Implementasi Algoritma Boyer-Moore untuk Memanipulasi Foto dengan *Magic Color*

Vidia Anindhita - 13512034
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
vidianindhita@students.itb.ac.id

***Abstrak*—Di era digital saat ini manipulasi foto sudah sering sekali digunakan, mulai dari manipulasi ukuran hingga pewarnaan dari foto itu sendiri. Berbagai perangkat lunak untuk memanipulasi foto pun telah banyak dikembangkan. Manipulasi foto terutama yang berkaitan dengan warna tentunya berhubungan dengan penggunaan pixel dari sebuah foto itu sendiri. Representasi dari pixel dalam sebuah foto adalah berupa angka, sehingga dalam memanipulasi foto, angka-angka tersebutlah yang akan diubah. Algoritma string matching, dalam hal ini menggunakan algoritma Boyer-Moore, akan digunakan guna menemukan pixel-pixel yang akan diubah tersebut. Dalam makalah ini akan dijelaskan mengenai bagaimana implementasi algoritma Boyer-Moore dalam menemukan pixel yang nantinya berhubungan dalam manipulasi foto.**

***Indeks*—manipulasi, citra, algoritma Boyer-Moore, pixel, warna, RGB.**

I. PENDAHULUAN

Perkembangan zaman pada era nformasi nampaknya tidak hanya membuat aplikasi media sosial meningkat, namun juga mempengaruhi dunia fotografi dengan makin maraknya aplikasi yang bermunculan guna mendukung aplikasi media sosial tersebut. Terutama di kalangan anak muda saat ini berbagai media sosial yang digunakan telah mendukung aplikasi pengolah gambar. Berbagai perangkat lunak pengolah gambar pun semakin banyak bermunculan guna mempercantik tampilan foto yang kemudian banyak digemari oleh masyarakat.

Fotografi era digital dengan perangkat lunak pengolah gambar telah membawa perubahan besar dalam dunia fotografi. Hal ini dikarenakan meningkatnya kesadaran masyarakat akan sejauh mana hasil foto dapat dimanipulasi. Begitu banyak perangkat lunak untuk memanipulasi foto, baik itu berupa perangkat lunak offline maupun online. Tidak hanya itu, berbagai informasi terkait manipulasi foto ini pun telah menjamur di dunia maya. Manipulasi foto pada era digital seperti sekarang ini sangat mudah dilakukan oleh siapa pun, tidak terkait latar belakang profesi orang tersebut.

Kemudahan yang diberikan oleh perangkat lunak pengolah gambar membuat masyarakat sangat bergantung pada perangkat lunak tersebut tanpa mengetahui bagaimana sesungguhnya perangkat lunak tersebut bekerja dalam mengolah gambar. Tidak sedikit masyarakat yang hanya menggunakan perangkat lunak tersebut tanpa mengetahui bagaimana proses pengolahannya. Hal ini tentunya akan menyebabkan masyarakat hanya akan menjadi pengguna tanpa bisa mengembangkan perangkat lunak tersebut sehingga dapat meningkatkan kreativitas dalam memanipulasi foto, tidak terbatas pada fitur-fitur yang disediakan oleh perangkat lunak pengolah gambar tersebut.

Dalam mengembangkan perangkat lunak pengolah gambar tentunya perlu mengetahui secara mendalam bagaimana suatu gambar dimanipulasi. Suatu gambar merupakan kumpulan pixel yang direpresentasikan oleh angka-angka. Manipulasi foto terkait pewarnaan akan menggunakan pixel-pixel dari gambar tersebut untuk diubah. Dalam mengubah pixel-pixel tersebut pertama-tama pixel harus dicocokkan dengan warna yang ingin diubah. Pencocokan pixel tersebut dapat menggunakan berbagai pendekatan algoritma. Dalam makalah ini akan dipaparkan implementasi dari algoritma Boyer-Moore yang akan digunakan untuk mencocokan pixel-pixel tersebut.

II. LANDASAN TEORI

2.1 Citra Digital

Citra secara umum terbagi menjadi dua, yaitu citra kontinu dan citra diskrit. Citra kontinu merupakan citra yang dihasilkan dari sistem optik yang menerima sinyal analog, misalnya mata manusia dan kamera analog.[1] Sedangkan citra diskrit ialah citra yang dihasilkan melalui proses digitalisasi dari citra kontinu.

