OPTICAL CHARACTER RECOGNITION PADA SMART PHONE MENGGUNAKAN CONTOUR ANALYSIS DAN FEATURE PRESELECTION

Dwi Cahyo Nugroho[1)], Mahmud Dwi Sulistiyo[2)], Bedy Purnama[3)]
[1,2,3]Fakultas Informatika, Universitas Telkom
[1,2,3]Jl. Telekomunikasi No. 1 Terusan Buah Batu, Bandung 40257
Telp.: (022) 7564108, Fax.: (022) 7565930
E-mail: dwikcahyo@live.com[1)], mahmuddwis@telkomuniversity.ac.id[2)],
bedypurnama@telkomuniversity.ac.id[3)]

Abstrak

Optical Character Recognition (OCR) merupakan sebuah proses pembacaan gambar scan tulisan tangan, ketik, atau cetak oleh komputer sehingga diperoleh data teks digital. Kajian tentang OCR akhir-akhir ini ditujukan untuk menghasilkan akurasi yang baik serta beban komputasi yang semakin ringan. Hal tersebut dilakukan untuk memenuhi kebutuhan penerapannya pada perangkat bergerak yang saat ini sudah banyak digunakan. Pada penelitian ini, diterapkan metode Contour Analysis dengan representasi bilangan kompleks, serta descriptor Intercorrelation dan Autocorrelation Function. Ditambahkan pula proses contour smoothing menggunakan algoritma Ramer-Douglas-Peucker. Pada proses Feature Preselection, dibandingkan antara algoritma Longest Common Subsequence dengan Euclidian distance. Observasi dilakukan dalam menentukan kombinasi nilai parameter terbaik pada sistem OCR yang dibangun. Hasil perbandingan menunjukkan bahwa LCS lebih efisien dalam menyeleksi fitur kontur daripada Euclidean distance. Selain itu, penggunaan template Arial juga diuji dengan typeface Sans-Serif lainnya. Secara umum, hasil yang diperoleh sangat memuaskan karena rata-rata akurasi terbaik untuk setiap pengujian mampu mencapai angka 100%.

Kata kunci: *optical character recognition, contour analisis, intercorrelation, autocorrelation, feature preselection, longest common subsequence*

Abstract

Optical Character Recognition (OCR) is a process allowing computers to read handwritten, typed, or printed scanned characters, and produce digital text data. Study related to OCR recently is aimed to yield a high accuracy rate as well as a low cost of computational load. Those are for satisfying the needs of mobile devices applications that is used by many people currently. In this research, the complex representation is used in the Contour Analysis, combined by Intercorrelation and Autocorrelation Function. In the Feature Preselection process, there are Longest Common Subsequence and Euclidian distance to be compared. An observation is done to determine the best values for some parameters of the OCR system. The comparison results show that LCS is more efficient in selecting contour features than Euclidian distance. Besides, the use of Arial template is tested using other Sans-Serif typefaces. Generally, the results are quite satisfying since the average of the best accuracy rates for all testing can reach 100%.

Keywords: *optical character recognition, contour analisis, intercorrelation, autocorrelation, feature preselection, longest common subsequence*

1. PENDAHULUAN

Optical Character Recognition (OCR) adalah konversi mekanik atau elektronik pada sebuah komputer dari gambar *scan* tulisan tangan, ketik, atau cetak ke dalam *encoded text* [9]. Teks hasil pembacaan tersebut kemudian dapat digunakan sebagai masukan (*command*) untuk proses lainnya. Pada perkembangannya, sistem OCR dikembangkan agar dapat menerima input secara *online* dengan cara meng-*capture* sebuah citra, lalu mendeteksi dan mengenali teks yang ada di dalamnya. Kemampuan OCR tersebut saat ini sudah banyak diterapkan pada berbagai perangkat bergerak, seperti PC tablet, *smart phone*, dan beragam bentuk *gadget* lainnya. Hal tersebut dilakukan untuk memenuhi kebutuhan fitur teknologi pada perangkat bergerak. Terlebih pengguna perangkat bergerak di Indonesia, khususnya *smart phone*, dalam beberapa tahun terakhir ini terus meningkat. Dalam sebuah penelitian, dari 240 juta penduduk Indonesia di akhir tahun 2013, pengguna perangkat bergerak sudah mencapai

300 juta [1]. Untuk penerapan sistem OCR pada *smart phone*, kriteria paling mendasar yang diperlukan ialah ketepatan dan kecepatan. Artinya, diharapkan tingkat akurasi yang dihasilkan tinggi, namun tetap menjaga ke-*real time*-an dari proses pengenalan karakter yang dilakukan. Dikarenakan spesifikasi setiap *smart phone* sangat bervariasi, maka aplikasi yang dibangun haruslah mampu mengakomodasi, terutama untuk spesifikasi yang rendah, agar tetap menjaga kenyamanan pengguna [2]. Oleh karena itu, dari sisi teknis, hal tersebut dapat dilakukan dengan cara mengoptimalkan atau mengurangi kompleksitas proses komputasi yang berjalan di dalam sistem.

