PENGENALAN CITRA KARAKTER MANDARIN MENGGUNAKAN METODE MATRIKS KUADRAN PADA MOBILE DEVICE

Samuel Mahatmaputra[1], David[2], Rosmina[3], Dewi Lestari[4]
[1,2,3,4]Jurusan Teknik Informatika, Fakultas Ilmu Komputer,Universitas Bina Nusantara
Jl. KH Syahdan No. 9, Kemanggisan-Palmerah, Jakarta 11480
Telp. (021) 5345830 ext. 2234, Faks. (021) 530 0244
E-mail: samuelmt@binus.edu

ABSTRAK
Artikel ini membahas tentang hasil penelitian berupa prototype aplikasi pengenalan karakter Mandarin pada handphone Android. Fitur penerjemahan mencakup dua arah dari bahasa Inggris ke dalam bahasa Mandarin menggunakan input berupa kata-kata dalam huruf latin; dan penerjemahan dari bahasa Mandarin ke dalam bahasa Inggris dengan input berupa foto karakter maupun stroke karakter Mandarin. Pengenalan karakter Mandarin dengan input foto ini dilakukan dengan menggunakan metode customized matriks kuadran. Paper ini menjelaskan cara kerja customized matriks kuadran yang diusulkan untuk efisiensi proses pengenalan karakter pada platform Android dengan sumber daya terbatas. Hasil dari penelitian ini adalah sistem pengenalan citra dan penjelasan arti dari suatu karakter Mandarin, serta mampu menampilkan karakter Mandarin sesuai dengan input berupa kata dalam bahasa Inggris pada handphone Android.

Kata Kunci: matrix kwadran, karakter mandarin, mobile, Android

1. PENDAHULUAN

Di era globalisasi ini, bahasa sudah berperan penting bagi pertukaran informasi dan budaya antar negara. Oleh karena itu, banyak orang yang berminat dalam mempelajari berbagai bahasa, misalnya bahasa Inggris, bahasa Mandarin, dan sebagainya.

Belajar bahasa Mandarin dapat dilakukan dengan mengikuti kursus ataupun belajar autodidak, misalnya dengan menonton film Mandarin, menggunakan internet, kamus, dan situs-situs penerjemah. Cara belajar seperti kamus dan situs penerjemah kurang praktis, kurang efisien, dan belum cukup membantu pengguna dalam mempelajari bahasa Mandarin. Hal ini dikarenakan bila pengguna menemukan karakter Mandarin dalam bentuk kertas (*hardcopy*), pengguna tidak dapat menggunakan situs penerjemah, karena untuk menulis karakter Mandarin, pengguna harus mengenal dan mengetahui pelafalan (*pinyin*) ataupun arti dari karakter tersebut. Pencarian melalui kamus juga kurang efisien, karena kamus Mandarin agak berbeda dengan kamus Inggris yang biasanya disusun secara abjad. Kamus Mandarin, ada yang disusun berdasarkan stroke (goresan), ada pula yang berdasarkan *pinyin* (pelafalan), dan sebagainya. Akibatnya, pengguna yang baru mempelajari bahasa Mandarin maupun pengguna yang belum terbiasa dengan kamus Mandarin akan susah mencari kata pada kamus Mandarin sehingga dapat memakan banyak waktu. Selain itu, untuk mencari arti suatu karakter Mandarin, dapat pula dilakukan dengan menanyakan kepada teman, saudara, ataupun kenalan yang mengerti bahasa Mandarin. Namun, cara tersebut juga tidak praktis, karena ada kemungkinan pengguna tidak memiliki kenalan yang menguasai bahasa Mandarin

Solusi yang diusulkan adalah pengembangan suatu aplikasi kamus Mandarin pada mobilephone yang bersifat portable untuk dapat mengenali karakter Mandarin dengan input berupa huruf maupun foto dan dapat menampilkan karakter Mandarin dari kata dalam bahasa Inggris pada platform Android.

Tujuan dari penelitian ini adalah membangun aplikasi kamus Mandarin pada platform Android dengan menerapkan konsep matriks kuadran. Konsep ini akan diterapkan pada pencarian karakter Mandarin dengan input foto, dimana dalam proses pengenalan karakter Mandarin pada foto tersebut, foto akan dibagi menjadi 4 (empat) kuadran. Selain itu, diharapkan penelitian ini nantinya dapat menjadi bahan referensi dalam proses pengenalan citra (foto).

