Perancangan Perangkat Lunak Red Eye Reduction dengan Teknik Intensity Color Checking

Hartono[1]
STMIK IBBI
Jl. Sei Deli No. 18 Medan, Telp. 061-4567111 Fax. 061-4527548
E-mail : hartonoibbi@gmail.com

Abstrak

Hasil foto baik dengan kamera biasa ataupun kamera digital bila diambil dari dengan pencahayaan yang tinggi terhadap orang, sering kali menghasilkan bintik merah pada bagian pupil mata yang istilah dengan red eye. Hal ini tentunya menyebabkan hasil photo menjadi tidak bagus. Dengan suatu perangkat lunak tertentu yang berfungsi sebagai pengolah gambar dapat dengan mudah menghilangkan red eye tersebut hanya dengan menggunakan sebuah tool yang disebut red eye reduction. Jadi dengan adanya program tersebut hasil photo dengan kamera digital dapat diedit untuk dihilangkan efek red eye sebelum dicetak. Red eye reduction menggunakan algoritma intensity color checking dalam proses penggantian pixel-pixel citra yang berwarna merah kemudian menggantinya dengan warna hitam keabu-abuan sesuai dengan proses intensitas yang dihasilkan. Program yang dapat mereduksi red eye dengan algoritma intensity color checking dengan cara memproses region tertentu yang dipilih. Hasil citra tersebut kemudian dapat dicetak ataupun disimpan kembali dalam format bitmap.

Kata Kunci : *Citra, Intensity color checking, pixel, Red eye.*

Abstract

The images either with a regular camera or digital camera when taken with a high luminance of the people, often times produce red spots on the term pupils with red eye. This is certainly cause the photo to be no good outcome. With one particular software that serves as an image processing can easily remove red eye is by using a tool called red eye reduction. So with the program is the result of photos with a digital camera can be edited to remove red eye effect before printing. Red eye reduction algorithm checking color intensity in the process of replacing the image pixels in red and then replace it with a greyish black color according to the intensity of the resulting process. Programs that can reduce red eye with color intensity algorithm processing by checking certain selected region. The results of the image can then be printed or stored back in bitmap format.

Keywords: *Image, checking color intensity, pixel, Red eye.*

1. Pendahuluan

Hasil foto baik dengan kamera biasa ataupun kamera digital bila diambil dari dengan pencahayaan yang tinggi terhadap orang, sering kali menghasilkan bintik merah pada bagian pupil mata, istilah dengan *red eye*. Hal ini tentunya menyebabkan hasil *photo* menjadi tidak bagus.

Red eye adalah masalah bagi fotografer yang menggunakan *flash*. Untungnya, *red eye* disebabkan oleh keadaan dapat diprediksi dan dapat mudah dihindari dengan sedikit persiapan hati-hati. Pada dasarnya *red eye* disebabkan oleh pantulan cahaya bagian dalam pupil orang tersebut dan kembali ke kamera. Dalam rangka untuk lebih memahami konsep ini, pertama perlu diketahui sedikit tentang cara kerja mata manusia. Di mata manusia, pupil adalah bagian yang mengembang dan kontrak untuk memungkinkan berbagai tingkat cahaya masuk ke mata mereka. Pupil yang memungkinkan seorang manusia melihat dengan jelas apakah seseorang berjalan di luar pada hari yang cerah atau berjalan oleh cahaya bulan. Jadi, ketika gambar diambil dengan *flash* di cahaya rendah, pupil tidak dapat menutup cukup cepat untuk memblokir cahaya, sehingga lampu kilat mencerminkan semua jalan di belakang mata dan tunas kembali ke kamera.

Cara termudah untuk menghindari *red eye* tempat pertama adalah menghindari situasi di mana harus menggunakan *flash* dalam cahaya rendah. Namun jika tidak dapat menghindarinya, dapat menggunakan

perangkat lunak pengedit foto untuk menghapus mata merah itu hanya membutuhkan waktu untuk menghapusnya terutama jika memiliki banyak gambar yang harus dilakukan.

