PERANCANGAN PEMINDAI DOKUMEN CETAK PORTABEL MENGGUNAKAN TESSERACT DAN OPENCV

W. Priharti[1], K. Sujatmoko[2], M.A.S. Abubakar[3],
[1,3]Program Studi S1 Teknik Elektro, Fakultas Teknik Elektro, Universitas Telkom
[2]Program Studi S1 Teknik Telekomunikasi, Fakultas Teknik Elektro, Universitas Telkom
waa
[1]wpriharti@telkomuniversity.ac.id, **[2]krissujatmoko@telkomuniversity.ac.id**,
[3]ariefsyahnakri@student.telkomuniversity.ac.id,

Diterima pada 20 Februari 2022; disetujui pada 21 Oktober 2022; dan diterbitkan pada 31 Desember 2022.

Abstrak

Dokumen cetak masih menjadi pilihan beberapa industri untuk menyimpan data-data penting dalam bentuk faktur, struk, dan dokumen cetak lainnya. Hal ini menimbulkan masalah ketika diperlukan bentuk data digital dari dokumen cetak tersebut. Oleh karena itu, dibutuhkan suatu sistem yang dapat mengonversi citra dokumen cetak menjadi *string* agar data tidak perlu dimasukkan ke komputer secara manual. Saat ini, teknologi yang mampu mengidentifikasi huruf pada citra adalah *Òptical Character Recognition (OCR) engine* yang di dalamnya sudah diprogram untuk melakukan segmentasi, ekstraksi ciri, klasifikasi, *training*, dan rekognisi. Salah satu OCR *engine* yang memiliki akurasi yang paling tinggi (96,38%) dengan lama pemrosesan paling cepat (4,60 detik) adalah Tesseract. Namun, akurasi Tesseract bergantung kepada kualitas citra dan *noise*, sehingga diperlukan pengolahan citra tambahan. Oleh karena itu, pada penelitian ini dirancang suatu alat pemindai dokumen cetak menggunakan OCR Tesseract dengan tahapan pengolahan citra: *grayscaling, unsharp masking, Otsu thresholding*, dan *dilatation* dengan *library* OpenCV. Dari hasil pengujian terhadap jenis *font* Arial, Calibri, Times New Roman, Dot Matrix, dan *Fake Receipt* ukuran 16, diperoleh rata-rata persentase kesalahan sebesar 2,58% untuk mengenali kata, 3,5% untuk mengenali kata dalam suatu kalimat, 10,5% untuk mengenali kata dalam paragraf, dan 9,5% untuk mengenali kata dalam dokumen struk.

Kata Kunci: ***Optical Character Recognition*, OpenCV, pengolahan citra, pemindai dokumen cetak portabel, Tesseract**

Abstract

Printed documents are still the choice of several industries to store company data such as invoices, receipts, and other printed documents. This creates problems when it is necessary to digitize the data. Therefore, we need a system that can convert the image of a printed document into a string so that the data does not need to be entered into the computer manually. Currently, the technology that can identify letters in an image is Optical Character Recognition (OCR) engine which has been programmed to perform segmentation, feature extraction, classification, training, and recognition. One of the OCR engines that has the highest accuracy (96.38%) with the fastest processing time (4.60 seconds) is Tesseract. However, Tesseract's accuracy depends on image quality and noise, so additional image processing is required. Therefore, in this project, document scanner was designed using OCR Tesseract with image processing stages: grayscaling, unsharp masking, Otsu thresholding, and dilatation with the OpenCV library. The test on Arial, Calibri, Times New Roman, Dot Matrix, and Fake Receipt size 16 fonts, has yielded mean percentage error of 2.58% for recognizing words, 3.5% for recognizing words in a sentence, 10.5% to recognize words in paragraphs, and 9.5% to recognize words in receipt documents.

***Key Words*: Optical Character Recognition, OpenCV, image processing, portable scanner, Tesseract**

1. Pendahuluan

Dokumen cetak masih menjadi pilihan beberapa industri untuk menyimpan data-data perusahaan seperti kuitansi dan faktur pada industri perbankan. Hal tersebut menimbulkan masalah ketika data pada dokumen cetak perlu diproses di komputer. Data perlu dimasukkan satu-persatu ke komputer. Hal ini membutuhkan waktu yang lama dan menguras tenaga. Saat ini sudah ada teknologi pemindai (*scanner*), namun pemindai hanya mampu untuk menangkap citra dari dokumen cetak,

Gambar 1. Arsitektur Tesseract

kemudian citra tersebut perlu diubah menggunakan perangkat lunak tambahan jika ingin mendapatkan hasil pindaian dalam bentuk data *string* atau teks digital.

