DAFTAR TABEL

BAB I
PENDAHULUAN

1.1. Latar Belakang Masalah

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

Optical Character Recognition (OCR) merupakan teknologi pengkonversian citra dokmen digital, dokumen cetak, maupun tulisan tangan menjadi teks yang bisa diolah 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 dengan menggunakan OCR *engine*. Tesseract merupakan salah satu perangkat lunak OCR *engine* bersifat *open source* dengan tingkat akurasi sebesar 96,38% [3] dengan lama pemrosesan waktu rata-rata 4,60 detik per citra [1] untuk dokumen cetak. Jika dibandingkan dengan OCR *engine open source* lainnya, seperti GOCR, OCRAD, dan Cuneiform, Tesseract lebih unggul dari sisi akurasi dan lama pemrosesan[4]. Akurasi dari Tesseract sangat berpengaruh kepada kualitas dari 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. Maka dari itu diperlukan proses pengolahan citra untuk meningkatkan akurasi Tesseract.

Berdasarkan penelitian yang sudah dipaparkan, pada Tugas Akhir ini penulis merancang alat pemindai yang mampu mengenali teks pada dokumen cetak. Tujuannya adalah mengonversi teks yang pada dokumen cetak menjadi teks digital yang bisa diproses oleh komputer. Alat memanfaatkan teknlogi 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 meminimalisir *noise*. Hasil dari alat ini adalah data *string* dari teks pada dokumen cetak.

1.2. Rumusan Masalah

Berdasarkan masalah yang sudah dipaparkan, berikut adalah rumusannya:

1. Bagaimana merancang alat pemindai untuk mengenali tulisan pada dokumen cetak?
2. Metode apa yang digunakan untuk mengubah teks pada citra hasil pemindaian menjadi teks digital?

1.3. Tujuan dan Manfaat

Tujuan dari penelitian ini adalah:

1. Merancang dan mengimplementasikan alat pemindai untuk membaca tulisan cetak menggunakan Raspberry Pi 4 Model B dengan kamera Raspberry V 2.1
2. Merancang sistem pengenalan huruf menggunakan Tesseract dengan akurasi lebih dari 90% (error 10%).

Manfaat dari Tugas Akhir ini adalah:

1. Mendapatkan data dari teks cetak.
2. Mengotomatisasi pekerjaan manual dalam mengmasukkan data dari dokumen cetak.

1.4. Batasan Masalah

Hal-hal yang dibatasi dalam Tugas Akhir ini yaitu

1. Format file citra yang digunakan adalah .jpg
2. *Font* huruf yang diuji adalah Arial, Calibri, Times New Roman, Dot Matrix, dan Fake Receipt
3. Ukuran huruf yang diuji adalah 11, 12, 14, dan 16
4. Jarak antara kamera dengan teks sejauh 27,5 cm

5. Ukuran kertas dokumen sebesar A4
6. Alat diuji dengan kondisi ruangan terbuka di waktu siang hari

1.5. Metode Penelitian

Metode yang dilakukan dalam menyusun Tugas Akhir ini terdiri dari beberapa tahapan, diantaranya adalah:

1. Studi literatur

   Studi literatur dilakukan untuk mengkaji OCR Tesseract dan tahapan pengolahan citra yang dibutuhkan dalam meningkatkan kualitas citra.

2. Perancangan sistem

   Merancang alat dan algoritma yang digunakan.

3. Pengujian

   Menguji alat dengan menghitung akurasi pembacaan teks dan lama pemrosesan.

4. Analisis Hasil Pengujian

   Menganalisis hasil akurasi pemindaian alat.

