PENERAPAN ALGORITMA *PATTERN MATCHING* KNUTH-MORRIS-PRATT DALAM PROGRAM *MOUSE CAM*

Kenji Prahyudi - 13508058
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
kenji_prahyudi@yahoo.com

***Abstrak*—Makalah ini membahas tentang bagaimana konsep dasar cara pembuatan program *mouse cam* dengan algoritma pencocokan *string* Knuth-Morris-Pratt. Metode – metode yang digunakan adalah metode pengolahan gambar *brightness & engraving* dan *thresholding*. Dalam prakteknya, metode dan algoritma yang digunakan untuk pembuatan program *mouse cam* masih sangat bervariasi dan belum pasti cara mana yang paling efektif dan efisien. Makalah ini akan membahas metode - metode yang diusulkan oleh penulis makalah ini sendiri. Dengan metode – metode ini, dapat dideteksi kemana gerakan pengguna untuk menggerakkan kursor, juga seberapa jauh kursor harus digerakkan.**

Indeks Kata—brightness& engraving, Knuth-Morris-Pratt, mouse cam, thresholding .

I. PENDAHULUAN

Komputer sudah menjadi teknologi yang sangat umum dan sangat berguna bagi semua kalangan. Mulai dari bermain *game*, mengerjakan tugas, membuat laporan keuangan, *browsing*, bahkan menentukan menu makan malam pun dapat dilakukan dengan komputer.

Dari semua itu, sangat disayangkan karena menurut fakta, tidak jarang orang yang tidak mau belajar menggunakan komputer dengan alasan kesulitan dalam menggunakan tetikus. Ada juga yang tidak dapat menggunakan tetikus karena keterbatasan panca indera.

Berdasarkan banyaknya kegunaan dari komputer, alangkah baiknya jika semua kalangan dapat menggunakannya. Salah satu solusi untuk membantu orang – orang yang mengalami kesulitan dalam menggunakan tetikus adalah *Mouse Cam*.

Mouse Cam adalah sebuah program yang menerima input pengguna melalui *webcam*, lalu gerakan yang dilakukan oleh pengguna digunakan untuk menggerakan kursor. Program *Mouse Cam* dapat dibuat berdasarkan pendekatan algoritma *pattern matching* Knuth-Morris-Pratt.

Algoritma Knuth-Morris-Pratt merupakan algoritma pencocokan yang tepat untuk menyelesaikan permasalahan pada kasus ini, karena perbandingan yang dilakukan hanyalah bit 0 dan 1, sehingga menyebabkan kemungkinan *pattern* dan *template* sama untuk beberapa bit secara kebetulan, lalu ternyata tidak cocok, cukup tinggi. Algoritma *pattern matching* Knuth-Morris-Pratt akan sangat efisien jika terdapat banyak persamaan pada *pattern* dan *template*.

II. METODE

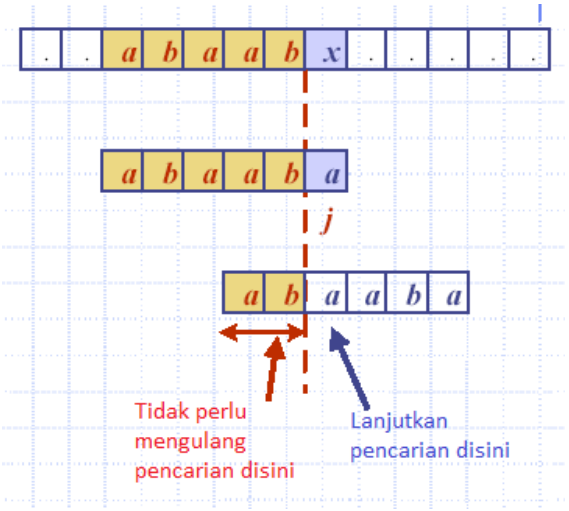
Dalam pembuatan program *Mouse Cam*, digunakan metode *pattern matching* Knuth-Morris-Pratt, metode *brightness & engraving*, dan metode *thresholding*.

