IJEIS, Vol.5, No.2, October 2015, pp. 111~122
ISSN: 2088-3714

Pengolahan Citra Digital untuk *Keyboard* Virtual Sebagai Antarmuka pada Aplikasi Berbasis *Web*

Raksaka Indra Alhaqq*[1], Agus Harjoko[2]
[1]Program Studi Elektronika dan Instrumentasi, FMIPA UGM, Yogyakarta
[2]Jurusan Ilmu Komputer dan Elektronika, FMIPA UGM, Yogyakarta
e-mail: ***[1]raksakaindra@gmail.com**, [2]aharjoko@ugm.ac.id

Abstrak

Sejak pertama kali komputer ditemukan, keyboard selalu menjadi alat utama yang menjadi penghubung interaksi antara manusia dan komputer. Saat ini banyak komputer yang menerapkan teknologi pengolahan citra untuk menjadikannya perantara interaksi antara komputer dan manusia.

Dalam penelitian ini, penulis mencoba untuk menerapkan teknologi pengolahan citra yang digunakan untuk keyboard virtual pada aplikasi web. Digunakan webcam untuk menangkap citra ujung jari telunjuk. Hasil capture citra akan dikirimkan ke server localhost untuk diproses dengan image processing. Untuk mendeteksi ujung jari telunjuk, digunakan metode Haar Cascade Classifier. Proses pendeteksian tersebut menghasilkan koordinat yang akan dikirimkan ke aplikasi web yang selanjutnya dijadikan acuan untuk menentukan posisi tombol pada keyboard virtual. Sehingga keyboard virtual akan menampilan karakter sesuai dengan yang ditunjuk oleh ujung jari telunjuk.

Dari hasil pengujian yang dilakukan, jarak optimal ujung jari telunjuk dengan webcam adalah 20 – 35 cm. Derajat kemiringan ujung jari telunjuk untuk dapat terdeteksi antara 0° – 10°. Sistem mampu mengenali ujung jari telunjuk pada ruangan berlatar belakang putih polos dan terdapat sedikit perabot. Waktu respon untuk menampilkan karakter keyboard virtual rata-rata 5,156 detik. Sehingga keyboard virtual pada sistem ini belum mampu dijadikan antarmuka pada aplikasi web, dikarenakan masih sulit digunakan dalam mengarahkan ujung jari telunjuk ke tombol karakter yang diinginkan.

Kata kunci*—aplikasi web, Haar Cascade Classifier, keyboard virtual, pengolahan citra*

Abstract

Since the first computer was founded, keyboard is always been a primary tool for interaction between humans and computers. Today, many computers use image processing technology to make interaction between computers and humans.

The author try to apply image processing technology that implemented to virtual keyboard on web application. Using a webcam to capture the tip of index finger and the results will be sent to the localhost server for processing with image processing. Using Haar Cascade Classifier method to detect the tip of index finger, it will produce coordinates that sent to the web application and it used as a reference for determining button positions on virtual keyboard. Virtual keyboard characters will display after appointed by the tip of index finger.

From testing results, optimal distance from index finger to webcam is 20 – 35 cm. System can recognize the tip of index finger on white background and room with few furnitures. Average response time for displaying virtual keyboard sentences is 3 minutes and 28.838 seconds. So the virtual keyboard on this system was not able to be used as interface on web application, because it difficult to use in directing the tip of index finger to the character keys.

Keywords—*web application, Haar Cascade Classifier, virtual keyboard, image processing*

Received March 26th,2014; Revised January 9th, 2015; Accepted October 1th, 2015

1. PENDAHULUAN

Sejak ertama kali komputer ditemukan, *keyboard* selalu menjadi alat utama yang menjadi penghubung interaksi antara manusia dan komputer. Seiring berkembangnya arus teknologi dan ilmu pengetahuan yang cepat, banyak perangkat komputer sekarang ini yang perlahan meninggalkan bentuk fisik dari *keyboard*, tetapi tidak meninggalkan fungsi utamanya. Sebut saja perangkat komputer tablet maupun *smartphone*.

Tidak ketinggalan pula, saat ini sudah banyak perangkat komputer yang menggunakan teknologi pengolahan citra (*image processing*), khususnya *computer vision*. Dengan *computer vision* manusia bisa berinteraksi dengan komputer hanya dengan gerakan-gerakan khusus pada anggota tubuh atau benda yang berada di sekitarnya tanpa memerlukan perangkat tambahan tertentu. Dengan metode ini pula, diharapkan mampu diterapkan pada *keyboard* virtual untuk berinteraksi dengan perangkat komputer.

Ada berbagai macam yang dapat dijadikan indikator untuk mendekteksi suatu objek dengan komputer. Bisa dengan warna, bentuk, dan anggota tubuh manusia. Pada penelitian yang telah dilakukan sebelumnya, deteksi objek diterapkan pada jari tangan dengan menggunakan *Haar Classifier* pada *library* OpenCV, selanjutnya *tracking* pergerakan jari tangan diterapkan metode *Kalman Filter* yang berfungsi untuk memprediksi posisi jari pada *frame* selanjutnya [1].

Terdapat juga penelitian mengenai penggunaan *keyboard* virtual yang dihubungkan ke berbagai perangkat seperti handphone, PDA, dan PC. *Keyboard* virtual yang dimakan "VistaKey" ini menggunakan antarmuka tombol *keyboard* dengan pancaran sinar proyektor dan sebuah *webcam* sebagai masukannya, sedangkan hasil keluarannya akan ditampilkan di berbagai perangkat bergerak melalui koneksi dengan *server* [2].

Pada penelitian ini dilakukan pendeteksian objek berupa ujung jari telunjuk tangan yang kemudian dapat diterapkan untuk penggunaan *keyboard* virtual sebagai antarmuka pada komputer dan aplikasi web. Pada peneltian ini, digunakan sebuah *webcam* untuk menangkap citra ujung jari telunuk. Kemudian data akan dikirim ke *server* untuk dilakukan pengolahan citra dan hasilnya akan digunakan untuk penggunaan *keyboard* virtual pada aplikasi *web*.

