PENERAPAN EUCLIDEAN DISTANCE PADA PENCOCOKAN POLA UNTUK KONVERSI CITRA KE TEKS

Febriliyan Samopa, Program Studi Sistem Informasi, FTIF, ITS
Yulianawati, Jurusan Teknik Informatika, FTIF, ITS

ABSTRAK

Pencocokan pola merupakan salah satu konsep yang sering dipergunakan dalam pada pencocokan pola. Dalam metode tersebut, citra masukan yang belum diketahui akan dicari kesamaannya dengan sekumpulan citra yang disebut sebagai citra template. Sejumlah algoritma pencocokan pola digunakan untuk mendapatkan hasil yang maksimal, dengan mengurangi nilai error dan waktu komputasi. Dalam penelitian ini digunakan algoritma untuk pencocokan pola yaitu algoritma Euclidean Distance, dan Sebagai pembanding digunakan algoritma Fast Fourier Transform. Percobaan dilakukan terhadap sejumlah objek dan sejumlah template. Citra template yang digunakan dalam penelitian ini adalah citra bitmap yang berupa karakter ASCII. Hasil akhir dari proses pencocokan pola adalah file teks yang berupa susunan karakter ASCII yang menyerupai citra masukan. Dari hasil pengamatan yang dilakukan dapat diketahui bahwa penggunaan algoritma Euclidean Distance memberikan hasil yang cukup maksimal dibandingkan dengan Algoritma Fast Fourier Transform, baik dari segi hasil file teksnya maupun dari segi waktu komputasinya.

Kata Kunci : Pencocokan Pola, Euclidean Distance, Fast Fourier Transform

1. PENDAHULUAN

1.1 Latar Belakang Masalah

Dengan semakin berkembangnya teknologi informasi dan semakin tersebarnya media digital, semakin berkembang pula metode pengolahan citra. Metode yang banyak dipergunakan dalam pengolahan citra saat ini adalah Pengenalan Pola *(Pattern Recognition).* Salah satu konsep pengenalan pola yang sangat banyak dipergunakan adalah metode pencocokan pola *(Pattern Matching).* Pencocokan pola adalah sebuah metode dengan mencari kesamaan pola antara suatu citra masukan yang belum diketahui dengan sekumpulan citra yang disebut sebagai citra template. Metode ini merupakan konsep yang sederhana dan dapat dipergunakan untuk mendeteksi kesamaan dari dua buah citra.

Pencocokan pola yang sering dilakukan adalah dengan mencocokkan sebuah citra masukkan dengan beberapa citra template. Semakin banyak citra template yang dipergunakan, semakin besar juga waktu komputasi yang dibutuhkan. Besarnya waktu komputasi yang diperlukan untuk proses pencocokan pola merupakan kelemahan dari proses ini. Besarnya waktu komputasi berbanding lurus dengan banyaknya jumlah citra template yang dipergunakan.

Salah satu penerapan dari pencocokan pula ini adalah untuk melakukan konversi dari citra bitmap menjadi teks. Algoritma yang dipergunakan untuk pencocokan pola pada penelitian ini menggunakan algoritma *Euclidean Distance (Euclidian Distance)* dan sebagai pembanding akan digunakan algoritma Transformasi Fast Fourier *(Fast Fourier Transform).* Hasil *output* dari sistem ini adalah file teks yang membentuk pola seperti citra masukan, selanjutnya dianalisa untuk mengetahui dari kedua algoritma tersebut yang menghasilkan *output* lebih mendekati gambar sebenarnya serta membandingkan waktu komputasi dari kedua algoritma tersebut. Dari hasil uji coba dan analisa dari kedua algoritma tersebut akhirnya dapat diketahui algoritma yang cocok diterapkan dalam metode pencocokan pola untuk permasalahan ini.

1.2 Masalah

Dalam penelitian ini dibangun suatu sistem untuk melakukan konversi dari citra bitmap menjadi teks. Permasalahan yang dihadapi adalah bagaimana mengubah citra masukan yang berupa citra bitmap menjadi karakter ASCI yang disusun menyerupai citra masukan. Dalam metode pencocokan pola pada sebuah citra dibagi dalam dua tahap, yaitu :

1. **Off Line Processing**
   Dalam tahap ini dibagi dalam dua proses, yaitu proses pengolahan citra template dan proses pengolahan citra masukkan.
   - Dalam proses pengolahan citra template dilakukan pelatihan terhadap citra template dengan cara membuat bitmap citra template yang berupa karakter ASCII dari *font Courier New* ukuran 8. Untuk selanjutnya citra template disegmentasi untuk membentuk matriks dengan ukuran 9 X 7 yang sesuai dengan ukuran piksel dari karakter tersebut. Tidak semua karakter ASCII bisa digunakan, karena ada beberapa

karakter ASCII yang memberikan efek tertentu bagi komputer, sehingga hasil *output* dari sistem tersebut tidak seperti yang diharapkan. Karekter ASCII yang dapat dipergunakan dalam proses ini hanya karakter ASCII dengan kode ASCII 32 sampai dengan 126. Untuk metode *Euclidean Distance*, citra template yang telah dibentuk dapat langsung dipergunakan dalam proses selanjutnya, sementara untuk algoritma *Fast Fourier Transform*, citra template yang telah terbentuk dilakukan Tansformasi Fast Fourier terlebih dahulu sebelum dimasukkan dalam tahap selanjutnya (tahap pencocokan pola).

- Proses yang kedua dari tahap ini adalah pengolahan citra masukkan. Pada proses ini memecah citra masukan menjadi matriks dengan ukuran yang sama dengan citra template (matriks 9 X 7).

2. On Line Processing

Tahap ini merupakan tahap pencocokan. Pada tahap ini dilakukan pencocokan antara citra masukkan dengan citra template dengan menghitung nilai korelasinya. Untuk algoritma *Euclidean Distance*, nilai korelasi dihitung berdasarkan rata-rata jarak antara dua titik, sedangkan algoitrma *Fast Fourier Transform*, nilai korelasi dihitung dengan menggunakan rumus Fast Fourier Transform.

Ada dua jenis ukuran kecocokan, yaitu ukuran kecocokan relatif dan ukuran kecocokan mutlak. Ukuran kecocokan relatif dipergunakan bila dalam suatu kumpulan citra telah diketahui sebelumnya terdapat tepat satu citra yang mengandung obyek serupa dengan citra template. Sedangkan ukuran kecocokan mutlak dipergunakan bila dalam suatu kumpulan citra tidak diketahui berapa jumlah citra yang mengandung obyek dari salah satu citra template. Dalam penelitian ini ukuran kecocokan yang dilakukan adalah ukuran kecocokan mutlak.