Citra diskrit inilah yang kemudian lebih dikenal dengan citra digital. Citra digital adalah citra *f(x,y)* yang telah dilakukan digitalisasi baik koordinat area maupun tingkat kecerahan. Nilai *f* dari koordinat *(x,y)* menunjukkan kecerahan atau *grayness level* dari citra tersebut. Berdasarkan proses digitalisasi maka pada citra digital akan dibuat kisi-kisi arah horizontal maupun vertical sehingga membentuk *array* 2 dimensi.

Di dalam computer citra digital dinyatakan dengan matriks N x M, dengan

N = jumlah baris $0 \leq y \leq N-1$
M = jumlah kolom $0 \leq x \leq M-1$

Dan direpresentasikan sebagai matriks N x M dalam bentuk sebagai berikut.

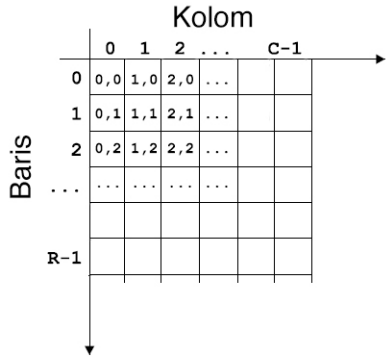
$$f(x,y)=\begin{bmatrix} f(0,0) & f(0,1) & f(0,M-1) \\ f(1,0) & f(1,1) & f(1,M-1) \\ f(N-1,0) & f(N-1,1) & f(N-1,M-1) \end{bmatrix}$$

2.2 Definisi Pixel

Pixel atau Pel, yang sejarahnya berasal dari kata akronim Bahasa Inggris yaitu Picture Element yang kemudian disingkat menjadi PIXEL, adalah sebuah unsur gambar atau representasi sebuah titik terkecil dalam sebuah gambar grafis.

Sebuah pixel secara spesifik merupakan sekumpulan angka. Secara konseptual, pixel adalah kumpulan dari angka yang merepresentasikan suatu warna.

Pada komputer, pixel ini merupakan elemen yang dialamatkan terkecil yang ditampilkan pada layar komputer sehingga elemen ini mewakili gambar di layar. Alamat dari sebuah pixel sesuai dengan koordinat fisiknya. Pixel diatur berdasarkan baris dan kolom. Baris biasanya diberi nama y, dan kolom diberi nama x. Sistem koordinat dari sebuah citra digital dapat direpresentasikan sebagai matriks berikut.



Gambar 1. Koordinat Pixel

Berdasarkan gambar di atas diketahui bahwa koordinat dari x dan y menunjukkan pixel. Nilai x atau kolom akan semakin besar dari kiri ke kanan dan nilai y akan semakin besar dari atas ke bawah di mana setiap pixel memiliki nilai berupa tingkat kecerahan.

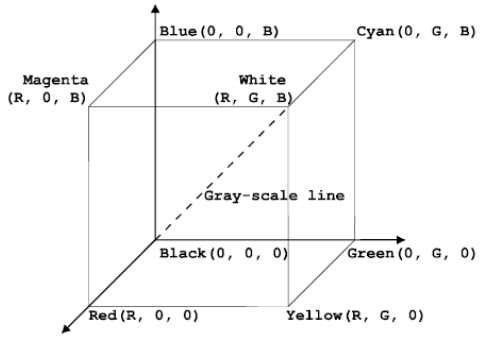
2.3 Definisi Warna RGB

Warna merupakan efek yang diciptakan dalam visual manusia karena adanya eksitasi radiasi elektromagnetik atau biasa disebut cahaya. Secara spesifik, warna adalah hasil persepsi dari cahaya di daerah spectrum elektromagnetik yang dapat dilihat, yang memiliki panjang gelombang 400nm hingga 700nm, yang tertangkap di retina manusia. Retina memiliki tiga sel reseptor warna yang berbeda-beda.

Pada teori colorimetry modern, suatu warna didasarkan pada perbedaan eksitasi dari tiga reseptor cahaya di dalam retina. Ketiga warna tersebut yaitu merah, hijau, dan biru (dalam bahasa Inggris red, green, blue yang disingkat RGB). Ketiga komponen warna ini kemudian menentukan warna suatu objek.

Perbedaan warna muncul berdasarkan tingkat intensitas dari masing-masing komponen RGB tersebut. Di dalam layar komputer terdapat bintik-bintik kecil yang dapat memancarkan warna merah, hijau, dan biru yang jika masing-masing intensitasnya ditentukan maka akan membentuk warna-warna tertentu.

