PERBANDINGAN ALGORITMA TEMPLATE MATCHING DAN FEATURE EXTRACTION PADA OPTICAL CHARACTER RECOGNITION

Raden Sofian Bahri [1], Irfan Maliki[2]

[1,2]Program Studi Teknik Informatika
Fakultas Teknik dan Ilmu Komputer Universitas Komputer Indonesia
Jl. Dipati Ukur No. 112-116 Bandung

ABSTRAK

Algoritma template matching merupakan metode yang sederhana dan banyak digunakan untuk mengenali pola. Kelemahan algoritma ini adalah terbatasnya model yang akan dijadikan *template* sebagai pembanding pada basis data seperti bentuk, ukuran, dan orientasi. Algoritma feature extraction menjawab masalah model *template* seperti bentuk, ukuran, dan orientasi yang ada pada algoritma template matching dengan cara memetakan ciri-ciri objek citra yang akan dikenali. Optical character recognition digunakan untuk menerjemahkan karakter pada citra digital menjadi format teks.

Penerapannya yang sederhana membuat algoritma template matching banyak digunakan dalam OCR. Kedua algoritma diuji berdasarkan akurasi hasil pengenalan, waktu yang dibutuhkan, kompleksitas, pengembangan, dan proses yang dibutuhkan oleh masing-masing algoritma dalam OCR. Berdasarkan akurasi, pengembangan, dan waktu, algoritma feature extraction lebih unggul dibandingkan algoritma template matching pada OCR.

Kata kunci : *template matching*, *feature extraction*, *dan optical character recognition*.

1. PENDAHULUAN

Algoritma template matching merupakan metode yang sederhana dan banyak digunakan untuk mengenali pola. Algoritma ini bekerja dengan cara mengevaluasi pola citra yang akan dibandingkan dengan pola citra template pada basis data. Kelemahan algoritma ini adalah terbatasnya model yang akan dijadikan template sebagai pembanding pada basis data seperti bentuk, ukuran, dan orientasi[10].

Algoritma feature extraction menjawab masalah model template seperti bentuk, ukuran, dan orientasi yang ada pada algoritma template matching dengan cara memetekan ciri-ciri objek citra yang akan dikenali. Ciri-ciri yang dipetakan dari citra yang akan dikenali dan diklasifikasikan terhadap ciri-ciri citra template yang disimpan pada basis data. Algoritma feature extraction memiliki kelemahan dalam menentukan ciri-ciri khusus citra yang akan dikenali[10].

Optical character recognition digunakan untuk menerjemahkan karakter pada citra digital menjadi format teks. Penerapannya yang sederhana membuat algoritma template matching banyak digunakan dalam OCR[1]. Namun, penggunaan algoritma template matcing terhadap OCR masih memiliki tingkat akurasi pengenalan yang kecil. Hal ini terjadi karena terbatasnya template yang disimpan dalam basis data sedangkan model template, yaitu font terus bertambah. Kelebihan algoritma feature extraction yang tidak terbatas dalam model template diharapkan bisa meningkatkan akurasi pengenalan terhadap OCR.

Untuk mengetahui apakah algoritma feature extraction lebih baik daripada algoritma template matching terhadap OCR, maka dibangunlah aplikasi yang bisa membandingkan kedua metode tersebut.

2. TINJAUAN PUSTAKA

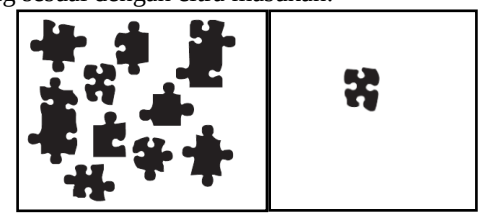
A. Pengenalan Pola

Pengenalan pola dapat dikatakan sebagai kemampuan mengenali objek berdasarkan ciri-ciri dan pengetahuan yang pernah diamatinya dari objek-objek tersebut. Tujuan dari pengenalan pola adalah mengklasifikasi dan mendeskripsikan pola atau objek kompleks melalui pengetahuan sifat-sifat atau ciri-ciri objek tersebut.