1.3 Batasan Masalah

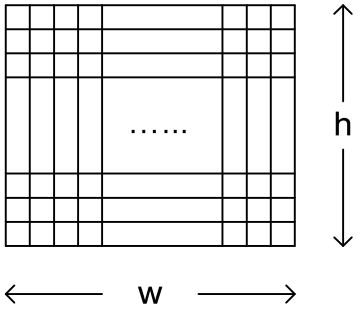
Pada penelitian ini, batasan-batasan masalah yang diberikan adalah sebagai berikut :

1. Citra template adalah suatu citra bitmap yang dibentuk dari karakter ASCII dengan ukuran 9 X 7 piksel. Karakter ASCII yang digunakan adalah *Font Courier New* dengan nomor kode ASCII 32 sampai dengan 126.
2. Citra masukkan adalah citra dalam format bitmap dengan 256 tingkat keabuan.
3. Ukuran Citra masukkan adalah citra bitmap dengan ukuran kelipatan 9 X 7 piksel, seperti 180 x 140 piksel, 270 x 210 piksel, 360 x 280 piksel. Jika ukuran citra masukan bukan kelipatan 9 x 7, maka piksel sisanya diabaikan.
4. Hasil Output dari penelitian ini adalah karakter ASCII yang tersusun menyerupai citra masukkan yang disimpan dalam format teks (TXT).
5. Aplikasi pembangun sistem ini dengan menggunakan **Visual Basic 6.0** dan menggunakan **OCX Image Gear 1.0**.

2. PENERAPAN EUCLIDEAN DISTANCE DAN TRANSFORMASI FOURIER PADA SISTEM PENCOCOKAN POLA

2.1 Citra Dinyatakan Sebagai Matriks

Sebuah citra dapat dinyatakan sebagai sebuah matriks. Apabila tinggi dan lebar citra adalah h dan w piksel, maka matriks yang akan terbentuk mempunyai ordo h x w. Misalkan sebuah citra dengan ukuran h x w pixel seperti dibawah ini :



Gambar 2.1 Contoh citra dengan ukuran h x w piksel

Maka matriks yang terbentuk adalah :

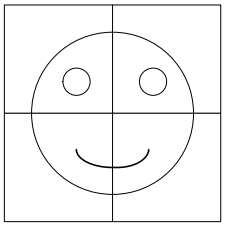
$$D = \begin{bmatrix} \gamma_{1,1} & \gamma_{1,2} & \cdots & \gamma_{1,w} \\ \gamma_{2,1} & \lambda_{2,2} & \cdots & \gamma_{2,w} \\ \cdots & \cdots & \cdots & \cdots \\ \gamma_{h,1} & \gamma_{h,2} & \cdots & \gamma_{h,w} \end{bmatrix} \qquad (2.1.1)$$

Matriks tersebut adalah representasi data digital dari sebuah citra dengan merupakan intensitas atau tingkat keabuan.

2.2 Citra dipandang sebagai sebuah vektor

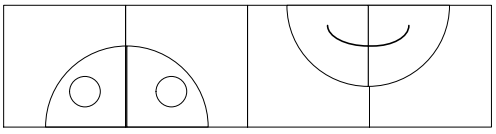
Sebuah citra dapat dipandang sebagai sebuah vektor. Apabila lebar dan tinggi adalah w dan h piksel, maka banyaknya komponen dari vektor ini adalah w x h. Setiap piksel dikodekan oleh satu komponen vektor.

Membangun vektor dari sebuah citra dilakukan dengan penggabungan sederhana, yaitu baris dari sebuah gambar diletakkan saling bersebelahan dengan baris-baris yang lain secara berurutan. Basis dari ruang lingkup citra dikomposisikan oleh vektor-vektor. Misalkan ada sebuah citra dengan ukuran 2 x 2 seperti berikut :



Gambar 2.2 Contoh transformasi citra ke vektor

Citra tersebut dapat dinyatakan sebagai :



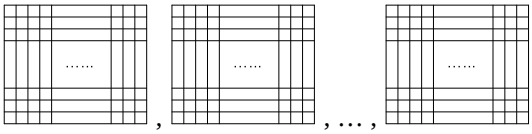
Gambar 2.3 Transformasi citra ke bentuk vektor

Gambar diatas jika ditulis dalam bentuk vektor menjadi :

$$\overline{\gamma} = [\gamma_{1,1}, \gamma_{1,2}, \gamma_{2,2}, \gamma_{2,3}] \qquad (2.2.1)$$

2.3 Transformasi Matriks Kolom

Sebuah citra dapat dinyatakan sebagai sebuah matriks. Sebuah citra dapat pula dinyatakan sebegai sebuah vektor. Misalkan sebuah basis data citra template yang terdiri dari P citra. Ukuran gambar diasumsikan N*N piksel dan tiap piksel dikodekan 8 bit atau 256 tingkat keabuah *(gray level)*. Setiap gambar dapat dipandang sebagai sebuah vektor kolom dengan ukuran (N*N)x1. Keseluruhan basis data citra dapat dinyatakan dalam sebuah matriks dimana masing-masing kolom merupakan vektor dari masing – masing citra dalam basis data tersebut. Matriks ini disebut sebagai matriks kolom. Dibawah ini misalkan citra template ke-p adalah :



Gambar 2.4 Transformasi citra ke bentuk matriks.

Citra tersebut dapat dinyatakan sebagai sebuah vektor $\gamma_{i,p}$, yaitu :

$$\gamma_p = [\gamma_{1,p}, \gamma_{2,p}, \ldots, \gamma_{N*N,p}] \qquad (2.3.1)$$

dimana $\gamma_{i,p}$ merupakan tingkat keabuan yang posisinya bersesuaian dengan gambar yang ditandai pada gambar 3.3. Untuk P citra template, matriks kolom yang dibentuk adalah sebagai berikut :

$$P_n = \begin{bmatrix} \gamma_{1,1} & \gamma_{1,2} & \cdots & \gamma_{1,p} \\ \gamma_{2,1} & \gamma_{2,2} & \cdots & \gamma_{2,p} \\ \cdots & \cdots & \cdots & \cdots \\ \gamma_{N*N,1} & \gamma_{N*N,2} & \cdots & \gamma_{N*N,p} \end{bmatrix} \qquad (2.3.2)$$

Pn adalah matriks kolom dengan jumlah citra sebanyak p, sedangkan $\gamma_{i,j}$ adalah elemen matriks kolom, i mewakili ukuran elemen matriks pada masing-masing citra sedangkan j adalah urutan citra.