Dalam sistem RGB, suatu warna dapat didefinisikan berdasarkan tingkat intensitas dari masing-masing komponen merah, hijau, dan biru. Ketiga komponen tersebut dapat digambarkan sebagai koordinat tiga dimensi, yang biasanya disebut sebagai kubus RGB. Di bawah ini merupakan kubus RGB yang merepresentasikan penggunaan RGB.



Gambar 2. Kubus RGB[2]

Berdasarkan kubus RGB tersebut dapat dilihat penggunaan RGB dengan menentukan insentitas dari masing-masing komponen. Koordinat berdasarkan warna (merah, hijau, biru) atau (R, G, B). Pada koordinat (0,0,0) berarti warna hitam karena ketiga intensitas warna tersebut 0, sedangkan bila (R, G, B) atau masing-masing komponen memiliki intensitas penuh, atau (255, 255, 255) maka berwarna putih. Tingkat intensitas tiap komponen berikisar dari 0 hingga 255. 0 adalah tingkat paling gelap sedangkan 255 merupakan tingkat paling terang.

Berikut merupakan penyimpanan pixel pada komputer berdasarkan RGB.

1 byte	1 byte	1 byte
Merah (Red)	Hijau (Green)	Biru (Blue)

Gambar 3. Penyimpanan Pixel Berdasarkan RGB

Penyimpanan RGB masing-masing komponen memerlukan 1 byte memori, sehingga untuk ketiga komponen dibutuhkan 3 byte. Format penyimpanan warna sistem RGB adalah true color, dan jumlah kombinasi warna untuk RGB adalah 2^{24} atau setara dengan 16 juta warna.

2.4 Pattern Matching

Pattern matching merupakan proses pencocokkan *string* pada sebuah teks yang diberikan. Persoalan pencarian *string* dapat dirumuskan sebagai berikut.

Diberikan:

1. Teks (*text*), yaitu long string yang panjangnya *n* karakter.
2. *Pattern*, yaitu string dengan panjang *m* karakter

($m < n$) yang akan dicari di dalam teks.[3]

Pattern matching sering diimplementasikan pada pencarian kata pada mesin pencari, analisis data dengan auto-correction, dan juga dalam hal ini adalah pencarian pixel dari sebuah gambar yang akan dimanipulasi. Berikut merupakan contoh dari *pattern matching*.

Pattern : **221, 115, 30**

Teks : 255, 0, 150, **221, 115, 30**, 101, 55, 250

Target

Dalam analisis manipulasi foto ini yang akan dicocokkan adalah pixel dari citra itu sendiri. Dalam hal ini pixel merupakan kumpulan angka RGB yang terdiri dari 3 byte yaitu jika dilihat pada contoh di atas adalah 221, 115, 30 yang artinya merah (red) dengan intensitas 221, hijau (green) dengan intensitas 115, dan biru (blue) dengan intensitas 30 adalah warna RGB yang akan diganti.

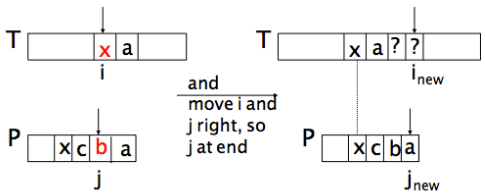
2.5 Algoritma Boyer-Moore

Algoritma Boyer-Moore merupakan salah satu algoritma yang digunakan dalam pencarian string. Algoritma Boyer-Moore ini pertama kali dijelaskan oleh R. M. Boyer dan J. S. Moore. Ide utama dari algoritma Boyer-Moore ini adalah pencarian string dengan cara melompat sejauh mungkin untuk mengurangi jumlah pergeseran.

Dengan menggunakan algoritma ini, secara rata-rata proses pencarian akan menjadi lebih cepat jika dibandingkan dengan menggunakan algoritma lainnya. Hal ini dikarenakan algoritma ini mencocokkan string dari pattern yang paling belakang, jika tidak sama maka akan langsung melompat mencari karakter yang cocok.

