BAB II
LANDASAN TEORI

2.1 Studi Pustaka

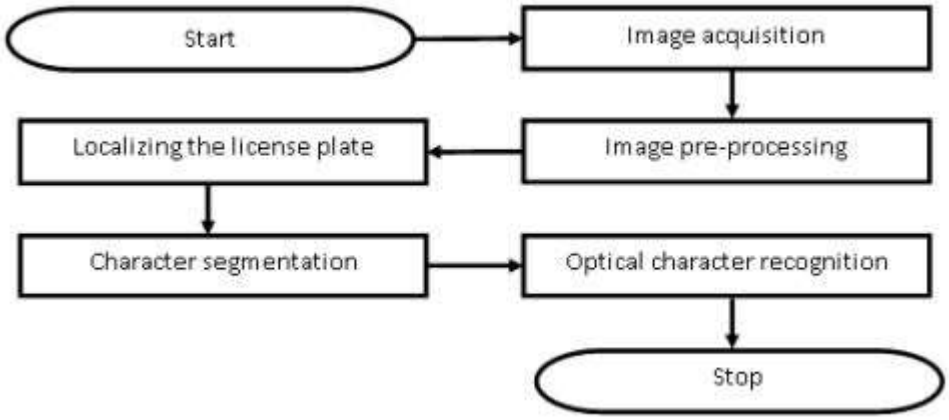
Penelitian yang dilakukan sebelumnya di Indonesia menggunakan Tesseract dalam melakukan proses pengenalan plat nomor kendaraan. Pada penelitian ini juga didapatkan hasil rata-rata akurasi dari Tesseract yang diambil dari 5 sampel plat nomor kendaraan. Pada penelitian yang dilakukan oleh A. Michael, A. R. Dayat, N. A. Banyal[1], pengambilan gambar dilakukan pada jarak 40 cm - 60 cm, dan penelitian yang dilakukan oleh T. S. Gunawan, A. Mutholib, M. Kartiwi[2] yang mengambil gambar hanya gambar plat kendaraan saja. Berdasarkan penelitian yang dilakukan oleh penulis, mengambil jarak 50 cm, 100 cm, dan 150 cm. Pengambilan jarak tersebut untuk melihat jarak akurasi dari Google Vision dan Tesseract dari jarak yang berbeda yang dilakukan oleh peneliti sebelumnya. Pada penelitian yang dilakukan oleh T. S. Gunawan, A. Mutholib, M. Kartiwi[2], didapatkan hasil rata-rata 71% dari 5 sampel yang ada. Pada penelitian tersebut, pengambilan gambar plat nomor kendaraan dilakukan secara langsung dan diambil dari jarak dekat, sehingga hanya mengambil gambar plat nomor kendaraan saja tanpa mengambil gambar kendaraannya.

Pengenalan plat nomor kendaraan otomatis merupakan salah satu teknik penting sebagai bagian dari sistem transportasi cerdas yang dapat digunakan untuk mengidentifikasi kendaraan hanya dengan memahami plat nomor. Sistem pengenalan plat kendaraan secara umum terdiri atas 3 (tiga) tahapan penting dalam yaitu deteksi plat nomor, segmentasi karakter, dan pengenalan karakter[1].

2.1.1 Desain Pengenalan Plat Nomor Kendaraan Otomatis di Platform Smartphone Android

Pengenalan plat nomor otomatis dikenal dengan banyak istilah lain, yaitu *Automatic Vehicle Identification* (AVI), *Automatic License Plate Recognition* (ALPR), *Car Plate Recognition* (CPR) dan *License Plate Recognition* (LPR)[2]. Seluruh persyaratan mengacu pada sistem pengenalan yang menggunakan kamera untuk membaca karakter pada plat nomor kendaraan.