A. Algoritma *Pattern Matching* Knuth-Morris-Pratt

Untuk mencocokan dua *string*, yaitu *string* sebagai *string* dikirim (*pattern)* dengan *string* sebagai *string* diacu *(template).* Definisi algoritma Knuth-Morris-Pratt adalah sebagai berikut :

- Misalkan *A* adalah alfabet dan $x = x1x2…xk$ adalah *string* yang panjangnya *k* yang dibentuk dari karakter-karakter di dalam alfabet *A*.
- Awalan (*prefix*) dari x adalah upa-string (*substring*) u dengan
- $u = x1x2…xj – 1 , j \hat{I} \{1, 2, …, k\}$ dengan kata lain, *x* diawali dengan *u*.
- Pinggiran (*border*) dari *x* adalah upa-string *r* sedemikian sehingga

  $r = x1x2…xj – 1$ dan
  $u = xj – b\ xj – b + 1 …xj$,
  $j \in \{1, 2, …, k\}$
- Dengan kata lain, pinggiran dari *x* adalah upastring yang keduanya awalan dan juga akhiran sebenarnya dari *x*.



Gambar A.1. Contoh kasus pencarian *string* dengan algoritma Knuth-Morris-Pratt

B. Metode *Brightness & Engraving*

Metode ini sebenarnya dua metode yang berbeda, yaitu metode *Brightness* dan metode *Engraving*. Dengan metode ini, sebuah gambar dapat diproses sehingga dapat dipakai untuk kebutuhan pemrograman, misalnya konversi gambar menjadi *bit list.*

Metode *Brightness* sendiri, sangat lazim digunakan pada desain fotografis, yaitu untuk membuat gambar terlihat lebih cerah atau lebih gelap dari gambar aslinya.

Metode *Engraving* adalah metode konversi gambar dengan memproses gambar biasa dengan sebuah gambar arsiran hitam putih. Sebagai contoh, di bawah ini terdapat gambar yang dikonversi dengan metode engraving dengan arsiran diagonal dari kiri bawah ke kanan atas gambar.

Gambar B.1. Gambar ruangan sebelum dilakukan metode *engraving.*

Gambar B.2. Gambar ruangan setelah dilakukan metode *brightness.*

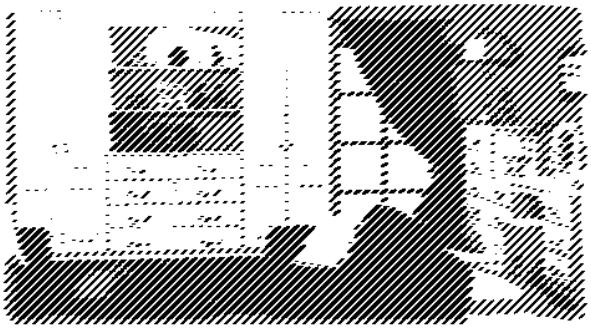
Gambar B.3. Gambar ruangan setelah dilakukan metode *engraving.*

Metode *engraving* dapat dilakukan dengan bentuk arsiran yang lain, misalnya garis tegak lurus, garis mendatar, garis berbentuk gelombang, bahkan titik – titik. Intensitas dari arsiran pun sangat memengaruhi hasil dari pemrosesan gambar.

C. Thresholding

Metode *thresholding* adalah metode untuk mengkonversi sebuah gambar yang memiliki lebih dari dua macam warna menjadi dua macam warna saja, biasanya hitam dan putih. Konversi ini dilakukan dengan mengukur 'tingkat kehitaman' dari sebuah gambar *grayscale.* Sebagai contoh saja, jika nilai RGB *(Red Green Blue)* pada sebuah pixel melebihi nilai *n,* misal 70, pixel tersebut dinyatakan sebagai hitam / bit 1, selain itu, pixel dinyatakan sebagai putih / bit 0.

Nilai *n* dapat ditentukan secara manual, atau dengan menghitung rata – rata dari tingkat kehitaman dari semua pixel yang ada, walaupun akan menggunakan lebih banyak memori.



Gambar C.1. Hasil *thresholding* dari Gambar B.3.

