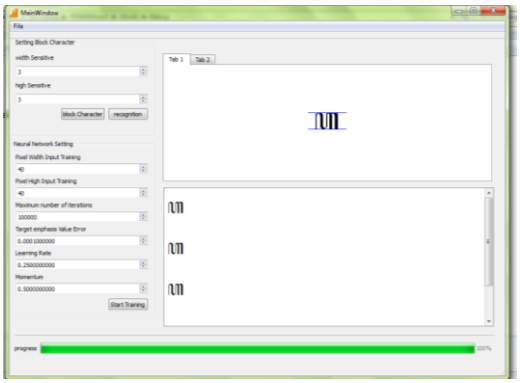
Setelah tombol Start Training ditekan, akan muncul jendela dialog input baris karakter. Aplikasi akan menemukan berapa karakter yang ada. Kemudian pada tiap baris karakter ditentukan jenis aksara jawanya. Seperti pada gambar 11.



Gambar 11. Implementasi Antarmuka Proses Pengenalan Karakter 3

Setelah proses training selesai, *user* dapat menyimpan hasil *training* dalam *file format* .txt, dengan memilih menu File kemudian Save Training Result.

Untuk dapat melakukan pengenalan karakter aksara jawa, *file format* .txt hasil dari training dibuka dahulu dengan memilih menu File kemudian Open Training Result. Setelah itu tekan tombol recognition untuk pengenalan karakter. Hasil pengenalan karakter akan muncul di *textbox* kanan bawah. Seperti pada gambar 12.



Gambar 12**.** Implementasi Antarmuka Proses Pengenalan Karakter 4

4.3. Pengujian Algoritma PSO

Pengujian dilakukan dengan dengan komputer *Processor* Core i3-2350M, *Memory RAM* 4.0 GB, *Video Card Onboard* NVIDIA GeForce GT 620M 1GB.

Terdapat kedupuluh citra aksara jawa dasar (*Aksara nglegéna*) untuk masing masing implementasi (CPU, Naif, *full device*) yang dibagi menjadi 4 buah citra uji. Keempat citra digital tersebut diuji dengan nilai *cluster* sebanyak 2. Jumlah partikel 20, 100 dan 1000. Jumlah iterasi sebanyak 20, 40, dan 60. Keduapuluh citra aksara jawa dasar (*Aksara nglegéna*) yang digunakan sebagai bahan uji adalah empat citra berformat JPG dan memiliki ukuran 300 X 40 pixel, sehingga tiap citra berukuran 5184 data.

Dari hasil pengujian diatas didapatkan bahwa implementasi segmentasi citra dengan menggunakan PSO secara pararel memberikan peningkatan performa pemrosesan yang signifikan dibandingkan dengan segmentasi citra dengan PSO yang tidak pararel.

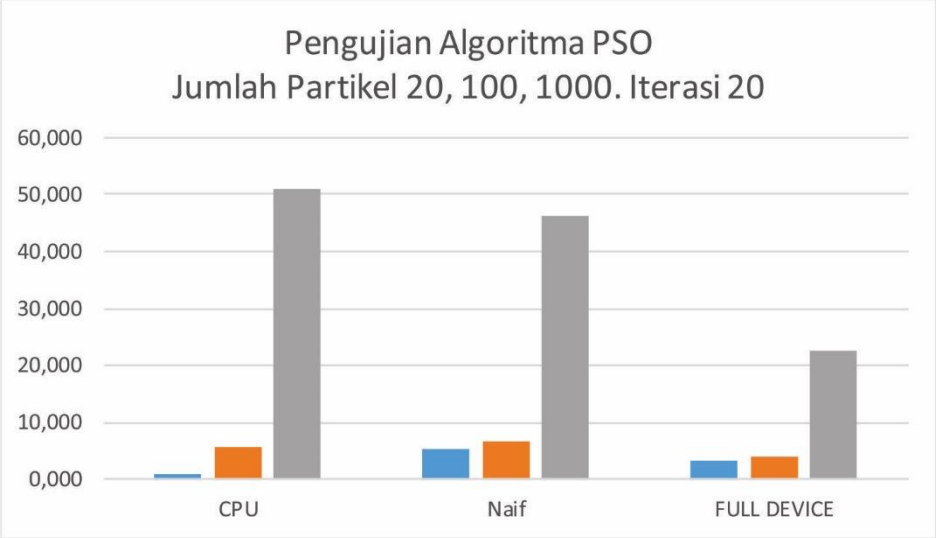
Jika diketahui *n* adalah jumlah partikel dan *i* adalah jumlah iterasi, maka kompleksitas waktu *worst case* dari implementasi diatas adalah :