2.4 Ukuran Jarak

Ukuran jarak [GON92] untuk piksel p, q, dan z dengan koordinat (x,y) dan (u,v), dimana D adalah fungsi jarak jika:

a. $D(p,q) \geq 0$ ($D(p,q) = 0$ jika $p = q$)
b. $D(p,q) = D(q,p)$ dan
c. $D(p,z) \leq D(p,q) + D(q,z)$

(2.4.1)

Euclidean Distance antara p dan q didefinisikan sebagai :

$$D(p,q) = [(x - y)^2 - (y - t)^2]^{1/2} \qquad (2.4.2)$$

Untuk ukuran jarak, piksel yang memiliki jarak kurang dari atau sama dengan harga r dari (x,y) adalah sebuah titik yang terdapat pada lingkaran dimana untuk radius r berpusat di (x,y). D4 adalah jarak atau disebut juga *city-block distance* antara p dan q didefinisikan sebagai :

$$D(p,q) = | x - s | + | y - t | \qquad (2.4.3)$$

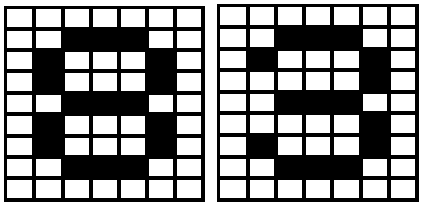
Piksel tersebut mempunyai D4, yang merupakan jarak yang dimulai dari (x,y) kurang dari

atau sama dengan nilai r yang merupakan radius dari sebuah lingkaran yang berpusat pada (x,y).

2.5 Menghitung Nilai Korelasi dengan Algoritma *Euclidean Distance*

Sebuah gambar dikatakan sama dengan gambar yang lain apabila setiap posisi dan nilai elemen pikselnya terletak pada posisi baris dan kolom yang sama. Seperti telah dijelaskan sebelumnya bahwa sebuah gambar dapat direpresentasikan sebagai sebuah matriks, sehingga dapat dikatakan sebuah gambar dianggap sama apabila setiap sel dari matriks tersebut memiliki nilai elemen yang sama pada posisi baris dan kolom yang sama.

Gambar dibawah ini menjelaskan tentang posisi piksel pada sebuah gambar dengan ukuran 9 X 7



Gambar 2.5 Citra ukuran 9 x 7 piksel

Sekilas kedua gambar tersebut terlihat sama, namun ada nilai piksel yang tidak sama. Jika gambar tersebut direpresentasikan kedalam bentuk matriks dengan nilai piksel = 1, maka :

0	0	0	0	0	0	0
0	0	1	1	1	0	0
0	1	0	0	0	1	0
0	1	0	0	0	1	0
0	0	1	1	1	0	0
0	1	0	0	0	1	0
0	1	0	0	0	1	0
0	0	1	1	1	0	0
0	0	0	0	0	0	0

0	0	0	0	0	0	0
0	0	1	1	1	0	0
0	1	0	0	0	1	0
0	0	0	0	0	1	0
0	0	1	1	1	0	0
0	0	0	0	0	1	0
0	1	0	0	0	1	0
0	0	1	1	1	0	0
0	0	0	0	0	0	0

2.6 Citra dalam bentuk matriks

Dari matriks tersebut, terlihat bahwa pada baris ke-4 kolom ke-2, baris ke-6 kolom ke-2 ada nilai piksel ang tidak sama. Jika dihitung dengan *Euclidean Distance* maka ada nilai jarak antara dua elemen matriks tersebut yang dapat dihitung dengan menggunakan persamaan :

$$d(x,y) = \| x-y\| = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + ... + (x_n - y_n)^2} \qquad (2.5.1)$$

Pada penelitian ini, sebuah citra inputan yang akan dicari kesamaanya dengan sejumlah template. Salah satu algoritma yang digunakan adalah menerapkan persamaan rumus *Euclidean Distance*.

Jika citra inputan sebagai *f(x,y)* sedangkan sebuah citra template adalah *f'(x,y)* maka untuk menghitung berapa nilai jarak antara dua citra tersebut yang dapat jgua disebut sebagai nilai error adalah :

$$e(x,y) = f'(x,y) - f(x,y) \qquad (3.5.2)$$

Pada kasus pencocokan pola ini, ukuran matriks citra template dan citra masukan adalah sama. Jika ukuran matriks tersebut adalah M x N maka nilai errr antara citra masukkan dan citra template tersebut adalah:

$$e(x,y) = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} [f'(x,y) - f(x,y) \qquad (3.5.3)$$

Yang digunakan dalam penelitian ini adalah menghitung jarak rata-rata antara citra masukkan dan citra keluaran kemudian dicari akar kuadrat nilai rata-rata dengan menggunakan persamaan :

$$e(x,y) = \left[\frac{1}{MN} \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} [f'(x,y) - f(x,y)]^2 \right]^{\frac{1}{2}} \qquad (3.5.4)$$

Untuk selanjutnya nilai tersebut disebut disebut sebagai korelasi. Kesamaan citra inputan dan citra template dapat dihitung dari nilai korelasi .Semakin kecil nilai korelasi, maka kedua citra tersebut makin sama bentuknya. Pada penelitian ini, dicari nilai korelasi yang paling kecil untuk menggantikan citra inputan dengan citra template yang berupa karakter ASCII

2.6 Menghitung Nilai Korelasi dengan Algoritma *Fast Fourier Transform* (FFT).

Metode selanjutnya yang digunakan pada penelitian ini untuk pencocokan pola dengan menggunakan Transformasi Fourier. Kesamaan antara citra masukkan dan citra template dihitung dengan mencari nilai korelasi tertinggi. Berikut ini akan dijelaskan urutan-urutan langkah untuk mendapatkan *Fast Fourier Transform.*

2.6.1 Pengembangan intuitif

Menurut E. Oran Brigham [BRIG74], Transformasi Fourier Diskrit dapat dituliskan sebagai berikut :

$$X(n) = \sum_{k=0}^{N-1} X0(k).e^{-j2\pi k/N} \qquad (2.6.1)$$

Persamaan (2.6.1) menggambarkan perhitungan dari N persamaan. Sebagai contoh, jika N = 4 dianggap :

$$W = e^{-j2\pi/N} \qquad (2.6.2)$$

Kemudian persamaan (2.6.2) bisa dituliskan sebagai berikut :

$$\begin{aligned}
X(0) &= X_0(0)W^0 + X_0(1)\,W^0 + X_0(2)\,W^0 + X_0(3)\,W^0 \\
X(1) &= X_0(0)W^0 + X_0(1)\,W^1 + X_0(2)\,W^2 + X_0(3)\,W^3 \\
X(2) &= X_0(0)W^0 + X_0(1)\,W^2 + X_0(2)\,W^4 + X_0(3)\,W^6 \\
X(3) &= X_0(0)W^0 + X_0(1)\,W^3 + X_0(2)\,W^6 + X_0(3)\,W^9
\end{aligned} \qquad (2.6.3)$$

