PENJEJAKAN WAJAH SECARA WAKTU NYATA MENGGUNAKAN JARINGAN SYARAF TIRUAN FBR

Dewi Agushinta R [1]
Sigit Widiarto [2]
[1]Jurusan Ilmu Komputer/Teknologi Informasi, [2] Jurusan Teknik Informatika
Universitas Gunadarma
Jl. Margonda Raya 100 Pondok Cina, Depok 16424
[1] dewiar@staff.gunadarma.ac.id, [2] indies_boyz@yahoo.com

ABSTRAK

Jaringan syaraf tiruan fungsi basis radial adalah sebuah model dari arsitektur jaringan yang telah diaplikasikan secara luas. Banyak peneliti menggunakan jaringan FBR sebagai metode mereka untuk mengembangkan sistem pengenalan wajah. Keuntungan menggunakan FBR ini adalah perhitungan yang sederhana dan secara keseluruhan sangat tahan, artinya dalam kondisi perubahan apapun baik orientasi, skala, pose maupun pencahayaan, kinerja dari jaringan FBR tidak menurun secara drastis, sehingga dapat digunakan untuk mendeteksi wajah secara waktu nyata.Masalah utama dari penjejakan wajah yang digunakan dalam lingkungan waktu nyata adalah kecepatan proses, ketelitian perhitungan serta ketepatan penjejakan wajah. Sistem penjejak wajah ini digunakan pada objek yang bergerak dengan posisi wajah menghadap ke depan dan ke samping. Penelitian ini menggunakan maksimal 3 orang sebagai objek pendeteksian dari sistem. Metode utama yang digunakan adalah metode jaringan syaraf tiruan fungsi basis radial.

***Kata Kunci**: waktu nyata, jaringan syaraf tiruan, fungsi basis radial, sistem pengenalan wajah, pengolahan citra*

ENDAHULUAN

Dalam beberapa tahun belakangan ini, anyak peneliti tertarik untuk melakukan enelitian dalam sistem pengenalan wajah. Itu karenakan sistem pengenalan wajah manusia erupakan area aktif yang dapat digunakan eh para peneliti dari berbagai latar belakang, aik dari pengolahan citra, pengenalan pola dan si komputer. Bebagai macam teknik dan etode bisa digunakan untuk memproses wajah an proses *penjejakan* wajah, seperti endekatan jaringan syaraf, metode eigenfaces an metode Markov chain. Banyak peneliti erkonsentrasi pada algoritma segmentasi, kstrasi fitur, pengenalan wajah.

Sebuah sistem visi waktu nyata yang dapat elokalisasikan wajah melalui video, erupakan sebuah teknik dalam pengolahan tra khususnya sistem pengenalan wajah. Ada eberapa pendekatan yang dapat dilakukan alam proses pengenalan wajah, di antaranya dalah pendekatan jaringan syaraf tiruan FBR .

Jaringan syaraf tiruan FBR adalah sebuah odel dari arsitektur jaringan yang telah iaplikasikan secara luas. Banyak peneliti enggunakan jaringan FBR sebagai metode mereka untuk mengembangkan sistem pengenalan wajah. Rosenblum dkk. yang mengembangkan sistem pengenalan ekspresi wajah berdasarkan pergerakan dengan menggunakan arsitektur jaringan syaraf tiruan FBR. Kemudian Ranganath dkk. menampilkan sebuah sistem otomasi pengenalan dan penjejakan wajah menggunakan pendekatan FBR. Howell dan Buxton membandingkan jaringan FBR dengan teknik jaringan saraf tiruan lainnya dalam sebuah aplikasi pengenalan wajah yang meliputi identifikasi individu menggunakan informasi dari video resolusi rendah. Keuntungan menggunakan FBR ini adalah perhitungan yang sederhana dan secara keseluruhan sangat tahan, artinya dalam kondisi perubahan apapun baik orientasi, skala, pose maupun pencahayaan, kinerja dari jaringan FBR tidak menurun secara drastis.

