Jurnal Sistem Komputer dan Informatika (JSON)
Volume 2, Nomor 1, September 2020
e-ISSN 2685-998X
DOI 10.30865/json.v2i1.2458

Penerapan Algoritma Filtering Derivatif Untuk Penajaman Kualitas Pada Citra

Khairul

Program Studi Teknik Informatika, Universitas Budi Darma, Medan, Indonesia
Email: khairul95mtb@gmail.com
Submitted: **02/09/2020**; Accepted: **27/09/2020**; Published: **30/09/2020**

Abstrak–Algoritma filtering derivatif adalah perumusan konsep turunan matematik diawali dengan memahami konsep-konsep aplikasi kalkulus khususnya teori differensial (turunan) dalam rumusan digital. Metode yang digunakan dalam perumusan teori ini adalah mencari, menghimpun dan mempelajari beberapa teori yang terkait dengan differensial dan aplikasinya. Teknik derivasi dapat dijelaskan sebagai penghitungan laju nilai kecerahan pada citra digital diskrit, atau dikenal sebagai teknik derivasi citra. Penajaman akan diperoleh dengan menghitung beda nilai kecerahan antara nilai kecerahan yang bersebelahan dalam kolom dan baris yang sama. Citra merupakan hasil keseluruhan dari suatu sistem perekaman data. Secara teoritis citra dapat dikelompokkan menjadi 2 (dua) macam, yaitu citra kontinu dan citra diskrit (citra digital). Citra kontinu dihasilkan dari sistem optik yang menerima sinyal analog. Contoh: mata manusia, kamera analog. Sedangkan citra digital dihasilkan melalui proses digitalisasi terhadap citra kontinu. Contoh: kamera digital.

Kata Kunci: Pengolahan Citra, Penajaman, Kualitas, Algoritam Filtering Derivatif.

Abstract–Derivative filtering algorithm is the formulation of mathematical derivative concepts beginning with understanding the concepts of calculus applications, especially differential theory (derivatives) in digital formulations. The method used in the formulation of this theory is to seek, collect and study several theories related to differences and their applications. The derivation technique can be described as calculating the rate of brightness values in a discrete digital image, also known as an image derivation technique. Sharpening will be obtained by calculating the difference in brightness values between adjacent brightness values in the same column and row. Image is the overall result of a data recording system. Theoretically, images can be grouped into 2 (two) types, namely continuous images and discrete images (digital images). A continuous image is generated from an optical system that receives an analog signal. Example: human eye, analog camera. Meanwhile, digital images are generated through the process of digitizing continuous images. Example: digital camera.

Keywords: Image Processing, Sharpening, Quality, Derivative Filtering Algorithm.

1. PENDAHULUAN

Teknologi *remote sensing* atau penginderaan jauh menawarkan kemudahan dalam pemetaan suatu wilayah. Saat ini penginderaan jauh sangat sering digunakan karena citra dapat dibuat secara cepat meskipun berada pada daerah yang sulit ditempuh melalui daratan sehingga sangat di perlukan untuk pemetaan daerah bencana dan citra yang dihasilkan menggambarkan objek di permukaan bumi dengan wujud dan letak objek yang mirip dengan objek yang sebenarnya. Citra penginderaan jauh juga dapat menggambarkan objek yang berada di dalam tanah ataupun objek yang berada di dasar laut yang mencapai kedalaman tertentu sehingga dapat dilakukan pengenalan objeknya. Pada dasarnya pengubahan citra menggunakan algoritma *filtering derivatif* dilakukan dengan membuat gambar-gambar *frame* di antara gambar asal dan gambar tujuan dan menghasilkan *film* atau *image* yang lebih bagus. Pengolahan citra digital merupakan manipulasi dan interpretasi digital dari citra penginderaan jauh dengan bantuan komputer. Citra digital yang akan dibahas dalam paper ini adalah citra penginderaan jauh [1].

Pada proses ini terjadi konstruksi dari sejumlah objek yang menggambarkan transisi berangsur–angsur dari suatu objek ke objek yang lainnya. Proses perubahan bentuk ini banyak di gunakan pada aplikasi dibidang hiburan, animasi komputer, visualisasi ilmiah dan pendidikan. Perubahan tersebut harus terjadi secara beraturan dan konsisten untuk mencapai citra tujuan. Sistem penajaman ini merupakan salah satu sistem yang bertujuan untuk proses perubahan bentuk ini banyak di gunakan pada aplikasi dibidang hiburan, animasi komputer, visualisasi ilmiah dan pendidikan.

Image enhancement dapat dicirikan dalam dua hal yaitu operasi titik dan operasi lokal. Operasi titik mengubah nilai kecerahan setiap piksel di dalam suatu data citra secara terpisah, dan operasi lokal mengubah nilai tiap piksel dalam hubungannya dengan nilai kecerahan piksel di sekitarnya. Kualitas penajaman citra berhubungan dengan kejelasan ridge *structure* pada sisi citra. Sistem penajaman pada sisi citra bertujuan untuk mengidentifikasi pola gambar.

Penajaman kualitas citra diperlukan karena seringkali citra yang dijadikan objek mempunyai kualitas yang buruk, misalnya citra yang mengalami derau, kabur dan sebagainya. Untuk memperoleh kualitas keindahan gambar maka dilakukan peningkatan kualitas citra. Peningkatan kualitas citra merupakan proses awal dalam peningkatan mutu citra. Kualitas citra yang baik jika memiliki kontras yang baik dan dapat menggambarkan struktur *ridges* dan *valleys* yang jelas. Citra yang baik akan memiliki kontras yang baik dan akan dengan baik menggambarkan *ridges* dan *valleys*, jika citra sidik jari memiliki kualitas yang kurang baik maka akan memiliki kontras yang kurang sehingga akan kurang jelas menggambarkan batasan-batasan *ridges* (bukit). Perbaikan sisi citra ini akan sangat membantu untuk meningkatkan kualitas dari ekstraksi citra, dengan menentukan nilai konstanta untuk mendapatkan hasil yang terbaik [2].

