PENGENALAN POLA

Muhammad Ilham Rizqyawan - 23214340

Hendy Irawan - 23214344

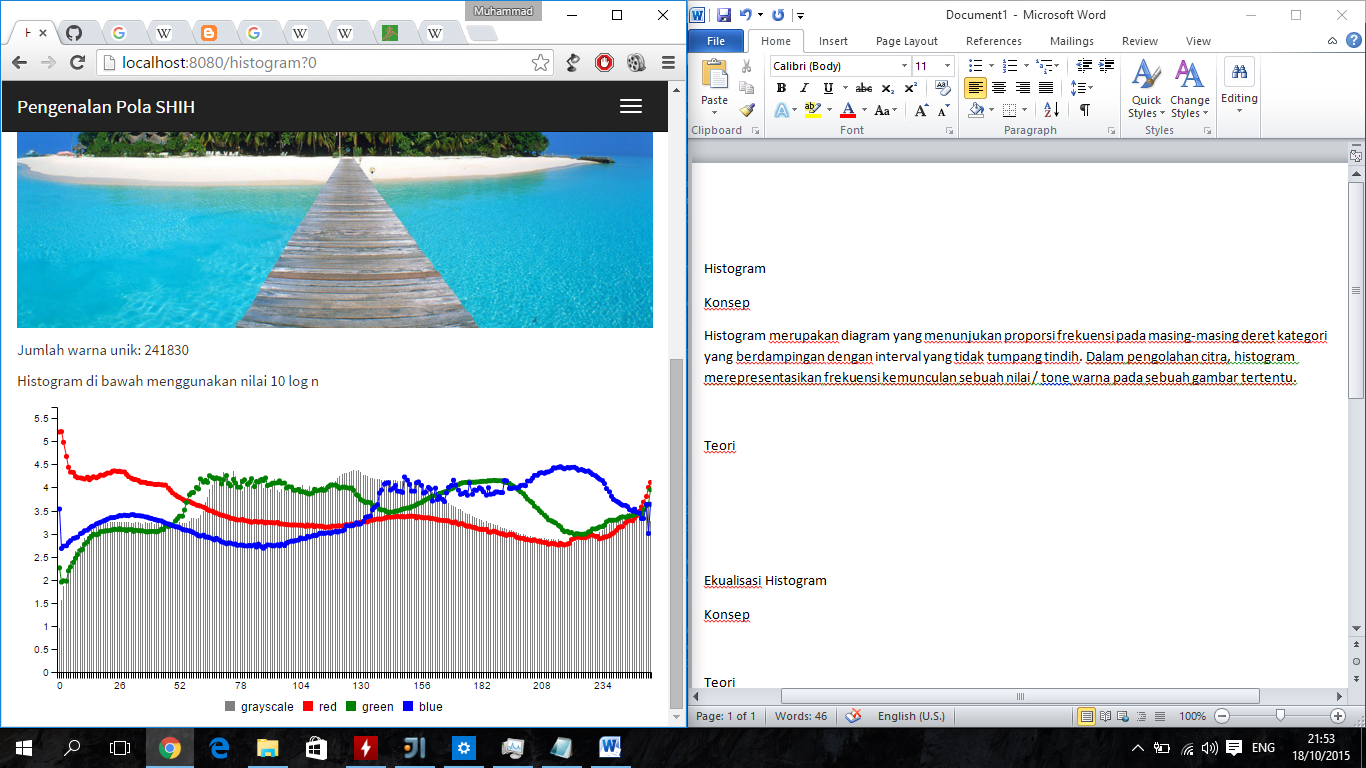
Harto Gunardi - 23213349

Sigit Ari Wijanarko - 23214352

**Histogram**

**Konsep**

Histogram merupakan diagram yang menunjukan proporsi frekuensi pada masing-masing deret kategori yang berdampingan dengan interval yang tidak tumpang tindih. Dalam pengolahan citra, histogram merepresentasikan frekuensi kemunculan sebuah nilai / tone warna pada sebuah gambar tertentu.