LANDASAN TEORI

Jaringan syaraf tiruan FBR memiliki arsitektur jaringan *feedforward*, dimana jaringan ini memiliki tiga buah lapisan yaitu input, *hidden* dan output yang saling terhubung antara lapisan tersebut. Dalam setiap lapisan, jaringan ini

memiliki algoritma tersendiri. Pada lapisan input, terdapat algoritma untuk menentukan vektor input yang digunakan dalam jaringan ini. Hal yang kedua adalah antara lapisan input ke lapisan *hidden*, dimana lapisan ini menentukan dua (2) buah parameter yang sangat penting, yaitu pusat dari *centroid*, dan lebar dari fungsi radial. Simpul *hidden* menghasilkan nilai output tertinggi jika input vektor dekat dengan pusatnya, dan nilai output akan sebaliknya jika jarak antara input dan pusatnya membesar. Ketiga adalah algoritma untuk menentukan bobot antara lapis *hidden* dengan lapis output. Algoritma ini menggunakan perhitungan *pseudo-inverse matrix*.

Sumber data dalam proses pengenalan wajah secara waktu nyata adalah berkas gambar yang bergerak, yang kemudian dipecah menjadi beberapa bingkai atau bagian terkecil dari sebuah citra bergerak. Bingkai ini nantinya akan diproses lebih lanjut dalam proses pelatihan dan penjejakan wajah. Untuk proses pengambilan contoh, diambil 12 wajah pembelajaran untuk tiap-tiap orang. Proses pengambilan contoh wajah ini dilakukan berdasarkan perubahan posisi, perubahan pose wajah dan lainnya dalam setiap bingkai. Setiap bingkai yang diambil menjadi contoh dikurangi ukuran bingkainya dari 240 x 320 menjadi 40 x 32. Untuk setiap n orang dari sejumlah bingkai yang tersedia, diambil masing-masing 12 bingkai; Dari setiap bingkai yang diambil, diambil posisi wajah sebesar 40 x 32. Proses ini dilakukan secara manual. Rata – rata untuk setiap bingkai ditentukan. Sub-*sampling* wajah diambil dengan mengambil vektor kolom kelipatan empat (4). Sub-sampling dimasukkan ke dalam vektor L, dimana vektor L tersebut selanjutnya dimasukkan ke dalam matriks X, dengan X=[X;L].

Dengan mengambil sub-*contoh* kelipatan 4 untuk setiap kolom, untuk setiap 40 x 32, diharapkan kinerja dari sistem ini dapat lebih efisien, cepat dan tepat. Terlalu banyak komponen yang dimasukkan ke dalam sub-*contoh* ini, membuat perhitungan menjadi semakin lama, dan sedikit komponen yang dimasukkan juga dapat mengurangi kinerja dari sistem, karena kurangnya contoh dalam proses pembelajaran dalam FBR cukup mempengaruhi kinerja sistem secara keseluruhan.

Algoritma *k-means clustering* dapat digunakan untuk mencari vektor input. Di dalam algoritma ini banyak digunakan jarak *euclidean* untuk mencari jarak antara vektor input. Pada tahap penentuan *contoh* X dengan ukuran 320 x 24, yang berarti ada 24 contoh wajah dengan tiap contoh mempunyai 320 komponen. Artinya setiap vektor kolom (panjang kolom 320) mewakili satu bagian dari contoh wajah yang akan dilakukan proses pelatihan. Setiap vektor kolom dibagi menjadi N bagian kelas, dan setiap kelas melakukan proses *k-means clustering*. Untuk setiap kelas, vektor kolom tersebut mewakili satu titik dan setiap titik akan diproses di dalam algoritma. Algoritmanya adalah:

1. Mengmbil sembarang titik $\mathbf{c}_k$ dan lebar dari titik adalah σ_k ($\sigma_k = 0$);
2. Mencari titik terdekat $\mathbf{c}_l$ dari kelas yang sama menggunakan rumus jarak *Euclidean*;
3. Menghitung titik tengah dari 2 titik ini. Nilai tengah ini mempunyai lebar, σ dengan rumus $\sigma = (\| \mathbf{c}_k, \mathbf{c}_l \| / 2) + \sigma_k$;
4. Menghitung jarak D dari titik tengah yang baru ke titik terdekat dari kelas yang lain. Jika $D > \lambda\sigma$, maka gabungan dari $\mathbf{c}_k$ dan $\mathbf{c}_l$ diterima dan mulai lagi langkah ke 2. Jika kondisinya tidak memungkinkan, gabungan $\mathbf{c}_k$ dan $\mathbf{c}_l$ ditolak, dan ditemukan dari 2 titik tersebut dan lebarnya, lalu mulai dari langkah 1;
5. Langkah 1 – 5 diulangi sampai semua cluster dari setiap kelas digunakan.

Nilai λ adalah bilangan bulat berkisar $1 \leq \lambda \leq 3$. Nilai λ yang digunakan dalam penelitian ini adalah 2. Pada algoritma k-means clustering tersebut, parameter input didapat dari proses *sub-sampling* pada pemilihan contoh wajah. Contoh – contoh tersebut dimasukkan ke dalam matriks X. Langkah pertama adalah membagi matriks X menjadi N kelas, sesuai banyaknya wajah yang hendak dijejaki. Kemudian untuk setiap kelas dilakukan langkah–langkah tersebut. Langkah pertama adalah mengambil sembarang titik $\mathbf{c}_k$ dan lebar dari titik adalah σ_k ($\sigma_k = 0$). Definisi titik adalah vektor kolom setiap matriks X. Misal titik $\mathbf{c}_k$ adalah vektor pertama dari matriks X tersebut dan lebar dari titik

ersebut adalah σ_k, dan σ_k diatur 0, karena ebelumnya belum ada perhitungan *k-means lustering*.

Kemudian langkah kedua adalah mencari tik terdekat antara c_k dengan c_l dari kelas yang ama menggunakan rumus jarak *Euclidean*. etiap vektor tersebut dicari jaraknya dengan ungsi norm untuk kelas yang sama kemudian icari nilai paling kecil di antara nilai tersebut. etelah itu dilakukan langkah ketiga, yaitu encari nilai tengah dari kedua vektor tersebut. ari titik tengah tersebut, jarak antara titik engah dengan salah satu titik tersebut adalah ilai lebar σ. Kemudian dari titik tengah ini dicari arak paling dekat dengan titik di kelas yang lain. ilai ini adalah nilai D. Agar titik c_k dan c_l dapat igabung, maka nilai D itu harus lebih besar dari σ. Atau dengan kata lain D >2σ, dan setelah igabung kembali ke langkah 2 dengan titik c_k ekarang adalah titik tengah dari c_k dan c_l ebelumnya. Jika kondisi nya ditolak maka embali ke langkah pertama.

Kemudian setelah didapat dua parameter c an σ dari perhitungan *k-means clustering*, aka fungsi aktivasi *Gaussian* di dalam simpul *idden* bisa didapat. Matriks Gauss diatur ilainya menjadi [ ]. Untuk setiap vektor kolom ari matriks X, dan untuk setiap vektor kolom ari matriks pusat C dilakukan perhitungan jarak *uclidean*, dengan parameter input adalah atriks input X dan matriks pusat C. Hasil dari erhitungan jarak dikuadratkan, dan dibagi engan nilai σ Kemudian dilakukan perhitungan ungsi aktivasi *Gaussian* menggunakan fungsi ksponensial. Nilai selanjutnya dimasukkan ke lalam matriks Gauss.