Ada tiga pendekatan dalam pengenalan pola yaitu secara sintaks, statistik, dan semantik[7]. Pengenalan pola secara sintaks dilakukan berdasarkan ciri-ciri objek. Pengenalan pola secara statistik dilakukan berdasarkan komputasi matematis. Pendekatan dengan semantik berarti pola dikenali dalam tataran yang lebih abstrak.

B. *Algoritma Template Matching*

Pada dasarnya template matching adalah proses yang sederhana. Suatu citra masukan yang mengandung template tertentu dibandingkan dengan template pada basis data. Template ditempatkan pada pusat bagian citra yang akan dibandingkan dan dihitung seberapa banyak titik yang paling sesuai dengan template. Langkah ini diulangi terhadap keseluruhan citra masukan yang akan dibandingkan. Nilai kesesuaian titik yang paling besar antara citra masukan dan citra template menandakan bahwa template tersebut merupakan citra template yang paling sesuai dengan citra masukan.



Gambar 1. Ilustrasi template matching

Gambar 1 bagian kiri merupakan citra yang mengandung objek yang sama dengan objek pada template yang ada di sebelah kanan. Template diposisikan pada citra yang akan dibandingkan dan dihitung derajat kesesuaian pola pada citra masukan dengan pola pada citra template.

Tingkat kesesuaian antara citra masukan dan citra template bisa dihitung berdasarkan nilai eror terkecil [3] dengan menggunakan persamaan 1.

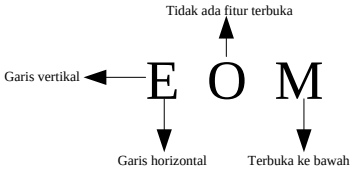
$$\min e = \sum_{(x,y)\epsilon W}\left(1_{x,y} - T_{x,y}\right)^2 \qquad (1)$$

I adalah pola pixel citra masukan yang akan dibandingkan. T adala pola pixel citra template. Template dengan nilai eror paling kecil adalah template yang paling sesuai dengan citra masukan yang akan dibandingkan.

Ukuran objek yang beragam bisa diatasi dengan menggunakan template berbagai ukuran. Namun hal ini membutuhkan tambahan ruang penyimpanan. Penambahan template dengan berbagai ukuran akan membutuhkan komputasi yang besar. Jika suatu template berukuran persegi dengan ukuran m x m dan sesuai dengan citra yang berukuran N x N, dan dimisalkan pixel m2 sesuai dengan semua titik citra, maka komputasi yang harus dilakukan adalah O(N2m2). Komputasi tersebut harus dilakukan dengan template yang tidak beragam. Jika parameter template bertambah, seperti ukuran template yang beragam, maka komputasi yang dilakukan juga akan bertambah. Hal ini yang menyebabkan metode template matching menjadi lamban.

C. *Feature Extraction*

Feature extraction merupakan salah satu cara untuk mengenali suatu objek dengan melihat ciri-ciri khusus yang dimiliki objek tersebut[6]. Tujuan dari *feature extraction* adalah melakukan perhitungan dan perbandingan yang bisa digunakan untuk mengklasifikasikan ciri-ciri yang dimiliki oleh suatu citra.



Gambar 2. Ilustrasi feature extraction

Gambar 2 merupakan ilustrasi citra karakter dengan ciri-cirinya. Ciri-ciri dari masing-masing citra *template* akan di simpan. Citra masukan yang akan dibandingkan akan dianalisis berdasarkan ciri-ciri citra. Ciri-ciri yang dimiliki citra masukan akan diklasifikasikan terhadap ciri-ciri citra *template*

D. *Klasifikasi*

Jika tujuan dari feature extraction adalah memetakan pola berdasarkan ciri-ciri yang dimiliki oleh suatu citra, maka klasifikasi bertujuan untuk mengenali citra dengan cara mengklasifikasikan ciri-ciri yang dimilikinya.