Beberapa metode yang sudah umum digunakan pada sistem OCR di perangkat bergerak, antara lain *Principal Component Analysis* (PCA), *Contour Analysis* (CA) [3], dan Tesseract *engine* [12]. Tesseract merupakan *engine* OCR yang popular digunakan karena akurasinya yang tinggi dan bersifat open source. Namun, Tesseract membutuhkan waktu komputasi yang tidak cepat karena menggunakan *statical classifier* [12].

CA menggunakan teknik *description extraction Inter Correlation Function* (ICF) dan *Auto Correlation Function* (ACF) mampu memberikan kompleksitas yang lebih kecil dibandingkan dengan menggunakan teknik lain [3]. Beberapa kajian sebelumnya menjelaskan bahwa CA mempresentasikan *contour* (kontur) sebagai jarak tiap *pixel* kontur ke sumbu x atau y [8][13]. Apabila kontur tersebut direpresentasikan dengan bilangan kompleks, maka akan lebih sederhana karena satu bilangan kompleks dapat menyatakan lebih dari satu bilangan *real*. Penggunaan bilangan kompleks dalam merepresentasikan kontur pada proses CA tersebut dapat dilakukan dengan menggunakan ACF dan ICF [3].

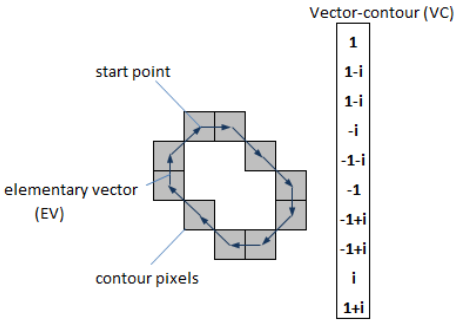
Perhitungan ACF dan ICF menjadi meningkat mengikuti banyaknya elemen kontur yang terdeteksi. Terlebih teknik tersebut sangat sensitif pada adanya *noise*. Untuk mengatasi hal tersebut, digunakan algoritma Ramer-Douglas-Peucker (RDP), yaitu algoritma penyerdehanaan kontur dengan mengurangi *noise* dan membuang elemen yang melebihi toleransi [4]. Untuk meringankan perhitungan, digunakan *contour preselection* pada proses *matching*, dengan menghitung jarak terlebih dahulu sebelum melakukan *matching* [7]. Pada tahapan tersebut, dapat digunakan algoritma *Longest Common Sequence* (LCS) [5] atau metode *Euclidean distance* [6].

2. METODE *OPTICAL CHARACTER RECOGNITION* (OCR)

2.1 Contour Analysis

2.1.1 Representasi Kontur

Dalam dunia *computer vision*, ada beberapa teknik yang popular untuk mempresentasikan sebuah kontur, di antaranya *code of freeman*, *two-dimensional coding*, dan *polygonal coding* [10]. Pada *Contour Analysis* (CA), teknik yang digunakan mirip dengan teknik-teknik tersebut, yaitu tetap menggunakan *chain code*. Setiap elemen *chain code* yang berhubungan satu sama lain dipresentasikan sebagai bilangan kompleks. Pada saat kontur sebuah citra di-*scan*, setiap vektor *offset* dicatat sebagai bilangan kompleks *a+ib*, di mana *a* merupakan *point offset* pada sumbu *x* dan *b* pada sumbu *y* [11].



Gambar 1 Representasi kontur dengan bilangan kompleks [11]

2.1.2 *Intercorrelation Function*

Intercorrelation Function (ICF) untuk dua buah kontur, yaitu Γ dan N [3] dinyatakan sebagai

$$\tau(m) = \left(\Gamma, \mathrm{N}^{(m)}\right), \quad m = 0, \dots, k-1 \tag{1}$$

k adalah panjang elemen dan $\mathrm{N}^{(m)}$ adalah kontur yang diterima dari N melalui pergeseran elemen vektornya. ICF menunjukkan seberapa mirip kontur Γ dan N jika hanya dilakukan *shifting* dari titik awal. Dengan adanya panjang maksimum vektor yang dinyatakan dalam persamaan berikut [3],

$$\tau_{max} = \max\left(\frac{\tau(m)}{|\Gamma||\mathrm{N}|}\right), \quad m = 0, \dots, k-1 \tag{2}$$

jelaslah bahwa τ_{max} merupakan proses pengukuran kesamaan dari dua buah kontur, yang tidak terpengaruh pada perubahan sudut, penskalaan, dan pergeseran titik awal. Jadi, nilai mutlak $|\tau_{max}|$ menunjukan level kemiripan kontur, dan τ_{max} memberikan seberapa besar sudut rotasi sebuah kontur terhadap kontur lain.

