BAB II
LANDASAN TEORI

Bab ini menjelaskan analisis berbagai teori dan hasil penelitian yang relevan dengan masalah yang diteliti. Dalam bagian ini melakukan sintesis terhadap teori yang relevan agar diperoleh legitimasi konseptual terhadap variabel yang diteliti.

2.1 Citra Digital

Citra digital merupakan suatu array yang berisi nilai-nilai real maupun kompleks serta direpresentasikan intensitasnya melalui suatu nilai. Citra digital disusun oleh fungsi f(x,y) yang memiliki ukuran M baris dan N kolom, (Mirah, 2018) Nilai x dan y sendiri merupakan nilai koordinat spasial, setiap piksel pada titik koordinat memiliki amplitude bernilai f yang merupakan tingkatan intensitas dari piksel tersebut.

Menurut (Kadir & Adhi, 2013), jenis citra yang dapat digunakan dalam pengolahan citra digital, terbagi dalam 3 jenis, yaitu citra biner, citra *grayscale*¸ serta citra warna. Ketiganya memiliki sebagai berikut:

1. Citra berwarna (RGB)

   Citra berwarna adalah citra yang setiap piksel diwakili oleh 3 komponen yaitu komponen warna R (red), G (green), dan B (blue). Setiap komponen, ataupun kanal warna tersebut menggunakan nilai berbasis delapan bit yang memiliki nilai dengan kisaran 0 sampai 255. Sehingga, warna yang dihasilkan adalah 255 x 255 x 255 atau setara dengan 16.581.375 warna.

2. Citra *Grayscale*

   Citra *grayscale* merupakan citra yang intensitasnya diwakili oleh nilai keabuan setiap piksel dan memiliki nilai dengan kisaran 0 sampai 255. Nilai 0 merupakan nilai terendah dengan warna hitam, sementara nilai 255 merupakan nilai tertinggi dengan warna putih.

3. Citra Biner

Citra biner merupakan citra monokrom dan hanya memiliki 2 kemungkinan intensitas warna, yaitu hitam (0) dan putih (1). Secara umum, citra biner sering disebutkan dengan sebutan citra hitam putih, ataupun citra *black and white.*

2.2 *Optical Character Recognition (OCR)*

Optical Character Recognition (OCR) merupakan teknologi yang bekerja mengubah citra yang berisi teks menjadi teks yang dapat diolah pada komputer. OCR juga didefinisikan sebagai pengenalan karakter pada citra. (Chandra, Pradipta, & Alamsyah, 2018) OCR digunakan pada tulisan yang ditulis menggunakan tangan ataupun tulisan hasil cetak.

2.3 *Pre Proccesing*

Pada tahap ini dikenal proses meningkatkan kualitas citra yang bertujuan untuk meningkatkan keberhasilan pada tahap pengolahan citra digital pada penelitian ini digunakan *grayscaling, low pass filtering, high pass filtering, otsu thresholding* dan *image resizing*.

2.3.1 *Grayscaling*

Citra *grayscale* adalah citra yang memiliki warna yang dipakai warna hitam sebagai warna minimal (0) dan warna putih (255) sebagai warna maksimalnya, sehingga warna antaranya adalah abu-abu (Indraani, Jumaddina, & Sinaga, 2014). Dari pendapat tersebut dapat disimpulkan citra *grayscale* merupakan citra yang memiliki range piksel dari 0 sampai dengan 255 dengan pola 8 bit di mana nilai 0 merepresentasikan warna hitam sementara 255 merepresentasikan warna putih.

Proses *grayscaling* mengurangi dimensi yang dimiliki oleh citra, dengan melakukan pemetaan citra tiga kanal warna menjadi hanya satu kanal warna yaitu warna keabuan (Lanaro, Nguyen, & Kasampalis, 2019). Sehingga 3 komponen pada citra RGB (*Red, Green, Blue)* setelah dikonversi menjadi citra *grayscale* hanya memiliki 1 komponen warna yaitu abu. Untuk melakukan proses konversi citra RGB menjadi *grayscale* diperlukan proses perhitungan.

Terdapat berbagai cara dalam melakukan proses *grayscaling*, diantaranya adalah menggunakan rumus yang ditunjukkan pada Persamaan 2.1

$$Grayscale = 0.21 * \mathrm{R} + 0.72 * \mathrm{G} + 0.07 * \mathrm{B} \quad (2.1)$$

Dimana, nilai 0.21, 0.72 dan 0.07 merupakan ketetapan dan R (Red) adalah intensitas piksel berwarna merah, G (Green) adalah intensitas piksel berwarna hijau, dan B (Blue) adalah intensitas pikel berwarna biru.

2.3.2 *Low Pass Filtering*

Low Pass Filter adalah suatu filter yang digunakan untuk menyeleksi piksel-piksel dari citra. Filter ini mempunyai sifat meloloskan yang berfrekuensi rendah dan menghilangkan yang berfrekuensi tinggi (Prakoso, 2017). *Low Pass Filter* menghasilkan citra yang lebih halus (*blur)*.

Salah satu metode LPF adalah dengan menggunakan metode *mean filtering*. Metode tersebut menggantikan nilai suatu piksel dengan menggunakan nilai rataan dari 8 piksel tetangga (Manalu & Sinurat, 2017). *Mean filtering* bekerja dengan cara mengkonvolusi nilai setiap piksel pada citra dengan suatu kernel. Pada studi kasus ini kernel yang dipakai merupakan kernel dari *low pass filter* dengan nilai 1/9 yang di tunjukan pada Persamaan 2.2.

$$Kernel = \begin{bmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{bmatrix} \quad (2.2)$$

2.3.3 *Sharpening (High Pass Filtering)*

Sharpening atau proses penajaman citra digunakan untuk memperjelas detail dari citra dan mengakibatkan citra terlihat lebih tajam (Tayja, Lestari, & Khair, 2018). Tujuan utama dari tahapan *sharpening* adalah menajamkan citra yang terlihat *blur* ataupun kabur dikarenakan tahapan *low pass filtering.* Salah satu metode yang digunakan untuk menajamkan citra adalah *High Pass Filtering* yaitu dengan mengkonvolusikan citra dengan menggunakan kernel *High Pass* yang ditunjukkan pada Persamaan 2.3.

$$Kernel = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{bmatrix} \quad (2.3)$$

2.3.4 *Otsu Tresholding*

Otsu thresholding merupakan metode segmentasi yang sederhana dalam teknik segmentasi, sehingga dapat lebih mudah dalam melakukan pembagian wilayah-wilayah yang homogen berdasarkan kriteria keserupaan untuk mengenali objek. Proses sebelum dilakukan pengamatan terhadap citra hasil segmentasi, terlebih dahulu harus melalui proses *input* citra, agar mudah ke proses selanjutnya. Proses selanjutnya adalah melakukan penambahan kecerahan pada citra agar memperbaiki kualitas citra. Setelah itu dilakukan proses segmentasi citra dengan metode *otsu thresholding* dan melalui pendekatan analisis diskriminan sehingga dapat memaksimumkan variabel tersebut agar objek dengan latar belakang dapat terpisah secara otomatis (Utami, 2017). Untuk lebih memaksimalkan hasil citra tersegmen dilakukan proses morfologi, pada penelitian ini *morofologi* dilakukan dengan operasi erosi. Untuk mengetahui lebih jelas batas ambang sebuah citra setelah melalui proses segmentasi yaitu dengan menggunakan histogram, serta dilakukan penghitungan validasi dengan metode PSNR dan MSE untuk mengetahui tingkat kesuksesan dalam proses segmentasi.

Otsu thresholding menentukan nilai *threshold* terbaik dengan cara menentukan nilai *within class variance* terkecil dan memiliki nilai *between class variance* terbesar (Suryapradipta, 2018). Kedua nilai tersebut didapatkan dari Persamaan 2.4 dan Persamaan 2.5 berikut :

$$Within\ Clss\ Variance\ ({\sigma^2}_W) = ({W_b}.{\sigma^2}_b) + ({W_f}.{\sigma^2}_f) \quad (2.4)$$

$$\text{Between Class Variance } ({\sigma^2}_B) = W_b W_f \left(\mu_f - \mu_b\right)^2 \quad (2.5)$$

$$W_b(t) = \sum_{i=1}^{t} P(i) \quad (2.6)$$

$$\mu_b(t) = \frac{\sum_{i=1}^{t} i * P(i)}{W_b(t)} \quad (2.7)$$

$$W_f(t) = \sum_{i=t+1}^{l} P(i) \tag{2.8}$$

$$\mu_f\,(t) = \frac{\sum_{i=t+1}^{l} i * P(i)}{W_f(t)} \tag{2.9}$$

$$\sigma^2{}_b\,(t) = \frac{\sum_{i=1}^{l}\left(i - \mu_b(t)\right)^2 * P(i)}{W_b(t)} \tag{2.10}$$

$$\sigma^2{}_f\,(t) = \frac{\sum_{i=t+1}^{l}\left(i - \mu_f(t)\right)^2 * P(i)}{W_f(t)} \tag{2.11}$$

Keterangan :

W_b = nilai dari *weight background*. Nilai dari *weight background* didapatkan dari Persamaan 2.6.

μ_b = nilai dari *mean background*. Nilai dari *mean background* didapatkan dari Persamaan 2.7.

W_f = nilai dari *weight foreground*. Nila dari *weight foreground* didapatkan dari Persamaan 2.8.

μ_f = nilai dari *mean foreground*. Nilai dari *mean foreground* didapatkan dari Persamaan 2.9.

$\sigma^2{}_b$ = nilai dari *variance background*. Nilai dari *variance background* didapatkan dari Persamaan 2.10.

$\sigma^2{}_f$ = nilai dari *variance foreground.* Nilai dari *variance foreground* didapat dari Persamaan 2.11.

Nilai $P(i)$ = nilai probabilitas histogram dari nilai *grayscale* yang diamati (i = 1,.., l)

Nilai t = nilai *threshold* yang sedang dilakukan pengujian.

Nilai l = nilai piksel tertinggi pada *histogram* citra.

Nilai i = nilai piksel terendah pada histogram citra.

Persamaan-persamaan tersebut didapatkan dari penelitian yang dilakukan oleh (Yousefi, 2015). Nilai *threshold* yang telah didapatkan kemudian menjadi sebuah acuan pemisahan piksel berwarna putih dan berwarna hitam dengan menggunakan Persamaan 2.12 berikut (Suryapradipta, 2018) :

$$f(x,y) = \begin{cases} 255, apabila\ f(x,y) > T \\ 0, apabila\ f(x,y) \leq T \end{cases} \quad (2.12)$$

T adalah nilai titik ambang (*threshold)* yang telah didapatkan dari metode otsu. Apabila nilai piksel melebihi nilai *threshold* maka intensitas piksel tersebut bernilai putih (1 atau 255) dan sebaliknnya apabila nilai piksel tersebut kurang dari nilai *threshold* maka nilai intensitas piksel tersebut bernilai hitam (0).

2.4 Segmentasi *Mathematical Morphology*

Segmentasi adalah proses dalam membagi-bagi citra menjadi beberapa komponen-komponen ataupun objek yang saling terpisah (Mirah, 2018). Tahapan ini bertujuan untuk membagi citra menjadi bagian-bagian pokok yang mengandung informasi penting.

2.4.1 *Mathematical Morphology*

Morfologi matematika adalah suatu teknik dasar pengolahan citra digital yang berdasarkan pada bentuk dari objek-objek ataupun piksel-piksel yang menyusun citra. (Trisnadik, Hidayatno, & Isnanto, 2013). Dua operasi dasar dari operasi morfologi adalah operasi dilasi dan operasi erosi. Secara sederhana, operasi dilasi adalah proses penebalan objek-objek pada citra sehingga objek tersebut meluas. Sementara, berbanding terbalik, operasi erosi adalah proses penipisan objek dari citra.

Dalam proses morfologi, pemilihan bentuk dan ukuran *structuring elements (strel)* menjadi peranan penting dan mempengaruhi hasil dari morfologi. *Structuring elements* atau biasa disebut *strel* atau *SE* adalah kumpulan piksel-piksel yang membentuk himpunan kecil, dan berukuran lebih kecil dibandingkan citra yang diolah (Arini, Fahtianto, Agusta, & Muharam, 2015).

Dua proses utama dalam melakukan segmentasi *Mathematical Morphology* memanfaatkan matriks strel *vertical* dan *horizontal* (Trisnadik, Hidayatno, & Isnanto, 2013). Pada penelitian ini, proses segmentasi bertujuan untuk menemukan lokasi baris dan kata dengan memanfaatkan operasi morfologi dilasi menggunakan matriks strel *horizontal* dan *vertical*.

1. Matriks *Strel Horizontal*

Matriks *horizontal* berfungsi untuk menebalkan objek secara *horizontal* (sejajar sumbu x). Dengan menggunakan morfologi dilasi, maka citra memiliki garis tebal secara *horizontal*. Objek tebal tersebut menunjukan posisi dari baris teks pada citra. Hasil dari dilasi tersebut kemudian menjadi acuan untuk dilakukan pemotongan pada citra dan mengurangi background yang tidak diperlukan oleh sistem. Gambar 2.1 merupakan contoh dari matriks strel horisontal dengan ukuran 1x7 piksel.

1	1	1	(1)	1	1	1

Gambar 2.2 Matriks strel *horizontal* (1x7 piksel)

(Trisnadik, Hidayatno, & Isnanto, 2013)

2. Matriks *Strel Vertical*

Matriks *Vertical* berfungsi untuk menebalkan objek secara *vertical* (sejajar sumbu y). Pada penelitian ini matriks *vertical* bertujuan untuk menebalkan huruf-huruf yang merepresentasikan suatu citra guna menunjukan posisi dari setiap huruf pada citra. Hasil dari dilasi dengan matriks *vertical* merupakan acuan untuk memotong huruf-huruf yang terdapat pada citra. Gambar 2.2 menunjukan contoh matriks *strel vertical* dengan ukuran 7x1 piksel.