2. METODE PENELITIAN

2.1 Perancangan Sistem Secara Keseluruhan

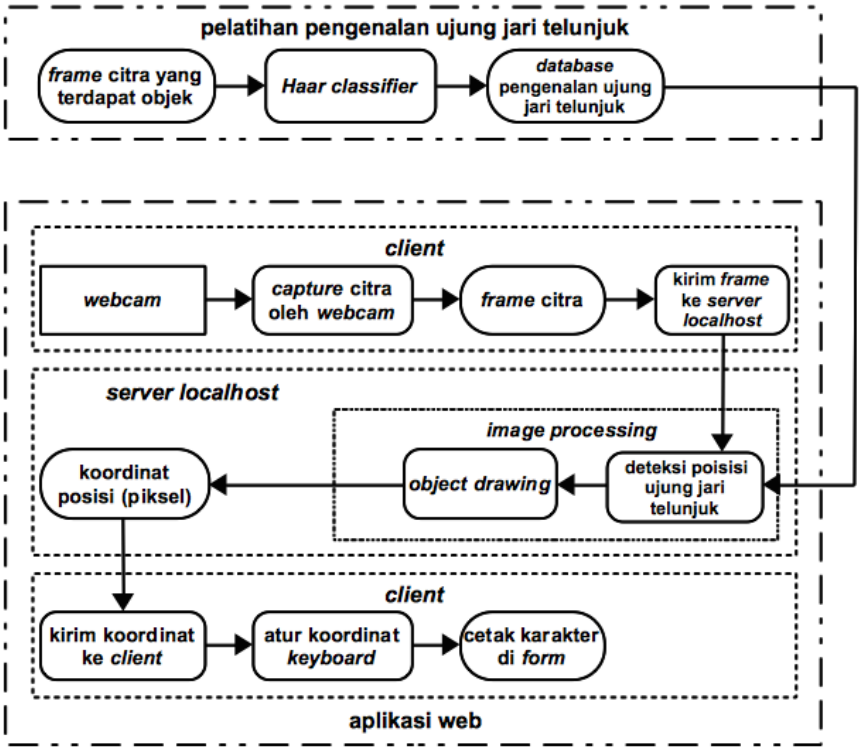
Secara umum, sistem yang dibuat pada penelitian ini terdiri dari dua tahapan, yaitu tahap pelatihan pengenalan ujung jari telunjuk dan tahap pengenalan yang bekerja pada aplikasi *web*. Pada Gambar 1 menunjukkan blok diagram sistem secara keseluruhan, dimana tahap awal adalah melakukan pelatihan pengenalan ujung jari telunjuk pada citra dengan menggunakan metode *Haar Cascade Classifier*. Hasil dari pelatihan pengenalan ujung jari ini akan diperoleh nilai fitur *Haar* yang selanjutnya akan disimpan dalam *database* pengenalan ujung jari telunjuk yang nantinya digunakan sebagai acuan dalam *image processing* pada sistem aplikasi *web*.

Pada tahap pengenalan yang bekerja pada aplikasi *web*, terdapat *client* dan *server* yang bekerja dalam satu *notebook* yang sama. *Client* merupakan sistem yang menjalankan aplikasi *web* melalui *browser*, sedangkan *server* yang digunakan adalah *server localhost*.

Langkah awal yang dilakukan adalah melakukan inialisasi *webcam* oleh *browser*. *Browser* yang digunakan pada penelitian ini adalah Google Chrome versi 31 yang di dalamnya terdapat fitur dukungan *webcam* API (*Application Programming Interface*). Dengan adanya fitur ini sebuah *webcam* dapat merekam citra dan menampilkannya langsung melalui *browser*.

Webcam pada *notebook* yang digunakan dalam penelitian ini diatur untuk dapat bekerja dengan laju 5 *frame per second* (fps), yang artinya setiap 1 detik dapat mengambil *frame* citra objek sebanyak 5 buah. Setiap *frame* yang berhasil ditangkap oleh *webcam* selanjutnya akan dikirim satu per satu ke *server localhost* melalui koneksi WebSockets.

Frame citra yang telah sampai di *server localhost* akan diproses satu per satu pada tahap *image processing* dengan menggunakan *database* pengenalan ujung jari yang telah dihasilkan pada proses pelatihan pengenalan ujung jari telunjuk. Setelah ujung jari telunjuk terdeteksi, maka akan didapatkan koordinat posisi ujung jari telunjuk tersebut. Langkah selanjutnya, dilakukan penggambaran berbentuk lingkaran berwarna hijau oleh *server localhost* berdasarkan koordinat posisi tadi.



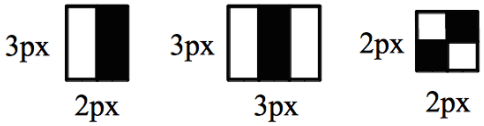
Gambar 1 Blok diagram sistem secara keseluruhan

Setelah dilakukan penggambaran, koordinat posisi yang terdiri dari koordinat piksel horizontal dan vertikal akan dikirimkan ke *client*. Koordinat piksel posisi ujung jari telunjuk inilah yang akan dijadikan acuan dalam menentukan posisi pada masing-masing tombol pada *keyboard* virtual. Sehingga pada bagian akhir proses akan menampilkan huruf pada *form* sesuai dengan koordinat yang ditunjuk oleh ujung jari telunjuk tangan.

2.2 Perancangan Perangkat Lunak

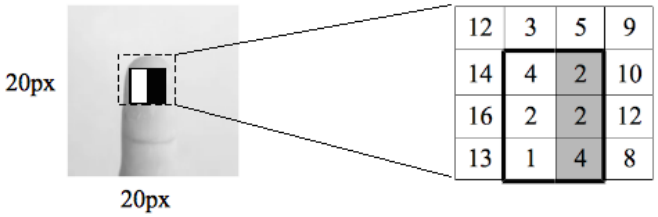
2.2.1 Pelatihan Pengenalan Ujung Jari Telunjuk pada Citra

Untuk dapat mengenali objek yang diinginkan berupa ujung jari telunjuk tangan kanan maupun tangan kiri pada citra, digunakan metode *Haar Cascade Classifier* pada 1100 sampel citra yang di dalamnya terdapat objek yang akan dikenali. Untuk melakukan pengenalan ujung jari telunjuk menggunakan *Haar Cascade Classifier*, terdapat tiga langkah yang harus dilakukan. Langkah pertama, menerapkan *Haar template* pada citra yang terdapat ujung jari telunjuk. Terdapat tiga jenis *Haar template* berdasarkan jumlah persegi yang ada di dalamnya, yaitu persegi dengan daerah gelap dan terang seperti yang tampak pada Gambar 2.



