Perancangan Awal Antarmuka Gesture Tangan Berbasis Visual

Rudy Hartanto[1], Marcus Nurtiantoro Aji[2]

***Abstract*—Human computer interaction has a long history to become more intuitive. For human being, gesture of different kind is one of the most intuitive and common communication. However, vision-based hand gesture recognition is a challenging problem, which are involved complex computation, due to high degree of freedom in humand hand.**

In this paper we use hand gesture captured by web-cam instead of mice, for natural and intuitive human-computer interaction. Skin detections method is used to create a segmented hand image and to differentiate with the background. A contours and convex hull algorithm is used to recognize hand area as well as the number of finger tip of hand gesture image to be mapped with button. And for moving detection of hand gesture, the Lucas-Kanade pyramidal algorithm were used.

The result shows that this system can operate well so we can interact with computer using our hand gesture instead using a mice.

***Intisari*—Interaksi antara komputer dan manusia telah mengalami sejarah yang panjang untuk dapat menjadi lebih intuitif. Bagi manusia, perubahan sikap merupakan salah satu bentuk komunikasi yang paling umum dan intuitif. Namun, pengenalan gerak tangan merupakan masalah yang cukup menantang, dimana membutuhkan perhitungan secara kompleks, akibat dari tingginya derajat kebabasan dari tangan manusia.**

Dalam paper ini digunakan penangkapan gerakan tangan menggunakan kamera web, untuk hubungan antara manusia dan komputer yang lebih alami dan intuitif. Metode deteksi kulit digunakan untuk membuat gambar tangan tersegmentasi dan dapat dibedakan dengan latar belakang. Algoritma hull digunakan untuk mengenali daerah tangan serta jumlah ujung jari gambar isyarat tangan untuk dipetakan dengan tombol. Lucas-Kanade piramidal algoritma selanjutnya digunakan untuk mendeteksi gerakan tangan yang bergerak.

Hasil penelitian menunjukkan bahwa sistem ini dapat beroperasi dengan baik sehingga kita dapat berinteraksi dengan komputer dengan menggunakan gerakan tangan, tanpa perlu menggunakan tetikus.

***Kata Kunci*— hand gesture, human computer interaction, contours, convex hull, Lukas-Kanade Algorithm.**

I. PENDAHULUAN

Perkembangan antarmuka pengguna telah meningkat dengan pesat terutama pada perangkat konsumer. Penggunaan antarmuka sentuh (*touch screen*) telah menjadi umum digunakan pada perangkat seperti Smart Phone, dan antarmuka kontrol pada kamera digital. Pada perkembangannya cenderung mengarah pada penggunaan gesture tubuh manusia, terutama tangan sebagai antarmuka utama yang lebih alami dan intuitif, seperti antarmuka *multi-touch* pada Microsoft Surface dan sensor gerakan pada Wii Remote. Namun sampai saat ini antarmuka pengguna sebagian besar masih terbatas pada gerakan di atas permukaan 2 dimensi, seperti penggunaan mouse dan layar sentuh.

Paper ini membahas study tentang rancangan awal sistem antarmuka yang dapat menangkap gerakan tangan pengguna sebagai ganti dari antarmuka dengan menggunakan mouse. Sistem ini akan melacak posisi 3 dimensi dan konfigurasi tangan. Dimungkinkan untuk menangkap tidak hanya gerakan tangan secara keseluruhan tetapi juga posisi jari-jari tangan. Dengan demikian sistem ini memungkinkan pengguna untuk dapat berinteraksi menggunakan gerakan tangan secara penuh dalam ruang 3 dimensi sebagai pengganti permukaan 2 dimensi seperti yang selama ini ada.

Gerakan tangan pengguna akan ditangkap oleh webcam. Gambar jari tangan tersebut akan diidentifikasi dengan menggunakan metoda template matching dan selanjutnya dapat dilacak posisi gerakan dan konfigurasi jari untuk menentukan perintah yang harus dijalankan.

II. DASAR TEORI

A. Tinjauan Pustaka

Beberapa penelitian yang terkait dengan penggunaan gesture tangan dalam berinteraksi dengan sistem (komputer) sebagai ganti mouse telah dilakukan. Penelitian tersebut meliputi penggunaan kamera sebagai antarmuka pengguna [1] , penggunaan antarmuka berbasis glove [2], penggunaan web cam sebagai pelacak visual [3], penggunaan gesture tangan sebagai antarmuka [4]. Sedangkan pengembangan sistem pelacakan jari untuk pengetikan di udara oleh [5], selanjutnya pengembangan interpretasi visual dari gestur tangan [6], dan review tentang interaksi manusia komputer dengan menggunakan gestur tangan [7].

Dan Maynes, Terry Winograd, dan Takeo Igarashi mengembangkan suatu tool untuk merancang suatu interaksi berbasis kamera yang disebut dengan Eyepatch. Sasaran penelitian ini adalah untuk menyederhanakan proses pengembangan aplikasi visual. Dengan menggunakan Eyepatch sebagai tool yang memungkinkan pemrogram pemula untuk mengekstrak data-data yang berguna dari *life video* dan melakukan streaming data ke tool prototipe cepat seperti Adobe Flash atau Microsoft Visual Basic.

José P. Molina memperkenalkan suatu metodologi untuk memfasilitasi proses pengembangan antarmuka pengguna 3-D, yang dinamakan TRES-D. Sampai saat ini hampir semua penelitian tentang antarmuka pengguna 3-D fokus utamanya pada teknologi dan bagaimana penggunaannya, dan perhatian utamanya adalah pada pemilihan perangkat keras, perangkat lunak, dan teknik

[1] *Jurusan Teknik Elektro dan Teknologi Informasi Fakultas Teknik Universitas Gadjah Mada, Jln. Grafika 2 Yogyakarta 55281 INDONESIA (e-mail: rudy@mti.ugm.ac.id)*

[2] *Jurusan Teknik Elektro dan Teknologi Informasi Fakultas Teknik Universitas Gadjah Mada, Jln. Grafika 2 Yogyakarta 55281 INDONESIA (e-mail: mna@mti.ugm.ac.id)*

interaksinya. Proses pengembangannya sendiri sebenarnya merupakan bagian yang penting, namun umumnya terabaikan atau dokumentasinya sangat kurang.