1
1
1
(1)
1
1
1

Gambar 2.3 Matriks strel *vertical* (7x1 piksel)

(Trisnadik, Hidayatno, & Isnanto, 2013)

2.4.2 *Plot Contour*

Contour atau dalam bahasa Indonesia disebut kontur adalah rangkaian piksel-piksel pada tepian objek yang membentuk batas daerah (Munir, 2004). Menurut definisi lainnya, kontur adalah keadaan yang diakibatkan oleh adanya perubahan intensitas pada piksel-piksel tetangga, sehingga terdapat tepian objek pada citra (Amikom Purwokerto, 2017). Proses *plot contour* bertujuan untuk menentukan koordinat objek yang terdapat pada citra (Kumaseh, Latumakulita, & Nainggolan, 2013). Sehingga, pada penelitian ini tahapan *plot contour* digunakan untuk menentukan koordinat-koordinat tepian dari kontur yang dihasilkan pada proses morfologi matematika yang telah dilakukan sebelumnya. Koordinat-koordinat tersebut kemudian menjadi dasar proses segmentasi teks yang terdapat pada citra.

2.5 *Resize*

Resize merupakan proses mengubah ukuran besar citra dalam satuan piksel (Anas, 2016). Tahapan *resizing* pada normalisasi dilakukan dengan tujuan menyesuaikan ukuran citra latih & citra uji. Perubahan ukuran citra dapat menghasilkan citra yang lebih besar maupun lebih kecil dari citra asli.

2.6 *Directional Feature Extraction*

Feature extraction adalah proses mengenai bagaimana mendapatkan informasi dari suatu data yang merupakan informasi yang dianggap paling cocok untuk kemudian dilakukan klasifikasi atau menurut pengertian lainnya, *Feature Extraction* atau dalam bahasa Indonesia (Aggarwal, Singh, & Singh, 2015), Ekstraksi fitur atau ekstraksi ciri adalah suatu algoritma yang bertujuan guna memet pola dari citra berdasarkan ciri-ciri yang dimiliki oleh citra tersebut (Bahri & Maliki, 2012). Maka, ekstraksi ciri mendapatkan nilai-nilai yang merupakan ciri unik dari tiap-tiap objek ataupun dari tiap-tiap citra. Salah satu metode untuk mendapatkan ciri tersebut adalah *Directional Feture Extraction*.

Directional Feature Extraction merupakan salah satu metode ekstraksi ciri yang menganalisa ciri arah dari tiap-tiap objek yang dideteksi (Putri, Irawan, & Ahmad, 2017). Ciri / Fitur yang diambil oleh metode ini adalah nilai-nilai gradien

dari tiap-tiap piksel. Gradien adalah nilai kuantitas vektor yang merupakan komponen arah dan dihitung dengan melibatkan kedua arah baik arah horizontal dan arah vertikal (Aggarwal, Singh, & Singh, 2015).

Penghitungan gradien dilakukan dengan menggunakan operator *sobel* baik operator vertikal maupun operator horizontal sehingga dalam perhitungannya dapat diketahui tepian dari objek baik secara vertikal maupun horizontal. Hasil operasi *sobel* secara horizontal ditunjukkan pada Gambar 2.4, sementara hasil operasi vertical ditunjukkan pada Gambar 2.5.

Gambar 2.4 Sobel Horizontal

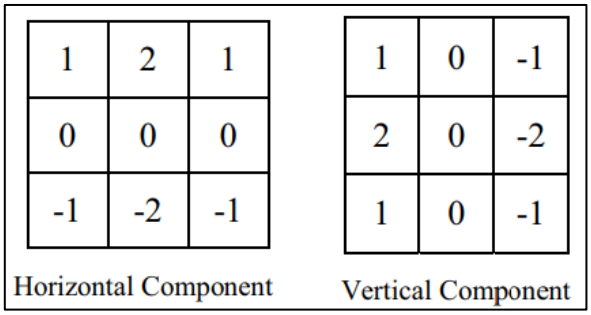
Gambar 2.5 Sobel Horizontal

(Wang, 2018)

Langkah-langkah utama dari algoritma *gradient feature extraction* adalah sebagai berikut :

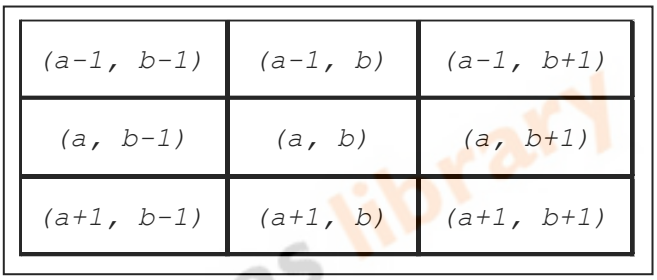
1. Menentukan arah *horizontal* dan *vertikal* dari citra dengan operator *sobel*. Apabila dicontohkan citra yang diolah adalah citra *I* berukuran M x N piksel, dimana piksel dari citra dinotasikan dengan *variabel a* dan *b. Variabel a* memiliki rentang nilai mulai dari 1 hingga M, dan *b* memiliki rentang nilai mulai dari 1 hingga N, piksel *(a, b)* memiliki 8 buah piksel tetangga seperti yang ditunjukkan pada Gambar 2.7. Piksel tetangga tersebut digunakan untuk di konvolusikan menggunakan operator *sobel* yang ditunjukkan pada Gambar 2.6. Operator horizontal komponen digunakan untuk menentukan nilai G_x(a, b), sementara operator vertikal komponen digunakan untuk menemukan nilai G_y(a, b). Persamaan untuk mendapatkan nilai arah horizontal (Gx) dan vertikal

(Gy) ditunjukkan pada Persamaan 2.13 dan Persamaan 2.14 (Aggarwal, Singh, & Singh, 2015).



Gambar 2.6 Operator *Sobel* Horizontal Dan Vertikal

(Aggarwal, Singh, & Singh, 2015)



Gambar 2.7 Piksel Tetangga (a,b)

(Aggarwal, Singh, & Singh, 2015)

$$Gx(a,b) = I(a-1,b-1) + 2 * I(a-1,b) + I(a-1,b+1) - I(a+1,b-1) - 2 * I(a+1,b) - I(a+1,b+1) \quad (2.13)$$

$$Gy(a,b) = I(a-1,b-1) + 2 * I(a,b-1) + I(a+1,b-1) - I(a-1,b+1) - 2 * I(a,b+1) - I(a+1,b+1) \quad (2.14)$$

(Aggarwal, Singh, & Singh, 2015)

2. Menghitung nilai arah gradien piksel (g) dengan menghitung nilai *inverse tangent* dari hasil pembagian gradien vertikal dengan gradien horizontal. Untuk menghitung arah gradien pada piksel (a,b), maka dirumuskan menjadi Persamaan 2.15 berikut (Aggarwal, Singh, & Singh, 2015).

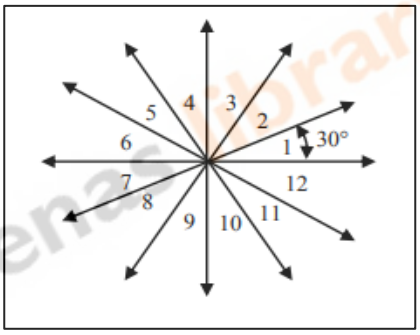
$$g = \tan - 1[Gy(a,b)/Gx(a,b)] \quad (2.15)$$

3. Mencari nilai direksi, nilai g bernilai mulai dari 0 sampai 2π atau setara dengan 6.28, untuk kemudian dicari nilai arahnya berdasarkan Tabel 2.1 berikut.

Tabel 2.1 Nilai direksi

Gradient Values (g)	Direction	Gradient Values (g)	Direction
G = -1	0	π <g <= 7 π/6	7
0 <= g <= π/6	1	7 π/6 < g <= 4 π/3	8
π/6 < g <= π/3	2	4 π/3 <g <= 3 π/2	9
π/2 < g <= 2 π/3	3	3 π/2 <g <= 5 π/3	10
π/2 < g <= 2 π/3	4	5 π/3 < g <= 11 π/6	11
2 π/3 < g <= 5 π/6	5	11 π/6 < g<= 2 π	12
5 π/6 < g <= π	6	--	--

(Sumber: International Journal of Computer Applications)



Gambar 2.8 Pembagian Direksi Kemiringan Piksel

(Putri, Irawan, & Ahmad, 2017)

Nilai-nilai tersebut menjadi ciri tiap piksel yang apabila digabungkan menjadi ciri suatu objek, yang mana pada penelitian ini nilai-nilai tersebut menjadi ciri dari huruf-huruf yang menyusun teks pada citra.

2.7 *Genetic Modified K-Nearest Neighbor (GMKNN)*

Algoritma *Modified K-Nearest Neighbor (MKNN)* merupakan pengembangan dari metode KNN dengan penambahan beberapa proses yaitu, perhitungan nilai validitas dan perhitungan bobot. Algoritma k-nearest neighbor (KNN) merupakan algoritma kalsifikasi yang sangat sederhana dengan cara mengelompokkan data baru dengan K tetangga terdekat (Wafiyah, Hidayat, & Perdana, 2017).

Algoritma *Modified K-Nearest Neighbor (MKNN)* masih belum mampu mengatasi permasalahan kNN dalam hal nilai k yang masih bias. Oleh karena itu perlu dilakukan optimasi menggunakan *Genetic Algorithm* sebagai operator dalam menentukan nilai k yang bertujuan mengatasi kelemahan tersebut (untuk selanjutnya MkNN dengan adanya penambahan operator *Genetic Algorithm* disebut dengan *Genetic Modified K-Nearest Neighbor (GMKNN).* Kemudian algoritma *hybrid* ini digunakan sebagai model pada saat klasifikasi (Mutrofin, Izzah, Kurniawardhani, & Masrur, 2015). Pada algoritma *Genetic Modified K-Nearest Neighbor* terbagi menjadi dua tahapan yaitu tahapan *Genetic Algorithm* dan *Modified K-Nearest Neighbor* dengan tahapan sebagai berikut :

1. Menentukan populasi awal, Misalkan populasi yang diinginkan adalah 5 maka secara *random* membangkitkan kromosom sebanyak 5 buah dengan menggunakan bilangan biner 0 dan 1.
2. Membuat matriks jarak *euclidean* antar data *training* dan matriks *similarity*. perhitungan jarak *euclidean* antar data *training* dilakukan untuk menentukan jarak terdekat data berdasarkan nilai k yang diperoleh dari kromosom menggunakan Persamaan 2.16.

$$d(x_a, x_b) = \sqrt{\sum_{i=1}^{r} (x_{ai} - x_{bi})^2} \tag{2.16}$$

Keterangan :

x_a = nilai ke-a dari data x.

x_b = nilai ke-b dari data y.

x_{ai} = nilai ke-a variabel ke-i dari data x.

x_{bi} = nilai ke-b variabel ke-i dari data y.

r = jumlah variabel.

d = jarak *euclidean.*

Jarak *euclidean* dihitung sampai dengan data *training* terakhir. Kemudian dilakukan perhitungan *similarity* berdasarkan Persamaan 2.17 yang berguna untuk proses perhitungan validitas dalam menentukan nilai *fitness*.

$$S(x_a, x_b) = \begin{cases} 1\ jika\ x_a = x_b \\ 0\ jika\ x_a \neq x_b \end{cases} \quad (2.17)$$

Persamaan 2.17 menyat bahwa jika x dan y yang bertipe nominal atau ordinal maka s(x,y) bernilai 1 jika mempunyai nilai yang sama dan sebaliknya maka bernilai 0.

3. Melakukan perhitungan validitas dan *fitness* populasi. Nilai validitas ini merupakan dasar dari perhitungan nilai *fitness* pada algoritma genetika. Perhitungan validitas dilakukan sebanyak ukuran populasi berdasarkan nilai kromosom dengan Persamaan 2.18.

$$validitas(\text{a}) = \frac{1}{k}\sum_{i=1}^{k} S(target(x_a), taget(n_i(x_b))) \quad (2.18)$$

Keterangan :

$validitas(\text{a}) =$ validitas data *training* ke-a.

k = jumlah tetangga antar data.

S = *similarity* (nilai 1 = terbaik).

$target(x_a)$ = nilai ke-b variabel ke-i dari data y.

$taget(n_i(x_b))$ = label kelas jarak terdekat pada data *training*.

Selanjutnya yaitu menghitung nilai *fitness* yang digunakan sebagai fungsi optimasi dalam penentuan nilai K yang paling optimal. Untuk memperoleh nilai *fitness* dilakukan dengan cara menghitung rata-rata validitas berdasarkan Persamaan 2.19.

$$f_i(\text{x}) = \frac{\sum_{a=1}^{u} validitas(a)}{u} \quad (2.19)$$

Keterangan :

u = jumlah data training.

$f_i(k)$ = fungsi fitness untuk K pada populasi ke-i.

4. Melakukan Penyilangan (*Crossover*), Penyilangan merupakan operator dalam algoritma genetika yang bertujuan untuk melahirkan kromosom baru yang mewarisi sifat induknya sebagaimana proses reproduksi yang terjadi dalam kehidupan alam.
5. Melakukan Mutasi, mutasi merupakan operator dalam algoritma genetika yang bertujuan untuk mengubah gen-gen tertentu dalam sebuah kromosom.
6. Menghitung *FitnessOffspring* dan Evaluasi Model, *Offspring* yang telah dihasilkan oleh proses penyilangan dan mutasi selanjutnya dilakukan perhitungan nilai *fitness* seperti pada perhitungan sebelumnya. Selanjutnya dilakukan evaluasi model dengan menggabungkan individu *offspring* hasil penyilangan dan mutasi dan populasi awal pada satu tabel yang menjadi populasi total dan kemudian dihitung nilai *fitness-nya*.
7. Melakukan Proses Seleksi, Hasil evaluasi model yang telah diurutkan berdasarkan *rangking-nya* kemudian dilakukan proses seleksi pada tiap kromosom yang menjadi populasi baru pada generasi selanjutnya. Pada tahap seleksi ini dipilih populasi sebanyak *popsize* sehingga dari seluruh populasi total maka hanya dipilih sebanyak kromosom yang memiliki nilai *fitness* yang paling tinggi.
8. Penentuan Nilai K Optimal, Penentuan nilai K yang optimal diperoleh dari optimasi yang dilakukan dengan menggunakan algoritma genetika melalui kriteria berhenti yang telah ditentukan dan tercapainya batas generasi yang telah ditentukan.
9. Menghitung jarak *euclidean* data *training* dengan data testing, dengan dilakukan perhitungan jarak *euclidean* data training dan data testing dengan menggunakan persamaan 2.16
10. Menghitung Nilai *Weight Voting*, Perhitungan *weight voting* dilakukan untuk mencari bobot data testing terhadap data *trainingnya*. Dengan menggunakan K optimal yang di dapat maka dicari tetangga terdekat untuk masing-masing data testing. Setelah mengetahui jarak *euclidean* terhadap masing-masing data testing dilakukan perhitungan *weight voting* pada data testing dengan Persamaan 2.20.

$$w(x_a, y_b) = validitas(a) * \frac{1}{d(x_a, y_b) + 0{,}5} \quad (2.20)$$

Keterangan :

w = Bobot antara data *training* dan data testing.