Dengan suatu perangkat lunak tertentu yang berfungsi sebagai pengolah gambar dapat dengan mudah menghilangkan *red eye* tersebut hanya dengan menggunakan sebuah *tool* yang disebut *red eye reduction*. Jadi dengan adanya program tersebut hasil *photo* dengan kamera digital dapat diedit untuk dihilangkan efek *red eye* sebelum dicetak. Karena setiap orang pasti menginginkan kualitas *photo* yang bagus tanpa ada cacat. Masalah yang timbul adalah foto yang telah dibuat tidak mungkin difoto ulang, sehingga diperlukan suatu perangkat lunak yang dapat menghilangkan *red eye*.

Citra adalah kumpulan piksel-piksel yang disusun dalam larik dua dimnesi. Indeks baris dan kolom (*x*, *y*) dari sebuah piksel dinyatakan dalam bilangn bulat. Piksel (0,0) terletak pada sudut kiri atas pada citra, indeks x bergerek ke kanan dan indeks y bergerak ke bawah. Konvensi ini dipakai merujuk pda cara penulisan larik yang digunakan dalam pemrograman komputer. **(Ahmad, 2005 : 14)**

Citra yang baik adalah citra yang dapat menampilkan suatu objek yang dimaksud secara seutuhnya. Objek yang dimaksud adalah objek yang mengandung keindahan (nilai artistik) dan juga kejelasan untuk penganalisaan dan maksud-maksud lainnya. Dengan kata lain, citra yang baik adalah citra yang dapat menampilkan nilai artistik dan intrinsik objek tersebut dengan baik.

1.1 Representasi Citra

Sebuah citra digital a(x,y) yang diuraikan dalam sebuah ruang diskrit 2 dimensi diperoleh dari sebuah citra analog f(x,y) dalam sebuah ruang kontinu 2 dimensi melalui proses sampling yang sering dikenal sebagai digitisasi. Sebuah citra digital dapat dianggap sebagai suatu matriks (*array*) dimana baris dan kolomnya menunjukkan sebuah titik pada citra dan nilai elemen matriks menunjukkan warna (*gray level*) pada titik tersebut. Elemen dari *array* digital tersebut disebut piksel atau *picture elements (pixels).* Pada umumnya, citra digital yang direpresentasikan dengan a(x,y) terdiri dari beberapa properti yaitu lebar (*width*), tinggi (*height*), kedalaman warna (*depth*), dan juga warna (*color*) / intensitas *gray-level* untuk masing-masing piksel dalam citra tersebut.

Citra *grayscale* hanya terdiri dari warna keabu-abuan. Warna dalam citra *grayscale* adalah komposisi RGB (*Red*, *Green*, *Blue*), dimana ketiga-tiganya mempunyai nilai intensitas *graylevel* yang sama, jadi hanya perlu menspesifikasi nilai intensitas *graylevel* tunggal saja untuk setiap piksel dan hal inilah yang merupakan perbedaan antara citra *grayscale* dan citra berwarna. Intensitas *gray level* untuk citra *grayscale* disimpan dalam integer 8 bit (256 warna) dari putih ke hitam.

Citra warna juga dikenal dengan *integer image* tetapi berbeda dengan citra *grayscale*, yakni suatu citra warna mengandung satu set nilai tersusun (*a set of ordered values*). Setiap set nilai tersusun mewakili satu '*shade*' warna atau '*hue*'.

Format penyimpanan citra dapat berupa BMP (*Bitmap*), JPEG (*Joint Photographic Experts Group*), GIF (*Graphics Interchange Format*), TIFF (*Tagged Image File Format*), WMF (*Windows Metafile*), PNG (*Portable Network Graphics*), PIXAR (*Pixar Image Computers*), PCD (*Photo CD*), ICO (*Icon*), dan lain-lain. Dimana format BMP merupakan format standard untuk suatu citra digital.