Hasil penelitian yang dilakukan memberi gambaran bagaimana penerapan metode TRES-D dalam pengembangan tiga antarmuka berbasis glove yang berbeda. Tidak hanya memperlihatkan keuntungan yang diperoleh dengan penggunaan perangkat tersebut, tetapi juga keuntungan penggunaan kerangka-kerja metodologi TRES-D.

Giancarlo Iannizzotto melakukan penelitian tentang pendekatan antarmuka pengguna berbasis pelacakan visual (*visual tracking*) sebagai ganti dari perangkat antarmuka keyboard dan mouse. Antarmuka pengguna berbasis pelacakan visual cukup menjanjikan, karena tidak memerlukan tambahan perangkat lain seperti gloves, tapi dapat diimplementasikan dengan menggunakan kamera web (*webcam*). Masalah yang dihadapi pada penggunaan webcam adalah kondisi pencahayaan yang variatif, dan tingkat kompleksitas komputasi yang tinggi dengan menggunakan algoritma yang ada saat ini.

Proyek penelitian ini berhasil mengembangkan suatu perangkat waktu nyata berbasis visual yang dapat menggantikan keyboard dan mouse. Perangkat ini mampu beroperasi dalam kondisi pencahayaan yang bervariasi, dengan menggunakan webcam murah yang berjalan pada PC.

William T. Freeman dan Michael Rot mengembangkan metoda untuk mengenali gesture tangan berdasarkan pada teknik pengenalan pola dengan menggunakan orentiasi lokal dari histogram. Histogram orientasi digunakan sebagai vektor fitur untuk klasifikasi dan interpolasi gesture. Metode ini cukup sederhana dan komputasinya cepat dan mempunyai keandalan terhadap perubahan pencahayaan.

Pada penelitian ini diimplementasikan versi waktu nyata yang dapat membedakan pustaka dari 10 gesture tangan yang berbeda. Seluruh komputasi dilakukan pada workstation, dan perangkat keras khusus hanya digunakan untuk mendigitalisasi citra. Pengguna dapat mengoperasikan komputer atau bermain game dengan menggunakan kontrol gesture tangan.

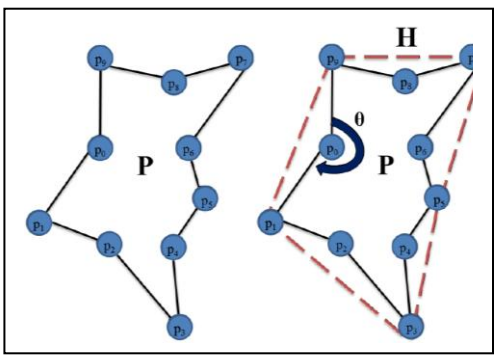
Kazuhiro Terajima, Takashi Komuro, dan Masatoshi Ishikawa berhasil melakukan penelitian tentang sistem pelacak jari cepat untuk antarmuka pengetikan di udara. Mereka berhasil mengembangakan suatu sistem yang dapat melacak gerakan tangan dan jari di ruang 3-dimensi dari satu kamera dengan pesat frame tinggi, dan dapat mengenali gerakan tangan saat menulis di udara. Menggunakan metoda berbasis *template-matching* dengan menggunakan metoda tekstur tangan untuk mengurangi pengaruh latar belakang tanpa perlu membuat tanda untuk pelacakan.

Dengan menggunakan kamera kecepatan tinggi memungkinkan pengenalan gerakan penulisan yang cepat yang sulit dilacak dengan menggunakan kamera standard. Agar dapat merealisasikan pengenalan secara waktu nyata, maka dikembangkan perangkat keras yang dapat memparalelisasi dan percepatan pengolahan citra. Sistem ini berhasil mengenali gerakan penulisan tangan sampai kecepatan 138 fps (*frame per second*) dan dengan latensi 29 mili detik.

Vladimir I. Pavlovic, Rajeev Sharma, dan Thomas S. Huang melakukan penelitian tentang interpretasi visual dari gesture tangan untuk interaksi manusia komputer. secara khusus interpretasi visual dari gesture tangan dapat membantu dalam pencapaian interaksi manusia komputer yang lebih mudah dan alami. Pada penelitian ini dibahas model gesture tangan yang mempunyai sifat spasial dan dinamis yang dapat mengakomodasi seluruh tipe alami dari gestur tangan manusia. Dua kelas model digunakan untuk menginterpretasi gesture tangan untuk interaksi. Pertama dengan menggunakan model tangan manusia 3-D, sementara yang kedua berdasarkan pada penampakan tangan manusia pada citra. Pengamatan dilakukan terhadap parameter model dan analisis fitur serta pengaruhnya pada interpretasi gestur tangan. Hasil penelitian penunjukkan bahwa model tangan 3D menawarkan pemodelan gesture tangan yang lengkap, namun mempunyai kekurangan dalam hal kesederhanaan dan efisiensi komputasi yang sangat diinginkan. Penelitian ini menyarankan beberapa metoda yang dapat meningkatkan efektifitas antarmuka gestur untuk interaksi manusia dan komputer. Integrasi dari gestur tangan dengan model komunikasi alami yang lain berpotensi untuk menjawab masalah ini.