$validitas(a)$ = validitas data *training.*

$d(x_a, y_b)$ = jarak antar data *training.*

11. Melakukan voting pada data testing untuk menentukan kelas prediksi, nilai *weight voting* yang telah diperoleh dari data testing selanjutnya dilakukan voting berdasarkan besarnya *weight voting* tergantung pada kelas data *training-nya* dengan persamaan 2.21.

$$vote(kelas) = \sum_{i=1}^{n} w(x_a, y_b)_n \quad (2.21)$$

Keterangan :

$vote(kelas)$ = voting kelas berdasarkan bobot data testing dan data *training* pada kelas yang sama .

$w(x_a, y_b)_n$ = nilai *weight voting* data *training* dengan data testing ke-i pada kelas yang sama.

Dalam perhitungan voting, fungsi voting ini adalah untuk menentukan data uji itu lebih cenderung ke kelas apa, voting sangat membantu ketika sebuah data uji memiliki kecenderungan pada kelas yang lebih dari satu kelas. Setelah melakukan perhitungan, voting diurutkan dari nilai terbesar ke nilai terkecil dan voting dengan nilai terbesar menjadi kelas yang terpilih (Mutrofin, Izzah, Kurniawardhani, & Masrur, 2015).

2.8 Pengujian Kinerja Sistem

Dalam penelitian ini untuk mengukur kinerja sistem dari klasifikasi citra dengan mengukur *accuracy, precision, recall,* dan *f measure*. *Precision* bagian dari citra yang diambil dengan tepat/relevan. Sedangkan *recall* bagian dari citra yang tepat/relevan yang diambil oleh sistem. Sedangkan *accuracy* merupakan tingkat

kedekatan antara nilai prediksi dengan nilai aktual. *F Measure* merupakan ukuran akurasi uji dari perhitungan *precision* dan *recall* (Pardede & Husada, 2015).

$$Accuracy = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{FP} + \mathrm{TN} + \mathrm{FN}} \quad (2.22)$$

$$Precision = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} \quad (2.23)$$

$$Recall = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \quad (2.24)$$

$$F\ Measure = 2\ x \frac{(\mathrm{PrecisionxRecall})}{(Precision + Recall)} \quad (2.25)$$

Keterangan :

TP *(True Positive) = Correct Result*

FP (*False Positive) = Unexpected Result*

FN (*False Negative) = Missing Result*

TN (*True Negative*) = *Correct Absence of result.*

2.9 Studi Kasus

Studi kasus digunakan untuk memberi contoh perubahan yang terjadi pada piksel-piksel citra setelah dilakukan pengolahan dengan menggunakan metode-metode yang digunakan pada penelitian ini.

2.9.1 Studi Kasus *Grayscale*

Pada tahapan ini citra masukan berupa citra RGB dikonversi menjadi citra *grayscale*. Untuk dapat melakukan tahapan *grayscaling*, dimisalkan citra masukan merupakan citra RGB yang dicuplik dengan ukuran 4x4 piksel dengan matriks piksel seperti pada Tabel 2.2.

Tabel 2.2 Matriks citra RGB

(0,0) R=220 G=204 B=143	**(0,1)** R=219 G=205 B=144	**(0,2)** R=208 G=189 B=130	**(0,3)** R=128 G=110 B=211
(1,0) R=214, G=198, B=136	**(1,1)** R=216 G=202 B=141	**(1,2)** R=240 G=226 B=165	**(1,3)** R=219 G=129 B=115

(2,0) R=224 G=208 B=146	**(2,1)** R=209 G=193 B=133	**(2,2)** R=253 G=243 B=181	**(2,3)** R=221 G=212 B=130

Pada Tabel 2.2 terlihat koordinat 0,0 citra memiliki nilai piksel R = 220, G = 204, B = 143. Dengan menggunakan Persamaan 2.1, maka didapatkan nilai hasil perhitungan citra *grayscale* sebagai berikut :

Perhitungan piksel (0.0) :

Grayscale = 0,21 * 220 + 0,72 * 204 + 0.07 * 143

Grayscale = 46,2 + 146,88 + 10,1

Grayscale = 203.18 = 203

Proses perhitungan tersebut dilakukan sampai semua piksel yang ada pada citra (Tabel 2.2) dihitung menggunakan Persamaan 2.1 sehingga menghasilkan citra *grayscale* dengan nilai piksel *grayscale* seperti pada Tabel 2.3.

Tabel 2.1 Matriks Citra Grayscale

(0,0) 203	**(0,1)** 204	**(0,2)** 189	**(0,3)** 121
(1,0) 197	**(1,1)** 201	**(1,2)** 220	**(1,3)** 147
(2,0) 207	**(2,1)** 192	**(2,2)** 241	**(2,3)** 208

2.9.2 Studi Kasus *Low Pass Filetring*

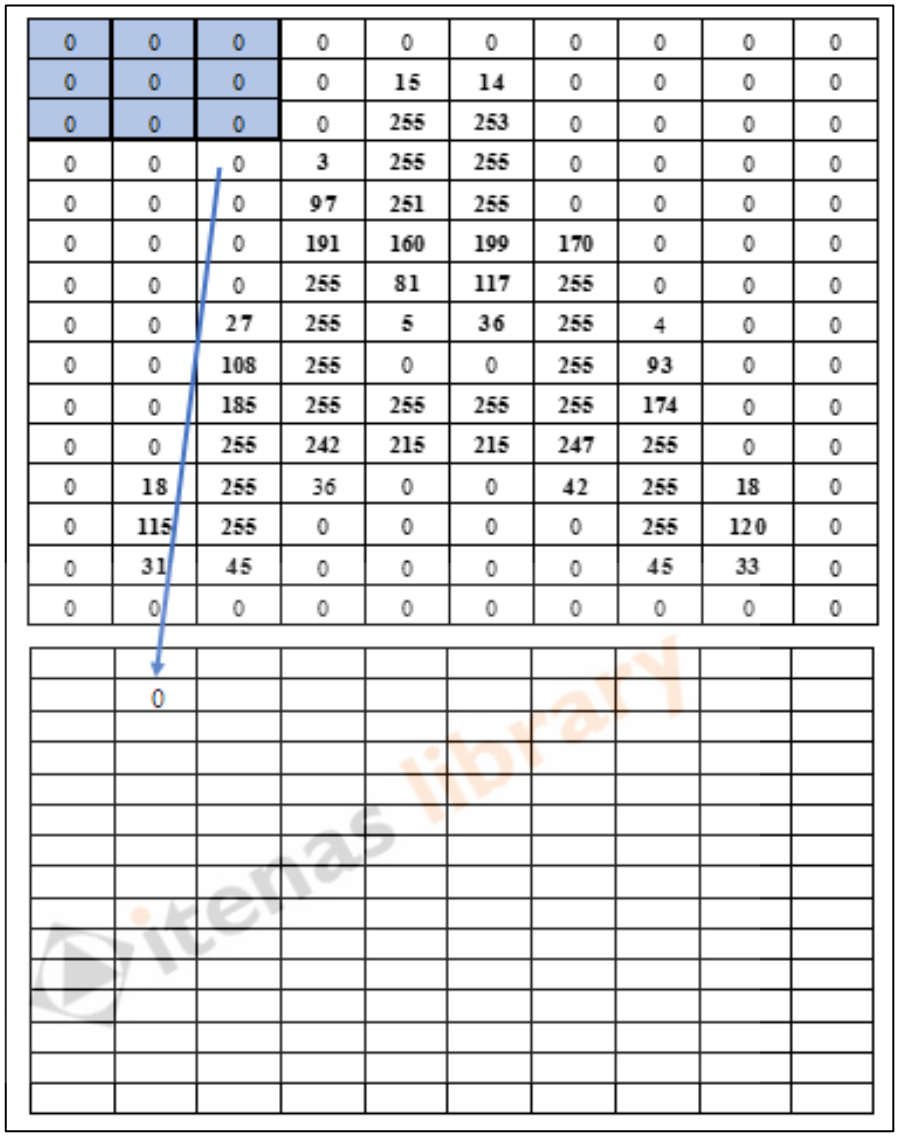
Pada studi kasus Low Pass Filter diperlihatkan perubahan piksel yang menyusun citra awal menjadi citra yang telah difilter dengan metode *Low Pass.* Fungsi *low pass filter* adalah untuk menghaluskan citra sehingga mengurangi *noise* yang terdapat pada citra. Kernel *low pass filter* yang digunakan pada penelitian ditunjukkan pada Persamaan 2.2. Dimisalkan citra yang diolah memiliki ukuran 10 x 15 piksel dengan nilai piksel seperti pada Gambar 2.9 berikut.

0	0	0	0	0	0	0	0	0	0
0	0	0	0	15	14	0	0	0	0
0	0	0	0	255	253	0	0	0	0
0	0	0	3	255	255	0	0	0	0
0	0	0	97	251	255	0	0	0	0
0	0	0	191	160	199	170	0	0	0
0	0	0	255	81	117	255	0	0	0
0	0	27	255	5	36	255	4	0	0
0	0	108	255	0	0	255	93	0	0
0	0	185	255	255	255	255	174	0	0
0	0	255	242	215	215	247	255	0	0
0	18	255	36	0	0	42	255	18	0
0	115	255	0	0	0	0	255	120	0
0	31	45	0	0	0	0	45	33	0
0	0	0	0	0	0	0	0	0	0

Gambar 2.9 Citra input *low pass filter*

Dalam tahapan *low pass filtering*, dilakukan konvolusi menggunakan kernel *low pass* dengan cara seperti berikut :

1. Tempatkan kernel pada bagian sudut kiri atas citra, kemudian kalikan piksel citra dengan piksel pada kernel yang menyentuh citra tersebut, jumlahkan nilai hasil perkalian tersebut lalu simpan pada citra yang menempel dengan koordinat (0,0)

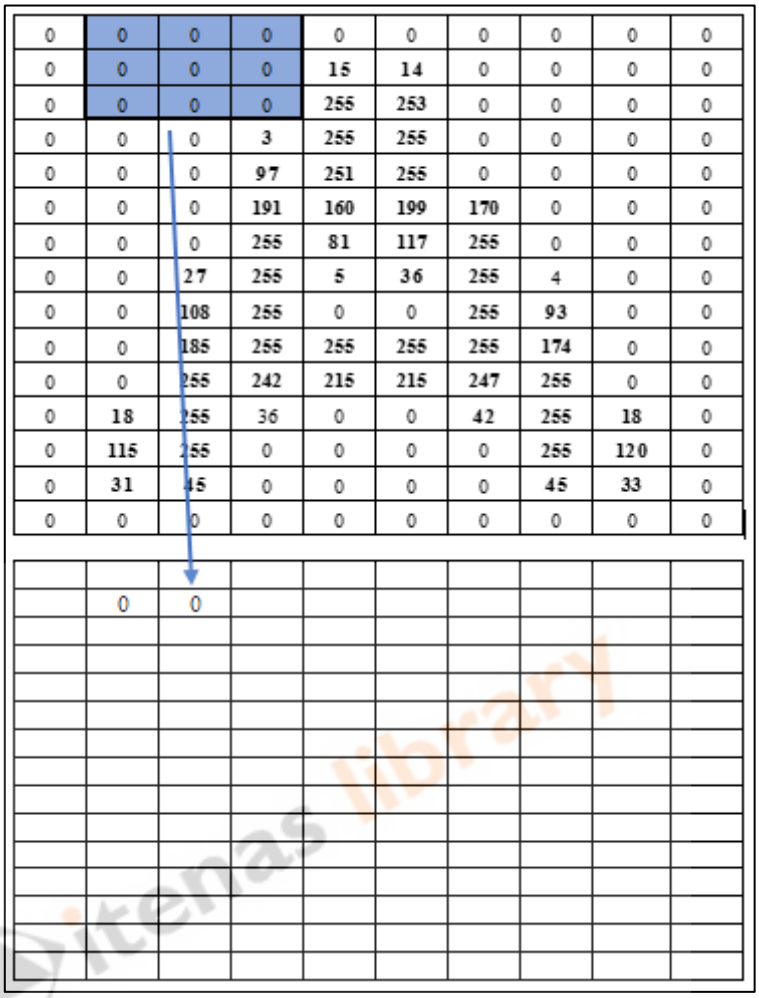


Gambar 2.10 Langkah pertama konvolusi

Seperti yang terlihat pada Gambar 2.10, maka nilai hasil dari konvolusi adalah 0, yang didapatkan dari :

$$lpf(1,1) = \left(0 * \frac{1}{9}\right) + \left(0 * \frac{1}{9}\right) + \left(0 * \frac{1}{9}\right) + \left(0 * \frac{1}{9}\right) + \left(0 * \frac{1}{9}\right) + \left(0 * \frac{1}{9}\right) + \left(0 * \frac{1}{9}\right) + \left(0 * \frac{1}{9}\right) + \left(0 * \frac{1}{9}\right) = 0$$

2. Geser posisi kernel ke kanan, sehingga didapatkan nilai hasil konvolusi untuk kolom selanjutnya. Seperti yang terlihat pada Gambar 2.11.