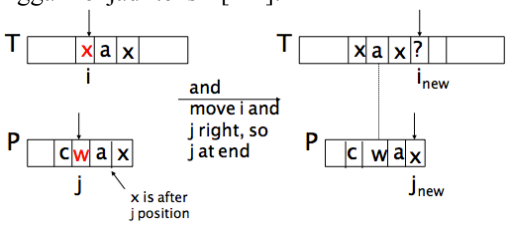
Terdapat dua teknik dalam melakukan pencarian dengan algoritma Boyer-Moore. Teknik pertama adalah teknik *looking-glass*, yaitu teknik pencarian *pattern* pada teks dengan pencocokan dimulai dari akhir *string* pada *pattern* ke depan. Sedangkan teknik kedua yaitu teknik *character-jump*, yaitu teknik menggeser pattern ketika tidak terjadi kecocokan (*mismatch*) sejauh karakter tertentu bergantung pada bagaimana ketidakcocokan itu terjadi. Karena hal itu terdapat tiga kasus dalam algoritma Boyer-Moore yang menandakan bagaimana ketidakcocokan itu terjadi.

a) Kasus pertama, yaitu jika di dalam pattern P terdapat x di suatu tempat, maka geser P ke kanan sampai tempat kemunculan terakhir x di P.



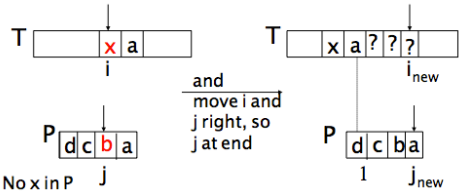
Gambar 4. Kasus Pertama Mismatch

b) Kasus kedua, yaitu ketika pattern P mengandung x namun kemunculan x berada di kanan P[i] dan untuk menggeser ke kemunculan terakhir x tersebut tidak dimungkinkan, maka geser P sebanyak satu karakter sehingga menjadi teks T[i+1].



Gambar 5. Kasus Kedua Mismatch

c) Kasus ketiga, yaitu bila kedua kasus di atas tidak terpenuhi, yaitu bila karakter x tidak terdapat pada pattern. Jika kasus ketiga ini terjadi, maka geser pattern P ke kanan sejauh teks T[i+1].



Gambar 6. Kasus Ketiga Mismatch

Sebelum melakukan pencocokan teks dengan algoritma Boyer-Moore, terdapat pra-proses yang harus diimplementasikan agar penggunaan algoritma ini semakin mudah. Pra-proses sebelum mengeksekusi algoritma ini dinamakan sebagai fungsi kemunculan terakhir (*last-occurrence*). Fungsi kemunculan terakhir ini adalah fungsi yang berguna untuk menentukan kemunculan terakhir dari tiap karakter pada pattern dan kemudian akan disimpan pada sebuah array. Misalkan menggunakan pattern sebelumnya yaitu "221, 115, 30" maka fungsi kemunculan terakhir adalah sebagai berikut.

x	0	1	2	3	5	,	<<spasi>>
L(x)	12	7	2	11	8	9	10

Gambar 7. Fungsi Kemunculan Terakhir

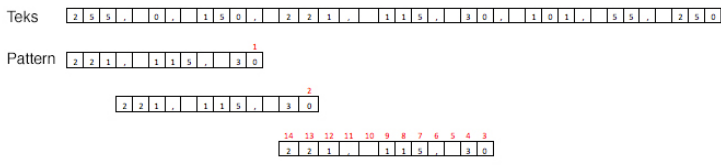
Pixel di sini merupakan sebuah string yang terdiri dari angka, serta koma dan spasi untuk memisahkan tiap komponen warna.

Pencocokan pixel sendiri dapat menggunakan algoritma Boyer-Moore. Contohnya diberikan teks dan *pattern* sebagai berikut.

Teks (T) : 255, 0, 150, 221, 115, 30, 101, 55, 250

Pattern (P) : 221, 115, 30

Maka pencarian menggunakan algoritma Boyer-Moore adalah sebagai berikut.



Gambar 8. Pencarian Pixel dengan Algoritma Boyer-Moore

Berdasarkan gambar di atas dapat dilihat bahwa dengan menggunakan algoritma Boyer-Moore dengan teks dan

pattern yang telah disebutkan sebelumnya, maka terdapat total 14 jumlah perbandingan. Jumlah perbandingan ini jauh lebih sedikit jika dibandingkan dengan menggunakan algoritma string matching lainnya seperti Brute Force atau Knuth-Morris-Pratt karena harus membandingkan string dari kiri ke kanan.

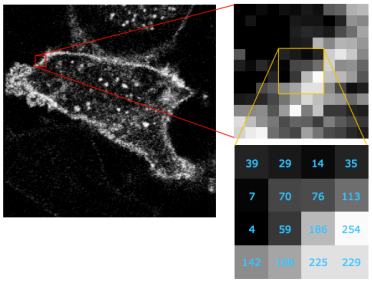
Kompleksitas waktu pencarian string dengan menggunakan algoritma Boyer-Moore dalam kasus terburuknya adalah O(*nm+A*) dengan *A* merupakan panjang string yang digunakan.