Gambar 2 *Haar Template*

Untuk menentukan ribuan *Haar template* pada sebuah citra secara efisien, dilakukan langkah kedua yaitu, menghitung nilai fitur dengan menggunakan *integral image*. *Integral image* adalah sebuah citra yang nilai setiap pikselnya merupakan hasil akumulasi dari nilai piksel atas dan kirinya [3]. Teknik penghitungan ini bisa dilakukan pada *Haar template* yang diterapkan pada citra seperti yang ditunjukkan oleh Gambar 3. Pada citra yang terdapat ujung jari telunjuk berukuran 20 x 20 piksel akan diterapkan *Haar template* dengan 2 persegi gelap dan terang berukuran 2 x 3 piksel. Pada citra yang akan dilatih, di dalamnya terdapat nilai-nilai piksel, nilai inilah yang akan dirubah ke nilai citra integral dengan menggunakan *integral image*.



Gambar 3 Penerapan *Haar template* pada objek ujung jari telunjuk

Seperti yang ditunjukkan oleh Gambar 3, dilakukan pencuplikan nilai piksel dari citra yang dilatih dengan ukuran 4 x 4 piksel. Kemudian diterapkan *Haar template* dengan ukuran 2 x 3 piksel. Nilai piksel pada daerah yang dilakukan pencuplikan tadi dihitung dengan menggunakan teknik *integral image*, sehingga menghasilkan nilai citra integral seperti yang ditunjukkan Gambar 4.

12	12+3	12+3+5	12+3+5+9
12+14	12+3+14 +4	12+3+14 +4+5+2	12+3+14 +4+5+2 +9+10
12+14+16	12+3+14 +4+16+2	12+3+14 +4+16+2 +5+2+2	12+3+14 +4+16+2 +5+2+2 +9+10+12
12+14+16 +13	12+3+14 +4+16+2 +13+1	12+3+14 +4+16+2 +13+1+5 +2+2+4	12+3+14 +4+16+2 +13+1+5 +2+2+4+9 +10+12+8

12	15	20	29
26	33	40	59
42	51	60	91
55	65	78	177

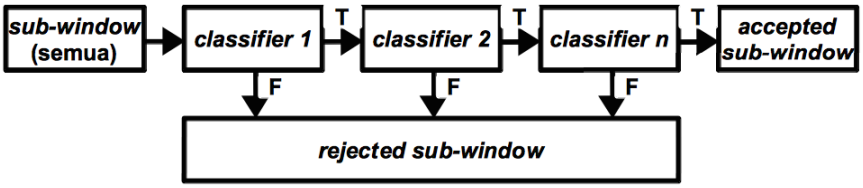
Gambar 4 Perhitungan dengan *integral image*

Nilai fitur *Haar* = | (nilai piksel daerah hitam) – (nilai piksel daerah putih) |
= | [78+15 – (65+20)] – [65+12 – (55+15)] |
= | 8 – 7 |
= 1

Selanjutnya adalah menghitung nilai piksel dari masing-masing daerah gelap dan terang pada area *Haar template*. Dari perhitungan di atas, pengurangan piksel daerah gelap dan terang akan didapatkan nilai fitur *Haar* sebesar 1. Nilai fitur *Haar* sebesar 1 inilah yang akan digunakan sebagai parameter klasifikasi objek yang terdeteksi sebagai ujung jari atau bukan. Jika nilai fitur berada di atas 1, maka citra dianggap terdapat ujung jari telunjuk. Sebaliknya jika nilai fitur di bawah 1, maka citra dianggap tidak terdapat ujung jari telunjuk. Sehingga nilai fitur *Haar* yang didapat akan dijadikan nilai ambang atau *threshold*. Perhitungan nilai fitur ini akan berlanjut dengan menggunakan fitur *Haar* lain dan di posisi yang berbeda pada citra yang dilakukan pelatihan.

Setelah mendapatkan nilai fitur *Haar*, dilakukan langkah ketiga yaitu, melatih *classifier* berdasarkan berbagai nilai fitur *Haar* dengan menggunakan algoritma *machine learning* yang disebut Adaptive Boost atau AdaBoost. Pada *classifier* di dalamnya terdapat *weak classifier* yang terdiri dari nilai fitur, posisi dan, ukuran *Haar tempalate* yang dilatih pada citra. Untuk mendapatkan *classifier* yang kuat dilakukan metode pengklasifikasian bertingkat, dimana *input* dari setiap tingkatan merupakan *output* dari tingkatan sebelumnya. Semakin tinggi tingkat *classifier*, maka semakin banyak pula jumlah *weak classifier* yang ada. Hal ini mengakibatkan semakin sulitnya suatu *sub-window* untuk berhasil melewati tingkatan *classifier* tersebut, sehingga jumlah *sub-window* yang dieleminasi akan semakin banyak, dan jumlah *sub-window* yang berhasil lolos ke *classifier* tingkat selanjutnya akan semakin sedikit. Urutan dalam melatih *classifier* ditentukan oleh bobot nilai fitur yang diberikan AdaBoost. Nilai fitur dengan bobot paling besar diletakkan pada proses *classifier* yang pertama kali yang bertujuan untuk menghapus daerah citra yang bukan ujung jari telunjuk secepat mungkin. Untuk tingkatan *classifer* yang berikutnya diberikan nilai fitur dengan bobot yang lebih kecil dari tingkat *classifier* yang pertama. Begitu juga dengan nilai bobot fitur untuk tingkatan *classifier* yang berikutnya [4]. Hasil dari melatih *classifier* ini bertujuan untuk mendapatkan suatu keputusan apakah di dalam citra terdapat ujung jari telunjuk yang diinginkan atau tidak. Gambar 5 menunjukkan alur kerja dari klasifikasi bertingkat untuk melatih *classifier*.