5. Kesimpulan

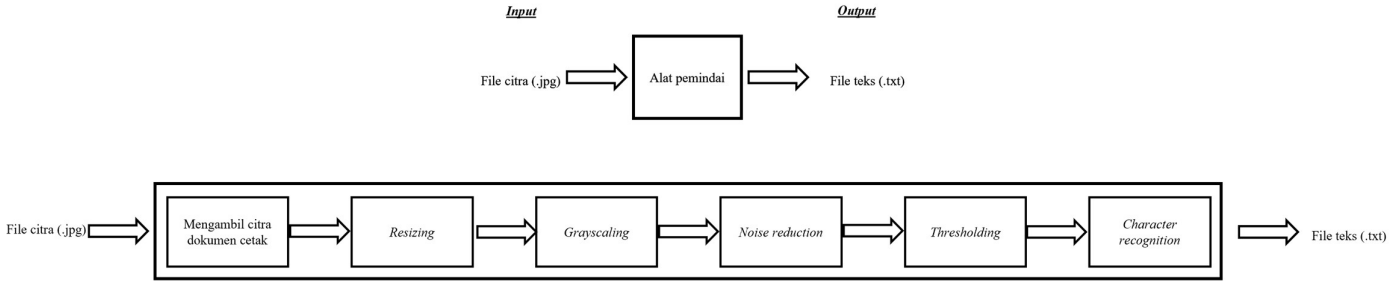
   Menyimpulkan hasil analisis dan menyusun saran yang bisa dikembangkan dari penelitian yang sudah dilakukan.

BAB II
TINJAUAN PUSTAKA

2.1. Desain Konsep Solusi

Pada tugas akhir ini peneliti merancang sebuah alat pemindai yang mana tidak hanya menangkap citra dari dokumen namun juga menekstrak teks pada dokumen dalam bentuk data *string*. Diagram fungsi dari alat ditampilkan pada Gambar 2.1. Penjelasan rinci dari diagram fungsi alat sebagai berikut:

1. Alat akan mengambil citra dokumen cetak yang ingin dipindai menggunakan kamera setelah itu file citra akan dikirimkan ke mikrokomputer.
2. Tahapan awal pengolahan citra pada citra adalah *resizing*. Teknik ini mengkonversi variasi warna pada citra menjadi hitam dan putih.
3. Setelah itu citra di-*grayscaling.* Teknik ini mengkonversi variasi warna pada citra menjadi hitam dan putih.
4. Selanjutnya tahapan *noise reduction* untuk mengurangi *noise* pada citra
5. Citra memasuki tahapan *thresholding* dengan tujuan mempertajam karakter (huruf dan angka) pada citra
6. Citra hasil pengolahan citra akan memasuki tahapn *character recognitoin* untuk membaca tulisan pada citra sehingga menghasilkan file teks.



Gambar 2.1 Diagram konsep solusi

2.2. Sistem OCR dengan Tesseract

Optical Character Recognition merupakan pengkonversian citra dokmen digital, dokumen cetak, maupun tulisan tangan menjadi teks yang bisa diolah mesin[1]. OCR dibutuhkan saat suatu informasi teks bisa dibaca oleh manusia dan mesin[5]. Saat ini ada banyak *software* dan algoritma untuk pengaplikasian OCR, salah satunya adalah OCR *engine*. OCR *engine* adalah *software* yang didalamnya sudah termasuk proses-proses pengenalan huruf. Salah satu OCR *engine* yang bisa digunakan secara *open source* adalah Tesseract. Tesseract menawarkan sistem OCR dengan tingkat akurasi yang tinggi, 6,4 % *error rate* pada resolusi citra 250 *pixel per inch* yang ditangkap menggunakan alat *scanner*[6]. Dalam tesis yang dilakukan oleh Martin Tomascheck dengan judul "*Evaluation of off-the-shelf OCR Technologies*" memaparkan bahwa diantara *open-source* OCR *engine* (GOCR, Tesseract, Cuneiform, dan Ocrad) Tesseract memiliki akurasi yang paling tinggi dan memiliki waktu pemrosesan yang paling cepat dibanding lainnya[4]. Pengujian dilakukan terhadap citra digital dari dokumen cetak dengan kulitas citra 400 dpi. Perbandingan dari OCR *engine* dalam mengenali teks pada dokumen cetak ditampilkan pada Tabel 2.1[4].