1.2. Proses Citra

Semua citra dalam sistem komputer perlu dikodekan menggunakan simbol diskrit. Sebuah citra digital a(x,y) yang diuraikan dalam sebuah ruang diskrit dua dimensi diperoleh dari sebuah citra analog dalam ruang kontinue melalui proses sampling atau digitasi. Sebuah citra digital dianggap suatu matriks dimana baris dan kolomnya menunjukkan sebuah titik pada citra dan nilai elemen menunjukkan warna pada titik tersebut. Elemen dari *array*

Banyaknya piksel tiap satuan luas tergantung pada resolusi yang digunakan. Keanekaragaman warna piksel tergantung pada *bit depth* yang dipakai. Semakin banyak jumlah piksel tiap satuan luas, semakin baik kualitas gambar yang dihasilkan dan tentu akan semakin besar ukuran *file*-nya. (Gonzalez, 1992 : 8)

Citra analog dibagi dalam N baris dan M kolom sehingga diperoleh citra digital a(*x*, *y*) dengan memberikan nilai diskrit bagi setiap titik. Pada umunya, citra digital yang direpresentasikan dengan a(*x*, *y*) merupakan sebuah fungsi dari banyak variabel yang mencakup kedalaman / *depth* (*z*), warna / *color* (λ), dan waktu / *time* (*t*) atau kata lain, representasi citra digital yang sebenarnya dilambangkan dengan a(*x*, *y*, *z*, λ, *t*

1.3 Bit Depth (Kedalaman Warna)

Bit depth merupakan jumlah *bit* yang digunakan untuk merepresentasikan tiap *pixel*. *Bit depth* adalah jumlah *bit* untuk tiap *pixel*. Semakin banyak jumlah *bit* yang digunakan untuk menrepresentasikan sebuah *pixel*,

yang berarti semakin tinggi kedalaman *pixel*-nya, maka semakin tinggi pula kualitasnya, dengan resiko jumlah *bit* yang diperlukan menjadi lebih tinggi. (Gonzalez, 1992: 10)

Dengan 1 *byte* (8 *bit*) untuk tiap *pixel*, diperoleh 2^8 atau 256 *level* intensitas. Dengan *level* intensitas sebanyak itu, umumnya mata manusia sudah dapat dipuaskan. Kedalaman *pixel* paling rendah terdapat pada *binary-value image* yang hanya menggunakan 1 bit untuk tiap *pixel*, sehingga hanya ada dua kemungkinan bagi tiap *pixel*, yaitu 0 (hitam) atau 1 (putih).

1.4 Format Citra

Format *file* citra adalah bentuk standar dari proses organisasi dan penyimpanan gambar. *File* citra terkomposisi oleh data *pixel* atau *vector* (*geometric*) yang di-*rasterized* menjadi *pixel* saat ditampilkan dalam sebuah *vector graphic display*. *Pixel* yang mengkomposisi citra disusun sebagai sebuah *grid* (kolom dan baris); setiap *pixel* terdiri dari representasi magnitudes dari kecerahan dan warna.

Ada banyak *format file* grafik, jika termasuk *format* tipe *proprietary*. *Format* PNG, JPEG, dan GIF adalah format yang umum digunakan untuk menampilkan citra di *Internet*. *Format* grafik ini dijelaskan secara singkat di bawah, dibagi menjadi 2 keluarga grafik utama: *raster* dan *vector*. (Gonzalez, 1992 : 11)

1.5 Pengolahan Citra

Pengolahan citra merupakan suatu proses terhadap citra masukan untuk menghasilkan citra keluaran sesuai dengan yang diinginkan, dimana dalam prosesnya, pengolahan citra melibatkan manipulasi piksel-piksel dari suatu citra untuk membentuk citra yang lain. Berbagai jenis operasi mungkin diterapkan pada piksel-piksel dari citra asli untuk menghasilkan suatu citra yang baru.