Polisi Inggris cabang pengembangan ilmiah tercatat sebagai penemu sistem *Automatic Number-Plate Recognition* (ANPR) pada tahun 1976. Sistem *Automatic Number-Plate Recognition* memiliki kemampuan untuk mengekstrak dan mengenali karakter plat dari suatu gambar[3]. Terdiri dari kamera untuk menangkap nomor plat, mendeteksi lokasi karakter pada gambar dan kutipan karakter untuk menyimpulkan pixel ke numerik karakter yang mudah dibaca.



Gambar 2.1 Gambar flowchart dari sistem ANPR[2]

Gambar 2.1 merupakan alur kerja dari penelitian yang dilakukan. Tahap pertama adalah *Image Acquisition*. *Image Acquisition* adalah tahap penting sistem *Automatic Number-Plate Recognition* (ANPR) karena menyediakan input data untuk proses selanjutnya. Ada banyak cara untuk memperoleh gambar lisensi plat nomor kendaraan. Pada tahap ini dilakukan untuk mengambil data gambar plat nomor kendaraan dan menyimpan data tersebut.

Selanjutnya adalah tahap *Image pre-processing*. Pada tahap ini dilakukan peningkatkan kualitas dari gambar yang diambil sebelumnya. Gambar yang diambil kemudian diubah menjadi bentuk *gray scale* yang kemudian dilakukan *filtering image* dengan menggunakan *non-linear medium filter*. Tahap ini juga dilakukan untuk mempermudah dalam melakukan proses berikutnya yaitu melakukan lokalisasi plat nomor kendaraan.

Tahap selanjutnya adalah lokalisasi plat nomor kendaraan. Pada tahap ini dilakukan proses untuk mendeteksi wilayah persegi panjang plat nomor dalam

gambar yang diambil. Dalam deskripsi manusia, plat nomor adalah sesuatu yang berbentuk kecil dari plastik, maupun plat logam disisipkan ke kendaraan untuk tujuan identifikasi yang resmi. Namun, mesin tidak mengerti deskripsi ini. Oleh karena itu, kita perlu menemukan deskripsi alternatif dari plat nomor yang berdasarkan deskriptor yang akan dapat dimengerti untuk mesin.

Pada tahap *Character Segmentation* dilakukan deteksi teks yang ada pada area plat nomor kendaraan yang sudah dilokalikasasi. Tujuan dari tahap ini sendiri adalah untuk mendapatkansegmentasi dari setiap karakter yang ada kemudian nantinya akan dibutuhkan untuk proses selanjutnya yaitu pengenalan karakter. Pada tahap ini jg dilakukan deteksi karakter yang berjenis *joined character* dan juga *broke character*. *Joined character* adalah karakter yang segmentasinya menempel pada karakter sebelahnya ataupun sebelumnya. Dan *broke character* adalah karakter yang segmentasi nya rusak sehingga susah untuk dikenali.

Tahap terakhir yang adalah pengenalan karakter. Pengenalan karakter sendiri dilakukan setelah mendapatkan segmentasi karakter pada tahap sebelumnya, dan kemudian setelah dilakukan pengenalan karakter dari segmentasi karakter tersebut, karakter tersebut akan diekstrasi menjadi hasil output akhir yang berbentuk teks.

No	Image Capture	Recognition Result	Recognition Percentage
1		B3289BJH	87.5 %
2		D328SHJH	50 %
3		B8692LH	71.43 %
4		B85926N	71.43 %
05		HS929SIF	75 %
		Average:	71%

Gambar 2.2 Gambar hasil akurasi dari Tesseract[2]

Gambar 2.2 menunjukan hasil dari akurasi pengenalan plat nomor kendaraan yang dilakukan oleh Tesseract pada penelitian yang lalu. Pada penelitian tersebut didapatkan hasil rata-rata 71% dari 5 sampel yang ada.