Agar proses pengenalan ujung jari bisa diterapkan pada aplikasi *web* melalui tahap *image processing*, maka nilai fitur dan *classifier* yang telah dilatih melalui pengklasifikasian bertingkat akan disimpan dalam bentuk *database* pengenalan objek ujung jari telunjuk.



Gambar 5 Alur kerja klasifikasi bertingkat untuk melatih *classifier*

2.2.2 Mengakses Webcam dengan Browser

Proses inialisasi ini akan dilakukan oleh *browser* yang memiliki fitur dukungan *webcam* API (*Application Programming Interface*). Dengan menggunakan HTML5 `getUserMedia` API yang berbasis pada pemrogaman JavaScript yang telah dicantumkan pada sebuah aplikasi *web*, maka API ini secara otomatis akan dapat mengakses *webcam* dan *microphone* pada *notebook*. Langkah selanjutnya adalah melakukan *recording* dan menampilkannya secara langsung di aplikasi *web*. *Webcam* yang digunakan pada penelitian ini melakukan *capture* citra dengan ukuran 640 x 480 piksel. Untuk menampilkan hasil *capture* dari *webcam*, digunakan teknik penggambaran tiap *frame* dengan HTML5 Canvas yang dilakukan setiap 200 ms, sehingga *frame* yang telah digambarkan seolah-olah bergerak seperti video hasil *recording* dari *webcam*. Hasil penggambaran oleh HTML5 Canvas akan menghasilkan *frame* citra berbentuk data *raw* (*blob*).

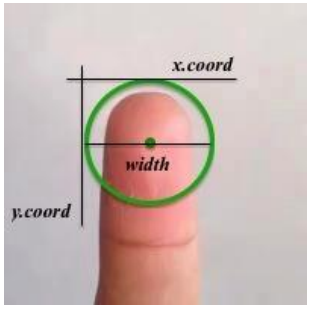
2.2.3 Pengiriman Frame Citra ke Server Localhost

Hasil penggambaran tiap *frame* citra dengan HTML5 Canvas yang berbentuk data *raw* akan dikirimkan secara langsung ke *server localhost*. *Frame* hasil penggambaran dari HTML5 Canvas berbentuk data *raw* ini akan dirubah terlebih dahulu ke format *image* yaitu `jpg`. Setelah berhasil membuat *image* dengan format `jpg`, maka dilakukan proses pengiriman ke *server localhost* melalui koneksi WebSockets. Setelah sampai di *server localhost*, *frame* citra akan disimpan ke *buffer* yang bersifat sementara (*temporary*).

2.2.4 Deteksi Posisi Ujung Jari Telunjuk

Untuk melakukan deteksi posisi ujung jari telunjuk pada tahap *image processing* ini, dibutuhkan *database* hasil dari pelatihan pengenalan ujung jari telunjuk pada tahap yang pertama. Setiap *frame* citra yang disimpan dalam *buffer*, akan dilakukan proses pendeteksian dengan menggunakan metode *cascade classifier* atau klasifikasi bertingkat seperti yang ditunjukkan prosesnya pada Gambar 5. *Frame* citra yang telah diambil dari *buffer* akan melewati tahap *classifier* pertama dengan inputan adalah seluruh *sub-window* pada *frame* citra. Semua *sub-window* pada *frame* citra yang telah berhasil melewati *classifier* pertama akan dilanjutkan ke tahap *classifier* kedua, kemudian *sub-window* yang juga berhasil melewati *classifier* kedua akan dilanjutkan ke tahap *classifier* ketiga, dan seterusnya. Apabila suatu *sub-window* berhasil melewati semua tingkatan *classifier*, maka *sub-window* tersebut dinyatakan sebagai ujung jari telunjuk. Sedangkan untuk *sub-window* yang gagal melewati suatu tingkat *classifier* akan langsung dieleminasi dan dinyatakan sebagai bukan ujung jari telunjuk.

Frame citra yang berhasil terdeteksi di dalamnya terdapat ujung jari telunjuk akan menghasilkan nilai koordinat awal dan panjang area dari ujung jari telunjuk. Untuk menentukan posisi ujung jari telunjuk, perlu dicari nilai titik tengah dari area ujung jari telunjuk yang terdeteksi seperti yang ditunjukkan pada Gambar 6.



Gambar 6 Area pada ujung jari telunjuk yang terdeteksi

Untuk menghitung nilai titik tengah dari area ujung jari telunjuk yang terdeteksi digunakan persamaan (1), yaitu nilai koordinat awal pada tepi area (*x.coord* dan *y.coord*) kemudian ditambahkan dengan setengah nilai panjang area (*width* / *height*). Maka akan didapatkan nilai koordinat posisi ujung jari telunjuk yang sebenarnya pada sumbu *x* (horizontal) maupun sumbu *y* (vertikal). Nilai koordinat piksel yang dihasilkan berkisar antara 0 – 640 piksel untuk koordinat horizontal dan 0 – 480 piksel untuk koordinat vertikal sesuai dengan ukuran *frame* citra hasil *capture* dari *webcam* yaitu 640 x 480 piksel.

$$x\ Axis = x.coord + width / 2 \quad (1)$$
$$y\ Axis = y.coord + height / 2$$

Dengan didapatkannya kedua nilai koordinat posisi objek di atas, maka nilai tersebut yang akan dijadikan acuan untuk menentukan posisi masing-masing tombol pada *keyboard* virtual. Untuk memudahkan pembacaan nilai koordinat posisi objek pada aplikasi *web* di *client*, maka nilai tersebut dirubah ke dalam format JSON yang mendukung bahasa pemrogaman JavaScript. Format yang disajikan adalah `{ x = xAxis, y = yAxis }`.