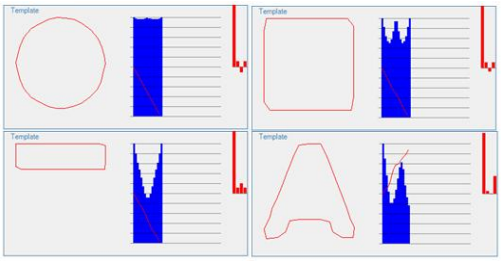
2.1.3 *Autocorrelation Function*

Autocorrelation Function (ACF) merupakan fungsi ICF dengan kondisi N = Γ. Artinya, ACF adalah fungsi ICF untuk sebuah kontur dengan dirinya sendiri berfungsi sebagai titik awal [3].

$$v(m) = \left(\Gamma, \Gamma^{(m)}\right), \quad m = 0, \dots, k-1 \tag{3}$$

Beberapa fitur yang dimiliki ACF [11] antara lain sebagai berikut.

1. ACF tidak bergantung pada pilihan titik awal kontur. Jika dilihat dari perkalian skalar di atas, perubahan titik awal hanya akan menyebabkan perubahan pada urutan eleman dan tidak menyebabkan perubahan pada hasil penjumlahan nilai elemen.
2. *Norm* ACF simetrik terhadap titik tengah *k/2* sebagaimana ACF merupakan total pasangan perkalian dari elemen vektor kontur, setiap pasangan bertemu dua kali dari 0 ke k.
   Sebagai contoh, jika *N = (n1, n2, n3, n4)*, maka nilai ACF untuk *n* yang berbeda adalah
   ACF(0)=(n1,n1)+(n2,n2)+(n3,n3)+(n4,n4)
   ACF(1)=(n1,n2)+(n2,n3)+(n3,n4)+(n4,n1)
   ACF(2)=(n1,n3)+(n2,n4)+(n3,n1)+(n4,n2)
   ACF(3)=(n1,n4)+(n2,n1)+(n3,n2)+(n4,n3)
   ACF(4)=(n1,n1)+(n2,n2)+(n3,n3)+(n4,n4)
   Perhatikan bahwa ternyata ACF(1) dan ACF(3) adalah sama. Di samping itu, mengingat bahwa untuk bilangan kompleks *(a,b) = (a,b)**, kita menerima bahwa *ACF(1) = ACF(3)**, di mana tanda (*) menunjukkan konjugat kompleks nomor. Sama seperti *ACF(1)* dan *ACF(3)*, *Norm* dari *ACF(0)* dan *ACF(4)* juga sama. Lebih lanjut, ACF juga dapat diartikan sebagai bagian fungsi dari interval *0* sampai *k/2* karena bagian sisanya bersimetris dengan bagian yang pertama.
3. Apabila sebuah kontur memiliki bagian yang simetri, maka ACF-nya juga memiliki bagian yang simetri.



Gambar 2 Grafik ACF pada kontur [11]

   Norm ACF direpresentasikan dengan warna biru, sedangkan ACF direpresentasikan hanya untuk interval *0* hingga *k/2*. Pada gambar di atas, semua kontur yang memiliki simetri juga memiliki ACF yang simetris, kecuali pada kontur yang terakhir.
4. ACF dapat digunakan untuk mengetahui karakteristik bentuk pada kontur.
5. *Norm* ACF tidak tergantung pada skala, posisi, rotasi dan pilihan titik awal kontur.

2.2 Algoritma *Ramer-Douglas-Peucker*

Algoritma *Ramer-Douglas-Peucker* (RDP) merupakan algoritma penyederhanaan kurva atau kontur yang populer digunakan untuk menghasilkan kontur dengan kualitas yang bagus dan halus [4]. Pada setiap langkah prosesnya, RDP melakukan penyederhanaan serangkaian titik per segmen garis dari poin pertama sampai poin terakhir. Pada titik yang paling jauh dari segmen garis tersebut, jika jaraknya ada di bawah *threshold*, maka penyederhanaan akan diterima (berhenti). Selain itu, maka algoritma akan merekursif lagi rangkaian titik yang dimulai dari dua sub-rangkaian sebelum dan sesudah titik yang sedang ditunjuk (titik terjauh dari segmen garis).
Pada kondisi *best case*, algoritma ini memiliki kompleksitas sebesar *Ω(n)*. Sedangkan pada kondisi *worst case*, algoritma ini mencapai *O(mn)*. Adapun waktu proses selama algoritma ini berjalan ada pada kisaran *2(n log m)*. Walaupun kurang optimal, algoritma ini umumnya mampu menghasilkan kualitas aproksimasi penyederhanaan terbaik secara objektif dan subjektif dibandingkan algoritma-algoritma heuristik lainnya [4].