Jurnal Sistem Komputer dan Informatika (JSON)
Volume 2, Nomor 1, September 2020
e-ISSN 2685-998X
DOI 10.30865/json.v2i1.2458

Pada penelitian sebelumnya oleh Ari Ismul Hadi, 2006, dengan judul penelitian "Analisis Citra Digital Dengan Menggunakan Teknik Penajaman Citra" menyatakan bahwa kecerahan rona objek pada citra bergantung atas besarnya tenaga pantulan yang datang dari objek. Berdasarkan warna yang digunakan, foto berwarna dapat dibedakan atas, foto berwarna semu (*false color*) atau foto infra merah berwarna dan foto warna asli (*true color*). Pada foto berwarna semu, warna objek tidak sama dengan warna foto. Objek seperti vegetasi yang berwarna hijau dan banyak memantulkan spektrum infra merah, akan tampak merah (*reddish*) pada foto, air akan tampak biru (*blueish*), dan tanah akan tampak hijau (*greenish*). Sedangkan untuk foto warna asli hasilnya akan tampak sesuai dengan aslinya.

Oleh karena itu, perbaikan kualitas sisi citra seharusnya menjadi prioritas utama sebelum mengidentifikasi parameter-parameter yang berupa ciri *(feature)* dari objek didalam citra, untuk selanjutnya parameter tersebut digunakan dalam menginterpretasi citra. Salah satu metode yang dapat melakukan proses peningkatan kualitas sisi citra 3 dimensi adalah algoritma *filtering derivatif.*

Algoritma *filtering derivatif* adalah perumusan konsep turunan matematik diawali dengan memahami konsep-konsep aplikasi kalkulus khususnya teori differensial (turunan) dalam rumusan digital. Metode yang digunakan dalam perumusan teori ini adalah mencari, menghimpun dan mempelajari beberapa teori yang terkait dengan *differensial* dan aplikasinya. Teknik derivasi dapat dijelaskan sebagai penghitungan laju nilai kecerahan pada citra digital diskrit, atau dikenal sebagai teknik derivasi citra. Penajaman akan diperoleh dengan menghitung beda nilai kecerahan antara nilai kecerahan yang bersebelahan dalam kolom dan baris yang sama. Berdasarkan ini dihasilkan filter derivasi turunan pertama (*filter derivasi gradien*) dan filter derivasi turunan kedua (*filter derivasi Laplacian*) [3].

2. METODE PENELITIAN

2.1 Citra

Citra merupakan hasil keseluruhan dari suatu sistem perekaman data. Secara teoritis citra dapat dikelompokkan menjadi 2 (dua) macam, yaitu citra kontinu dan citra diskrit (citra digital). Citra kontinu dihasilkan dari sistem optik yang menerima sinyal analog. Contoh: mata manusia, kamera analog. Sedangkan citra digital dihasilkan melalui proses digitalisasi terhadap citra kontinu. Contoh: kamera digital, scanner [4].

2.2 Kontras

Jika sebuah citra yang mempunyai nilai keabuan yang tidak terlalu berbeda untuk semua titik, maka citra tersebut akan kelihatan kurang kontras. Hal ini disebabkan citra tersebut memiliki kurva histogram yang sempit, dengan tepi kiri dan kanan yang berdekatan, sehingga titik tergelap dalam citra tersebut tidak mencapai hitam pekat dan titik paling terang dalam citra itu tidak berwarna putih cemerlang. Peningkatan kontras dapat dilakukan dengan menggunakan persamaan berikut:

$$Ko = G (Ki – P) + P \quad (1)$$

Dengan G adalah koefisien penguatan kontras, dan P adalah nilai skala keabuan yang dipakai sebagai pusat pengontrasan [7].

2.3 Algoritma Filtering Derivatif

Selain menggunakan sebuah matrix neighbor teknik spatial filtering menggunakan sebuah matriks lagi yang dinamakan mask. Matriks ini ukurannya harus sama besarnya dengan matrix neighbor yaitu N x N. Didalam mask inilah nantinya disimpan jenis operasi yang akan dilakukan pada matrix neighbor. Berikut ini dijelaskan proses yang akan kita lakukan terhadap matrix mask dan matrix neighbor.

1. Lakukan loop untuk seluruh titik pada gambar (gunakan dua for loop. Loop pertama untuk sumbu X dan loop kedua untuk sumbu Y).
   a. Rekam titik yang sedang diperiksa dan juga titik sekitarnya ke dalam matrix neighbor.
   b. Isi matrix mask dengan angka.
   c. Kalikan matrix neighbor dengan matrix mask secara scalar (Output[x,y] = Mask[x.y] * Neighbor[x,y]). Maksudnya adalah kalikan cell matrix mask [0,0] dengan matrix neighbor[0,0] dan simpan hasilnya dalam matriks lain misalnya output [0,0]. Kemudian kalikan cell matrix mask [0,1] dengan matrix neighbor [0,1] demikian seterusnya hingga setiap cell dari matriks telah dikalikan. Jumlahkan seluruh isi cell dari matrix Output. Hasil penjumlahan ini adalah titik baru yang akan kita letakkan pada layer output. Dan secara umum proses pelatihan untuk metode ini terbagi dalam 3 tahapan yaitu :
      1. Feedforward input training pattern
         Merupakan tahap untuk melakukan perhitungan dengan mencari nilai output pada setiap layer yang akan menjadi nilai input pada layer selanjutnya, dan nilai output pada layer terakhir merupakan hasil akhir (output) dari sistem tersebut
      2. Associated error
         Merupakan tahap untuk melakukan perhitungan error dan nilai koreksi pada layer terakhir berdasarkan