Tabel 2.1 Perbandingan OCR *engine*

OCR engine	Akurasi (%)
Tesseract	95
Cuneiform	90
GOCR	70
OCRAD	78

Tesseract memiliki angka akurasi yang tinggi pada dokumen yang dipindai dengan alat pemindai khusus, namun untuk citra yang ditangkap dengan kamera digital atau kamera ponsel akurasi Tesseract menurun, atau bahkan Tesseract tidak mampu membaca tulisan pada citra[7]. Masalah yang muncul biasanya adalah pencahayaan yang rendah, latar citra yang kompleks, dan *noise*[8]–[10]. Oleh sebab itu pengolahan citra sebelum proses OCR menjadi hal yang penting dalam peningkatan akurasi dari OCR.

Tabel 2.2 Penelitian Terkait

Judul	Nama Penulis (Tahun)	Alat pengambilan citra	Metode OCR dan Pengolahan citra	Kekurangan	Kesimpulan Penelitian	*Referensi*
Aplikasi Pendeteksi Plat Nomor Negara Indonesia Menggunakan OpenCV dan Tesseract OCR pada Arduino Studio	Sangsaka Wira Utama dan Apriani Kusmawardhani (2017)	Kamera ponsel	Tesseract dan grayscaling menggunakan OpenCV	Kurangnya akurasi OCR karena tidak ada pengolahan citra untuk minimalisir *noise*	Apliksi pengenal plat nomor kendaraan menggunakan Android dapat dilakukan dengan menggunakan OpenCV dan Tesseract OCR.	[9]
Pencarian Informasi Pajak Kendaraan Berdasarkan Plat Nomor Menggunakan Pustaka Tesseract dan OpenCV	Eko Suharyanto (2020)	Kamera ponsel	Tesseract dan penulis tidak menjelaskan metode pengolahan citra yang digunakan pada OpenCV	Penulis tidak menjelaskan proses pengolahan citra yang digunakan, tidak ada metode untuk meminimalisir *noise*	Akurasi sistem 95,5%.	[10]
Aplikasi Penerjemah Bahasa Inggris – Indonesia dengan Optical Character Recognition Berbasis Android	Anisa Eka Utami, Oky Dwi Nurhayati, dan Kurniawan Teguh Martono (2016)	Kamera ponsel	Tesseract dan penulis tidak menjelaskan metodes pengolahan citra	Kurangnya akurasi sistem OCR disebabkan oleh kurangnya cahaya, tidak sesuainya karakter, huruf terlalu kecil, tulisan miring, dan citra kurang fokus	Akurasi sistem 97,5%	[8]
OCR Accuracy Improvement on Document Images Through a Novel Pre-processing Approach	A. El Harraj dan N. Raissouni (2015)	Penulis tidak menjelaskan kamera yang digunakan	Tesseract dan metode pengolahan citra yag digunakan adalah meningkatkan *brigghtness*, konversi *grayscale, unsharp masking,* dan Otsu *thresholding*	Sistem masih gagal membaca beberapa karakter	Akurasi sistem setelah dilakukan pengolahan citra meningkat 2 – 7,17 %	[7]

Pada Tugas Akhir kali ini penulis merancang alat pemindai dengan menggunakan OCR Tesseract. Untuk meningkatkan akurasi alat tahapan pengolahan citra dilakukan terhadap citra. Fokus utama dari pengolahan citra kali ini adalah mengurangi *noise* serta meningkatkan akurasi dari OCR Tessract berdasarkan penelitian yang sudah dilakukan pada Tabel 2.1. Citra ditangkap menggunakan kamera Raspberry Pi. Tahapan pengolahan citra yang dilakukan adalah grayscaling*, unsharp mask* untuk meminimalisir *noise,* Otsu *thresholding*

untuk memperjelas huruf, dan yang terakhir adalah *dilation.* Untuk pemograman pengolahan citra menggunakan *library* OpenCV.