Klasifikasi adalah proses untuk menemukan model atau fungsi yang menjelaskan atau membedakan konsep atau kelas data, dengan tujuan untuk dapat memperkirakan kelas dari suatu objek yang labelnya tidak diketahui [3].

Proses klasifikasi biasanya dibagi menjadi dua fase yaitu fase learning dan fase test. Pada fase learning, sebagian data yang telah diketahui kelas datanya diumpankan untuk membentuk model perkiraan. Kemudian pada fase test model yang sudah terbentuk diuji dengan sebagian data lainnya untuk mengetahui akurasi dari model tersebut. Bila akurasinya mencukupi model ini dapat dipakai untuk prediksi kelas data yang belum diketahui.

Teknik ini dapat memberikan klasifikasi pada data baru dengan memanipulasi data yang ada yang telah diklasifikasi dan dengan menggunakan hasilnya untuk menghitung jarak antara ciri-ciri citra template dan citra masukan.

E. *K-Nearest Neighbour*

k-nearest neighbor (k-NN atau KNN) bekerja dengan cara membandingkan data uji dan data training/template [3]. k-NN mencari pola data template yang paling mendekati dengan data uji. Dekat atau jauhnya tetangga dihitung berdasarkan jarak Euclidean. Dengan kata lain, kedua data dibandingkan berdasarkan atribut-atributnya.

k-NN memiliki beberapa kelebihan yaitu tangguh terhadap data template yang noisy dan efektif apabila jumlah data template besar. Sedangkan kelemahan dari k-NN adalah perlunya menentukan nilai parameter k (jumlah dari tetangga terdekat), pembelajaran berdasarkan jarak tidak jelas mengenai jenis jarak apa yang harus digunakan dan atribut

mana yang harus digunakan untuk mendapatkan hasil yang terbaik, dan biaya komputasi cukup tinggi karena diperlukan perhitungan jarak dari tiap query instance pada keseluruhan training sample. Persamaan 2 adalah persamaan yang digunakan untuk menghitung jarak [3].

$$d(p,q)^2 = d(p,q)^2 = \sum_{i=1}^{n} (q_1 - p_i)^2$$

Nilai k yang terbaik untuk algoritma ini tergantung pada data. Pada umumnya, nilai k yang tinggi akan mengurangi efek noise pada klasifikasi, tetapi membuat batasan antara setiap klasifikasi menjadi lebih kabur.

Langkah-langkah yang digunakan pada metode k-NN adalah sebagai berikut:

1. Tentukan parameter k. k adalah jumlah tetangga terdekat. Nilai k yang diguankan pada penelitian ini adalah 1.
2. Hitung jarak antara ciri-ciri citra template dan citra masukan dengan menggunakan persamaan 2.
3. Urutkan jarak dari kecil ke besar.
4. Gunakan parameter k sebagai acuan dalam mengambil sejumlah data berdasarkan urutan pada nomor 3.

Ketepatan algoritma k-NN ini sangat dipengaruhi oleh fitur-fitur unik setiap citra yang akan dikenali. Riset terhadap algoritma ini sebagian besar membahas bagaimana memilih dan memberi bobot terhadap fitur, agar performa klasifikasi menjadi lebih baik

F. *Kompleksitas Algoritma*

Algoritma yang baik adalah algoritma yang mangkus (efficient) [9]. Kemangkusan algoritma diukur dari waktu (time) eksekusi algoritma dan kebutuhan ruang (space) memori.

Algoritma yang mangkus ialah algoritma yang meminimumkan kebutuhan waktu dan ruang. Kebutuhan waktu dan ruang suatu algoritma bergantung pada ukuran masukan (n), yang menyatakan jumlah data yang diproses. Kemangkusan algoritma dapat digunakan untuk menilai algoritma yang baik dari sejumlah algoritma penyelesaian masalah. Besaran yang dipakai untuk menerangkan model abstrak pengukuran waktu/ruang ini adalah kompleksitas algoritma. Ada dua macam kompleksitas algoritma, yaitu kompleksitas waktu dan kompleksitas ruang.