Optical Character Recognition (OCR) merupakan teknologi konversi citra dokumen digital, dokumen cetak, maupun tulisan tangan menjadi teks yang dapat diolah oleh mesin [1]. Teknologi OCR mengklasifikasikan karakter-karakter atau pola-pola pada citra untuk dicocokkan dengan huruf atau angka sehingga menghasilkan data yang mampu diproses oleh komputer (*data string*) [2]. Salah satu cara untuk mengaplikasikan OCR adalah dengan menggunakan OCR *engine*. Tesseract merupakan salah satu perangkat lunak OCR *engine open source* dengan tingkat akurasi sebesar 96,38% [3] dengan lama pemrosesan waktu rata-rata 4,60 detik per citra [2] untuk dokumen cetak. Akurasi dari Tesseract sangat berpengaruh kepada kualitas citra dan ukuran *font* dokumen cetak. Akurasi Tesseract turun drastis jika kualitas citra kurang dari 300 dpi. Yang menjadi faktor utama dalam penurunan akurasi Tesseract adalah *noise* pada citra dengan kualitas buruk. Oleh karena itu, diperlukan proses pengolahan citra untuk meningkatkan akurasi Tesseract.

Berdasarkan penelitian yang sudah dipaparkan, pada penelitian ini penulis merancang alat pemindai yang mampu mengenali teks pada dokumen cetak. Tujuannya adalah mengonversi teks pada dokumen cetak menjadi teks digital yang dapat diproses oleh komputer dengan tingkat akurasi lebih dari 90% (*error* 10%). Alat ini memanfaatkan teknologi OCR dengan metode Tesseract. Dokumen cetak ditangkap oleh kamera yang terintegrasi oleh Raspberry Pi 4 Model B. Sebelum citra dikenali dengan Tesseract, citra melalui tahapan pengolahan citra untuk meminimalisasi *noise*. Hasil dari alat ini adalah data string dari teks pada dokumen cetak.

2. Dasar Teori

2.1 OCR Tesseract

Tesseract adalah OCR *engine* yang dikembangkan oleh HP pada tahun 1984 – 1994. HP pertama kali memperkenalkan Tesseract kepada publik saat UNLV *Annual Test of* OCR *Accuracy*, tahun 1995. Sejak tahun 2005, HP merilis Tesseract kepada publik untuk menjadi perangkat lunak *opensource* yang bebas untuk digunakan dan algoritma dibaliknya dipublikasikan secara bebas [4]. Arsitektur Tesseract dijelaskan pada Gambar 1.

Proses pengenalan huruf pada citra diawali dengan *adaptive thresholding*. Selanjutnya adalah proses *page layout analysis*, atau analisis tata letak, di mana

Gambar 2. Hasil grayscale citra menggunakan algoritma Luminance

Tesseract mengenali antara area teks dengan area non-teks. Selanjutnya adalah mendeteksi garis dan kata. Proses tersebut untuk mendeteksi teks yang miring. Terakhir adalah proses pengenalan karakter pada citra yang dilakukan sebanyak dua kali. Proses kedua akan mendeteksi kembali karakter-karakter yang tidak dapat dikenali pada pengenalan yang pertama [5].

Pada citra yang dihasilkan oleh kamera atau pemindai belum sepenuhnya siap untuk dilakukan proses pengenalan huruf dengan OCR Tesseract karena adanya *noise* pada citra. Efek yang dihasilkan adalah akurasi OCR Tesseract menurun atau bahkan mesin tidak mampu mengenali huruf yang ada pada gambar. Hal ini dapat dihindari dengan tahapan pengolahan citra sebelum pengenalan huruf [6].