Gambar 1 - Histogram

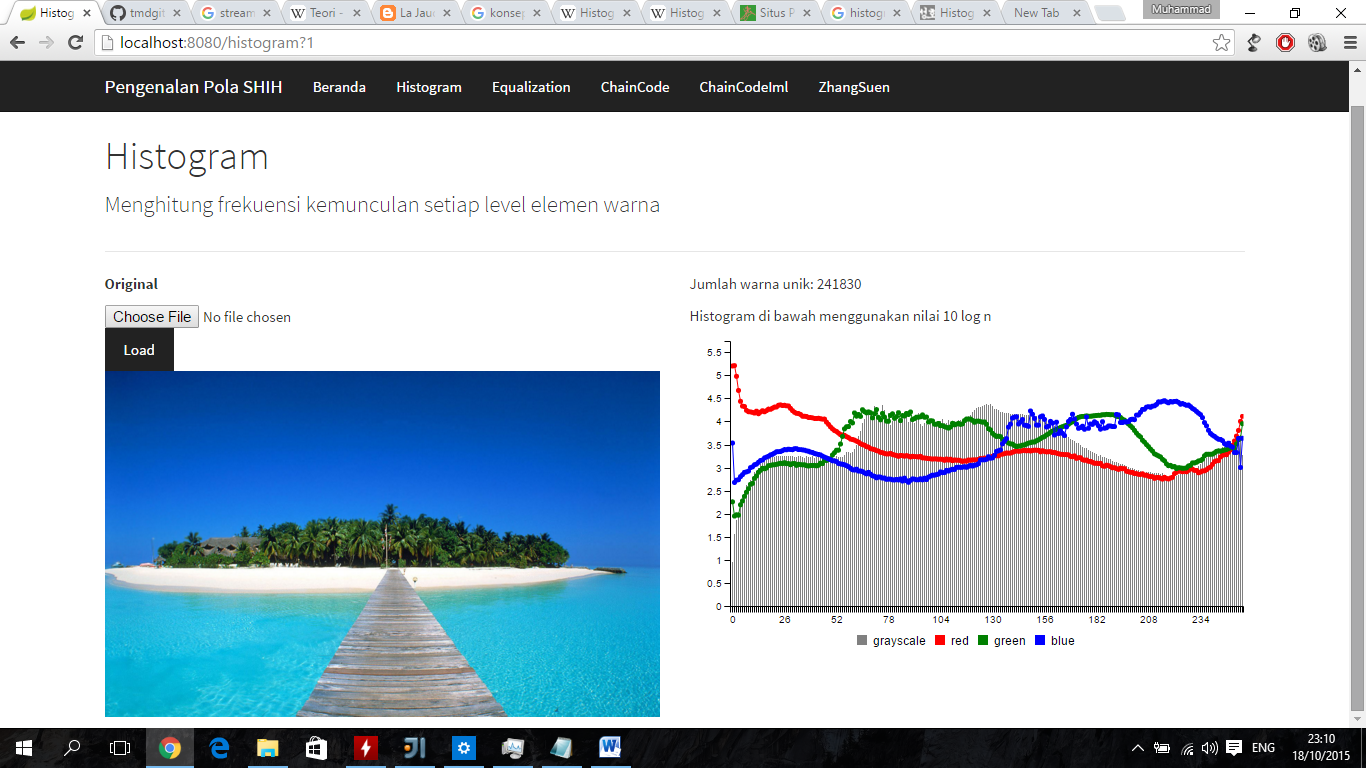
**Teori**

Pada gambar RGB 24-bit, terdapat tiga channel warna yaitu red, green dan blue, yang masing-masing mempunyai 256 variasi tone / shades yaitu mulai dari 0 sampai 255. Histogram kemudian memvisualisasikan frekuensi kemunculan tone warna pada sebuah channel. Tentu saja histogram tidak hanya diperuntukan bagi gambar RGB 24-bit, tetapi juga bisa digunakan untuk format gambar yang lain.

Secara matematis, histogram adalah fungsi *mi* yang menghitung jumlah dari angka yang terdapat dalam sebuah tone warna. Histogram dapat ditulis dengan persamaan berikut:

Dimana n merupakan jumlah bilangan yang ditemukan pada masing-masing tone warna. i adalah observasi pada deret tone warna, dan k adalah jumlah total dari bins / deret tone warna.

**Screenshot Aplikasi**



Gambar 2 - Screenshot App: Output Histogram

**Ekualisasi Histogram**

**Konsep**

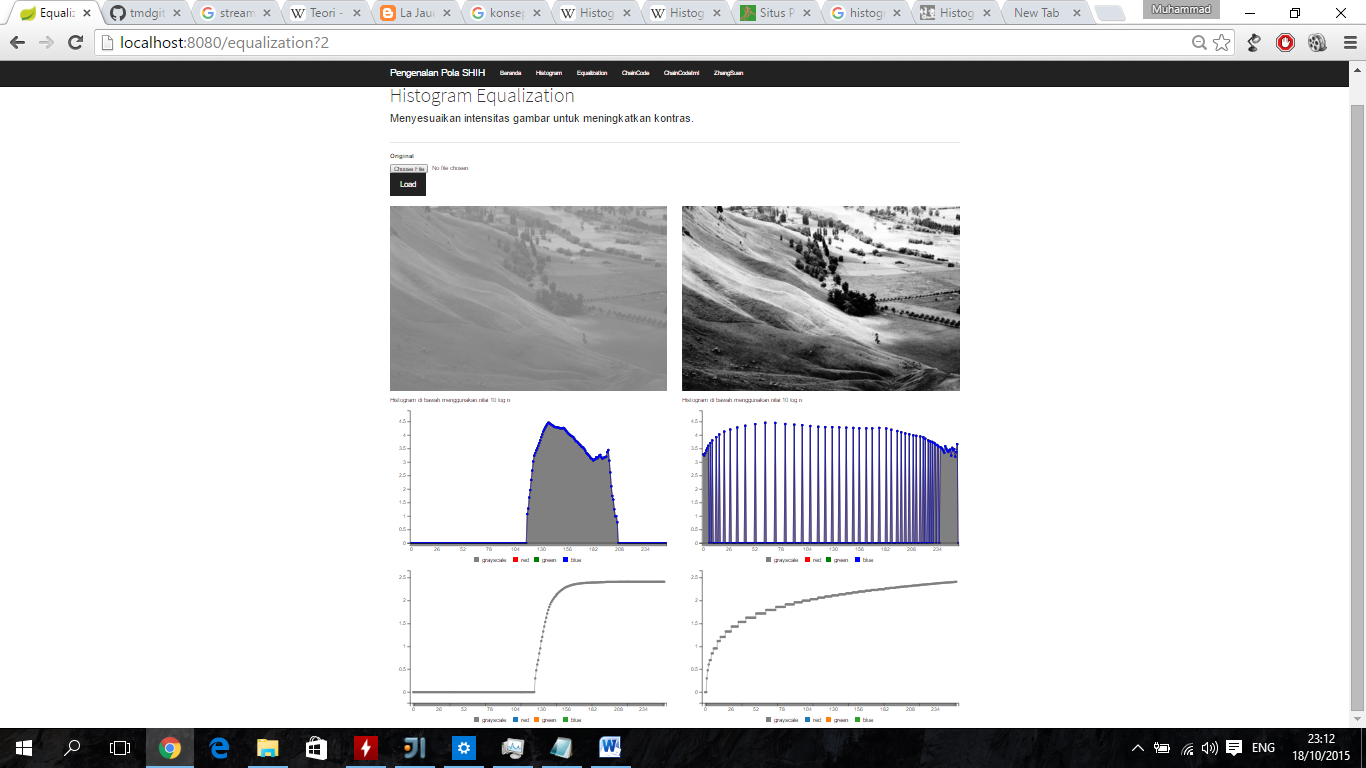
Sebelum sebuah gambar diproses, sebuah gambar harus diberi treatment khusus agar gambar tersebut layak proses. Banyak gambar yang akan diproses terlalu terang atau terlalu gelap sehingga menyulitkan komputer untuk memposesnya. Salah satu teknik dalam peningkatan mutu citra adalah ekualisasi histogram. Konsep dasar dari ekualisasi histogram adalah dengan men-stretch histogram, seningga perbedaan antar warna menjadi lebih besar.