2.3. Optical Character Recognition

Optical Character Recognition adalah proses klasifikasi dari pola-pola yang muncul pada citra untuk nantinya dicocokkan dengan huruf dan angka. OCR termasuk dalam bidang *machine recognition techniques* lebih tepatnya *automatic identication,* yang mana sebuah proses pengenalan objek secara otomatis[2]. Data dimasukkan, diproses, dimanipulasi, dikeluarkan, dan disimpan oleh komputer tanpa campur tangan manusia. Pengenalan objek yang dilakukan dalam OCR adalah adalah pola-pola huruf dan angka yang muncul pada gambar. Awalnya huruf-huruf tersebut dimasukkan ke mesin. Mesin mempelajari huruf-huruf tersebut dan mengklasifikasikannya sesuai label yang diberikan. Proses tersebut lebih dikenal dengan proses *training*. Saat mesin melakukan pengenalan huruf, mesin akan mencocokkan huruf dengan data *training.* Mesin akan menghasilkan data prediksi hasil pencocokan[2].

Dalam proses pengenalan huruf dengan OCR hal pertama yang dilakukan adalah memindai dokumen yang ingin baca oleh OCR dan menghasilkan citra digital dari dokumen. Setelah itu citra melalui tahapan *image preprocessing,* dimana citra diolah agar kualitas citra menjadi lebih baik dan memudahkan proses pengenalan huruf. Selanjutnya huruf diekstrrak satu-persatu dan dilakukan pencocokan dengan data yang didapatkan pada proses *training*. Terakhir mesin akan menghasilkan kata-kata hasil dari proses pengenalan huruf dari teks[5].

2.4. Pengolahan citra

Citra yang dihasilkan oleh kamera atau pemindai belum sepenuhnya siap untuk melakukan proses pengenalan huruf karena munculnya *noise*. Efek yang dihasilkan adalah akurasi OCR menurun atau bahkan mesin tidak mampu mengenali huruf yang ada pada gambar. Hal ini bisa dihindari dengan tahapan pengolahan citra sebelum pengenalan huruf[11]. Dalam proses OCR pencahayaan yang tidak merata terhadap objek saat pengembilan citra tidak mempengaruhi

akurasi sistem. *Noise* bisa menjadi salah satu faktor dalam menurunnya akurasi sistem, namun untuk jumlah *noise* dibawah 10% Gaussian *noise* (citra diambil keadaan pencahayaan rendah) dan resolusi citra sebesar 600 dpi akurasi sistem tidak terpengaruh[12]. Maka dari itu pada penelitian kali ini berfokus pada mengurangi *noise* pada citra dengan teknik *unsharp mask* serta meningkatkan resolusi pada gambar dengan teknik *rescale*. Disamping itu teknik yang dilakukan pada citra adalah adalah *grayscale* dan *thresholding* dimana mampu meningkatkan akurasi sistem.

2.4.1. Luminance *Grayscale*

Teknik *grayscale* adalah merubah warna yang ada 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* data sehingga tidak efisien[7]. Teknik *grayscale* juga mampu meningkatkan akurasi dari OCR[13]. Banyak algoritma yang bisa digunakan dalam teknik *greyscle* sesuai kebutuhan dan keperluan tertentu. Algoritma yang terbaik untuk sistem OCR adalah algoritma Luminance[14]. Persamaan algoritma Luminance[7] dijelaskan pada persamaan 2.1. Gambar 2.2 menampilkan hasil dari teknik *grayscale* menggunakan algoritma Luminance. Algoritma Luminance menjadi algoritma standar yang digunakan *software* maupun *library* pengolahan citra, diantaranya MATLAB dan OpenCV.