1. $O(i * 2n)$ pada implementasi CPU, karena terdapat 2 buah *loop* sebanyak n elemen di dalam *loop* i.
2. $O(i * n)$ pada implementasi naif, karena *loop* yang digunakan untuk meng*update* partikel dipararelkan sehingga kompleksitas *loop* tersebut menjadi $O(1)$ atau konstan.
3. $O(i * \lg n)$ pada implementasi *full device*, karena loop untuk meng*update* partikel dan mencari gBest dipararelkan sehingga menjadi $O(1)$ (konstan), dan untuk mencari nilai *fitness* terkecil digunakan algoritma *parallel reduction* yang memiliki kompleksitas $O(\lg n)$.

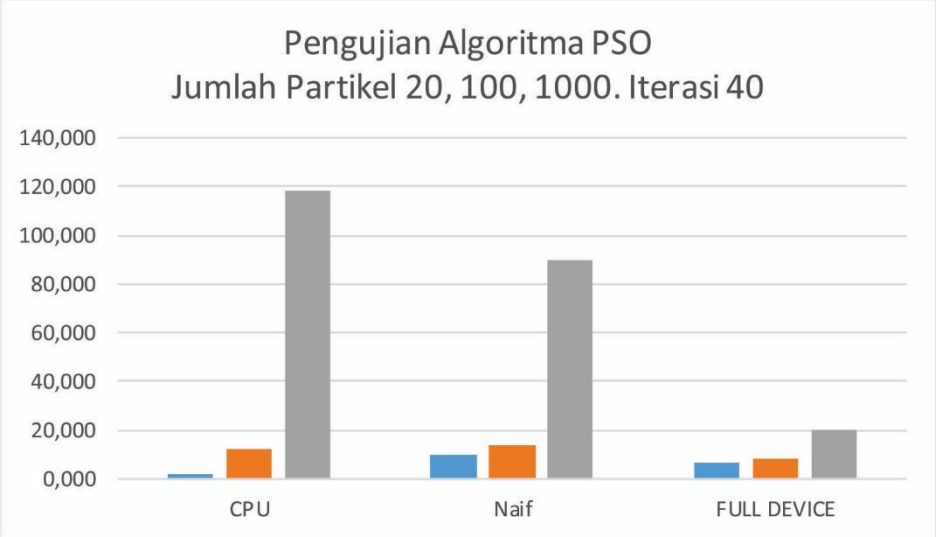
Dilihat dari kompleksitas waktu setiap implementasi diatas, maka secara teoritis, untuk jumlah partikel cukup banyak, implementasi *full device* adalah yang tercepat karena waktu pemrosesannya relatif terhadap logaritma basis dua dari jumlah partikel, diikuti oleh implementasi naif, dan iplementasi pada CPU adalah implementasi yang paling lambat.

Untuk jumlah partikel yang cukup sedikit (20 partikel), implementasi pada CPU adalah implementasi yang berjalan paling cepat. Hal itu disebabkan karena peningkatan kecepatan kecepatannya tertutupi oleh *overhead* pada bagian lain, misalnya inisialisasi memori, *copy* data dan inisialisasi *kernel*.