**Teori**

Gambar yang memiliki kontras yang kurang baik cenderung memiliki histogram yang dominan dan berkumpul di suatu daerah. Ekualisasi histogram men-stretch histogram dengan cara mentransformasi nilai warna suatu pixel ke nilai yang lain melalui Look Up Table (LUT). LUT sendiri dihitung melalui jumlah atau nilai dari histogram kumulatif kemudian dikali intensitas maksimum dibagi jumlah pixel gambar tersebut. Untuk lebih jelasnya lihat persamaan di bawah ini:

Dimana x’ merupakan nilai hasil transformasi T(x), *ni* merupakan nilai dari *cumulative histogram, max intensity* merupakan intensitas maksimum dari gambar (dalam hal ini 255), dan *N* merupakan jumlah pixel dari sebuah gambar (*N* = *width* × *height*).

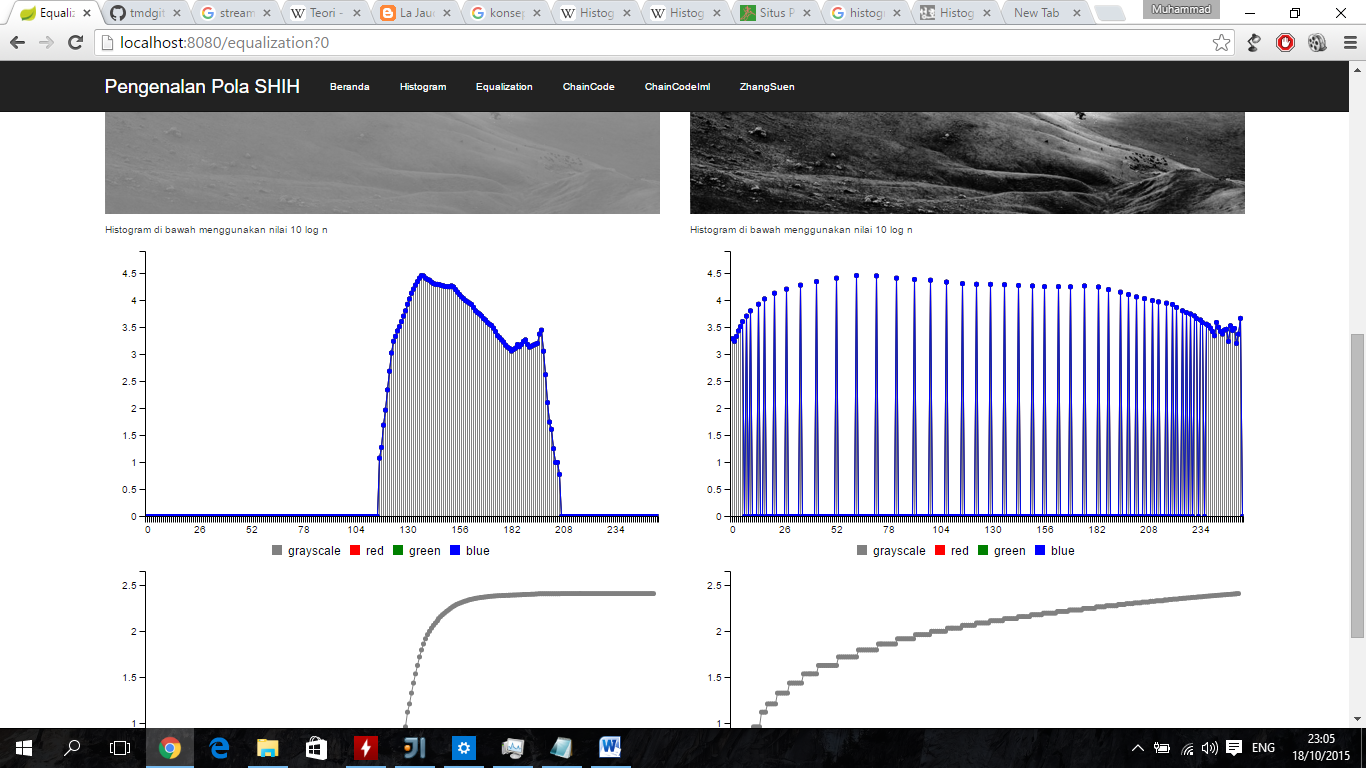
**Screenshot Aplikasi**



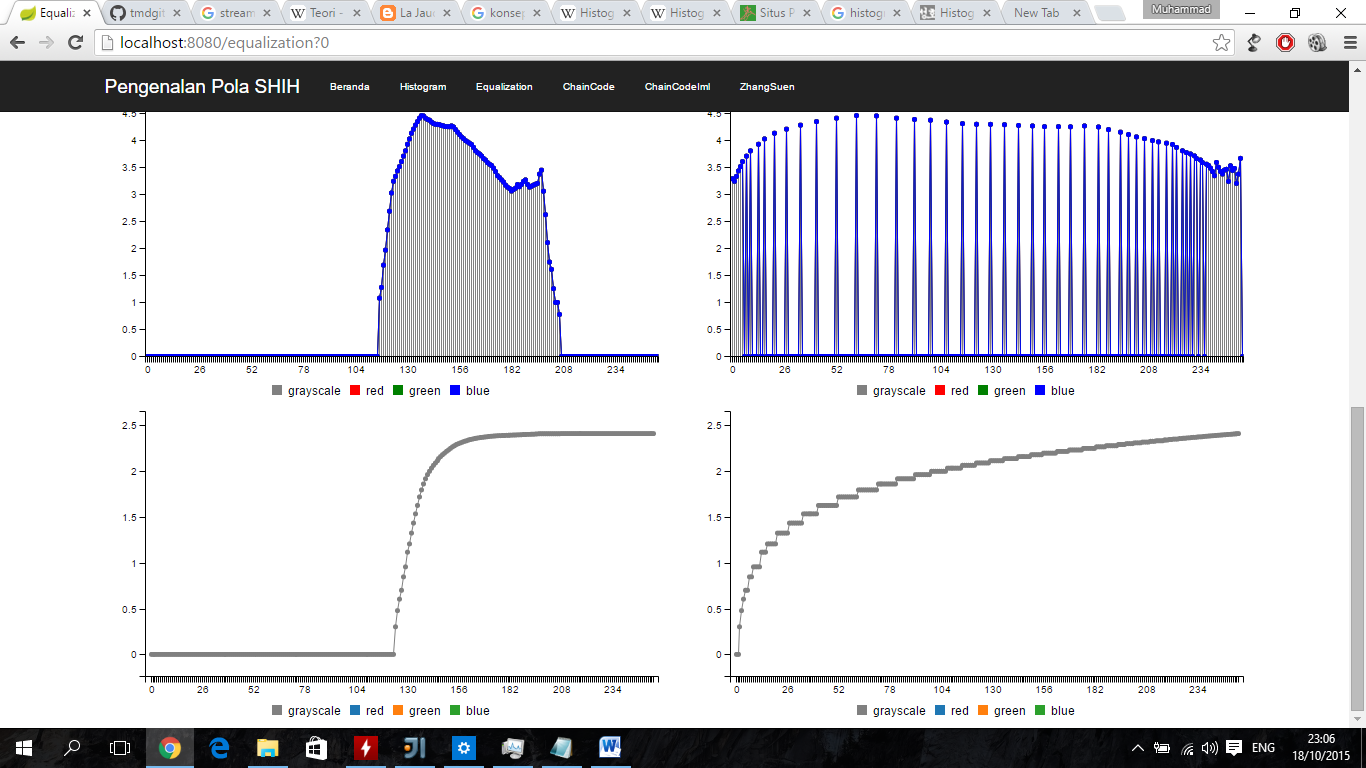
Gambar 3 - Ekualisasi Histogram



Gambar 4 – Gambar sebelum dan sesudah diekualisasi



Gambar 5 - Histogram sebelum dan sesudah



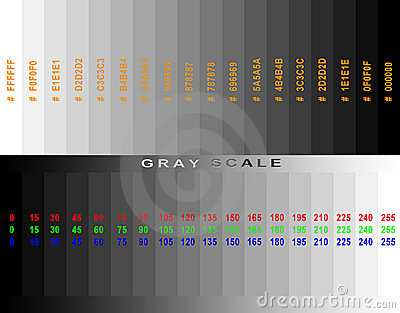
Gambar 6 - Cummulative Histogram sebelum dan sesudah

**ChainCode dan Kode Belok**

**Konsep :**

**Grayscale**

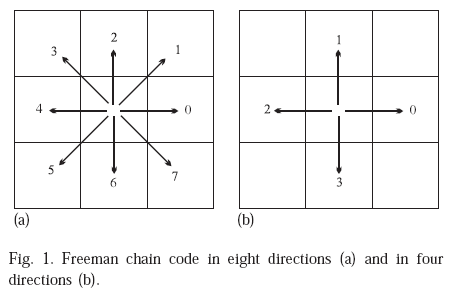
Grayscale adalah warna yang berada dalam rentang gradasi antara hitam dan putih. Image Grayscale memiliki banyak variasi nuansa abu-abu sehingga berbeda dengan image hitam-putih. Warna ini menyatakan derajat keabuan suatu pixel. Grayscale juga disebut monokromatik karena tidak memiliki warna lain selain variasi intensitas putih dan hitam. Warna ini dapat dicari dengan mencari hasil rata-rata. Untuk menghasilkan grayscale adalah :

**

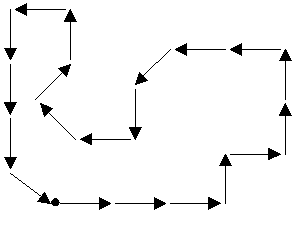
Gambar 7 – Gray scale

**Chaincode**

Chaincode dapat digunakan untuk menggambarkan struktur morfologi suatu objek dalam citra. Chaincode merupakan usaha untuk mendapatkan tepi dari objek, yaitu dengan cara mengelilingi tepian objek. Dalam hal ini, ada 2 acuan arah, yaitu menggunakan 4 arah dan menggunakan 8 arah (FCCE: Freeman Chain Code of Eight Directions).[[1]](#footnote-1)



