PENGENDALI POINTER DENGAN *GAZE TRACKING* MENGGUNAKAN METODE *HAAR CLASSIFIER* SEBAGAI ALAT BANTU PRESENTASI (*EYE POINTER*)

Edi Satriyanto (edi@eepis-its.edu)
Fernando Ardilla
Risa Indah Agustriany Lubis
Politeknik Elektronika Negeri Surabaya
Kampus ITS Keputih Sukolilo Surabaya 60111
Telp(62)31-5910040 fax ((+62)31-5964677

ABSTRACT

The application that builded in this research is a pointer controller using eye movement (eye pointer). This application is one of image processing applications, where the users just have to move their eye to control the computer pointer. This eye pointer is expected able to assist the usage of manual pointer during the presentation. Since the title of this research is using gaze tracking that follow the eye movement, so that is important to detect the center of the pupil. To track the gaze, it is necessary to detect the center of the pupil if the eye image is from the input camera. The gaze tracking is detected using the three-step hierarchy system. First, motion detection, object (eye) detection, and then pupil detection. For motion detection, the used method is identify the movement by dynamic compare the pixel ago by current pixel at t time. The eye region is detected using the Haar-Like Feature Classifier, where the sistem must be trained first to get the cascade classifier that allow the sistem to detect the object in each frame that captured by camera. The center of pupil is detect using integral projection.
The final step is mapping the position of center of pupil to the screen of monitor using comparison scale between eye resolution with screen resolution. When detecting the eye gaze on the screen, the information (the distance and angle between eyes and a screen) is necessary to compute pointing coordinates on the screen. In this research, the accuracy of this application is equal to 80% at eye movement with speed 1-2 second. And the optimum mean value is between 5 and 10. The optimum distance of user and the webcam is 40 cm from webcam.

Key words : eye pointer, haar classifier, motion detection, object detection

Hasil peneilitian Irawan dan Satriyanto (2008) menjelaskan bahwa penggunaan komputer yang membutuhkan interaksi antara manusia dan komputer dewasa ini semakin banyak dikembangkan. Oleh karena itu saat ini banyak bermunculan perangkat komputer yang berguna untuk memudahkan manusia dalam berinteraksi dengan komputer. Saat ini mulai berkembang sebuah teknologi yang menginginkan agar komputer juga dapat melihat seperti halnya manusia dapat melihat. Dengan adanya kamera maka alat ini bisa dijadikan sebagai mata bagi komputer untuk dapat melihat lingkungan di sekitarnya.

Dua teknologi yang mengarah ke perkembangan itu dan sekaligus juga mendukungnya adalah *Computer Vision* dan *Image Processing*. *Computer Vision* mempunyai tujuan untuk membuat

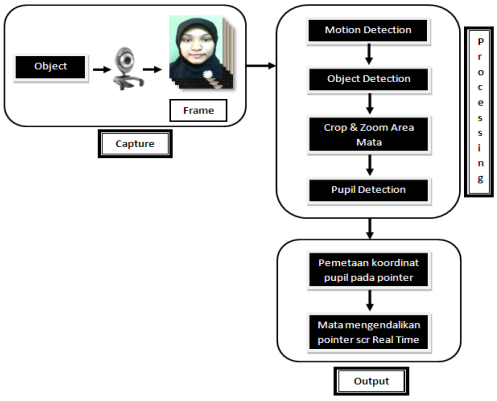
keputusan yang berguna tentang obyek fisik nyata dan pemandangan berdasarkan *image* yang didapat dari sensor. Secara sederhana *Computer Vision* ingin membangun sebuah mesin pandai yang dapat melihat. *Image Processing* merupakan salah satu jenis teknologi untuk menyelesaikan masalah mengenai pemrosesan gambar. Dalam *Image Processing* gambar yang ada diolah sedemikian rupa sehingga gambar tersebut lebih mudah untuk diproses. Penelitian Dian dan Satriyanto (2006) menjelaskan kamera dipakai sebagai sensor untuk menggerakkan pointer pada saat melakukan presentasi. Untuk itu diperlukan teknologi *Computer Vision* dan juga *Image Processing*, sehingga kamera dapat berperan sebagai alat bantu presentasi dan mempermudah pengguna dalam melakukan presentasi, khususnya bagi mereka yang mengalami cacat fisik pada bagian tangan.

