Implementasi Tesseract OCR untuk Pembuatan Aplikasi Pengenalan Nota pada Android

Yoel Andreas[1], Kartika Gunadi[2], Anita Nathania Purbowo[3]
Program Studi Informatika, Fakultas Teknologi Industri, Universitas Kristen Petra
Surabaya, Indonesia
+6281216055020[1], +628123528487[2], +62818571336[3]
yoelandreas12@gmail.com[1], kgunadi@petra.ac.id[2], anitaforpetra@gmail.com[3]

ABSTRAK

Perkembangan jaman yang serba praktis membuat manusia lebih condong mencari jalan cepat untuk melakukan sesuatu. Sama hal nya saat kita ingin untuk mencatat pengeluaran yang sudah kita keluarkan dalam 1 hari, tentu memakan waktu untuk melakukannya. Untuk meneyelesaikan masalah tersebut, dapat menggunakan aplikasi untuk membaca nota dengan menggunakan kamera perangkat android, aplikasi tersebut dapat membantu untuk mencatat pengeluaran beserta pengkategorian pengeluarannya.

Untuk mencapai hal tersebut, maka perlu melakukan *Optical Character Recognition* yang dapat dilakukan dengan menggunakan *Tesseract-OCR*. Hasilnya akan di olah untuk mendapatkan pengeluaran, kategori, beserta nama *item* nya. Untuk mendapatkan hasil yang maksimal maka diperlukan beberapa tahap *pre-processing* pada gambar yang akan digunakkan. Pengujian dilakukan dengan studi skenario dan mencoba beberapa kasus, misalnya nota dengan *dotted fonts,* atau nota yang memiliki banyak garis.

Hasil pengujian menunujukan bahwa hasil OCR dari *Tesseract* sangat bergantung pada tahap *pre-processing* yang dilakukkan. *Tesseract* sendiri akan mengalami penurunan performa saat mengolah gambar dengan *dotted fonts*. Apabila tahap *pre-processing* tidak dapat menyatukan huruf – huruf yang terpisah karena titik – titik maka *tesseract* mengalami penurunan akurasi yang sangat drastis. Nota yang memiliki banyak garis juga mengurangi performa dari *tesseract*. Hasil *tesseract* saat melakukan HCR juga terpengaruh bagaimana tulisan tangan yang akan proses, jika tulisan tangan menyambung ataupun tidak rapi, maka *tesseract* akan mengalami kesulitan dalam melakukan proses HCR

Kata Kunci: *Optical Character Recognition*, *Tesseract*, Nota, OCR, HCR.

ABSTRACT

The development of a practical era makes humans more inclined to find a fast way to do something. The same thing when we want to record the expenses we have spent in the day, of course it takes time to do it. To solve this problem, you can use the application to read the receipt using the Android device's camera, the application can help to record expenses and categorize their expenses.

To achieve this, it is necessary to do Optical Character Recognition which can be done using Tesseract-OCR. The results will be processed to get expenses, categories, and item names. To get maximum results, several stages of pre-processing are needed on the image to be used. The test is carried out with a scenary study and tried several cases, for example notes with dotted fonts, or notes that have many lines.

The test results show that the OCR results from the Tesseract are very dependent on the pre-processing stage being carried out. Tesseract itself will experience a decrease in performance when processing images with dotted fonts. If the pre-processing stage cannot unite separate letters due to dots, the tesseract has a very drastic decrease in accuracy. Notes with multiple lines also reduce the performance of the tesseract. The results of the tesseract when conducting Handwritten Character Recognition are also affected by how the handwriting are written, if the handwriting is cursive or not neat, then the tesseract will have difficulty in carrying out the HCR process.