2.2 *Luminance Grayscale*

Teknik *grayscale* adalah teknik pengolahan citra dengan mengubah warna yang timbul pada citra, yaitu merah, hijau, dan biru, menjadi warna hitam dan putih. Alasan utama penggunaan citra *grayscale* adalah untuk menyederhanakan algoritma sistem dan mengurangi beban komputasi. Citra yang berwarna membutuhkan lebih banyak *training* sehingga sistem tidak efisien [7]. Algoritma yang terbaik untuk diimplementasikan dalam sistem OCR adalah algoritma Luminance [8]. Algoritma ini dinyatakan dalam persamaan (1) dan implementasi pada citra ditunjukkan pada Gambar 2.

$$Luminance = 0,299 \times R + 0,587 \times G + 0,114 \times B \quad (1)$$

di mana:
R = nilai warna merah
G = nilai warna hijau
B = nilai warna biru

2.3 Unsharp mask

Unsharp mask atau *edge enchancement filter* adalah teknik untuk mempertajam teks dan mengurangi *noise* [9]. Proses *unsharp mask* terdiri dari dua tahapan. Pertama adalah proses *filtering* dengan Gaussian *filtering* dengan tujuan mereduksi *noise*. Hasil filtering akan digabungkan dengan citra hasil *grayscaling* untuk menajamkan (*sharping*) karakter-karakter pada citra

Gambar 3. Hasil citra pada proses *unsharp mask*

Gambar 4. Hasil citra pada proses Otsu *Tresholding*

dokumen [7]. Persamaan matematis dari *unsharp mask* [7] ditunjukkan pada persamaan 2:

$$v = y + \gamma x - y \qquad \gamma > 0 \qquad (2)$$

di mana:
v = Hasil *unsharp mask*
y = citra hasil Gaussian *filter*
x = citra masukan
γ= *gain*

Tingkat ketajaman citra bergantung kepada *gain* yang diberikan. Gambar 3 menampilkan hasil dari proses *unsharp mask*.

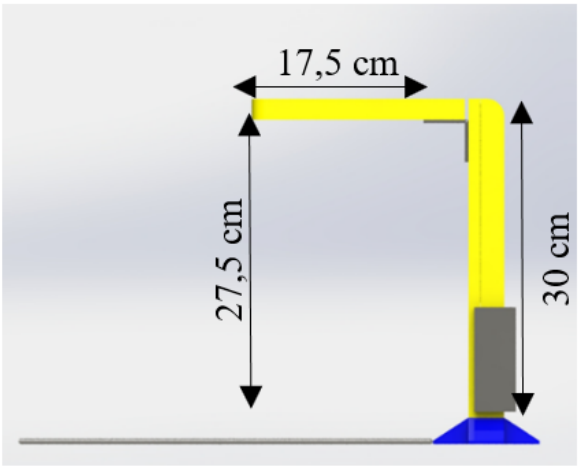
2.4 *Otsu Thresholding*

Thresholding adalah proses mengekstraksi objek yang ada pada citra dengan menerapkan nilai *threshold* pada tiap piksel sehingga setiap piksel dapat diklasifikasikan sebagai *foreground* dan *background*. Nilai *threshold* memberikan nilai biner pada tiap piksel di mana jika nilai piksel biner lebih dari nilai *threshold*, maka piksel tersebut akan diberi nilai satu dan sebaliknya nol [7]. Metode *thresholding* umum digunakan dalam pengenalan teks dan simbol, contohnya untuk memproses dokumen [10]. Salah satu algoritma yang banyak digunakan untuk pengolahan citra untuk mengenali karakter adalah *Otsu thresholding*. Kelebihan dari *Otsu thresholding* adalah algoritmanya sendiri yang mencari nilai threshold untuk setiap citranya [10]. Hasil proses dari *Otsu thresholding* ditunjukkan pada Gambar 4.

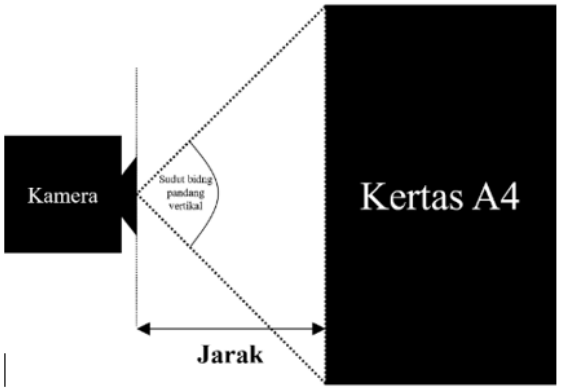
3. Perancangan dan Desain Sistem

3.1 Perancangan alat

Alat-alat yang digunakan dalam perancangan alat pemindai adalah Raspberry Pi 4 Modul B dengan spesifikasi RAM 4 GB LDDR4 dan penyimpanan sebesar 64 GB. Untuk menangkap citra dokumen digunakan modul kamera Raspberry Pi V 2.1 dengan



Gambar 5. Desain pemindai



Gambar 6. Skema perhitungan jarak objek terhadap kamera

resolusi kamera 8 Megapiksel dengan ukuran 1 piksel sebesar 1 µm [11]. Alat ini dirancang menggunakan perangkat lunak Solidworks dan dicetak dengan *3D printer*. Desain alat pemindai ditampilkan pada Gambar 5.