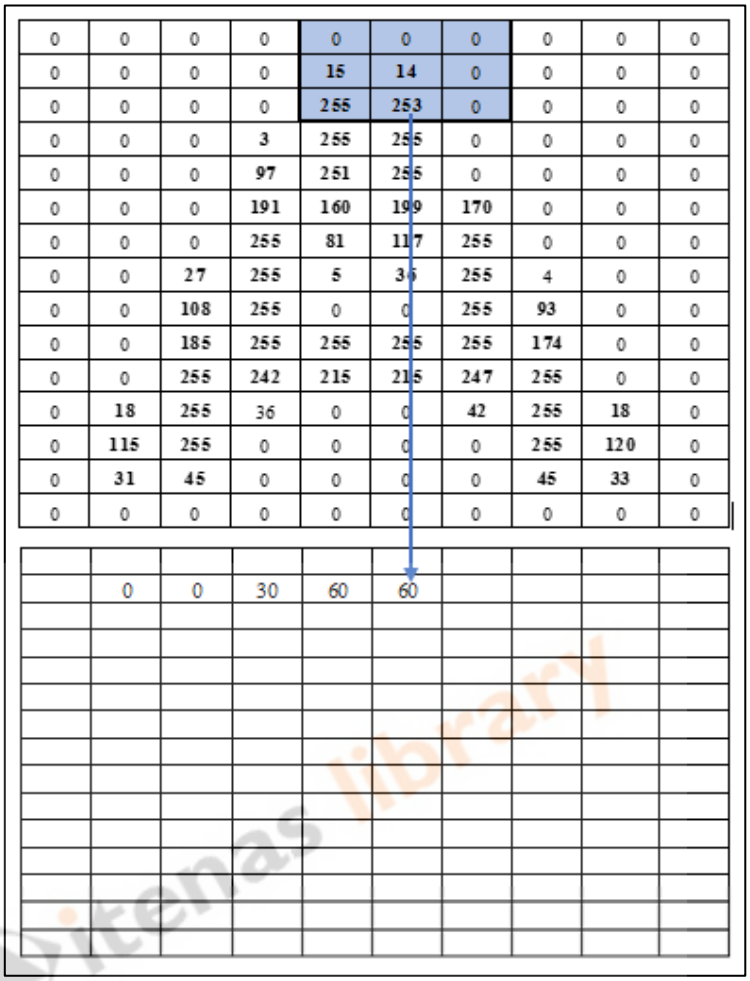


Gambar 2.11 Langkah kedua konvolusi

3. Hasil konvolusi yang didapatkan adalah 0, didapatkan berdasarkan hasil perhitungan berikut.

$$lpf(2,1) = \left(0 * \frac{1}{9}\right) + \left(0 * \frac{1}{9}\right) + \left(0 * \frac{1}{9}\right) + \left(0 * \frac{1}{9}\right) + \left(0 * \frac{1}{9}\right) + \left(0 * \frac{1}{9}\right) + \left(0 * \frac{1}{9}\right) + \left(0 * \frac{1}{9}\right) + \left(0 * \frac{1}{9}\right) = 0$$

4. Sebagai contoh perhitungan dengan nilai piksel citra yang tidak bernilai 0, maka pada Gambar 2.12 berikut terlihat contoh perhitungan pada piksel (2,5). Nilai perhitungan konvolusi mendapatkan nilai 60, yang didapatkan dari perhitungan berikut.



Gambar 2.12 Proses Konvolusi

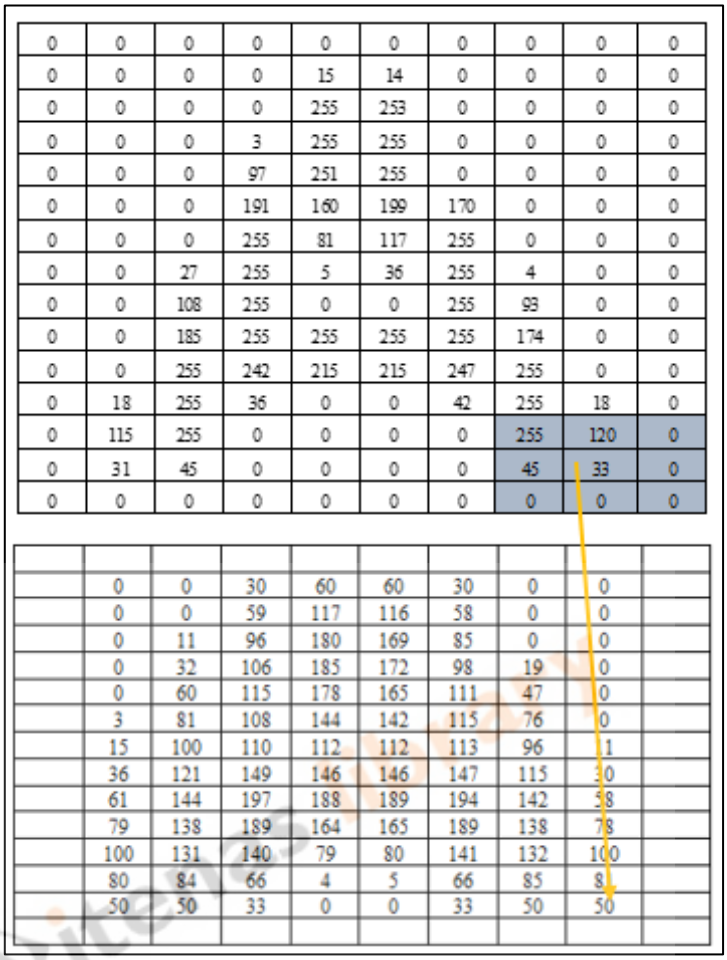
$$lpf(5,1) = \left(0 * \frac{1}{9}\right) + \left(0 * \frac{1}{9}\right) + \left(0 * \frac{1}{9}\right) + \left(15 * \frac{1}{9}\right) + \left(14 * \frac{1}{9}\right) + \left(0 * \frac{1}{9}\right) + \left(255 * \frac{1}{9}\right) + \left(254 * \frac{1}{9}\right) + \left(0 * \frac{1}{9}\right)$$

$$= 1{,}667 + 1{,}556 + 28{,}33 + 28{,}111 = 59{,}664 \approx 60$$

5. Langkah-langkah tersebut terus dilakukan hingga kernel menyentuh sudut kanan bawah dari citra seperti yang ditunjukkan pada Gambar 2.13. Proses perhitungan terakhir menghasilkan nilai 50 yang didapatkan dari perhitungan berikut.

$$lpf(8,13)\left(255 * \frac{1}{9}\right) + \left(120 * \frac{1}{9}\right) + \left(0 * \frac{1}{9}\right) + \left(45 * \frac{1}{9}\right) + \left(33 * \frac{1}{9}\right) + \left(0 * \frac{1}{9}\right) + \left(0 * \frac{1}{9}\right) + \left(0 * \frac{1}{9}\right) + \left(0 * \frac{1}{9}\right) = 28{,}333 + 13{,}333 + 5 + 3{,}667 +$$

$$= 50{,}329 \approx 50$$



0	0	0	0	0	0	0	0	0	0
0	0	0	0	15	14	0	0	0	0
0	0	0	0	255	253	0	0	0	0
0	0	0	3	255	255	0	0	0	0
0	0	0	97	251	255	0	0	0	0
0	0	0	191	160	199	170	0	0	0
0	0	0	255	81	117	255	0	0	0
0	0	27	255	5	36	255	4	0	0
0	0	108	255	0	0	255	93	0	0
0	0	185	255	255	255	255	174	0	0
0	0	255	242	215	215	247	255	0	0
0	18	255	36	0	0	42	255	18	0
0	115	255	0	0	0	0	255	120	0
0	31	45	0	0	0	0	45	33	0
0	0	0	0	0	0	0	0	0	0

0	0	30	60	60	30	0	0	
0	0	59	117	116	58	0	0	
0	11	96	180	169	85	0	0	
0	32	106	185	172	98	19	0	
0	60	115	178	165	111	47	0	
3	81	108	144	142	115	76	0	
15	100	110	112	112	113	96	11	
36	121	149	146	146	147	115	30	
61	144	197	188	189	194	142	58	
79	138	189	164	165	189	138	78	
100	131	140	79	80	141	132	100	
80	84	66	4	5	66	85	81	
50	50	33	0	0	33	50	50	

Gambar 2.13 Langkah terakhir konvolusi

6. Sehingga, nilai akhir proses konvolusi pada tahapan *Low Pass Filtering* ditunjukkan pada Gambar 2.14 berikut.

0	0	30	60	60	30	0	0	
0	0	59	117	116	58	0	0	
0	11	96	180	169	85	0	0	
0	32	106	185	172	98	19	0	
0	60	115	178	165	111	47	0	
3	81	108	144	142	115	76	0	
15	100	110	112	112	113	96	11	
36	121	149	146	146	147	115	30	
61	144	197	188	189	194	142	58	
79	138	189	164	165	189	138	78	
100	131	140	79	80	141	132	100	
80	84	66	4	5	66	85	81	
50	50	33	0	0	33	50	50	

Gambar 2.14 Hasil tahapan *Low Pass Filtering*

Terdapat baris dan kolom yang tidak terjamah kernel pada tiap-tiap sisi paling kanan, bawah, kiri, dan atas dari citra. Terdapat beberapa solusi untuk mengatasi hal tersebut, di antaranya dengan mengabaikannya, memberi nilai yang sama dengan piksel sebelahnya, dan menambah ukuran citra menggunakan konstanta atau nilai lain sehingga konvolusi pada pinggir citra dapat dilakukan (Gazali, Haryono, & Ohliati, 2012). Pada penelitian ini, nilai pada tepian citra yang tidak terproses konvolusi diberi nilai 0, karena pada tahapan-tahapan selanjutnya citra menjadi bernilai 0 dan 1, sehingga tepian citra tersebut menjadi *background* dari citra.

2.9.3 Studi Kasus *Sharpening (High Pass Filtering)*

Proses *sharpening* dilakukan untuk menajamkan kembali citra yang sebelumnya telah dihaluskan. Kernel *sharpening* yang digunakan pada penelitian adalah kernel *laplacian* yang ditunjukkan pada Persamaan 2.3. Sebagai studi kasus tahapan *sharpening* dimisalkan dilakukan konvolusi kernel tersebut pada citra hasil dari tahapan *low pass filtering* yang nilai pikselnya ditunjukkan pada Gambar 2.15.

	0	0	30	60	60	30	0	0	
	0	0	59	117	116	58	0	0	
	0	11	96	180	169	85	0	0	
	0	32	106	185	172	98	19	0	
	0	60	115	178	165	111	47	0	
	3	81	108	144	142	115	76	0	
	15	100	110	112	112	113	96	11	
	36	121	149	146	146	147	115	30	
	61	144	197	188	189	194	142	58	
	79	138	189	164	165	189	138	78	
	100	131	140	79	80	141	132	100	
	80	84	66	4	5	66	85	81	
	50	50	33	0	0	33	50	50	

Gambar 2.15 Citra input *sharpening*

Proses perhitungan citra setelah dikonvolusikan baris pertama dari citra dikonvolusikan menghasilkan nilai berikut :

hpf(1,1) = 0*-1+0*-1+0*-1+0*-1+0*9+0*-1+0*-1+0*-1+0*-1 = 0

hpf(1,2) = 0*-1+0*-1+0*-1+0*-1+0*9+30*-1+0*-1+0*-1+59*-1 = -89

hpf(1,3) = 0*-1+0*-1+0*-1+0*-1+30*9+60*-1+0*-1+59*-1+117*-1 = 34

hpf(1,4) = 0*-1+0*-1+0*-1+30*-1+60*9+60*-1+59*-1+117*-1+116*-1 = 158

hpf(1,5) = 0*-1+0*-1+0*-1+60*-1+60*9+30*-1+117*-1+116*-1+58*-1 = 159

hpf(1,6) = 0*-1+0*-1+0*-1+60*-1+30*9+0*-1+116*-1+58*-1+0*-1 = 36

hpf(1,7) = 0*-1+0*-1+0*-1+30*-1+0*9+0*-1+58*-1+0*-1+0*-1 = -88

hpf(1,8) = 0*-1+0*-1+0*-1+0*-1+0*9+0*-1+0*-1+0*-1+0*-1 = 0

Sementara, baris terakhir dari citra yang dikonvolusikan menghasilkan nilai berikut:

hpf(13,1) = 0*-1+80*-1+84*-1+0*-1+50*9+50*-1+0*-1+0*-1+0*-1 = 236

hpf(13,2) = 80*-1+84*-1+66*-1+50*-1+50*9+33*-1+0*-1+0*-1+0*-1 = 137

hpf(13,3) = 84*-1+66*-1+4*-1+50*-1+33*9+0*-1+0*-1+0*-1+0*-1 = 93

hpf(13,4) = 66*-1+4*-1+5*-1+33*-1+0*9+0*-1+0*-1+0*-1+0*-1 = -108

hpf(13,5) = 4*-1+5*-1+66*-1+0*-1+0*9+33*-1+0*-1+0*-1+0*-1 = -108

hpf(13,6) = 5*-1+66*-1+85*-1+0*-1+33*9+50*-1+0*-1+0*-1+0*-1 = 91

hpf(13,7) = 66*-1+85*-1+81*-1+33*-1+50*9+50*-1+0*-1+0*-1+0*-1 = 135

hpf(13,8) = 85*-1+81*-1+0*-1+50*-1+50*9+0*-1+0*-1+0*-1+0*-1 = 234

Secara keseluruhan, piksel hasil dari proses konvolusi ditunjukkan pada Gambar 2.16 berikut.

	0	-89	34	158	159	36	-88	0	
	-11	-196	37	283	285	62	-173	0	
	-43	-194	174	600	510	133	-260	-19	
	-103	-100	97	484	377	114	-170	-66	
	-176	95	141	465	340	165	4	-142	
	-229	218	72	254	228	173	191	-230	
	-206	277	29	-49	-57	68	257	-218	
	-117	277	223	111	113	216	244	-152	
	31	326	534	347	362	515	329	19	
	137	201	520	249	261	520	208	132	
	388	303	405	-102	-93	409	310	386	
	305	106	73	-367	-358	68	112	312	
	236	137	93	-108	-108	91	135	234	

Gambar 2.16 Hasil konvolusi *sharpening*

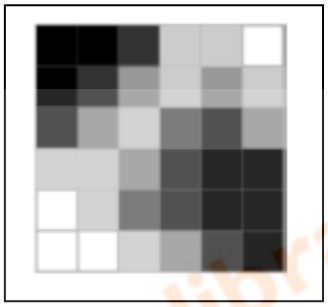
Apabila dilihat dari hasil proses konvolusi, terdapat nilai negatif serta nilai yang memiliki nilai lebih dari nilai keabuan maksimum yaitu 255. Oleh karena itu, tiap-tiap piksel yang memiliki nilai kurang dari 0 diubah nilainya menjadi bernilai 0 (intensitas minimum keabuan), dan nilai yang lebih dari 255 dirubah nilainya menjadi bernilai 255. Sehingga citra yang dihasilkan memiliki nilai seperti yang ditampilkan pada Gambar 2.17 berikut.

