Algoritma *Brute Force* dalam *Pattern Matching* pada Aplikasi Pendeteksian Potongan Citra

Ananta Pandu Wicaksana 13510077
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
ananta.pandu@students.itb.ac.id

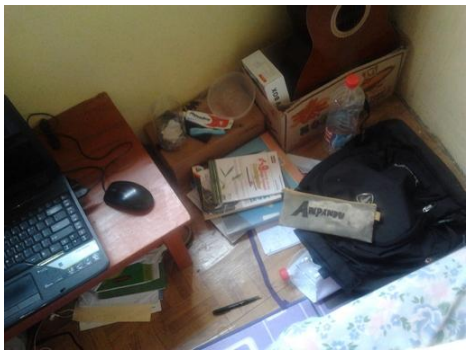
***Abstrak*—Terkadang, kita ingin mencari letak suatu potongan citra pada citra lain, misalkan wajah teman kita pada foto kelulusan SMA, atau benda tertentu pada foto kamar yang berantakan. Untuk itu, dibuatlah aplikasi sederhana pendeteksi potongan citra yang menerapkan prinsip *Pattern Matching* dengan menggunakan algoritma *Brute Force*.**

***Kata Kunci* — Citra, Potongan, Matriks, *Pattern Matching*, *Brute Force*.**

I. PENDAHULUAN

1.1 Latar Belakang

Citra, atau gambar, adalah benda yang sangat penting dalam kehidupan kita. Citra dapat memiliki makna atau informasi yang jauh lebih banyak dibandingkan dengan sebuah kalimat. Benda-benda yang tergambar pada sebuah citra adalah informasi-informasi yang bisa kita ambil. Citra ada yang memiliki sedikit informasi, ada pula yang memiliki banyak informasi. Pada citra yang memiliki banyak informasi, terkadang kita mengalami kesulitan untuk mengambil informasi yang benar-benar kita butuhkan. Sebagai contoh: mencari sebuah benda pada sebuah foto kamar yang berantakan.



Gambar 1 - Foto kamar yang berantakan

Gambar 2 - Foto sebuah benda. Pertanyaannya adalah: Adakah potongan foto ini dalam Gambar 1? Jika ada, dimanakah letaknya?

1.2 Tujuan

Makalah ini dibuat dengan tujuan :

a. Menambah pemahaman tentang pemecahan persoalan menggunakan algoritma *brute force*.
b. Menambah pemahaman tentang pengolahan citra.
c. Membuat pengaplikasian solusi dalam bentuk program.

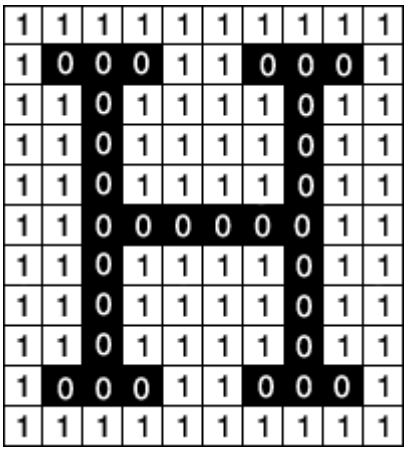
II. DASAR TEORI

2.1 Citra Digital

Citra Digital adalah foto elektronik yang diambil dari , di antara lain: scan dari dokumen, foto, manuskrip, teks, dan karya seni. Citra digital dipetakan sebagai larik dua dimensi (matriks) dari titik atau elemen gambar yang biasa disebut dengan pixel. Setiap pixel diberi nilai tonal (hitam, putih, abu-abu atau warna) yang diwakili dalam kode biner (nol dan satu). Angka biner, yang biasa disebut bit, untuk setiap pixel disimpan secara berurutan oleh komputer dan sering dikurangi menjadi representasi matematika (terkompresi). Bit tersebut kemudian

diinterpretasikan dan dibaca oleh komputer untuk menghasilkan versi analog untuk ditampilkan atau dicetak.[1]

Seperti pada contoh citra dwiwarna dibawah, bit merepresentasikan warna dari sebuah titik pada citra. Pada gambar biasa, satu bit dapat merepresentasikan bermacam-macam warna.