2.2.5 Object Drawing

Proses tahap kedua di bagian *image processing* ini akan melakukan penggambaran pada ujung jari yang terdeteksi (*object drawing*). Penggambaran dilakukan dengan mengacu pada nilai titik tengah dan panjang area pada ujung jari yang terdeteksi. Proses penggambaran yang dilakukan adalah dengan menggambarkan bentuk lingkaran berwarna hijau pada ujung jari telunjuk yang terdeteksi.

Apabila suatu *frame* citra tidak berhasil mendeteksi ujung jari telunjuk, maka proses *object drawing* tidak akan dilakukan. Sehingga *frame* citra yang berhasil maupun tidak berhasil

dilakukan penggambaran akan disimpan di kembali dalam *buffer* yang bersifat sementara/*temporary*.

2.2.6 Menampilkan Karakter dan Merancang Antarmuka Aplikasi Web

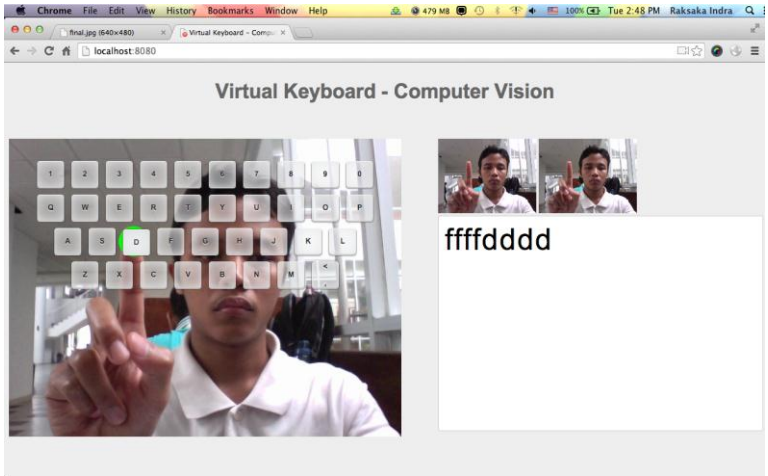
Untuk menampilkan karakter dari tombol-tombol *keyboard* virtual digunakan pemrograman JavaScript. Karakter yang akan ditampilkan adalah huruf dari A – Z dan angka dari 0 – 9. Setiap karakter yang ditampilkan akan ditentukan terlebih dahulu nilai koordinat posisinya. Untuk menampilkan karakter yang dipilih oleh ujung jari telunjuk, maka nilai koordinat posisi ujung jari telunjuk yang telah terdeteksi dalam format JSON akan dibaca oleh JavaScript. Karakter akan tampil ketika koordinat posisi ujung jari telunjuk mempunyai nilai yang sama dengan *range* nilai koordinat yang telah ditentukan pada masing-masing karakter *keyboard* virtual. Misal, koordinat tombol karakter Q pada *keyboard* virtual diatur koordinat posisinya antara $50 \leq x \leq 86$ dan $93 \leq y \leq 129$. Ketika koordinat posisi ujung jari telunjuk yang terdeteksi bernilai x = 70 dan y = 100, maka karakter Q akan tampil pada *form*.

2.2.7 Implementasi Sistem

Implementasi sistem ini menggunakan perangkat keras dan perangkat lunak sebagai berikut:

1. MacBook Pro MB990, 2.26 Ghz Intel Core 2 Duo, 4GB 1067 Mhz DDR3 RAM
2. Sistem Operasi Mac OS X 10.8.5 Mountain Lion
3. iSight (*webcam*) pada MacBook Pro
4. OpenCV v2.5
5. Node.js v0.10.20
6. Browser Google Chrome v31.0
7. jQuery v2.0.0

Untuk melakukan implementasi sistem pada penelitian ini adalah dengan menjalankan *server localhost* melalui Terminal. Setelah sistem *server localhos*t berjalan, buka *browser* Google Chrome dan ketikkan alamat URL http://localhost:8080/. Ketika mengakses alamat tersebut, *browser* akan melakukan inialisasi dan mengaktifkan *webcam*. *Webcam* melakukan *capture* citra dan menampilkannya di aplikasi *web*, posisikan ujung jari telunjuk mengarah ke arah *webcam* sehingga ujung jari telunjuk ter-*capture* dan langsung terlihat di aplikasi *web*. Selang seper sekian detik, ujung jari akan terdeteksi dengan ditandai lingkaran berwarna hijau. Arahkan ujung jari telunjuk ke tombol karakter pada *keyboard* virtual yang diinginkan. Maka akan terlihat efek menekan tombol *keyboard* virtual dan selang beberapa detik karakter yang dipilih akan ditampilkan di *form* pada aplikasi *web*. Hasil implementasi sistem berupa antarmuka pada aplikasi *web* ditunjukkan pada Gambar 7.



Gambar 7 Hasil implementasi sistem pada aplikasi *web* di Google Chrome

3. HASIL DAN PEMBAHASAN

3.1 Pengujian Pengaruh Intensitas Cahaya dalam Ruangan

Pengujian pengaruh intensitas cahaya digunakan di dalam ruangan dengan 4 kondis yang berbeda. Untuk mengukur besarnya intensitas cahaya, penulis menggunakan *light meter* dengan merk Krisbow® seri KW06-288. *Light meter* berfungsi untuk mengukur tingi rendahnya intensitas cahaya dengan satuan *lux* yang ada di sekitar ruangan pengujian.