2. TINJAUAN PUSTAKA

Metode matriks kuadran merupakan metode yang digunakan dalam pengenalan citra pada aplikasi ini. Metode ini mencakup proses *thresholding*, *noise removal*, *thinning*, *cropping*, pendeteksian titik sudut, pencarian *stroke*, dan *recognition* (pengenalan) karakter. Metode ini dinamakan **matriks kuadran** dikarenakan adanya pengecekan per piksel seperti pada proses *thresholding*, *noise removal*, dan sebagainya. Selain itu, pada metode ini, juga dilakukan proses pembagian stroke menjadi 4 (empat) kuadran (pada proses pencarian stroke) dan pembagian karakter menjadi 4 (empat) kuadran (pada proses *recognition* karakter).

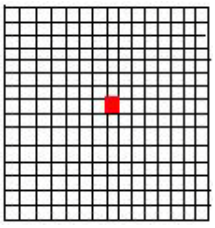
Pada aplikasi ini, *thresholding* yang digunakan adalah *adaptive threshold* (*dynamic threshold*). Sementara untuk proses *thinning*, digunakan algoritma *Zhang-Suen*.

ISSN: 1907-5022

3. PEMBAHASAN

Pada sistem pengenalan citra ini, foto (citra) karakter Mandarin akan melalui proses *resize* terlebih dahulu. Citra akan di-*resize* menjadi ukuran 160 x 200. Lalu, citra akan melalui proses *thresholding*. Proses *thresholding* ini bertujuan untuk mengubah gambar *gray-scale* ataupun warna-warni menjadi gambar biner (hitam putih) berdasarkan nilai *threshold* (patokan) tertentu. Proses *threshold* yang digunakan pada penelitian ini adalah *adaptive threshold*, dimana *adaptive threshold* akan mengubah nilai patokan tergantung dari daerah yang sedang dicek. Adapun cara kerja *adaptive threshold* sebagai berikut :

1. Menentukan *neighborhood size* (daerah di sekeliling titik yang ingin dicek apakah titik tersebut ingin dijadikan putih atau hitam). *Neighborhood size* yang digunakan di penelitian ini adalah 15x15 karena berdasarkan hasil percobaan, 15x15 menghasilkan *image* yang lebih sesuai dengan harapan kita.
2. Menentukan konstanta, yang didapat dari ukuran lebar citra dikalikan tinggi citra dibagi 2. Konstanta ini akan dipakai untuk membantu fungsi *thresholding*.
3. Menghitung *mean* (rata-rata) warna berdasarkan *neighborhood size*.
   a. Tentukan warna RGB dari masing-masing pixel yang ada digambar tersebut :
      R = (warna pixel & 0x00ff0000) >> 16
      G = (warna pixel & 0x0000ff00) >> 8
      B = warna pixel & 0x000000ff
      RGB = (R + G + B) / 3
   b. Titik merah menunjukkan titik yang sedang dicek, apakah titik tersebut akan berubah jadi hitam (0xff000000) atau putih (0xffffffff). Daerah putih (*neighborhood*) menunjukkan daerah sekitar titik yang sedang dicek.



Gambar 1. Contoh Kasus *Thresholding*

   c. Hitung *mean* sesuai dengan *neighborhood size*. Rumusnya adalah total nilai dari semua warna RGB yang ada pada piksel *neighborhood* dibagi dengan banyaknya piksel *neighborhood* dikurangin dengan konstanta yang telah ditentukan pada langkah 2.
4. Setelah *mean* didapat, titik yang sedang dicek tersebut akan dibandingkan dengan nilai *mean*.
   a. Jika titik tersebut memiliki warna RGB lebih besar dari *mean* maka titik tersebut akan diubah menjadi titik putih (0xffffffff).
   b. Jika titik tersebut memiliki warna RGB lebih kecil sama dengan *mean* maka titik tersebut akan diubah menjadi titik hitam(0xff000000).

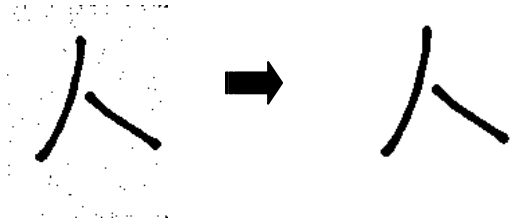
Langkah 3 dan 4 dilakukan berulang-ulang sampai semua piksel yang ada pada citra tersebut telah dicek dan sudah berubah menjadi hitam putih.

Setelah citra menjadi hitam putih, dilakukan proses *noise removal*. Proses ini ditujukan untuk menghilangkan *noise* (derau/titik hitam yang bukan merupakan bagian dari objek yang ingin dikenali). Pada proses *noise removal* ini, Gaussian blur tidak diterapkan, karena Gaussian blur akan membuat citra menjadi semakin kabur, tetapi tidak menghilangkan titik-titik hitam yang bukan bagian dari objek. Adapun langkah-langkah *noise removal* sebagai berikut :

1. Menentukan luas wilayah pengecekan. Luas wilayah yang dipilih adalah ukuran 6x6 karena berdasarkan hasil percobaan, ukuran ini dianggap yang paling efektif dalam menghilangkan titik-titik hitam yang bukan merupakan bagian objek.
2. Hitung total piksel hitam pada wilayah pengecekan.
3. Hitung total piksel pada wilayah pengecekan.
4. Jika total piksel dibagi total piksel hitam lebih besar dari 1 maka :
   a. Jika piksel yang sedang dicek adalah piksel hitam maka piksel tersebut akan berubah menjadi piksel putih.
   b. Jika piksel yang sedang dicek adalah piksel putih maka piksel tersebut akan berubah menjadi piksel hitam.