Obyek yang menjadi acuan penggerak pointer ini adalah gerakan mata user yang sedang melakukan presentasi. Pointer ini sendiri nantinya akan digerakkan secara *real time* oleh mata user dengan mengikuti arah pandang/gerakan mata (*gaze tracking*), sehingga dengan penggunaan mata sebagai pengendali pointer pada saat presentasi, diharapkan memudahkan user dalam melakukan presentasi tanpa harus repot menggerakkan mouse komputer. Hal ini tentunya memudahkan pengguna, khususnya para penyandang cacat fisik (pada bagian tubuh motorik) dalam melakukan presentasi. Di samping itu penelitian ini diharapkan mampu membantu peranan pointer manual pada saat melakukan presentasi.

METODE PENELITIAN

Blok Dagram Sistem

Metode yang digunakan untuk mengenali sebuah obyek yang bergerak sangat beraneka ragam. Salah satunya adalah dengan mengidentifikasi dari bentuk (fitur) sebuah obyek yang diamati. Dalam membangun aplikasi *eye pointer*, untuk mendeteksi obyek yang bergerak dalam penelitian ini digunakan metode *Haar Classifier*. Hasil dari training yang dilakukan adalah sebuah pohon keputusan yang di sebut dengan *cascade classifier*. *Cascade* inilah yang kemudian akan mendeteksi obyek yang telah dilatihkan pada sistem tadi. Dalam penelitian ini obyek tersebut adalah sepasang mata manusia. Arah gerakan mata inilah yang kemudian akan menggerakkan pointer pada layar komputer. Berikut ini adalah blok diagram sistem yang diterapkan pada aplikasi.



Gambar 1. Blok diagram sistem *Eye Pointer*

Gambar 1 merupakan blok diagram dari sistem *eye pointer*. Dalam pembuatan aplikasi *eye pointer*, dilakukan pendeteksian gerak dan obyek untuk menentukan ada atau tidaknya obyek yang diharapkan pada tiap frame yang diproses. Apabila mata sudah terdeteksi, maka langkah berikutnya adalah mencari posisi pupil untuk kemudian dipetakan pada koordinat layar komputer. Aplikasi ini berjalan secara *real-time* mengikuti gerakan mata user.

Deteksi Pergerakan (*Motion Detection*)

Pada penelitian Irawan dan Satriyanto (2008) metode yang digunakan untuk mendeteksi pergerakan adalah menggunakan algoritma komputasi dengan membandingkan data piksel yang lalu dengan data piksel saat ini pada satuan t waktu. Medan pergerakan memberikan informasi yang bernilai mengenai karakteristik (misalnya kurva dan orientasi) dan kedalaman dari permukaan sebagai gerakan relatif di sekeliling obyek dan sistem sensor.

Dalam deteksi pergerakan obyek terdapat suatu proses *matching* yang disebut *Region-Bases Matching* yaitu membandingkan dua obyek yang telah mengalami pergeseran terhadap waktu, sehingga diperoleh nilai *mean* dari posisi pergeseran tersebut.

Langkah–langkah yang dilakukan untuk mendeteksi adanya gerakan adalah dengan cara sebagai berikut :

- Frame yang tertangkap diubah ke bentuk *gray scale*.
- Tiap frame akan dibandingkan dengan frame sebelumnya.
- Bila timbul suatu perbedaan nilai piksel pada posisi yang sama di tiap frame, maka program akan menghitung perbedaan yang terjadi antara dua *image* dengan cara menghitung rata-rata (*mean*) dari semua nilai *gray value* dalam suatu gambar.
- Nilai *mean* yang didapat akan dibandingkan dengan nilai *threshold* yang ditentukan oleh user. Nilai *threshold* dipakai sebagai acuan pendeteksian gerakan.
- Semakin kecil batas nilai *threshold* maka deteksi pergerakan akan semakin sensitif.
- Jika nilai *mean* melebihi batas nilai *threshold* yang ditentukan maka gerakan akan terdeteksi.
- Rumus dari nilai *mean* yang dipakai adalah :

$$mean = \frac{\sum_{1}^{p}\sum_{1}^{l}\left|pixel(p,l)_{f} - pixel(p,l)_{f-1}\right|}{p \times l} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots (1)$$

