PERANCANGAN DAN IMPLEMENTASI *MOTION DETECTOR* PENGONTROL AKSI KURSOR *MOUSE* MENGGUNAKAN METODE *COLOR TRACKING*

Derry Aditiya[1], Budhi Irawan[2], Muhammad Nasrun[3]

[1,2,3]Fakultas Teknik Elektro, Universitas Telkom
[1]derry.aditiya@gmail.com, [2]budhiirawan@telkomuniversity.ac.id, [3]nasrun@telkomuniversity.ac.id

Abstrak

Perangkat *input* komputer yang semakin berkembang menghadirkan bentuk interaksi baru dalam memberikan perintah kepada komputer. Hal ini berawal dari perangkat *keyboard*, lalu ditambah dengan perangkat *mouse* sebagai *pointing device* pada GUI (*Graphical User Interface*). Saat ini interaksi dengan komputer menggunakan gerak tubuh manusia adalah teknologi *input* terbaru. Dengan menggunakan teknologi ini, interaksi dengan komputer dirasa menjadi lebih atraktif. Sistem pendeteksi gerak dapat dibangun menggunakan perangkat sederhana, seperti *webcam* pada komputer untuk bisa melakukan pendeteksian gerak berdasarkan optik/citra. Dalam implementasinya, deteksi gerak berdasarkan citra bisa menggunakan beragam metode yang salah satunya adalah *color tracking*. Dengan menggunakan *color tracking*, pengamatan gerak akan tertuju hanya pada objek dengan intensitas RGB yang telah ditentukan. Aplikasi ini menyajikan bentuk interaksi baru antara pengguna dengan komputer menggantikan beberapa fungsi perangkat *mouse*, yaitu memindahkan posisi kursor *mouse*, klik, dan menahan klik.

Kata Kunci: ***motion detector, mouse, color tracking, Java Media Framework***

Abstract

Computer input devices are increasingly evolving to bring new forms of interaction in giving commands to the computer. At first, the keyboard and then the mouse as a pointing device currently on the GUI (Graphical User Interface). At the moment, interaction with a computer using the motion of the human body can be regarded as the latest technological developments inputs. By using this technology, interaction with the computer is considered to be more attractive. Motion detection system can be built by using simple tools such as a webcam on a computer to be able to perform motion detection based optical/image. In the implementation, motion detection based on image could use variety of methods, one of which is color tracking. By using color tracking, so the observations of motion will be focused only on the object with the specified RGB intensity. This application presents a new form of interaction between user and computer to replace some functions of the mouse device, which are move the mouse cursor, click, and holding click.

***Keywords*: motion detector, mouse, color tracking, Java Media Framework**

1. Pendahuluan

Perangkat *input* komputer dulu hanya menggunakan *keyboard*, yang mengharuskan pengguna mengetik setiap perintah untuk menjalankan sebuah proses. Lalu muncul *mouse*, mulai marak digunakan sebagai *pointing device* setelah ditemukannya GUI (*Graphical User Interface*), dengannya pengguna bisa memberikan perintah kepada komputer hanya dengan menekan tombol dan/atau menggerakan *mouse*. Saat ini sudah banyak perangkat *input* yang digunakan pada komputer, salah satunya adalah teknologi *input* melalui sensor gerak (*motion detector*).

Color tracking merupakan pendeteksi gerak yang menggunakan pengolahan citra yang berasal dari *frame* hasil *capture* perangkat *webcam*. Terdapat beberapa keuntungan apabila teknologi deteksi gerak ini diterapkan pada komputer, yaitu interaksi antara manusia dengan komputer bisa dilakukan tanpa berkontak langsung serta menjadi lebih atraktif.

2. Video Digital [3]

Video digital pada dasarnya tersusun atas serangkaian *frame* yang ditampilkan pada layar dengan kecepatan tertentu, bergantung pada laju *frame* yang diberikan (dalam *frame*/detik). Jika laju *frame* cukup tinggi, mata manusia tidak dapat menangkap gambar per *frame*, melainkan menangkapnya sebagai rangkaian yang kontinu.