Keywords: *Optical Character Recognition, Tesseract, Receipt, OCR, Handwritten Character Recognition, HCR.*

1. PENDAHULUAN

Pada zaman modern sekarang, teknologi telah menjadi bagian dari kehidupan manusia sehari – hari. Teknologi sudah dapat membantu manusia dalam menjalankan aktivitas mereka sehari – hari. Seiring perkembangan tersebut, aplikasi *mobile* juga berkembang pesat dan sangat diminati oleh beragam kalangan masyarakat karena kemudahan dalam penggunaanya, serta sifatnya yang dapat digunakan dimana saja. Salah satu sistem operasi yang paling diminati adalah Android, sekitar 74,85% pasar sistem operasi pada perangkat *mobile* sudah didominasi oleh Android [3]

Salah satu masalah yang dapat dialami manusia adalah dalam mengelola keuangan sehari – hari, agar lebih mudah dalam mengelola keuangan dan mengetahui pengeluaran apa saja yang sudah dilakukkan dalam hari – hari sebelum nya. Hal ini dapat dipermudah dengan menggunakan aplikasi Android yang dapat mengenali nota pembayaran tersebut dan mendapatkan *item* dari nota tersebut, lalu akan dikategorikan sesuai kategori nya. Dengan begitu, pembayaran yang telah dilakukan dapat tercatat dan diketagorikan sesuai kategori pengeluarannya.

Untuk melakukan proses pengambilan data dari nota, akan dilakukan menggunakan *Optical Character Recognition (OCR)* untuk mendapatkan *text* dari sebuah gambar yang diambil. OCR sampai saat ini juga masih dikembangkan dengan berbagai metode klasifikasi karakter [1] Pada saat ini implementasi OCR dipermudah dengan adanya *library Tesseract OCR* dan tingkat akurasinya juga tergolong tinggi. Proses tersebut bisa dilakukan untuk mendapatkan data nama *item* dari nota dan di masukkan kedalam aplikasi. Sehingga aplikasi dapat langsung mengkategorikan pengeluaran tersebut dan pengguna dapat menyimpan *history* pengeluarannya.

2. TINJAUAN STUDI

Pada percobaan pada implementasi penggunaan *Tesseract OCR* untuk meng ekstraksi data seperti nama *item* dan harga, *heuristics rules based approach* dapat digunakan [6].

Percobaan lain mengusulkan penggunaan metode *Dynamic Time Wrapping (DTW)* menggunakan *k nearest neighbor (*KNN*)* untuk melakukan OCR pada nota berbahasa makedonia. Namun hasil menunjukan bahwa *tesseract* orisinil memiliki akurasi yang lebih tinggi sebanyak 6% [2].

Percobaan lain mengusulkan melakukan OCR dengan menggunakan Jaringan Saraf Tiruan dengan tingkat akurasi 94,7% [1].

3. DESAIN SISTEM

Proses dalam pembuatan sistem meliputi *image pre-processing*, melakukan *Optical Character Recognition* dengan *tesseract,* melakukan ekstraksi data yang diperlukan, dan desain *user interface*. Terdapat *back end* dan *front end* dalam pembuatan sistem, *back end* bertugas untuk memproses gambar yang diupload serta mengirim data setelah terproses ke database dan ke frontend. Sedangkan *front end* berguna untuk menampilkan data yang didapat serta mempermudah proses untuk pengambilan gambar.

3.1 *Image Edge Detection*

Proses Image Edge Detection adalah suatu proses untuk mencari edge yang terdapat pada suatu gambar. Proses ini dilakukan untuk mendapatkan letak nota di dalam gambar. Untuk mendapatkan edge dari gambar dapat menggunakan cv2 *canny* [5]. Lalu setelah mendapatkan edge, dilakukan proses findcontour, lalu dilakukan proses cv2 approxPolyDP untuk mendapatkan lokasi tiap sudut nya.

3.2 *Perspective Transformation*

Untuk mendapatkan hasil yang didapat dari *tesseract* lebih akurat, dapat melakukan proses untuk menghilangkan *background* [6]. Untuk melakukan proses tersebut dapat menggunakan sudut yang sudah didapatkan akan diurutkan berdasarkan posisi nya sesuai urutan (*top-left, top-right, bottom-right, bottom-left*). Setelah itu, dengan data yang koordinat yang sudah didapat akan dapat menentukan ukuran lebar dan panjang gambar yang baru. Lalu akan menggunakan opencv untuk melakukan *perspective transformation* untuk menghasilkan nota dengan tampilan *top-down view* dengan *background* yang sudah dihilangkan