Perhitungan jarak antara objek dengan kamera bergantung kepada bidang pandang dari kamera yang digunakan. Skema perhitungan jarak objek dengan kamera ditampilkan pada Gambar 6. Variabel yang diketahui adalah bidang pandang horizontal dan vertikal kamera sebesar masing-masing 62,2º dan 48,8º, dan panjang bidang horizontal dan vertikal kertas A4 sebesar masing-masing 29,7 cm dan 21 cm.

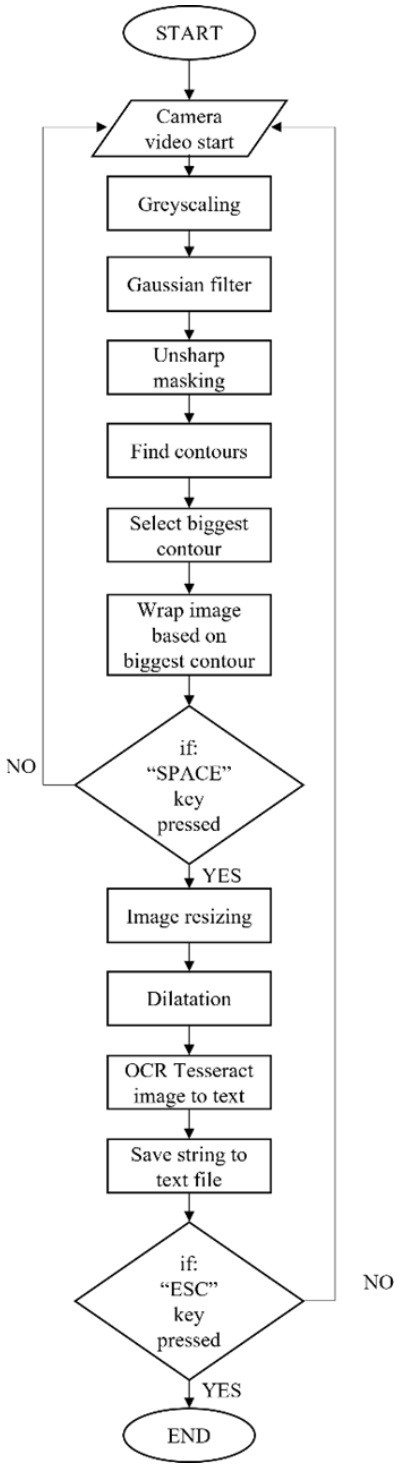
Dengan menerapkan teorema Pythagoras sederhana, dengan N adalah panjang objek dan B adalah bidang pandang kamera, jarak objek terhadap kamera dapat dirumuskan seperti ditunjukkan pada Persamaan 3.

$$Jarak = \frac{N}{\tan(\frac{B}{2})} \times \frac{1}{2} \quad (3)$$

Setelah dilakukan perhitungan, diperoleh jarak dengan *Field of View* (FoV) kamera horizontal sebesar 23,33 cm dan FoV vertikal sebesar 24,75 cm. Untuk memberikan ruang lebih pada citra maka diatur jarak antara kamera dengan objek sebesar 27,5 cm.