Gambar 8 – arah 8 dan arah 4 chaincode

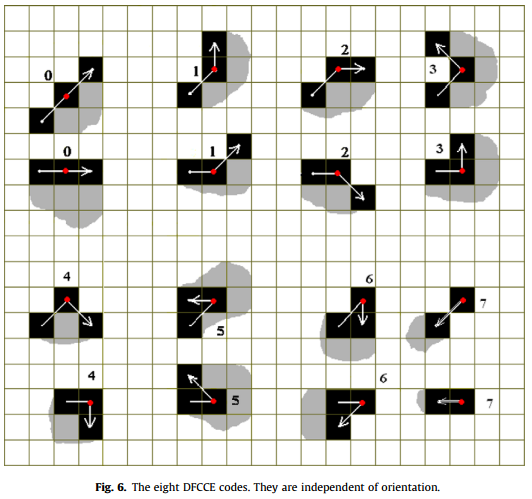


Gambar 9 – Proses chaincode

**Kode Belok**

Kode belok merupakan representasi arah belokan dari suatu objek. Hampir sama dengan chaincode, dalam proses pencarian dapat dilakukan dengan mengelilingi objek dengan menandai setiap belokan atau dapat juga dengan menyederhanakan chaincode. Dalam memperoleh kode belok dapat memilih beberapa acuan arah yaitu :

1. Menggunakan acuan 2 arah, dalam hal ini menggunakan acuan belok kanan dan belok kiri. Atau dapat diseder hananakan menjadi + dan -.
2. Menggunakan acuan 8 arah, yaitu Directional Freeman Chain Code of Eight Directions (DFCCE)[[2]](#footnote-2). Untuk lebih jelasnya bisa melihat gambar sebagai berikut :



Gambar 10 – 8 arah kode belok

**Teori :**

## **Pengenalan Pola menggunakan Chaincode**

Dalam proses ini gambar dengan warna RGB diubah ke grayscale, kemudian menentukan tingkat keabuan objek yang akan dicari, misalkan untuk tingkat keabuan 240. Seperti yang dijelaskan dalam konsep chaincode, dalam proses iterasi dapat menggunakan acuan 8 arah. Proses iterasi akan mengelilingi warna yang mempunyai tingkat keabuan diatas 240 dan mencatat semua arah sampai ke titik awal proses chaincode.

Gambar RGB

gambar Grayscale

Menentukan warna object

Menemukan objek dengan scanline

Iterasi Chaincode

Gambar 11 – Proses mendapatkan chaincode

Kita dapat menyimpan hasil chaincode setiap objek, yang digunakan sebagai data training. Dalam proses pengenalan dapa digambarkan sebagai berikut :

Gambar

Chaincode

Chaincode data training

Chaincode = training

Hasil Pengenalan

Gambar 12 – Proses pengenalan pola

Kelemahan dari chaincode adalah jika gambar sedikit berbeda, maka kemungkinan besar gambar tidak akan dikenali. Karena chaincode yang dihasilkan berbeda.