Kompleksitas waktu, T(n), diukur dari jumlah tahapan komputasi yang dibutuhkan untuk menjalankan algoritma sebagai fungsi dari ukuran masukan n.

Kompleksitas ruang, S(n), diukur dari memori yang digunakan oleh struktur data yang terdapat di dalam algoritma sebagai fungsi dari ukuran masukan n.

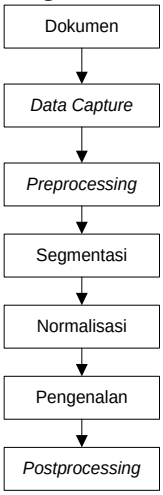
Kompleksitas waktu dihitung berdasarkan jumlah tahapan komputasi operasi yang dilakukan dalam sebuah algoritma. Dalam prakteknya, operasi yang dihitung hanya operasi khas yang mendasari suatu algoritma. Misalnya, operasi khas di dalam algoritma pencarian di dalam larik adalah perbandingan elemen larik. Operasi khas algoritma pengurutan adalah perbandingan dan pertukaran elemen.

G. *Optical Character Recognition*

Optical character recognition (OCR) adalah sebuah sistem komputer yang dapat membaca huruf, baik yang berasal dari sebuah pencetak (printer atau mesin ketik) maupun yang berasal dari tulisan tangan.

OCR adalah aplikasi yang menerjemahkan gambar karakter (image character) menjadi bentuk teks dengan cara menyesuaikan pola karakter per baris dengan pola yang telah tersimpan dalam database aplikasi.

Hasil dari proses OCR adalah berupa teks sesuai dengan gambar output scanner dimana tingkat keakuratan penerjemahan karakter tergantung dari tingkat kejelasan gambar dan metode yang digunakan[5]. Secara umum blok diagram kerja OCR dapat dilihat pada gambar 3.



Gambar 3. Blok Diagram Kerja OCR

3. HASIL DAN PEMBAHASAN

Secara garis besar, sistem dibangun sesuai dengan blok diagram kerja OCR pada gambar 3. Namun, pada tahap pengenalan, dilakukan dua kali pengenalan yaitu pengenalan dengan algoritma *template matching* dan *feature extraction*.

A. *Preprocessing*

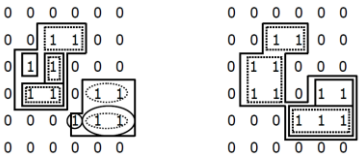
Proses preprocessing terdiri dari proses grayscaling dan binerisasi. Kedua proses ini

dilakukan untuk mengubah intensitas pixel. Proses ini menghasilkan citra dengan warna biner yaitu hitam dan putih.

B. ***Segmentasi***

Proses pensegmentasian terdiri dari segmentasi baris dan segmentasi karakter. Segmentasi karakter dilakukan dengan menggunakan metode Connected component analysis.

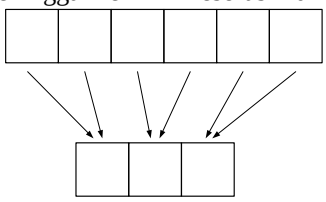
Connected component analysis bekerja dengan cara memeriksa intensitas pixel di sekitar pixel yang sedang dianalisis.



Gambar 4. Ilustrasi Connected Component Analysis

C. ***Normalisasi***

Pada tahap normalisasi, proses yang dilakukan adalah proses penskalaan terhadap citra agar citra memiliki resolusi tetap. Pada penelitian ini, citra diskalakan sehingga memiliki resolusi 10 x 12 pixel.



Gambar 5. Perubahan ukuran pixel

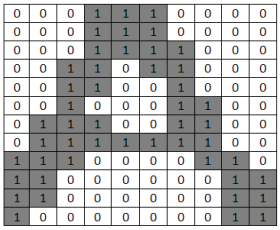
D. ***Proses Pengenalan***

Proses pengenalan dilakukan oleh algoritma *template matching* dan *feature extraction*. Pembahasan mengenai kedua algoritma ini terdapat pada sub bab analisis algoritma.