3.2 Desain Sistem

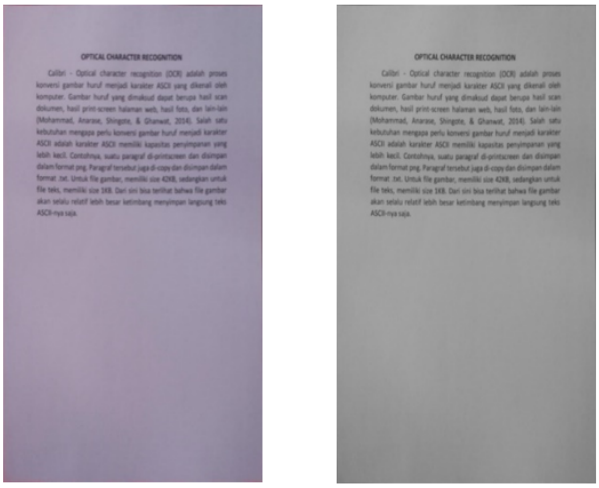
Sistem diprogram menggunakan bahasa pemograman Python 3.7 dengan OpenCV sebagai *library* pengolahan citra. Pertama-tama, program menyalakan kamera dan pada hasil tangkapan kamera dilakukan *grayscaling*. Citra hasil *grayscaling* diblur dengan Gaussian *filter* untuk mengurangi *noise* pada citra. Citra yang blur kembali diproses untuk memperjelas karakter-karakter pada citra dengan teknik *unsharp masking*. Selanjutnya dilakukan operasi *thresholding* dengan metode Otsu *thresholding* agar karakter pada dokumen lebih jelas untuk rekognisi karakter. Tahapan selanjutnya adalah program akan mencari kontur-kontur yang timbul pada citra dan berusaha mencari kontur yang terbesar yang merupakan kontur kertas itu sendiri. Setelah diperoleh koordinat kontur kertas, program akan memotong citra sesuai koordinat kontur. Tujuan dari tahapan ini adalah agar diperoleh hasil citra yang hanya berisi karakter yang akan direkognisi saja dan meluruskan dokumen jika saat pengambilan citra dokumen masih miring. Jika pengguna menekan tombol spasi, maka citra akan memasuki tahapan *resizing* untuk meningkatkan resolusi gambar saat rekognisi. Sebelum dilakukan tahapan rekognisi, citra melalui proses *dilatation*. Terakhir, hasil citra akan direkognisi dengan Tesseract dan hasilnya akan disimpan dalam suatu *file teks*. Gambar 7 menampilkan diagram alir dari pemrograman.



Gambar 7. Diagram alir sistem

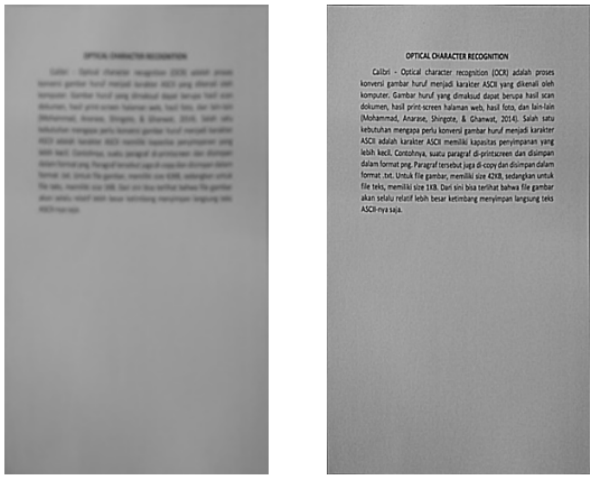
4. Pembahasan

Sistem pemindai portable menggunakan kamera digunakan untuk mengambil gambar seperti yang ditunjukkan pada Gambar 8 (a). Selanjutnya dilakukan pengolahan citra dimana pada tahap pertama dilakukan Luminance *grayscale* seperti yang ditampilkan pada Gambar 8 (b). Tahapan selanjutnya adalah pengolahan citra Gaussian *blurring* dengan menggunakan ukuran *kernel* (11,11) dengan nilai sigmaX sebesar 10. Nilai sigmaX menentukan nilai simpangan baku dari nilai-nilai pada *kernel*. Nilai *gain* 3 digunakan pada proses *unsharp masking*. Hasil dari citra Gaussian *blurring* dan unsharp masking ditampilkan pada Gambar 9 (a) dan (b). Pengolahan citra yang dilakukan setelah itu adalah Otsu *thresholding* seperti ditampilkan pada Gambar 10. Gambar ini merupakan citra yang kemudian



(a) (b)

Gambar 8. (a) Citra tangkapan kamera dan (b) Citra hasil Luminance *grayscaling*



(a) (b)

Gambar 9. (a) Citra hasil Gaussian *blurring* dan (b) Citra hasil *unsharp masking*

diolah menggunakan Tesseract untuk diubah menjadi karakter atau teks digital.