Maka persamaan (2.6.3) dapat dituliskan dalam bentuk matriks :

$$\begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ X(3) \end{bmatrix} = \begin{bmatrix} W^0 & W^0 & W^0 & W^0 \\ W^0 & W^1 & W^2 & W^3 \\ W^0 & W^2 & W^4 & W^6 \\ W^0 & W^3 & W^6 & W^9 \end{bmatrix} \begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ X(3) \end{bmatrix} \qquad (2.6.4)$$

atau lebih tepat ditulis :

$$X(n) = W^{nk}X0(k) \qquad (2.6.5)$$

Persamaan (2.6.5) menyatakan bahwa W dan $X_0(k)$ adalah bilangan kompleks, maka ada N^2 perkalian komplek dan N(N-1) penjumlahan komplek yang dibutuhkan untuk perhitungan matriks. FFT ini sukses, karena algoritma ini mengurangi jumlah perkalian dan penjumlahan yang dibutuhkan pada perhitungan persamaan (3.6.8) . Selanjutnya akan dijelaskan bagaimana pengurangan jumlah perkalian ini bisa dihasilkan.

Untuk menjelaskan algoritma FFT, perlu untuk memilih sejumlah titik sampel dari $X_0(k)$ sehubungan dengan relasi $N = 2^{\gamma}$, dimana γ adalah integer. Persamaan (2.6.1) dihasilkan dari $N = 4 = 2^{\gamma} = 2^2$, kemudian kita dapat mengaplikasikan FFT kedalam persamaan tersebut.

Langkah pertama dalam membentuk algoritma FFT untuk contoh ini adalah menulis ulang persamaaan 2.6.4 sebagai berikut :

$$\begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ X(3) \end{bmatrix} = \begin{bmatrix} W^0 & W^0 & W^0 & W^0 \\ W^0 & W^1 & W^2 & W^3 \\ W^0 & W^2 & W^4 & W^6 \\ W^0 & W^3 & W^6 & W^9 \end{bmatrix} \begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ X(3) \end{bmatrix} \qquad (2.6.6)$$

Matriks (2.6.6) diderivasi dari persamaan (2.6.4) dengan menggunakan hubungan $W^{nk} = W^{nk \bmod (N)}$.

Dengan mengetahui bahwa [nk mod (N)] adalah sisa pembagian dari nk dengan N, jika N = 4, n=2 dan k= 3 maka :

$$W^6 = W^2 \qquad (2.6.7)$$

Karena $W^{nk} = W^6 = \exp[(-j2\pi/4(6)]$
$= \exp[-j3\pi] \quad = \exp[-j\pi]$
$= \exp[(-j\pi/4)(2)] = W^2$
$= W^{nk \bmod (N)}$

(2.6.8)

Langkah kedua dalam membentuk FFT adalah menfaktor dari matriks bujur sangkar dalam persamaan (3.6.10) sebagai berikut :

$$\begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ X(3) \end{bmatrix} = \begin{bmatrix} 1 & W^0 & 0 & 0 \\ 1 & W^2 & 0 & 0 \\ 0 & 0 & 1 & W^4 \\ 0 & 0 & 1 & W^3 \end{bmatrix} \begin{bmatrix} 1 & 0 & W^0 & 0 \\ 0 & 1 & 0 & W^0 \\ 1 & 0 & W^2 & 0 \\ 0 & 1 & 0 & W^2 \end{bmatrix} \begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ X(3) \end{bmatrix} \qquad (2.6.9)$$

Perkalian dua matriks dari persamaan (2.6.9) menghasilkan matriks persamaan (2.6.6) dengan 2 baris dan baris 1 dibalik. Anggap vektor baris yang dibalik ditunjukkan dengan :

$$\overline{X(n)} = \begin{bmatrix} X(0) \\ X(2) \\ X(1) \\ X(3) \end{bmatrix} \qquad (2.6.10)$$

Pemecahan (2.6.9) dari (2.6.6) adalah kunci dari efisiensi algoritma FFT, karena bisa mengurangi jumlah perkalian dan penjumlahan. Kita akan menguji jumlah perkalian yang dibutuhkan untuk menghitung persamaan. Pertama anggap :

$$\begin{bmatrix} X_1(0) \\ X_1(1) \\ X_1(2) \\ X_1(3) \end{bmatrix} = \begin{bmatrix} 1 & 0 & W^0 & 0 \\ 0 & 1 & 0 & W^0 \\ 1 & 0 & W^2 & 0 \\ 0 & 1 & 0 & W^2 \end{bmatrix} \begin{bmatrix} X_0(0) \\ X_0(1) \\ X_0(2) \\ X_0(3) \end{bmatrix}$$

(2.6.11)

Kolom vektor $X_1(k)$ adalah hasil dari dua matriks pada sisi kanan persamaan (2.6.5).

Elemen $X_1(0)$ dihitung dengan satu perkalian komplek dan satu penjumlahan komplek. Elemen $X_1(1)$ juga dihitung dengan satu perkalian komplek dan satu penjumlahan komplek.

$$X_1(0) = X_0(0) + W^0 X_0(2)$$
$$X_1(1) = X_1(1) + W^0 X_0(3)$$

(2.6.12)

Hanya satu penjumlahan komplek yang dibutuhkan untuk menghitung $X_1(2)$. Hal ini karena $W^0 = -W^2$ sehingga :

$$X_1(2) = X_0(0) + W^2 X_0(2) = X_0(0) – W^0 X_0(2)$$

(2.6.13)

dimana perkalian komplek $W^0 X_0(2)$ dihitung dalam $X_1(0)$. Dengan keadaan yang sama $X_1(3)$ dihitung hanya dengan satu penjumlahan dan tidak ada perkalian.

$$X_1(3) = X_0(1) + W^2 X_0(3) = X_0(1) – W^0 X_0(3)$$

(2.6.14)

vektor $X_1(k)$ didapatkan dengan 4 penjumlahan dan 2 perkalian komplek. Kita lanjutkan melengkapi persamaan (3.6.14)

$$\begin{bmatrix} X(0) \\ X(2) \\ X(1) \\ X(3) \end{bmatrix} = \begin{bmatrix} X_2(0) \\ X_2(2) \\ X_2(1) \\ X_2(3) \end{bmatrix} = \begin{bmatrix} 1 & W^0 & 0 & 0 \\ 1 & W^2 & 0 & 0 \\ 0 & 0 & 1 & W^4 \\ 0 & 0 & 1 & W^3 \end{bmatrix} \begin{bmatrix} X_1(0) \\ X_1(1) \\ X_1(2) \\ X_1(3) \end{bmatrix}$$