## **Pengenalan Pola menggunakan Kode Belok**

Telah dijelaskan dalam konsep, bahwa kode belok merupakan data belokan dari objek. Salah satunya dapat diperoleh dengan menganalisi chaincode. Dalam menentukan 8 arah dapat dilakukan dengan perbandingan arah chaincode sekarang dengan arah chaincode yang akan datang. Lihat pada gambar konsep kode belok dan chaincode. Misal arah chaincode 1, arah berikutnya 1, maka Kode belok = 0.

|  |  |  |
| --- | --- | --- |
| (FccEB – FccEA) mod 8 | Keterangan | DFCCE |
| 0 | Lurus | 0 |
| 1 | Belok kiri 45° | 1 |
| 2 | Belok kiri 90° | 3 |
| 3 | Belok kiri 135° | 5 |
| 4 | Balik arah | 7 |
| 5 | Belok kanan 135° | 6 |
| 6 | Belok kanan 90° | 4 |
| 7 | Belok kanan 45° | 2 |

Dengan kode belok akan memperbaiki kesalahan chaincode.

1. Kode belok tidak mengenal arah, sehingga untuk objek dengan beda posisi masih bisa dikenali.
2. Perbedaan ukuran tidak menjadi masalah, karena yang digunakan adalah jumlah belokan

**Thinning Algorithm**

**Konsep**

Masalah umum dari pengenalan pola terletak di efektifitas dan efisiensi dari pengekstraksian fitur pembeda dari pola. Metode analisis garis adalah pendekatan kuat untuk mengenali beberapa jenis dari pola digital tertentu seperti karakter alphanumeric dan ideographs. Garis skeleton dari suatu objek didapat dari penipisan bagian luar dari objek tersebut sedemikian rupa sehingga menyisakan garis yang mewakili bentuk objek awal tadi.

**Teori**

Salah satu thinning algorithm yang cepat dan efektif adalah algoritma yang diusulkan oleh Zhang dan Suen. Algoritma ini memproses gambar biner yang didefinisikan sebagai matrix IT dimana setiap pixel IT(i, j) hanya bernilai 0 atau 1. Transformasi iterative dilakukan terhadap matrix IT titik ke titik berdasarkan nilai dari titik tetangga yang diasumsikan sebagai berikut:

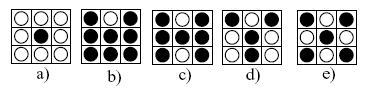
Table 1 Neighbors dari (i, j)

|  |  |  |
| --- | --- | --- |
| P9  (i – 1, j – 1) | P2  (i – 1, j) | P3  (i – 1, j + 1) |
| P8  (i, j – 1) | P1  (i, j) | P4  (i, j + 1) |
| P7  (i + 1, j – 1) | P6  (i + 1, j) | P5  (i + 1, j + 1) |

Algoritma ini mengandung dua sub-iterasi, suatu titik P1 dihapus dari pola jika memenuhi semua persyaratan berikut:

1. 2 ≤ B(P1) ≤ 6
2. A(P1) = 1
3. P2 \* P4 \* P6 = 0
4. P4 \* P6 \* P8 = 0

Dimana A(P1) adalah jumlah dari pola 01 dalam urutan dari P2, P3, P4, sampai P2 lagi yang mana merupakan tetangga dari P1.



Gambar 14.   
0 = white, 1 = black

Gambar 13. A(P1)

Nilai A(P1) untuk masing-masing konfigurasi di atas adalah:

1. A(P1) = 1.
2. A(P1) = 2.
3. A(P1) = 3.
4. A(P1) = 4.

B(P1) merupakan jumlah elemen bukan nol dari tetangga P1.

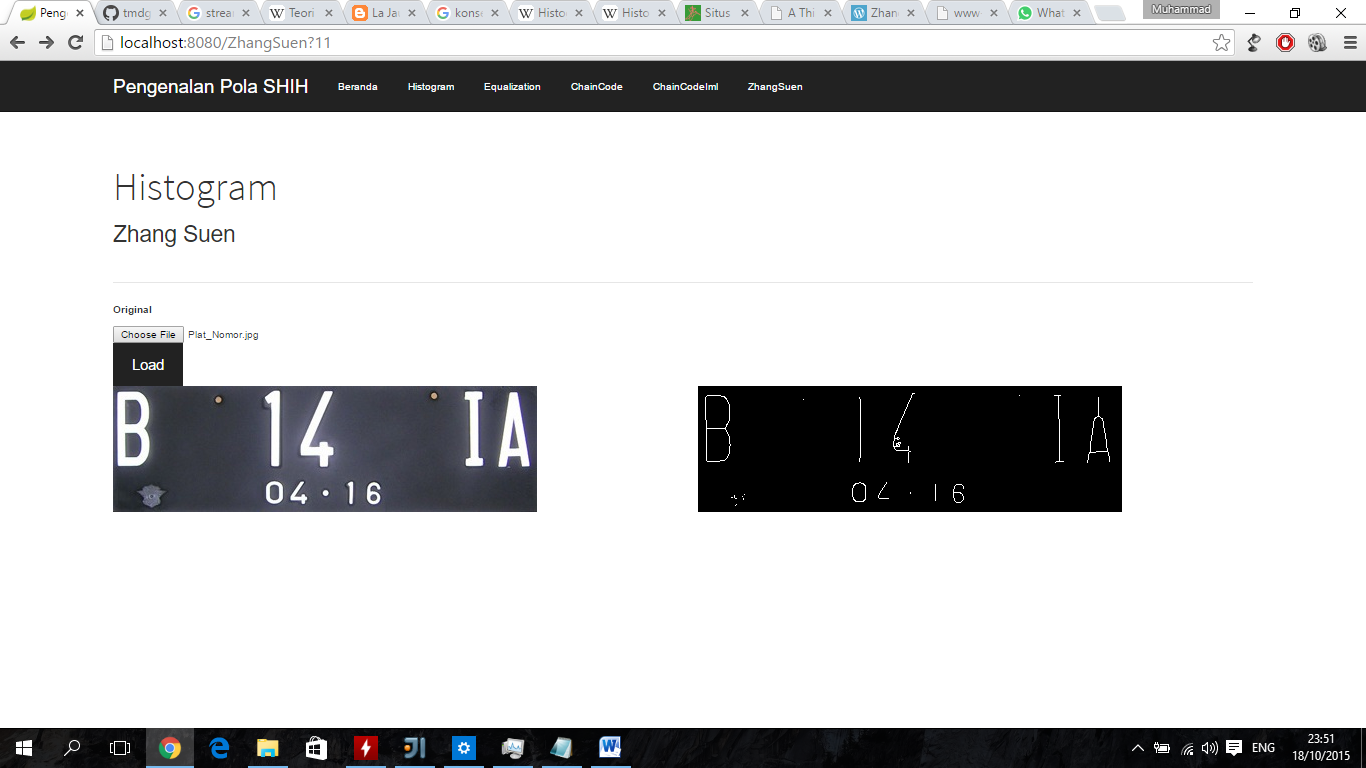
Jika P1 memenuhi semua kriteria diatas, P1 akan ditandai dan dimasukkan ke list pixel yang akan dihapus. Setelah sub-iterasi selesai maka pixel – pixel yang ditandai akan dihapus.