Proses pengecekan ini dilakukan pada setiap piksel yang ada pada gambar.

Adapun contoh hasil penerapan *noise removal* sebagai berikut :



Gambar 2. Hasil *Threshold dan h*asil *noise removal*

Setelah itu, citra akan melalui proses *thinning*. *Thinning* adalah suatu proses untuk menghapus piksel yang tidak diinginkan dan mengubah garis-garis pada gambar menjadi lebih sederhana dan mempertahankan bentuk objeknya. Tujuan dilakukannya *thinning* yaitu untuk menghilangkan piksel tertentu pada objek sehingga ketebalan objek tersebut sebesar satu piksel. Untuk *thinning*, digunakan algoritma Zhang-Suen, yaitu sebagai berikut :

 ISSN: 1907-5022

1. Tandai semua piksel *foreground* (angka 1) yang memenuhi kondisi 1 sampai 4.
2. Ubah piksel yang sudah ditandai menjadi 0 (*background*). Langkah 1 dan 2 ini untuk selanjutnya disebut sebagai langkah A.
3. Tandai semua piksel *foreground* (angka 1) yang memenuhi kondisi 5 sampai 8.
4. Ubah piksel yang sudah ditandai menjadi 0 (*background*). Langkah 3 dan 4 ini untuk selanjutnya disebut sebagai langkah B.

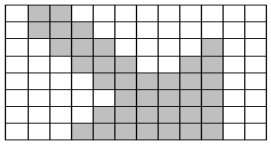
   Langkah A sampai dengan B ini dilakukan berulang hingga tidak ada perubahan yang terjadi pada gambar. Adapun kondisi yang digunakan adalah sebagai berikut :

```
1. 2 <= N(P1) <= 6
2. S(P1) = 1
3. P2 * P4 * P6 = 0
4. P4 * P6 * P8 = 0
5. 2 <= N(P1) <= 6
6. S(P1) = 1
7. P2 * P4 * P8 = 0
8. P2 * P6 * P8 = 0
```

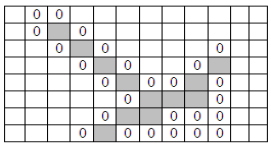
Gambar 3. Algoritma susunan kondisi

Keterangan :
- N(P1) adalah jumlah tetangga yang dimiliki oleh piksel P1 yang tidak bernilai 0 (nol).
- S(P1) adalah jumlah perpindahan nilai dari 0 (nol) ke 1 (satu) mulai dari P2, P3 sampai dengan P9 secara berurut.
- P2 * P4 * P6 = 0, memiliki arti P2 atau P4 atau P6 bernilai 0 (nol).
- P4 * P6 * P8 = 0, memiliki arti P4 atau P6 atau P8 bernilai 0 (nol).

Contoh gambar sebelum dan sesudah *thinning* :



Gambar 4. Contoh Kasus Sebelum *Thinning*



Gambar 5. Contoh Kasus Sesudah *Thinning*

Setelah proses *thinning*, dilakukan proses *cropping*. *Cropping* merupakan proses pemotongan yang dilakukan pada citra untuk menghilangkan *space* kosong pada citra sehingga citra hanya berisi karakter Mandarin. Adapun langkah-langkah untuk proses *cropping* :

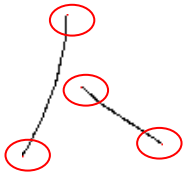
1. Menentukan titik x1 dan titik y1
   a) Pengecekan titik hitam dimulai dari ujung kiri atas citra sampai ujung kanan bawah citra secara berulang.
   b) Mencari posisi x dan y dari titik hitam yang pertama kali ditemukan.
   c) Jika sudah ditemukan titik hitam yang pertama, perulangan akan langsung dihentikan.
2. Menentukan titik x2,y2
   a) Pengecekan titik hitam dimulai dari ujung kanan bawah citra sampai ujung kiri atas citra secara berulang.
   b) Mencari posisi x dan y dari titik hitam yang pertama kali ditemukan.
   c) Jika sudah ditemukan titik hitam yang pertama, perulangan akan langsung dihentikan.
3. Menghitung lebar dan tinggi citra hasil *cropping*
   a) Lebar citra : x2-x1 +1
   b) Tinggi citra : y2-y1+1
4. Membuat citra baru berdasarkan (x1, y1), (x2, y2), lebar dan tinggi yang ditemukan

Setelah itu, dilakukan proses pendeteksian *edge*, Pada penelitian ini, Canny *edge detection* tidak digunakan dikarenakan yang diperlukan dalam pembangunan aplikasi ini adalah memecah satu karakter menjadi *stroke-stroke*, sedangkan Canny *edge detection* hanya akan menghasilkan *edges* yang masih satu kesatuan.