Kompresi citra (*Image Compression*) merupakan suatu operasi untuk mengkompresi atau memadatkan ukuran file dari suatu citra, dimana tujuan dari kompresi data citra adalah untuk mengurangi jumlah bit yang digunakan untuk menyimpan citra, dengan usaha semaksimum mungkin tidak terjadi penurunan mutu citra dibandingkan dengan citra aslinya. (Murni, 1992: 5)

1.6 Efek-Efek Citra

Pengolahan citra juga mencakup pemberian suatu efek tertentu pada citra. Tujuannya adalah memberi kesan artistik pada citra itu sendiri sehingga citra bertambah indah. Program pengolah citra seperti *Adobe PhotoShop* ataupun *Corel PhotoPaint* mempunyai fungsi *built-in* untuk menambah efek tertentu. (Murni, 1992: 10)

1.7 Red Eye

Warna merah berasal dari cahaya yang membias pada retina pada mata manusia. Banyak hewan, termasuk anjing, kucing, dan rusa, retinanya mempunyai suatu lapisan pembias khusus yang disebut dengan *tapetum lucidum* yang berfungsi seperti sebuah cermin pada bagian belakang mata mereka. Jika suatu cahaya lampu seperti senter atau lampu sorot menyinari mata pada waktu malam hari, mata mereka akan tampak bersinar seperti memantulkan suatu cahaya putih.

Banyak kamera saat ini telah mempunyai fitur "*red eye reduction*". Pada kamera ini, *flashing* akan dilakukan sebanyak dua kali, pertama sebelum gambar diambil, kemudian sekali lagi ketika gambar sebenarnya diambil. *Flash* yang pertama menyebabkan pupil pada mata manusia berkontraksi, sehingga mengurangi "*red eye*" secara signifikan. Trik yang lain adalah mematikan semua cahaya dalam ruangan, juga dapat mengakibatkan pupil berkontraksi.

Cara lain untuk mengurangi atau memperkecil "*red eye*" adalah memindahkan *flash* dari lensa. Pada kebanyakan kamera kecil. *flash* hanya satu atau dua inci jauhnya dari lensa, jadi refleksi yang berasal dari lensa akan dihasilkan pada film. Jika *flash* pada kamera dapat dibongkar pasang dan memegangnya beberapa *feet* jauhnya dari lensa maka hal ini akan cukup menolong. Selain itu disarankan jika akan mengambil *photo* usahakan bagian belakang dari objek yang akan difoto tidak mempunyai sumber pencahayaan yang terang langsung menuju ke kamera.

1.8 Intensity Color Checking

Intensity color checking merupakan suatu algoritma yang dipergunakan untuk mengganti nilai komponen RGB yang sudah citra dengan nilai RGB yang lain tetapi dengan memperhitungkan faktor intesitasnya. Jadi pada algoritma ini proses penggantian tidak hanya langsung mengganti warna *pixel* dengan warna lain tetapi dengan warna yang mempunyai intensitas yang sama dengan *pixel* asal.

Bentuk dari algoritma *intensitas color checking* adalah kalkulasi untuk mencari nilai intesitas rata-rata dari komponen warna RGB dari suatu *pixel*. Secara umum bentuk aplikasi dari algoritma intesitas *color*

checking ini dapat dilihat pada contoh berikut ini: (http://www.catenary.com)

```
For i = 0 to 2r – 1
        For j = 0 to 2r – 1
            pixelcolor = GetPixel(Picture, l, j)
        r = pixelcolor Mod 256
        g = (pixelcolor \ 256) Mod 256
        b = pixelcolor \ 256 \ 256
                I = (R + G + B) \ 3
                Gray = (RedConstant * R + GreenConstant * G +
                    BlueConstant * B) \ 1000
                        R = Abs(Gray - I)
                        G = Abs(Gray - I)
                        B = Abs(Gray - I)
                SetPixel Picture, RGB(R, G, B)
        Next j
Next i
```

1.9 Euclidean Distance

Euclidean Distance merupakan suatu cara yang dipergunakan untuk membandingkan kemiripan antara suatu *pixel* dengan *pixel* hasil pada citra yang telah diproses dengan membandingkan jarak antara suatu *pixel* dengan *pixel* tetangganya. *Euclidean Distance* disebut *Euclidean Norm*.