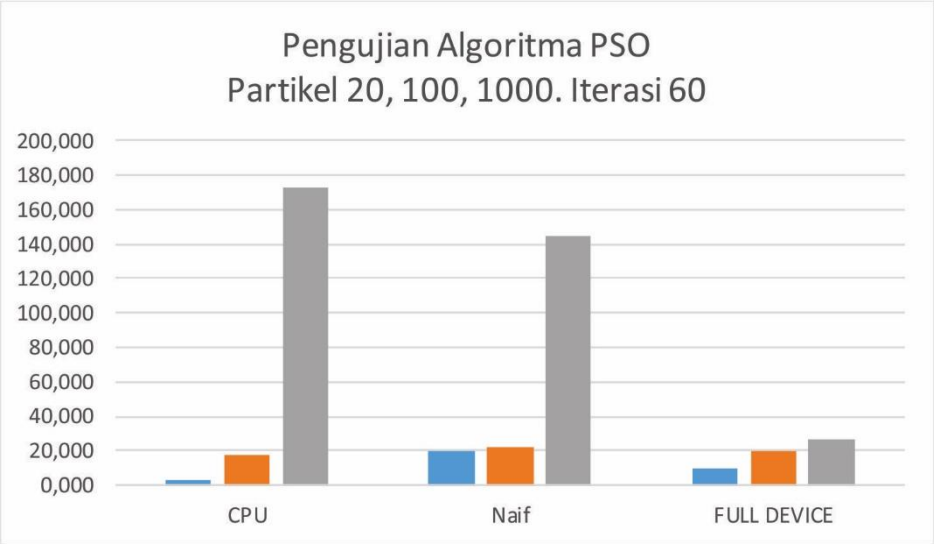
Pada jumlah partikel yang cukup bannyak (100 dan 1000 partikel), implementasi pada GPU berjalan lebih cepat daripada implementasi pada CPU karena *overhead* pada pemrosesan pararel pada CUDA tidak signifikan dibandingkan peningkatan pemrosesannya. Implementasi *full device* adalah implementasi yang paling cepat, yaitu hampir setengah dari waktu pemrosesan pada implementasi CPU, diikuti oleh implementasi naif.



Gambar 13**.** Hasil Pengujian Algoritma PSO 20 iterasi



Gambar 14**.** *Hasil Pengujian Algoritma PSO 40 iterasi*



Gambar 15**.** *Hasil Pengujian Algoritma PSO 60 iterasi*

4.4. Analisis Proses Pelatihan

Pada proses ini sistem melakukan proses pelatihan di mana tiap aksara jawa dilatihkan tiap karakter sesuai dengan nilai input vektornya. Jenis karakter yang dilatihkan adalah *Hanacaraka* 30pt. Berikut nilai-nilai parameter yang digunakan dalam proses pelatihan :

Maksimum Iterasi tiap pola : 10000

Sedangkan parameter yang digunakan sebagai arsitektur JST adalah sebagai berikut

Pola Input : Tinggi = 40, Lebar = 40
Pola Output : Tinggi = 40, Lebar = 40
Jumlah Neuron : 1600

Proses yang terjadi pada pelatihan adalah citra masukan tersebut diubah ke dalam citra biner dengan menggunakan metode *grayscaling* dan *thresholding*. Intensitas tiap-tiap *pixel* citra yang diubah ke dalam citra biner yang kemudian dirubah ke dalam bentuk bipolar 1 dan 0 dalam sebuah matrik vektor yang sudah disiapkan, kemudian vektor *input* di *encode* untuk menghasilkan sebuah *input* yang digunakan dalam JST.

4.5. Pengujian Sistem Pengenalan Karakter

Pengujian dilakukan dengan dengan komputer *Processor* Core i3-2350M, *Memory RAM* 4.0 GB, *Video Card Onboard* NVIDIA GeForce GT 620M 1GB. Untuk pengujian pertama dilakukan dengan citra yang berisi karakter yang merepresentasikan setiap huruf dan angka dengan ukuran 30pt, 40pt dan 50pt.