(2.6.15)

$X_2(0)$ didapatkan dengan satu perkalian dan satu penjumlahan komplek. Demikian juga dengan $X_2(2)$ didapat dengan satu perkalian dan satu penjumlahan.

$$X_2(0) = X_1(0) + W^0 X_1(1)$$
$$X_2(2) = X_1(2) + W^1 X_1(2)$$

(2.6.16)

$X_2(1)$ dihitung dengan satu penjumlahan saja karena $W^0 = - W^2$

$$X_2(1) = x_1(0) + W^2 x_1(1) = X_1(0) - W^0 X_1(1)$$

(2.6.17)

$X_2(3)$ juga dihitung hanya dengan satu penjumlahan karena $W^4 = - W^3$

$$X_2(3) = X_1(2) + W^3 x_1(3) = X_1(2) – W^4 x_1(3)$$

(2.6.18)

Perhitungan X(n) dari (2.6.19) membutuhkan total 4 perkalian komplek dan 8 penjumlahan komplek. Penjumlahan dari X(n) dengan (2.6.4), membutuhkan 16 perkalian komplek dan 12 penjumlahan komplek. Proses faktorisasi matriks memasukkan 0 kedalam faktor matriks, maka hasilnya bisa mengurangi jumlah perkalian yang dibutuhkan. Karena waktu perhitungan ditentukan sekali oleh banyaknya perkalian, kita bisa melihat efisiensinya algoritma FFT ini.

Untuk $N = 2^\gamma$, algoritma FFT memfaktorkan matriks N*N kedalam γ matriks (masing-masing N X N), yang masing-masing matriks yang difaktorkan sangat berguna untuk mengurangi jumlah perkalian dan penjumlahan komplek. Jika kita mengembangkan hasil dari uji coba sebelumnya, bisa dilihat bahwa FFT memerlukan $N\ \gamma/2 = 4$ perkalian komplek dan $N\ \gamma = 8$ penjumlahan komplek, dimana dengan cara langsung (2.6.4) memerlukan N^2 perkalian komplek dan N(N-1) penjumlahan komplek. Jika diasumsikan bahwa waktu perhitungan adalah sebanding dengan jumlah perkalian maka rasio perkiraan dari waktu perhitungan metode langsung dan waktu perhitungan FFT bisa dituliskan :

$$\frac{N2}{N\gamma/2} = \frac{2N}{\gamma}$$

(2.6.19)

Yang mana untuk $N = 1024 = 2^{10}$ pengurangan perhitungannya lebih dari 200 dibanding 1.
Perhitungan persamaan (2.6.9) menghasilkan X(n) sebagai pengganti X(n) :

$\overline{X(n)} = \begin{bmatrix} X(0) \\ X(2) \\ X(1) \\ X(3) \end{bmatrix}$ = sebagai pengganti dari X(n) = $\begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ X(3) \end{bmatrix}$

(2.6.20)

Pengaturan kembali adalah sifat dari proses memfaktorkan dalam matriks dan juga pengaturan kembali ini memerlukan teknik untuk penyusunan dari ke X(n). bisa ditulis dengan mengganti argumen n masing-masing dengan biner.

$$\begin{bmatrix} X(0) \\ X(2) \\ X(1) \\ X(3) \end{bmatrix} = \begin{bmatrix} X(00) \\ X(10) \\ X(01) \\ X(11) \end{bmatrix} \qquad (2.6.21)$$

Perhatikan bahwa jika argumen biner pada (2.6.21) bit-bitnya dibalik (contoh 01 menjadi 10, 10 menjadi 01) maka :

$$\overline{X(n)} = \begin{bmatrix} X(00) \\ X(10) \\ X(01) \\ X(11) \end{bmatrix} \text{ dibalik menjadi } = \begin{bmatrix} X(00) \\ X(01) \\ X(10) \\ X(11) \end{bmatrix} = \text{X(n)} \qquad (2.6.22)$$

Ini adalah cara langsung untuk mengembangkan hasil secara umum untuk pengaturan kembali FFT. Untuk N>4 tidak praktis jika menggambarkan proses faktorisasi matriks analog dengan persamaan (2.6.9).

3. UJI COBA & EVALUASI

Uji coba dilakukan dengan membandingkan antara citra masukkan dan metode pencocokan pola yang digunakan pada penelitian ini. Uji coba dilakukan pada komputer dengan spesifikasi sebagai berikut :

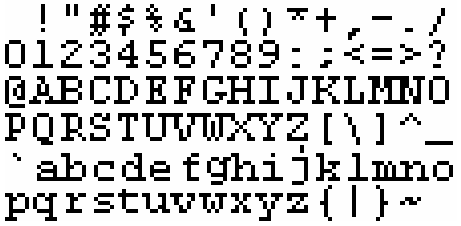
1. Processor Pentium 2.
2. Memory 256 Mega Byte
3. Hard Disk 20 Giga Byte

Dalam uji coba ini ada beberapa hal yang harus diperhatikan meliputi :

1. Citra Template :

Pencocokan pola membutuhkan citra template yang akan digunakan sebagai acuan. Citra template yang digunakan pada penelitian ini adalah citra template yang dibuat dari karakter ASCII dengan ukuran *font Courier New* ukuran 8 sehingga terbentuk file bitmap dengan ukuran 9 X 7 yang berupa karakter ASCII. Tidak semua karakter ASCII dapat digunakan, dari hasil percobaan yang dilakukan hanya karakter ASCII dengan kode ASCI 32 sampai dengan 126 saja yang dapat digunakan sebagai citra tempate. Jika kode ASCII selain nomor 32 sampai dengan 126, maka output yang dihasilkan saat penyimpanan file teks hasil pencocokan pola tidak seperti yang diharapkan. Karakter ASCII yang ditampilkan bukanlah karakter ASCII template yang dimaksud, akan tetapi akan terbentuk karakter ASCII baru.