Dalam sub-iterasi kedua, kondisi penghapusan adalah sebagai berikut:

1. 2 ≤ B(P1) ≤ 6
2. A(P1) = 1
3. P2 \* P4 \* P8 = 0
4. P2 \* P6 \* P8 = 0

Penghapusan dilakukan sampai tidak ada lagi titik yang bisa dihapus.

**Screenshot Aplikasi**



Gambar 15 – Thinning plat nomor menggunakan Zhang Suen Algorithm

**Otsu Thresholding**

**Konsep**

Edge detection adalah metode untuk mendapatkan tepi suatu objek dalam sebuah image. Edge detection berisi sebuah model untuk mengidentifikasi suatu titik dalam image digital dimana titik tersebut memiliki perubahan kecerahan (brightness) yang sangat tajam atau terdapat diskontinuitas. Titik titik tersebut yang berjajar akan membentuk sebuah garis yang diidentifikasi sebagai tepi (edge).

Thresholding adalah proses mengubah citra berderajat keabuan menjadi citra biner atau hitam putih sehingga dapat diketahui daerah mana yang termasuk obyek dan background dari citra secara jelas. Citra hasil thresholding biasanya digunakan lebih lanjut untuk proses pengenalan obyek serta ekstraksi fitur. Metode thresholding secara umum dibagi menjadi dua, yaitu :

1. Thresholding global

Thresholding dilakukan dengan mempartisi histogram dengan menggunakan sebuah threshold (batas ambang) global T, yang berlaku untuk seluruh bagian pada citra.

2. Thresholding adaptif

Thesholding dilakukan dengan membagi citra menggunakan beberapa sub citra. Lalu pada setiap sub citra, segmentasi dilakukan dengan menggunakan threshold yang berbeda.

Otsu adalah salah satu teknik untuk melakukan pencarian Thresholding secara otomatis. Operasi yang dilakukan cukup sederhana. Operasi-operasi tersebut didasarkan pada analisa statistik yang diperoleh dari informasi histogram citra.

**Teori**

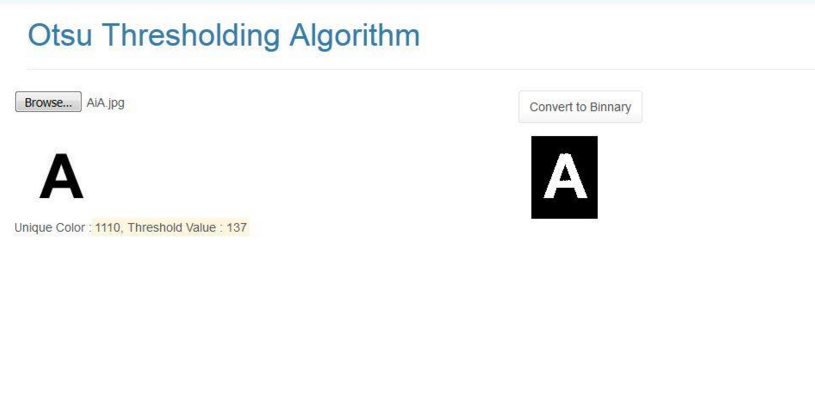
Untuk menentukan threshold citra biner, metode otsu menggunakan analisa statistik histogram. Proses tersebut dapat di dirumuskan sebagai berikut:

keterangan :

* f(x,y) adalah pixel dari citra input, x sebagai baris dan y sebagai kolom.
* T adalah nilai threshold.
* g(x,y) adalah nilai pixel baru hasil dari threshold.

Tujuan dari metode otsu adalah membagi histogram citra *gray level*kedalam dua daerah yang berbeda secara otomatis tanpa membutuhkan bantuan *user*untuk memasukkan nilai ambang Pendekatan yang dilakukan oleh metode otsu adalah dengan melakukan analisis diskriminan yaitu menentukan suatu variabel yang dapat membedakan antara dua atau lebih kelompok yang muncul secara alami. Analisis Diskriminan akan memaksimumkan variable tersebut agar dapat membagi objek latardepan (*foreground*) dan latarbelakang (*background*). Adapun code sebagai berikut :

|  |
| --- |
| public void findThreshold2() {  double sum = 0;  for (int i = 1; i < 256; ++i)  sum += i \* hist.lv[3][i];//merupakan histogram dari pixel 0-255  double sumB = 0;  double wB = 0;  double wF = 0;  double mB;  double mF;  double max = 0.0;  double between = 0.0;  double threshold1 = 0.0;  double threshold2 = 0.0;  for (int i = 0; i < 256; ++i) {  wB += hist.lv[3][i];  if (wB == 0)  continue;  wF = hist.totalPix - wB;  if (wF == 0)  break;  sumB += i \* hist.lv[3][i];  mB = sumB / wB;  mF = (sum - sumB) / wF;  between = wB \* wF \* (mB - mF) \* (mB - mF);  if ( between >= max ) {  threshold1 = i;  if ( between > max ) {  threshold2 = i;  }  max = between;  }  }  thresholdValue = (int)(( threshold1 + threshold2 ) / 2.0);  } |