Setiap *frame* merupakan gambar digital dan setiap gambar digital direpresentasikan dengan sebuah matriks yang tiap elemennya merepresentasikan nilai intensitas. Jika I adalah matriks dua dimensi, $I(x,y)$ adalah nilai intensitas yang sesuai pada posisi baris x dan kolom y pada matriks tersebut. Titik-titik *sample* pada tempat

image yang di-*sampling* disebut dengan *picture elements* (piksel).

Karakteristik video digital ditentukan oleh beberapa faktor yaitu resolusi (*resolution*) atau dimensi *frame* (*frame dimension*), kedalaman piksel, dan laju *frame* (*frame rate)*. Karakteristik-karakteristik ini akan menentukan tawar-menawar antara kualitas video dan jumlah bit yang dibutuhkan untuk menyimpan atau mentransmisikannya.

3. *Motion Detection* [2, 5]

Deteksi gerak adalah proses untuk mengkonfirmasi perubahan posisi dari suatu objek relatif terhadap sekitarnya atau perubahan dalam lingkungan relatif terhadap suatu objek. Deteksi gerak ini dapat dibuat melalui metode mekanik dan elektronik. Gerakan dapat dideteksi dengan memanfaatkan beberapa hal, di antaranya adalah suara (sensor akustik), opasitas (sensor optik dan proses gambar video, dan inframerah), geomagnet (sensor magnetik, magnetometer), refleksi energi transmisi (radar laser inframerah, sensor ultrasonik, dan sensor radar gelombang mikro), induksi elektromagnetik (detektor *inductive-loop*), dan getaran (*triboelectric* dan seismik).

4. *Color Tracking* [5]

Color tracking merupakan salah satu metode deteksi gerak berdasarkan pengamatan optik/gambar. Metode ini bekerja dengan menggunakan nilai dari warna yang dimiliki setiap piksel pada gambar. Metode ini bisa diterapkan dalam pendeteksian gerak melalui *webcam* atau perangkat video lain karena pada dasarnya sebuah video merupakan proses menampilkan *frame* pada selang waktu tertentu. Setelah mendeteksi keberadaan suatu warna pada gambar, selanjutnya dilakukan *marking* pada seluruh piksel yang memiliki warna tersebut. Hasil dari *marking* tersebut menjadi sebuah data yang akan digunakan pada proses selanjutnya.

5. *Java Media Framework* (JMF) [1]

Java Media Framework (JMF) menyediakan sebuah kesatuan arsitektur untuk mengatur perolehan, pemrosesan, dan pengiriman data media berbasis waktu. JMF didesain untuk mendukung kebanyakan standar tipe media, seperti AIFF, AU, AVI, GSM, MIDI, MPEG, QuickTime, TMF, dan WAV. *Data source* dan *player* adalah bagian dari integral API tingkat tinggi dari JMF untuk mengatur *capture*, presentasi, dan pemrosesan *time-based* media. JMF menyediakan *developer* Java dengan API yang mudah dipakai untuk mendukung *time-based* media ke dalam program Java, selama mempertahankan fleksibilitas dan ekstensibilitas yang dibutuhkan untuk mendukung aplikasi media tingkat tinggi.

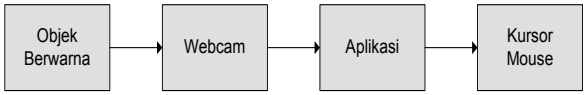
JMF API secara utama terdiri dari antarmuka yang mendefinisikan tindakan dan interaksi objek yang digunakan untuk merekam, memproses, dan menampilkan media berbasis waktu. Implementasi dari antarmuka ini beroperasi dalam struktur kerangka kerja. Dengan menggunakan objek perantara yang dinamakan *manager*, JMF membuatnya mudah untuk mengintegrasikan implemetasi baru dari antarmuka kunci yang dapat digunakan dengan tidak terikat *class* lain yang telah ada. JMF menggunakan empat jenis *manager*, yaitu:

a. *Manager*, untuk menangani konstruksi *player*, *processors*, *data sources*, dan *data sink*.
b. *Package manager*, untuk mengelola daftar paket yang mengandung class JMF, seperti custom *player*, *processor*, *data sources*, dan *Data Sinks*.
c. *Capture device manager*, untuk mengelola daftar peralatan perekam yang tersedia.
d. *Plug-in manager*, untuk mengelola daftar *plug-in* komponen pemrosesan JMF yang tersedia, seperti *multiplexer*, *demultiplexer*, *codecs*, *effects*, dan *renders*.