Citra template yang digunakan pada penelitian ini adalah sebagai berikut :



Gambar 3.1 Citra Template yang digunakan

2. Citra Masukan

Citra masukkan merupakan citra bitmap dengan ukuran kelipatan 9 X 7 piksel dengan 256. Jika citra masukkan bukanlah citra bitmap dengan ukuran kelipatan 9 X 7 piksel, maka akan diambil kelipatan 9 X 7 terdekat. Misal citra masukkan berukuran 15 X 20, maka yang menjadi citra masukkan adalah citra bitmap dengan ukuran 14 X 18, sisanya diabaikan Berapapun ukuran citra masukan yang berukuran kelipatan 9 X 7 piksel akan disegmentasi menjadi berukuran 9 X 7 piksel. Jika citra masukkan berukuran 360 X 280 maka akan terbentuk 40 bagian citra inputan dengan masing-masing bagian berukuran 9 X 7 piksel. Hal ini dikarenakan ukuran citra masukkan harus sama dengan ukuran citra template.Berikut ini adalah contoh dari citra masukkan :

Gambar 3.2 Contoh citra masukkan berukuran 280 X 360 dan 371 X 477

3.1 Tahapan Uji Coba

Uji coba dilakukan terhadap proses pencocokan pola dengan menggunakan metode *Euclidean Distance* dan Transformasi Fourier. Berikut ini akan dijelaskan tahap yang dilakukan untuk kedua metode tersebut.

3.1.1 Tahap Uji coba dengan menggunakan metode Euclidean Distance

Berikut ini algoritma yang digunakan untuk melakukan proses Euclidis :

1. Citra masukkan di-*gray-scale*, hitam putih atau dithreshold.
2. Melakukan perhitungan dengan menggunakan jarak rata-rata antra dua titik untuk semua citra template dengan citra masukkan.
3. Hasil dari proses tersebut adalah nilai korelasi, untuk selanjutnya dicari nilai korelasi terkecil untuk semua template.
4. Melakukan pencocokan antara citra template dan citra masukkan dengan cara mencari korelasi terkecil antara citra template dan citra masukkan. Template yang memiliki nilai korelasi terkecil akan digunakan untuk mengganti citra inputan pada posisi yang sama pada citra masukkan.
5. Nilai *error* dihitung dari jumlah total nilai *error* dari proses pencocokan dibagi dengan jumlah potongan citra masukkan.

Berikut ini adalah ukuran bitmap yang digunakan sebagai citra inputan yang memiliki ukuran yang berbeda-beda.

Tabel 3.1 Ukuran Piksel Citra Masukan

UKURAN BITMAP DALAM SATUAN PIKSEL						
NO	NAMA FILE	FORMAT	DIMENSI	DIMENSI 9 x 7 TERDEKAT	JUMLAH POTONGAN	UKURAN (bytes)
1	Input 01	BMP	280 x 360	280 x 360	40 x 40	100,800
2	Input02	BMP	358 X 346	357 x 346	51 x 38	373,680
3	Input03	BMP	387 X 386	385 x 378	55 x 42	404,820
4	Input04	BMP	280 x 360	280 x 360	40 x 40	100,800
5	Input05	BMP	280 x 360	280 x 360	40 x 40	100,800
6	Input06	BMP	280 x 360	280 x 360	40 x 40	302,400
7	Input07	BMP	280 x 360	280 x 360	40 x 40	100,800
8	Input08	BMP	280 x 360	280 x 360	40 x 40	155,967
9	Input09	BMP	280 x 360	280 x 360	40 x 40	100,800
10	Input10	BMP	280 x 360	280 x 360	40 x 40	100,800
11	Input11	BMP	298 x 346	294 x 342	42 x 38	31,140,067
12	Input12	BMP	238 x 337	238 x 333	34 x 37	80,880
13	Input13	BMP	371 x 477	371 x 477	53 x 53	535,194
14	Input14	BMP	164 x 173	161 x 171	23 x 19	28,372

Pada Tabel 3.2 berikut ini adalah uji coba yang dilakukan terhadap 14 citra inputan dengan menggunakan algoritma *Euclidean Distance* (ED) dan citra inputan di-*gray scale*

Dari tabel 3.2 tersebut, nilai *error* terkecil ada pada citra input01,sedangkan nilai *error* terbesar pada citra input03. Sebagai perbandingan, dilakukan juga percobaan terhadap citra masukan tersebut dengan citra masukan dijadikan monokrom terlebih dahulu seperti yang dapat dilihat pada Tabel 3.3.

Dari Tabel 3.3 tersebut dapat diketahui bahwa perubahan citra masukan dari gray scale menjadi monokrom tidak merubah nilai *error*, artinya tidak terjadi perubahan pola pada piksel-piksel pembentuk citra masukan.

Dari hasil uji coba yang dilakukan, ditinjau dari hasil file teks yang terbentuk, bahwa kemiripan antara citra masukan dengan hasil file teksnya tergantung dari pola dan intensitas piksel citra masukan. Sebagai contoh citra masukan pada gambar 3.2, jika dilakukan pencocokan pola dengan *Euclidean Distance*, hasilnya seperti yang terdapat pada Gambar 3.3

Tabel 3.2 Hasil uji coba dengan citra masukan di-*gray scale*

NILAI ERROR DENGAN ALGORITMA ED BITMAP DI-GRAY SCALE			
NO	NAMA FILE	NILAI ERROR	Prosen
1	Input 01	52862.51	0.13%
2	Input02	175530.31	0.36%
3	Input03	252896.48	0.47%
4	Input04	175530.31	0.36%
5	Input05	17295.39	0.42%
6	Input06	171038.79	0.42%
7	Input07	167610.17	0.41%
8	Input08	155967	0.38%
9	Input09	159402	0.39%
10	Input10	173864	0.43%
11	Input11	11927815	0.29%
12	Input12	51380	0.16%
13	Input13	179562.88	0.25%
14	Input14	38678.76	0.38%

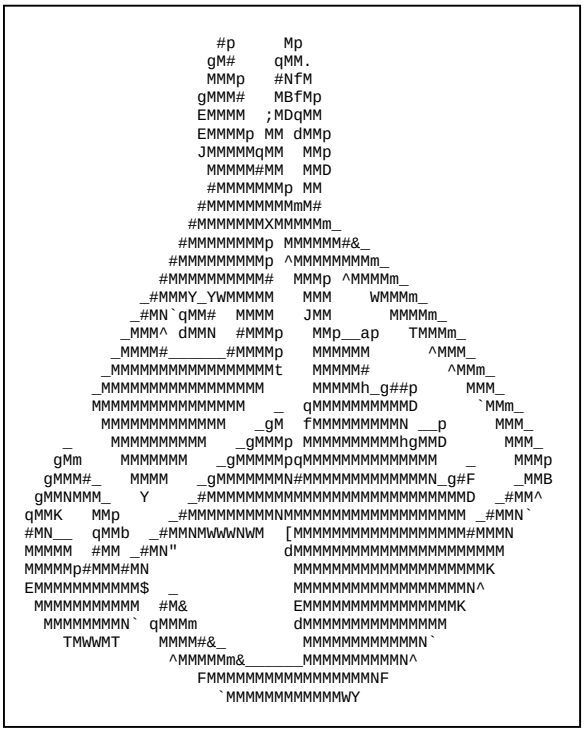
Tabel 3.3 Citra masukan dijadikan monokrom