Dalam pengujian ini dilakukan dengan tujuan apakah sistem bisa mengenali dengan baik tiap pola karakter dengan membandingkan terhadap ukuran karakter yang sama dengan yang dilatihkan. Untuk beberapa karakter untuk ukuran *font* tertentu sistem tidak bisa mengenali karakter tersebut dengan benar, ini dimungkinkan disebabkan oleh proses normalisasi yang dilakukan oleh sistem, karena pada proses normalisasi (*stretch*) semua karakter diseragamkan ukurannya menjadi 40 x 40, hal inilah yang kemudian menjadikan sistem salah dalam mengenali pola yang dimaksud dikarenakan terjadinya perubahan nilai vektor input suatu karakter akibat proses normalisasi tersebut sehingga jaringan stabil pada pola lain.

Pada sistem yang dibuat didesain dengan asumsi citra yang digunakan sebagai *input* memiliki *noise* yang sekecil mungkin karena bisa menurunkan akurasi JST dalam melakukan pengenalan, walaupun dalam perspektif penglihatan mata manusia masih bisa terbaca sebagai karakter itu sendiri tetapi tidak dengan sistem. Akibat adanya *noise* tersebut yang memungkinkan sistem tidak bisa mengenali karakter yang dimaksud dengan benar sehingga sistem akan mengambil dan menampilkan suatu pola yang memiliki kedekatan antara vektor *input* dengan pola target.

Keberhasilan sistem dalam mengenali suatu karakter sangat berpengaruh dari hasil pemisahan aksara yang baik. Kemungkinan terbesar kesalahan pengenalan pada ukuran *font* yang lebih besar adalah kesalahan pada saat proses normalisasi yang menyebabkan JST stabil pada pola lain.

4. Simpulan

Dari hasil penelitian dapat ditarik beberapa kesimpulan yaitu :

1. Perangkat lunak untuk segmentasi citra digital aksara jawa dengan PSO yang berjalan pada CPU dan GPU dengan CUDA telah berhasil dibangun.
2. Perangkat lunak untuk pengenalan aksara jawa dengan algoritma *backpropagation neural network* telah berhasil dibangun.
3. Secara umum, segmentasi citra digital dengan PSO yang berjalan pada GPU berjalan lebih cepat dibandingkan dengan segmentasi citra digital dengan PSO yang berjalan pada CPU. Dengan percepatan pada segmentasi citra digital dengan PSO yang berjalan pada GPU sekitar dua kali lipat dibandingkan segmentasi citra digital dengan PSO yang berjalan pada CPU.
4. Kualitas hasil *clustering* dari algoritma PSO yang berjalan pada GPU sebanding dengan kualitas hasil *clustering* dari algoritma PSO yang berjalan pada CPU.
5. Pada proses pengenalan karakter aksara jawa, rata-rata kesalahan pengenalan terdapat pada karakter yang memiliki kemiripan bentuk.
6. Rata-rata hasil pengenalan untuk setiap jenis ukuran karakter yang tidak dilatihkan bergantung pada ukuran karakter yang telah dilatihkan, apabila jenis karakter tersebut mirip dengan salah satu model karakter yang dilatihkan maka rata-rata hasil pengenalan akan tinggi. Persentase rata rata akurasi hasil pengenalan tiap karakter untuk jenis *font* sama dengan yang dilatihkan mencapai 75%.

Daftar Pustaka

[1] T. Sutojo, E. Mulyanto, V. Suhartono, O. D. Nurhayati and Wijanarto, Teori Pengolahan Citra Digital, Yogyakarta: Andi, 2009.

[2] A. R. Widiarti, "SEGMENTASI CITRA DOKUMEN TEKS SASTRA JAWA MODERN MEMPERGUNAKAN PROFIL PROYEKSI," *SIGMA, Vol. 10, No. 2, Juli 2007,* pp. 167-176, 2007.

[3] R. Mafrur, M. Andestoni, M. S. Ahdi, N. S. Fajri and A. Muhantini, "PENGENALAN HURUF JAWA MENGGUNAKAN METODE LEARNING VECTOR QUANTIZATION (LVQ)," *Jurnal Informatika Teknik Informatika UIN Sunan Kalijaga Yogyakarta,* 2012.