D. Konsep Pembuatan Program *Mouse Cam*

Pertama – tama, mari lihat contoh gambar yang akan digunakan oleh program *Mouse Cam* yang diambil dari *webcam.*

Gambar D.1. Contoh gambar dari webcam

Setelah mendapatkan gambar dari *webcam*, akan dilakukan perekaman titik – titik yang dibantu oleh tetikus (jika pengguna tidak dapat menggunakan tetikus, dapat dibantu oleh pengguna lain). Titik - titik yang diambil akan digunakan sebagai acuan perintah pengganti tetikus.

Gambar D.2. Contoh pengambilan titik - titik

Dalam gambar ini, dimisalkan lingkaran hijau adalah titik untuk gerakan kursor. Lingkaran merah adalah acuan untuk pengganti perintah klik kiri pada tetikus, sedangkan lingkaran biru untuk perintah klik kanan (Dalam prakteknya, penentuan titik sebaiknya dicerminkan, karena hasil gambar yang diterima *webcam* adalah berlawanan arah dengan arah aslinya).

Setelah menentukan titik, pengguna dapat langsung menggerak – gerakkan bagian yang 'ditunjuk' sebagai acuan penggerak kursor, dalam contoh ini adalah hidung. Untuk mendeteksi gerakan pengguna, gambar akan dimanipulasi dengan meningkatkan *brightness* gambar dan melakukan teknik *engraving* dengan bentuk arsiran titik – titik dan intensitas yang cukup tinggi. Dengan metode tersebut, gambar siap untuk diproses.

Pemilihan metode *engraving* dibandingkan dengan *grayscaling* (konversi gambar menjadi keluarga warna abu – abu) dilakukan karena saat dilakukan *thresholding* setelah gambar diproses dengan metode *grayscaling*, hasilnya terlalu monoton, yaitu akan terlalu banyak bit yang bernilai 0 atau terlalu banyak bit yang bernilai 1, sehingga akan sangat sulit untuk mendeteksi gerakan user.

Gambar D.3. Hasil manipulasi Gambar D.1

Dari gambar tersebut, kita dapat melakukan perubahan dari gambar menjadi teks dengan menggunakan metode t*hresholding.*

Selanjutnya, dilakukan metode yang sama pada gambar – gambar berikutnya yang direkam oleh *webcam* dalam durasi tertentu.

Gambar D.4. Perbandingan gambar sebelum dan sesudah bergerak

Gambar D.5. Hasil manipulasi Gambar D.4

Dari metode tersebut, dapat dilihat perubahan nilai – nilai bit pada teks hasil konversi. Sebagai contoh, misalkan hasil konversi pada daerah titik hijau pada Gambar D.2 adalah sebagai berikut :

```
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 1 1 1 0 1 1 1 0
0 1 1 1 1 0 1 1 1 1
0 1 1 1 0 0 0 0 0 1
0 0 1 1 0 0 0 0 0 0
```

Gambar D.6. Contoh bit pada daerah titik hijau pada gambar yang belum bergerak (Gambar D.2)

```
0 0 0 0 0 0 0 0 0 0
0 1 1 1 0 0 1 1 0 0
1 1 1 1 0 1 1 1 1 0
1 1 1 0 0 0 0 0 0 0
0 1 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
```

Gambar D.7. Contoh bit pada daerah titik hijau pada gambar yang sudah bergerak

Dengan menggunakan algoritma *pattern matching* Knuth-Morris-Pratt, deretan bit pada Gambar D.6

dibandingkan dengan deretan bit pada Gambar D.7. Jika pattern bit 0 dan 1 pada Gambar D.6 terdapat pada deretan bit di sebelah kiri atas pada Gambar D.7, maka dapat disimpulkan bahwa gerakan pengguna pada titik acuan penggerak kursor adalah ke kiri atas. Sehingga, kursor digerakkan ke arah kiri atas sejauh sekian pixel, tergantung sensitivitas yang diinginkan pengguna.