Jurnal Sistem Komputer dan Informatika (JSON)
Volume 2, Nomor 1, September 2020
e-ISSN 2685-998X
DOI 10.30865/json.v2i1.2458

target yang telah diberikan sebagai pelatihan dibandingkan. Error pada layer terkahir ini akan digunakan untuk menghitung error dan nilai koreksi pada layer sebelumnya, demikian juga untuk 2 layer sebelumnya akan menggunakan error pada layer di atasnya.

3. Adjustment weight

Merupakan tahap untuk mengubah bobot yang dimiliki oleh setiap node pada tiap layer mulai dari layer output dan turun sampai pada hidden layer yang pertama.

Ke tiga tahapan tersebut diulangi terus menerus sampai mendapatkan nilai error yang diinginkan. Setelah training selesai dilakukan, hanya tahap pertama yang diperlukan untuk memanfaatkan Metode tersebut. Saat umpan maju (feedforward), setiap unit input (X_i) akan menerima sinyal input dan akan menyebarkan sinyal tersebut pada tiap hidden unit (Z_j). Setiap hidden unit kemudian akan menghitung aktivasinya dan mengirim sinyal (Z_j) ke tiap unit output. Kemudian, setiap unit output (Y_k) juga akan menghitung aktivasinya (Y_k) untuk menghasilkan respon terhadap input yang diberikan jaringan.

Saat proses pelatihan (training) setiap unit output membandingkan aktivasinya (Y_k) dengan nilai target (desired output) untuk menentukan besarnya error. Berdasarkan error tersebut, dihitung faktor δ_k. Faktor δ_k digunakan untuk mendistribusikan error dari output kembali ke layer sebelumnya. Dengan cara yang sama, faktor δ_i juga dihitung pada hidden unit Z_j. Faktor δ_k digunakan untuk memperbaharui bobot antara hidden layer dan input layer. Setelah semua faktor δ ditentukan, bobot untuk semua layer di sesuaikan secara bersamaan. Pembaharuan bobot W_{jk} (dari hidden unit Z_j ke unit output Y_k) dilakukan berdasarkan faktor δ_k dan aktivasi Z_j dari hidden unit Z_j. Sedangkan, pembaharuan bobot V_{ij} (dari input unit X_i ke hidden unit Z_j) dilakukan berdasarkan faktor δ_j dan aktivasi X_i dari input.

Secara detail, langkah-langkah pelatihan sebagai berikut:

Langkah 1. : Inisialisasi bobot dan bias
Baik bobot maupun bias dapat diset dengan sembarang angka (Random) dan biasanya, angka di sekitar 0 dan 1 atau -1 (bias positif atau negatif).

Langkah 2. : Jika stop condition masih belum terpenuhi (false), jalankan langkah 3-10

Langkah 3. : Untuk setiap pasangan pelatihan (data training), lakukan langkah 4-9. Perambatan maju (feedforward)

Langkah 4. : Setiap unit input (X_i, i= 1,....,n) menerima sinyal input X_i dan menyebarkan sinyal tersebut pada seluruh unit pada hidden unit. Perlu diketahui bahwa input X_i yang dipakai di sini adalah input training data yang sudah diskalakan.

Langkah 5. : Setiap hidden unit (Z_j, j=1,.,p) akan menjumlahkan sinyal-sinyal input yang sudah berbobot, termasuk biasnya,

$$z\_in_j = v_{0j} + \sum_{i=1}^{n} x_i v_{ij} \quad \text{..... (2)}$$

dan memakai fungsi aktivasi yang telah ditentukan untuk menghitung sinyal output dari hidden unit yang bersangkutan.

$$z_j = f(z\_in_j) \quad \text{..... (3)}$$

lalu mengirim sinyal output tersebut ke seluruh unit pada unit output

Langkah 6. : Setiap unit output (Y_k, k=1,..,m) akan menjumlahkan sinyal- sinyal input yang sudah berbobot, termasuk biasnya,

$$y\_in_k = w_{0k} + \sum_{j=1}^{p} z_i w_{jk} \quad \text{..... (4)}$$

dan memakai fungsi aktivasi yang telah ditentukan untuk menghitung sinyal output dari unit output yang bersangkutan

$$y_j = f(y\_in_k) \quad \text{..... (5)}$$

lalu mengirim sinyal output ini ke seluruh unit pada unit output.

Langkah 7. : Setiap unit output (Y_k, k = 1,.,m) menerima suatu target pattern (desired output) yang sesuai dengan input training pattern untuk menghitung kesalahan (error) antara target dengan output yang dihasilkan jaringan.

$$\delta_k = t_k - y_k f'(y\_in_k) \quad \text{..... (6)}$$

Sebagaimana input training data, output training data t_k juga telah diskalakan menurut fungsi aktivasi yang dipakai. Faktor δ_k digunakan untuk menghitung koreksi error (ΔWjk) yang nantinya akan dipakai untuk memperbaharui Wjk, di mana

$$\Delta w_{jk} = \alpha \delta_k z_j \quad \text{..... (7)}$$

Jurnal Sistem Komputer dan Informatika (JSON)
Volume 2, Nomor 1, September 2020
e-ISSN 2685-998X
DOI 10.30865/json.v2i1.2458

selain itu, juga dihitung koreksi bias ΔW_{0k} yang nantinya akan dipakai untuk memperbaharui ΔW_{0k}, di mana:

$$\Delta w_{0k} = \alpha \delta_k \quad \text{..... (8)}$$

Faktor δk kemudian dikirimkan ke layer yang berada pada Langkah 8.