[4] N. Nurmila, A. Sugiharto and E. A. Sarwoko, "ALGORITMA BACK PROPAGATION NEURAL NETWORK UNTUK PENGENALAN POLA KARAKTER HURUF JAWA," *Jurnal Masyarakat Informatika Vol1 No1*, 2010.

[5] B. Karundeng, K. I. Eng and A. S. Nugroho, "AN EVALUATION OF FEATURE EXTRACTION ALGORITHMS FOR AUTOMATIC LANGUAGE TRANSCRIPTION SYSTEM FOR ANCIENT HANDWRITING JAVANESE MANUSCRIPTS," *Proceeding, International Seminar on Industrial Engineering and Management,* 2012.

[6] A. R. Widiarti and E. Winarko, "Widiarti-Winarko Algorithm for Grouping Syllables Result from the Javanese Literature Document Image Recognition," *Recent Researches in Communications and Computers,* 2012.

[7] D. v. d. Merwe and . A. Engelbrecht, "Data Clustering using Particle Swarm Optimization," *Evolutionary Computation, 2003. CEC '03,* 2003.

[8] C.-Y. Cheo and F. Ye, "Particle Swarm Optimization Algorithm and Its Application to Clustering Analysis," *Networking, Sensing and Control, 2004 IEEE International Conference on,* 2004.

[9] M. G. H. Omran, "Particle Swarm Optimization Methods for Pattern Recognition and Image Processing," *PhD Thesis Doctor in the Faculty of Engineering, Built Environment and Information Technology, University of Pretoria, November 2004 ,* 2004.

[10] M. G. Omran, A. P. Engelbrecht and A. Salman, "Dynamic Clustering using Particle Swarm Optimization with Application in Unsupervised Image Classification," *Proc. 5th World Enformatika Conf. (ICCI),* 2005.

[11] X. Cui, T. E. Potok and P. Palathingal, "Document Clustering using Particle Swarm Optimization," *Proceedings 2005 IEEE Swarm Intelligence Symposium,* 2005.

[12] A. Abraham, S. Das and S. Roy, "Swarm Intelligence Algorithms for Data Clustering," *SOFT COMPUTING FOR KNOWLEDGE DISCOVERY AND DATA MINING,* 2008.

[13] K. Murugesan and D. Palaniswami, "EFFICIENT COLOUR IMAGE SEGMENTATION USING MULTI-ELITIST- EXPONENTIAL PARTICLE SWARM OPTIMIZATION," *Journal of Theoretical and Applied Information Technology Vol 18 No1,* 2010.

[14] F. M. A. Mohsen, M. M. Hadhoud and K. Amin, "A new Optimization-Based Image Segmentation method By Particle Swarm Optimization," *(IJACSA) International Journal of Advanced Computer Science and Applications, Special Issue on Image Processing and Analysis,* 2011.

[15] J.-M. Yih, Y.-H. Lin and H.-C. Liu, "Clustering Analysis Method based on Fuzzy C-Means Algorithm of PSO and PPSO with Application in Image Data," *Proceedings of The 8th WSEAS International,* 2008.

[16] J.-M. Yih, Y.-H. Lin and H.-C. Liu, "Clustering Analysis Method based on Fuzzy C-Means Algorithm of PSO and PPSO with Application in Real Data," *INTERNATIONAL JOURNAL OF GEOLOGY vol4 no1,* 2007.

[17] F. Ye and C.-Y. Chen, "Alternative KPSO-Clustering Algorithm," *Tamkang Journal of Science and Engineering Vol 6 No 5,* 2005.

[18] C.-J. Lin, J.-G. Wang and C.-Y. Lee, "Pattern recognition using neural-fuzzy networks based on improved particle swam optimization," *Expert Systems with Applications Vol36,* 2009.

[19] B. Santosa and . M. K. Ningrum, "Cat Swarm Optimization for Clustering," *IEEE International Conference of Soft Computing and Pattern Recognition,* 2009.