Perkiraan bobot antara lapis *hidden* dan apis output dapat menggunakan perhitungan atriks *pseudo-invers*. Langkah awal adalah encari nilai eigen dan vektor eigen dari matriks auss yang di dapat sebelumnya.

$$A = PaQ^T \qquad (1)$$

imana **A** adalah matriks Gauss, dengan **P** adalah matriks eigenvektor AA^T, **Q** adalah atriks eigenvektor A^TA dan **a** adalah matriks iagonal hasil dari perhitungan eigen dari **Q**. emudian setelah didapat nilai dari perhitungan eigen tersebut, maka selanjutnya adalah mencari nilai **TQD**.

$$TQD=T*Q*diag(phi) \qquad (2)$$

Dengan **T** adalah matriks target yang diatur sebelumnya, **Q** didapat dari perhitungan eigen sebelumnya dan **diag(phi)** adalah nilai diagonal dari **phi** dengan **phi** adalah:

```
[nl,nc]=size(a)
UN=ones(nl,nc)
phi=UN./d                    (3)
```

dengan nilai **size(a)** adalah menunjukkan ukuran dari matriks **a** yang didapat dari perhitungan eigen dan **nl** > **nc**, **UN** adalah matriks yang bernilai satu sebesar ukuran dari **a**, dan **phi** adalah hasil pembagian kanan elemen array antara **UN** dengan **d**.

Bagian kedua dari implementasi jaringan syaraf FBR ini adalah bagian penjejakan wajah. Pada bagian ini nilai atau atribut dari setiap wajah yang didapat pada bagian pelatihan akan digunakan sebagai parameter input dalam penjejakan wajah. Setiap bingkai dari rentetan bingkai tersebut akan dikenakan perhitungan penjejakan secara waktu nyata. Artinya setiap bingkai nanti akan ada proses pengkotakan wajah dengan berjalannya waktu. Semakin cepat dan tepat, maka semakin baik kinerja dari sistem FBR dalam proses penjejakan wajah tersebut.

Setiap wajah yang akan dijejaki memiliki empat buah atribut yang akan digunakan sebagai parameter input dalam proses penjejakan. Atribut itu adalah C, r, TQD, dan P. C dan r didapat dari proses *k-means clustering* dalam proses pelatihan jaringan. Sedangkan vektor TQD dan matriks P, adalah koefisien dari perhitungan dari lapis hidden ke lapis output. Dimana setiap atribut – atribut ini mewakili nilai tiap wajah yang akan dideteksi.

Langkah pertama adalah menentukan banyaknya wajah yang akan dijejaki dalam sistem ini. Dibuat beberapa contoh sistem penjejakan dengan maksimal tiga wajah yang akan dijejaki agar dapat dilihat perbedaan dari ketiga penerapan tersebut. Sebelumya, sistem akan menyeleksi kondisi sebanyak 2^n kondisi,

dengan *n* adalah banyaknya wajah. Jadi jika ada 1 wajah yang akan dideteksi, maka sistem akan menyeleksi 2^1 kondisi. Begitu juga dengan 2 dan 3 wajah. Untuk 2 wajah ada 2^2 kondisi yang harus diseleksi. Setiap wajah yang akan dideteksi memiliki variabel *li* dan *co* sebagai penunjuk atau koordinat dari lokasi wajah yang akan dijejaki. Atribut setiap wajah yang dijejaki adalah C, r, TQD, P, li, dan co. C adalah nilai pusat wajah, merupakan hasil perhitungan dalam *k-means clustering* pada bagian pelatihan. Nilainya berupa matriks (m,n). Atribut r adalah nilai lebar wajah. Atribut r merupakan jarak dari pusat, dan didapat dari perhitungan *k-means clustering*. Nilainya berupa vektor (1,n). Atribut TQD merupakan koefisien dari hasil perhitungan *pseudo-inverse* matriks, nilainya berupa vektor (1,n). Atribut P merupakan koefisien hasil perhitungan *pseudo-inverse* matriks. Nilai P berupa matriks (n,n). Atribut li dan co merupakan koordinat dalam menentukan lokasi dari wajah yang akan dideteksi. Atribut li merupakan baris dan co adalah kolom. Kecuali *li* dan *co*, atribut lainya selalu tetap. Variabel *li* dan *co* ini sewaktu-waktu dapat berubah menjadi 1 jika wajah yang hendak dideteksi tidak terdeteksi.