Tabel 1 Hasil pengujian pengaruh intensitas cahaya dalam ruangan

No.	Kondisi Ruangan	Intensitas Cahaya (*lux*)	Jumlah Percobaan	Keterangan
1.	Gelap	5 – 10	10	0 terdeteksi, 10 *error*
2.	Cahaya lampu sedang	30 – 50	10	9 terdeteksi, 1 *error*
3.	Cahaya matahari	60 – 80	10	10 terdeteksi, 0 *error*
4.	Cahaya lampu terang	100 – 125	10	10 terdeteksi, 0 *error*

Berdasarkan hasil pengujian yang telah dilakukan sebanyak 10 kali percobaan, dapat dilihat pada Tabel 1 bahwa sistem dapat mendeteksi ujung jari telunjuk dengan baik pada 3 ruangan, yaitu ruangan dengan cahaya lampu sedang, lampu terang, dan terkena cahaya matahari. Pada ruangan dengan lampu terang dan ruangan yang terkena cahaya matahari, sistem berhasil mendeteksi ujung jari telunjuk sebanyak 10 kali. Sedangkan pada ruangan dengan cahaya lampu sedang, sistem berhasil mendeteksi ujung jari telunjuk sebanyak 9 kali. Dengan demikian, salah satu faktor utama yang mempengaruhi pendeteksian objek adalah intensitas cahaya pada ruang uji, semakin gelap kondisi ruangan, maka semakin kecil nilai intensitas cahayanya sehingga sistem akan semakin sulit untuk mendeteksi objek ujung jari telunjuk.

3.2 Pengujian Jarak Ujung Jari Telunjuk Terhadap Webcam

Pengujian jarak objek ujung jari telunjuk terhadap *webcam* dilakukan untuk mengetahui jarak maksimal dan minimal yang dapat dideteksi oleh sistem. Pengujian ini dilakukan dengan cara melakukan variasi jarak horizontal antara *webcam* dengan ujung jari telunjuk pada ruangan terang dengan intensitas cahaya antara 100 – 125 *lux*.

Tabel 2 Hasil pengujian jarak ujung jari telunjuk terhadap *webcam*

No.	Jarak (cm)	Jumlah Percobaan	Keterangan
1.	10	10	10 terdeteksi, 0 *error*
2.	15	10	10 terdeteksi, 0 *error*
3.	20	10	10 terdeteksi, 0 *error*
4.	25	10	9 terdeteksi, 1 *error*
5.	30	10	10 terdeteksi, 0 *error*
6.	35	10	9 terdeteksi, 1 *error*
7.	40	10	8 terdeteksi, 2 *error*
8.	45	10	6 terdeteksi, 4 *error*
9.	50	10	6 terdeteksi, 4 *error*
10.	55	10	3 terdeteksi, 7 *error*
11.	60	10	1 terdeteksi, 9 *error*
12.	65	10	0 terdeteksi, 10 *error*
13.	> 65	10	0 terdeteksi, 10 *error*

Dari hasil pengujian yang dilakukan sebanyak 10 kali percobaan, seperti yang terlihat pada Tabel 2 bahwa jarak yang dapat dideteksi oleh sistem melalui *webcam* terletak pada jarak 5 hingga 60 cm. Pada jarak 10 – 50 cm menghasilkan objek yang paling banyak terdeteksi, sedangkan pada jarak lebih dari 50 cm objek sudah sulit terdeteksi oleh sistem melalui *webcam*. Semakin jauh jarak objek terhadap *webcam*, maka semakin kecil objek sehingga sistem sulit untuk melakukan pendeteksian. Untuk jarak yang paling optimal adalah antara 20 – 35 cm, karena pada jarak tersebut ukuran ujung jari telunjuk yang terdeteksi sesuai dengan ukuran tombol pada *keyboard* virtual. Sedangkan pada jarak kurang dari 20 cm, ukuran ujung jari yang terdeteksi terlalu besar dikarenakan semakin dekat dengan *webcam*. Akibatnya dua tombol *keyboard* virtual akan tertekan secara bersamaan.

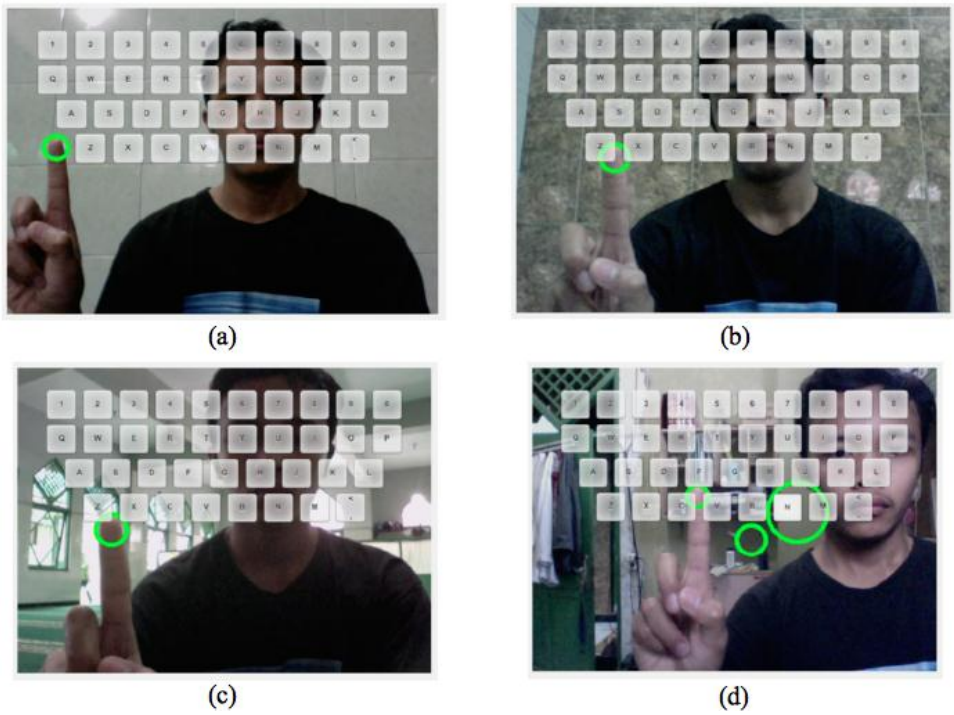
3.3 Pengujian dengan Variasi Latar Belakang/Background Ruangan

Latar belakang atau *background* adalah salah satu hal yang dapat mempengaruhi pemrosesan pengolaha citra. Oleh karena itu, perlu dilakukan pengujian sistem dengan variasi latar belakang di dalam ruangan.