NILAI ERROR DENGAN ALGORITMA ED BITMAP DI-BLACK WHITE			
NO	NAMA FILE	NILAI ERROR	Prosen
1	Input 01	65448.06	0.16%
2	Input02	150406.22	0.30%
3	Input03	269553.19	0.46%
4	Input04	193462.88	0.47%
5	Input05	173358.49	0.42%
6	Input06	117528.33	0.29%
7	Input07	167528.33	0.41%
8	Input08	121726.74	0.30%
9	Input09	131702	0.32%
10	Input10	187866.98	0.40%
11	Input11	110546.26	0.27%
12	Input12	51380.5	0.16%
13	Input13	196721.74	0.27%
14	Input14	3867.76	0.35%

Agar hasilnya mendekati citra masukan, beberapa hal yang perludiperhatikan adalah :

1. Citra masukan jangan terlalukecil, hal ini dikarenakan tiap 9 x 7 piksel citra masukan hanya digantikan oleh satu citra template.

2. Jangan terlalu banyak obyek dalam citra masukan, idealnya dalam satu citra masukan terdapat satu obyek yang akan dilakukan pencocokan pola.
3. Beda antara *background dan* obyek harus jelas setelah citra masukan di-*gray scale* atau dimonokrom. Jika warna *background* cukup gelap jika, maka warna obyek sebaiknya warna terang.
4. Jika citra masukan adalah citra bitmap wajah, sebaiknya garis-garispembentuk wajahnya jelas, terutamabagian mata, hidung, mulut dan telinga.



Gambar 3.3 Hasil file teks untuk citra03

Berikut ini adalah waktu komputasi yang dibutuhkan untuk kedua analisa tersebut :

Tabel 3.4 Waktu Komputasi citra masukan di-*gray scale*

LAMA WAKTU KOMPUTASI DENGAN ALGORITMA ED BITMAP DI-GRAY SCALE					
NO	NAMA FILE	PREPARING DATE TIME	PROCESS TIME	DISPLAY RESULT TIME	TOTAL COMPUTATION
1	Input 01	0:00:33	0:00:36	0:00:33	0:01:42
2	Input02	0:00:40	0:00:54	0:00:37	0:02:11
3	Input03	0:00:48	0:01:00	0:00:44	0:02:32
4	Input04	0:00:48	0:00:49	0:00:30	0:02:07
5	Input05	0:00:32	0:00:46	0:00:30	0:01:48
6	Input06	0:00:33	0:00:48	0:00:30	0:01:51
7	Input07	0:00:36	0:00:50	0:00:31	0:01:57
8	Input08	0:00:36	0:00:50	0:00:31	0:01:57
9	Input09	0:00:36	0:00:50	0:00:31	0:01:57
10	Input10	0:00:36	0:00:50	0:00:31	0:01:57
11	Input11	0:00:36	0:00:50	0:00:31	0:01:57
12	Input12	0:00:36	0:00:50	0:00:31	0:01:57
13	Input13	0:00:36	0:00:50	0:00:31	0:01:57
14	Input14	0:00:01	0:00:03	0:00:00	0:00:04

Tabel 3.5 Waktu Komputasi dengan citra masukan dimonokrom

LAMA WAKTU KOMPUTASI DENGAN ALGORITMA ED BITMAP DI-BLACK WHITE					
NO	NAMA FILE	PREPARING DATE TIME	PROCESS TIME	DISPLAY RESULT TIME	TOTAL COMPUTATION
1	Input 01	0:00:04	0:00:07	0:00:01	0:00:12
2	Input02	0:00:34	0:00:48	0:00:37	0:01:59
3	Input03	0:00:49	0:00:59	0:00:44	0:02:32
4	Input04	0:00:33	0:00:44	0:00:31	0:01:48
5	Input05	0:00:33	0:00:43	0:00:31	0:01:47
6	Input06	0:00:33	0:00:43	0:00:30	0:01:46
7	Input07	0:00:33	0:00:39	0:00:31	0:01:43
8	Input08	0:00:32	0:00:40	0:00:30	0:01:42
9	Input09	0:00:33	0:00:40	0:00:31	0:01:44
10	Input10	0:00:32	0:00:43	0:00:30	0:01:45
11	Input11	0:00:33	0:00:39	0:00:31	0:01:43
12	Input12	0:00:57	0:01:12	0:00:55	0:03:04
13	Input13	0:00:07	0:00:16	0:00:02	0:00:25
14	Input14	0:00:01	0:00:03	0:00:00	0:00:04

Berdasarkan hasil waktu komputasi tersebut dapat dilihat bahwa semakin besar ukuran piksel citra masukan semakin besar juga waktu komputasi yang dibutuhkan.

3.1.2 Tahap uji coba dengan Metode Transformasi Fourier.

Berikut ini adalah tahap yang dilakukan untuk proses FFT :

1. Citra template dilakukan FFT
2. Citra masukkan di- *gray-scale*, hitam putih atau threshold .
3. Citra masukkan dilakukan FFT
4. Hasil citra masukkan yang telah dibuat sesuai ukuran template 9 X 7 piksel diubah menjadi matriks dengan ukuran 16 X 16 piksel,dengan sisa dari 9 X 7 dijadikan warna putih.
5. Citra masukkan dilakukan FFT.

6. Hasil dari proses tersebut adalah nilai korelasi, untuk selanjutnya dicari nilai korelasi terbesar untuk semua template.
7. Melakukan pencocokan antara citra template dan citra masukkan dengan cara mencari korelasi terbesar antara citra template dan citra masukkan. Template yang memiliki nilai korelasi terbesar akan digunakan untuk mengganti citra inputan pada posisi yang sama pada citra masukkan.

Seperti juga algoritma sebelumnya, pada uji coba ini digunakan algortima *Fast Fourier Trasnform* dengan citra masukkan di- *gray scale* untuk ukuran citra masukkan yang berbeda. Jika menggunakan algoritma Fast Fourier Transform ini, waktu yang dibutuhkan sangat lama dan hasil file teks tidak seperti yang diharapkan. Besar kecilnya nilai *error* tidak bisa dijadikan acuan bahwa hasil keluaran dari proses pencocokan pola ini sukses. Pada input02 dan input03 hasil *error*-nya kecil, akan tetapi hasil output sama sekali tidak menyerupai citra masukan.