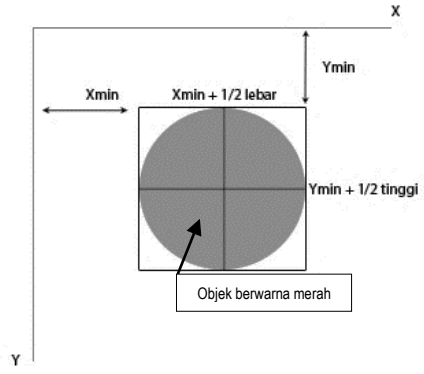
6. Deskripsi Sistem

Secara umum, cara kerja dari sistem yang dirancang (Gambar 1) dapat dijelaskan sebagai berikut:

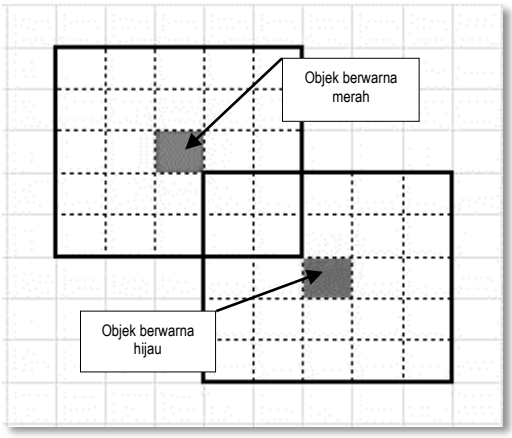
a. Objek berwarna, setiap pergerakan atau perpindahan posisinya akan menjadi masukan dari aplikasi.
b. *Webcam*, perangkat keras ini akan menjadi media pengamatan terhadap pergerakan objek berwarna.
c. Aplikasi, hasil *capture* dari *webcam* akan diolah melalui beberapa proses yaitu *color tracking*, *marking*, dan kalibrasi dalam aplikasi ini.
d. Kursor *mouse* akan melakukan aksi yang telah ditentukan oleh aplikasi berdasarkan dari hasil pengolahan *frame capture webcam*.



Gambar 1. Diagram Blok Aplikasi



Gambar 2. Titik Tengah dari Objek Berwarna



Gambar 3. Kondisi Saat Area Sensor Bersinggungan

7. *Color Tracking*

Dari hasil rekam *webcam*, selanjutnya dilakukan analisis tiap piksel untuk mendapatkan nilai intensitas RGB penyusun warnanya. Sebelumnya telah ditentukan nilai intensitas warna yang menjadi batas untuk memfilter piksel (*threshold*). Piksel dengan intensitas warna yang sesuai dengan filter akan digunakan untuk mendapatkan koordinat dari objek berwarna tersebut.

7.1. Deteksi Koordinat Objek Warna

Koordinat objek berwarna yang diperlukan dalam aplikasi ini adalah titik tengah dari objek (Gambar 2). Titik tengah tersebut diperoleh dari perhitungan tinggi dan lebar objek warna dalam *frame* hasil *capture*. Kemudian dilakukan proses pembandingan antar koordinat piksel (x,y) yang telah lolos dari proses filter sehingga didapatkan nilai X_{min}, X_{max}, Y_{min}, dan Y_{max}. Nilai-nilai tersebut yang kemudian digunakan sebagai koordinat untuk melakukan penandaan (*marking*) pada area objek berwarna.