B. Dasar Teori

1) Convex hull: Convex hull merupakan persoalan klasik dalam komputasi geometri. *Convex hull* digambarkan secara sederhana dalam sebuah bidang sebagai pencarian subset dari himpunan titik pada bidang tersebut, sehingga jika titik-titik tersebut dijadikan poligon maka akan membentuk poligon yang *konveks* [8]. Suatu poligon dikatakan *konveks* jika garis yang menghubungkan antar kedua titik dalam poligon tersebut tidak memotong garis batas dari poligon. *Convex hull* suatu obyek P didefinisikan sebagai area poligon *convex* terkecil yang melingkupi P. Oleh karena itu, untuk suatu himpunan titik $N \{p_0, p_1, p_2, \ldots, p_N\} \epsilon P$, maka dapat dinyatakan bahwa *hull H* dapat disusun dengan *M* titik dari himpunan *N* untuk membuat suatu area *konveks* poligon minimum. Dari Gbr. 1 dapat dinyatakan bahwa *Convex hull* dibuat dengan mengambil sudut interior θ, dari tiga titik yang bersebelahan $\{p_1, p_0, p_9\}$. *Jika* $\theta > \pi$ maka p_0 dianggap sebagai titik refleks dan p_0 bukan anggota *M*. Himpunan akhir *H* adalah $\{p_1, p_9, p_7, p_5, p_3\}$ [9].



Gbr. 1 Generalisasi logika *Convex hull*

Pencarian *convex hull* dari sebuah himpunan titik Q (CH(Q)) yaitu dengan mencari sebuah *convex set* terkecil yang memuat seluruh titik pada Q. *Convex hull* dari

sebuah himpunan titik Q (CH(Q)) pada n dimensi adalah seluruh irisan dari semua *convex set* yang mengandung Q. Sehingga untuk N buah titik {p1, p2, ..., pN} ∈ P, *convex hull* merupakan himpunan *convex combination* yang dinyatakan dengan:

$$CH(Q) \equiv \left\{ \sum_{j=1}^{N} \lambda_j p_j ; \lambda_j \geq 0; \sum_{j=1}^{N} \lambda_j = 1 \right\} \quad (1)$$

3) Convexity defect Convexity defect: adalah sebuah fitur dalam *OpenCV* yang berfungsi untuk menemukan *defect* antara *convex hul*l yang terbentuk dengan kontur dari poligon. *Defect* tersebut berguna untuk menemukan *feature* pada sebuah poligon, salah satunya yaitu untuk mendeteksi jari tangan manusia [8].

Untuk lebih jelas mengenai metode *convexity defect* ini dapat dilihat pada Gbr. 2.

Gbr. 2 *Convexity defect* sebuah poligon [10]

Dari Gbr. 2 di atas, terlihat *convex hull* digambarkan dengan garis merah yang menyelubungi poligon dengan garis konturnya yang berwarna hitam. Simbol "s" dan "e" menunjukkan "*start point*" dan "*end point*" dari *convexity defect* tersebut. Sedangkan simbol "d" melambangkan "*depth point*", yaitu titik kontur yang terletak antara "s" dan "e" yang merupakan titik terjauh antara kontur dengan garis *convex hull* yang dilambangkan dengan "se". Simbol "h" yaitu "*depth*" atau kedalaman dari *defect* yang merupakan jarak dari "d" hingga garis "se" [10]

Start point, *end point*, *depth point*, dan *depth*, keempat elemen tersebut yang akan digunakan untuk dapat menemukan *feature* dari sebuah poligon sehingga dapat diolah lebih lanjut.

4) Segmentasi warna kulit (skin detection): Segmentasi warna kulit banyak digunakan untuk aplikasi pengenalan wajah, deteksi badan, ataupun anggota badan. *Skin detection* ini bertujuan untuk dapat mendeteksi warna kulit dari setiap citra yang ditangkap oleh kamera. Citra yang diperoleh umumnya memiliki format RGB. Untuk memudahkan pendeteksian warna kulit, format RGB ini diubah ke format YcrCb untuk memisahkan intensitas Y dengan chromacity yang dinyatakan dalam dua variabel Cr dan Cb. Dalam memodelkan warna kulit hanya informasi Cr dan Cb yang dipakai, sehingga pengaruh perubahan intensitas dapat dihilangkan. Pada daerah saturasi dari cahaya yang tertangkap kamera, harga Cr dan Cb sangat stabil, sehingga nilai Cr dan Cb merupakan informasi handal untuk proses klasifikasi warna. Konversi dari RGB ke YCrCb ini menggunakan rumus :

$$\begin{bmatrix} Y \\ Cb \\ Cr \end{bmatrix} = \begin{bmatrix} 0.29900 & 0.587000 & 0.114000 \\ -0.168736 & -0.331264 & 0.500000 \\ 0.500000 & -0.418688 & -0.081312 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \quad (2)$$

dengan Y adalah luminans (intensitas warna), Cb adalah komponen warna biru, dan Cr adalah komponen warna merah [11].

Setelah dilakukan konversi ke format YCrCb, selanjutnya dilakukan perubahan citra ke dalam bentuk citra biner menggunakan proses *thresholding*. Perubahan ini dilakukan untuk mempermudah proses selanjutnya. Segmentasi warna kulit dilakukan dengan cara menentukan nilai *range* dari Y, Cr, dan Cb. Sehingga untuk setiap piksel dalam citra, jika berada dalam *range* tersebut, maka akan dianggap sebagai warna kulit, sedangkan untuk yang berada di luar *range* tersebut akan dianggap sebagai *background* (latar belakang) [11].

5) Pencarian Titik Pusat Citra Tangan: Proses pencarian titik pusat citra tangan setelah disegementasi dapat dilakukan dengan menggunakan persamaan 3 berikut [11].

$$\bar{x} = \frac{\sum_{i=0}^{k} x_i}{k} \ , \ \frac{\sum_{i=0}^{k} y_i}{k} \quad (3)$$

Dengan x_i dan y_i adalah koordinat x,y dari i piksel pada area tangan, dan k menunjukkan jumlah piksel pada area tersebut. Setelah diketahui lokasi pusat tangan, selanjutnya dapat dihitung area telapak tangan untuk mendapatkan ukuran tangan.