METODE PENELITIAN

Spesifikasi arsitektur perangkat keras yang digunakan dalam penelitian ini adalah *main board* ECS K7VTA3, *processor* AMD 1.33 MHz, *memory* DDR 256 MB; VGA AGP 16 MB, dan harddisk 40 GB 7200 RPM. Spesifikasi perangkat lunak yang digunakan adalah sistem operasi Microsoft Windows XP Professional, bahasa pemrograman MATLAB versi 7.0.4, perangkat lunak pengolah berkas *video* MS Movie Maker.

Sistem pengenalan wajah dengan menggunakan jaringan FBR memiliki dua bagian proses, yaitu bagian pelatihan jaringan serta bagian penjejakan wajah. Pada bagian yang pertama, proses pelatihan digunakan untuk melatih jaringan agar dapat mengingat, mengenali dan memproses dengan cepat sehingga memudahkan sistem untuk menjejaki wajah. Proses ini meliputi proses *k-means clustering*, proses fungsi aktivasi *Gaussian*, serta proses perhitungan matriks *pseudo-inverse*. Bagian yang kedua dari sistem ini adalah bagian penjejakan wajah. Bagian ini memproses setiap *bingkai* yang akan dijejak wajahnya berdasarkan informasi yang diterima dari proses pelatihan.

Dalam penelitian ini lebih berkonsentrasi pada tahap penjejakan wajah, rekonstruksi sistem dan analisis sistem pengenalan wajah dengan menggunakan jaringan syaraf tiruan FBR secara keseluruhan.

UJI COBA DAN PEMBAHASAN

Uji coba dilakukan dengan mengasumsikan bahwa telah terjadi proses pelatihan jaringan FBR dan hasilnya digunakan pada proses penjejakan. Uji coba mengambil contoh dari berkas video. Program dijalankan sebanyak 10 kali berturut–turut untuk mencari rata–rata kecepatan proses dari setiap penjejakan wajah.

Uji Coba Menjejak Satu Wajah

Proses menjejak satu wajah menggunakan 40 bingkai. Tidak ada tingkat kesulitan yang cukup membuat sistem ini terganggu. Objek selama 40 bingkai tersebut bergerak dari kiri ke arah kanan pembaca. Gambar 1 adalah salah satu hasil *capture* pada saat sistem bekerja. Kecepatan proses dibutuhkan sekitar 14–15 detik. Rerata kecepatan yang dibutuhkan adalah 15.11709 detik, seperti yang ditunjukkan Tabel 1. Tingkat ketepatan dalam mengkotaki wajah adalah 100 % karena untuk setiap bingkai, wajah terkotaki dengan sempurna.

Gambar 1. Salah satu bingkai dari hasil dari penjejakan satu wajah

Tabel 1.
Hasil uji coba penjejakan satu wajah

Running Ke-	∑Bingkai	∑Bingkai Yang Terjejaki	Prosentasi Keberhasilan (%)	Waktu (detik)
1	40	40	100	18.4213
2	40	40	100	14.7760
3	40	40	100	14.8520
4	40	40	100	14.7137
5	40	40	100	14.7221
6	40	40	100	14.7684
7	40	40	100	14.7416
8	40	40	100	14.7243
9	40	40	100	14.7319
10	40	40	100	14.7466
Rerata = ∑ waktu / Banyaknya running program				15.11709

Coba Menjejak Dua Wajah.
Proses menjejak dua wajah menggunakan bingkai. Proses penjejakan dua wajah galami sedikit tingkat kesulitan yang cukup buat sistem ini terganggu, yaitu objek an kotak hitam bergerak di belakang objek an kotak putih selama beberapa bingkai. ma bingkai itu berlangsung, sistem melakukan perhitungan sedikit lebih lama karena objek tadi tertutup oleh objek lainnya. Objek dengan kotak hitam bergerak membelakangi objek dengan kotak putih dari arah kanan ke arah kiri pembaca, sedangkan objek dengan kotak putih diam berdiri di tengah–tengah. Gambar 2 adalah salah satu hasil *capture* pada saat sistem bekerja.