Langkah 8. : Setiap hidden unit (Zj, j = 1,.,p) menjumlahkan input delta (yang dikirim dari layer pada langkah 7) yang sudah berbobot.

$$\delta\_in_j = \sum_{k=1}^{m} \delta_k W_{jk} \quad \text{..... (9)}$$

Kemudian, hasilnya dikalikan dengan turunan dari fungsi Aktivasi yang digunakan jaringan untuk menghasilkan faktor koreksi error j, di mana

$$\delta_j = \delta\_in_j \, f'(z\_in_j) \quad \text{..... (10)}$$

Faktor δ_j digunakan untuk menghitung koreksi error (ΔV_{ij}) yang nantinya akan dipakai untuk memperbaharui V_{ij}, di mana:

$$\Delta v_{ij} = \alpha \delta_j x_i \quad \text{..... (11)}$$

Selain itu, juga dihitung koreksi bias ΔV_{0j} yang nantinya akan dipakai untuk memperbaharui V_{0j}, di mana:

$$\Delta v_{0j} = \alpha \delta_j \quad \text{..... (12)}$$

Pembaharuan bobot (adjustment) dan bias

Langkah 9. : Setiap unit output (Y_k, k = 1,.,m) akan memperbaharui bias dan bobotnya dari setiap hidden unit (j = 0,.,p),

$$w_{jk}(baru) = w_{jk}(lama) + \Delta w_{jk} \quad \text{..... (13)}$$

demikian pula, setiap hidden unit (Z_j, j= 1,., p) akan memperbaharui bias dan bobotnya dari setiap unit input (i =0,…,n)

$$w_{jk}(baru) = w_{jk}(lama) + \Delta v_{jk} \quad \text{..... (14)}$$

Langkah 10. : Memeriksa stop condition.
Alternatif kondisi berhenti selain menggunakan maksimum epoh juga dapat menggunakan target error Mean Square Error (MSE). Jika target MSE sudah mencapai target error, maka proses pelatihan akan berhenti.
MSE dinyatakan dengan

$$MSE = \frac{\sum_{i=1}^{n} (t - output)^2}{n} \quad \text{..... (15)}$$

Dimana :

n	:	adalah banyaknya pelatihan
X_1 .. X_n	:	Masukan
Y_1 .. Y_n	:	Keluaran
Z_1 .. Z_n	:	Nilai lapisan tersembunyi
V_{ij}	:	Bobot antara lapisan masukan dan lapisan tersembunyi
W_{jk}	:	Bobot antara lapisan tersembunyi dan lapisan keluaran
δ	:	Galat informasi
α	:	Konstanta berkelanjutan

Langkah-langkah mencari MSE yaitu :

1. Hitung selisih antara nilai target dengan nilai output dari pelatihan.
2. Kuadratkan setiap selisih tersebut.
3. Jumlahkan semua kuadrat selisih dari tiap-tiap data pelatihan dalam satu epoh.
4. Bagi hasil penjumlahan tersebut dengan banyaknya data pelatihan.

Setelah proses pelatihan, jaringan saraf backpropagation diaplikasikan dengan menggunakan fase feedforward dari algoritma pelatihan. Algoritma untuk pengujian adalah sebagai berikut :

1. Operasi pada hidden layer : menjumlahkan bobot akhir input beserta bobot akhir bias ke hidden.

$$z\_in_j = v_{0j} + \sum_{i=1}^{n} x_i v_{ij} \quad \text{..... (16)}$$

Fungsi aktifasi untuk menghitung sinyal outputnya :

Jurnal Sistem Komputer dan Informatika (JSON)
Volume 2, Nomor 1, September 2020
e-ISSN 2685-998X
DOI 10.30865/json.v2i1.2458

$$z_j = f(z\_in_j) \quad \text{..... (17)}$$

2. Operasi pada output layer : menjumlahkan bobot akhir hidden ke output beserta bobot akhir bias ke output.

$$y\_in_k = w_{0k} + \sum_{j=1}^{p} z_i w_{jk} \quad \text{..... (18)}$$

Gunakan fungsi aktifasi untuk menghitung sinyal outputnya :

$$y_j = f(y\_in_k) \quad \text{..... (19)}$$

3. HASIL DAN PEMBAHASAN

Peningkatan kualitas citra merupakan suatu tahapan operasi pengolahan citra. Operasi peningkatan kualitas citra digunakan untuk meningkatkan fitur tertentu pada citra sehingga tingkat keberhasilan dalam pengolahan citra menjadi tinggi. Dalam peningkatan kualitas citra dapat dilakukan dengan memperbaiki kontras, menambahkan warna, menghaluskan garis-garis yang bergerigi sehingga tampak lebih bersih, menyaring ketidak seragaman sinyal kiriman yang membawa gambar dan mempertajam sudut-sudut yang kabur serta mengkoreksi distorsi yang disebabkan alat optis atau tampilan. Citra direpresentasikan dalam bentuk matrik memiliki dimensi yang besar dengan baris dan kolom menunjukkan sebuah titik pada citra serta kesesuaian nilai elemen matrik mengidentifikasikan level pada titik tersebut. Citra keabuan didasari pada pengenalan warna dengan menggunakan salah satu operator klasik yaitu mask laplace. Data yang digunakan dalam penelitian ini adalah beberapa citra wajah untuk pengujian, dimana citra yang digunakan merupakan citra grayscale dengan image size 256 x 256 pixel, selanjutnya citra tersebut akan mengalami proses untuk mendapatkan kemiripan yang merupakan citra hasil (Citra Output).