III. Analisis Pemecahan Masalah

3.1 Representasi Foto dalam Pixel

Dalam makalah ini digunakan algoritma Boyer-Moore untuk mencocokan pixel pada sebuah foto. Manipulasi foto yang akan dijelaskan pada makalah ini akan berbasis pada pengubahan pixel dari foto tersebut.

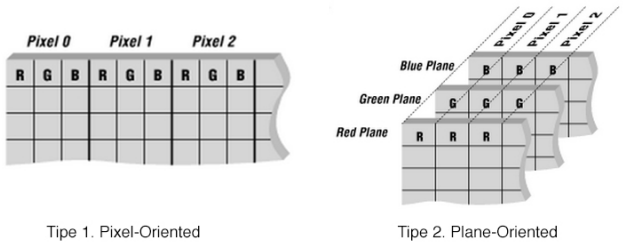
Foto merupakan sebuah citra yang terdiri dari sejumlah pixel yang membentuk warna. Semakin baik resolusi sebuah foto maka jumlah pixel dalam suatu foto akan semakin banyak. Jika sebuah foto direpresentasikan menjadi kumpulan pixel maka akan terdapat representasi pixel dengan matriks N x M. Berikut merupakan representasi foto dalam pixel dengan pewarnaan *grayscale*.



Gambar 9. Representasi Foto Grayscale[4]

Dapat dilihat pada gambar di atas bahwa jika sebuah gambar dilihat dari sangat dekat maka terlihat warna yang terkotak-kotak. Warna ini memiliki intensitas pencahayaannya masing-masing. Intensitas yang mendekati 0 akan menunjukkan warna gelap (hitam) dan warna dengan intensitas mendekati 255 akan menunjukkan warna terang (putih).

Namun dalam manipulasi foto yang akan dipaparkan pada makalah ini tidak hanya menggunakan pewarnaan *grayscale* saja, namun akan menggunakan pewarnaan RGB. Mengenai representasi pixel RGB dapat dilihat pada gambar di bawah ini.

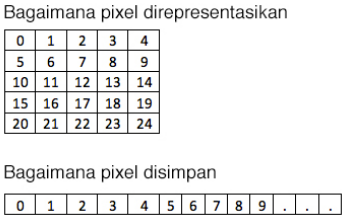


Gambar 10. Representasi Gambar RGB[5]

Terdapat dua jenis representasi dari pixel untuk pewarnaan RGB. Tipe pertama adalah pixel-oriented, yaitu pixel direpresentasikan oleh sebuah matriks, dengan setiap tiga kolom merepresentasikan sebuah pixel, yang masing-masing kolom merupakan intensitas dari warna merah, hijau, dan biru. Sedangkan tipe kedua merepresentasikan pixel RGB dengan tiga buah matriks, yang masing-masing matriks merepresentasikan sebuah warna RGB. Pada makalah ini tipe representasi pixel RGB yang digunakan adalah tipe pertama.

3.2 Analisis Pencocokan Pixel dengan Algoritma Boyer-Moore

Setelah sebuah foto direpresentasikan dalam kumpulan pixel yang berupa matriks N x M, maka pencocokan pixel kemudian akan lebih mudah. Namun sebelum dilakukan pencocokan pixel, matriks tersebut akan dikonversi menjadi sebuah *array*. Hal ini dikarenakan di dalam komputer, pixel disimpan pada sebuah *array*.



Gambar 11. Representasi Pixel dan Penyimpanan Pixel

Pada algoritma Boyer-Moore terdapat teks dan pattern yang akan dicocokan. Pada manipulasi foto dalam makala ini, teks merupakan kumpulan pixel yang telah disimpan pada sebuah sebuah *array*, dan *pattern* merupakan sebuah pixel berupa satu warna RGB. Mengenai contoh dari bagaimana sebuah pixel dicocokan dengan kumpulan pixel menggunakan algoritma Boyer-Moore telah dipaparkan pada sub-bab 2.5 tentang penjelasan algoritma Boyer-Moore.

IV. Implementasi Manipulasi Foto dengan *Magic Color* Menggunakan Algoritma Boyer-Moore

Magic color merupakan sebuah efek dalam manipulasi foto yang akan memunculkan warna yang diinginkan saja. MisalnUnya warna yang ingin dimunculkan adalah warna merah sedangkan warna selain merah akan menjadi *grayscale*.