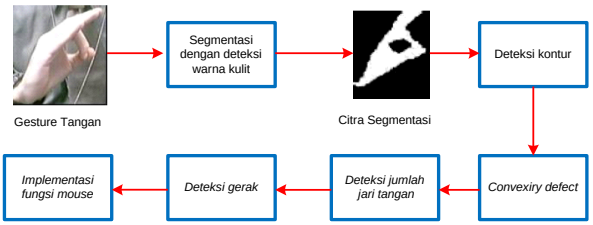
Untuk mendapatkan ukuran tangan dapat dibuat suatu lingkaran dengan menaikkan jari-jari lingkaran dari pusat koordinat sampai lingkaran mencapai piksel hitam pertama. Saat algoritma telah mendapatkan piksel hitam pertama, akan kembali ke nilai jari-jari sebelumnya.

Algoritma ini menggunakan asumsi bahwa saat lingkaran mencapai piksel hitam pertama, setelah menggambar lingkaran yang semakin membesar, maka panjang dari pusat adalah jari-jari dari tangan. Dalam hal ini segementasi citra menjadi bagian yang amat penting, karena untuk menandai batas tangan melalui piksel hitam batas segmentasi.

III. PERANCANGAN SISTEM

Langkah perancangan sistem antarmuka gestur tangan berbasis visual diperlihatkan pada Gbr. 3. Langkah perancangan diawali dengan pengambilan citra gestur tangan dengan menggunakan *web-camera*. Citra hasil tangkapan kamera akan disegmentasi dengan menggunakan *skin detection*. Selanjutnya dilakukan proses pencarian kontur dan penghitungan *convexity defect* yang akan digunakan untuk menghitung jumlah jari tangan yang terlihat. Jumlah jari tangan ini akan digunakan untuk identifikasi perintah sebagai pengganti tombol klik pada *mouse*.

Langkah berikutnya adalah proses penghitungan titik pusat tangan dan pendeteksi gerakan tangan untuk menentukan arah gerakan tangan pengguna yang akan digunakan untuk menggerakkan kursor.



Gbr. 3 Diagram alir sistem antarmuka gestur tangan

A. *Segmentasi*

Proses segmentasi digunakan untuk mendapatkan obyek yang akan diidentifikasi yang dalam hal ini adalah citra tangan. Agar citra tangan dari pengguna yang diambil dari kamera dapat diidentifikasi sebagai suatu perintah untuk menggerakkan kursor. Dalam *computer vision*, segmentasi mengacu pada proses pemisahan citra digital kedalam segmen-segmen. Tujuannya adalah untuk menyederhanakan atau mengubah representasi citra kedalam bentuk yang lebih informatif dan mudah untuk dianalisis.

Segmentasi citra digunakan untuk melokalisasi obyek dan batas obyek (garis, kurva) dalam citra, lebih jauh lagi, merupakan proses pemberian label pada setiap piksel dalam citra sedemikian rupa sehingga piksel dengan label yang sama akan mempunyai karakteristik visual yang sama. Hasil segmentasi citra berupa himpunan segmen yang secara kolektif melingkupi seluruh citra, atau seperangkat kontur yang menjadi batas suatu obyek.

Algoritma lain yang lebih sederhana adalah dengan mendeteksi perbedaan antara warna kulit tangan pengguna dengan latar belakang, yang dikenal dengan algoritma *skin detection*. Metode ini akan mendeteksi warna kulit dari citra yang ditangkap. Citra yang diperoleh dari kamera *webcam* memiliki format RGB. Untuk memodelkan warna kulit, format RGB ini dikonversi ke format YCrCb untuk memisahkan antara intensitas dan *chromacity* (warna) yang masing-masing dinyatakan dengan Cr dan Cb. Konversi ini dilakukan dengan menggunakan persamaan berikut.

$$\begin{aligned} Y &= 0.299*R + 0.587*G + 0.114*B \\ Cr &= (R - Y)*0.713 + 128 \\ Cb &= (B - Y)*0.564 + 128 \end{aligned} \quad (4)$$

Setelah dilakukan konversi ke format YCrCb selanjutnya citra dikonversi kedalam bentuk citra biner dengan proses pengambangan (*tresholding*). Perubahan ini dilakukan untuk mempermudah proses selanjutnya. Deteksi warna kulit dilakukan dengan cara menentukan rentang nilai dari Y, Cr, dan Cb, yang merupakan nilai warna kulit pengguna untuk digunakan sebagai batas ambang dalam menentukan apakah nilai piksel putih (obyek tangan) atau hitam (latar belakang).

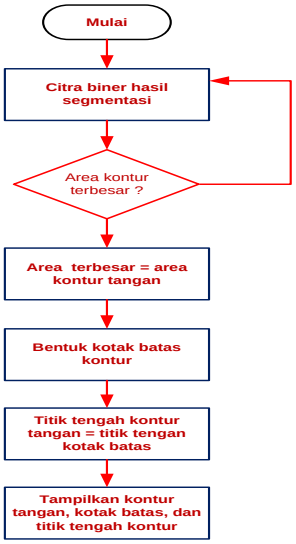
$$w(x,y) = \begin{cases} 255, & jika\ Cr1 < Cr < Cr2\ \ dan\ Cb1 < Cb < Cb2 \\ 0, & Selainnya \end{cases} \quad (5)$$

Dengan Cr1 dan Cr2 serta Cb1 dan Cb2 adalah rentang nilai warna kulit tangan yang dapat diterima. Nilai ini dapat disesuaikan dengan warna kulit pengguna berdasarkan pengamatan.

Untuk setiap piksel dalam citra, jika nilai *Cr* dan *Cb* berada dalam rentang tersebut, maka akan dianggap sebagai warna kulit dan nilainya akan diubah menjadi 255 (warna putih). Sedangkan untuk yang berada di luar rentang itu, akan dianggap sebagai latar belakang dan akan diubah nilainya menjadi 0 (warna hitam). Dengan metode ini, didapatkan citra biner hasil *skin detection* dengan warna putih merepresentasikan tangan pengguna dan warna hitam merepresentasikan *background.*

B. *Deteksi Kontur*

Kontur adalah sebuah runtun nilai piksel yang sama. Kontur ditemukan jika terdapat perbedaan titik-titik yang tinggi dengan tetangganya. Hal ini terjadi didasarkan pada hasil deteksi tepi pada citra biner hasil *skin detection* antara piksel putih (obyek yang memiliki warna menyerupai tangan) dan piksel hitam (obyek-obyek selain tangan atau *background*). Metode ini memiliki kemungkinan *galat* yang cukup tinggi apabila obyek dalam suatu citra memiliki warna yang hampir sama. Oleh karena itu, harus diberi batasan bahwa dalam suatu citra hanya boleh terdapat satu obyek (dalam hal ini tangan pengguna). Gbr. 4 adalah diagram alir proses deteksi kontur.