Untuk melakukan penelitian ini, sumber data berupa citra RGB bertipe bmp. Citra ini akan di konversi dengan software yang dibangun dengan software Visual Basic. Hasil akhirnya adalah citra yang telah diubah kedalam format grayscale.

Proses pertama adalah mengambil nilai R, G dan B dari suatu citra bertipe RGB. Pada tipe .bmp citra direpresentasikan dalam 24 bit, sehingga diperlukan proses untuk mengambil masingmasing 3 kelompok 8 bit dari 24 bit tadi. Sebagai contoh suatu pixel memiliki nilai RGB 24 bit sebagai berikut :

111100001111000011111111, untuk mendapatkan masing-masing nilai R, G dan B dilakukan operasi-operasi sebagai berikut. Untuk mendapatkan nilai R dilakukan operasi modulo dengan bilangan 256 sebagai berikut :

Nilai R = 111100001111000011111111 mod 10000000
= 11111111

Sedangkan untuk nilai G, dapat dicari dengan cara sebagai berikut:

Nilai G = (111100001111000011111111 and 111111100000000)/100000000
= 11110000

Untuk Nilai B, dapat dicari dengan menggunakan rumus Nilai B= (111100001111000011111111 and 111111110000000000000000)/10000000000000000
= 11110000 sehingga dari nilai pixel 111100001111000011111112 atau 15790335 diperoleh nilai :

R =11111111 = 255
G = 11110000 = 240
B = 11110000= 240

Sehingga diperoleh triplet RGB= (255,240,240).

Setelah nilai triplet RGB kita peroleh, maka kita bisa mendapatkan nilai grayscale dari pixel tersebut. Ide dasarnya sebenarnya adalah membuat band tunggal dari 3 band RGB tadi dengan rumus tertentu. Pada penelitian ini digunakan rumus :

red = (red * 5) \ 10
green = (green * 8) \ 10
blue = (blue * 3) \ 10
gray = ((red + green + blue) * 10) \ 16

Dengan mengaplikasikan prosedur tadi pada semua pixel akan kita dapatkan citra dengan format grayscale. Berikut ini nilai RGB dari masing-masing pixel citra diatas :

Tabel 1. Nilai RGB

Pixel	R	G	B
0	222	220	204
1	220	220	212
2	237	207	192
3	220	220	212

Jurnal Sistem Komputer dan Informatika (JSON)
Volume 2, Nomor 1, September 2020
e-ISSN 2685-998X
DOI 10.30865/json.v2i1.2458

Pixel	R	G	B
4	220	228	220
5	220	220	212
6	220	228	220
7	228	228	218
8	220	228	220
9	228	230	228
10	212	220	212
11	220	220	212
12	220	220	212
13	212	212	196
14	220	220	212
15	204	204	188
16	152	157	152
17	190	187	190
18	179	181	168
19	187	202	197
20	187	202	197
21	204	204	188
22	212	220	212
23	212	212	204
24	221	212	212

Berdasarkan nilai RGB pada tabel diatas maka dihitung nilai warna greylevel-nya, maka didapatkan hasil sebagai berikut :

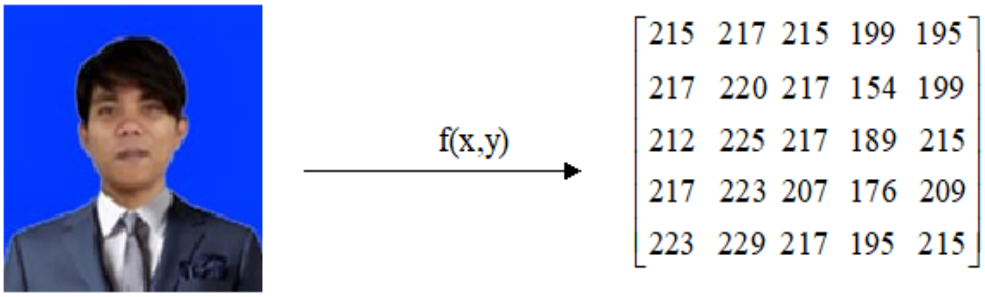
$$f(x,y)=\frac{R+G+B}{3}$$

Tabel 2. Nilai Greylevel

Pixel	R	G	B	Greylevel
0	222	220	204	215
1	220	220	212	217
2	237	207	192	212
3	220	220	212	217
4	220	228	220	223
5	220	220	212	217
6	220	228	220	220
7	228	228	218	225
8	220	228	220	223
9	228	230	228	229
10	212	220	212	215
11	220	220	212	217
12	220	220	212	217
13	212	212	196	207
14	220	220	212	217
15	204	204	188	199
16	152	157	152	154
17	190	187	190	189
18	179	181	168	176
19	187	202	197	195
20	187	202	197	195
21	204	204	188	199
22	212	220	212	215
23	212	212	204	209
24	221	212	212	215

Nilai greylevel pada Tabel 2 dikonversi ke dalam bentuk matrik sebagai berikut:

Jurnal Sistem Komputer dan Informatika (JSON)
Volume 2, Nomor 1, September 2020
e-ISSN 2685-998X
DOI 10.30865/json.v2i1.2458