Berikut ini adalah sebuah foto yang akan dimanipulasi dengan menggunakan efek *magic color*.

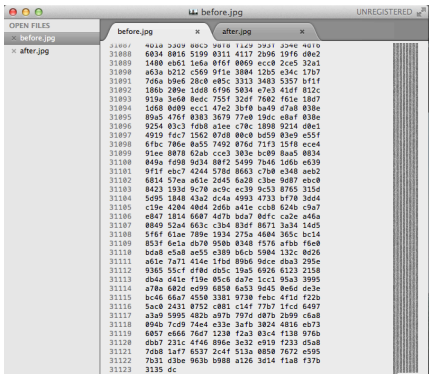
Gambar 11. Foto yang akan dimanipulasi

Dengan menggunakan efek magic color, maka foto tersebut akan diubah pewarnaannya. Warna merah akan tetap muncul namun warna selain merah akan menjadi *grayscale* seperti pada foto di bawah ini.

Gambar 12. Hasil foto yang telah dimanipulasi dengan efek *magic color*

Dengan menggunakan algoritma Boyer-Moore maka pixel dari warna yang ingin tetap dimunculkan akan dicari. Misalkan dari foto di atas, warna yang ingin tetap muncul adalah warna merah. Untuk itu pixel dari warna merah ini akan dicari dari kumpulan pixel yang membentuk foto tersebut. Kemudian pixel warna selain merah akan diubah menjadi pixel grayscale.

Berikut ini merupakan representasi warna dari foto yang belum dimanipulasi di atas bila di lihat dengan *text editor*.



Gambar 13. Representasi warna dilihat dengan text editor

Jika foto tersebut dilihat menggunakan text editor, maka akan diketahui susunan warna RGB dari foto tersebut. Namun representasi warna RGB tersebut berupa heksadesimal dari warna RGB, bukan berupa intensitas warna dari masing-masing komponen sebagaimana telah dijelaskan pada sub-bab sebelumnya. Heksadesimal di sini misalkan terdapat warna 222d3d, dengan dua angka pertama adalah untuk warna merah, dua angka selanjutnya adalah untuk warna hijau, dan dua angka terakhir adalah untuk warna biru. Bila dijadikan intensitas warna RGB, maka akan menjadi merah (red) 34, hijau (green) 45, dan biru (blue) 61. Namun pada makalah ini akan digunakan heksadesimal dalam representasi RGB.

Dengan menggunakan tipe pixel-oriented seperti yang telah dijelaskan pada sub-bab 3.1 terkait representasi foto dalam pixel, maka kumpulan pixel tersebut akan direpresentasikan menjadi sebuah array dan dibagi tiap komponen. Dikarenakan pada foto tersebut terdiri dari 155615 pixel, maka pada makalah ini hanya akan diambil sampel saja untuk menunjukkan penggunaan algoritma Boyer-Moore pada pencocokan pixel pada manipulasi foto ini. Berikut merupakan sampel dari pixel foto tersebut yang akan dicocokan menggunakan algoritma Boyer-Moore.

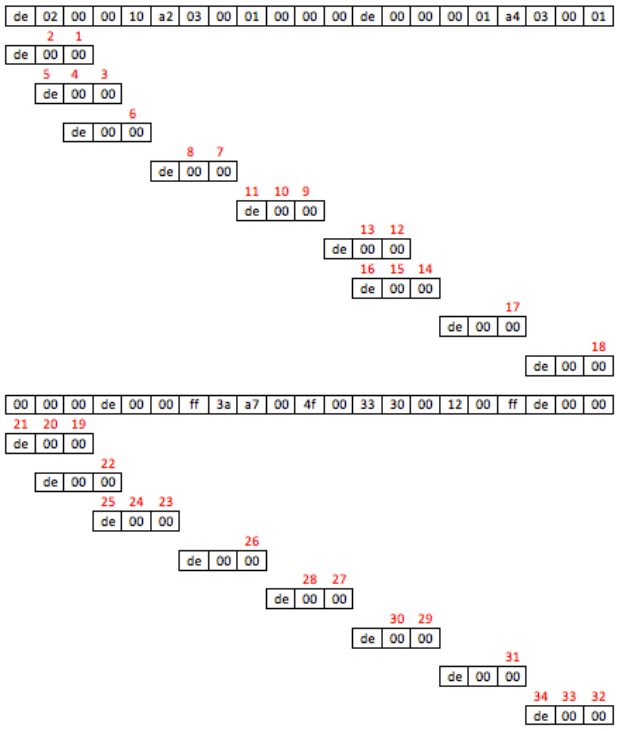
de	02	00	00	10	a2	03	00	01	00	00	00	de	00	00	00	01	a4	03	00	01