Gambar 6. Gambar Asli dan Hasil Canny *Edge Detection*

Gambar 7. Hasil *cropping, stroke* 1, *stroke* 2

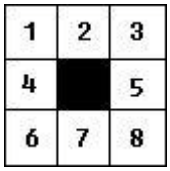
Algoritma yang dipakai terbagi menjadi 2, yaitu proses deteksi titik-titik sudut dan proses menemukan stroke maupun jenis stroke. Contoh hasil deteksi titik sudut:



Gambar 8. Hasil *Edge Detection* (lingkaran merah menunjukkan letak *edge*)

Proses kerja deteksi titik sudut yang dipakai sebagai berikut :

1. Pengecekan dilakukan terhadap setiap pixel yang terdapat di dalam citra tersebut.
2. Setiap piksel dari citra tersebut dilakukan pengecekan 8 arah.



Gambar 9. Gambar Piksel dari Citra 1

Titik hitam merupakan piksel yang sedang dicek, 1-8 adalah piksel-piksel disekeliling piksel yang sedang dicek.

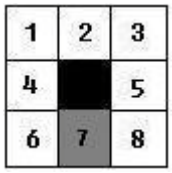
3. Jika 8 arah dari piksel yang sedang dicek, terdapat total 0 atau 1 piksel hitam maka piksel tersebut adalah titik sudut.
4. Jika 8 arah dari piksel yang sedang dicek, terdapat 2 piksel hitam maka piksel tersebut dilakukan pengecekan sebagai berikut :
   a. Jika posisi 2 piksel hitam tersebut terletak pada posisi 2 dan 5 maka piksel hitam yang sedang dicek adalah titik sudut.
   b. Jika posisi 2 piksel hitam tersebut terletak pada posisi 2 dan 4 maka piksel hitam yang sedang dicek adalah titik sudut.
   c. Jika posisi 2 piksel hitam tersebut terletak pada posisi 5 dan 7 maka piksel hitam yang sedang dicek adalah titik sudut.
   d. Selain dari kondisi di atas, piksel hitam yang sedang dicek bukan titik sudut.
5. Jika 8 arah dari piksel yang sedang dicek, terdapat lebih dari 2 piksel hitam maka piksel tersebut bukan titik sudut.

Setiap titik sudut yang ditemukan akan disimpan dalam *Vector* agar titik-titik tersebut dapat digunakan untuk menemukan *stroke-stroke* yang terkandung dalam karakter tersebut.

Setelah titik sudut ditemukan, tahap berikutnya adalah pencarian dan penentuan tipe *stroke*. Proses pencarian *stroke* merupakan proses untuk menggabungkan satu titik sudut ke titik sudut lainnya. Adapun langkah-langkah pencarian *stroke* sebagai berikut :

1. Pengecekan dimulai dari setiap titik sudut ke 8 arah.
2. Jika 8 arah dari titik sudut yang sedang dicek terdapat 1 piksel hitam maka titik sudut tersebut akan menelusuri piksel tersebut sampai ketemu titik sudut yang lain.

   Contoh :



Gambar 10. Piksel dari Citra 2

Pada contoh di atas, titik sudut menemukan terdapat piksel hitam pada posisi 7 maka titik pengecekan selanjutnya dilakukan pada piksel posisi 7 tersebut. Pengecekan akan dilakukan terus menerus sampai menemukan titik sudut yang lain.

3. Jika 8 arah dari piksel yang sedang dicek terdapat 2 piksel hitam (bercabang 2) maka dilakukan pengecekan lebih lanjut, piksel mana yang akan dipilih.

   Kasus 1 :

Gambar 11. Gambar Kasus 1

Kotak hitam menunjukkan posisi piksel hitam sekarang. Kotak merah menunjukkan posisi sebelum sampai ke piksel hitam. Kotak abu-abu pada kotak 5, dan 7 menunjukkan piksel-piksel hitam yang kemungkinan akan dilalui selanjutnya. Untuk menentukan piksel mana yang harus dipilih, dilakukan pengecekan piksel-piksel sebelumnya lebih ke arah mana.

   a. Jika piksel-piksel sebelumnya lebih ke arah bawah (posisi 7) maka piksel hitam yang dipilih selanjutnya adalah piksel hitam posisi 7.
   b. Jika piksel-piksel sebelumnya lebih ke arah kanan (posisi 5) maka piksel yang selanjutnya dipilih adalah piksel hitam posisi 5.