Kapabilitas sistem diuji melalui empat macam pengujian, yaitu uji identifikasi kata, uji identifikasi kata dalam kalimat, uji identifikasi kata dalam paragraf, dan uji identifikasi kata dalam dokumen struk. Semua pengujian menggunakan jenis *font* Arial, Calibri, Times New Roman, Dot Matrix, dan Fake Receipt dengan ukuran *font* 11, 12, 14, 16. Jenis dan ukuran *font-font* ini dipilih karena umum digunakan dalam dokumen cetak, terutama faktur dan struk. Pada setiap dokumen dilakukan penangkapan citra sebanyak enam kali. Performansi sistem dihitung berdasarkan persentase kesalahan (*error*) yang dihitung dengan persamaan:

$$Kesalahan\ (\%) = 100 - (\frac{B}{S} \times 100) \qquad (4)$$

Dalam hal ini, variabel B adalah jumlah kata yang berhasil diidentifkasi alat dan S adalah total semua kata pada dokumen. Persentase kesalahan pada setiap pengujian ditunjukkan masing-masing pada Gambar 11-14.

Berdasarkan persentase kesalahan hasil pengujian yang ditunjukkan pada Gambar 11-14, dapat dinilai bahwa sistem yang dihasilkan memiliki rentang kesalahan yang cukup tinggi (80–90%) untuk mengidentifikasi karakter-karakter yang memiliki *font* berukuran kecil. Rentang kesalahan semakin menurun dengan meningkatnya ukuran karakter hingga mencapai kesalahan 0% pada pengenalan karakter dengan ukuran *font* 16. Hal ini terjadi karena adanya peningkatan resolusi citra karakter seiring dengan peningkatan ukuran *font*. Sebagai contoh, pada citra huruf "B" dengan jenis *font* Arial, resolusi citra hanya sebesar 27×42 piksel untuk *font* berukuran 11; 32×45 piksel untuk ukuran *font* 12; 38×59 piksel untuk ukuran *font* 14; dan 40×63 piksel untuk ukuran *font* 16. Pada citra karakter "B" dengan ukuran *font* 11, terlihat bahwa garis atas karakter tidak terlalu jelas sehingga pada beberapa pengujian dikenali oleh Tesseract dengan huruf "K." Perbedaan resolusi citra huruf bagi masing-masing ukuran *font* 11, 12, 14 dan 16 bagi *font* Arial ini ditunjukkan pada Gambar 15.

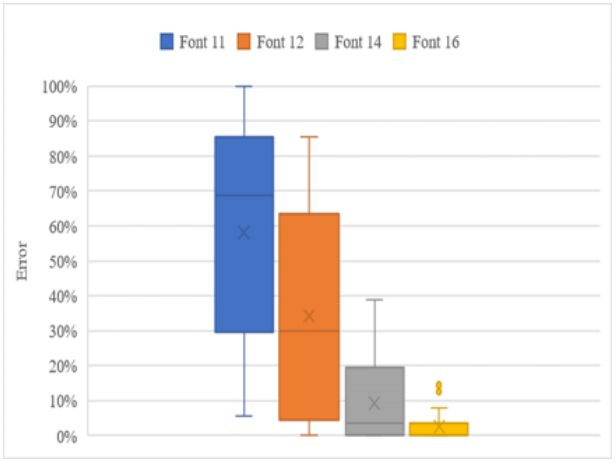
Selain itu, tingginya rentang kesalahan pada pengujian *font* berukuran kecil dapat juga terjadi karena kurangnya kemampuan kamera untuk menangkap citra secara detail dan kurangnya kualitas citra untuk pengenalan tulisan. Berdasarkan penelitian sebelumnya yang sudah dilakukan, akurasi OCR Tesseract akan maksimum jika kualitas citra lebih dari 300 dpi, namun citra yang dihasilkan oleh sistem ini hanya sebesar 75 dpi. Hal ini dapat disebabkan oleh beberapa faktor, antara lain kualitas kamera yang digunakan, atau kurang meratanya pencahayaan ketika pengambilan gambar dilakukan.