7.2. Kalibrasi

Karena perbedaan resolusi antara *frame* dan monitor (*display*), maka dibutuhkan sebuah skala kalibrasi. Sehingga, koordinat titik tengah objek pada *frame* berada pada koordinat dalam monitor secara proporsional. Kalibrasi ini menjadikan titik tengah dari *frame* dan titik tengah dari monitor sebagai acuan. Selanjutnya nilai dari $X_{monitor}$ dan $Y_{monitor}$ merupakan koordinat piksel pada monitor.

```
Xmonitor= tengah lebar monitor –
          ((tengah lebar frame –
          Xframe)*skala)

Ymonitor= tengah tinggi monitor –
          ((tengah tinggi frame –
          Yframe)*skala)
```

7.3. Aksi Kursor *Mouse*

Koordinat titik objek berwarna merah yang telah dikalibrasi kedalam koordinat monitor (*display*) akan menjadi posisi kursor *mouse* berada. Koordinat titik objek hijau yang telah dikalibrasi juga berada pada bidang *display*, tetapi bersifat tidak terlihat.

Interaksi kedua titik ini merupakan pemicu sebuah aksi dari kursor *mouse*. Interaksi yang memungkinkan antara kedua titik tersebut, antara lain:

a. saling bergerak bebas tidak berdekatan,
b. kedua objek tersebut bersinggungan, dan
c. kedua objek tersebut bergerak beriringan dalam kondisi bersinggungan.

Kedua titik dikatakan bersinggungan bila berada pada koordinat yang hampir sama, hal ini dikarenakan tidak memungkinkannya kedua titik memiliki koordinat yang sama (Gambar 3). Cara untuk mengatasi masalah ini adalah dengan menambahkan area sensor pada masing-masing koordinat titik. Berikut adalah aksi dari kursor *mouse* yang terjadi berdasarkan interaksi dari kedua titik objek:

a. Kursor berpindah koordinat; aksi ini terjadi pada saat interaksi dari kedua objek warna saling bergerak bebas dan tidak ada area sensor yang bersinggungan.
b. Klik; kursor *mouse* akan melakukan aksi klik kiri apabila area sensor kedua objek warna bersinggungan.
c. Menahan klik (*drag*/*block*); jika pada saat terjadi aksi klik kiri lalu kedua objek bergerak beriringan dengan area sensor tetap bersinggungan, maka kondisi ini sama dengan aksi *drag* atau *block* yang biasa digunakan dengan menggunakan *mouse*.

8. Implementasi dan Pengujian

Spesifikasi perangkat keras dan perangkat lunak minimum yang bisa digunakan dalam implementasi aplikasi ini, antara lain:

a. Perangkat keras:
   1) Prosesor: AMD Athlon™ 64 X2 @ 2.60GHz (2 CPUs)
   2) RAM: 1 GB DDR2 RAM
   3) *Webcam* VGA: VGA Sensor (resolusi 640×480 piksel)
b. Perangkat lunak:
   1) Sistem operasi: Windows 7 Professional 32-bit
   2) *Software* pemrogram: Java SE Development Kit 7
   3) *Software* pendukung: Java Media Framework (JMF) 2.1.1e

Dalam implementasi, objek berwarna yang digunakan adalah cahaya dari rangkaian LED. Cahaya dari LED dipilih karena memiliki beberapa kelebihan, salah satunya adalah tidak terpengaruh oleh intensitas cahaya ruangan sehingga bisa

digunakan pada ruangan dengan intensitas cahaya yang kurang (Gambar 4).

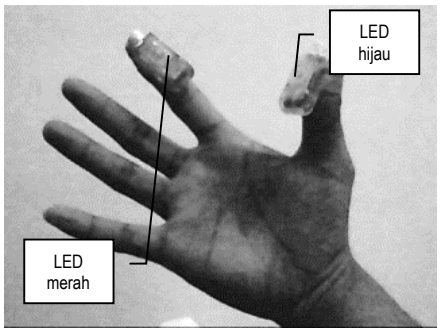
8.1. Pengujian Resolusi, Jarak, dan Skala Kalibrasi

Pada tahap ini dilakukan percobaan tentang pengaruh besarnya skala kalibrasi terhadap kehalusan pergerakan kursor. Pada skala 2,5, meskipun mulai terlihat lompatan pada pergerakan kursor tetapi masih dalam ukuran kecil. Sedangkan untuk skala di atas 2,5, lompatan kursor bisa dilihat dengan jelas dan sudah dapat dianggap mengganggu karena lompatan yang terjadi cukup besar.