Rumusan *Euclidean Distance* adalah sebagai berikut:

$$L_2(A,B) = \sum_{i=1}^{N} (A_i - B_i)^2$$

Dimana A dan B adalah vektor yang akan dicari jaraknya dan N adalah banyaknya elemen vektor yang akan dibandingkan. Nilai L merupakan suatu nilai hasil perbandingan. Fungsi perpangkatan di atas digunakan agar tidak dihasilkan nilai negatif.

2. Metode Penelitian

Langkah-langkah yang ditempuh untuk menyelesaikan Permasalahan di atas adalah sebagai berikut:

- Mendapatkan teori-teori atau penjelasan-penjelasan tentang efek red eye dan segala sesuatu yang berhubungan dengan pemrosesan citra.
- Mengadakan pembahasan dan menyusun tahapan perancangan perangkat lunak.
- Membuat algoritma-algoritma yang digunakan dalam membangun perangkat lunak.
- Membangun perangkat lunak berdasarkan algoritma-algoritma yang dibuat
- Menganalisa dan mengimplementasikan perangkat lunak yang telah dibuat.

2.1 Tahapan Perancangan

Aplikasi ini dirancang dengan menggunakan metodologi *waterfall* dimana penjelasan dari langkah-langkah ini adalah sebagai berikut:

1. Pengumpulan Data
   Dilakukan pengumpulan data terutama teori-teori dan bahan yang berhubungan dengan topik yang dibahas.
2. Analisis
   Dilakukan analisis terutama mengenai cara kerja *red eye reduction* yang dinyatakan dalam bentuk contoh perhitungan dan memahami cara kerja dari algoritma.
3. Perancangan *Interface*
   Dilakukan untuk merancang *user interface* untuk fungsi-fungsi seperti yang disebutkan di atas. Rancangan *interface* akan dilakukan secara langsung pada IDE (*Integrated Development Environment*) dari *Visual Basic 6.0*.
4. *Testing*
   Digunakan untuk melakukan pengujian atas program yang dirancang dan bila terdapat kesalahan akan dilakukan koreksi. Adapun cara pengujian program dilakukan dengan menguji sejumlah *input file*. Apabila

ditemukan kesalahan maka program akan diperiksa ulang untuk menentukan kesalahan dalam hal implementasi.

3. Hasil dan Pembahasan

Proses pada algoritma *red eye reduction* adalah sebagai berikut:

a. *Input image matrix* RGB 24-bit
b. Define matriks *region* ditentukan oleh *user* for certain *area red eye*. Proses untuk *matrix region* ini terdiri atas:
   1. *Move* intensitas *pixel* dari matrik *input*
   2. *Do* intensitas *color checking*.
c. *Update* matrik *input with current result*

Proses pada algoritma *blur* adalah sebagai berikut:

a. *Input image matrix* RGB 24-bit
b. Do iteration until *image width* x *image height*.
c. *Input* and get *pixel* on x, y.
d. For each *Red*, *Green*, *Blue* component by shift value with *constatant*.
e. Set *pixel result to image matrix*.