E. ***Pengenalan dengan Algoritma Template Matching***

Pengenalan pola dengan menggunakan metode *template matching* dilakukan dengan cara membandingkan citra masukan dengan citra *template*. Citra masukan dihitung berdasarkan banyaknya titik yang sesuai dengan citra *template*.

Pixel citra biner ditelusuri mulai dari kiri atas hingga ke kanan bawah. Citra biner dengan *pixel* berwarna hitam akan direpresentasikan dengan nilai 1. Sedangkan *pixel* citra yang berwarna putih akan direpresentasikan dengan nilai 0. Gambar 6 adalah gambar yang mengilustrasikan angka 1 dan 0 yang mewakili nilai *pixel* citra.



Gambar 6. Citra Hitam Putih dengan Nilai *Pixel* 1 dan 0.

Deretan angka biner pada citra masukan akan dihitung dengan deretan angka biner pada citra *template*. *Template* dengan nilai eror terkecil merupakan *template* citra yang paling sesuai dengan citra masukan.

Contoh berikut ini adalah penerapan pengenalan karakter dengan menggunakan metode *template matching* yang menggunakan persamaan 1.

Tabel 1 adalah tabel contoh deretan angka biner citra karakter *template*

Tabel 1. Karakter dan Deretan Angka Pola Pixel

Karakter	Deretan angka biner citra *template*
A	0 0 1 1 1 0 0 0 0
B	1 1 1 0 0 0 1 1 0
C	1 0 1 0 0 0 1 1 1
D	0 1 1 1 0 1 0 0 1

Deretan berikut ini adalah angka biner citra karakter masukan.

0 0 0 1 0 1 0 1 0

Tabel 2 adalah perhitungan eror dengan menggunakan persamaan 1:

Tabel 2. Perhitungan Nilai Eror

Kar	Perhitungan Nilai Eror
A	(0-0)^2 + (0-0)^2 + (0-1)^2 + (1-1)^2 + (0-1)^2 + (1-0)^2 + (0-0)^2 + (1-0)^2 + (0-0)^2 = 4
B	(0-1)^2 + (0-1)^2 + (0-1)^2 + (1-0)^2 + (0-0)^2 + (1-0)^2 + (0-1)^2 + (1-1)^2 + (0-0)^2 = 6
C	(0-1)^2 + (0-0)^2 + (0-1)^2 + (1-0)^2 + (0-0)^2 + (1-0)^2 + (0-1)^2 + (1-1)^2 + (0-1)^2 = 6
D	(0-0)^2 + (0-1)^2 + (0-1)^2 + (1-1)^2 + (0-0)^2 + (1-0)^2 + (0-0)^2 + (1-0)^2 + (0-1)^2 = 5

Jika nilai eror masing-masing *template* sudah diketahui, maka cari nilai eror terkecil. Nilai eror terkecil adalah *template* yang paling sesuai dengan karakter. Nilai terkecil dari hasil perhitungan pada tabel 2 adalah 4. Maka citra *template* yang paling sesuai dengan citra masukan adalah citra karakter A.

F. ***Pengenalan dengan Feature Extraction***

Pemetaan ciri-ciri khusus yang dilakukan terhadap citra karakter adalah keterbukaan citra, jumlah garis vertikal dan horizontal, jumlah perpotongan pixel hitam di bagian tengah citra secara vertikal dan horizontal, rasio citra, dan

histogram pixel hitam pada sembilan bagian citra yang dibandingkan dengan resolusi citra karakter.

Jika seluruh ciri-ciri citra karakter sudah didapatkan, maka ciri-ciri tersebut diklasifikasikan dengan ciri-ciri yang ada pada basis data. Metode klasifikasi yang digunakan pada penelitian ini adalah metode k-NN atau k *Nearest Neighbour*.

Berikut ini adalah contoh perhitungan k-NN untuk mengenali karakter.