0	0	0	0	0	0	0	0	0	0
0	0	0	**34**	**158**	**159**	**36**	0	0	0
0	0	0	**37**	**255**	**255**	**62**	0	0	0
0	0	0	**174**	**255**	**255**	**133**	0	0	0
0	0	0	**97**	**255**	**255**	**114**	0	0	0
0	0	**95**	**141**	**255**	**255**	**165**	4	0	0
0	0	**218**	**72**	**254**	**228**	**173**	**191**	0	0
0	0	**255**	**29**	0	0	**68**	**255**	0	0
0	0	**255**	**223**	**111**	**113**	**216**	**244**	0	0
0	**31**	**255**	**255**	**255**	**255**	**255**	**255**	**19**	0
0	**137**	**201**	**255**	**249**	**255**	**255**	**208**	**132**	0
0	**255**	**255**	**255**	0	0	**255**	**255**	**255**	0
0	**255**	**106**	**73**	0	0	**68**	**112**	**255**	**0**
0	**236**	**137**	**93**	0	0	**91**	**135**	**234**	**0**
0	0	0	0	0	0	0	0	0	0

Gambar 2.17 Hasil akhir proses *sharpening*

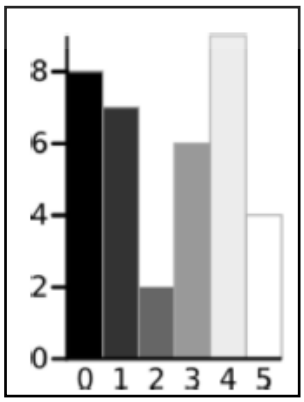
2.9.4 Studi Kasus *Otsu Tresholding*

Pada studi kasus ini, dilakukan pengimplementasian metode *Otsu* dalam mencari nilai *threshold* terbaik pada citra. Pada penelitian ini citra yang menjadi masukan pada tahapan *Otsu thresholding* merupakan citra hasil *sharpening*. Studi kasus ini menjelaskan bagaimana sistem dapat menentukan nilai hasil *threshold* terbaik, namun hanya pada citra yang memiliki 6 level keabuan, yaitu bernilai 0 hingga 5 dengan menggunakan studi kasus yang didapatkan dari buku elektronik yang ditulis oleh (Greensted, 2010)



Gambar 2.18 Citra input *otsu tresholding*

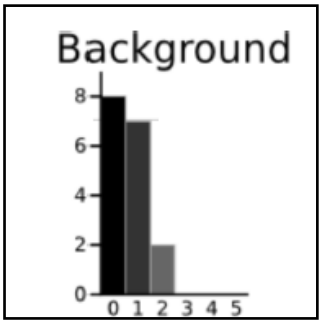
Gambar 2.18 menunjukan citra intensitas piksel pada citra berderajat keabuan yang dicari nilai *threshold* terbaiknya. Sementara pada Gambar 2.19 ditunjukan sebaran intensitas piksel pada citra yang direpresentasikan dalam bentuk histogram.



Gambar 2.19 *Histogram* penyebaran intensitas piksel

Metode *Otsu thresholding* bekerja dengan cara mencari nilai *variance* atau nilai sebaran terkecil dari nilai *background* dan nilai *foreground* dari citra yang dipisahkan oleh nilai-nilai piksel pada citra. Dimisalkan dilakukan pencarian nilai

variances pada nilai *threshold = 3*. Maka, didapatkan histogram nilai *background* seperti yang ditunjukkan pada Gambar 2.20.



Gambar 2.20 *Histogram* background citra

Dari histogram tersebut, ditunjukkan bahwa piksel bernilai 0 berjumlah 8 piksel, piksel bernilai 1 berjumlah 7 piksel, dan piksel bernilai 2 berjumlah 2 piksel. Dari keterangan tersebut, didapatkan nilai *weight background* (W_b) dengan menggunakan Persamaan 2.6 dan *mean background* (μ_b) dengan menggunakan Persamaan 2.7 Berikut merupakan contoh perhitungan dari persamaan dan keterangan tersebut.

$$Weight\ Background(W_b) = \frac{8+7+2}{36} = 0{,}4722$$

$$Mean\ Background(\mu_b) = \frac{(0\ *\ 8\ ) + (1 * 7) + (2 * 2)}{17} = 0{,}6471$$

Setelah mendapatkan nilai *weight background* (W_b) dan *mean background* (μ_b) dilanjutkan dengan mencari nilai *variance background* $(\sigma^2{}_b)$ dengan menggunakan Persamaan 2.5 . Berikut merupakan salah satu contoh perhitungannya.

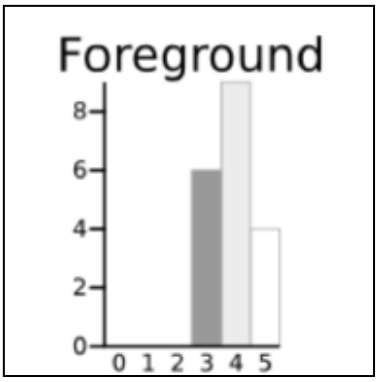
$$Variance\ Background(\sigma^2{}_b)$$

$$= \frac{((0-0{,}6471)^2 * 8) + ((1-0{,}6471)^2 * 7) + ((2-0{,}6471)^2 * 2)}{17}$$

$$= \frac{(0{,}4187 * 8) + (0{,}1246 * 7) + (1{,}8304 * 2)}{17} = 0{,}4637$$

Selanjutnya yaitu mencari nilai *variance foreground* $(\sigma^2{}_f)$. Untuk mencari nilai *variance foreground* $(\sigma^2{}_f)$, maka perlu didapatkan nilai dari *weight*

foreground (W_f) dan *mean foreground* (μ_f). Gambar 2.11 berikut menunjukkan *histogram* nilai piksel *foreground* apabila nilai *threshold* yang digunakan adalah 3.



Gambar 2.21 *Histogram* Foreground Citra

Dari Gambar 2.21 tersebut ditunjukkan bahwa piksel berintensitas 3 sebanyak 6 piksel, piksel bernilai 4 sebanyak 9 piksel, dan piksel bernilai 5 sebanyak 4 piksel. Maka didapatkan nilai-nilai *weight foreground* (W_f) dengan menggunakan Persamaan 2.8 dan *mean foreground* (μ_f) dengan menggunakan Persamaan 2.9 yang digunakan untuk mencari nilai *variance foreground* (${\sigma^2}_f$) dengan menggunakan Persamaan 2.11.

$$Weight\ Foreground(W_{f)} = \frac{6+9+4}{36} = 0{,}5278$$

$$Mean\ Foreground(\mu_f) = \frac{(3\ *\ 6\ ) + (4*9) + (5*4)}{19} = 3{,}8947$$

$$Variance\ Foregrond({\sigma^2}_f)$$

$$= \frac{((3-3{,}8947)^2 * 6) + ((4-3{,}8947)^2 * 9) + ((5-3{,}8947)^2 * 4)}{19}$$

$$= \frac{(4{,}8033 * 6) + (0{,}0997 * 9) + (4{,}8864 * 4)}{19} = 0{,}5152$$

Langkah selanjutnya adalah mencari nilai *Within Class Variance* (${\sigma^2}_W$) dengan menggunakan Persamaan 2.4 dan nilai *Between Class Variance* (${\sigma^2}_B$) dengan menggunakan Persamaan 2.5. Berikut merupakan hasil perhitungan yang dilakukan menggunakan kedua persamaan tersebut pada studi kasus ini.

$$Within\ Class\ Variance\ (\sigma^2{}_W) = (W_b.\sigma^2{}_b) + (W_f.\sigma^2{}_f)$$
$$= (0{,}4722 * 0{,}4637) + (0{,}5278 * 0{,}5152) = 0{,}4909$$

$$\text{Between Class Variance } (\sigma^2{}_B) = W_b W_f (\mu_f - \mu_b)^2$$
$$= 0{,}4722 * 0{,}5278(3{,}8947 - 0{,}6471)^2 = 2{,}6287$$

Perhitungan-perhitungan tersebut dilakukan terhadap semua kemungkinan nilai piksel mulai dari nilai 0 hingga nilai 5. Hasil perhitungan tersebut ditunjukan sebagai berikut (Tabel 2.4).

Tabel 2.1 Hasil kemungkinan setiap nilai *threshold*

Threshold	T=0	T=1	T=2	T=3	T=4	T=5
Weight Background (W_b)	0	0,22	0,42	0,47	0,64	0,89
Mean Background (μ_b)	0	0	0,42	0,65	1,26	2,03
Variance Background ($\sigma^2{}_b$)	0	0	0,25	0,46	1,41	2,53
Weight Foreground ($W^2{}_{f)}$	1	0,78	0,59	0,53	0,36	0,11
Mean Foreground (μ_f)	2,36	3,04	3,72	3,89	4,31	5,00
Variance foreground ($\sigma^2{}_f$)	3,12	1,96	0,78	0,51	0,21	0
Within Class Variance ($\sigma^2{}_W$)	3,12	1,53	0,56	0,49	0,97	2,25
Between Class Variance ($\sigma^2{}_B$)	0	1,59	2,56	2,63	2,14	0,87

Nilai *threshold* terbaik dipilih berdasarkan nilai yang memiliki hasil perhitungan *within class variance* terkecil dan memiliki nilai *between class variance* terbesar. Pada Tabel 2.4, nilai *within class variance* terkecil dan memiliki nilai between *class variance terbesar* dimiliki oleh nilai *threshold* = 3, maka pada studi kasus memiliki nilai *threshold* = 3.

Studi kasus selanjutnnya yaitu mengkonversi citra derajat keabuan (*grayscale)* menjadi citra biner (*threshold)*. Dimisalkan citra memiliki matriks piksel yang ditunjukan pada Gambar 2.22.

0	0	0	0	0	0	0	0	0	0
0	0	0	30	60	60	30	0	0	0
0	0	0	59	117	116	58	0	0	0
0	0	11	96	180	169	85	0	0	0
0	0	32	106	185	172	98	19	0	0
0	0	60	115	178	165	111	47	0	0
0	3	81	108	144	142	115	76	0	0
0	15	100	110	112	112	113	96	11	0
0	36	121	149	146	146	147	115	30	0
0	61	144	197	188	189	194	142	58	0
0	79	138	189	164	165	189	138	78	0
0	100	131	140	79	80	141	132	100	0
0	80	84	66	4	5	66	85	81	0
0	50	50	33	0	0	33	50	50	0
0	0	0	0	0	0	0	0	0	0

Gambar 2.22 Citra nilai *grayscale*

Pada studi kasus ini dimisalkan hasil dari *Otsu thresholding* menghasilkan nilai *threshold* terbaik yaitu 50. Maka dengan menggunakan Persamaan 2.12, didapatkan persamaan untuk membagi citra kedalam dua kelas sebagai berikut :

$$f(x,y) = \begin{cases} 1, apabila\ f(x,y) > 50 \\ 0, apabila\ f(x,y) \leq 50 \end{cases}$$

Nilai piksel yang lebih besar dari 50 bernilai 0 (hitam), sebaliknya nilai piksel yang lebih kecil dari 50 bernilai 1 (putih), sehingga menghasilkan matriks citra biner yang di tunjukan pada Gambar 2.23.

0	0	0	0	0	0	0	0	0	0
0	0	0	0	255	255	0	0	0	0
0	0	0	255	255	255	255	0	0	0
0	0	0	255	255	255	255	0	0	0
0	0	0	255	255	255	255	0	0	0
0	0	255	255	255	255	255	0	0	0
0	0	255	255	255	255	255	255	0	0
0	0	255	255	255	255	255	255	0	0
0	0	255	255	255	255	255	255	0	0
0	255	255	255	255	255	255	255	255	0
0	255	255	255	255	255	255	255	255	0
0	255	255	255	255	255	255	255	255	0
0	255	255	255	0	0	255	255	255	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

Gambar 2.23 Matriks citra hasil *thresholding*

2.9.5 Studi Kasus Segmentasi *Mathematical Morphology*

Pada studi kasus ini, dilakukan implementasi dari tahapan segmentasi dengan tujuan menentukan bagian objek dan menghilangkan bagian citra yang tidak diperlukan. Pada penelitian ini citra masukan untuk melakukan segmentasi menggunakan morfologi matematika merupakan citra hasil *thresholding* seperti pada Gambar 2.22. Dimisalkan citra hasil *thresholding* dengan ukuran matriks 10x15 piksel seperti pada Gambar 2.24 yang merupakan *input* dari tahapan segmentasi.

0	0	0	0	0	0	0	0	0	0
0	0	0	0	255	255	0	0	0	0
0	0	0	255	255	255	255	0	0	0
0	0	0	255	255	255	255	0	0	0
0	0	0	255	255	255	255	0	0	0
0	0	255	255	255	255	255	0	0	0
0	0	255	255	255	255	255	255	0	0
0	0	255	255	255	255	255	255	0	0
0	0	255	255	255	255	255	255	0	0
0	255	255	255	255	255	255	255	255	0
0	255	255	255	255	255	255	255	255	0
0	255	255	255	255	255	255	255	255	0
0	255	255	255	0	0	255	255	255	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

Gambar 2.24 Citra input tahapan segmentasi

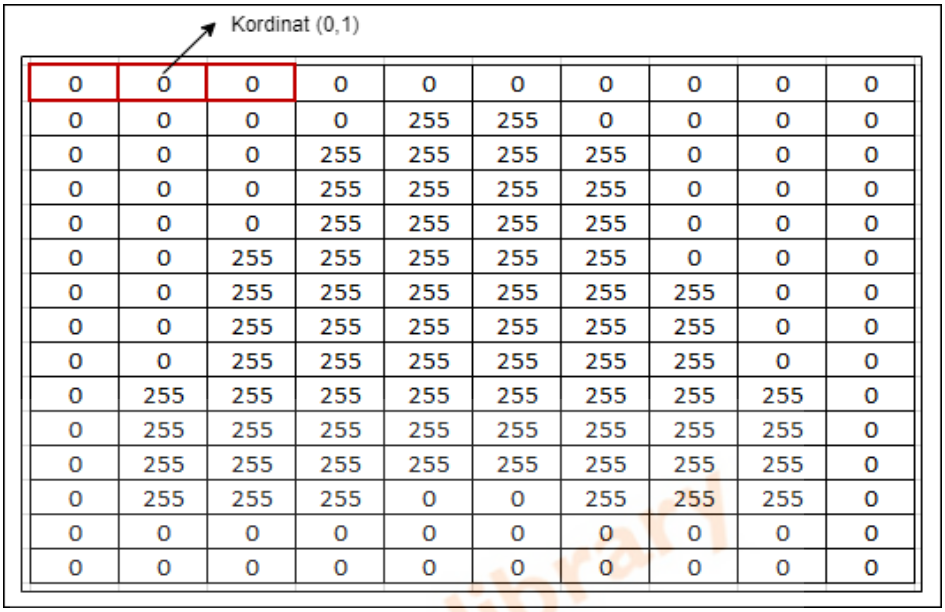
Metode Segmentasi *Mathematical Morphology* bekerja dengan cara melakukan morfologi dilasi. Dimisalkan pada studi kasus ini dilasi dilakukan dengan menggunakan kernel 1x3 pada piksel yang bernilai 255 seperti pada Gambar 2.25.