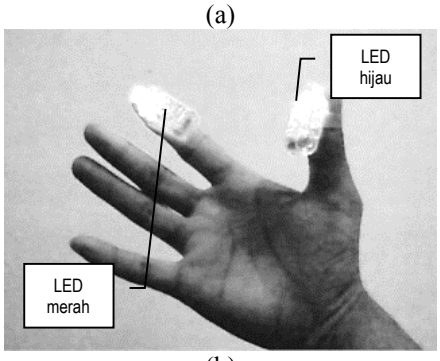
Jadi, berdasarkan skala yang telah dipilih, maka jarak maksimum dalam penggunaan aplikasi ini adalah 2 meter. Karena bisa dilihat bahwa dengan skala sebesar 2,5 jarak yang membuat pengguna tidak perlu berjalan untuk menjangkau bagian tepi dari sumbu *x* adalah 2 meter.

8.2. Pengujian *Marking*

Diperlukan *range* nilai RGB dari warna yang akan di-*marking*. Gambar 5 merupakan hasil *marking* yang dilakukan aplikasi terhadap objek warna yang telah di-*sampling* sebelumnya. Dari gambar dapat dilihat bahwa proses *marking* sudah bisa berjalan dengan baik. Dalam skenario ini aplikasi tetap dapat melakukan *marking* terhadap objek dengan jarak 2 meter dari *webcam*. Untuk bisa melakukan *marking* terhadap seluruh permukaan dari objek berwarna, maka perlu mengambil sampel tidak hanya pada titik tetap, tetapi juga pada beberapa bagian objek.

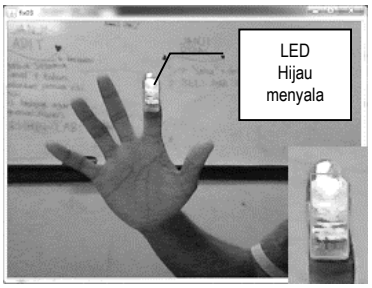


(a)

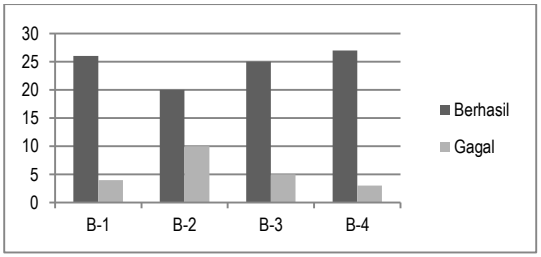


(b)

Gambar 4. (a) Rangkaian LED Mati (b) Rangkaian LED Menyala



Gambar 5. *Marking* terhadap Objek Warna



Gambar 6. Grafik Hasil Pengujian Aksi Klik

Tabel 1. Hasil Pengujian Aksi Menahan Klik

Aksi	Percobaan	Gagal	Berhasil
Kiri ke Kanan	18	2	16
Kanan ke Kiri	18	0	18
Atas ke Bawah	30	3	27
Bawah ke Atas	30	1	29
Total	**96**	**6**	**90**

9. Pengujian Aksi Kursor *Mouse*

Pada skenario ini dilakukan pengujian terhadap tingkat keberhasilan dalam melakukan aksi klik dan menahan klik dengan jarak antara pengguna dengan *webcam* sejauh 2 meter dengan skala kalibrasi 2,5. Sedangkan radius *range* sensor yang digunakan yaitu 20 piksel.