3.1 Menghitung Penggantian Pixel dengan Intensity Color Checking

Proses penggantian *pixel* dengan melakukan perhitungan *intensity color checking* diatur dalam persamaan utama sebagai berikut:

```
I = (R + G + B) \ 3
Gray = (222 * R + 707 * G + 71 * B) \ 1000
R = |Gray – I|
G = |Gray – I|
B = |Gray – I|
```

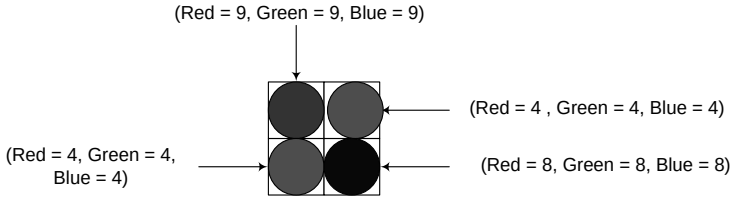
Dimana R (*Red*), G (*Green*), B (*Blue*) merupakan merupakan komponen warna *pixel* yaitu untuk warna merah, hijau dan biru yang dihasilkan dengan melalukan pembagian bulat dan *module* dengan nilai 256. I merupakan variabel untuk menghitung rata-rata intesitas *pixel* dengan pembagian bulat (menggunakan tanda *back slash*) sedangkan *gray* merupakan nilai tingkat keabuan *pixel* sebagai nilai sementara yang digunakan untuk dikurangkan dengan nilai intensitas I. Konstanta 222, 707, dan 71 merupakan konstanta ketentuan.

Agar algoritma di atas lebih mudah dipahami maka berikut ini peneliti menyertakan contoh perhitungan dengan menggunakan algoritma di atas dengan hanya menyertakan suatu contoh citra *input* yang terdiri atas empat *pixel* seperti terlihat pada Gambar 1.

(a) Posisi Masing-Masing *Pixel* (b) Komponen Warna *Pixel*

Gambar 1 Contoh *Pixel Input*

Untuk perhitungan ketiga *pixel* lainnya akan sama seperti perhitungan untuk *pixel* pada posisi (0, 0) sehingga hasil akhirnya diperlihatkan pada Gambar 2:



Gambar 2 Contoh *Pixel* Hasil Setelah Dilakukan *Red Eye Reduction*

Untuk mendapatkan perbandingan nilai intensitas setelah dan sebelum direduksi maka akan dianalisis

dengan *Euclidean Distance*. Perbandingan akan dilakukan antara *pixel* 1 dengan *pixel* 2, *pixel* 3, dan *pixel* 4 dari contoh di atas.

Untuk analisis ini hanya menggunakan *pixel* dengan komponen warna merah saja karena pada citra yang mengandung *red eye* biasanya *pixel* inilah yang dominan. Langkah awal adalah menghitung nilai L dengan rumusan *Euclidean Distance* untuk citra yang belum direduksi.

Untuk *pixel* (0, 0) dengan *pixel* (0, 1)
Nilai *red* pada *pixel* (0, 0) → R = 255
Nilai *red* pada pixel (0, 1) → R = 204
$A_1 = 255$; $B_1 = 204$
Untuk *pixel* (0, 0) dengan *pixel* (1, 0)
Nilai *red* pada *pixel* (0, 0) → R = 255
Nilai *red* pada *pixel* (1, 0) → R = 255
$A_2 = 255$; $B_2 = 255$
Untuk *pixel* (0, 0) dengan *pixel* (1, 1)
Nilai *red* pada *pixel* (0, 0) → R = 255
Nilai *red* pada *pixel* (1, 0) → R = 255
$A_3 = 255$; $B_3 = 255$

Perhitungan akan menggunakan rumusan *Euclidean Distance* sebagai berikut:

$$L_1 = L(A_i, B_i) = \sum_{i=1}^{3} (A_i - B_i)^2$$

$$L_1 = (255 - 204)^2 + (255 - 255)^2 + (255 - 255)^2$$

$$L_1 = 2601 + 0 + 0$$

$$L_1 = 2601$$

Langkah berikutnya adalah menghitung nlai L kembali tetapi untuk posisi *pixel* yang sama dengan citra yang telah direduksi.