Tabel 3 Hasil dengan pengujian variasi latar belakang ruangan

No.	Jenis *Background* Ruangan	Jumlah Percobaan	Keterangan
1.	Putih dan polos	10	10 terdeteksi, 0 *error*
2.	Coklat dan bercorak	10	6 terdeteksi, 4 *error*
3.	Ruangan I	10	9 terdeteksi, 1 *error*
4.	Ruangan II	10	8 terdeteksi, 2 *error*

Dari hasil pengujian yang dilakukan sebanyak 10 kali percobaan, seperti yang ditunjukkan oleh Tabel 3 bahwa sistem dapat mendeteksi ujung jari dengan baik pada ruangan dengan *background* berwarna putih dan polos, terbukti dengan hasil percobaan yang berhasil sebanyak 10 kali. Hasil pengujian dengan latar belakang ruangan yang berwarna putih polos ditunjukkan pada Gambar 8 (a).



Gambar 8 (a) Pengujian pada latar belakang putih polos (b) Pengujian pada latar belakang coklat bercorak (c) Pengujian pada ruangan dengan sedikit perabot (d) Pengujian pada ruangan dengan banyak perabot

Pada ruangan berlatar belakang berwarna coklat dan bercorak, sistem sedikit mengalami kesulitan dalam mendeteksi ujung jari telunjuk, terbukti hanya 6 kali yang berhasil terdeteksi. Hal ini dikarenakan warna coklat pada latar belakang memiliki warna yang hampir sama dengan warna kulit pada ujung jari telunjuk. Sistem mengalami kesulitan membedakan latar belakang dengan ujung jari dikarenakan nilai fitur yang dihasilkan antara daerah latar belakang dan objek di bawah nilai ambang (*threshold*). Sedangkan agar berhasil mendeteksi objek yang diinginkan, nilai fitur harus di atas nilai ambang (*threshold*). Hasil pengujian dengan latar belakang ruangan yang berwarna coklat dan bercorak ditunjukkan pada Gambar 8 (b).

Pada ruangan dengan latar belakang yang memiliki sedikit perabot, sistem dapat mendeteksi ujung jari telunjuk dengan baik, terbukti dengan 9 kali berhasil mendeteksi ujung jari di dalam percobaan. Hasil pengujian dengan latar belakang ruangan yang memiliki sedikit perabot ditunjukkan pada Gambar 8 (c).

Sedangkan pada ruangan dengan latar belakang yang memiliki banyak perabot, sistem juga masih dapat mendeteksi ujung jari telunjuk, terbukti berhasil melakukan pendeteksian sebanyak 8 kali. Namun, seperti yang ditunjukkan hasil pengujian pada Gambar 8 (d), terjadi kesalahan pada sistem dalam melakukan pendeteksian. Sistem juga mendeteksi benda-benda lain dengan ditandai adanya lingkaran berwarna hijau pada *frame*. Hal ini disebabkan benda-benda tersebut memiliki bentuk yang hampir sama dengan bentuk ujung jari telunjuk. Sehingga sistem menganggapnya sebagai ujung jari telunjuk.

Sehingga sistem akan mendeteksi ujung jari telunjuk dengan baik pada latar belakang yang berwarna putih polos atau di dalam ruangan dengan sedikit perabot di dalamnya. Sedangkan pada latar belakang berwarna coklat dan bercorak, sistem mengalami kesulitan mengenali ujung jari telunjuk dan terjadi kesalahan dalam mendeteksi objek selain ujung jari telunjuk.

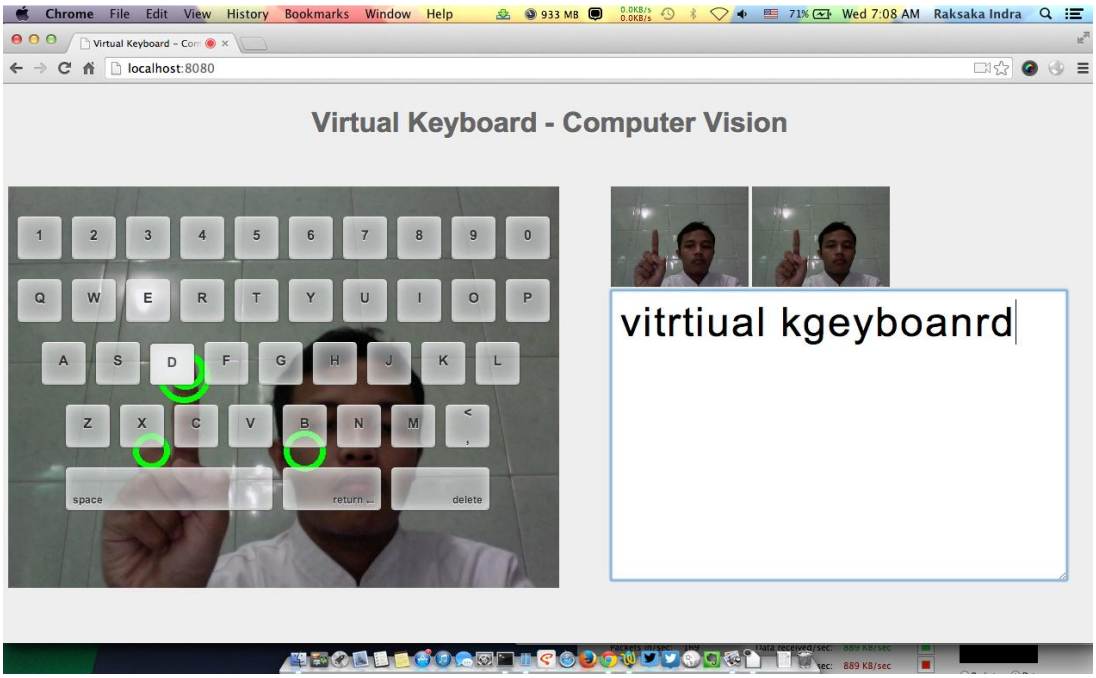
3.4 Pengujian Ketepatan dan Waktu Respon dalam Menampilkan Karakter Keyboard Virtual

Perlu dilakukan pengujian ketepatan dalam menampilkan karakter untuk mengetahui tingkat keakuratan kinerja yang diberikan sistem melalui *keyboard* virtual sebagai antarmuka. Pengujian ini dilakukan dengan menuliskan susunan kata yang diinginkan dengan menggunakan *keyboard* virtual. Susuanan kata yang akan ditampilkan adalah "VIRTUAL KEYBOARD". Pengujian ini masih dilakukan pada ruangan dengan latar belakang putih polos dan dengan jarak ujung jari terhadap *webcam* 20 – 35 cm.