Tabel 3.6 Nilai error hasil pencocokan pola dengan *Fast Fourier Trasnform*

NILAI ERROR DENGAN ALGORITMA FFT BITMAP DI-GRAY SCALE			
NO	NAMA FILE	NILAI ERROR	Prosen
1	Input 01	1,600,000.00	3.92%
2	Input02	0.00	0.00%
3	Input03	1,600,000.00	2.70%
4	Input04	1,600,000.00	3.92%
5	Input05	1,600,000.00	3.92%
6	Input06	1,600,000.00	3.92%
7	Input07	1,600,000.00	3.92%
8	Input08	1,600,000.00	3.92%
9	Input09	1,600,000.00	3.92%
10	Input10	1,600,000.00	3.92%
11	Input11	1,600,000.00	3.92%
12	Input12	1,560,000.00	3.60%
13	Input13	1,560,000.00	2.77%
14	Input14	529,000.00	4.75%

Tabel 3.7 Waktu komputasi pencocokan pola dengan *Fast Fourier Trasnform*

LAMA WAKTU KOMPUTASI DENGAN ALGORITMA FFT BITMAP DI-GRAY SCALE					
NO	NAMA FILE	PREPARING DATE TIME	PROCESS TIME	DISPLAY RESULT TIME	TOTAL COMPUTATION
1	Input 01	0:00:03	0:22:15	0:00:02	0:22:20
2	Input02	0:00:00	0:02:10	0:00:02	0:02:12
3	Input03	0:00:03	0:02:02	0:00:02	0:02:07
4	Input04	0:00:03	1:02:15	0:00:02	1:02:20
5	Input05	0:00:03	1:02:15	0:00:02	1:02:20
6	Input06	0:00:03	1:06:59	0:00:30	1:07:32
7	Input07	0:00:03	1:00:22	0:00:01	1:00:26
8	Input08	0:00:03	1:00:00	0:00:01	1:00:04
9	Input09	0:00:32	1:00:22	0:00:09	1:01:03
10	Input10	0:00:34	1:16:59	0:00:30	1:18:03
11	Input11	0:00:03	1:02:03	0:00:01	1:02:07
12	Input12	0:00:02	0:00:14	0:00:02	0:00:18
13	Input13	0:00:34	0:16:59	0:00:30	0:18:03
14	Input14	0:00:03	0:02:01	0:00:01	0:02:05

Pada Input02, nilai *error* yang dihasilkan adalah nol, akan tetapi file teksnya tidak ada kemiripan sama sekali dengancitra masukan.

Gambar 3.4 Citra masukan input02 dan hasil file teksnya

Dari beberapa kali percobaan terhadap citra masukan, dapat dilakukan analisa bahwa citra masukan yang obyeknya terdiri dari garis tepi obyek dan garis pembentuk obyek tidak dapat dilakukan pencocokan pola dengan FFT, hasilnya sangat tidak sesuai. Namun lain dengan algoritma *Euclidean Distance*, justru obyek seperti di atas memberikan hasil yang sangat maksimal.

Selain itu waktu komputasi FFTsangat lama jika dibandingkan dengan algoritma *Euclidean Distance*. Untuk satu kali proses pencocokan akan membu-tuhkan komputasi sebesar: 16 x 16 x M x N x 95, dengan M dan N adalah jumlahsegmentasi yang terjadi padacitra inputan, 16 adalah ukuran citra masukkan dan citra template, 95 adalah jumlah template yang digunakan.

4. KESIMPULAN DAN SARAN

4.1 Kesimpulan

Dari hasil uji coba dan analisanya dapat diambil kesimpulan sebagai berikut :

1. Besar kecilnya nilai *error* tidak ditentukan oleh ukuran citra masukkan, akan tetapi ditentukan oleh intensitas dan pola dari citra masukan. Semakin besar nilai intensitas dari citra masukan, maka akan menghasilkan output yang hampir mendekati citra masukan. Jika insentitas citra masukan kecil, maka akan kesulitan dalam mendeteksi piksel obyek tersebut, dan akan dianggap putih sehingga pada hasil outputnya akan banyak informasi yang hilang. Hal ini mengakibatkan citra masukan tidak dan hasil output tidak sesuai.
2. Besar kecilnya nilai *error* tidak menentukan kemiripan antara citra masukan dan hasil output. Nilai *error* yang kecil, tidak berarti bahwa hasil output mendekati citra masukan.
3. Algoritma *Euclidean Distance* sangat cocok digunakan untuk pencocokan pola pada kasus ini jika dibandingkan dengan *Fast Fourier Transfor*m. Hasil dari sistem ini hampir mendekati dari gambar aslinya. Selain itu waktu komputasinya cukup cepat jika dibanding Fast Fourier Transform. Besarnya waktu komputasi berbanding lurus dengan besarnya citra masukan.
4. *Fast Fourier Transform* tidak cocok digunakan untuk mengimplementasikan pencocokan pola dalam kasus ini. Selain hasil file teksnya yang tidak sesuai juga waktu komputasinya sangat lama.
5. Jika ditinjau dari file teksnya, tidak semua citra bitmap dapat dijadikan sebagai citra masukan. Hanya citra masukan yang memiliki obyek dengan jelas yang dapat digunakan oleh citra masukkan, dan perbedaan warna *background* dan obyek dapat dibedakan dengan jelas. Serta garis-garis pembentuk obyek juga jelas. Citra masukan yang memiliki obyek yang banyak dalam satu. citra masukan akan mengalami kesulitan dalam pengenalan tiap obyeknya.
6. Kelemahan lain dari tugas akhir ini adalah jumlah citra template terbatas, sebab tidak semua kode ASCII dapat digunakan sebagai citra template

4.2 Saran

Dari hasil uji coba yang dilakukan terdapat kegagalan pada algoritma Fast Fourier Transform jika ditinjau dari hasil file teks dan waktu komputasinya. Untuk selanjutnya dapat dikembangkan dengan mencari algoritma lain yang lebih sesuai agar menghasilkan file teks yang mendekati citra masukan.

4. DAFTAR PUSTAKA

[GON93] Rafael C. Gonzales and Richard E. Woods, " Digital Image Processing", Addison – Wesley Publisihing Company, 1993

[BRIG74] E.Oran Brigham " The Fast Fourier Transform ", Pentice Hall Inc, 1974.

[ERS87] Okan K. Ersoy, "Fourier- Related Transforms, Fast Algorithms and Aplications", Prentice Hall, 1987.

[WIL86] William H., Brian P. Flannery, ,Saul A Teukolsky., William T Vetterling,., "Numerical Recipes " The art of Scientific Computation, Cambridge University Press, 1986.

[PIT93] Ioannis Pitas, Aristotle Universitu of Thessaloniki, " Digital Image Proccessing Algorithms", PrenticeHal Inc, 1993.

[IMG00] "ImageGear The Imaging Toolkit to Choice ",ActiveX Users's Guide All windows 32 bit Platforms, ImageGear v10, Accusoft Corporation. 2000.