3.3 *Image Skewing*

Tesseract memiliki kelemahan saat membaca *text* yang miring [6]. Pada proses ini, gambar yang sudah di proses sebelumnya akan di-*skew* untuk memperbaiki kemiringan pada gambar, agar text yang berada di gambar menjadi lurus. *Skewing* di batasi sebanyak 5 derajat maksimum. Image skewing dilakukan dengan cara mencoba gambar dimiringkan dari antara -5 sampai 5 derajat, lalu gambar akan di proyeksikan secara vertikal lalu akan dicari histogram dari pixel nya untuk mendapatkan perbedaan antara *peak* tertinggi nya. sudut dengan perbedaan *peak* tertinggi merupakan sudut terbaik.

3.4 *Morphological Transformation*

Dalam proses ini, *morphological transformation* akan digunakan dua kali. Kegunaan pertamanya adalah untuk mencari garis pada nota. Proses dimulai dengan meng-*invert* gambar biner terlebih dahulu. Lalu dengan menggunakan *morphological opening (*erosi lalu dilasi*)* dan kernel yang sudah ditetapkan, gambar hasil invert akan hanya menyisakan bentuk garis. Kemudian untuk mendapatkan letak dari garis, akan menggunakan cv2 findContours, yang akan disimpan dalam *variable*. Lalu gambar biner akan diperbaiki dengan cara cv2 drawContours dengan warna putih(255, 255, 255), dengan begitu garis pada nota akan hilang karena diwarnai dengan warna putih

Morphological transformation juga dapat digunakan untuk menutup gambar yang memiliki lubang - lubang kecil [4], sehingga *morphological transformation* juga dapat dimanfaatkan untuk menggabungkan huruf cetakan yang berupa *dotted fonts*. Untuk menggabungkanya maka akan menggunakan *morphological closing*.

3.5 *User Interface*

Gambar 1 merupakan tampilan pada aplikasi *receipt scanner*. Pada *tab home* akan menampilkan *pie chart* dari tiap kategori pengeluaran yang sudah disimpan. Di halaman tersebut juga terdapat tombol untuk memasukan gambar dari receipt yang akan di *scan.* Pada *tab details* akan menampilkan seluruh detail pengeluaran yang dapat di filter berdsarkan keinginan *user.* Pada *tab scan receipt*, dapat meng *upload* gambar *recipt* yang ingin di *scan*.

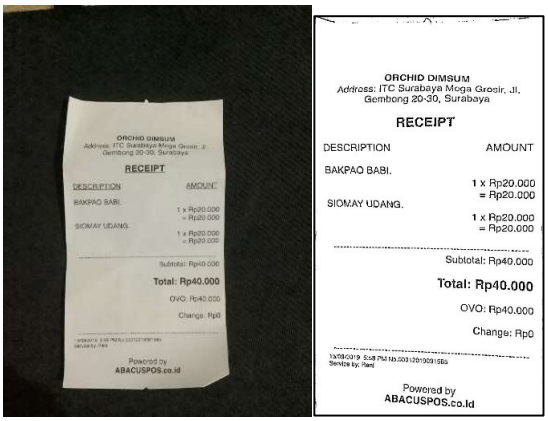
Gambar 1. Tampilan dari program *mobile*

4. PENGUJIAN SISTEM

Pada bab ini akan dilakukan pengujian pada sistem yang sudah dibuat. Pengujian dilakukan dengan cara melakukan beberapa skenario dan beberapa contoh di berbagai nota yang sudah dikumpulkan. Skenario yang telah disiapkan adalah nota dengan huruf balok, nota dengan *dotted fonts,* nota normal, nota dengan tulisan tangan.

4.1 Contoh kasus pertama dengan nota huruf balok

Pada pengujian, dilakukan terhadap nota dengan nota ketikan dengan huruf balok seperti pada Gambar 2. Lalu dijlanjutkan dengan tahap *image pre-processing* sehingga menghasilkan Gambar 2. Setelah itu dilakukan proses OCR menggunakan *tesseract* yang hasil nya dapat dilihat pada Gambar 3. Lalu dilanjutkan dengan meng-ekstraksi data yang diperlukan agar dapat mengklasifikasi kategori pengeluaran nota. Hasil klasifikasi dapat dilihat di Gambar 3. Dapat dilihat bahwa hasil yang didapatkan bagus dengan klasifikasi yang dilakukan benar semua. Dapat disimpulkan program bekerja dengan baik dalam pengujian dengan menggunakan nota huruf balok.