Tabel 2.
Hasil uji coba penjejakan dua wajah

Running Ke-	∑Bingkai	∑Bingkai Yang Terjejaki	Prosentasi Keberhasilan (%)	Waktu (detik)
1	49	49	100	44.3209
2	49	49	100	42.1492
3	49	49	100	42.2570
4	49	49	100	42.4438
5	49	49	100	42.2895
6	49	49	100	42.3085
7	49	49	100	42.2328
8	49	49	100	42.7052
9	49	49	100	42.2953
10	49	49	100	42.2100
Rerata = ∑ waktu / Banyaknya running program				42.52122

ISSN 0216-4728

Gambar 2 Salah satu bingkai dari hasil penjejakan dua wajah

Kecepatan proses dibutuhkan sekitar 42-43 detik. Tabel 2 menunjukkan waktu rata–rata yang dibutuhkan, yaitu 42.52122 detik. Tingkat ketepatan dalam mengkotaki wajah adalah 100% karena untuk setiap bingkai, wajah terkotaki dengan sempurna, walaupun sempat tertutup oleh objek lainya.

Uji Coba Menjejak Tiga Wajah

Proses menjejak tiga wajah menggunakan 41 bingkai. Ada beberapa tingkat kesulitan yang cukup membuat sistem ini terganggu yang dialami dalam proses penjejakan tiga wajah. Tingkat kesulitan muncul karena objek dengan kotak hitam bergerak di belakang objek dengan kotak abu–abu dan kotak putih selama beberapa bingkai.

Tabel 3.
Hasil uji coba penjejakan tiga wajah

Running Ke-	∑Bingkai	∑Bingkai Yang Terjejaki	Prosentasi Keberhasilan (%)	Waktu (detik)
1	41	38	92.7	115.4227
2	41	38	92.7	113.6682
3	41	38	92.7	113.3248
4	41	38	92.7	113.9039
5	41	38	92.7	114.9066
6	41	38	92.7	113.8888
7	41	38	92.7	113.6395
8	41	38	92.7	113.9421
9	41	38	92.7	113.9499
10	41	38	92.7	114.0108
Rerata = ∑ waktu / Banyaknya running program				114.06573

Gambar 3 Salah satu bingkai dari hasil penjejakan tiga wajah

Selama pergerakan itu berlangsung, sistem melakukan perhitungan sedikit lebih lama karena objek tadi tertutup oleh objek lainnya. Ditemukan ada 3 bingkai mengalami kegagalan dalam menjejak wajah, yaitu pada saat objek dengan kotak hitam melintas di belakang objek dengan kotak abu–abu, tidak terkotaki dengan baik. Artinya pada saat 3 bingkai tersebut hanya 2 objek saja yang terkotaki. Untuk pergerakannya sendiri objek dengan kotak hitam bergerak dari kanan ke arah kiri pembaca, sedangkan objek lainnya diam berdiri di tengah–tengah. Gambar 3 menunjukkan salah satu hasil *capture* pada saat sistem bekerja.

Kecepatan proses dibutuhkan sekitar 113–5 detik. Terlihat rerata yang dibutuhkan pada bel 3, yaitu sekitar 114.06573 detik. Tingkat tepatan dalam mengkotaki wajah adalah 92.7 karena pada beberapa bingkai, wajah tidak kotaki dengan sempurna, karena tertutup eh objek lainya

Ada banyak faktor sebenarnya yang empengaruhi proses dari sistem ini bekerja, ng dirangkum menjadi dua faktor, yaitu cepatan proses dan ketepatan dalam engkotaki wajah. Sistem waktu nyata tidak bas dari perhitungan waktu serta ketelitian rhitungan serta ketepatan dalam mengkotaki jah. Artinya, boleh saja sistem ini memproses ngan waktu yang sangat cepat akan tetapi a sistem tidak bekerja secara semestinya, ungkin tingkat ketepatannya berkurang, atau hkan tidak mengkotaki atau menjejak wajah ng seharusnya terjejaki, sehingga bisa katakan sistem tersebut tidak bekerja dengan ik. Dapat dikatakan bahwa dua faktor tersebut ngat mempengaruhi kinerja sistem secara seluruhan.