2.2 Image Processing

Dalam *imaging science*, *image processing* adalah pengolahan gambar menggunakan operasi matematika dengan menggunakan bentuk pemrosesan sinyal input yang adalah gambar, serangkaian gambar atau video, seperti foto atau video bingkai; hasil pengolahan gambar dapat berupa gambar atau serangkaian karakteristik atau parameter yang terkait dengan gambar[4]. Teknik *image processing* yang sering dipakai melibatkan gambar yang dianggap sebagai sinyal dua dimensi dan menerapkan teknik pemrosesan sinyal standar untuk gambar tersebut. Gambar juga diproses sebagai tiga dimensi sinyal dengan dimensi ketiga waktu atau z-sumbu.

Dua jenis metode yang digunakan untuk *image processing* adalah Analog dan Digital Image Processing. Analog atau visual teknik dari *image processing* dapat digunakan untuk *hardcopy* seperti cetakan dan foto-foto[5]. Analisis gambar menggunakan berbagai macam dasar-dasar interpretasi saat menggunakan teknik-teknik visual.

Dalam pemrosesannya, *image processing* memiliki beberapa tujuan, diantaranya:

1. *Visualization* – Melakukan observasi terhadap objek yang tidak terlihat.
2. *Image sharpening and restoration* – Untuk membuat gambar tampak labih bagus.
3. *Image retrieval* – Mencari dan mendapatkan gambar objek yang diinginkan.
4. *Measurement of pattern* – Melakukan pengukuran terhadap berbagai macam objek yang ada didalamnya.
5. *Image Recognition* – Melakukan pengenalan objek yang ada didalam gambar.

Dalam penelitian yang dilakukan pada pengenalan otomatis plat nomor kendaraan, *image processing* berperan dalam pemrosesan *Optical Character*

Recognition yang dilakukan. Dimana yang didalamnya terdapat *image scaling*, *image grayscaling*, dan *character recognition*.

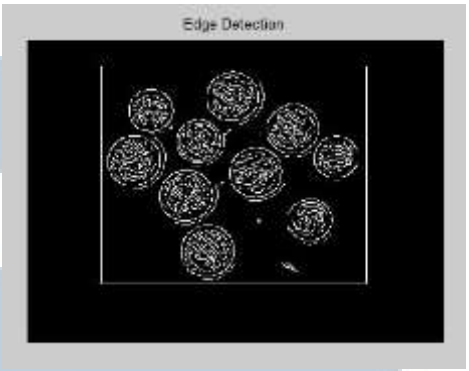
2.3 Edge Detection

Edge detection adalah teknik pengolahan gambar untuk menemukan batas-batas objek dalam gambar. *Edge detection* bekerja dengan mendeteksi diskontinuitas pada kecerahan sebuah gambar. *Edge detection* digunakan untuk segmentasi gambar dan ekstrasi data pada bidang pengolahan gambar, *computer vision* dan *machine vision*[6].

Physical edges memberikan informasi visual yang penting karena mereka berkorespondensi dengan diskontinuitas dalamsifat fisik, photometrical dan geometris sebuah objek. Prinsip dari *physical edges* berkorespondensi dengan variasi signifikan dalam reflektansi, iluminasi, orientasi, dan kedalaman permukaan sebuah objek. Karena gambar intensitas sering sebanding dengan adegan radiance, *physical edges* ditunjukkan dalam gambar oleh perubahan dalam fungsi intensitas[7]. Yang paling umum dari jenis variasi intensitas gambar merupakan tahapan, garis dan persimpangan.

Edge Detection mengacu pada proses mengidentifikasi dan menemukan tajam diskontinuitas pada sebuah gambar. Diskontinuitas adalah perubahan mendadak dalam pixel intensitas yang menandai batas-batas obyek dalam adegan. Metode klasik *edge detection* melibatkan *convolving* gambar dengan operator (filter 2-D), yang dibangun untuk peka terhadap besarnya gradien dalam gambar ketika mengembalikan nilai-nilai nol di seragam daerah[8].

Dalam tahapan *edge detection*, gambar juga di-*grayscaling* untuk dapat mempermudah deteksi terhadap garis-garis tepi yang ada pada objek dalam gambar. Setelah itu objek-objek yang ada didalam gambar akan diproses untuk mencari tepi garis dari setiap objek yang ada didalam sebuah gambar.