2.3 *Feature Preselection*

2.3.1 *Longest Common Subsequence*

LCS merupakan sebuah permasalahan klasik di bidang ilmu komputer dalam mencari *substring* yang sama dari dua buah *string*. Konsep dari permasalahan ini banyak digunakan dalam analisis *time series*, perbandingan file,

dan identifikasi biometrik [5]. Ada banyak solusi yang telah dikembangkan untuk LCS, di antaranya secara rekursif, *dynamic programming*, dan masih banyak lagi variannya sesuai dengan kasus yang dihadapi [5].
Solusi untuk LCS ditujukan untuk membandingkan serangkaian nilai pergeseran *string* yang didefinisikan dengan fungsi sebagai berikut [9].

$$LCS(R,S) = \begin{cases} 0 & if\ m = 0\ or\ n = 0 \\ LCS\big(Rest(R), Rest(S)\big) + 1 & if\ \forall d, |r_{d,1} - s_{d,1}| \leq \epsilon \\ max\ \{LCS\ (Rest(R), S), LCS\big(R, Rest(S)\big)\} & otherwise \end{cases} \quad (4)$$

Keterangan:
R,S : merupakan serangkaian nilai $(r_1, \ldots, r_m)$ dan $(s_1, \ldots, s_n)$
r_i : elemen ke i dari *R*
Rest(R) : *R* dengan element pertama dihapus
ϵ : *threshold* jarak antara dua elemen

2.3.2 *Euclidean Distance*

Merupakan teknik yang paling sederhana untuk mengukur jarak di antara 2 buah titik atau vektor pada sebuah dimensi ruang *feature*. Misalnya ada 2 buah vektor *p* dan *q*, di mana $p = (p_1, p_2, \ldots, p_3)$ dan $q = (q_1, q_2, \ldots, q_3)$, maka jarak antara vektor *p* dan *q* apabila dihitung dengan *Euclidian distance* adalah sebagai berikut [6].

$$d(p,q) = \sqrt{\sum_{i=1}^{n}(p_i - q_i)^2} \quad (5)$$

Dari persamaan tersebut di atas, dapat dilihat jika semakin mirip elemen-elemen pada 2 buah vektor yang sedang dihitung, maka hasil perhitungan *Euclidian distance* akan semakin kecil.

3. PERANCANGAN SISTEM

3.1. Deskripsi Sistem

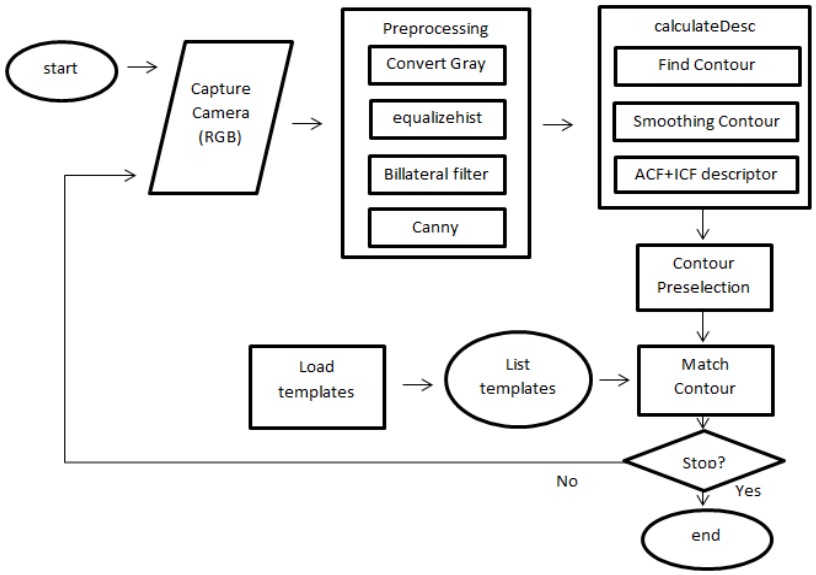
Pada penelitian ini, akan dibangun sistem OCR di atas platform Windows Phone. Seperti yang telah dijelaskan sebelumnya, metode yang digunakan ialah CA dengan ICF dan ACF. Sebelum tahap CA, dilakukan *pre-processing* terhadap citra hasil *capture*, yaitu dengan melakukan *image enhancement* dan reduksi *noise*. Selanjutnya, citra akan dikonversi menjadi citra biner dan dilakukan deteksi tepi/kontur menggunakan *Canny Edge detector*.