Posisi piksel hitam sekarang (kotak hitam) akan berubah menjadi titik sudut yang baru dan ditambahkan ke daftar titik sudut.

Kasus 2 :

Gambar 12. Gambar Kasus 2

Kondisi masih sama dengan kondisi kasus 1. Untuk menentukan piksel mana yang harus dipilih, dilakukan pengecekan piksel-piksel sebelumnya lebih ke arah mana.

   a. Jika piksel-piksel sebelumnya lebih ke arah kiri (posisi 4) maka piksel hitam yang dipilih selanjutnya adalah piksel hitam posisi 4.
   b. Jika piksel-piksel sebelumnya lebih ke arah kanan (posisi 5) maka piksel hitam yang dipilih selanjutnya adalah piksel hitam posisi 5.
   c. Jika piksel-piksel sebelumnya lebih ke arah bawah (posisi 7) maka proses pengecekan dihentikan dan titik-titik yang ditemukan sebelumnya akan menjadi 1 stroke.

4. Jika 8 arah dari piksel yang sedang dicek terdapat 3 piksel hitam (bercabang 3) maka

dilakukan pengecekan lebih lanjut lagi, piksel mana yang akan dipilih.
Contoh :

Gambar 13. Contoh Piksel pada validasi

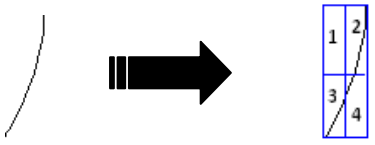
Kotak hitam menunjukkan posisi piksel hitam sekarang.
Kotak merah menunjukkan posisi sebelum sampai ke piksel hitam.
Kotak abu-abu pada kotak 4, 5, dan 7 menunjukkan piksel-piksel hitam yang kemungkinan akan dilalui selanjutnya.
Untuk menentukan piksel mana yang harus dipilih, dilakukan pengecekan piksel-piksel sebelumnya lebih banyak ke arah mana.

a) Jika piksel-piksel sebelumnya lebih ke arah bawah (posisi 7) maka piksel hitam yang dipilih selanjutnya adalah piksel hitam posisi 7.
b) Jika piksel-piksel sebelumnya lebih ke arah kanan (posisi 5) maka piksel hitam yang dipilih selanjutnya adalah piksel hitam posisi 5.
c) Jika piksel-piksel sebelumnya lebih ke arah kiri (posisi 4) maka piksel hitam yang dipilih selanjutnya adalah piksel hitam posisi 4.

5. Titik-titik sudut yang sudah dicek dan titik-titik yang telah membentuk 1 *stroke*, tidak akan dicek lagi.
6. *Stroke* yang ditemukan akan disimpan ke dalam *Vector* agar stroke tersebut bisa diidentifikasi pada proses berikutnya.

Selanjutnya yang dilakukan adalah proses pengidentifikasian jenis dari *stroke* yang ditemukan. Adapun proses kerja dari penentuan *stroke* sebagai berikut :

1. *Stroke* yang ditemukan dibagi menjadi 4 kuadran.
   Contoh :