Gambar 3 - Seperti terlihat pada gambar dwiwarna sederhana ini, setiap pixel diberi nilai tonal, dalam hal ini 0 contoh untuk hitam dan 1 untuk putih.

Oleh karena citra digital terdiri dari larik dua dimensi (matriks) berisi bit, seringkali disebut dengan *bitmap*.

2.2 *Pattern Matching*

Pattern Matching adalah sebuah teknik dalam analisis data otomatis, biasanya dilakukan pada komputer, dimana sekelompok sifat karakteristik dari suatu obyek yang tidak diketahui dibandingkan dengan satu set sifat karakteristik dari suatu objek yang diketahui, untuk menemukan identitas atau klasifikasi yang tepat dari objek yang tidak diketahui. [2]

Pattern Matching adalah sebuah teknik pencarian string yang berisi data teks atau biner untuk beberapa set karakter yang didasarkan pada pola pencarian tertentu. [3]

2.3 Algoritma *Brute Force*

Brute Force pendekatan yang lempang untuk memecahkan suatu masalah (*straightforward*). *Brute Force* biasanya didasarkan pada pernyataan masalah (*problem statement*) definisi konsep yang dilibatkan. Algoritma brute force memecahkan masalah dengan sangat sederhana, langsung, jelas (obvious way). Intinya: lakukan saja. [4]

Pada masalah pencarian dalam struktur data larik, Algoritma *Brute Force* menelusuri setiap elemen larik. Pada masalah pengurutan elemen larik, Algoritma *Brute Force* membandingkan setiap elemen yang ada pada larik untuk mementukan urutan. Pada permasalahan penelusuran seperti yang telah disebut diatas dan yang lainnya, algoritma menelusuri setiap elemen dengan iteratif, dari awal sampai akhir, tidak rekursif dan tidak menggunakan teknik non-iteratif lainnya.

Algoritma *Brute Force* memiliki kelebihan dibandingkan dengan algoritma lain: Algoritma *Brute Force* dapat digunakan untuk memecahkan hampir sebagian besar masalah (*wide applicability*). Algoritma *Brute Force* sederhana dan mudah dimengerti. Algoritma *Brute Force* menghasilkan algoritma yang layak untuk beberapa masalah penting seperti pencarian, pengurutan, pencocokan string, perkalian matriks. Algoritma *Brute Force* menghasilkan algoritma baku (standard) untuk tugas-tugas komputasi seperti penjumlahan/perkalian n buah bilangan, menentukan elemen minimum atau maksimum di dalam tabel (list).

Di lain hal, Algoritma *Brute Force* Metode juga memiliki kelemahan: Algoritma *Brute Force* jarang menghasilkan algoritma yang mangkus. Beberapa Algoritma *Brute Force* lambat sehingga tidak dapat diterima. Selain itu juga tidak sekontruktif/sekreatif teknik pemecahan masalah lainnya.

2.4 Matriks

Dalam matematika, matriks adalah dua dimensi yang terdiri dari angka, simbol, atau ekspresi, yang diatur dalam baris dan kolom. Sebuah individu dalam matriks disebut elemen atau entri.

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}.$$

Gambar 4 – Sebuah matriks berukuran m x n.

III. ANALISIS PEMECAHAN MASALAH

Permasalahan pencocokan gambar dapat diselesaikan dengan mula-mula memodelkan hal yang merupakan masalah utama terlebih dahulu, yaitu gambar tersebut. Seperti yang telah disebutkan pada dasar teori, gambar adalah sekumpulan pixel yang tersusun dalam larik dua dimensi, jadi pertama-tama kita modelkan gambar dengan sebuah matriks.

Matriks model akan memiliki elemen yang berupa nilai warna dari gambar tersebut. Matriks tersebut memiliki ukuran lebar x tinggi, sesuai ukuran gambar. Elemen (1,1) merepresentasikan pixel (1,1) dari gambar, dan seterusnya sampai (m,n).