Gambar 2. Nota dengan huruf balok (kiri), hasil *pre-processing* (kanan)

```
DRCHID DIMSUM
Address: ITC Surabaya Moga Grosir, JI.
Gembong 20-30, Surabaya

RECEIPT
DESCRIPTION AMOUNT
BAKPAO BABI.
1x Rp20.000
= Rp20.000
SIOMAY UDANG.
1 x Rp20.000
= Rp20.000

Subtotal: Rp40.000 |

Total: Rp40.000

OVO: Rp40.000 |

Change: Rpo —




5 3:58 PM No:0001 2019091565
Ice by: Rent 566


Powered b
| ABACUSPOS.co.id
```

```
bakpao babi.          20000 food
siomay udang.         20000 food
```

Gambar 3. Hasil dari *tesseract* (atas), hasil klasifikasi (bawah)

4.2 Contoh kasus kedua dengan nota *dotted font*

Pada pengujian kedua ini akan dilakukan pada nota yang memiliki *dotted font,* pengujian dilakukan dengan menggunakan Gambar 4.

Gambar 4. Nota dengan *dotted fonts*

```
quia

OR/L/2019 1322



225,000

21758
```

Gambar 5. Hasil konversi *tesseract*

Dari hasil yang didapatkan seperti pada Gambar 5, tanpa proses *morphological closing* (dilasi lalu erosi)*, tesseract* tidak dapat membaca *dotted fonts.* Lalu pengujian ini akan dilanjutkan dengan melakukan *morphological closing* pada gambar.

```
Kitchenette
Galaxy. Mall 6F Unit-206

TAHLE— 16 .
Far? OF: TIAT OYLTA
POS TitlasCashier .
“ “ 08/14/2019 12734 —

maru Aaa sa k marg ==



% Chicken Teriyaki _ 79,000
Garlio Fries 45,006
2 Ice Fea a 40,000
1 Hasi Boreng é 55,000
SUBTOTAL 279,000
Service . 13,950
PHL 29,85
TOTAL 322,245
YOUICHER OL “. > 100,00d
[1142619399990A84R] —
CASH 225,000
C aa ye Syl
KeaberiMs. patrice thesdara 1
Class=Basic .
<n= === ma aaa maa
. Closed Bill .
4——>———08/11/2019 LI me

Downlox] the ee fest
$i ag an Isnaya estyle Hember,
4 ton all the special benerats: —
```

Gambar 6. Hasil dari konversi *tesseract* setelah proses *morphological transformation*

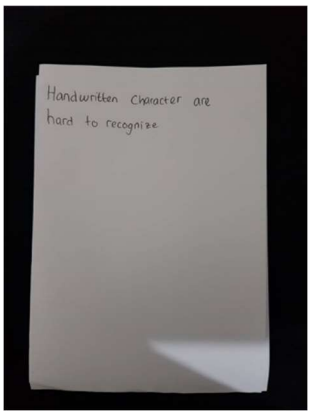
Dapat dilihat pada Gambar 6 bahwa hasil *tesseract* mengingkat setelah melakukan proses *morphological transformation.* Dari proses tersebut lalu akan dilanjutkan dengan ekstraksi data, hasil ekstraksi dapat dilihat pada Gambar 7, data yang diapat memiliki beberapa kesalahan seperti, ada 9 huruf yang salah teridentifikasi oleh *tesseract.*

```
chicken teriyaki      79000 food
garlio fries          45006 food
ice fea a             40000 food
hasi boreng é         55000 food
```

Gambar 7. Hasil klasifikasi

4.3 Contoh kasus dengan tulisan tangan

Pada pengujian ini, akan dilakukan pengujian terhadap tulisan tangan terlebih dahulu. Tulisan tangan dapat dilihat pada Gambar 8. Gambar 9 menunjukan hasil dari *handwritten character recognition.* Dari hasil yang didapat, dapat dilihat bahwa ada 4 huruf yang gagal di prediksi oleh *tesseract.* Selanjutnya akan melakukan pengujian dengan menggunakan Gambar 10. Hasil klasifikasi dapat dilihat pada Gambar 11.