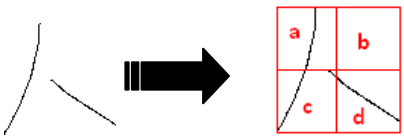
Gambar 14. Contoh pembagian kuadran

2. Hitung maksimal piksel hitam secara vertical, maksimal piksel hitam secara horizontal, total piksel hitam pada masing-masing kuadran (1, 2, 3, 4).
3. Menghitung lebar dan tinggi *stroke* dengan cara :
   a. Mencari x minimal dan y minimal dengan menelusuri gambar stroke dari ujung kiri atas sampai ujung kanan bawah. Titik hitam yang pertama kali ditemukan pada saat penelusuran adalah x minimal dan y minimal.
   b. Mencari x maksimal dan y maksimal dengan menelusuri gambar stroke dari ujung kanan bawah sampai ujung kiri atas. Titik hitam yang pertama kali ditemukan pada saat penelusuran adalah x maksimal dan y maksimal
   c. Lebar = x maksimal – x minimal
   d. Tinggi = y maksimal – y minimal
4. Mengecek apakah *stroke* tersebut merupakan stroke *vertical* dengan cara :
   a. Menentukan batas tinggi yang ditoleransikan bagi *stroke vertical*.
      Batas lebar yang ditoleransi *stroke vertical* adalah jika lebar *stroke* tersebut antara posisi sekarang + (tinggi / 5) dan posisi sekarang – (tinggi / 5).
      Rumus ini digunakan dengan kondisi lebar *stroke* tersebut harus lebih besar dari 1/5 * tinggi *stroke*.
      Jika kondisi di atas tidak terpenuhi maka *stroke* tersebut bukan *stroke vertical*.
   b. Jika lebar dari *stroke* tersebut masih dalam batas toleransi dan tingginya lebih besar dari 3 maka *stroke* tersebut adalah *stroke vertical*.
5. Mengecek apakah *stroke* tersebut merupakan *stroke* horizontal dengan cara :
   a. Menentukan batas tinggi yang ditoleransikan bagi *stroke* horizontal.
      Batas tinggi yang ditoleransi *stroke* horizontal adalah jika tinggi *stroke* tersebut antara posisi sekarang + (lebar / 5) dan posisi sekarang – (lebar / 5).
      Rumus ini digunakan dengan kondisi tinggi *stroke* tersebut harus lebih besar dari 1/5 * lebar *stroke*.
      Jika kondisi di atas tidak terpenuhi maka *stroke* tersebut bukan *stroke* horizontal.
   b. Jika tinggi dari *stroke* tersebut masih dalam batas toleransi dan lebarnya lebih besar dari 3 maka *stroke* tersebut adalah *stroke* horizontal.
6. Jika *stroke* yang sedang dicek bukan *stroke vertical* dan bukan *stroke* horizontal maka dilakukan pengecekan lebih lanjut sebagai berikut :
   a. Mengubah ukuran *stroke* yang sedang dicek menjadi ukuran 20x20.
      Proses ini harus dilakukan agar ukuran *stroke* yang sedang dicek sama dengan ukuran *stroke* yang sudah diproses dan disimpan sebagai data pada tabel MsStrokes.
   b. Melakukan proses thinning pada *stroke* yang merupakan hasil dari langkah 6a.

 ISSN: 1907-5022

c. Hitung maksimal piksel hitam secara *vertical*, maksimal piksel hitam secara horizontal, total piksel hitam pada masing-masing kuadran (1, 2, 3, 4).
d. Melakukan proses perbandingan antara data *stroke* yang sedang dicek dengan data *stroke* yang terdapat pada tabel MsStrokes. Suatu *stroke* dianggap sama dengan *stroke* yang dicek jika total piksel hitam pada masing-masing kuadran 1, 2, 3, 4 saling mendekati.

7. Setelah jenis *stroke* diketahui, selanjutnya yang dilakukan adalah mengetahui posisi *stroke* tersebut pada karakter yang dicari. Metode yang dipakai adalah dengan membagi karakter yang ingin dikenali menjadi 4 kuadran (a, b, c, d).



Gambar 15. Gambar Hasil Pembagian Kuadran

Cara menentukan letaknya adalah dengan mengetahui titik awal dan titik akhir dari *stroke* tersebut. Gambar *stroke* pada langkah 1 jika dilihat dari gambar karakter di atas maka *stroke* tersebut terletak pada kuadran a dan c.

8. *Stroke* yang sudah dikenali disimpan untuk proses pengenalan karakter (*image comparing*).

Setelah penentuan stroke, langkah berikutnya adalah perbandingan gambar (*image comparing*). Tujuan dari *image comparing* ini adalah menentukan persentase kemiripan suatu karakter dengan karakter yang dikenali. Cara kerja perhitungan persentase kemiripan suatu karakter sebagai berikut :