**Optical Character Recognition**

**Konsep**

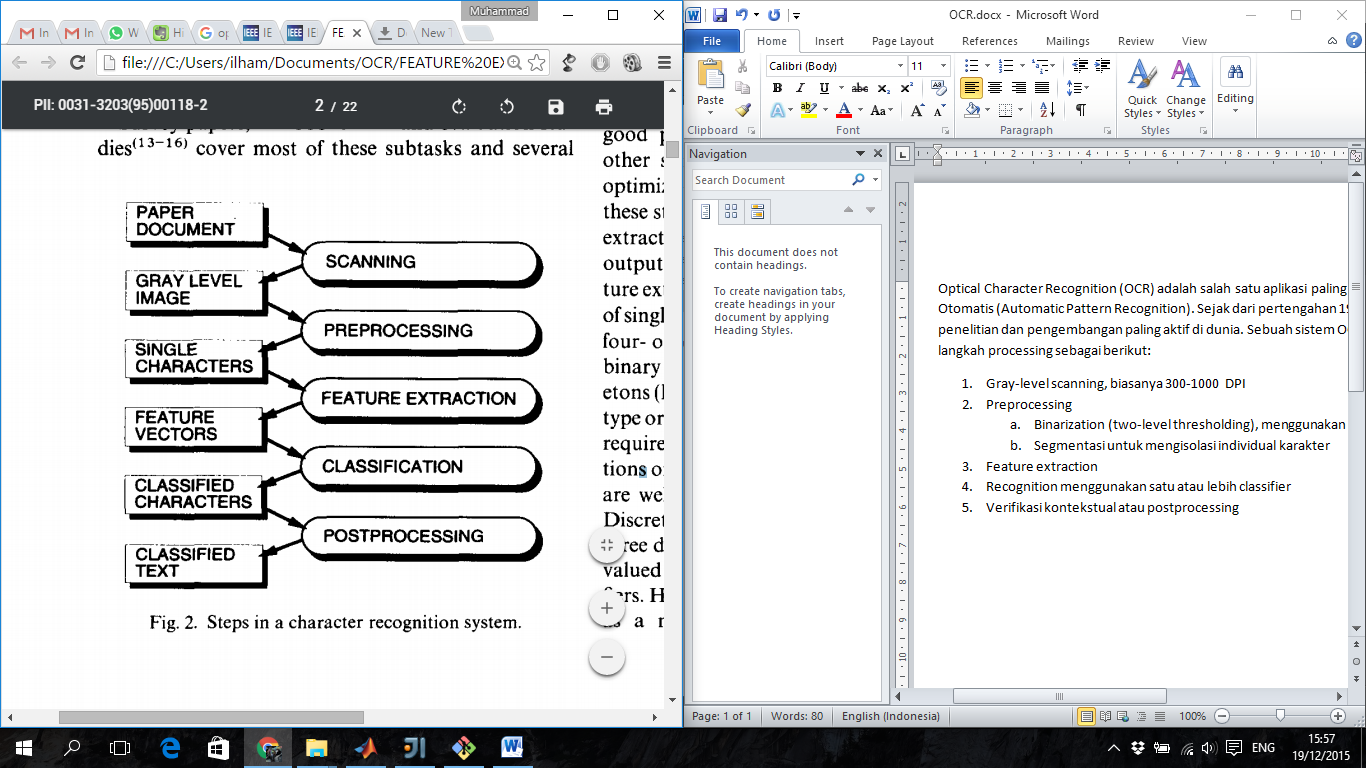
Optical Character Recognition merupakan alat atau teknik untuk menerjemahkan gambar dari tulisan tangan ataupun hasil ketikan, menjadi sebuah teks yang bisa dikenali dalam komputer. Konsep dasarnya adalah dengan memisahkan masing-masing karakter dalam suatu gambar, kemudian objek / karakter tersebut direkognisi dengan menggunakan metode-metode pengenalan pola gambar.

**Teori**

OCR adalah salah satu aplikasi paling sukses dari Pengenalan Pola Otomatis (Automatic Pattern Recognition). Sejak dari pertengahan 1950an, OCR telah menjadi bidang penelitian dan pengembangan paling aktif di dunia. Sebuah sistem OCR biasanya mengandung langkah-langkah processing sebagai berikut:

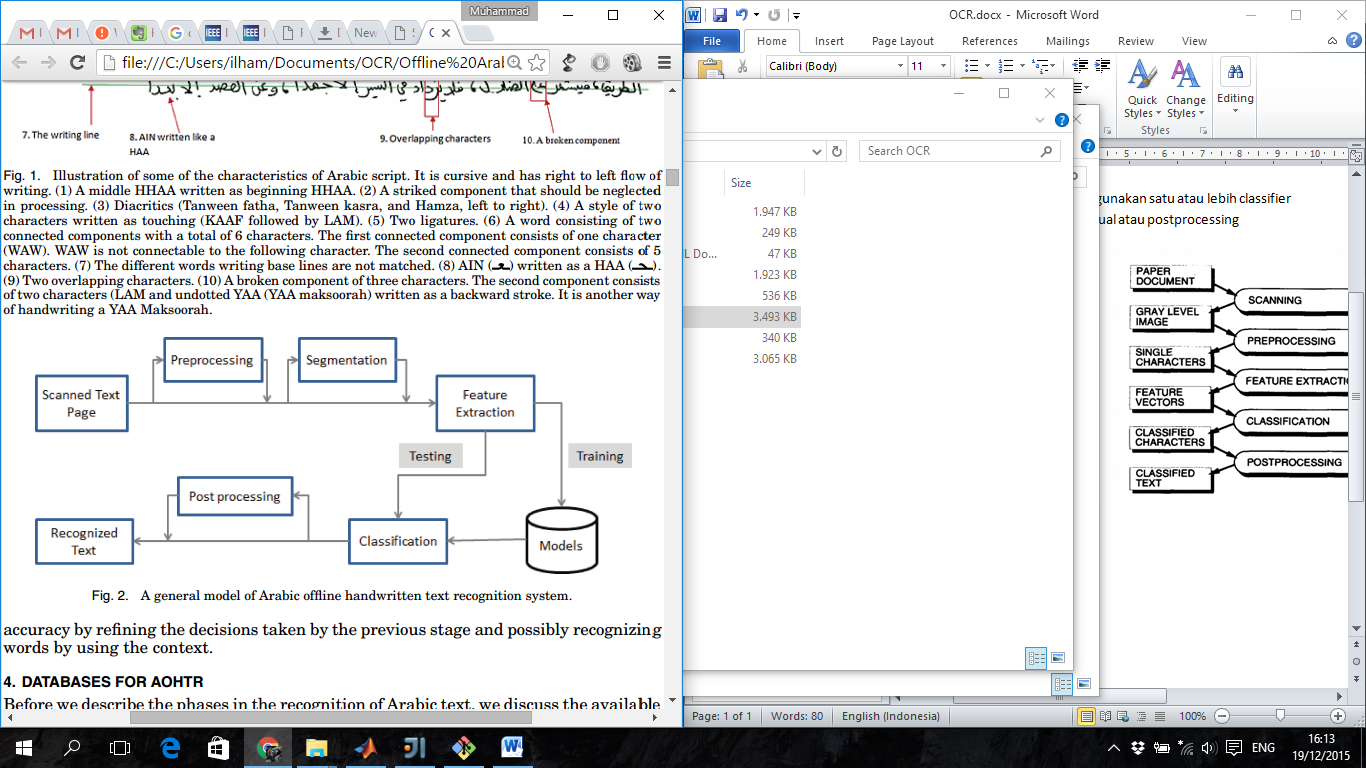
1. Gray-level scanning, biasanya 300-1000 DPI
2. Preprocessing
   1. Binarization (two-level thresholding), menggunakan global atau local adaptive method
   2. Segmentasi untuk mengisolasi individual karakter
3. Feature extraction
4. Recognition menggunakan satu atau lebih classifier
5. Verifikasi kontekstual atau postprocessing