Dalam kasus pencocokan citra ini, ada dua citra yang terlibat, yaitu gambar utama dan gambar potongan. Setelah selesai memodelkan kedua gambar, kita akan mendapatkan dua buah matriks, matriks gambar utama dan matriks gambar potongan.

(a)

4	1	2	3	6	7	8	5	9
6	5	8	9	4	1	3	2	7
3	9	7	5	2	8	1	6	4
9	6	4	2	3	5	7	8	1
7	3	5	1	8	4	6	9	2
8	2	1	7	9	6	4	3	5
1	4	9	6	5	3	2	7	8
5	8	3	4	7	2	9	1	6
2	7	6	8	1	9	5	4	3

(b)

8	5	9
3	2	7
1	6	4

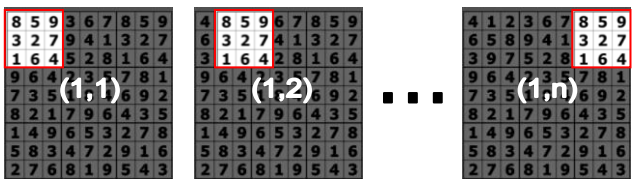
Gambar 5 – Ilustrasi matriks gambar utama (a) dan matriks gambar potongan (b).

Setelah mendapatkan matriks model, permasalahan yang semula pencarian lokasi potongan gambar menjadi pencarian lokasi potongan matriks. Pertanyaannya menjadi "Dimanakah koordinat matriks potongan di dalam matriks utama?".

Pada tahap selanjutnya, kita akan memakai konsep *pattern matching* dalam pencarian koordinat. Idenya adalah mencari tingkat kemiripan dari setiap koordinat. Kita akan menelusuri setiap elemen di matriks gambar utama untuk mendapatkan *tuple* (pasangan) koordinat-tingkat kemiripan. Pasangan-pasangan tersebut akan disimpan ke dalam sebuah larik untuk kemudian diproses, misalnya: mencari koordinat dengan tingkat kemiripan tertinggi, mencari sepuluh koordinat yang paling mirip, atau yang lainnya.

Tahap penelusuran untuk mendapatkan tingkat kemiripan akan menggunakan algoritma *brute force*. Matriks akan ditelusuri secara dua dimensi.

Gambar 6 – Penelusuran Matriks dari koordinat (1,1) sampai (m,n)



Pada gambar 6, setiap penelusuran menghitung tingkat kemiripan matriks pada koordinat yang ditelusuri. Tingkat kemiripan dihitung dengan rumus :

$$kemiripan(x,y) = 1 - \left(\frac{\sum_{\substack{0 \le i < x \\ 0 \le j < y}} kesalahan(i,j)}{(panjang\ x\ lebar)} \right)$$

Fungsi tersebut memiliki hasil dari 0 sampai dengan 1. Semakin tinggi nilai menunjukan potongan matriks semakin mirip dengan daerah yang sedang ditelusurinya.

Sedangkan fungsi kesalahan adalah fungsi yang menunjukkan sama atau tidaknya elemen potongan matriks.

$$kesalahan(x,y) = \begin{cases} 0, & A[x][y] \neq B[x][y] \\ 1, & A[x][y] = B[x][y] \end{cases}$$

Fungsi kesalahan bernilai 0 ketika elemen kedua matriks yang dibandingkan tidak sama, dan bernilai 1 ketika nilainya sama.

Dengan penelusuran *brute force* dan kedua fungsi tersebut, kita dapat memperoleh tingkat kemiripan dari setiap koordinat dan menyimpan ke dalam larik.

Tahap selanjutnya adalah mencari tingkat kemiripan tertinggi dari larik tersebut. Koordinat yang memiliki tingkat kemiripan tertinggi adalah Koordinat letak potongan gambar tersebut pada gambar utama.

IV. Implementasi dan Penerapan Algoritma

Pada tahap implementasi, penulis membuat sebuah program yang dijadikan percobaan untuk menyelesaikan permasalahan pencarian letak potongan citra ini. Program dibuat dengan bahasa Java dan dengan pustaka javax/swing sebagai antarmuka. Pada tahap pengembangan program, penulis menggunakan pustaka Marvin Framework untuk menggunakan fungsi-fungsi dasar pengolahan citra, karena di pustaka milik java penulis tidak menemukan fungsi-fungsi tersebut.