Tabel 4 Hasil pengujian ketepatan dalam menampilkan karakter *keyboard* virtual

No.	Percobaan ke-	Karakter/Kata yang Tampil	Jumlah*Error* Karakter	Waktu respon (menit)
1.	1	VIRTUAL KEJYMBOARVD	3	4:13.89
2.	2	VITRTIUAL KGEYBOANRD	4	4:28.32
3.	3	VPIRGTUAL KEYBOARD	2	2:50.46
4.	4	VITRNTUAL KTEUYBOARD	4	3:13.87
5.	5	VIRTUZAL KEYBJOARD	2	2:37.65

Dari hasil percobaan yang telah dilakukan sebanyak 5 kali dalam menampilkan susunan kata "VIRTUAL KEYBOARD", seperti yang ditunjukkan pada Tabel 4 bahwa *keyboard* vitual masih sering melakukan kesalahan (*error*) dalam menampilkan karakter yang tidak diinginkan dengan rata-rata *error* yang terjadi adalah 3 karakter. Kesalahan menampilkan karakter tersebut bisa disebabkan jarak antar tombol karakter *keyboard* virtual yang terlalu sempit sehingga salah dalam menujuk karakter, atau bisa juga disebabkan sistem mendeteksi objek selain ujung jari telunjuk. Kesalahan dalam pendeteksian objek selain ujung jari telunjuk dapat terlihat pada Gambar 9 dimana terdapat 2 lingkaran berwarna hijau yang terdeteksi selain ujung jari telunjuk, yaitu pada ruas tulang bagian luar jari tengah dan tulang pipi.



Gambar 9 Kesalahan pendeteksian selain ujung jari telunjuk pada pengujian ketepatan dalam menampilkan karakter *keyboard* virtual

Dari pengujian ini menujukkan *keyboard* virtual dalam aplikasi *web* ini belum bisa dijadikan sebagai antarmuka dikarenakan masih sulit digunakan dan tidak mendatangkan kemudahan bagi pengguna dalam berinteraksi, yang meliputi masih seringnya terjadi kesalahan menampilkan karakter yang tidak diinginkan dan waktu respon yang cukup lama dalam menuliskan susunan kata "VIRTUAL KEYBOARD" dengan rata-rata 3 menit 28,838 detik.

4. KESIMPULAN

Dari penelitian yang telah dilakukan, dapat diambil beberapa kesimpulan sebagai berikut:

1. Metode yang digunakan untuk mengenali ujung jari telunjuk dengan menggunakan metode *Haar Cascade Classifier* sudah berjalan dengan cukup baik.
2. Jarak optimal secara horizontal antara ujung jari telunjuk dan *webcam* adalah 20 – 35 cm.
3. Sistem mampu mengenali ujung jari telunjuk pada ruangan berlatar belakang warna putih polos dan ruangan dengan sedikit perabot di dalamnya. Sistem mengalami kesulitan mengenali ujung jari telunjuk pada ruangan berlatar belakang yang bercorak. Sedangkan pada ruangan dengan banyak perabot di dalamnya, sistem mengalami kesalahan berupa mendeteksi objek selain ujung jari telunjuk.
4. Waktu respon yang dibutuhkan untuk menampilkan susunan karakter "VIRTUAL KEYBOARD" pada *keyboard* virtual terhitung cukup lama dan kurang baik, dengan rata-rata 3 menit 28,838 detik.
5. *Keyboard* virtual pada sistem ini belum mampu dijadikan sebagai antarmuka pada aplikasi *web*, dikarenakan masih sulit digunakan dalam mengarahkan ujung jari telunjuk ke tombol karakter yang diinginkan sehingga tidak mendatangkan kemudahan bagi pengguna dalam berinteraksi, serta masih terdapat *error* dalam menampilkan susunan kata/karakter pada *keyboard* virtual.

5. SARAN

Beberapa saran dari penulis untuk pengembangan sistem lebih lanjut adalah sebagai berikut:

1. Sebaiknya sistem ini dikembangkan agar dapat berjalan di *cloud server*, sehingga sistem dapat diakses dan digunakan oleh banyak orang dengan menggunakan *browser*.
2. Perlu diujikan pada berbagai *browser* yang telah beredar luas dan dapat digunakan juga pada tablet atau *smartphone*.
3. Dikarenakan antarmuka berupa *keyboard* virtual masih kurang efektif ketika diimplementasikan, maka perlu dikembangkan antarmuka berupa interaksi menggambar atau dapat mengendalikan *game* menggunakan ujung jari telunjuk pada aplikasi *web*.

DAFTAR PUSTAKA

[1] Ramadijanti, N., Setiawardhana, dan Alhaqqi, R.M., 2010, Tracking Jari dengan Haar Cascade dan Filter Kalman pada Virtual Keyboard, *Invotek*, No.1, Vol.3, 1-9.

[2] Wijewantha, N.S., 2003, VistaKey: A Keyboard Without A Keyboard – A Type Of Virtual Keyboard, *Tesis*, Department of Computing Informatics Institute of Technology, Wellawatta, Sri Lanka.

[3] Santoso, H. dan Harjoko, A., 2013, Haar Cascade Classifier dan Algoritma AdaBoost untuk Deteksi Banyak Wajah dalam Ruang Kelas, *Jurnal Teknologi*, No.2, Vol.6, 108-115.

[4] Putro, M.D., Adji, T.B., dan Winduratna, B., 2012, Sistem Deteksi Wajah dengan Menggunakan Metode Viola-Jones, *Seminar Nasional "Science, Engineering and Technology"*, Yogyakarta.