Gambar 8. Contoh tulisan tangan

```
~and writken Character ae

Nard to recognize
```

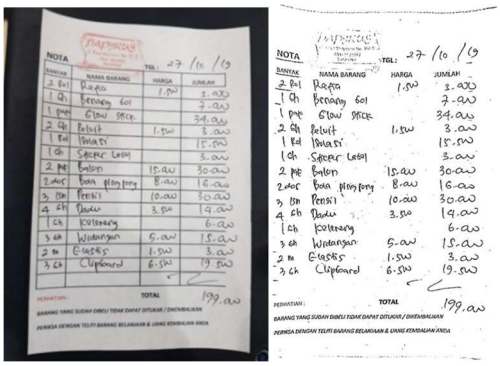
Gambar 9. Hasil konversi *tesseract*

Gambar 10. Contoh nota tulis tangan

```
nasi goreng                                  16000 food
 s teh manis                                  4000 food
```

Gambar 11. Hasil klasifikasi

Percobaan selanjutnya dilakukan menggunakan 12, dimana nota tulisan tangan tersebut tidak rapi. Dapat dilihat pada Gambar 13 *tesseract* gagal mendapatkan *text* dengan akurat. Dengan begitu proses klasifikasi tidak dapat dilakukan.



Gambar 12. Nota tulisan tangan (kiri), hasil *pre-processing* (kanan)

```
{-ch
2 ae
ee
> {Sh
a ch

[ £h

- y a
San
» th

PERHATIAN :

BARANG YANG SUDAH DIBELI TIDAK DAPAT DITUKAR / DIKEMBALIKAN

Rean
benone $0)
Clow _ $ice
PkLuH- a

kar

Sépoeer Case)

Laon

Rota ping tong. -

Pens) *
Qaar |.
Cotrrang
|Widangan:
Cais -
Kigkare

| TOYAL |

(w (C

HARGA JUMLAH _
(9 20 7 3. A)
(9 20 7 3. A)
FRB

84,008

[kw 3.00

[ew
BON
"8, Qu0 20-0

Rav |6-cvw

[0-cvo 50-AN

3.90 [4.07

6-A
G-ON |S. AS
15 Zod
Bw. (9-009
De et
'99 an

PERIKSA DENGAN TELITI BARANG BELANJAAN & UANG KEMBALIAN ANDA
```

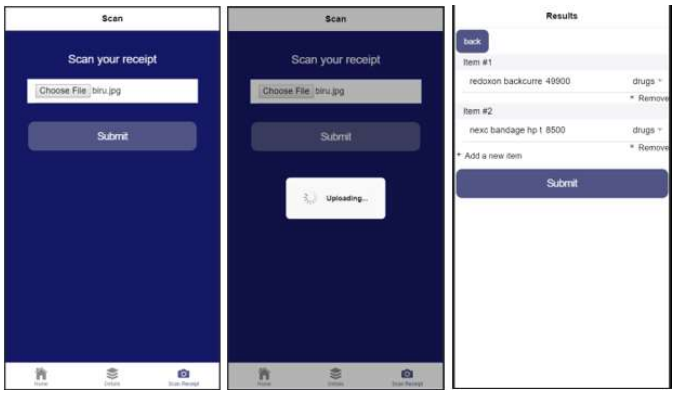
Gambar 13. Hasil konversi *tesseract*

4.4 Pengujian aplikasi *mobile*

Pengujian aplikasi *mobile* dilakukan menggunakan nota dengan *background* berwarna seperti pada Gambar 14.

Gambar 14. Nota dengan *background* berwarna biru

Gambar 15. Tampilan aplikasi *mobile*



Gambar 16. Tahap saat melakukan *scan receipt*

Gambar 15 menunjukan tampilan dari aplikasi *mobile.* Gambar 16 menunjukan proses yang dilakukan saat ingin melakukan *scan receipt*. Dapat dilihat hasil klasifikasi yang didapatkan sudah benar seluruh nya. Data akan disubmit lalu akan masuk ke dalam *database* dan akan di-*load* kedalam aplikasi. Tampilan setelah submit data dapat dilihat pada Gambar 17.