Hal yang perlu diperhatikan disini adalah, bit hasil konversi hampir tidak mungkin sama persis dengan pattern, karena pengaruh pencahayaan dan sebagainya, sehingga penggunaan algoritma ini dianjurkan untuk membandingkan bit dengan menggunakan persen toleransi kesalahan (misal 10%).

Untuk daerah acuan klik kiri dan klik kanan, dapat dilakukan metode yang serupa. Tetapi bedanya adalah, jika pattern tidak sesuai melebihi batas toleransi, maka akan dilakukan perintah klik. Sebagai contoh gambar, dapat dilihat pada Gambar 8 dan 9 di bawah ini.

Gambar D.8. Contoh gerakan perintah klik kiri

Gambar D.9. Hasil manipulasi Gambar D.8

Dari hasil manipulasi gambar tersebut, bit yang didapat dari gambar yang setelah bergerak diperkirakan sebagai berikut.

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

Gambar D.10. Contoh bit pada daerah titik merah pada gambar yang belum bergerak

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

Gambar D.11. Contoh bit pada daerah titik merah pada gambar yang sudah bergerak

Dengan menggunakan algoritma *pattern matching* Knuth-Morris-Pratt, dengan jelas dapat dipastikan perbedaan *pattern* pada Gambar D.10 dan D.11 melebihi toleransi kesalahan. Maka, perubahan ini diartikan oleh program sebagai perintah pengguna untuk mengeksekusi perintah titik merah (klik kiri). Jika pattern berubah kembali ke pattern yang semula, klik kiri dilepas.

Hal lain yang juga penting adalah, jika pengguna bergerak, semua titik acuan harus ikut bergerak sesuai dengan gerakan pengguna. Sebagai contoh, jika kasus seperti Gambar D.6 dan Gambar D.7, lokasi titik hijau sekarang harus bergeser sesuai dengan pergeseran bit di Gambar D.7. (Misal ke kiri atas sebanyak satu pixel).

Titik dapat diubah letaknya tergantung kemampuan pengguna. Misalnya untuk pengguna yang tidak bisa memejamkan sebelah matanya saja, akan lebih baik bagi pengguna untuk menaruh titik di tempat lain (misal mulut, pinggir mulut, dan sebagainya).

Gambar D.12. Contoh eksekusi perintah menggunakan mulut

III. *PSEUDO-CODE*

Pseudo-code yang dapat dipakai dalam program *mouse cam* adalah sebagai berikut :

```
procedure mouseCam()
        while true do
                ambil_gambar_dari_webcam()

        konversi_brightness_engrave(gambar_sek
arang)
                arrayBit_sekarang =
gambar_to_bit(gambar_sekarang)
                arah_gerak =
kmp_deteksi_arah(arrayBit_sebelumnya,
arrayBit_sekarang, TOLERANSI_ERROR)
                gerakkan_mouse(arah_gerak)
                arrayBit_sebelumnya =
arrayBit_sekarang
        end while

function kmp_deteksi_arah(bit[][]
bit_sebelumnya, bit[][] bit_sesudahnya,
integer toleransi_eror) -> arah
        arah arah_gerak
        for (i = 0; i <
BATAS_ATAS_BAWAH_PENCARIAN; i++)
                integer letak_ketemu =
kmp_search_with_error(geser(bit_sebelumnya,
i), bit_sesudahnya, toleransi_eror)

                if letak_ketemu <
LETAK_SENSOR.X then
                        arah_gerak.horizontal =
kiri

        arah_gerak.jumlah_horizontal =
LETAK_SENSOR.X - letak_ketemu
                else if letak_ketemu >
LETAK_SENSOR.X then
                        arah_gerak.horizontal =
kanan

        arah_gerak.jumlah_horizontal =
letak_ketemu - LETAK_SENSOR.X
                else
                        arah_gerak.horizontal =
diam
```

```
      arah_gerak.jumlah_horizontal = 0

              if i <
(BATAS_ATAS_BAWAH_PENCARIAN / 2) then
                      arah_gerak.vertikal =
atas
      arah_gerak.jumlah_vertikal=
BATAS_ATAS_BAWAH_PENCARIAN / 2 - i
              else if i >
(BATAS_ATAS_BAWAH_PENCARIAN / 2) then
                      arah_gerak.vertikal =
bawah

      arah_gerak.jumlah_vertikal= i -
BATAS_ATAS_BAWAH_PENCARIAN / 2
              else
                      arah_gerak.vertikal =
diam
      arah_gerak.jumlah_vertikal= 0
        return arah_gerak


function kmp_search_with_error(char[]
pattern, char[] template, integer
toleransi_eror)
      integer m = 0
      integer i = 0
      integer[] T
      integer error_sekarang = 0

      while (m + i < pattern.length) do
              if error_sekarang <
toleransi_eror then
                      if i = template.length
- 1then
                              return m
                      if template[i] !=
pattern[m + i] then
                              error_sekarang++
                      i = i + 1
              else
                      m = m + i - T[i]
                      if T[i] > -1 then
                              i = T[i]
                      else
                              i = 0


        return -1
```