Gambar 2.3 Original

Gambar 2.4 Edge Detection

Contoh penggunaan *Edge Detection*

2.4 *Optical Character Recognition* (OCR)

Optical Character Recognition (OCR) adalah pengenalan karakter teks cetak atau tertulis oleh komputer. *Optical Character Recognition* melibatkan photoscanning teks karakter, analisis gambar, dan menterjemahan dari gambar karakter ke kode karakter, seperti ASCII, umumnya digunakan dalam pengolahan data. *Optical Character Recognition* dapat ditelusuri ke teknologi melibatkan telegrafi dan menciptakan perangkat membaca untuk orang buta. Pada tahun 1914, Emanuel Goldberg mengembangkan sebuah mesin yang membaca karakter dan dikonversi mereka ke dalam kode standar telegraf[9]. Secara bersamaan, Edmund Fournier d'Albe mengembangkan Optophone, scanner genggam bahwa ketika dipindahkan ke halaman yang dicetak, menghasilkan nada yang berpadanan dengan spesifik huruf atau karakter. Di akhir 1920-an dan 1930-an Emanuel Goldberg mengembangkan apa yang disebutnya "Mesin statistik" untuk mencari arsip mikrofilm menggunakan sistem pengakuan optik kode. Pada tahun 1931, dia diberi nomor Paten Amerika Serikat 1,838,389 untuk penemuan. Paten diakuisisi oleh IBM.

Pada *Optical Character Recognition*, terdapat proses dimana melakukan *grayscaling* pada gambar pada saat pengenalan karakter dilakukan. *Grayscaling* sendiri merupakan teknik mengubah suatu gambar menjadi hitam-putih, dimana teknik ini digunakan untuk memisahkan background dengan karakter. Dalam *Optical Character Recognition* juga terdapat beberapa teknik pendukung dalam melakukan image processing yang dilakukan yaitu, *line and word detection*, dan *normalize*.

2.5 Google Vision

Google Vision API merupakan suatu set *Application Programming Interface* yang dikembangan oleh Google yang dimana dapat berkomunikasi dengan Google Services yang tersedia. Google Vision sendiri terdiri dari banyak fungsi untuk melakukan pemrosesan gambar, mulai dari facial detection, landmark detection, label detection, logo detection, safe search detection, dan optical character recognition. Terdapat *client library* dalam berbagai bahasa yang memungkinkan pengembang untuk menggunakan Google API dari dalam kode mereka, termasuk Jawa, JavaScript, .NET, Objective-C, PHP dan Python.

Google Cloud Vision memungkinkan developers untuk memahami isi dari suatu gambar dengan *encapsulating powerful machine learning models* REST API yang mudah digunakan[10]. Google Cloud Vision API dapat dengan cepat mengklasifikasikan gambar ke ribuan kategori (misalnya, "perahu layar", "singa", "Eiffel Tower"), yang mendeteksi objek individu dan wajah dalam gambar, dan menemukan serta membaca kata-kata yang terkandung dalam gambar.

Google Vision mendeteksi dan melakukan ekstraksi karakter dari setiap gambar. Misalnya, foto mungkin berisi tanda jalan tanda atau lalu lintas. Google Vision menerapkan teknik neural network untuk melakukan pembelajaran dan perbandingan huruf. Google Vision akan melakukan deteksi teks yang ada dan setelah itu melakukan pengenalan teks yang terdapat pada tahap google vision & translation API.

Gambar 2.5 Contoh text detection pada google vision[9]

OCR Google Vision sendiri dapat melakukan ekstrasi karakter yang ada pada sebuah gambar. Algoritma dari Google Vision sendiri tidak mendetail karena dari Google sendiri tidak memberikan izin untuk penjelasan algoritma yang mendetail.