Gambar dibawah ini menunjukkan input dan output dari masing-masing langkah processing pada OCR:



Gambar 16 – Input dan output masing-masing langkah processing pada OCR

Kemudian diagram blok tentang alur proses OCR bisa dilihat pada gambar dibawah ini:



Gambar 17 – Alur umum proses OCR

Pemilihan metode feature extraction mungkin merupakan faktor paling penting dalam mendapatkan performa rekognisi yang tinggi. Banyak feature extraction yang diusulkan oleh para peneliti, diantaranya: template matching, deformable templates, unitary image transforms, graph description, projection histograms, contour profiles, zonig, geometric moment invariants, zernike moments, spline curve approximation, dan fourier descriptors. Masing-masing metode tersebut bisa diaplikasikan ke satu atau lebih bentuk berikut:

1. Gray-level character image
2. Binary character image
3. Character contour
4. Character skeleton atau character graph

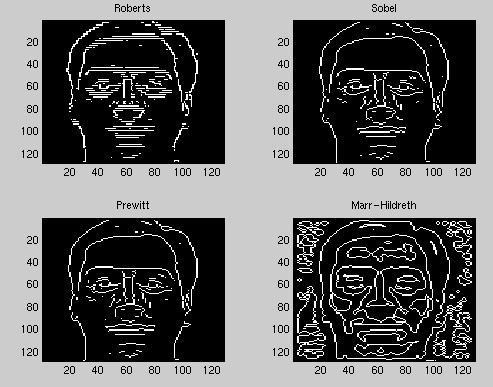
**Edge Detection**

**Konsep**

Edge detection adalah metode untuk mendapatkan tepi suatu objek dalam sebuah image. Edge detection berisi sebuah model untuk mengidentifikasi suatu titik dalam image digital dimana titik tersebut memiliki perubahan kecerahan (brightness) yang sangat tajam atau terdapat diskontinuitas. Titik titik tersebut yang berjajar akan membentuk sebuah garis yang diidentifikasi sebagai tepi (edge).

**Teori**

Ada banyak cara untuk melakukan deteksi tepi. Namun, kebanyakan dapat dikelompokkan menjadi dua kategori, yaitu Gradien dan Laplacian. Metode gradien mendeteksi tepi dengan mencari maksimum dan minimum di turunan pertama gambar. Metode Laplacian mencari zerocrossing di turunan kedua gambar untuk menemukan tepi. Gambar pertama ini menunjukkan tepi gambar dideteksi dengan menggunakan metode gradient (Roberts, Prewitt, Sobel) dan metode Laplacian (Marrs-Hildreth).



Gambar 18 – Edge detection dengan metode gradien (Roberts, Prewitt, Sobel) dan metode Laplacian (Marrs-Hildreth)

Perhatikan bahwa fitur wajah (mata, hidung, mulut) memiliki tepi yang sangat tajam. Ini yang menjadikan fitur fitur wajah tersebut (mata, hidung, mulut) menjadi titik referensi terbaik untuk membandingkan antara dua gambar.

Menurut Roberts, detektor tepi harus memiliki sifat sebagai berikut: tepi yang dihasilkan harus didefinisikan dengan baik, latar belakang harus berkontribusi noise sesedikit mungkin, dan intensitas tepi harus sesuai sedekat mungkin dengan apa yang persepsi manusia lihat. Dengan kriteria ini dalam pikiran dan berdasarkan kemudian berlaku teori psikofisik Roberts mengusulkan persamaan berikut:


y_{i,j} = \sqrt{x_{i,j}}



z_{i,j} = \sqrt{(y_{i,j} - y_{i+1,j+1})^2 + (y_{i+1,j} - y_{i, j+1})^2  } 


di mana x adalah nilai intensitas awal dalam gambar, z adalah turunan dihitung dan i, j mewakili lokasi dalam gambar.

Hasil operasi ini akan meng-highlight perubahan intensitas dalam arah diagonal. Salah satu aspek yang paling menarik dari operasi ini adalah kesederhanaannya; kernel kecil dan hanya berisi integer. Namun dengan kecepatan komputer saat ini keuntungan ini tidak akan terlalu signifikan dan Roberts cross memiliki kelemahan yaitu kepekaan terhadap noise.

1. Freeman, Herbert. "On the encoding of arbitrary geometric configurations."*Electronic Computers, IRE Transactions on* 2 (1961): 260-268. [↑](#footnote-ref-1)
2. Liu, Yong Kui, and Borut Žalik. "An efficient chain code with Huffman coding."*Pattern Recognition* 38.4 (2005): 553-557. [↑](#footnote-ref-2)