a. Pengujian Klik
   Setiap tombol coba diklik sebanyak 30 kali (Gambar 6). Kegagalan dari aksi klik pada tombol B-1 ada 4, tombol B-2 ada 10, tombol B-3 ada 3, dan tombol B-4 ada 5. Jadi, dari 120 kali percobaan, terjadi kegagalan sebanyak 22 kali. Maka, tingkat keberhasilan dari aksi klik yang dilakukan sebesar 81,66%.
b. Pengujian Menahan Klik
   Dalam pengujian kali ini akan dilakukan aksi menahan klik terhadap ikon yang berada pada layar *desktop*. Ikon tersebut akan dipindahkan dengan cara melakukan aksi *drag* dari posisi awal di bagian kiri kebagian kanan dari desktop. Selanjutnya, dilakukan percobaan sebaliknya, yaitu dipindahkan dari kanan ke kiri dan dari atas ke bawah (Tabel 1). Kegagalan yang terjadi dalam pengujian aksi klik dan menahan klik sebagian besar diakibatkan oleh bergeraknya jari pengguna saat akan melakukan klik. Hal ini sebenarnya wajar, karena secara natural bila ingin melakukan gerakan mencubit maka jari telunjuk dan ibu jari akan bergerak saling mendekat, tetapi dalam penggunaan aplikasi ini, hal tersebut mempengaruhi tingkat akurasi.

10. Penggunaan *Resource*

Dalam pengujian ini spesifikasi perangkat keras dan perangkat lunak yang digunakan adalah sebagai berikut:

a. Perangkat keras:
   1) Prosesor: Intel Core i5-2430M @ 2.40GHz
   2) RAM: 2 GB DDR3 RAM
   3) *Webcam*: Lenovo EasyCamera 1,3 MP
b. Perangkat lunak:
   1) Sistem operasi: Windows 7 Professional 32-bit
   2) *Software* pemrogram: Java SE Development Kit 7
   3) *Software* pendukung: Java Media Framework (JMF) 2.1.1e
   4) *Software* tambahan: NetBeans 7.0.1

Berdasarkan pengujian dengan perangkat-perangkat tersebut, aplikasi ini membutuhkan utilitas CPU yang cukup tinggi yaitu rata-rata sebesar 54,9%. Besarnya memori *heap size* tertinggi yang disediakan oleh JMF adalah 52,227 MB Sedangkan untuk penggunaan *memory* dengan *heap size* yang disediakan JMF terhadap aplikasi rata-rata sebesar 48,554 MB bisa dianggap bahwa aplikasi tidak membutuhkan *memory* yang besar.

11. Kesimpulan

Aplikasi yang dibangun sudah bisa menggantikan perangkat *mouse* sebagai *pointing device* dalam mengendalikan aksi kursor dalam berpindah posisi, klik, dan menahan klik. Aplikasi ini bisa menjadi media interaksi berjarak dengan komputer karena dapat diimplementasikan dalam penggunaan program komputer yang lain. Resolusi untuk penggunaan aplikasi ini adalah 800×600 piksel karena dengan resolusi tersebut aplikasi ini dapat memiliki jarak maksimum 2 meter dengan skala kalibrasi yang optimal (2,5) dan tingkat akurasi yang cukup baik. Diharapkan, pengembangan selanjutnya dari aplikasi ini adalah menambahkan pengontrol untuk aksi klik (klik kanan *mouse*) sehingga bisa menggantikan fungsi perangkat *mouse* secara keseluruhan.

Daftar Pustaka

[1] Erdem, A., E. Yardimci, Y. Atalay, dan V. Cetin, "*Computer Vision Based Mouse, A. E. Acoustics, Speech, and Signal Processing"*, Proceedings ICASS, IEEE International Conference, 2002.

[2] Nam Yanghee dan KwangYun Wohn, "*Recognition of SpaceTime Hand-Gestures using Hidden Markov Model*", Proceedings of ACM VRST'96, pp. 51-58, Juli 1996.

[3] Nariman, Dahlan, dkk., "*Multi-scale Virtual Environment with Gesture Interface*," Proceedings of VSMM'98, pp. 318-323, 1998.

[4] Segen, J. dan S. Kumar, "*Shadow Gestures: 3D Hand Pose Estimation Using a Single Camera*", Proceedings of IEEE Conference on Computer, 1999.

[5] Y. Sato, Y. Kobayashi, dan H. Koike., "*Fast Tracking of Hands and Fingertips in Infrared Images for Augmented Desk Interface*", Proceedings of IEEE, International Conference on Automatic Face and Gesture Recognition (FG), pp. 462-467, 2000.