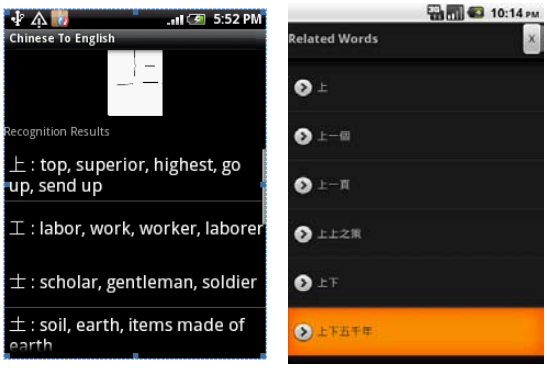
1. Membandingkan total *stroke* pada karakter yang ingin dikenali dengan karakter pada database.
   a. Jika total *stroke* pada karakter database lebih kecil dari total *stroke* pada karakter yang ingin dikenali – 1 maka karakter tersebut tidak dihitung persentasenya.
   b. Jika total *stroke* pada karakter database lebih besar dari total *stroke* pada karakter yang ingin dikenali + 1 maka karakter tersebut tidak dihitung persentasenya.
   c. Selain kondisi di atas, karakter pada database tersebut akan dihitung persentasenya.
2. Hitung persentase kemiripan dengan cara sebagai berikut :
   a. Inisialisasi total kemiripan dengan 0.
   b. Jika jenis *stroke* sama dan letak *stroke* pada karakter sama maka total kemiripan + 1
   c. Jika jenis *stroke* sama tetapi letak *stroke* pada karakter cuma satu yang sama maka total kemiripan + 0.5.
   d. Jika jenis *stroke* sama dan letak *stroke* pada karakter beda sama sekali maka total kemiripan + 0.3.
   e. Pengecekan dilakukan pada setiap *stroke* yang terdapat pada karakter yang ingin dikenali.
   f. Hitung persentase *stroke* yang sama dengan rumus :
      a) Jika total *stroke* karakter yang ingin dikenali lebih besar dari total *stroke* karakter pada *database* maka :
      Persentase *Stroke* = total kemiripan * 100 / total *stroke* karakter yang ingin dikenali
      b) Jika total *stroke* karakter yang ingin dikenali lebih kecil sama dengan total *stroke* karakter pada *database* maka :
      Persentase *Stroke* = total kemiripan * 100 / total *stroke* karakter pada *database*
   g. Hitung persentase total *stroke* yang sama dengan rumus :
      Persentase Total *Stroke* = 100 – *absolute*(totalCase - totalData) * 10
      totalCase = total *stroke* pada karakter yang ingin dikenali
      totalData = total *stroke* pada karakter yang ada di *database*
   h. Hitung total persentase kemiripan karakter yang dikenali dengan karakter yang ada di tabel MsChineseCharacters :
      Persentase Kemiripan = Persentase *stroke* * 0.8 + Persentase Total * 0.2
3. Hasil perhitungan pada langkah 2 (dua) akan disimpan pada *Vector* jika hasil tersebut masih dalam urutan 10 (sepuluh) persentase terbesar.

4. EVALUASI

Aplikasi pengenalan ini dibangun dengan bahasa Java pada Android. Oleh karena itu, untuk menjalankan aplikasi ini, pengguna membutuhkan handphone Android yang memiliki fitur kamera. Untuk mencari karakter Mandarin menggunakan input foto, pengguna cukup memilih menu *Chinese-English* dengan input *image*, dan aplikasi akan memberikan kesempatan bagi pengguna untuk mengambil foto karakter Mandarin. Lalu, hasil foto tersebut akan melalui proses pengenalan citra dan aplikasi akan menampilkan karakter yang berada dalam 10 (sepuluh) persentase kemiripan terbesar. Jika salah satu hasil pengenalan karakter di-*click*, aplikasi akan menampilkan kata-kata yang mengandung karakter hasil pengenalan tersebut. Selain itu, aplikasi ini juga mendukung pencarian kata Mandarin dengan input huruf / karakter Mandarin dan pencarian dengan input berupa kata dalam bahasa Inggris. Hasil yang ditampilkan berupa tulisan tradisional dan simplified, *pinyin* (cara baca), serta definisi dari suatu karakter Mandarin. Selain itu, terdapat fitur penambahan,

ISSN: 1907-5022

pengubahan, dan penghapusan kata. Berikut tampilan pengenalan karakter dengan input foto :



Gambar 16. Tampilan pada *mobile device*

Pada tahap evaluasi dilakukan uji sistem dengan parameter waktu yang dibutuhkan untuk pencarian baik pencarian dengan input kata dalam bahasa Inggris, pencarian dengan input huruf/karakter Mandarin, serta pencarian dengan input foto. Selain itu, evaluasi juga mencakup keberhasilan/ketepatan hasil pengenalan citra.

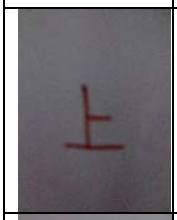
Tabel 1. Tabel evaluasi *Request* penerjemahan *English – Chinese* dan *Chinese-English*

Contoh Kasus	Waktu *Request*	Waktu Hasil Penerjemahan	Lama Proses
clear	25 Januari 2011 14:16:30	25 Januari 2011 14:16:34	4.08 detik
stream	25 Januari 2011 14:18:20	25 Januari 2011 14:18:23	3.18 detik
fancy	25 Januari 2011 14:21:10	25 Januari 2011 14:21:13	3.83 detik
彬(bīn)	25 Januari 2011 14:41:12	25 Januari 2011 14:41:16	4.06 detik
管(guǎn)	25 Januari 2011 14:44:20	25 Januari 2011 14:44:24	4.91 detik
换(huàn)	25 Januari 2011 14:46:12	25 Januari 2011 14:46:12	4.44 detik

Tabel 2. Tabel evaluasi *Request* penerjemahan *Chinese – English* dengan input foto

Contoh Kasus Citra dari Karakter Mandarin yang Difoto	Waktu *Request*	Waktu Hasil Pengenalan Citra	Lama Proses
	26 Januari 2011 13:38:25	26 Januari 2011 13:40:03	1 menit 38 detik
	26 Januari 2011 13:40:55	26 Januari 2011 13:42:45	1 menit 50 detik
	26 Januari 2011 13:59:05	26 Januari 2011 14:01:25	2 menit 20 detik