Kontur dikonversi menjadi bentuk bilangan kompleks. Proses penghalusan kontur dengan metode RDP juga dilakukan untuk mendapatkan hasil yang lebih baik. Selanjutnya, baru dilakukan perhitungan ICF dan ACF untuk menghasilkan *descriptor*. Perhitungan terhadap ICF dan ACF dilakukan untuk setiap kontur yang ditemukan, lalu *descriptor* yang dihasilkan dicocokkan dengan *template* kontur. Namun, sebelum dilakukan pencocokan kontur citra yang dibaca dengan kontur *template*, dilakukan proses pencocokan.

Sebelum proses pencocokan, dilakukan *Contour Preselection* dengan metode LCS atau *Euclidian distance* untuk mengurangi jumlah kontur yang menjadi *descriptor*, sehingga dapat mempercepat proses pencocokan. *Template* kontur didapatkan dari data latih yang jumlahnya kurang dari 60 kontur. Hal ini dilakukan agar waktu proses perhitungan tetap rendah. Setelah proses pencocokan, kontur yang dikenali ditandai dengan sebuah *box* dan karakter hasil pengenalannya.

3.2. Aplikasi OCR pada Windows Phone

Aplikasi OCR pada penelitian ini dibangun salah satunya dengan menggunakan perangkat lunak FastCV (*OpenCV wrapper* pada *Windows Phone*). Setelah simulasi dan pengujian berhasil, *prototype* aplikasi OCR tersebut baru ditanamkan ke sebuah *smart phone*. Alur proses yang berjalan pada aplikasi OCR tersebut ditunjukkan pada Gambar 3. Sistem menerima masukan berupa citra dari kamera berukuran 640x480 *pixel* yang diambil secara *real time* dari kamera *smart phone*. Citra yang masih dalam format RGB selanjutnya dikonversi menjadi citra *greyscale* pada tahap *pre-processing*. Pada *pre-processing* terdapat juga tahapan untuk memperbaiki kualitas citra. Selanjutnya, baru dilakukan proses CA, penghalusan kontur dengan RDP, perhitungan *descriptor* menggunakan ACF-ICF, *Contour Preselection*, dan pencocokan kontur, sehingga diperoleh hasilnya. Proses berulang kembali untuk mengenali karakter lainnya yang terbaca pada citra inputan.



Gambar 3 Flowchart aplikasi OCR yang dibangun

4. PENGUJIAN DAN ANALISIS

4.1. Observasi Beberapa Nilai Parameter OCR

Parameter pada sistem OCR yang akan diobservasi di sini yaitu *template count*, RDP *epsilon*, dan *connect point*. *Template count* merupakan jumlah keseluruhan titik pada *contour template* yang harus sama terhadap titik *contour sample* agar kedua kontur tersebut dapat dilakukan proses *matching*. RDP *epsilon* merupakan nilai toleransi algoritma RDP pada proses penghalusan kontur, di mana algoritma tersebut menghasilkan titik-titik kontur yang telah dikurangi *noise*-nya. *Connect point* merupakan pilihan (*true* atau *false*) menghubungkan titik-titik hasil RDP menjadi rangkaian titik yang saling berhubungan.

Tabel 1 Observasi beberapa nilai parameter OCR

Template count	RDP epsilon	Point connect	Rata2 akurasi (%)
12	2	TRUE	89.23
		FALSE	73.84
	3	TRUE	88.88
		FALSE	82.3
	4	TRUE	88.46
		FALSE	75.76
	Tanpa RDP		90.76
24	2	TRUE	93.46
		FALSE	61.53
	3	TRUE	93.37
		FALSE	79.23
	4	TRUE	95.92
		FALSE	76.92
	Tanpa RDP		93.07

Template count	RDP epsilon	Point connect	Rata2 akurasi (%)
32	**2**	**TRUE**	**100**
		FALSE	76.92
	3	TRUE	96.15
		FALSE	53.84
	4	TRUE	97.69
		FALSE	76.53
	Tanpa RDP		96.15
40	2	TRUE	94.23
		FALSE	76.92
	3	TRUE	94.61
		FALSE	69.62
	4	TRUE	93.46
		FALSE	74.07
	Tanpa RDP		88.84

Dari hasil tabel di atas terlihat bahwa kombinasi parameter yang paling baik adalah dengan menggunakan 32 titik *contour template*, dengan toleransi RDP sebesar 2, dan *connect point* yang bernilai *true*. Nilai *template count* berbanding lurus dengan jumlah *descriptor* ACF pada suatu *contour template*. Terlalu sedikit jumlah *descriptor* akan menyebabkan akurasi berkurang karena keambiguan *descriptor*. Sedangkan jumlah *descriptor* yang terlalu banyak pun akan mengurangi akurasi karena perhitungannya menjadi sangat ketat. Selain itu, proses perhitungannya pun akan semakin lama.