00	00	00	de	00	00	ff	3a	a7	00	4f	00	33	30	00	12	00	ff	de	00	00

Gambar14. Sampel pixel yang akan dicocokan

Pixel yang akan dicocokan, atau dalam hal manipulasi ini adalah warna yang akan tetap dimunculkan adalah "*de0000*" yang berarti warna merah. Karena seluruh warna merah dari foto tersebut akan dimunculkan, maka pencocokan string akan dilakukan hingga seluruh teks dicocokan (tidak berhenti ketika string ditemukan).

Sebelum melakukan manipulasi warna terhadap foto tersebut, dilakukan pencocokan pixel dengan menggunakan algoritma Boyer-Moore. Berikut pencocokan pixel "*de0000*".



Gambar 15. Pencocokan pixel dengan Boyer-Moore

Berdasarkan hasil di atas dapat dilihat bahwa dengan menggunakan algoritma Boyer-Moore untuk sampel pixel dari foto tersebut terdapat 34 total pencocokan. Langkah selanjutnya adalah bagaimana memanipulasi foto tersebut dengan mengganti pixel-pixel yang bukan "*de0000*" dengan *grayscale*.

Representasi pixel dalam memanipulasi foto di sini adalah dengan menggunakan *array*. Masing-masing komponen warna RGB memiliki indeks pada *array*. Saat algoritma pencocokan string di jalankan, ketika pixel ditemukan pada teks (kumpulam pixel), maka indeks dari komponen-komponen pixel yang sama tersebut akan di simpan pada *array* baru yang menampung indeks dari

string yang cocok. Selanjutnya akan dilakukan manipulasi foto dari *true color* RGB menjadi *grayscale* terhadap pixel yang tidak sama dengan *pattern*. Langkah-langkahnya adalah dilakukan iterasi ulang pada *array* pixel tersebut, lalu lihat pada *array* indeks yang menyimpan indeks dari *string* yang cocok. Untuk setiap pixel pada *array* pixel yang tidak terdapat pada *array* indeks, maka lakukan pengubahan menjadi *grayscale*, setelah itu lompat sebanyak panjang pattern pada array pixel agar pixel yang sama tidak diganti menjadi *grayscale*. Pada *array* indeks, lanjut ke indeks seterusnya dan lakukan iterasi kembali pada array pixel, ubah setiap pixel yang tidak ada pada array indeks demikian seterusnya.

Berikut merupakan *pseudo-code* untuk mengganti pixel selain pixel "*de0000*" dengan warna *grayscale*.

```
int i;
int a = 0;
int b = 0;

for (i = 0; i < Text.length(); i++) {
      if (true) {
            ArrIndeks[b] = i;
            b++;
      }
      i = i + pattern.length();
}

while (b<ArrIndeks.length()&&a< Text.length())
{
      for (i = a; i < ArrIndeks[b]; i++) {
            ChangetoGrayscale();
            a = i;
      }

      a = a + pattern.length();
      b++;
}
```

Dalam mengubah warna *true color* RGB menjadi *grayscale* dapat digunakan beberapa algoritma. Pada makalah ini hanya akan diimplementasikan satu algoritma saja dalam mengubah warna menjadi *grayscale*. Algoritma pengubahan warna menjadi *grayscale* adalah sebagai berikut.

```
for each Pixel in photo {
      Red = Pixel.Red
      Green = Pixel.Green
      Blue = Pixel.Blue

      Gray = (Red * 0.3 + Green * 0.59 + Blue *
      0.11)

      Pixel.Red = Gray
      Pixel.Green = Gray
      Pixel.Blue = Gray
}
```

Sehingga pada pixel yang bukan "*de0000*" akan diubah masing-masing menjadi pixel *grayscale* dengan algoritma di atas.

V. Analisis Implementasi

Berdasarkan implementasi manipulasi foto seperti yang telah dipaparkan pada bab sebelumnya, dapat diketahui bahwa pencocokan pixel guna memanipulasi foto dapat dilakukan dengan menggunakan algoritma Boyer-Moore. Algoritma pencocokan string ini akan mencari pixel warna yang akan tetap dimunculkan di foto sekaligus menyimpan indeks dari pixel yang sama tersebut ke dalam array baru.