[20] C.-C. LAI , "A Novel Image Segmentation Approach Based on Particle Swarm Optimization," *IEICE TRANS. FUNDAMENTALS Vol E89 No1,* 2006.

[21] N. O. S. Ba-Karait and S. M. Shamsuddin, "Handwritten Digits Recognition using Particle Swarm Optimization," 2008.

[22] J.-R. Zhang, J. Zhang, T.-M. Lok and M. R. Lyu, "A hybrid particle swarm optimization–back-propagation algorithm for feedforward neural network training," *Applied Mathematics and Computation 185 (2007),* p. 1026–1037, 2007.

[23] S. Lagudu and C. Sarma, "HAND WRITING RECOGNITION USING HYBRID PARTICLE SWARM OPTIMIZATION & BACK PROPAGATION ALGORITHM Volume 2, Issue 1, January 2013," *International Journal of Application or Innovation in Engineering & Management (IJAIEM),* 2013.

[24] L. Mussi, S. Cagnoni and F. Daolio, "GPU Based Road Sign Detection using Particle Swarm Optimization," *Intelligent Systems Design and Applications, ISDA '09. Vol9 No 9,* 2009.

[25] Y. Zhou and Y. Tan, "GPU-based Parallel Particle Swarm Optimization," *IEEE Congress on Evolutionary Computation (CEC 2009),* 2009.

[26] Y. Zhou and Y. Tan, "GPU Based Parallel Multi objective Particle Swarm Optimization," *International Journal of Artificial Intelligence Vol7 NoA11,* 2011.

[27] A. Kristiadi, Pranowo and P. Mudjihartono, "PARALLEL PARTICLE SWARM OPTIMIZATION FOR IMAGE SEGMENTATION," *SDIWC,* 2013.

[28] M. A. C. Liera, J. A. Castro and I. C. Liera, "Parallel particle swarm optimization using GPGPU," *XIV Convención de Ingeniería Eléctrica Universidad Central de Las Villas 14 al 18 de Junio del 2011, Santa Clara, Cuba,* 2011.

[29] M. Setayesh, M. Zhang and M. Johnston, "A Novel Local Thresholding Technique in PSO for Detecting Continuous Edges in Noisy Images," *Image and Vision Computing New Zealand, 2009. IVCNZ '09.,* 2009.

[30] M. P. Wachowiak and A. E. L. Foster, "GPU-Based Asynchronous Global Optimization with Particle Swarm," *Journal of Physics: Conference Series385 High Performance Computing Symposium 2012,* 2012.

[31] R. Farivar, D. Rebolledo, E. Chan and R. Campbell, "A Parallel Implementation of K-Means Clustering on GPUs," *WorldComp 2008,* 2008.

[32] X. Sierra-Canto, F. Madera-Ramirez and V. Uc-Cetina, "Parallel Training of a Back-Propagation Neural Network using CUDA," *Machine Learning and Applications (ICMLA) Vol 9 December,*

2010.

[33] S. Kajan and J. Slačka, "COMPUTING OF NEURAL NETWORK ON GRAPHICS CARD," *Technical Computing Bratislava 2010 18. ročník medzinárodnej konferencie,* 2010.

[34] H. Kharbanda and R. H. Campbell, "Fast Neural Network Training on General Purpose Computers," *IEEE International conference on High performance Computing - HIPC 2011,* 2011.

[35] M. Wojtera, "Accelerating Convolutional Neural Networks using General Purpose Processing on Graphical Processing Unit," *XI International PhD Workshop OWD 2009,* 2009.

[36] N. Krpan and D. Jakobovi´ c, "Parallel Neural Network Training with OpenCL," *MIPRO, 2012 Proceedings of the 35th International Convention,* 2012.

[37] J. Hofmann, "Evolving Neural Networks on GPUs," *GPUs for Genetic and Evolutionary Computation Competition at 2011 Genetic and Evolutionary Computation Conference July 12-16 2011,* 2011.

[38] J. Pendlebury, H. Xiong and R. Walshe , "Artificial Neural Network Simulation on CUDA," *Symposium on Distributed Simulation,* 2012.