Gambar 1. Matrik Citra Input

3.1 Penerapan Algoritma Filtering Derivatif

Proses penajaman citra keabuan merupakan suatu pemrosesan terhadap citra untuk menemukan batas tingkat grayscale dari pixel-pixel pada suatu citra. Ada banyak pendekatan yang dilakukan terhadap permasalahan. Pada penelitian ini, pendekatan dilakukan terhadap permasalahan deteksi dengan Grey Level Tranformation. Adapun urutan pendeteksian adalah sebagai berikut:

1. Lakukan konvolusi terhadap citra dengan menggunakan operator Laplace.

$$f(x,y)=\begin{bmatrix}215 & 217 & 215 & 199 & 195\\ 217 & 220 & 217 & 154 & 199\\ 212 & 225 & 217 & 189 & 215\\ 217 & 223 & 207 & 176 & 209\\ 223 & 229 & 217 & 195 & 215\end{bmatrix};\qquad g(x,y)=\begin{bmatrix}0 & -1 & 0\\ -1 & 4 & -1\\ 0 & -1 & 0\end{bmatrix}$$

Langkah-langkah konvolusi digambarkan sebagai berikut:

Langkah I : Tempatkan kernel pada sudut kiri atas, kemudian hitung nilai pixel pada posisi (0,0) dari kernel

$$\begin{bmatrix}215 & 217 & 215 & 199 & 195\\ 217 & 220 & 217 & 154 & 199\\ 212 & 225 & 217 & 189 & 215\\ 217 & 223 & 207 & 176 & 209\\ 223 & 229 & 217 & 195 & 215\end{bmatrix}\rightarrow\begin{bmatrix}* & * & * & * & *\\ * & 4 & * & * & *\\ * & * & * & * & *\\ * & * & * & * & *\\ * & * & * & * & *\end{bmatrix}$$

Proses perkalian konvolusi = 4; nilai ini dihitung dengan cara berikut :
(0x215) + (-1x217) + (0x215) + (-1x217) + (4x220) + (-1x217) + (0x212) + (-1x225) + (0x217) = 4

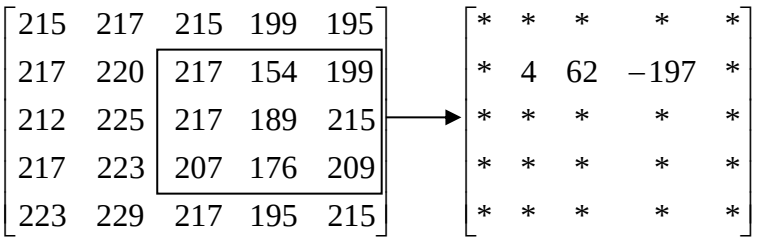
Langkah II : Geser kernel satu pixel ke kanan, kemudian hitung nilai pixel pada posisi (0,0) dari kernel

$$\begin{bmatrix}215 & 217 & 215 & 199 & 195\\ 217 & 220 & 217 & 154 & 199\\ 212 & 225 & 217 & 189 & 215\\ 217 & 223 & 207 & 176 & 209\\ 223 & 229 & 217 & 195 & 215\end{bmatrix}\rightarrow\begin{bmatrix}* & * & * & * & *\\ * & 4 & 62 & * & *\\ * & * & * & * & *\\ * & * & * & * & *\\ * & * & * & * & *\end{bmatrix}$$

Proses perkalian konvolusi = 62 ; nilai ini dihitung dengan cara berikut :
(0x217) + (-1x215) + (0x199) + (-1x220) + (4x217) + (-1x154) + (0x225) + (-1x217) + (0x189) = 62

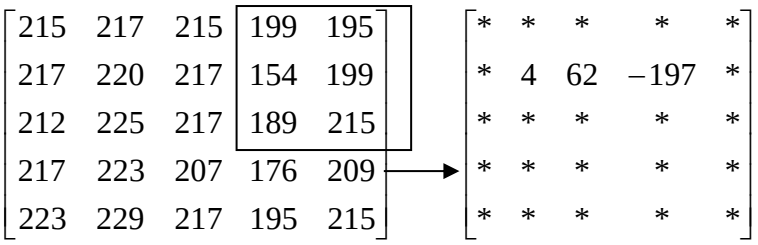
Langkah III : Geser kernel satu pixel ke kanan, kemudian hitung nilai pixel pada posisi (0,0) dari kernel

Jurnal Sistem Komputer dan Informatika (JSON)
Volume 2, Nomor 1, September 2020
e-ISSN 2685-998X
DOI 10.30865/json.v2i1.2458



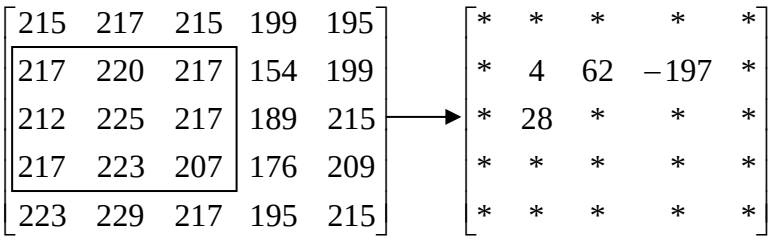
Proses perkalian konvolusi = -197 ; nilai ini dihitung dengan cara berikut :
(0x215) + (-1x199) + (0x195) + (-1x217) + (4x154) + (-1x199) + (0x217) + (-1x189) + (0x215) = -197

Langkah IV : Geser kernel satu pixel ke kanan, kemudian hitung nilai pixel pada posisi (0,0) dari kernel