Jika dianalisis berdasarkan jenis *font* dengan kondisi ukuran *font* yang sama (ukuran *font* 16), akurasi sistem tidak berbeda secara signifikan. Pada pengujian pengenalan huruf dalam kata, akurasi terendah dan tertinggi adalah masing-masing pada jenis *font* Calibri (99,34%) dan Times New Roman (95,94%). Pada pengujian pengenalan kata dalam kalimat, akurasi tertinggi pada jenis *font* Fake Receipt (98,22%) dan terendah pada jenis *font* Arial (95,51%). Untuk pengujian pengenalan kata dalam paragraf, *font* dengan akurasi tertinggi adalah Fake Receipt (98,22%) dan terendah Arial (95,51%). Terakhir, untuk pengujian pengenalan kata dalam dokumen struk, akurasi tertinggi diperoleh untuk jenis *font* Times New Roman (92,98%)

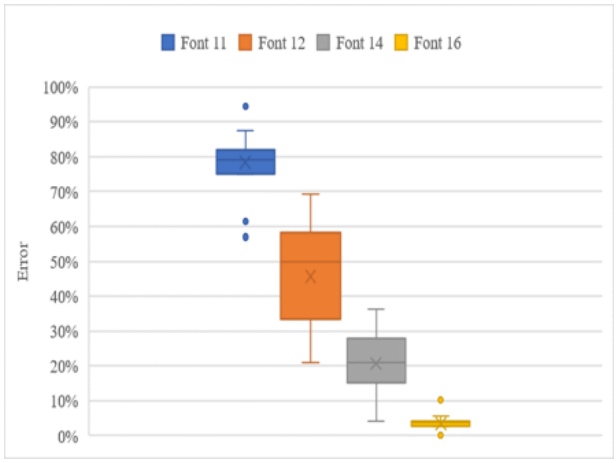
OPTICAL CHARACTER RECOGNITION

Calibri - Optical character recognition (OCR) adalah proses konversi gambar huruf menjadi karakter ASCII yang dikenali oleh komputer. Gambar huruf yang dimaksud dapat berupa hasil scan dokumen, hasil print-screen halaman web, hasil foto, dan lain-lain (Mohammad, Anarase, Shirgote, & Ghanwat, 2014). Salah satu kebutuhan mengapa perlu konversi gambar huruf menjadi karakter ASCII adalah karakter ASCII memiliki kapasitas penyimpanan yang lebih kecil. Contohnya, suatu paragraf di-printscreen dan disimpan dalam format png. Paragraf tersebut juga di-copy dan disimpan dalam format .txt. Untuk file gambar, memiliki size 42KB, sedangkan untuk file teks, memiliki size 1KB. Dari sini bisa terlihat bahwa file gambar akan selalu relatif lebih besar ketimbang menyimpan langsung teks ASCII-nya saja.

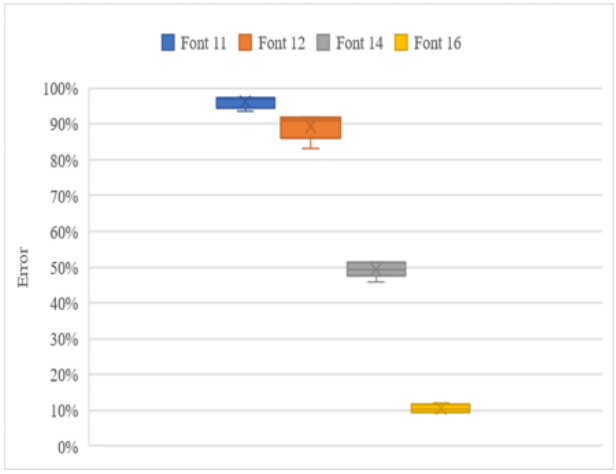
Gambar 10. Citra hasil Otsu thresholding



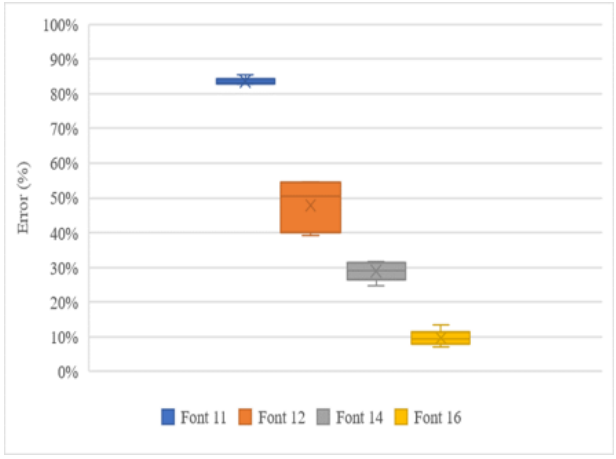
Gambar 11. Persentase kesalahan pada pengujian identifikasi kata