Gambar 2.6 Contoh gambar text detection pada sebuah dokumen[9]

2.6 Tesseract

Tesseract merupakan salah satu *Optical Character Recognition* open source. Tesseract pada awalnya dikembangkan sebagai perangkat lunak yang dikembangkan di Hewlett-Packard antara tahun 1985 hingga 1995. Setelah sepuluh tahun tanpa perkembangan apapun yang terjadi, Hewlett Packard dan UNLV merilis Tesseract sebagai sumber terbuka pada tahun 2005[11]. Tesseract saat ini sedang dikembangkan oleh Google dan dirilis di bawah Lisensi Apache, Version 2.0. Tesseract memiliki dukungan unicode (UTF-8), dan dapat mengenali lebih dari 100 bahasa. Tesseract dimulai sebagai proyek penelitian PhD di HP Labs, Bristol, dan memperoleh momentum sebagai perangkat lunak dan/atau hardware add-on untuk produk HP flatbed scanner.

Tesseract berada di atas tiga OCR mesin dalam hal ketepatan karakter pada 1995. Tesseract tersedia untuk Linux, Windows dan Mac OS X. Namun, karena sumber daya yang terbatas itu hanya ketat diuji oleh pengembang di bawah Windows dan Ubuntu.

Tesseract hingga tesseract versi 2 hanya bisa menerima gambar TIFF sederhana satu-kolom teks sebagai input. Versi awal ini tidak termasuk tata letak analisis, dan sehingga memasukkan multi-berbentuk kolom teks, Gambar, atau persamaan menghasilkan output yang tidak sempurna. Sejak versi 3.00 Tesseract telah mendukung format output teks, informasi posisi hOCR dan analisis tata letak halaman. Dukungan untuk beberapa format gambar baru telah ditambahkan menggunakan perpustakaan Leptonica. Tesseract dapat mendeteksi apakah teks monospace atau proporsional spasi.

Tesseract mungkin adalah mesin *Optical Character Recognition* pertama yang mampu menangani teks putih-hitam dengan mudah[12]. Pada tahap ini, outlines dikumpulkan bersama, dengan cara nesting, menjadi blobs. Blobs disusun ke dalam baris teks, dan garis-garis dan daerah region dianalisa untuk *fixed pitch*-nya atau proporsional teks. Baris teks yang dibagi dalam kata-kata yang berbeda menurut jenis spasi karakter. *Fixed Pitch* teks kemudian dipotong sesuai dengan jarak karakternya. Proportional teks dibagi per-kata menggunakan *definte spaces* dan *fuzzy spaces*.

Pengenalan karakter dilakukan dengan proses two-pass. Dalam pass yang pertama, dilakukan untuk mengenali setiap kata pada bergiliran. Setiap kata yang tidak sesuai dilewatkan ke classifier adaptif sebagai training data. Classifier adaptif kemudian mendapat kesempatan untuk lebih akurat mengenali teks setelah pengenalan pertama dilakukan. Karena classifier adaptif mungkin telah belajar sesuatu yang berguna untuk membuat kontribusi yang dekat bagian awal pengenalan, pass kedua dijalankan kembali dari awal halaman pengenalan, dimana kata-kata yang yang tidak dikenali cukup baik akan diperbaiki.

Berikut adalah tahapan yang dilakukan oleh tesseract dalam melakukan pengenalan karakter.