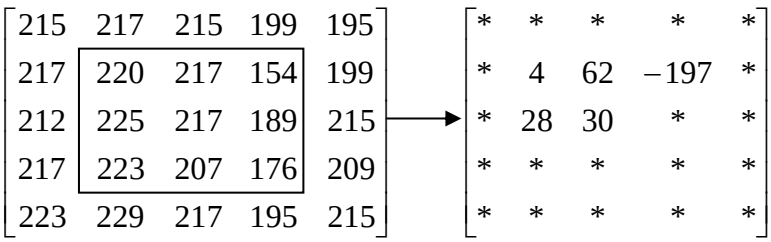
Konvolusi diabaikan karena pixel-pixel pada sudut kiri menggantung, maka pixel-pixel sudut kiri nilainya tetap sama seperti citra asal

Langkah V : Selanjutnya geser kernel satu pixel ke bawah, lalu mulai lagi melakukan konvolusi dari sisi kiri citra. Kemudian hitung nilai pixel pada posisi (0,0) dari kernel



Proses perkalian konvolusi = 28 ; nilai ini dihitung dengan cara berikut :
(0x217) + (-1x220) + (0x217) + (-1x212) + (4x225) + (-1x217) + (0x217) + (-1x223) + (0x207) = 28

Langkah VI : Geser kernel satu pixel ke kanan, Kemudian hitung nilai pixel pada posisi (0,0) dari kernel



Proses perkalian konvolusi = 30 ; nilai ini dihitung dengan cara berikut :
(0x220) + (-1x217) + (0x154) + (-1x225) + (4x217) + (-1x189) + (0x223) + (-1x207) + (0x176) = 30

Langkah VII : Geser kernel satu pixel ke kanan, Kemudian hitung nilai pixel pada posisi (0,0) dari kernel

Jurnal Sistem Komputer dan Informatika (JSON)
Volume 2, Nomor 1, September 2020
e-ISSN 2685-998X
DOI 10.30865/json.v2i1.2458

$$\begin{bmatrix} 215 & 217 & 215 & 199 & 195 \\ 217 & 220 & 217 & 154 & 199 \\ 212 & 225 & 217 & 189 & 215 \\ 217 & 223 & 207 & 176 & 209 \\ 223 & 229 & 217 & 195 & 215 \end{bmatrix} \rightarrow \begin{bmatrix} * & * & * & * & * \\ * & 4 & 62 & -197 & * \\ * & 28 & 30 & -6 & * \\ * & * & * & * & * \\ * & * & * & * & * \end{bmatrix}$$

Proses perkalian konvolusi = -6 ; nilai ini dihitung dengan cara berikut :
(0x217) + (-1x154) + (0x199) + (-1x217) + (4x189) + (-1x215) + (0x207) + (-1x176) + (0x209) = -6

Langkah VIII: Geser kernel satu pixel ke kanan, Kemudian hitung nilai pixel pada posisi (0,0) dari kernel

$$\begin{bmatrix} 215 & 217 & 215 & 199 & 195 \\ 217 & 220 & 217 & 154 & 199 \\ 212 & 225 & 217 & 189 & 215 \\ 217 & 223 & 207 & 176 & 209 \\ 223 & 229 & 217 & 195 & 215 \end{bmatrix} \rightarrow \begin{bmatrix} * & * & * & * & * \\ * & 4 & 62 & -197 & * \\ * & 28 & 30 & -6 & * \\ * & * & * & * & * \\ * & * & * & * & * \end{bmatrix}$$

Konvolusi diabaikan karena pixel-pixel pada sudut kiri menggantung, maka pixel-pixel pinggir kiri nilainya tetap sama seperti citra asal

Langkah IX : Selanjutnya geser kernel satu pixel ke bawah, lalu mulai lagi melakukan konvolusi dari sisi kiri citra. Kemudian hitung nilai pixel pada posisi (0,0) dari kernel

$$\begin{bmatrix} 215 & 217 & 215 & 199 & 195 \\ 217 & 220 & 217 & 154 & 199 \\ 212 & 225 & 217 & 189 & 215 \\ 217 & 223 & 207 & 176 & 209 \\ 223 & 229 & 217 & 195 & 215 \end{bmatrix} \rightarrow \begin{bmatrix} * & * & * & * & * \\ * & 4 & 62 & -197 & * \\ * & 28 & 30 & -6 & * \\ * & 14 & * & * & * \\ * & * & * & * & * \end{bmatrix}$$

Proses perkalian konvolusi = 14 ; nilai ini dihitung dengan cara berikut :
(0x212) + (-1x225) + (0x217) + (-1x217) + (4x223) + (-1x207) + (0x223) + (-1x229) + (0x217) = 14

Langkah X : Geser kernel satu pixel ke kanan, Kemudian hitung nilai pixel pada posisi (0,0) dari kernel

$$\begin{bmatrix} 215 & 217 & 215 & 199 & 195 \\ 217 & 220 & 217 & 154 & 199 \\ 212 & 225 & 217 & 189 & 215 \\ 217 & 223 & 207 & 176 & 209 \\ 223 & 229 & 217 & 195 & 215 \end{bmatrix} \rightarrow \begin{bmatrix} * & * & * & * & * \\ * & 4 & 62 & -197 & * \\ * & 28 & 30 & -6 & * \\ * & 14 & -5 & * & * \\ * & * & * & * & * \end{bmatrix}$$

Proses perkalian konvolusi = -5 ; nilai ini dihitung dengan cara berikut :
(0x225) + (-1x217) + (0x189) + (-1x223) + (4x207) + (-1x176) + (0x229) + (-1x217) + (0x195) = -5

Langkah XI : Geser kernel satu pixel ke kanan, Kemudian hitung nilai pixel pada posisi (0,0) dari kernel