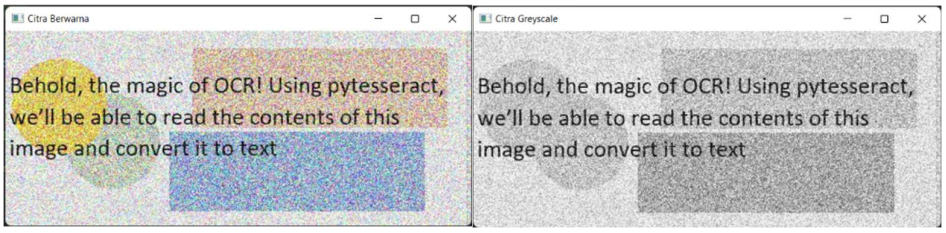
$$Luminance = 0{,}299 * R + 0{,}587 * G + 0{,}114 * B \quad (2.1)$$

dimana:

R = nilai warna merah

G = nilai warna hijau

B = nilai warna biru



Gambar 2.2 Hasil *grayscale* citra menggunakan algoritma Luminance

2.4.2. *Unsharp Masking*

Unsharp mask atau nama lainnya *edge enhancement filter* adalah teknik untuk mempertajam teks dan mengurangi *noise* pada citra. Citra akan melalui proses *filtering* menghasilkan citra yang buram. Setelah itu citra asli dikurangi dengan citra buram untuk mencari tepi yang ada pada citra, menghasilkan *unsharp mask*[7]. Proses *unsharp masking* terdiri dari dua tahapan, yang pertama adalah filtering dan menggabungkan citra *grayscaled* dengan citra hasil filtering. Tujuan dari proses *filtering* adalah untk mengurangi *noise* pada citra. Untuk proses *filter* citra teknik yang digunakan adalah Gaussian *filter*[15]. Tujuan dari penggunaan Gaussian filter adalah mengurangi *noise* pada citra dan mendeteksi tepi atau garis[16]. Persamaan matematis dari *unsharp mask* [15] adalah:

$$v = y + \gamma(x - y) \quad \gamma > 0 \tag{2.2}$$

dimana:

v = Hasil *unsharp mask*

y = citra hasil Gaussian *filter*

x = citra masukan

γ = *gain*

Besarnya ketajaman hasil *unsharp mask* bergantung kepada besarnya *gain* yang diberikan. Gambar 2.3 menampilkan hasil dari proses *unsharp mask*.

Gambar 2.3 Hasil unsharp mask

2.4.3. *Otsu Thresholding*

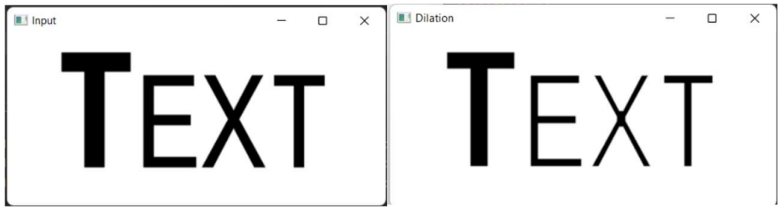
Thresholding adalah proses mengekstraksi objek yang ada pada citra dengan menerapkan nilai *threshold* pada tiap piksel sehingga setiap piksel dapat diklasifikan sebagai objek atau *background. Threshold* memberikan nilai biner pada tiap piksel dari citra *grayscale*. Jika nilai biner *greyscle* lebih dari nilai *threshold*, maka piksel tersebut akan diberi nilai biner satu dan sebaliknya diberi nilai nol[7]. Oleh karena itu tujuan utama dari *thresholding* adalah mempertegas objek yang ada pada citra. Gambar 2.4 menampilkan hasil dari teknik *thresholding.* Metode *thresholding* umum digunakan dalam pengenalan teks dan simbol, contohnya untuk memproses dokumen[17]. Salah satu algortima dalam *thresholding* adalah Otsu *threshold*. Metode Otsu menentukan nilai *threshold* berdasarkan data statistik dari citra.