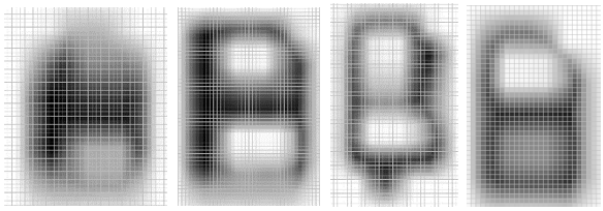
Gambar 12. Persentase kesalahan pada pengujian identifikasi kata dalam kalimat



Gambar 13. Persentase kesalahan pada pengujian identifikasi kata dalam paragraf



Gambar 14. Persentase kesalahan pada pengujian identifikasi kata dalam dokumen struk



Gambar 15. Resolusi citra font Arial "B" dari kiri ke kanan: 11, 12, 14, dan 16

dan terendah Calibri (86,73%). Berdasarkan hasil ini, dapat dinilai bahwa akurasi sistem pengenalan karakter menggunakan Tesseract tidak bergantung kepada jenis *font* namun bergantung pada ukuran *font* yang digunakan.

5. Kesimpulan

Sistem pemindai dokumen cetak portabel menggunakan Raspberry Pi 4 Model B dan kamera Raspberry Pi V2.1 menggunakan metode Tesseract berhasil mengidentifikasi tulisan pada dokumen cetak pada jenis *font* Arial, Calibri, Times New Roman, Dot Matrix, dan Fake Receipt dengan akurasi terbaik untuk ukuran *font* 16. Sistem ini juga berhasil mengubah citra menjadi teks digital dengan rata-rata persentase kesalahan sebesar 2,58% untuk mengenali kata, 3,5% untuk mengenali kata dalam suatu kalimat, 10,5% untuk mengenali kata dalam paragraf, dan 9,5% untuk mengenali kata dalam dokumen struk.

Daftar Pustaka

[1] A. Khormi, M. Alahmadi, and S. Haiduc, "A Study on the Accuracy of OCR Engines for Source Code Transcription from Programming Screencasts," in *Proceedings* - 2020 IEEE/ACM 17th *International Conference on Mining Software Repositories*, MSR 2020, Jun. 2020, pp. 65–75. doi: 10.1145/3379597.3387468.

[2] A. Chaudhuri, K. Mandaviya, P. Badelia, and S. K. Ghosh, "Optical character recognition systems," in *Studies in Fuzziness and Soft Computing*, vol. 352, Springer Verlag, 2017, pp. 9–41. doi: 10.1007/978-3-319-50252-6_2.

[3] J. A. Suitter, "Accuracy of Optical Character Recognition Software Google Tesseract," 2015. [Online]. Available: https://digitalcommons.usm.maine.edu /thinking _mattershttps: //digitalcommons.usm.maine.edu /thinking _matters/46

[4] R. W. Smith, "History of the Tesseract OCR engine: what worked and what didn't," in *Document Recognition and Retrieval XX*, Feb. 2013, vol. 8658, p. 865802. doi: 10.1117/12.2010051.

[5] Akhil S, "An overview of Tesseract OCR Engine A Seminar Report," 2016.

[6] N. Venkata Rao and D. Ascssastry, "Optical Character Recognition Technique Algorithms," *Journal of Theoretical and Applied Information Technology*, vol. 20, no. 2, 2016, [Online]. Available: www.jatit.org

[7] E. H. A. and R. N, "OCR Accuracy Improvement on Document Images Through a Novel Pre-Processing Approach," *Signal & Image Processing: An International Journal*, vol. 6, no. 4, pp. 01–18, Aug. 2015, doi: 10.5121/sipij.2015.6401.

[8] C. Kanan and G. W. Cottrell, "Color-to-grayscale: Does the method matter in image recognition?," *PLoS ONE*, vol. 7, no. 1, Jan. 2012, doi: 10.1371/journal.pone.0029740.

[9] G. Deng, "A generalized unsharp masking algorithm," *IEEE Transactions on Image Processing*, vol. 20, no. 5, pp. 1249–1261, May 2011, doi: 10.1109/TIP.2010.2092441.

[10] J. Yousefi, "Image Binarization using Otsu Thresholding Algorithm," 2011, doi: 10.13140/RG.2.1.4758.9284.

[11] Raspberry Pi Ltd, "Raspberry Pi Camera Documentation," *https://www.raspberrypi.com/documentation/ accessories/camera.htmlhardware-specification.*