Nilai RDP *epsilon* mempengaruhi seberapa besar toleransi algoritma RDP dalam menentukan sebuah titik merupakan *noise* atau bukan. Jika jarak sebuah titik ke segmen garis berada di bawah nilai RDP *epsilon*, maka titik tersebut dianggap *noise*. Sebaliknya, jika berada di atasnya, maka titik tersebut dianggap bukan *noise*. Semakin tinggi nilai RDP *epsilon*, semakin sederhana pula bentuk kontur dan jauh dari bentuk asli *contour*, sehingga akurasi berkurang.

Di samping itu, secara umum terlihat bahwa *connect point* yang bernilai *false* memiliki akurasi di bawah *true*. Jika menggunakan *connect point* bernilai *false*, titik-titik kontur hasil RDP yang telah berkurang langsung disamakan jumlahnya dengan jumlah *template count* yang telah ditentukan. Jika jumlahnya jauh lebih sedikit daripada *template count*, maka akan ditambah titik-titik di antara titik-titik hasil RDP hingga sama jumlahnya. Penyamaan tersebut menjadi tidak seimbang jika jarak antara titik-titik hasil RDP tidak *uniform*, sehingga berdampak pada proses *matching*. Sedangkan apabila *connect point* bernilai *true*, maka titik-titik hasil RDP saling dihubungkan. Penyamaan jumlah titik-titik hasil RDP dengan *template count* menjadi seimbang karena setiap hubungan antar titik akan berjarak sama, yaitu 1 *point*. Oleh karenanya, tingkat akurasi dengan *connect point* bernilai *true* cenderung lebih tinggi daripada *connect point* bernilai *false*.

4.2. Perbandingan Antara *Euclidian Distance* dan LCS

Ada dua metode yang akan dibandingkan pada proses *contour preselection* sebelum *matching*, yaitu LCS dengan *Euclidian distance*. Pengujian dilakukan dengan membaca 26 karakter huruf (A – Z) dengan kombinasi nilai parameter terbaik yang didapatkan sebelumnya untuk *template count*, RDP *epsilon*, dan *connect point*.

Tabel 2 Contour preselection menggunakan Euclidian Distance

No	Threshold ED	Total pembandingan contour σ(I)	Jumlah contour preselection σ(i)	Rasio optimasi (σi/σI)	Rata2 akurasi (%)
1	**0.7**	**650**	**150.33**	**0.23**	**100**
2	0.8	606.67	120.67	0.2	98.847
3	0.85	598	107.33	0.18	98.583
4	0.9	598	89	0.15	94.46
5	0.95	520	59	0.11	93.8

Tabel 3 Contour preselection menggunakan LCS

No	ϵ elemen	Threshold LCS	Total pembandingan contour σ(I)	Jumlah contour preselection σ(i)	Rasio optimasi (σi/σI)	Rata2 akurasi (%)
1	0.5	0.8	676	571.67	0.85	100
2		0.9	658.67	470.33	0.71	99.82
3	0.3	0.8	676	416.33	0.62	100
4		0.9	676	211.33	0.31	99.92
5	**0.1**	**0.8**	**658.67**	**69**	**0.1**	**100**
6		0.9	667.33	31	0.05	99.66
7	0.05	0.8	600.33	24	0.04	93.97
8		0.9	275.33	10.33	0.04	71.6

Dari kedua tabel hasil percobaan di atas, terlihat bahwa proses *contour preselection* menggunakan *Euclidean distance* dengan nilai parameter terbaiknya berhasil mencapai akurasi 100% dan menghasilkan kontur sebanyak 150. Adapun LCS beserta nilai parameter terbaiknya juga mampu mencapai akurasi 100% dan menghasilkan kontur hanya sebanyak 69. Hal tersebut tentu akan berpengaruh pada waktu pemrosesan pengenalan karakter karena jumlah kontur yang akan dicocokkan berbanding lurus dengan waktu pemrosesannya.