1. Tabel 3 adalah tabel citra karakter *template* yang dilengkapi dengan ciri-cirinya.

Tabel 3. Citra Karakter Template dan Atributnya

no	kar	ratio	t_kiri	t_kanan	int_v	garis_v	garis_h	Blok1
1	A	0.727273	0	0	2	0	1	0
2	B	0.727273	0	1	3	1	3	0.068182
3	C	0.818182	0	1	2	0	0	0.030303
4	D	0.818182	0	0	2	1	0	0.060606

2. Tabel 4 adalah ciri-ciri citra karakter masukan yang akan dikenali

Tabel 4. Citra Karakter Masukan dan Atributnya

no	kar	ratio	t_kiri	t_kanan	Int_v	Garis_v	Garis_h	Blok1
1	?	0.818182	0	0	2	0	1	0.010101

3. Tentukan faktor k. Faktor k menyatakan jumlah objek baru hasil pengenalan yang akan diambil. Jika k bernilai 2, maka dua urutan teratas akan diambil. Pada pengenalan karakter ini, k yang digunakan bernilai 1, maka hanya nilai hasil perhitungan terkecil yang akan diambil.
4. Tabel 5 adalah perhitungan k-NN yang dilakukan dengan persamaan 2 dengan menggunakan atribut citra template dan citra masukan

Tabel 5. Perhitngan Jarak Untuk Klasifikasi

No	Kar	Perhitungan	Hasil Perhitungan
1	A	(0.727273-0.818182)^2 + (0-0)^2 + (0-0)^2 + (0-0)^2 + (1-1)^2 + (2-2)^2 + (2-2)^2 + (0-0)^2 + (1-1)^2 + (0-0.010101)^2	0.000102
2	B	(0.727273-0.818182)^2 + (0-0)^2 + (1-0)^2 + (0-0)^2 + (0-1)^2 + (3-2)^2 + (1-2)^2 + (1-0)^2 + (3-1)^2 + (0-0.068182)^2	0.003373
3	C	(0.727273-0.818182)^2 + (0-0)^2 + (1-0)^2 + (0-0)^2 + (0-1)^2 + (2-2)^2 + (1-2)^2 + (0-0)^2 + (0-1)^2 + (0-0.030303)^2	0.000408
4	D	(0.727273-0.818182)^2 + (0-0)^2 + (0-0)^2 + (0-0)^2 + (0-1)^2 + (2-2)^2 + (2-2)^2 + (1-0)^2 + (0-1)^2 + (0-0.060606)^2	0.002551

5. Urutkan hasil perhitungan dari kecil ke besar. Tabel 6 adalah tabel yang berisi hasil perhitungan dengan menggunakan persamaan 2 dan diurutkan secara ascending.

Tabel 6. Hasil Perhitungan Jarak

No	Kar	Hasil Perhitungan
1	A	0.000102
3	C	0.000408
4	D	0.002551
2	B	0.003373

Berdasarkan parameter k, maka hanya ada satu data yang diambil yaitu karakter A.

G. *Analisis Algoritma*

Berdasarkan proses yang dibutuhkan kedua algoritma, algoritma feature extraction membutuhkan proses yang lebih banyak dibandingkan dengan algoritma template matching, yaitu proses pendeteksian keterbukaan citra, pendeteksian perpotongan garis tengah citra, perhitungan jumlah pixel hitam tiap blok, perhitungan jumlah garis vertikal dan horizontal, dan proses klasifikasi. Sedangkan algoritma template matching hanya membutuhkan proses pemetaan intensitas pixel citra karakter yang akan dikenali, perhitungan eror, dan pencarian nilai eror minimum.

Kompleksitas algoritma dihitung berdasarkan operasi dasar yang dilakukan kedua algoritma. Operasi dasar yang dilakukan pada algoritma template matching adalah sebagai berikut:

1. Pemetaan intensitas pixel. Pemetaan ini dilakukan satu kali saat citra karakter akan dikenali.
2. Perhitungan nliai eror minimum. Perhitungan yang dilakukan sebanyak citra template dinotasikan dengan n.
3. Pencarian nilai eror minimum dilakukan pada seluruh citra template. Jumlah pencarian yang dilakukan terhadap seluruh citra template dinotasikan dengan n.
4. Tabel 7 adalah perhitungan jumlah operasi dasar yang dilakukan pada algoritma template matching.