Tabel 3. Tabel uji coba keberhasilan pengenalan citra

Citra/Foto yang Diuji	Unicode Hasil	Unicode Basis *Data*	Hasil Citra	Status
	4E0A	4E0A	上	true
	4EA4	4EA4	交	true
	4EAE	4EAE	亮	True

Tabel 3. *Lanjutan*

Citra/Foto yang Diuji	Unicode Hasil	Unicode Basis *Data*	Hasil Citra	Status
	4E1D	4E1D	丝	true
	5B89	5B89	安	true

Pada tahap uji coba, terdapat beberapa kasus yang memiliki kemungkinan gagal, seperti pada table berikut:

Tabel 4. Tabel pengenalan citra yang gagal

Citra/Foto yang Diuji	Penyebab Gagalnya Pengenalan Citra
	Terlalu banyak *stroke* yang saling berpotongan sehingga membuat pengenalan *stroke* menjadi tidak akurat. Selain itu, ketebalan tulisan untuk setiap *stroke* juga berbeda sehingga proses *thinning* tidak berjalan dengan baik. Garis horizontal dengan ketebalan yg berbeda, pada saat dilakukan proses thinning, terdapat kemungkinan hasil *stroke* yang dihasilkan bukan merupakan garis horizontal.
	Karakter tidak terlalu jelas, tulisan terlalu tipis, dan citra hasil foto terlalu terang, sehingga proses pengolahan citra menjadi jelek dan pengenalan karakter tidak sesuai.
	karekater tidak sesuai dengan data yang ada. Garis vertikal yang seharusnya lurus, ditulis agak miring sehingga dikenali sebagai *stroke* lain, bukan sebagai *stroke* vertikal sehingga hasil pencarian *stroke* tidak sesuai dengan karakter aslinya.

5. KESIMPULAN

Berdasarkan hasil evaluasi, aplikasi ini memerlukan kondisi yang harus dipenuhi untuk mendapatkan hasil yang optimal :

a. Foto
   1. Warna tulisan harus lebih menonjol dibandingkan dengan warna latar belakang (*background*).
   2. Tidak ada tulisan lain di sekitar karakter Mandarin.
   3. Karakter Mandarin tidak boleh terlalu tipis. Dianjurkan untuk menggunakan spidol. Toleransi ketebalan tulisan yang diperbolehkan sekitar 2.7 mm (milimeter).

b. Tulisan
   1. Ketebalan tulisan untuk 1(satu) *stroke* tidak boleh berbeda.
   2. Kemiringan tulisan yang dapat ditoleransi adalah sebesar 10% dari panjang *stroke*.
   3. Panjang minimal untuk 1 (satu) buah *stroke* adalah 5 mm (milimeter).

Aplikasi ini mampu mengenali karakter Mandarin dari sebuah citra tanpa mengharuskan pengguna mengetahui cara penulisan karakter tersebut sehingga ataupun mencari karakter Mandarin melalui kamus. Namun, waktu yang dibutuhkan oleh aplikasi untuk mengenali karakter masih cukup banyak, yaitu sekitar 1 menit.

PUSTAKA

Holt, C. M., Stewart, A., Clint, M., Perrott, R.H. (1987). An improved parallel thinning algorithm, *Communications of the ACM, v.30 n.2*, 156-160.

Leea, H. J., Chena, B. (1992) Recognition of handwritten Chinese characters via short line segments. *Pattern Recognition Volume 25, Issue 5, 543-552:* Elsevier. Diakses pada 9 October 2010 dari http://www.sciencedirect.com/science?_ob=ArticleURL&_udi=B6V14-48MPNW1-1K7&_user=3955742&_coverDate=05%2F31%2F1992&_rdoc=1&_fmt=high&_orig=gateway&_origin=gateway&_sort=d&_docanchor=&view=c&_searchStrId=1691472942&_rerunOrigin=scholar.google&_acct=C000061873&_version=1&_urlVersion=0&_userid=3955742&md5=fed492840b377d881715eb25b31f1f27&searchtype=a

Parker, J. R. (1996). Algorithms for Image Processing and Computer Vision: Wiley

Shi, D., Gunn, S. R., Damper, R. I. (2001) A Radical Approach to Handwritten Chinese Character Recognition Using Active Handwriting Models. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'01) - Volume 1*. Diakses pada 12 October 2010 dari http://www.computer.org/portal/web/csdl/doi/10.1109/CVPR.2001.990539

Zhang, T. Y., Suen, C. Y. (1984). A fast parallel algorithm for thinning digital patterns, *Communications of the ACM, v.27 n.3*, 236-239.