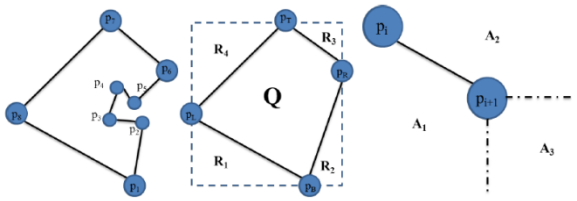
Gbr. 4 Diagram alir deteksi kontur

C. *Convexity defect*

Metode ini bertujuan untuk mendeteksi jari tangan pengguna yang tertangkap oleh kamera. Untuk dapat mengekstrak fitur tangan dengan baik diperlukan dua tahap proses, yaitu :

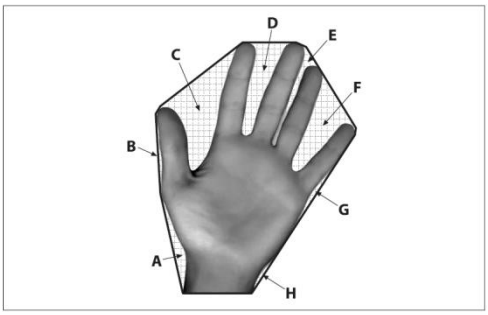
1) *Pembentukan convex hull menggunakan algoritma Sklansky: Convex hull* merupakan himpunan titik-titik yang membentuk sebuah poligon konveks yang melingkupi seluruh himpunan titik tersebut. Algoritma Sklansky yaitu sebuah algoritma linear sekuensial yang digunakan untuk membentuk *convex hull* dari himpunan titik atau sebuah poligon sederhana [9]. Algoritma Sklansky ’82 merupakan *update* dari algoritma Sklansky ’72 sebelumnya yang mempunyai kelebihan sangat bagus untuk poligon berbentuk menyerupai bintang sehingga sesuai dengan obyek penelitian ini yaitu poligon berbentuk tangan. *A*lgoritma Sklansky melakukan pemindaian berlawanan dengan arah jarum jam dengan tujuan untuk menghapus titik verteks p dari poligon P. Sklansky mendefinisikan P sebagai kumpulan titik yang membentuk poligon atau juga sebagai sebuah wilayah tertutup berbentuk poligon.

Gbr. 4 memperlihatkan bentuk segiempat Sklansky yang menggambarkan pembentukan *hull* menggunakan algoritma ini.



Gbr. 5 Segi-empat Sklansky

2) *Menentukan defect dari convex hull yang telah terbentuk: Defect* sendiri dapat didefinisikan sebagai area terkecil dalam *convex hull* yang mengitari tangan. Pencarian *defect* ini berguna untuk menemukan *feature* pada tangan, sehingga nantinya dapat dideteksi jari tangan dan diketahui berapa jumlahnya. Metode *convexity defect* pada tangan ini untuk lebih jelasnya dapat dilihat pada Gbr. 5.



Gbr. 6 *Convexity defect* citra tangan [8]

D. Deteksi Jumlah Jari Tangan

Langkah-langkah algoritma penghitungan jumlah jari tangan adalah sebagai berikut :

- Tentukan koordinat titik tengah (*x*,*y*) dari persegi panjang terkecil (*minimum area rectangle*) dari kontur tangan (selanjutnya disebut *box*).
- Tentukan titik koordinat (x,y) *startpoint* dan *depthpoint* dari *defect* yang ditemukan.
- Tentukan jumlah jari = 0.
- Untuk i = 0 hingga i < jumlah *defect*, lakukan :

Jika (*startpoint.y* < *box.center.y* atau *depthpoint.y* < *box.center.y*) dan
(*startpoint.y* < *depthpoint.y*) dan
(*jarak startpoint dan depthpoint* > *tinggi box* / 6,5)
Maka, jumah jari = jumlah jari + 1.

- Jumlah jari yang terdeteksi ditampilkan.

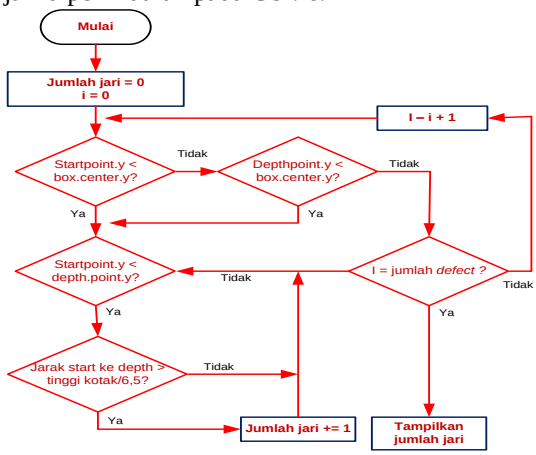
Nilai variabel "jumlah jari" ini akan terus bertambah mengikuti jumlah *defect* yang ditemukan, jika memenuhi ketiga syarat diatas yaitu :

1) *startpoint.y* < *box.center.y* atau *depthpoint.y* < *box.center.y:* Menunjukkan bahwa titik *startpoint* dan *depthpoint* berada di atas titik tengah kontur, atau dengan kata lain telapak tangan menghadap ke atas.

2) *startpoint.y* < *depthpoint.y:* Menunjukkan bahwa titik *startpoint* berada di atas *depthpoint*, atau dengan kata lain berarti jari tersebut menunjuk ke atas, bukan menggenggam.