Gambar 2.4 Hasil thresholding dengan metode Otsu

2.4.4. *Dilation*

Operasi morfologi adalah pemrosesan citra berdasarkan bentuknya. Operasi ini biasanya diterapkan pada citra biner. Nilai biner dari citra merepresentasikan dua keadaan: objek dan latar. Salah satu operasi yang sering digunakan adalah *dilation. Dilation* berfungsi untuk melebarkan latar

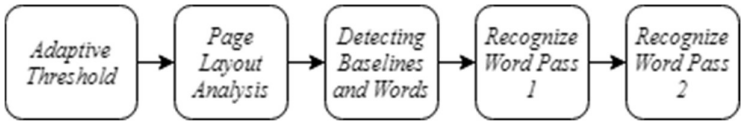
putih pada citra, mereduksi *noise* atau titik-titik gelap pada citra. Tujuan dari diterapkannya operasi *dilation* pada proses citra kali ini adalah mereduksi *noise* pada citra dan memisahkan huruf yang bersambung pada citra dengan melebarkan latar putih pada citra[18].



Gambar 2.5 Hasil proses *dilating*

2.5. 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 software *open source,* yang mana bebas untuk digunakan dan algoritma dibaliknya dipublikasikan secara bebas. Arsitektur dibalik Arsitektur Tesseract dijelaskan pada Gambar 2.6.



Gambar 2.6 Arsitektur Tesseract

a. *Adaptive threshold*

Metode threshold yang digunakan Tesseract adalah Otsu *threshold.* Proses ini bertujuan untuk mengekstraksi objek yang muncu

b. *Page Layout Analysis*

Page layout analysis, atau analisis tata letak, adalah proses pembagian antara area teks dan area non teks. Ini merupakan proses OCR pertama yang dilakukan Tesseract. Metode yang dilakukan adalah mendeteksi tabulasi pada teks[19].

c. Mendeteksi garis dan kata

Deteksi garis pada teks bertujuan untuk membaca teks yang miring. Dengan metode ini citra tidak perlu melalui proses *de-skew*, yang mana

menjaga kualitas dari citra[20]. Setelah itu Tesseract memotong tiap huruf yang ada pada teks.

Gambar 2.7 Deteksi kemiringan teks di Tesseract

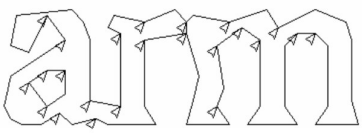
Gambar 2.8 Hasil pemotongan huruf dengan proporsi yang tepat

d. Pengenalan huruf

Tesseract mendeteksi apakah huruf-huruf yang sudah dipotong pada proses sebelumnya memiliki proporsi yang konstan pada tiap karakternya. Proses pengenalan huruf hanya berfokus pada huruf yang memiliki proporsi potongan yang tidak konstan. Biasanya ini terjadi pada teks yang memiliki huruf yang bersambung.

e. Klasifikasi karakter

Klasifikasi huruf pada Tesseract dilakukan sebanyak dua kali. Pertama Tesseract melakukan *static classification. Static classification* menggunakan *polygonal approximation* dari garis tepi huruf. Dalam proses *training* vektor 4 dimensi (garis x, garis y, arah, dan panjang) dati setiap elemen dari *polygonal approximation.* Dalam melakukan pengklasifikasian dari elemen polygonal approximation, Tesseract menggunakan algoritma *K-Nearest Neighbours* (KNN). Kedua, klasifikasi yang dilakukan adalah *adaptive classification.* Fitur dan klasifikasi yang digunakan pada tidak berbeda dengan *static calssification,* namun data *training* yang digunakan berasal dari data keluaran *static classification.* Proses *adaptive classification* bertujuan untuk mengoreksi jika ada kesalahan yang terjadi pada tahap klasifikasi yang pertama[19].



Gambar 2.9 Titik yang akan dipotong pada huruf yang bersambung