Untuk *pixel* (0, 0) dengan *pixel* (0, 1)
Nilai *red* pada *pixel* (0, 0) → R = 9
Nilai *red* pada pixel (0, 1) → R = 4
$A_1 = 9$; $B_1 = 4$
Untuk *pixel* (0, 0) dengan *pixel* (1, 0)
Nilai *red* pada *pixel* (0, 0) → R = 9
Nilai *red* pada *pixel* (1, 0) → R = 4
$A_2 = 9$; $B_2 = 4$
Untuk *pixel* (0, 0) dengan *pixel* (1, 1)
Nilai *red* pada *pixel* (0, 0) → R = 9
Nilai *red* pada *pixel* (1, 0) → R = 8
$A_3 = 9$; $B_3 = 8$

Perhitungan akan menggunakan rumusan *Euclidean Distance* sebagai berikut:

$$L_2 = L(A_i, B_i) = \sum_{i=1}^{3} (A_i - B_i)^2$$

$$L_2 = (9 - 4)^2 + (9 - 4)^2 + (9 - 8)^2$$

$$L_2 = 25 + 25 + 1$$

$$L_2 = 51$$

Jadi total nilai reduksi yang telah dilakukan adalah sebesar:

$$\text{Reduksi} = 2601 - 51 = 2550$$

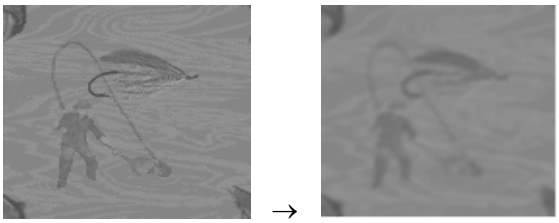
$$\text{Atau } \frac{2550}{2601} x\,100\% = 98\%$$

3.2 Region

Algoritma *intensity color checking* hanya sebatas pengambilan warna citra dan menset dengan nilai yang baru sehingga didapat hasil penggantian warna *pixel* yang baru. Agar dapat diterapkan maka algoritma di atas masih harus digabungkan dengan algoritma pemilihan *region*.

3.3 Efek Blur

Untuk memperindah hasil efek *red eye* maka pada program yang dibuat akan ditambahkan opsi untuk membuat hasil dengan efek *blur*. Tujuan dari efek *blur* ini adalah membuat suatu citra menjadi lebih kabur. Efek ini dapat dihasilkan dengan cara mengambil nilai RGB dari tiap *pixel*, kemudian menambahkan dengan nilai RGB dari *pixel* tetangganya. Hasil pertambahan kemudian dibagi dengan suatu bilangan bulat untuk mendapatkan nilai RGB yang baru seperti pada Gambar 3.



Gambar 3 Contoh Efek *Blur*

Untuk mengaplikasikan efek Blur ini ke dalam citra maka digunakan fungsi Blur yang dirancang dalam bentuk *module*. *Module-module* ini merupakan fungsi API (*Application Programming Interface*) *Windows* yaitu memanggil fungsi yang terdapat dalam GDI32.DLL yaitu sebuah *file* pustaka yang berisi fungsi-fungsi dasar penanganan grafik sistem operasi *Windows*.

3.4 Antarmuka Perangkat Lunak

Antarmuka merupakan suatu media interaksi (interaktif) antara komputer dengan pemakai (*user*). Pada sistem operasi yang berbasis grafis (*graphic user interface* atau GUI) seperti *Windows*, antarmuka dari suatu perangkat lunak biasanya berupa jendela (*window*). Melalui jendela ini pemakai dapat berinteraksi dengan perangkat lunak yang digunakannnya.