4.3. Akurasi OCR pada Beberapa Typeface Sans-Serif

Setelah dilakukan percobaan dan observasi untuk mengetahui pengaruh nilai-nilai parameter pada sistem OCR yang dibangun sekaligus menentukan nilai-nilai mana yang terbaik, maka selanjutnya dilakukan percobaan untuk mengetahui kemampuan sistem OCR ini pada beberapa jenis huruf (*typeface*) yang berbeda. *Typeface* yang digunakan sebagai *template* adalah Arial, sedangkan yang akan dikenali adalah *typeface* Sans-Serif, yaitu Calibri, Futura, dan MS Sans-Serif. Ada 26 huruf (A – Z) yang akan dikenali pada jarak jauh, sedang, dan dekat. Definisi jarak jauh, sedang, dan dekat adalah ketika 'tinggi' setiap huruf adalah sekitar 24, 38, dan 56 *pixel*.

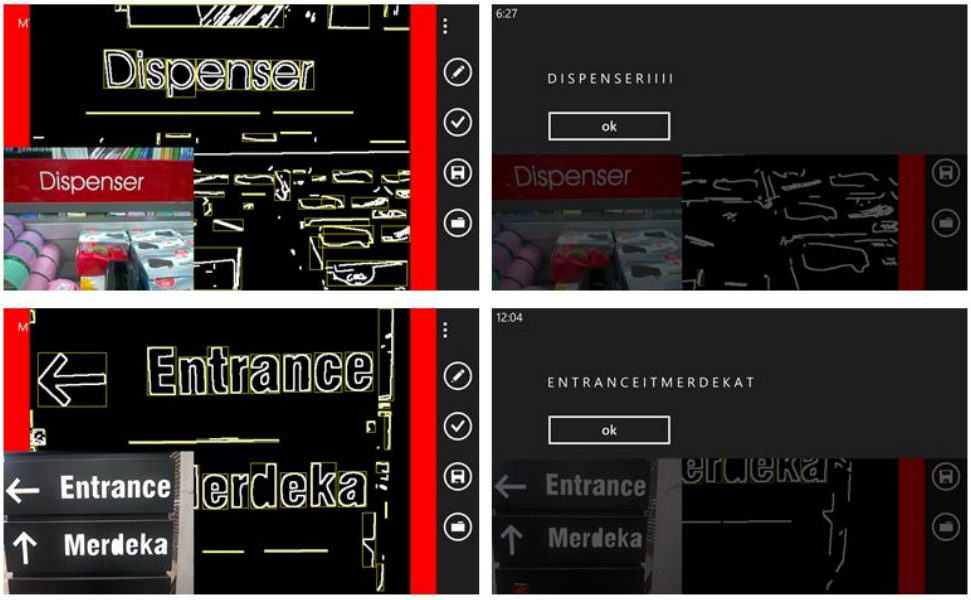
Tabel 4 Pengujian aplikasi OCR pada beberapa typeface Sans-Serif

Typeface	Rata2 akurasi (%) pada jarak		
	Jauh (24 px)	Sedang (32 px)	Dekat (56 px)
Calibri	79.99	88.84	97.3
Futura	**80.76**	91.15	98.46
MS Sans-Serif	80.38	**97.69**	**99.23**

Gambar 4 Kondisi pengujian aplikasi OCR (kiri ke kanan: jauh, sedang, dekat)

Berdasarkan tabel 4 di atas, MS Sans-Serif memiliki rata-rata akurasi yang lebih baik dibandingkan kedua *typeface* lain yang diujikan. Hal ini disebabkan MS Sans-Serif merupakan *neo-grotesque typeface* yang memiliki kedekatan karakteristik bentuk dengan Arial. Selain itu, dapat terlihat pula bahwa secara umum, semakin dekat jarak kamera dengan tulisan yang akan dikenali, semakin tinggi pula rata-rata akurasinya. Hal ini terjadi karena ketika semakin kecil citra huruf yang akan dikenali, detail setiap konturnya akan berkurang, sehingga *contour sample* yang diperoleh semakin sedikit pula.

Adapun pengujian aplikasi OCR yang dibangun di sini juga telah diujikan pada beberapa *natural scenes*. Hasil dari pembacaan karakter oleh aplikasi secara umum masih sangat dipengaruhi oleh *noise* lingkungan, yaitu objek-objek selain huruf di sekitar tulisan yang dianggap sebagai huruf. Hal tersebut perlu menjadi perhatian dan penanganan khusus pada penelitian selanjutnya.