Program ini memiliki fitur yang mengizinkan pengguna menginput gambar utama dan potongannya,

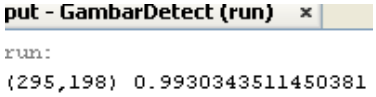
lalu menunjukkan letak/koordinat potongan gambar tersebut. Hasil pencarian ditandai dengan kotak merah pada gambar utama. Untuk lebih jelas, dapat dilihat pada gambar-gambar berikut.

Gambar 7 – Tampilan awal program

Gambar 8 – Tampilan program setelah mendeteksi potongan gambar.

Pada gambar 7 dan 8, terlihat adanya perbedaan, yaitu kotak merah yang menandai lokasi hasil pencarian. Dapat dilihat, bagian dari gambar utama yang ditandai sama persis dengan potongan gambar.

Berikut adalah koordinat beserta tingkat kemiripan dari hasil yang didapatkan.



Gambar 9 – Koordinat dari hasil, beserta tingkat kemiripannya

Koordinat dan tingkat kemiripan tampil pada output di IDE netbeans.

Berikut adalah implementasi kode dari fungsi penentuan tingkat kemiripan. Pada program yang dibuat, fungsi kemiripan dan kesalahan yang sudah disebutkan di atas digabung menjadi sebuah fungsi.

```
function getNilai(x,y: integer)→ boolean
{ Merupakan fungsi untuk mendapatkan tingkat
kemiripan }

Kamus

a : double
countbeda : integer
luas : integer
i,j : integer

Algoritma

a ← 0.0
countbeda ← 0
luas ← G2Width*G2Height
i ← 0, j ← 0

while (i< G2Width) {
    while (j< G2Height) {
        if (pixelG1(i+x,j+y)!=pixelG2(I,j))
                countbeda ← countbeda +1
        j ← j+1
    }
    i ← i+1
}
a ← 1-(countbeda/luas)

→ a
```

Pada fungsi di atas, perbandingan pixel yang dilakukan adalah sebanyak berikut.

$$\sum_{i=1}^{n}\sum_{j=1}^{n} 1 = n \times n = n^2$$

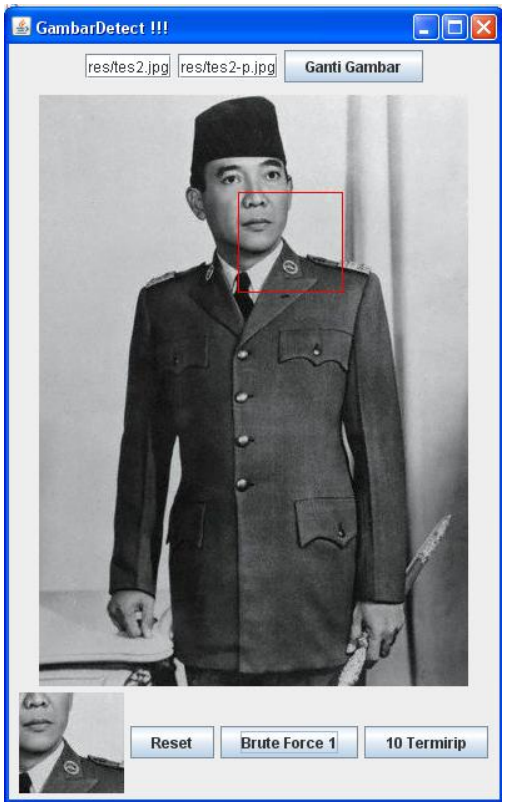
Rumus diatas memiliki kompleksitas sebesar $O(n^2)$.