3) *Jarak startpoint dan depthpoint* > *tinggi box / 6,5:* Menunjukkan bahwa kontur tangan yang terdeteksi diusahakan cukup telapak tangan saja, tidak boleh terlalu panjang hingga lengan pengguna. Hal ini bertujuan agar pengoperasian *mouse* lebih lancar dan efektif.

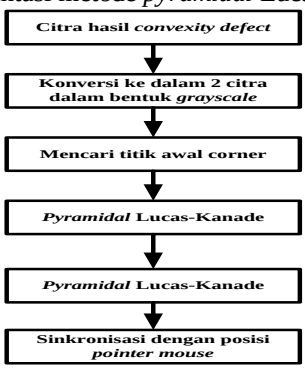
Diagram alir deteksi dan penghitungan jumlah jari diperlihatkan pada Gbr. 6.



Gbr. 7 Diagram alir deteksi dan penghitungan jumlah jari

E. Deteksi Gerak (motion detection)

Deteksi gerak dilakukan dengan *tracking* (pelacakan) terhadap gerakan tangan pengguna. Teknik ini diaplikasikan menggunakan metode *pyramidal* Lucas-Kanade. Gbr. 7 memperlihatkan diaram alir langkah-langkah implementasi metode *pyramidal* Lucas-Kanade.



Gbr. 8 Diagram alir deteksi gerak

F. Implementasi fungsi mouse

Fungsi *mouse* pada program ini dapat diimplementasikan menggunakan fungsi *mouse_event*. Fungsi *mouse_event* ini merupakan API *call* untuk mensintesis aksi *mouse* berupa gerakan *mouse* dan *button click*. Fungsi ini disimpan dalam user32.dll dan kompatibel untuk Windows NT 3.1 dan Windows 95 ke atas. Sintaks dari fungsi ini yaitu sebagai berikut :

```
VOID WINAPI mouse_event(
    __in  DWORD dwFlags,
    __in  DWORD dx,
    __in  DWORD dy,
    __in  DWORD dwData,
    __in  ULONG_PTR dwExtraInfo
);
```

Adapun untuk dapat menjalankan *move*, klik kiri, klik kanan, *double click*, dan *drag*, maka digunakan fungsi mouse_event sebagai berikut :

1) *Move:* yaitu dengan menggunakan *MOUSEEVENTF_MOVE*

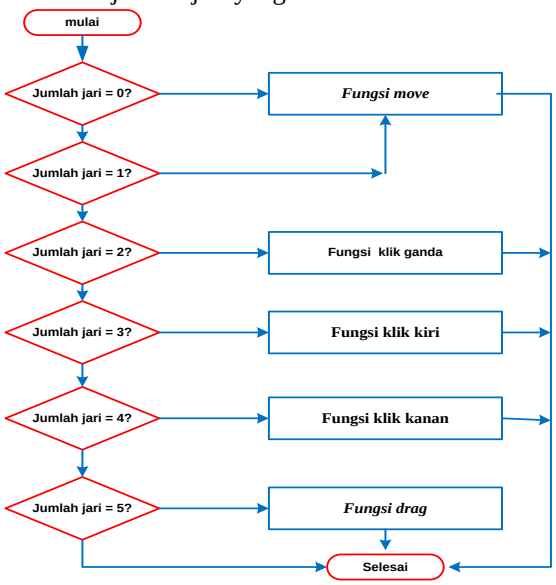
2) *Klik kiri:* yaitu dengan melakukan *MOUSEEVENTF_LEFTDOWN* yang diikuti dengan *MOUSEEVENTF_LEFTUP*.

3) *Klik kanan:* yaitu dengan menggunakan *MOUSEEVENTF_RIGHTDOWN* yang diikuti dengan *MOUSEEVENTF_RIGHTUP*

4) *Double click*, yaitu dengan melakukan klik kiri dua kali secara berurutan.

5) *Drag:* yaitu dengan menggunakan *MOUSEEVENTF_LEFTDOWN* secara terus menerus (kontinu), sambil melakukan *move* ke arah yang diinginkan.

Fungsi-fungsi *mouse* yang dijalankan mengikuti jumlah jari yang terdeteksi. Gbr. 8 memperlihatkan diagram alir fungsi *mouse* yang akan dijalankan berdasarkan jumlah jari yang terdeteksi.



Gbr. 9 Diagram alir implementasi fungsi *mouse*

IV. PEMBAHASAN

Sistem Antarmuka Gesture Tangan Berbasis Visual telah berhasil diimplementasikan dan diuji. Pengujian dilakukan dengan menggunakan sebuah laptop berbasis prosesor Intel I-7 dengan memori 2 GHz dan hard disk 500 Gbyte. Menggunakan sistem operasi Windows 7 dengan perangkat lunak pemrograman Dev C++ versi 4.9.9.2 yang merupakan program *open source*. Kamera yang digunakan adalah webcam Philips SPC900NC 1,3 MP dengan kecepatan 60 fps, dan Logitec QuickCam S 5500, 1,3 MP 30 fps.

A. *Proses segmentasi dengan deteksi warna kulit*

Proses segmentasi bertujuan untuk mendapatkan citra obyek gestur tangan dan memisahkan dengan latar belakang dengan deteksi warna kulit (*skin detection*). Citra *RGB* yang diperoleh dari kamera webcam dikonversi ke format *YCrCb* untuk memisahkan antara intensitas dan chromacity (warna) yang masing-masing dinyatakan dengan *Cr* dan *Cb*. Seperti yang diperlihatkan pada Gbr. 9. Sedangkan citra hasil segmentasi dengan deteksi warna kulit diperlihatkan pada Gbr. 10.