255	255	255

Gambar 2.25 Kernel segmentasi 1x3

Pada Gambar 2.25 merupakan kernel untuk melakukan proses morfologi dilasi pada matriks citra yang direpresentasikan oleh Gambar 2.26. Berikut merupakan tahapan yang dilakukan untuk melakukan operasi morfologi dilasi pada citra.

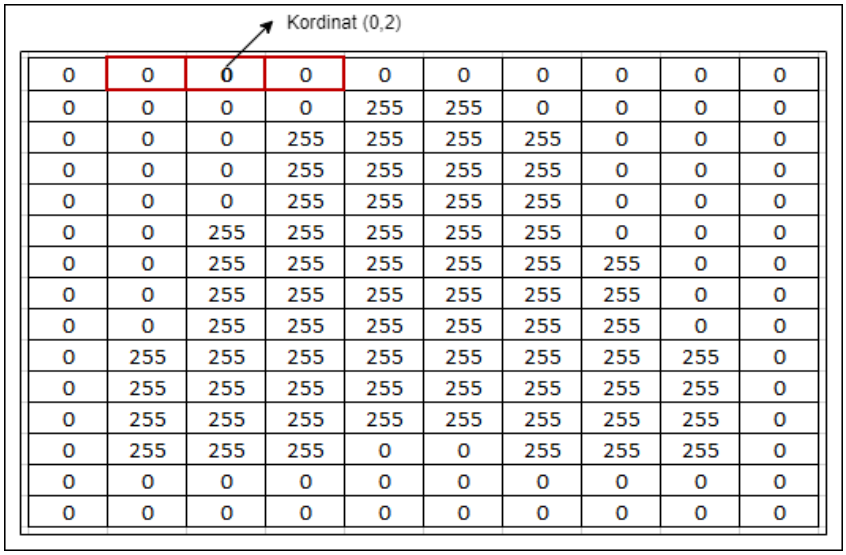
1. Tempatkan kernel pada bagian sudut kiri atas citra seperti pada Gambar 2.26, sehingga titik tengah kernel (*hotspot*) berada pada kordinat (0,1) dan lakukan pemeriksaan apakah titik tengah bernilai 255 dan tetangga piksel bernilai 0.



Gambar 2.26 Tahapan *morfologi* dilasi pada kordinat *hotspot* (0,1)

Pada Gambar 2.26 nilai piksel pada *hotspot* (kordinat 0,1) bernilai 0 sehingga piksel tetangga pada jendela 1x3 tidak dilakukan proses morfologi dilasi.

2. Selanjutnya geser posisi kernel ke kanan sehingga posisi *hotspot* berada pada kordinat (0,2) seperti pada Gambar 2.27 dan lakukan pemeriksaan seperti langkah sebelumnya.

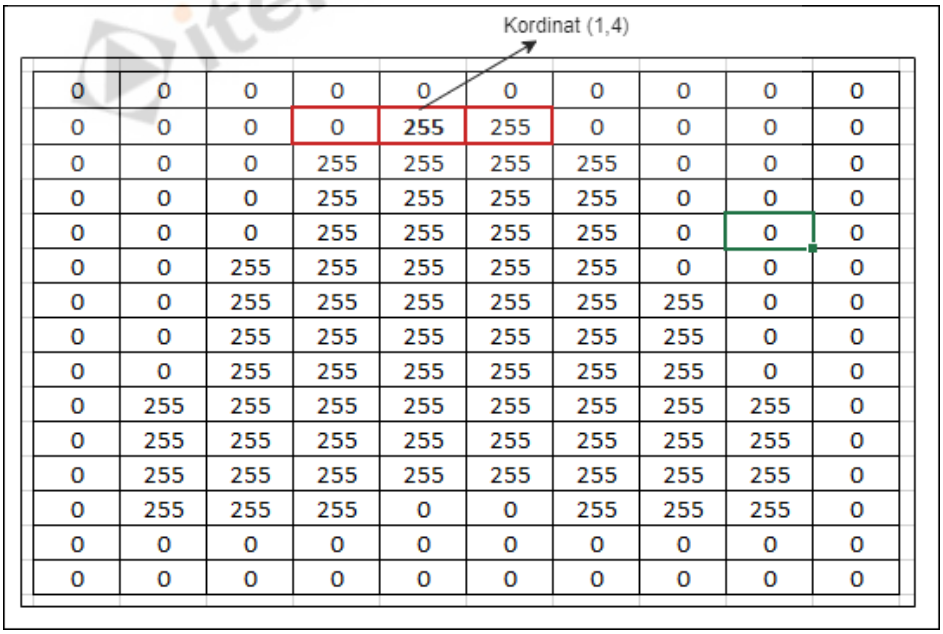


0	0	0	0	0	0	0	0	0	0
0	0	0	0	255	255	0	0	0	0
0	0	0	255	255	255	255	0	0	0
0	0	0	255	255	255	255	0	0	0
0	0	0	255	255	255	255	0	0	0
0	0	255	255	255	255	255	0	0	0
0	0	255	255	255	255	255	255	0	0
0	0	255	255	255	255	255	255	0	0
0	0	255	255	255	255	255	255	0	0
0	255	255	255	255	255	255	255	255	0
0	255	255	255	255	255	255	255	255	0
0	255	255	255	255	255	255	255	255	0
0	255	255	255	0	0	255	255	255	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

Gambar 2.27 Tahapan *morfologi* dilasi pada kordinat *hotspot* (0,2)

Sama seperti tahapan sebelumnya pada Gambar 2.20 nilai piksel pada *hotspot* bernilai 0 sehingga piksel tetangga pada jendela 1x3 tidak melakukan konversi.

3. Sebagai contoh tahapan morfologi dilasi yang tidak memiliki nilai 0, maka dilakukan tahapan dilasi pada kordinat *hotspot* (1,4) seperti pada Gambar 2.28.



0	0	0	0	0	0	0	0	0	0
0	0	0	0	255	255	0	0	0	0
0	0	0	255	255	255	255	0	0	0
0	0	0	255	255	255	255	0	0	0
0	0	0	255	255	255	255	0	0	0
0	0	255	255	255	255	255	0	0	0
0	0	255	255	255	255	255	255	0	0
0	0	255	255	255	255	255	255	0	0
0	0	255	255	255	255	255	255	0	0
0	255	255	255	255	255	255	255	255	0
0	255	255	255	255	255	255	255	255	0
0	255	255	255	255	255	255	255	255	0
0	255	255	255	0	0	255	255	255	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

Gambar 2.28 Tahapan *morfologi* pada kordinat *hotspot* (1,4)

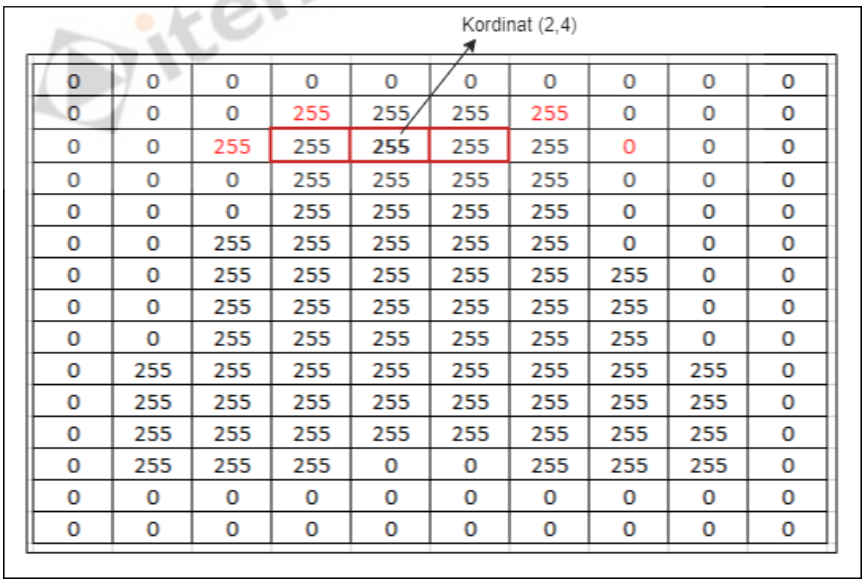
Pada Gambar 2.28 nilai piksel pada *hotspot* bernilai 255 namun piksel tetangga sebelah kiri *hotspot* bernilai 0, maka pada tahapan ini dilakukan

proses konversi piksel pada piksel tetangga sebelah kiri *hotspot* menjadi 255 sehingga menghasilkan matriks piksel seperti pada Gambar 2.29.

0	0	0	0	0	0	0	0	0	0
0	0	0	255	**255**	255	0	0	0	0
0	0	0	255	255	255	255	0	0	0
0	0	0	255	255	255	255	0	0	0
0	0	0	255	255	255	255	0	0	0
0	0	255	255	255	255	255	0	0	0
0	0	255	255	255	255	255	255	0	0
0	0	255	255	255	255	255	255	0	0
0	0	255	255	255	255	255	255	0	0
0	255	255	255	255	255	255	255	255	0
0	255	255	255	255	255	255	255	255	0
0	255	255	255	255	255	255	255	255	0
0	255	255	255	0	0	255	255	255	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

Gambar 2.29 Hasil *morfologi* pada kordinat *hotspot* (1,4)

4. Sebagai contoh tahapan morfologi dilasi yang memiliki kedua tetangga bernilai 255, maka dilakukan tahapan dilasi pada kordinat *hotspot* (2,4) seperti pada Gambar 2.30.

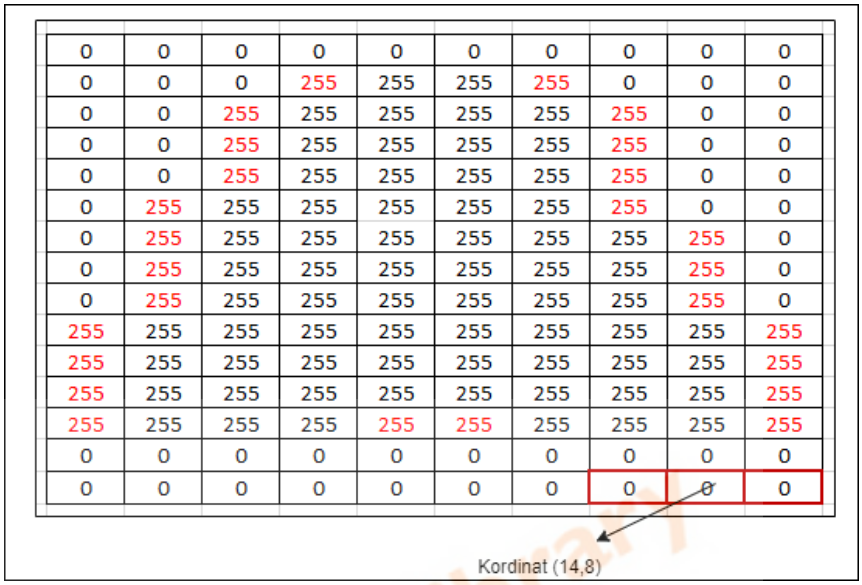


0	0	0	0	0	0	0	0	0	0
0	0	0	255	255	255	255	0	0	0
0	0	255	255	**255**	255	255	0	0	0
0	0	0	255	255	255	255	0	0	0
0	0	0	255	255	255	255	0	0	0
0	0	255	255	255	255	255	0	0	0
0	0	255	255	255	255	255	255	0	0
0	0	255	255	255	255	255	255	0	0
0	0	255	255	255	255	255	255	0	0
0	255	255	255	255	255	255	255	255	0
0	255	255	255	255	255	255	255	255	0
0	255	255	255	255	255	255	255	255	0
0	255	255	255	0	0	255	255	255	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

Gambar 2.30 Hasil *morfologi* pada kordinat *hotspot* (2,4)

Pada Gambar 2.30 nilai piksel pada *hotspot* bernilai 255 namun piksel tetangga sebelah kiri *hotspot* bernilai 255, maka pada tahapan ini tidak dilakukan proses konversi piksel pada kedua piksel tetangga.

5. Langkah-langkah tersebut terus dilakukan hingga kernel menyentuh sudut kanan bawah dari citra dengan koordinat *hotspot* (14,8) seperti yang ditunjukkan pada Gambar 2.31.



Gambar 2.31 Hasil *morfologi* akhir pada kordinat *hotspot* (14,8)

6. Hasil akhir proses konvolusi dari kordinat *hotspot* (0,1) sampai dengan (14,8) pada tahapan *low pass filtering* ditunjukkan pada Gambar 2.32.

0	0	0	0	0	0	0	0	0	0
0	0	0	255	255	255	255	0	0	0
0	0	255	255	255	255	255	255	0	0
0	0	255	255	255	255	255	255	0	0
0	0	255	255	255	255	255	255	0	0
0	255	255	255	255	255	255	255	0	0
0	255	255	255	255	255	255	255	255	0
0	255	255	255	255	255	255	255	255	0
0	255	255	255	255	255	255	255	255	0
255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

Gambar 2.32 Hasil proses *morfologi* dilasi menggunakan kernel (1,3)

Setelah dilakukan morfologi dilasi, pada tahapan segmentasi selanjutnya yaitu melakukan tahapan *plot contour* untuk menentukan batas vertikal dan horziontal dari objek. Objek yang memiliki nilai piksel 255 pada bagian ujung

merupakan *contour* yang dijadikan acuan untuk dilakukan segmentasi seperti pada Gambar 2.33.