Dalam hal ini dilakukan dua kali iterasi untuk memanipulasi sebuah foto. Iterasi pertama digunakan untuk mencocokan pixel sekaligus menyimpan indeks pixel tersebut. Dan iterasi kedua adalah untuk mengubah warna selain bukan pixel yang telah ditentukan dengan grayscale. Berdasarkan hal tersebut dapat dilihat bahwa untuk memanipulasi foto dengan mencocokkan pixel dengan algoritma Boyer-Moore dirasa kurang efektif karena iterasi yang dilakukan harus dua kali. Algoritma *image processing* dengan menggunakan rumus matematika seperti pada MATLAB akan jauh lebih efektif dalam memanipulasi sebuah foto.

Pada prosesnya setiap kali dilakukan manipulasi akan dibuat sebuah *array* baru yang akan menampung indeks dari pixel yang cocok. Penyimpanan sebuah *array* tentunya akan lebih memakan tempat pada memori. Sehingga setiap kali proses dilakukan, memori pada komputer akan lebih banyak terpakai bila hanya menggunakan algoritma *string matching* ini saja untuk memanipulasi sebuah foto.

Untuk meningkatkan efektivitas proses manipulasi foto, maka tidak hanya mengandalkan penggunaan algoritma *string matching* saja, namun masih harus menggunakan algoritma-algoritma lain seperti algoritma pada *image processing* berdasarkan kalkulasi matematika sehingga tidak perlu dilakukan iterasi sebanyak dua kali dan memori yang digunakan pada komputer akan jauh lebih sedikit.

VI. Kesimpulan

Penggunaan algoritma Boyer-Moore dalam mencocokan pixel dalam proses manipulasi sebuah foto lebih efektif dibandingkan dengan menggunakan algoritma Brute Force ataupun Knuth-Morris-Prath dalam hal pencocokan string. Hal ini dikarenakan pada algoritma Boyer-Moore string yang dicocokan adalah dari kanan ke kiri sehingga jumlah pencocokan akan jauh lebih sedikit dan lebih efektif. Namun dalam manipulasi foto secara keseluruhan, algoritma Boyer-Moore yang digunakan untuk mencocokan pixel akan kurang efektif jika pada proses manipulasi foto ini hanya digunakan algoritma string matching saja untuk mengganti pixel menjadi warna yang diinginkan. Karena jika hanya menggunakan algoritma pencocokan string saja, harus dilakukan dua kali iterasi untuk menemukan pixel kemudian mengganti pixel tersebut. Memori yang dibutuhkan untuk menjalani proses manipulasi foto ini juga lebih besar jika hanya menggunakan algoritma pencocokan string tanpa menggunakan algoritma lain dalam memanipulasi foto tersebut. Karena terdapat beberapa algoritma image processing yang lebih baik

digunakan dalam manpulasi foto.

VII. UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih kepada Tuhan YME atas karunia-Nya karena telah selesainya makalah ini. Tak lupa penulis mengucapkan terima kasih kepada Dr. Ir. Rinaldi Munir dan Ibu Masayu Leila Khodra atas bimbingannya dalam memberikan tugas makalah ini serta ilmu yang diberikan mengenai strategi algoritma. Pihak-pihak yang terkait dalam pembuatan makalah ini pun tak lupa penulis mengucapkan terima kasih atas bantuannya selama proses pengerjaan makalah ini. Akhir kata penulis mengucapkan terima kasih kepada Institut Teknologi Bandung atas dukungan terhadap selesainya makalah ini.

REFERENCES

[1] http://digilib.ittelkom.ac.id/index.php?option=com_content&view=article&id=840:definisi-citra-digital&catid=18:multimedia&Itemid=14
[2] http://www.mathworks.com/company/newsletters/articles/how-matlab-represents-pixel-colors.html
[3] Munir, Rinaldi. 2007. Diktat Kuliah IF2211. Strategi Algoritma. Bandung: Institut Teknologi Bandung.
[4] Larsen, Morten. Image Analysis The Basics: Image.
[5] http://www.fileformat.info/mirror/egff/ch02_02.htm
[6] How Matlab Represents Color [Online] http://www.processing.org/tutorials/pixels/
[7] Image Processing with Matlab [Online] http://www.getreuer.info/tutorials/matlabimaging

PERYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 19 Mei 2014

Vidia Anindhita (13512034)