Ada tiga hal yang harus diperhatikan rhubungan dengan kecepatan proses, yaitu nyaknya wajah, Jumlah wajah jelas sekali rpengaruh dalam menentukan kecepatan oses dari sistem. Penjejakan satu wajah untuk bingkai hanya membutuhkan rata–rata waktu kitar 15.11709 detik. Penjejakan dua wajah ngan 49 bingkai membutuhkan rata–rata aktu 42.52122 detik. Proses penjejakan tiga ajah, dengan bingkai yang relatif sama, yaitu kitar 41 bingkai, membutuhkan waktu rata-ta sekitar 114.06573 detik. Artinya semakin nyak wajah yang hendak dijejak dan dikotaki, aka proses di dalam sistem ini akan semakin nyak. Jika proses ini semakin banyak, maka sa dikatakan proses itu akan memakan waktu ng cukup signifikan.

Arsitektur perangkat keras serta perangkat nak sistem komputer juga bisa mempengaruhi cepatan. Program penejejakan ini dicoba alankan menggunakan spesifikasi sistem mputer jauh di atas spesifikasi yang gunakan. Hasil percobaan menunjukkan rbedaan dan bahkan lebih baik. Pengaruh rangkat lunak dapat dilihat dari Krishna dan in (1997), bahwa menyebutkan bahwa sistem operasi dapat mempengaruhi kecepatan dari sebuah sistem yang bekerja pada lingkungan waktu nyata. Dengan sistem operasi yang dapat menjadwalkan tugas dan manajemen memori dengan baik, maka akan mempengaruhi waktu prosesnya.

Tingkat kesulitan berhubungan dengan faktor ketepatan menjejak wajah. Tingkat kesulitan adalah dimana sistem kurang bekerja dengan baik karena wajah yang hendak dijejaki dalam keadaan melihat ke atas (posisi wajah tidak frontal), wajah tertutup oleh objek lain, dan wajah belum atau tidak muncul.

Ketepatan dalam mengkotaki wajah merupakan hasil dari sistem penjejakan bekerja. Jika sistem dapat mengkotaki wajah dengan baik maka sistem tersebut bekerja dengan tingkat ketepatan yang baik. Sebaliknya jika sistem tidak dapat mengkotaki dengan baik, karena hal–hal dalam tingkat kesulitan yang telah dijelaskan sebelumnya, maka sistem tersebut tidak bekerja dengan semestinya. Penyebab ketidaktepatan di antaranya adalah kurangnya contoh atau contoh dalam proses pelatihan yang mewakili pada saat wajah dalam bingkai tersebut diproses, penentuan koordinat *li* dan *co* yang tidak tepat.

PENUTUP

Kesimpulan

Jaringan syaraf FBR merupakan sebuah metode yang cukup baik untuk digunakan dalam sistem pengenalan wajah. Semakin banyak wajah yang hendak dijejaki maka akan membutuhkan waktu untuk perhitungan lebih banyak dan juga membutuhkan waktu lebih lama. Jaringan syaraf FBR merupakan metode yang sangat dipengaruhi oleh arsitektur jaringannya. Jumlah wajah yang hendak dijejaki harus ditentukan terlenih dahulu, dan objek yang hendak dijejaki akan menjadi lapisan output dari arsitektur jaringan syaraf FBR. Jika

Sistem pengenalan wajah menggunakan metode FBR ini bersifat statis. Untuk satu wajah hanya akan menghasilkan hasil yang baik jika menggunakan sistem untuk menjejak satu wajah. Begitu juga dengan dua dan tiga wajah. Sistem penjejakan wajah ini menjejak satu wajah adalah 40 bingkai per 15.11709 detik dengan persentase keberhasilan adalah 100%.