0	0	0	0	0	0	0	0	0	0
0	0	0	255	255	255	255	0	0	0
0	0	255	255	255	255	255	255	0	0
0	0	255	255	255	255	255	255	0	0
0	0	255	255	255	255	255	255	0	0
0	255	255	255	255	255	255	255	0	0
0	255	255	255	255	255	255	255	255	0
0	255	255	255	255	255	255	255	255	0
0	255	255	255	255	255	255	255	255	0
255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

Gambar 2.33 Penentuan batas vertikal dan horizontal

Seperti pada Gambar 2.33, sistem menarik garis lurus sejajar sumbu y dan sejajar sumbu x, yang merupakan batas dari objek sesuai dengan titik terluar piksel yang bernilai 255. Sehingga pada tahapan ini membentuk sebuah batas yang direpresentasikan oleh garis merah pada Gambar 2.30 berbentuk segi empat. Wilayah yang dibungkus oleh garis merah merupakan acuan untuk dilakukan pemotongan, sehingga pada tahapan ini wilayah diluar garis merah dibuang dan menghasilkan matriks citra hasil segmentasi seperti yang ditunjukan pada Gambar 2.34.

0	0	0	255	255	255	255	0	0	0
0	0	255	255	255	255	255	255	0	0
0	0	255	255	255	255	255	255	0	0
0	0	255	255	255	255	255	255	0	0
0	255	255	255	255	255	255	255	0	0
0	255	255	255	255	255	255	255	255	0
0	255	255	255	255	255	255	255	255	0
0	255	255	255	255	255	255	255	255	0
255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255

Gambar 2.34 Citra hasil segmentasi

2.9.6 Studi Kasus *Directional Feature Extraction*

Pada tahapan ekstraksi ciri, citra yang digunakan sebagai *input* adalah citra hasil segmentasi huruf. Dimisalkan citra hasil segmentasi adalah Gambar 2.35 berikut.

0	0	0	**76**	**255**	**255**	**76**	0	0	0
0	0	0	**76**	**255**	**255**	**76**	0	0	0
0	0	**85**	**195**	**255**	**255**	**195**	**85**	0	0
0	0	**60**	**160**	**255**	**255**	**160**	**60**	0	0
0	0	**51**	**148**	**255**	**255**	**148**	**51**	0	0
0	**48**	**127**	**207**	**255**	**2555**	**255**	**161**	**48**	0
0	**178**	**127**	**66**	**221**	**221**	**221**	**238**	**178**	0
0	**179**	**128**	0	0	0	0	**128**	**179**	0
0	**178**	**238**	**155**	0	0	**155**	**238**	**178**	0
0	**178**	**255**	**235**	**187**	**187**	**235**	**255**	**178**	0
0	**178**	**255**	**255**	**255**	**255**	**255**	**255**	**178**	0
119	**214**	**255**	**219**	**136**	**136**	**219**	**255**	**214**	**119**
255	**195**	**170**	**119**	0	0	**119**	**170**	**195**	**255**
204	**61**	0	0	0	0	0	0	**61**	**204**
0	0	0	0	0	0	0	0	0	0

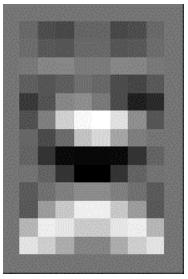
Gambar 2.35 Citra input directional feature extraction

Proses pertama yang dilakukan terhadap citra adalah melakukan konvolusi dengan menggunakan 2 buah kernel sobel seperti yang telah dijelaskan pada Gambar 2.6. Proses konvolusi dilakukan dengan cara yang sama pada studi kasus sebelumnya, yaitu *low pass filtering* dan *sharpening*. Piksel – piksel hasil konvolusi horizontal ditunjukkan pada Gambar 2.36 berikut.

0	0	0	0	0	0	0	0	0	0
0	-85	-289	-323	-119	-119	-323	-289	-85	0
0	-60	-204	-228	-84	-84	-228	-204	-60	0
0	34	115	128	47	47	128	115	34	0
0	-163	-229	-161	-2347	-4695	-2591	-345	-197	0
0	-432	-248	122	184	29	-299	-625	-543	0
0	-263	74	668	3272	5620	3098	190	-229	0
0	-111	-311	-68	574	729	353	66	0	0
0	-125	-488	-784	-796	-796	-784	-488	-125	0
0	-17	-134	-472	-865	-865	-472	-134	-17	0
0	-191	-20	83	169	169	83	-20	-191	0
0	-204	289	612	901	901	612	289	-204	0
0	476	882	829	627	627	829	882	476	0
0	815	654	408	119	119	408	654	815	0
0	0	0	0	0	0	0	0	0	0

Gambar 2.36 Piksel hasil konvolusi kernel horizontal

Sementara, secara kasat mata, citra hasil konvolusi kernel horizontal tersebut terlihat seperti pada Gambar 2.37.



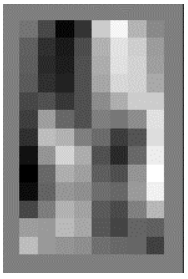
Gambar 2.37 Citra hasil konvolusi sobel horizontal

Selain kernel horizontal, citra yang memiliki piksel pada Gambar 2.6 dikonvolusikan terhadap kernel *sobel* vertikal dan menghasilkan piksel-piksel seperti yang ditampilkan pada Gambar 2.38.

0	0	0	0	0	0	0	0	0	0
0	-85	-423	-935	-597	597	935	423	85	0
0	-230	-626	-790	-394	394	790	626	230	0
0	-256	-663	-764	-357	357	764	663	256	0
0	-289	-615	-731	-2657	309	2997	663	323	0
0	-432	-354	-554	-4958	107	4975	605	611	0
0	-509	244	-188	-2658	0	2232	114	765	0
0	-621	493	400	0	-155	-511	-338	732	0
0	-859	168	672	358	-358	-672	-168	859	0
0	-1003	-168	374	251	-251	-374	168	1003	0
0	-901	-216	187	131	-131	-187	216	901	0
0	-442	-11	408	285	-285	-408	11	442	0
0	238	208	459	321	-321	-459	-208	-238	0
0	493	198	170	119	-119	-170	-198	-493	0
0	0	0	0	0	0	0	0	0	0

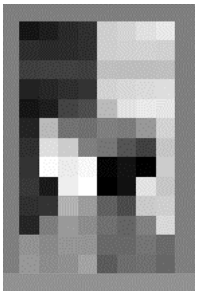
Gambar 2.38 Piksel hasil konvolusi kernel vertikal

Piksel-piksel tersebut membentuk suatu citra yang dapat dilihat pada Gambar 2.39.



Gambar 2.39 Citra hasil konvolusi sobel vertikal

Dua buah citra hasil konvolusi terhadap kernel *sobel* horizontal dan kernel vertikal tersebut kemudian digunakan untuk mencari *gradient* dengan Persamaan 8. Dimana *Gx* adalah hasil pengkonvolusian kernel vertikal, dan *Gy* adalah hasil konvolusi kernel horizontal. Persamaan tersebut menghasilkan citra seperti yang ditunjukkan pada Gambar 2.40 berikut.



Gambar 2.40 Citra hasil gradien sobel

0	0	0	0	0	0	0	0	0	0
0	-2,3562	-2,1702	-1,9034	-1,7675	1,76755	1,90342	2,17017	2,35619	0
0	-1,826	-1,8858	-1,8518	-1,7808	1,78085	1,85177	1,88582	1,82598	0
0	-1,4388	-1,3991	-1,4048	-1,4399	1,4399	1,4048	1,39905	1,43876	0
0	-2,0843	-1,9272	-1,7876	-2,2943	3,07587	2,28367	2,0506	2,11847	0
0	-2,3562	-2,1819	-1,354	-1,5337	1,30613	1,63082	2,37245	2,29734	0
0	-2,0477	1,27633	-0,2743	-0,6822	0	0,62433	0,54042	1,86165	0
0	-1,7477	2,13358	1,73919	0	-0,2095	-0,9663	-1,378	1,5708	0
0	-1,7153	2,81004	2,43297	2,71895	-2,7189	-2,433	-2,81	1,7153	0
0	-1,5877	-2,2441	2,47152	2,85918	-2,8592	-2,4715	2,24408	1,58774	0
0	-1,7797	-1,6631	1,15307	0,6594	-0,6594	-1,1531	1,66313	1,77969	0
0	-2,0032	-0,038	0,588	0,30636	-0,3064	-0,588	0,03804	2,0032	0
0	0,46365	0,2316	0,50566	0,47317	-0,4732	-0,5057	-0,2316	-0,46365	0
0	0,54402	0,29398	0,39479	0,7854	-0,7854	-0,3948	-0,294	-0,54402	0
0	0	0	0	0	0	0	0	0	0

Gambar 2.41 Citra berisi gradien piksel

Hasil dari citra yang berisi *gradient* tiap-tiap piksel pada Gambar 2.41 tersebut digeneralisasikan ke dalam 12 arah sesuai aturan yang tertera pada Tabel 2.1. Hasil dari proses tersebut ditunjukkan pada Gambar 2.42.

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	4	4	5	5	0
0	0	0	0	0	4	4	4	4	0
0	0	0	0	0	3	3	3	3	0
0	0	0	0	0	6	5	4	5	0
0	0	0	0	0	3	4	5	5	0
0	0	3	0	0	0	2	2	4	0
0	0	5	4	0	0	0	0	4	0
0	0	6	5	6	0	0	0	4	0
0	0	0	5	6	0	0	5	4	0
0	0	0	3	2	0	0	4	4	0
0	0	0	2	1	0	0	1	4	0
0	1	1	1	1	0	0	0	0	0
0	2	1	1	2	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

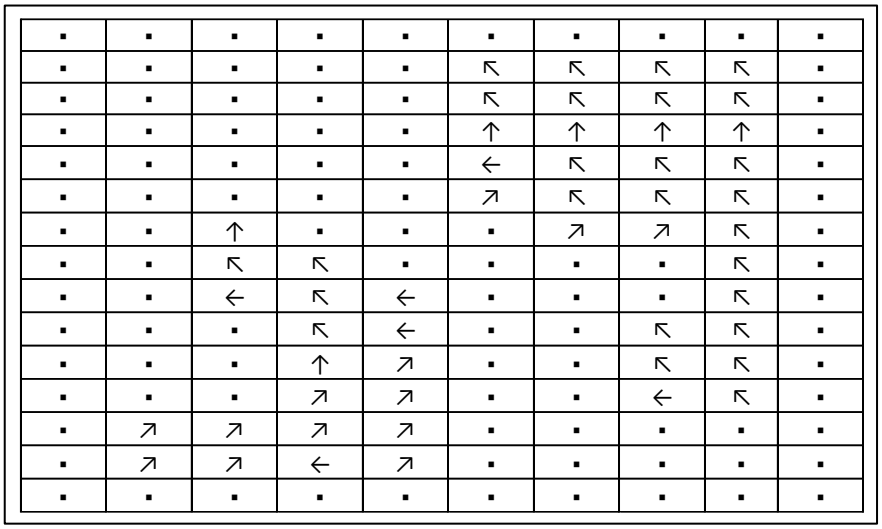
Gambar 2.42 Ciri generalisasi arah

Atau, apabila secara derajat arah, maka ciri tersebut ditunjukkan pada Gambar 2.43.

0 °	0 °	0 °	0 °	0 °	0 °	0 °	0 °	0 °	0 °
0 °	0 °	0 °	0 °	0 °	120 °	120 °	150 °	150 °	0 °
0 °	0 °	0 °	0 °	0 °	120 °	120 °	120 °	120 °	0 °
0 °	0 °	0 °	0 °	0 °	90 °	90 °	90 °	90 °	0 °
0 °	0 °	0 °	0 °	0 °	180 °	150 °	120 °	150 °	0 °
0 °	0 °	0 °	0 °	0 °	90 °	120 °	150 °	150 °	0 °
0 °	0 °	90 °	0 °	0 °	0 °	60 °	60 °	120 °	0 °
0 °	0 °	150 °	120 °	0 °	0 °	0 °	0 °	120 °	0 °
0 °	0 °	180 °	150 °	180 °	0 °	0 °	0 °	120 °	0 °
0 °	0 °	0 °	150 °	180 °	0 °	0 °	150 °	120 °	0 °
0 °	0 °	0 °	90 °	60 °	0 °	0 °	120 °	120 °	0 °
0 °	0 °	0 °	60 °	30 °	0 °	0 °	30 °	120 °	0 °
0 °	30 °	30 °	30 °	30 °	0 °	0 °	0 °	0 °	0 °
0 °	60 °	30 °	30 °	60 °	0 °	0 °	0 °	0 °	0 °
0 °	0 °	0 °	0 °	0 °	0 °	0 °	0 °	0 °	0 °

Gambar 2.43 Ciri generalisasi derajat arah

Apabila ciri arah derajat tersebut, digeneralisasikan kembali ke dalam 4 arah mata angin, maka didapatkan arah-arah dari piksel seperti yang ditunjukkan pada Gambar 2.44 berikut.



Gambar 2.44 Ciri generalisasi arah mata angin

2.9.7 Studi Kasus *Genetic Modified K-Nearest Neighbor (GMKNN)*

Setelah Ciri dari objek yang dihasilkan dari tahapan ekstraksi ciri kemudian disimpan dalam format *file* csv untuk di jadikan dataset yang digunakan untuk mengidentifikasi. Sebagai contoh, pada studi kasus ini dibuat sebagai 3 label yaitu label *Arial*, *Raleway*, dan *Montserrat*. Dengan ciri seharusnya 255 ciri, namun pada studi kasus ini dicontohkan menggunakan 3 ciri yang dihasilkan pada *Directional Feature Extraction*

Tabel 2.5 Data Training

No	Ciri	Kelas
1.	[1 3 2]	Arial
2.	[1 2 3]	Arial
3.	[1 3 1]	Raleway
4.	[1 2 2]	Raleway
5.	[2 5 1]	Montserrat
6.	[3 7 1]	Montserrat

Selanjutnya melakukan tahap awal yaitu algoritma genetika. Sebelum melakukan tahap awal perlu dilakukan penentuan parameter yang digunakan.