Gbr. 10 Citra asli dan konversi YcrCb

Gbr. 11 Citra hasil Segmentasi

B. *Deteksi Kontur*

Citra biner hasil segmentasi akan digunakan untuk pencarian kontur tangan. Proses ini menggunakan fungsi pada OpenCV yaitu *cvFindContours*. Hasil dari *cvFindContours* ini berupa kontur-kontur yang ditemukan dan jumahnya disimpan pada pointer *&contours*. Setiap kontur dapat diakses melalui kontur menggunakan *h_next* dan *v_next*. Parameter *h_next* untuk mengakses kontur lain di luar kontur sebelumnya, sedangkan *v_next* untuk mengakses kontur di dalam kontur.

Agar pada setiap citra yang ditangkap hanya satu kontur saja yang diproses (misal terdapat obyek lain yang serupa dengan tangan namun lebih kecil), maka perlu ditentukan wilayah dalam kontur mana yang terbesar. Jadi hanya kontur dengan wilayah yang paling besar lah yang akan dibentuk konturnya, sedangkan kontur yang lain tidak perlu. Hasil pencarian kontur ini diperlihatkan pada Gbr. 11.

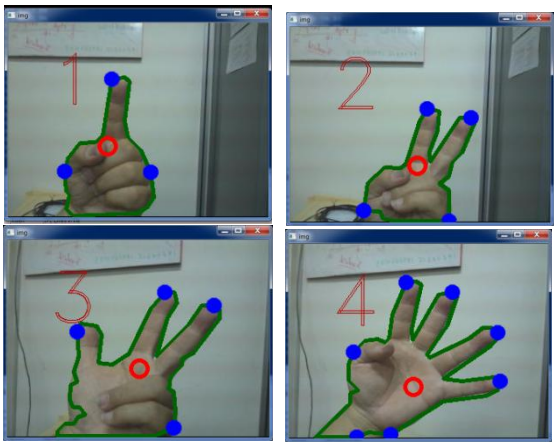
Gbr. 12 Hasil pencarian kontur

C. *Convexity defect dan Deteksi jumlah jari tangan*

Metode ini diawali dengan pembentukan *convex hull*, yang dari *hull* ini akan didapatkan area-area *defect* yang dapat digunakan untuk mendeteksi jari tangan. *Convex hull* yang ditemukan akan ditandai dengan lingkaran kecil berwarna biru seperti diperlihatkan pada Gbr. 12.

Gbr. 13 *Convexity defect* pada kontur jari tangan

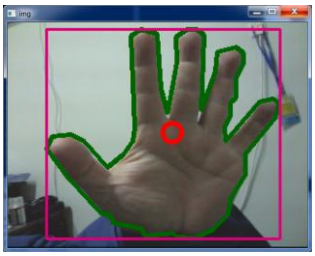
Hasil *convex hull* yang telah diperoleh akan digunakan untuk menentukan jumlah jari tangan. Dalam praktek faktor pencahayaan tidak selalu dapat sempurna yang akan menyebabkan jumlah lingkaran yang terdeteksi tidak hanya di ujung jari saja. Dalam hal ini perlu ditentukan bahwa lingkaran yang digukan untuk penghitungan hanya pada bagian atas saja, dan jumlah maksimumnya adalah 5 sesuai dengan jumlah jari tangan. Selanjutnya akan dihitung dan ditampilkan pada layar. Dari hasil pengamatan seperti yang diperlihatkan pada Gbr. 13 sistem mampu mendeteksi jumlah jari yang diperlihatkan oleh pengguna dari satu jari sampai lima jari.



Gbr. 14 Hasil deteksi dan penghitungan jumlah jari tangan

D. Deteksi Gerak

Deteksi gerakan tangan ini diperlukan agar pointer *mouse* dapat mengikuti gerakan tangan pengguna. Dalam proses ini dibentuk sebuah *kotak batas* yang akan melingkupi kontur tangan, sudut kiri atas kotak ditandai dengan *pt1* dan sudut kanan bawah ditandai dengan *pt2*. Sudut-sudut tersebut nantinya akan digunakan sebagai parameter fungsi *cvRectangle*, untuk menampilkan kotak yang melingkupi kontur tangan, seperti ditunjukkan pada Gbr. 14.



Gbr. 15 Penentuan titik pusat kontur tangan berdasarkan titik pusat kotak

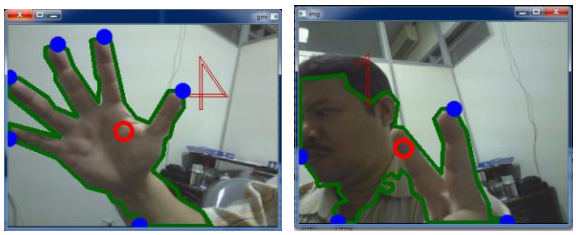
Untuk menemukan titik pusat kontur tangan, dilakukan dengan mencari titik pusat kotak yang melingkupi kontur tangan, seperti yang diperlihatkan pada Gbr. 14. Nantinya, dengan menggunakan *box.center.x* dan *box.center.y* sebagai koordinat titik pusat kontur tangan, gerakan *mouse* dapat diatur agar mengikuti gerakan dari kontur tangan ini. Untuk menandai titik tengah dari kontur tangan, dibentuk sebuah lingkaran kecil berwarna merah seperti diperlihatkan pada Gbr. 14.

E. Pembahasan

Dari hasil pengamatan secara umum program sudah berjalan dengan baik. Pusat gesture tangan yang diwakili oleh lingkaran kecil sudah dapat mengikuti gerakan tangan pengguna. Proses segmentasi dengan deteksi warna kulit cukup baik dalam mendeteksi gestur tangan pengguna dan penghitungan kontur.

Aplikasi ini juga sudah dapat menghitung jumlah jari pengguna yang akan diasosiasikan dengan perintah pada tombol *mouse*. Gerakan gesture tangan pengguna juga sudah dapat diikuti dengan baik oleh sistem, yang dalam pengamatan ditandai dengan titik pusat tangan yang berupa lingkaran kecil berwarna merah.