Tabel 7. Perhitungan Jumlah Operasi Dasar Algoritma Template Matching

No.	Operasi dasar	Jumlah iterasi
1.	Pemetaan pixel	1 kali
2.	Perhitungan nilai eror minimum	n kali
3.	Pencarian nliai eror terkecil	n kali
Total		**2n+1**

5. Berdasarkan hasil perhitungan jumlah operasi dasar algoritma template matching pada tabel

3.2, maka kompleksitas algoritma template matching adalah 2n+1.

algoritma *feature extraction* adalah sebagai berikut:

1. Pemetaan fitur citra karakter. Pemetaan ini dilakukan satu kali saat citra karakter akan dikenali.
2. Perhitungan jarak dengan menggunakan persamaan 2-7. Perhitungan yang dilakukan sebanyak citra template dinotasikan dengan n.
3. Pencarian jarak minimum dilakukan pada seluruh citra template. Jumlah pencarian yang dilakukan terhadap seluruh citra template dinotasikan dengan n.
4. Tabel 8 adalah perhitungan jumlah operasi dasar yang dilakukan pada algoritma *feature extraction*.

Tabel 8. Perhitungan Jumlah Operasi Dasar Algoritma Feature Extraction

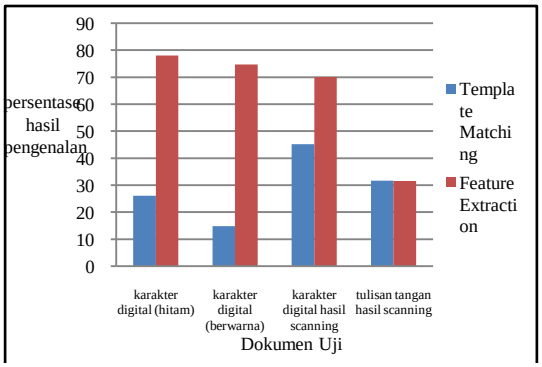
No.	Operasi dasar	Jumlah iterasi
1.	Pemetaan fitur citra karakter	1 kali
2.	Perhitungan jarak	n kali
3.	Pencarian jarak terkecil	n kali
Total		**2n+1**

5. Berdasarkan hasil perhitungan jumlah operasi dasar *feature extraction* pada tabel 3.3, maka kompleksitas algoritma *feature extraction* adalah 2n+1.

Berdasarkan hasil perhitungan kompleksitas kedua algoritma, algoritma *feature extraction* dan algoritma *template matching* memiliki kompleksitas yang sama yaitu 2n+1.

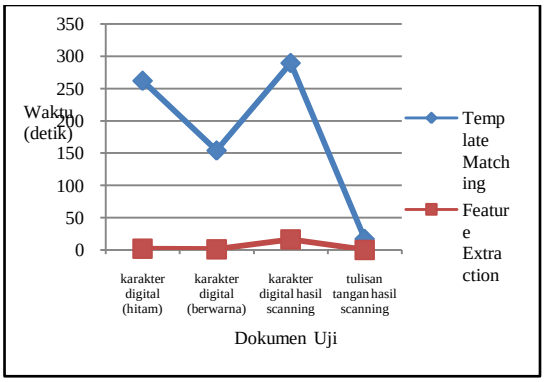
Dari lima proses pemetaan fitur pada algoritma *feature extraction*, masih ada kemungkinan penambahan fitur-fitur khusus citra karakter. Salah satu contohnya adalah fitur jumlah *stroke* (jumlah garis yang membentuk karakter). Sedangkan pada algoritma *template matching*, kecil kemungkinan penambahan fitur karena algoritma *template matching* bekerja berdasarkan pemetaan intensitas pixel.