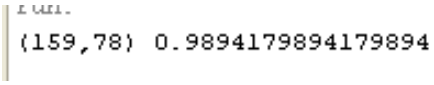
Berikut ini algoritma penelusuran yang menjalankan fungsi getNilai dan meletakkan hasilnya ke dalam larik.

```
Procedure cariLetak (larik posisi)

{prosedur pencarian tingkat kemiripan untuk
seluruh elemen matriks}

Kamus

b : double

batasx, batasy : integer

Algoritma

b ← 0.0

batasx ← (G1Width -G2Width)+1

batasy ← (G1Height -G2Height)+1

for (int i←0; i< batasx; i++)

    for (int j←0; j< batasy; j++)

            b = getNilai(i,j)

            posisi[i* batasy +j] ← <i,j,b>
```

V. HASIL PENGUJIAN DAN ANALISIS

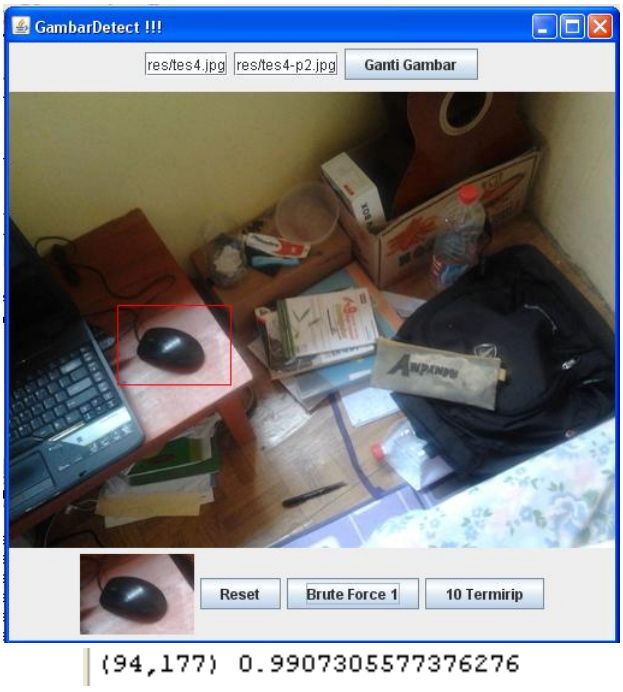
Dilakukan sejumlah pengujian terhadap program dengan melakukan variasi pada gambar utama dan potongan gambar.





Gambar 10 – Gambar Ir. Soekarno, hasil terdeteksi pada (159,78), tingkat kemiripan 0,9894.

Gambar 10 – Sebuah foto dengan potongan gambar kaki, hasil terdeteksi pada (9,123), tingkat kemiripan 0,9912.



Gambar 11 – Sebuah foto kamar berantakan dengan potongan gambar tetikus, (94,177) - 0,9907.

VI. KESIMPULAN

Kesimpulan yang dapat diambil dari pembahasan di atas antara lain:

1. Algoritma Brute Force cocok untuk menyelesaikan permasalahan pencarian tingkat kemiripan.
2. Algoritma Brute Force ini merupakan algoritma yang mampu menyelesaikan masalah dengan cara sederhana, langsung, dan jelas meskipun bukanlah algoritma yang mangkus.
3. Untuk ukuran citra yang lebih besar, proses penentuan tingkat kemiripan akan lebih lambat, sehingga diperlukan algoritma yang lebih efektif.

Saran untuk pengembangan selanjutnya:

1. Diperlukan penelitian lebih lanjut dengan menggunakan algoritma lainnya sehingga proses penentuan tingkat kemiripan ini dapat lebih efektif lagi.

REFERENSI

[1] http://www.library.cornell.edu/preservation/tutorial/intro/intro-01.html (akses: 21 Desember 2012)

[2] http://www.thefreedictionary.com/pattern+matching (akses: 21 Desember 2012)

[3] http://work.lauralemay.com/samples/perl.html (akses: 21 Desember 2012)

[4] http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/stmik.htm (akses: 21 Desember 2012)

PERNYATAAN

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, 21 Desember 2012

Ananta Pandu Wicaksana

13510077

LAMPIRAN

Berikut adalah gambar- gambar dari aplikasi. Lampiran tidak bertujuan untuk memeuhi kuota lima halaman.