Dalam *pseudo-code* di atas, terdapat satu prosedur dan dua fungsi utama, yaitu :

- prosedur mouseCam() : prosedur utama yang menjalankan tahapan – tahapan untuk menggerakkan kursor.
- fungsi kmp_deteksi_arah() : fungsi untuk mendeteksi kemana gerakan pengguna dengan menggunakan fungsi kmp_search_with_error().
- fungsi kmp_search_with_error() : fungsi Knuth-Morris-Pratt dengan toleransi error.

Sedangkan konstanta yang dipakai adalah :

- BATAS_ATAS_BAWAH_PENCARIAN : batas maksimum gerakan pengguna secara vertikal, dalam satuan piksel.
- LETAK_SENSOR.X : letak titik acuan penggerak kursor pada pengguna menurut sumbu horizontal.
- TOLERANSI_ERROR : toleransi berapa kali kesalahan yang ditoleransi dalam pencarian menggunakan Knuth-Morris-Pratt denga toleransi error.

IV. KESIMPULAN DAN SARAN

Mouse cam adalah program yang sangat cocok untuk pengguna komputer yang memiliki keterbatasan alat indera. Tetapi, untuk pengaturan titik – titik acuan penggerak kursor, masih harus menggunakan tetikus, sehingga untuk pengguna yang memiliki keterbatasan alat indera ataupun mengalami kesulitan dalam menggunakan tetikus, sebaiknya dibantu oleh orang yang mampu menggunakan tetikus.

Dalam program *mouse cam*, toleransi eror yang digunakan dalam algoritma Knuth-Morris-Pratt harus seimbang dengan konsistensi bit yang diproses dari gambar yang diambil dari webcam. Tingkat sensitivitas pada metode *brightness & engraving* juga harus disesuaikan dengan gambar yang diambil oleh *webcam*, agar hasilnya seakurat mungkin.

Metode *engraving* menggunakan arsiran titik dengan intensitas yang sangat tinggi dapat diganti dengan menaikkan kontras lalu melakukan *grayscaling*. Tapi, prosedur yang dilakukan akan lebih banyak.

Untuk mendapatkan hasil yang lebih akurat, sebaiknya dilakukan eksperimen terhadap gabungan dari metode – metode lain dalam pemrosesan gambar maupun dalam pencocokan gambarnya.

V. UCAPAN TERIMA KASIH

Makalah ini dibuat sebagai tugas dari mata kuliah IF3051 Strategi Algoritma – Sem. I Tahun 2010/2011. Penulis mengucapkan terima kasih sebesar – besarnya kepada Bpk. Ir. Rinaldi Munir, M.T. yang telah mengajar materi – materi kuliah di semester ini, sehingga makalah ini dapat dibuat dengan lancar.

REFERENSI

[1] Addison, Robert Sedgewick – Wesley, *Algorithm in C++*. 1992, ch, 19.
[2] http://en.wikipedia.org/wiki/Knuth%E2%80%93Morris%E2%80%93Pratt_algorithm
[3] http://en.wikipedia.org/wiki/Image_editing

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, 28 November 2010

Kenji Prahyudi
13508058