1. Menentukan populasi awal, Misalkan populasi yang diinginkan adalah 3 maka secara acak membangkitkan kromosom sebanyak 3 buah dengan menggunakan bilangan biner 0 dan 1 dan hasil acak sebagai berikut.

Tabel 2.6 Insialisasi Populasi

Individu	Kromosom	Biner
P1	2	010
P2	4	100
P3	3	011

2. Membuat matriks jarak *euclidean* antar data *training* dan matriks *similarity*, perhitungan jarak *euclidean* antar data *training* dilakukan untuk menentukan jarak terdekat data berdasarkan nilai k yang diperoleh dari kromosom menggunakan Persamaan 2.16.
   - Jarak data *training* 1 ke data *training* 1.

     $d(x_1, x_1) = \sqrt{(1-1)^2 + (3-3)^2 + (2-2)^2}$

     $d(x_1, x_1) = 0$
   - Jarak data *training* 1 ke data *training* 2.

     $d(x_1, x_2) = \sqrt{(1-1)^2 + (3-2)^2 + (2-3)^2}$

     $d(x_1, x_2) = 1.414214$
   - Jarak data *training* 1 ke data *training* 3.

     $d(x_1, x_3) = \sqrt{(1-1)^2 + (3-3)^2 + (2-1)^2}$

     $d(x_1, x_3) = 1$
   - Jarak data *training* 1 ke data *training* 4.

     $d(x_1, x_4) = \sqrt{(1-1)^2 + (3-2)^2 + (2-2)^2}$

     $d(x_1, x_4) = 1$
   - Jarak data *training* 1 ke data *training* 5.

     $d(x_1, x_5) = \sqrt{(1-2)^2 + (3-5)^2 + (2-1)^2}$

     $d(x_1, x_5) = 2.44949$

- Jarak data *training* 1 ke data *training* 6.

  $d(x_1, x_6) = \sqrt{(1-3)^2 + (3-7)^2 + (2-1)^2}$

  $d(x_1, x_6) = 4.582576$

Jarak *euclidean* dihitung sampai dengan data *training* terakhir. Kemudian dilakukan perhitungan *similarity* berdasarkan Persamaan 2.17 yang berguna untuk proses perhitungan validitas dalam menentukan nilai *fitness*.

- *Similarity* data *training* 1 ke data *training* 1.

  $S(x_1, x_1) = Arial = Arial$

  $S(x_1, x_1) = 1$

- *Similarity* data *training* 1 ke data *training* 2.

  $S(x_1, x_2) = Arial = Arial$

  $S(x_1, x_2) = 1$

- *Similarity* data *training* 1 ke data *training* 3.

  $S(x_1, x_3) = Arial = Raleway$

  $S(x_1, x_3) = 0$

- *Similarity* data *training* 1 ke data *training* 4.

  $S(x_1, x_4) = Arial = Raleway$

  $S(x_1, x_4) = 0$

- *Similarity* data *training* 1 ke data *training* 5.

  $S(x_1, x_5) = Arial = Montserrat$

  $S(x_1, x_5) = 0$

- *Similarity* data *training* 1 ke data *training* 6.

  $S(x_1, x_6) = Arial = Montserrat$

  $S(x_1, x_6) = 0$

Adapun tabel nilai perhitungan jarak *euclidean* antar data *training* dapat dilihat pada Tabel 2.7.

Tabel 2.7 Perhitungan Jarak *Euclidean* antar Data *Training*

Data Training	1	2	3	4	5	6
1	0	1.414214	1	1	2.44949	4.582576
2	1.414214	0	2.236068	1	3.741657	5.744563
3	1	2.236068	0	1.414214	2.236068	4.472136
4	1	1	1.414214	0	3.316625	5.477226
5	2.44949	3.741657	2.236068	3.316625	0	2.236068
6	4.582576	5.744563	4.472136	5.477226	2.236068	0

Adapun tabel nilai perhitungan dan matriks *similarity* antar data *training* dapat dilihat pada Tabel 2.8.

Tabel 2.8 Perhitungan *Similarity* antar Data *Training*

Data Training	1	2	3	4	5	6
1	1	1	0	0	0	0
2	1	1	0	0	0	0
3	0	0	1	1	0	0
4	0	0	1	1	0	0
5	0	0	0	0	1	1
6	0	0	0	0	1	1

3. Melakukan perhitungan validitas dan *fitness* populasi. Nilai validitas ini merupakan dasar dari perhitungan nilai *fitness* pada algoritma genetika. Perhitungan validitas dilakukan sebanyak ukuran populasi berdasarkan nilai kromosom. Sebagai contoh dilakukan perhitungan dengan menggunakan kromosom pertama yaitu 3 dengan Persamaan 2.18. Tabel nilai perhitungan validitas data *training* Tabel 2.9.

$$validitas(1) = \frac{1}{3}(1 + 0 + 0) \; = \; 0{,}3$$

$$validitas(2) = \frac{1}{3}(1 + 0 + 0) = 0{,}3$$

$$validitas(3) = \frac{1}{3}(1 + 0 + 0) = 0{,}3$$

$$validitas(4) = \frac{1}{3}(1 + 0 + 0) = 0{,}3$$

$$validitas(5) = \frac{1}{3}(1 + 0 + 0) = 0{,}3$$

$$validitas(6) = \frac{1}{3}(1 + 0 + 0) = 0{,}3$$

Adapun tabel nilai perhitungan validitas data *training* dari seluruh kromosom pada Tabel 2.9.

Tabel 2.9 Perhitungan Validitas Data Training

Data Training	K = 2 Validitas	K= 4 Validitas	K = 3 Validitas
1	0,5	0,25	0,3
2	0,5	0,25	0,3
3	0,5	0,25	0,3
4	0,5	0,25	0,3
5	0,5	0,25	0,3
6	0,5	0,25	0,3

Selanjutnya yaitu menghitung nilai *fitness* yang digunakan sebagai fungsi optimasi dalam penentuan nilai K yang paling optimal. Nilai K optimal dicapai dengan syarat berhenti jika nilai *fitness* yang mencapai 0,9 dan dengan maksimal generasi sebanyak 5. Untuk memperoleh nilai *fitness* dilakukan dengan cara menghitung rata-rata validitas berdasarkan Persamaan 2.19, sebagai contoh perhitungan fungsi *fitness* populasi pertama dengan K=3 adalah sebagai berikut :

- Nilai Fitness K = 2

$$f_i(\mathrm{K}=3) = \frac{0{,}5+0{,}5+0{,}5+0{,}5+0{,}5+0{,}5}{6}$$

$$f_i(\mathrm{K}=3) = \frac{3}{6}$$

$$f_i(\mathrm{K}=3) = 0{,}5$$

- Nilai Fitness K = 4

$$f_i(\mathrm{K}=4) = \frac{0{,}25+0{,}25+0{,}25+0{,}25+0{,}25+0{,}25}{6}$$

$$f_i(\mathrm{K}=4) = \frac{1{,}5}{6}$$

$$f_i(\mathrm{K}=4) = 0{,}25$$

- Nilai Fitness K = 3

$$f_i(\mathrm{K}=5) = \frac{0{,}3+0{,}3+0{,}3+0{,}3+0{,}3+0{,}3}{6}$$

$$f_i(\mathrm{K}=6) = \frac{1{,}8}{6}$$

$$f_i(\mathrm{K}=7) = 0{,}3$$

Adapun tabel nilai perhitungan validitas data *training* dari seluruh kromosom pada Tabel 2.10.

Tabel 2.10 Perhitungan Fungsi Fitness

Individu	Kromosom	*Fitness*
P1	2	0,5
P2	4	0,25
P3	3	0,3

4. Melakukan Penyilangan (*Crossover*), Penyilangan merupakan operator dalam algoritma genetika yang bertujuan untuk melahirkan kromosom baru yang mewarisi sifat induknya sebagaimana proses reproduksi yang terjadi dalam kehidupan alam. Berdasarkan perhitungan maka *offspring* yang diperoleh adalah sebanyak 1 individu. Proses Penyilangan disajikan pada Tabel 2.11.

Tabel 2.11 Hasil Proses Penyilangan (Crossover)

Parent	Penyilangan	Hasil	Nilai
2 dan 2	010 dan 010	010	2
2 dan 3	010 dan 011	011	3

5. Melakukan Mutasi, mutasi merupakan operator dalam algoritma genetika yang bertujuan untuk mengubah gen-gen tertentu dalam sebuah kromosom. Dilakukan mutasi yang dipilih secara acak individu pada populasi yang dilakukan mutasi yang dapat dilihat pada Tabel 2.12.

Tabel 2.12 Hasil Proses Mutasi

Parent	Mutasi	Hasil	Nilai
3	011	100	4
3	011	100	4

6. Menghitung *FitnessOffspring* dan Evaluasi Model, *Offspring* yang telah dihasilkan oleh proses penyilangan dan mutasi selanjutnya dilakukan perhitungan nilai *fitness* seperti pada perhitungan sebelumnya. Selanjutnya dilakukan evaluasi model dengan menggabungkan individu *offspring* hasil penyilangan dan mutasi dan populasi awal pada satu tabel yang menjadi populasi total dan kemudian dihitung nilai *fitness-nya* dapat dilihat pada Tabel 2.13.

Tabel 2.13 Hasil Evaluasi Model yang Telah Diurutkan

Individu	Kromosom	*Fitness*
C1	2	0,56
P1	2	0,5
C2	3	0,38
P3	3	0,3

Individu	Kromosom	*Fitness*
M1	4	0,28
M2	4	0,28
M1	4	0,28

7. Melakukan Proses Seleksi, Hasil evaluasi model yang telah diurutkan berdasarkan *rangking-nya* kemudian dilakukan proses seleksi pada tiap kromosom yang menjadi populasi baru pada generasi selanjutnya. Pada tahap seleksi ini dipilih populasi sebanyak *popsize* sehingga dari seluruh populasi total maka hanya dipilih sebanyak kromosom yang memiliki nilai *fitness* yang paling tinggi dapat dilihat pada Tabel 2.14.

Tabel 2.14 Hasil Seleksi Elitisme

Individu	Kromosom	*Fitness*
P1	2	0,5
C1	2	0,56
C2	3	0,38

8. Penentuan Nilai K Optimal, Penentuan nilai K yang optimal diperoleh dari optimasi yang dilakukan dengan menggunakan algoritma genetika melalui kriteria berhenti yang telah ditentukan yaitu 5 generasi. Setelah proses algoritma genetika dihentikan maka diperoleh populasi baru yang memiliki nilai *fitness* yang paling tinggi. Nilai K optimal berdasarkan tahapan algoritma genetika adalah 2 dengan nilai *fitness* 0,56.
9. Selanjutnya menghitung jarak *euclidean* data *training* dengan data testing. Berikut ini merupakan data testing yang dapat dilihat pada Tabel 2.15.

Tabel 2.15 Data Testing

No	Ciri
1.	[2 7 5]

Dilakukan perhitungan jarak *euclidean* data *training* dan data *testing* dengan menggunakan persamaan 2.16.

- Jarak data *testing* ke data *training* 1.

$$d(x_1, x_1) = \sqrt{(2-1)^2 + (7-3)^2 + (5-2)^2}$$

$$d(x_1, x_1) = 5.09902$$

- Jarak data *testing* ke data *training* 2.

$$d(x_1, x_2) = \sqrt{(2-1)^2 + (7-2)^2 + (5-3)^2}$$

$$d(x_1, x_2) = 5.477226$$

Adapun tabel nilai perhitungan jarak *euclidean* data *training* dan data *testing* pada Tabel 2.16

Tabel 2.16 Perhitungan Jarak Euclidean Data Training dan Data Testing

Data Training	Data Test
1	5.09902
2	5.477226
3	5.744563
4	5.91608
5	4.472136
6	4.123106

10. Menghitung Nilai *Weight Voting*, Perhitungan *weight voting* dilakukan untuk mencari bobot data testing terhadap data *trainingnya*. Dengan menggunakan K optimal yaitu 2 yang di dapat maka dicari tetangga terdekat untuk masing-masing data testing dan jarak terdekat adalah data training ke 6 dan 5. Setelah mengetahui jarak *euclidean* terhadap data testing dilakukan perhitungan *weight voting* pada data testing dengan Persamaan 2.20.

- *Weight Voting data testing ke data training ke 6*

$$w(x_6, y_1) = validitas(6) * \frac{1}{d(x_6, y_1) + 0{,}5}$$

$$w(x_6, y_1) = 0{,}5 * \frac{1}{4.123106 + 0{,}5}$$

$$w(x_6, y_1) = 0{,}243$$

- *Weight Voting data testing ke data training ke 5*

$$w(x_5, y_1) = validitas(5) * \frac{1}{d(x_5, y_1) + 0{,}5}$$

$$w(x_5, y_1) = 0{,}5 * \frac{1}{4.472136 + 0{,}5}$$

$$w(x_5, y_1) = 0{,}224$$

Adapun tabel nilai perhitungan jarak *euclidean* data *training* dan data *testing* berdasarkan nilai k pada Tabel 2.17

Tabel 2.17 Jarak Euclidean Terdekat Data Testing berdasarkan nilai k

Data Testing	Jarak terdekat pada data training	Kelas data training
1	6	Montserrat
1	5	Montserrat

11. Melakukan voting pada data testing untuk menentukan kelas prediksi, nilai *weight voting* yang telah diperoleh dari data testing selanjutnya dilakukan voting berdasarkan besarnya *weight voting* tergantung pada kelas data *training-nya* dengan persamaan 2.21. Karena K optimal K = 2 maka terdapat dua nilai *weight voting* untuk masing-masing kelas dan kelas yang memiliki nilai voting tertinggi menjadi kelas data testing.

- *Voting* kelas *arial*

$$vote(arial) = w(x_1 y_1) + w(x_2 y_1)$$

$$vote(arial) = 0$$

- *Voting* kelas *raleway*

$$vote(raleway) = w(x_3 y_1) + w(x_4 y_1)$$

$$vote(raleway) = 0$$

- *Voting* kelas *montserrat*

  $vote(montserrat) = w(x_5 y_1) + w(x_6 y_1)$

  $vote(montserrat) = 0{,}243 + \ 0{,}224$

  $vote(montserrat) = 0{,}467$

12. Berdasarkan hasil perhitungan *weight voting* hasil predikisi dari data testing pada Tabel 2.15 adalah termasuk kelas *montserrat.*