Berdasarkan hasil pengenalan, algoritma *feature extraction* dapat mengenali citra karakter lebih baik dibandingkan dengan algoritma *template matching*. Hal ini terlihat pada gambar 7 yang mengilustrasikan persentase hasil pengenalan kedua algoritma terhadap dokumen uji.



Gambar 7. Persentase Hasil Pengenalan Terhadap Masing-Masing Dokumen Uji.

Algoritma *feature extraction* membutuhkan waktu yang lebih singkat dibandingkan dengan algoritma *template matching* dalam mengenali citra karakter yang terlihat pada gambar 8. Hal ini terjadi karena pada algoritma *template matching*, setiap *template* karakter memiliki 120 deretan pixel yang harus dibandingakan dengan 120 deretan pixel citra yang akan dikenali. Sedangkan pada algoritma *feature extraction*, terdapat 18 fitur yang harus dibandingkan.



Gambar 8. Grafik Waktu Masing-Masing Tipe Dokumen Uji

Berdasarkan analisis algoritma yang dilakukan, algoritma *feature extraction* lebih unggul dibandingkan dengan algoritma *template matching*. Keunggulan tersebut terlihat pada hasil pengukuran kedua algoritma berdasarkan hasil pengenalan, pengembangan, dan waktu

4. KESIMPULAN

Berikut ini adalah beberapa kesimpulan yang bisa didapat dari perbandingan algoritma *template matching* dan *feature extraction* pada OCR yang dibangun.

1. Berdasarkan hasil pengujian, algoritma feature extraction memiliki tingkat akurasi yang lebih baik daripada algoritma template matching. Algoritma template matching sulit

dikembangkan, karena algoritma template matching bekerja dengan menyesuaikan intensitas pixel. Algoritma feature extraction memiliki peluang untuk bisa dikembangkan terutama pada ciri-ciri khusus citra karakter. Salah satu contohnya adalah fitur stroke (jumlah garis yang membentuk karakter). Berdasarkan proses yang dibutuhkan kedua algoritma, algoritma feature extraction membutuhkan proses yang lebih banyak dibandingkan dengan algoritma *template matching*. Berdasarkan hasil perhitungan kompleksitas kedua algoritma, algoritma *feature extraction* dan algoritma *template matching* memiliki kompleksitas yang sama yaitu 2n+1. Berdasarkan hasil pengujian, algoritma *template matching* membutuhkan waktu yang lebih lama dibandingkan algoritma *feature extraction* dalam mengenali dokumen uji.

2. Algoritma *feature extraction* lebih baik digunakan pada OCR dibandingkan algoritma *template matching*.

DAFTAR PUSTAKA

[1] A. Saeed, "Implementation of Optical Character Recognition for Mobile Phones ", Engineering Department LANCASTER UNIVERSITY, 2008.

[2] [2] G. X. Ritter and J. N. Wilson, Handbook of Computer Vision Algorithms in Image Algebra: CRC Press 1996.

[3] [3] J. Han and M. Kamber, *Data Mining: Concepts and Techniques*, Second ed. San Fransisco: Elsevier, 2006.

[4] [4] M. Adri, "Computer Vision Basic Concept," pp. 4-6, 4/4 2009.

[5] [5] M. CHERIET, *et al.*, *Character Recognition Systems.* New Jersey: John Wiley & Sons, 2007.

[6] [6] M. S. Nixon and A. S. Aguado, *Feature Extraction and Image Processing*, First ed. London: Newnes, 2002.

[7] [7] Novhard. (2007, 5/6/2011). *Pattern Recognition atau Pengenalan Pola*. Available: http://novhard.wordpress.com/2007/09/07/pattern-recognition-atau-pengenalan-pola/

[8] [8] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, Second ed. New Jersey: Prentice-Hall, 2002.

[9] [9] R. Munir, "Kompleksitas Algoritma," Bandung, 2009.

[10] [10] S. M. C. Y. Suen and K. Yamamoto, "Historical Review of OCR Research and Development," *IEEE*, vol. 80, p. 1031, 1992.