Sistem yang menjejak dua wajah dengan 49 bingkai membutuhkan waktu rata-rata 42.52122 detik dengan persentase keberhasilan 100% juga. Sistem yang menjejak tiga wajah dengan jumlah bingkai 41 mempunyai waktu rata-rata 114.06573 detik dengan persentase keberhasilan 92.7%.

Saran

Akan lebih baik jika sistem ini bersifat dinamis, artinya berapapun wajah yang hendak dideteksi dan dijejaki, sistem telah menyediakan sebuah penghitung sesuai dengan banyaknya wajah. Dengan membuat tampilan grafis yang baik, tentu akan membuat sistem ini terlihat lebih menawan. Dengan hasil pewaktuan yang telah disebutkan pada bagian kesimpulan, banyak kemungkinan untuk dapat mempercepat dari sistem ini, seperti penggabungan algoritma pengoptimal seperti algoritma genetika ke dalam sistem ini agar sistem ini dapat bekerja lebih optimal dan mungkin lebih cepat.

DAFTAR PUSTAKA

Andry Pujiriyanto. 2004. **Cepat Mahir Matlab**, Ilmu Komputer.Com, http://ilmukomputer.com/ hmgm.geoph.itb.ac.id/~andrypuji.

Anita Desiani dan Muhammad Arhami. 2005. **Pemrograman Matlab**. Andi. Yogyakarta.

Burdsall, B.; Giraud-Carrier, C.. 1997b. **GA-FBR: A Self-optimising FBR Network.** Third International Conference on Artificial Neural Networks and Genetic Algorithms. Norwich UK. 346-349.

Benyamin Kusumoputro. 2004. **Pengembangan Hemisphere Structure Of Hidden Lapisan Neural Networks Dan Optimasi Strukturnya menggunakan Algoritma Genetika.** Prosiding Semiloka Teknologi Simulasi dan Komputasi Serta Aplikasi. Bandung.

Goldberg, Jack. L. 1991. **Metrics Theory And Its Application**, McGraw-Hill International Editions. Singapore.

Hany Ferdinando. 2005. **Inteligence Sistem Course.** Materi Kuliah. Universitas Kristen Petra. Surabaya.

Krishna, C.M and Shin, Kang. G. 1997. **Waktu nyata Sistems**, McGraw-Hill International Editions. Singapore.

Leung, H dkk. 2002. **Detection of Small Objects in Clutter Using a GA-FBR neural Network.** Department of Electrical and Computer Engineering, University of Calgary.

Rinaldi Munir. 2004. **Pengolahan Citra Digital**. Informatika. Bandung.

Rully Soelaiman dkk. 2005. **Pengembangan Sistem Pengenalan Wajah Dengan Metode Pengklasifikasian Hibrid Berbasis Jaringan Fungsi Radial Basis Dan Pohon Keputusan Induktif.** Fakultas Teknologi Informasi, Institut Teknologi Surabaya.

Siang, Jong Jek. 2005. **Jaringan Syaraf Tiruan dan Pemrogramannya Menggunakan Matlab**. Andi. Yogyakarta.

Wang, Lei and Courant, Michele. 2002. **Multiuser Detection Based On The Immune Strategy FBR Network.** Proceedings of the 9th International Conference on Neural Information Processing (ICNIP-02), vol.3. Orchid Country Club, Singapore. 1485-1489.

Wang, Y dkk. January 2004. **Face Recognition Based On Kernel Radial Basis Function Networks.** Asian Conference on Computer Vision. Korea.

Yang, Fan and Paindavoine, M. September 2003. **Implementation of FBR neural Network on embedded sistem : Real time face penjejakan and identity verification.** IEEE Transactions on Neural Networks. Vol. 14. No. 5. 1162-1175.

Zhang, L dkk. April 2002. **GA-FBR Neural Network Based Maximum Power Titik Tracking For Grid-Connected Photovoltaic Sistems.** International Conference on Power Electronics Machines and Drives (CP487). Bath, UK. 18 -23.