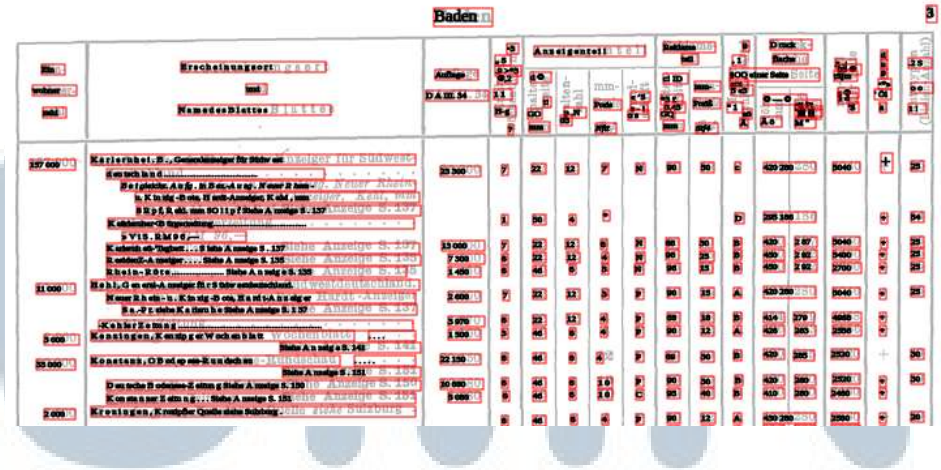
2.6.1 Pencarian Garis dan Kata

Pencarian baris dan kata merupakan tahapan awal dari proses yang dilakukan oleh Tesseract untuk melakukan pengenalan karakter. Pencarian baris dan kata sendiri terdiri dari beberapa tahapan proses, yaitu pencarian garis, *baseline fitting*, dan *fixed pitch detection and chopping*.

2.6.1.1 Pencarian Garis

Algoritma pencarian garis adalah salah satu dari beberapa bagian dari tesseract yang sebelumnya telah diterbitkan. Algoritma pencarian garis tersebut dirancang sehingga halaman yang miring dapat dikenali tanpa harus melakukan perataan sudut kemiringan, hal tersebut menyelamatkan hilangnya kualitas gambar. Bagian penting dari proses adalah penyaringan *blob* dan garis konstruksi[13].

Langkah terakhir dari proses pembuatan garis adalah menyatukan *blobs* yang *overlap* dengan setidaknya setengah horizontal, dan menempatkan tanda-tanda *diacritical* bersama dengan *base* yang benar.



Gambar 2.7 Contoh gambar line finding[12]

Gambar 2.7 adalah contoh dari pencarian garis dan karakter yang dilakukan oleh Tesseract, yang dimana garis merah yang ada menandakan deteksi karakter dan melakukan penempatan garis sebagai edges dari kalimat karakter tersebut.

2.6.1.2 Baseline Fitting

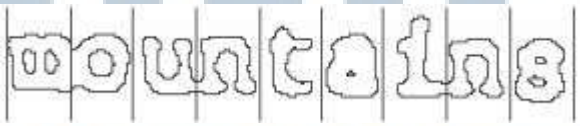
Setelah baris teks telah ditemukan, baselines kemudian dipasang dengan lebih tepat menggunakan spline kuadrat. Ini adalah salah satu yang digunakan untuk sistem *Optical Character Recognition* (OCR), dan membuat Tesseract dapat menangani halaman yang ememiliki baselines yang melengkung, yang adalah sebuah hal yang umum dalam proses *scanning*[14].

Baselines kemudian dipasangkan dengan partisi dari blobs yang menjadi kelompok dengan perpindahan cukup terus-menerus untuk original baseline yang lurus. Sebuah *quadratic spline* dipasangkan untuk partisi terpadat, (diasumsikan menjadi baseline) oleh squares yang cocok. *Quadratic spline* memiliki keuntungan bahwa perhitungan ini cukup stabil, tetapi kerugian yang dapat diskontinuitas timbul ketika beberapa spline segmen diperlukan.

Gambar 2.8 Contoh baseline fitting[12]

2.6.1.3 Fixed Pitch Detection and Chopping

Tesseract melakukan pengetesan baris teks yang sudah dilakukan untuk menentukan apakah mereka adalah sudah fixed pitch atau tidak. Dimana menemukan teks fixed pitch, Tesseract memotong kata-kata menjadi karakter yang menggunakan pitch, dan menonaktifkan potongan-potongan teks dan associator pada kata-kata untuk langkah pengenalan kata.