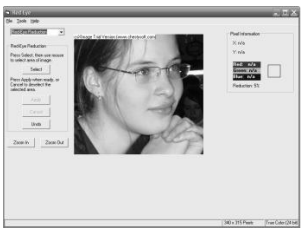
3.5 Implementasi Sistem

Implementasi sistem program ini mencakup spesifikasi kebutuhan perangkat keras (*hardware*) dan spesifikasi perangkat lunak (*software*). Untuk menggunakan program ini, jalankan *file executeable* (RedEyeRemoval.EXE) dari lokasi di mana *file* tersebut di-*copy*-kan, ataupun jika program diinstalasi maka *icon shortcut* program dapat dijalankan dari tombol Start → Program Windows.

Setelah selesai *user* dapat menyimpan dengan menekan mengakses melalui menu "*Save*" ataupun "*Save As*" bila akan disimpan dengan nama lain ataupun untuk mencetak gambar tersebut dapat mengakses melalui menu *File* → *Print*.

Untuk memperhalus citra ini maka *user* dapat menggunakan efek blur untuk sedikit membuat kabur citra dengan memilih pada pilihan blur pada bagian *combo box*.

Untuk melakukan pencetakan maka dapat dilakukan dengan memilih pada menu *File* → *Print*. Setelah itu akan ditampilkan *dialog* pencetakan seperti Gambar 4.9 berikut ini. Pada *dialog* ini terdapat pilihan untuk memilih jenis *printer* yang aktif pada sistem dan ukuran kertas melalui pilihan *paper size*. Selain itu juga terdapat pilihan apakah akan dilakukan pencetakan secara mendatar ataupun secara vertikal. Penentuan jarak posisi kiri atas dan unit pencetakan dapat diatur pada pilihan berikutnya. Jumlah kopian dapat ditentukan pada bagian "*Copies*" dan skala pencetakan dapat dipilih melalui pilihan "*Scale*".



Gambar 4. Tampilan proses penghilangan Red Eye


Gambar Awal


Sesudah proses


Gambar Awal


Sesudah proses

Gambar 5. Contoh output

4. Kesimpulan

Perangkat lunak yang dirancang dapat dipakai untuk menghilangkan *red eye* yang sering dihasilkan pada pemotretan gambar menggunakan kamera. Dengan menggunakan teknik *intensity color checking* maka proses penggantian warna merah pada bagian *red eye* gambar terlihat halus dan bagus. Bagian *red eye reduction* ini hanya dapat diterapkan pada suatu region titik tertentu pada citra dengan menginput radius dari titik untuk menentukan daerah yang akan dihilangkan *red eye*-nya.

Referensi

Buku Teks :

[1] Ahmad, Usman, *Pengolahan Citra Digital & Teknik Pemrogramannya*, Cetakan Pertama, Graha Ilmu, Yogyakarta, 2005.
[2] Basuki, Achmad, Jozua F. Palandi, Fatcurrochman, *Pengolahan Citra Digital Menggunakan Visual Basic*, Cetakan Pertama, Graha Ilmu, Yogyakarta, 2005.
[3] Dunteman, G. H., *Principal Components Analysis*, Sage Publications, 1989.
[4] Gomes, J. dan Velho, L., *Image Processing For Computer Graphics*, Translated by Silvio Levy, Springer, Rio de Janeiro, 1996.
[5] Gonzales, R. C., *Digital Image Processing*, Addison – Wesley Publishing Company., 1992
[6] Hadi R, *Pemrograman Windows API dengan Microsoft Visual Basic*, PT. Elex Media Komputindo, Jakarta, 2001
[7] Halvorson M, *Microsoft Visual Basic 6.0 Professional Step by Step*, PT. Elex Media Komputindo, Jakarta, 2000
[8] Munir, Rinaldi, *Pengolahan Citra Digital Dengan Pendekatan Algoritmik*, Penerbit Informatika, 1992.
[9] http://www.catenary.com, tanggal akses 19 Maret 2011
[10] http://www.imaginghardware.com, tanggal akses 20 Feb 2011
[11] http://www.wotsit.com, tanggal akses 20 Feb 2011
[12] Visual Basic, http://en.wikipedi.org/visual_basic, tanggal akses 25 Januari 2011.