Namun ada beberapa keterbatasan program, yaitu perubahan pencahayaan lingkungan masih berpengaruh dalam proses segmentasi, khususnya pada proses penghapusan latar belakang. Yang kedua adalah pengguna harus mengenakan jaket yang menutup lengan bawah, sehingga hanya kulit gesture tangan yang terlihat oleh kamera. Karena proses segmentasi menggunakan metoda deteksi warna kulit, sehingga jika (kulit) lengan atau muka pengguna ikut tertangkap kamera, akan menyebabkan proses segmentasi menjadi tidak valid. Yaitu kulit lengan maupun kepala jadi ikut tersegmentasi sebagai obyek tangan,



a) Kulit lengan ikut dalam kontur b) Kulit muka ikut dalam kontur

Gbr. 16 Kesalahan proses penghitungan jari akibat sebagian lengan atau kepala ikut tersegmentasi dan dijadikan kontur

Pada kasus Gbr. 15.a) kulit lengan pengguna ikut terdeteksi dan dilingkupi kontur. Hal ini menyebabkan jumlah *convex hull* menjadi tidak valid dan berakibat pada hasil penghitungan jumlah jari menjadi salah. Demikian pula pada Gbr. 15.b), dengan ikut terdeteksinya kulit muka pengguna, maka selain akan menyebabkan kesalahan dalam penghitungan jari, juga posisi titik pusat tangan juga menjadi salah, dalam hal ini bergeser pada ujung jari tengah. Hal ini akan berakibat pada proses deteksi gerak tangan yang salah juga.

Mengingat beberapa keterbatasan aplikasi seperti di atas, maka direkomendasikan agar pengguna menggunakan baju lengan panjang atau jaket yang berwarna gelap (tidak sama dengan warna kulit), agar yang terdeteksi oleh aplikasi hanya tangan pengguna. Demikian pula disarankan agar pencahayaan ruang cukup terang, sehingga perbedaan warna kulit tangan pengguna dengan latar belakang atau obyek yang lain dapat dibedakan dengan jelas.

V. KESIMPULAN

Dari hasil pengamatan dan pembahasan dapat ditarik kesimpulan sebagai berikut.

A. Kesimpulan

- Program aplikasi yang dibuat secara umum sudah dapat berjalan dengan baik dan dapat menggantikan fungsi mouse dan tombo-tombolnya.
- Algoritma segmentasi dengan deteksi warna kulit cukup baik untuk dapat memisahkan obyek tangan dengan latar belakang.
- Algoritma *convexity* defect dapat mendeteksi jumlah jari pengguna yang terlihat, yang selanjutnya dapat digunakan untuk menentukan atau menggantikan perintah-perintah pada tombol *mouse*.
- Sistem juga dapat mendeteksi dan mengikuti gerakan tangan pengguna dengan baik.

B. Saran

Disamping keberhasilan aplikasi ini untuk menggantikan peran mouse dalam berinteraksi dengan komputer, masih terdapat beberapa keterbatasan yang perlu disempurnakan pada penelitian berikutnya.

- Pengaruh perubahan dan intensitas cahaya lingkungan masih dapat menyebabkan kesalahan dalam proses segmentasi. Sebaiknya diusahakan intensitas cahaya yang cukup terang agar warna kulit tangan pengguna dapat dibedakan dengan latar belakang. Cara yang lain adalah dengan memperbaiki algoritma segmentasi yang digunakan agar lebih kebal terhadap perubahan pencahayaan lingkungan.
- Pengguna masih perlu menggunakan baju lengan panjang atau jaket dengan warna gelap, agar kulit lengan tidak ikut tertangkap kamera. Untuk mengatasinya perlu dicoba dengan menggunakan algoritma lain seperti *template matching* untuk mendeteksi pola gestur tangan, sehingga nantinya pengguna tidak perlu khawatir jika lengannya atau mukanya ikut tertangkap kamera akan mengakibatkan program menjadi tidak berfungsi dengan baik.

REFERENSI

[1] Maynes, D., Aminzade, Terry Winograd, Takeo Igarashi." Eyepatch: Prototyping Camera-based Interaction through Examples". Symposium on User Interface Software and Technology, Newport, Rhode Island, USA., ACM, 2007..

[2] Molina, José P., Arturo S. García, Diego Martinez, Francisco J. Manjavacas, Victor Blasco, Victor López & Pascual González, "The Development of Glove-based Interfaces with the TRES-D Methodology", VRST '06 Proceedings of the ACM symposium on Virtual reality software and technology ACM New York, NY, USA, 2006.

[3] Iannizzotto, Giancarlo, Massimo Villari, Lorenzo Vita, "Hand tracking for human-computer interaction with Graylevel VisualGlove: Turning back to the simple way", PUI '01 Proceedings of the 2001 workshop on Perceptive user interfaces, ACM New York, NY, USA, 2001.

[4] Freeman, William T., Michal Roth, "Orientation Histograms for Hand Gesture Recognition", Mitsubishi Electric Research Laboratories Cambridge Research Center, TR-94-03a December 1994.

[5] Terajima, Kazuhiro, Takashi Komuro, Masatoshi Ishikawa, "Fast Finger Tracking System for In-air Typing Interface", CHI 2009, April 4 – 9, Boston, MA, USA, 2009.

[6] Pavlovic, Vladimir I., Rajeev Sharma, and Thomas S. Huang, "Visual interpretation of hand gestures for human-computer interaction: A review," IEEE Trans. on PAMI, July 1997.

[7] Ic, Rajeev Sharma, and Thomas S. Huang, "Visual interpretation of hand gestures for human-computer interaction: A review," IEEE Trans. on PAMI, July 1997.

[8] Bradski, Gary and Adrian Kaehler, "Learning OpenCV", O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472, 2008

[9] Youssef, Menatoallah M., "Hull Convexity Defect Features For Human Action Recognition", Dissertation, The School of Engineering of the University of Dayton, 2011.

[10] Intel, "Open Source Computer Vision Library", Intel Corporation 2000-200

[11] Hojoon Park, "A Method for Controlling Mouse Movement using Real- Time Camera", Master Theses and Project Report, Brown Computer Science, 2010.