Jurnal Sistem Komputer dan Informatika (JSON)
Volume 2, Nomor 1, September 2020
e-ISSN 2685-998X
DOI 10.30865/json.v2i1.2458

$$\begin{bmatrix} 215 & 217 & 215 & 199 & 195 \\ 217 & 220 & 217 & 154 & 199 \\ 212 & 225 & 217 & 189 & 215 \\ 217 & 223 & 207 & 176 & 209 \\ 223 & 229 & 217 & 195 & 215 \end{bmatrix} \rightarrow \begin{bmatrix} * & * & * & * & * \\ * & 4 & 62 & -197 & * \\ * & 28 & 30 & -6 & * \\ * & 14 & -5 & -96 & * \\ * & * & * & * & * \end{bmatrix}$$

Proses perkalian konvolusi = -96 ; nilai ini dihitung dengan cara berikut :
(0x217) + (-1x189) + (0x215) + (-1x207) + (4x176) + (-1x209) + (0x217) + (-1x195) + (0x215) = -96

Langkah XII : Geser kernel satu pixel ke kanan, Kemudian hitung nilai pixel pada posisi (0,0) dari kernel

$$\begin{bmatrix} 215 & 217 & 215 & 199 & 195 \\ 217 & 220 & 217 & 154 & 199 \\ 212 & 225 & 217 & 189 & 215 \\ 217 & 223 & 207 & 176 & 209 \\ 223 & 229 & 217 & 195 & 215 \end{bmatrix} \rightarrow \begin{bmatrix} * & * & * & * & * \\ * & 4 & 62 & -197 & * \\ * & 28 & 30 & -6 & * \\ * & 14 & -5 & -96 & * \\ * & * & * & * & * \end{bmatrix}$$

Konvolusi diabaikan karena pixel-pixel pada sudut kiri menggantung, maka pixel-pixel sudut kiri nilainya tetap sama seperti citra asal

Langkah XIII : Selanjutnya geser kernel satu pixel ke bawah, lalu mulai lagi melakukan konvolusi dari sisi kiri citra. Kemudian hitung nilai pixel pada posisi (0,0) dari kernel

$$\begin{bmatrix} 215 & 217 & 215 & 199 & 195 \\ 217 & 220 & 217 & 154 & 199 \\ 212 & 225 & 217 & 189 & 215 \\ 217 & 223 & 207 & 176 & 209 \\ 223 & 229 & 217 & 195 & 215 \end{bmatrix} \rightarrow \begin{bmatrix} * & * & * & * & * \\ * & 4 & 62 & -197 & * \\ * & 28 & 30 & -6 & * \\ * & 14 & -5 & -96 & * \\ * & * & * & * & * \end{bmatrix}$$

Konvolusi diabaikan karena pixel-pixel pada sudut bawah menggantung, maka pixel-pixel sudut bawah, nilainya tetap sama seperti citra asal

Langkah XIV sampai dengan Langkah XVI sama dengan Langkah XIII, dimana konvolusi diabaikan sehingga pixel-pixel sudut bawah, nilainya tetap sama seperti citra asal.

4. KESIMPULAN

Dari hasil penelitian yang penulis lakukan, dapat ditarik beberapa kesimpulan yang terkait dengan proses penelitian maupun dengan isi dari penelitian itu sendiri.

1. Sumber data berupa citra RGB bertipe bmp. Citra ini akan di konversi dengan software yang dibangun dengan software Visual Basic.
2. Citra direpresentasikan dalam bentuk matrik memiliki dimensi yang besar dengan baris dan kolom menunjukkan sebuah titik pada citra serta kesesuaian nilai elemen matrik mengidentifikasikan level pada titik tertentu.
3. Analisa sistem dilakukan sesuai dengan tingkat kebutuhan user dan sistem baik dari segi masukan yang dibutuhkan dan keluaran sistem yang diinginkan.
4. Algoritma filtering derivatif merupakan metode penajaman citra yang berupa citra masukan dan citra hasil kualitas, tujuannya adalah untuk kualitas citra keabuan objek citra didalam basis data menggunakan warna RGB dari objek wajah tertentu.

REFERENCES

[1] Awaluddin. M. 2010, “Penajaman dan Segmentasi Citra Pada Pengolahan Citra Digital” Jurnal Teknik, Vol. 31, No.1, ISSN : 0852-1697

[2] Fauzi, Yuliani, 2005, “Implementasi Filtering Derivatif Dalam Mengolah Citra Satelit Pada Software Envi”, Jurnal Gradien, Vol. 1, No.2,

[3] Putra, Darma, "Pengolahan Citra Digital", 2010, Penerbit Andi, Yogyakarta.
[4] S, Riyanto, dkk, 2005,"Step by Step Pengolahan Citra Digital" hal ; 23, Penerbit Andi, Yogyakarta.
[5] K, Abdul, 2013, "Teori dan Aplikasi Pengolahan Citra, Penerbit Andi, Yogyakarta.
[6] Sitorus, S, dkk, "Pengolahan Citra Digital, 2006, Penerbit Andi, Yogyakarta.
[7] Munir, Rinaldi, 2004, "Pengolahan Citra Dengan Pendekatan Algoritmik", Penerbit Andi Offset, Yogyakarta.
[8] Ahmad, Usman, 2005, "Pengolahan Citra Digital", Penerbit Alex Media, Jakarta.
[9] Nugroho, Adi, 2009, "Rekayasa Perangkat Lunak Menggunakan UML dan Java" Penerbit Andi, Yogyakarta.
[10] Aditya, Arif Primananda, 2013, "Dasar-Dasar Pemrograman Database Dekstop Dengan Visual Basic.Net 2008", PT. Elex Media Komputindo, Jakarta.