Gambar 17. Tampilan setelah data di *upload*

5. KESIMPULAN DAN SARAN

Berdasarkan hasil pengamatan dan pengujian pada sistem, dapat disimpulkan beberapa hal sebagai berikut:

- Berdasarkan hasil pengujian pada sistem, maka dapat disimpulkan bahwa *tesseract* sangat bergantung pada tahap *pre-processing* gambar sebelum melakukan proses OCR.
- Berdasarkan hasil pengujian pada nota dengan huruf balok dapat disimpulkan bahwa, *tesseract* berperforma baik pada cetakan dengan huruf balok. Dapat disimpulkan juga bahwa *tesseract* berperforma baik jika jarak baris pada nota renggang.
- Berdasarkan hasil pengujian pada nota dengan *dotted fonts*, dapat disimpulkan bahwa, *tesseract* berperforma buruk pada cetakan dengan font putus – putus (*dotted fonts*). Dengan menggunakan proses *morphology closing* dapat meningkatkan performa *tesseract.* Selain itu, dari hasil yang diperoleh, dapat disimpulkan juga bahwa *tesseract* susah dalam membaca karakter dengan yang terdiri dari garis *horizontal* (*underscore,* sama dengan), terlihat bahwa *tesseract* mencoba membaca symbol tersebut sebagai karakter huruf.
- Berdasarkan hasil pengujian pada nota dengan tulisan tangan, dapat disimpulkan bahwa, *tesseract* dapat melakukan proses HCR. Namun performa *tesseract* dalam melakukan HCR menurun apabila tulisan tangan merupakan tulisan yang bersambung. Selain itu, garis – garis pada nota juga mengurangi akurasi *tesseract* dalam membaca *text* pada gambar. Selain itu, *tesseract* juga mengalami kesulitan untuk membaca tulisan yang tidak rapi (tidak sejajar, ukuran *font* yang tidak tetap).
- Berdasarkan hasil pengujian nota dengan *background* berwarna, dapat disimpulkan bahwa program mampu bekerja dengan *background* berwarna.

Dengan kesimpulan yang sudah dinyatakan, ada beberapa hal yang dapat dijadikan sebagai saran untuk pengembangan selanjutnya yaitu:

- Lakukan tahap preproses pada gambar dengan dotted fonts untuk menyatukan jarak putus – putus pada huruf.
- Untuk menghasilkan hasil yang baik, disaran kan untuk menghilangkan background disekitar nota.
- Pada gambar yang memiliki banyak garis, sebaik nya dilakukan beberapa tahap preproses terlebih dahulu untuk menghilangkan garis pada nota sepenuhnya.

6. DAFTAR PUSTAKA

[1] Apriyanti, K., & Widodo, T.W. 2016. Implementasi Optical Character Recognition Berbasis Backpropagation untuk Text to Speech Perangkat Android. *Indonesian Journal of Electronics and Instrumentation Systems (IJEIS). 6(1). 13-24.*

[2] Gjoreski, M., Zajkovski, G., Bogatinov, A., Madjarov, G., Gjorgjevikj, D. 2014. Optical character recognition applied on receipts printed in Macedonian Language. International Conference on Informatics and Information Technologies (CIIT). 59-62.

[3] *Mobile Operating System Market Share Worldwide – April 2019. 2019. Statcounter. Retrieved from* http://gs.statcounter.com/os-market-share/mobile/worldwide

[4] Morphological Transformations. Retrieved from https://docs.opencv.org/3.4/d9/d61/tutorial_py_morphological_ops.html

[5] Structural Analysis and Shape Descriptors. Retrieved from https://docs.opencv.org/2.4/modules/imgproc/doc/structural_analysis_and_shape_descriptors.html

[6] Ullah, R., Sohani, A., Rai, A., Ali, F., & Messier, R. (2018). OCR Engine to Extract Food-Items, Prices, Quantity, Units from Receipt Images, Heuristics Rules Based Approach. *International Journal of Scientific & Engineering Research*, Vol.9(2). pp. 1334-1341