Gambar 2.9 Contoh gambar fixed pitch detection and chopping[12]

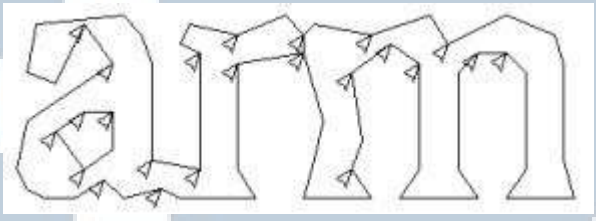
2.6.2 Pengenalan Kata

Bagian dari proses pengenalan untuk setiap character recognition engine adalah untuk meng-indentifikasi bagaimana sebuah kata seharus nya disegmentasikan menjadi karakter-karakter. Awal dari output segmentasi dari

pencarian baris adalah melakukan klasifikasi terlebih dahulu. Dan setelah itu baru melakukan pengenalan kata.

2.6.2.1 Chopping Joined Characters

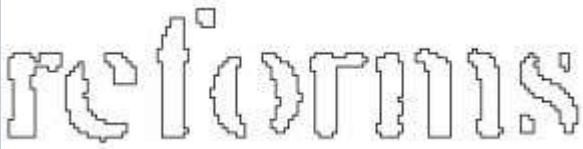
Tesseract berupaya untuk meningkatkan hasil dengan memotong blobs dengan pendekatan yang terburuk dari *classifier* karakter. kandidat *chop point* yang dipotong ditemukan dari simpul cekung dari pendekatan poligonal *outline*, dan mungkin memiliki cekung baik vertex berlawanan maupun segmen garis[15]. Mungkin diperlukan hingga 3 pasang *chop point* untuk berhasil memisahkan karakter dari ASCII set.



Gambar 2.10 Contoh kandidat chop point dan chop[12]

2.6.2.2 Broken Characters

Ketika potensial chop telah habis, jika kata yang telah di chop masih tidak cukup baik, maka kata tersebut akan dimasukan pada associator. Associator melakukan pencarian (terbaik pertama) dari grafik segmentasi yang mungkin kombinasi maksimal dari chopped blobs menjadi kandidat suatu karakter.



Gambar 2.11 Contoh broken characters[12]

2.6.3 Training data

Ketika classifier mampu mengenali karakter yang rusak dengan mudah, classifier tidak terlatih pada karakter rusak. Pada kenyataannya, classifier dilatih pada hanya 20 sampel 94 karakter dari 8 font di satu ukuran, tetapi dengan 4 atribut (normal, bold, italic, bold italic), membuat total 60160 platihan sampel[16]. Training data digunakan untuk melatih dan mencocokan setiap

karakter yang ada. Setelah karakter diterima dari proses sebelumnya, karakter yang ada kemudian dicocokan dengan karakter-karakter yang ada dalam *database Training Data* dan kemudian memberikan hasil karakter yang terdekat dengan pengenalan yang dilakukan.

2.6.4 Adaptive Classifier

Tesseract tidak membuat sebuah template classifier, tetapi menggunakan fitur yang sama dan classifier sebagai statis classifier. Hanya perbedaan yang signifikan antara classifier statis dan classifier yang adaptif, terlepas dari data platihan, adalah bahwa classifier adaptif menggunakan isotropik dasar/x-height normalisasi, sedangkan classifier statis menormalkan karakter oleh centroid (saat pertama) untuk saat-saat posisi dan kedua untuk ukuran normalisasi anisotropik.

Dasar/x-height normalisasi memudahkan untuk membedakan huruf besar dan huruf kecil karakter serta sebagai meningkatkan imunitas terhadap noise. Manfaat utama dari normalisasi pada karakter adalah penghapusan rasio aspek font dan beberapa derajat lebar font stroke. Hal ini juga membuat pengenalan sub dan superscripts lebih sederhana, tetapi membutuhkan fitur classifier untuk membedakan beberapa huruf besar dan kecil karakter.

Gambar 2.12 Contoh gambar baseline and normalized letters[11]