Gambar 5 Hasil pembacaan aplikasi OCR pada natural scenese (kiri: hasil capture, kanan: hasil pembacaan)

5. SIMPULAN DAN SARAN

5.1. Simpulan

Berdasarkan hasil pengujian dan analisis pada penelitian ini, dapat disimpulkan bahwa:

1. Hasil observasi terhadap beberapa parameter pada sistem OCR yang dibangun menunjukkan bahwa dari nilai-nilai yang diujikan, rata-rata akurasi terbaik diperoleh dengan *template count* sebanyak 32 titik, RDP *epsilon* sebesar 2, dan *connect point* yang bernilai *true*.
2. Hasil perbandingan antara kedua metode pada proses *Feature Preselection* menunjukkan bahwa LCS lebih baik daripada *Euclidean distance* karena dengan tingkat akurasi yang sama-sama mencapai 100%, LCS mampu menghasilkan jumlah kontur preseleksi yang jauh lebih sedikit dibandingkan *Euclidian distance*.
3. Aplikasi OCR yang diuji dengan *typeface* MS Sans-Serif memberikan akurasi terbaik dibandingkan dua *typeface* lainnya. Jarak antara kamera dengan tulisan yang akan di-*capture* cukup berpengaruh, di mana semakin jauh akan semakin berkurang akurasinya, begitu pun sebaliknya.
4. Penggunaan aplikasi OCR yang dibangun untuk membaca *natural scenes* sudah cukup baik. Kekurangannya disebabkan oleh *noise* lingkungan yang ikut terbaca dan dikenali sebagai karakter.

5.2. Saran

Adapun beberapa hal yang disarankan untuk pengembangan pada penelian selanjutnya antara lain:

1. Beberapa metode alternatif dapat diterapkan pada proses *Feature Preselection*, seperti *Dynamic Time Warping*, *Edit Distance with Real Penalty*, atau *Edit Distance on Real Sequences*.
2. Untuk mengurangi *noise* lingkungan pada pembacaan *natural scenes* dapat ditambahkan *feature descriptor* untuk deteksi objek, seperti *Histogram of Oriented Gradient* (HOG).
3. Studi kasus OCR dapat dikembangkan untuk mengenali karakter yang memiliki *closed contour*.

6. DAFTAR RUJUKAN

[1] Andi. 22 Desember 2013. Tren Teknologi Masa Depan 2014 di Indonesia. [Online] Tersedia: http://www.jurnallogistik.com/2013/12/trenteknologi-2014.html

[2] Anonim. 2011. Kriteria Aplikasi Mobile Yang Baik. [Online] Tersedia: http://vinnaanggraeni.blogspot.com/2011/05/kriteria-aplikasi-mobile-yang-baik.html

[3] Furman Y.A. 2003. Introduction to the Contour Analysis. Application to Image Processing and Signal. FML.

[4] Heckbert, Paul S., and Michael Garland. 1997. Survey of Polygonal Surface Simplification Algorithms. Carnegie-Mellon Univ. Pittsburgh, PA School of Computer Science.

[5] Khan, Rahim, Mushtaq Ahmad, and Muhammad Zakarya. 2013. Longest Common Subsequence Based Algorithm for Measuring Similarity Between Time Series: A New Approach. World Applied Sciences Journal 24.9.

[6] Kin-Pong Chan and Ada Wai-Chee Fu. 1999. Efficient Time Series Matching by Wavelets. Proceedings of the 15th International Conference on Data Engineering (ICDE '99), ACM.

[7] Larsson, Fredrik, Michael Felsberg, and P-E. Forsse'n. 2011. Correlating Fourier Descriptors of Local Patches for Road Sign Recognition. Computer Vision, IET 5.4: 244-254.

[8] Mirza Triaksono. 2008. Implementasi Optical Character Recognition dengan Pendekatan Metode Struktur Menggunakan Ekstraksi Ciri Vektor dan Region. Institut Teknologi Telkom, Bandung, Tugas Akhir.

[9] Morse, Michael D., and Jignesh M. Patel. 2007. An Efficient and Accurate Method for Evaluating Time Series Similarity. Proceedings of the 2007 ACM SIGMOD international conference on Management of data, ACM.

[10] Øivind Due Trier. 1995. Feature Extraction Methods for Character Recognition - A Survey. Pattern Recognition, Volume 29, Issue 4, page 641-662.

[11] Pavel Torgashov. June 2011. Contour Analysis for Image Recognition in C. [Online] Available: http://www.codeproject.com/Articles/196168/Contour-Analysis-for-Image-Recognition-in-C

[12] R. Smith. 2006. An Overview of the Tesseract OCR Engine. Proceeding of International Conference Document Analysis Recognition, page 629-633.

[13] Teguh Catur Prakosa. 2009. Implementasi dan Analisis Handwriting Character Recognition Menggunakan Ekstraksi Ciri Titik, Normalized Contour Analysis, dan Backpropagation. Institut Teknologi Telkom, Bandung, Tugas Akhir.