Dimana :

p = panjang frame (piksel)
l = lebar frame (piksel)
f = frame

Deteksi Obyek Menggunakan *Haar Classifier*

Untuk melakukan pendeteksian obyek, sistem ini menggunakan metode *Haar Classified* di mana sebelum sistem dapat melakukan proses pendeteksian, sistem harus dilatih terlebih dahulu untuk menghasilkan suatu *cascade classifier* seperti dalam penelitian Magee, *et.al*. Training dilakukan dengan melatih sistem dalam mengenali data positif (data obyek) serta data negatif (data yang tidak mengandung obyek). Jumlah sampel yang banyak serta jumlah *stage* yang banyak pula akan membuat sistem menjadi lebih akurat dalam melakukan pendeteksian obyek. Namun kendalanya adalah proses training yang dilakukan akan memakan waktu yang lama.

Cascade classifier yang dihasilkan digunakan untuk membuang *sub-window* mayoritas (data negatif) sebelum nantinya dilakukan pengklasifikasian yang lebih kompleks untuk mendapatkan tingkat error yang rendah pada saat pendeteksian data positif. Keseluruhan format proses pendeteksian adalah dari rangkaian pohon keputusan (*decission tree*), yang kemudian disebut sebagai *cascade*.

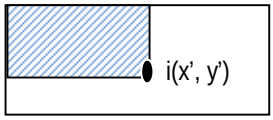
Tiap *sub window* dideteksi dengan menghitung nilai fitur dari *sub-window* yang sedang di proses. Penghitungan nilai ini menggunakan metode *integral image*. *Integral Image* pada lokasi x,y merupakan jumlah dari piksel-piksel mulai dari atas sampai piksel sebelah kiri dari x,y. Perhitungan integral proyeksinya dapat di proses dengan rumus di bawah ini :

$$ii(x,y) = \sum_{x' \le x, y' \le y} i(x',y') \quad \text{..... (2)}$$

Di mana : $ii(x,y)$ adalah total *integral image* pada nilai tiap piksel pada daerah yang diarsir pada titik i(x,y).

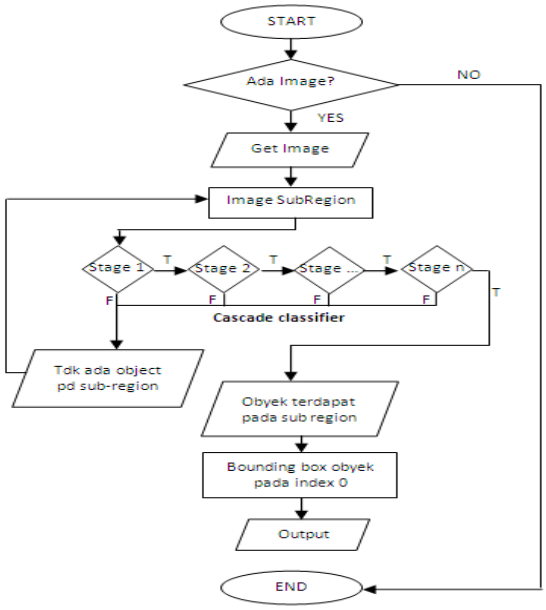
$i(x',y')$ adalah *integral image* pada tiap piksel pada daerah yang diproses.

Gambar 2 menjelaskan *integral image* pada nilai tiap piksel pada daerah yang diarsir pada titik i (x,y).



Gambar 2. *Integral image* pada titik i.

Diagram Proses Pendeteksian Obyek adalah sebagai berikut.



Gambar 3. Diagram deteksi obyek

Deteksi Pupil

Untuk mendapatkan nilai titik tengah bola mata, maka terlebih dahulu dilakukan proses *tresholding* pada bagian mata yang telah dicrop. Pada umumnya proses *tresholding* dilakukan setelah mengubah image RGB menjadi *image gray value*. Namun pada penelitian ini proses *tresholding* dilakukan langsung pada image RGB, hal ini dimaksudkan agar proses segmentasi warna pada area mata dapat terbagi dengan baik**,** seperti dalam penelitian Pamuji (2007).

Gambar 4. Perbedaan *treshold* warna dan *grayscale*

Tahap selanjutnya adalah mengubah hasil *treshold* RGB ini ke bentuk biner dengan membuang semua warna kecuali warna hitam, sehingga diperoleh tampilan seperti berikut :

Gambar 5. *Image biner*

Tahap ini merupakan tahap terakhir dalam menentukan titik tengah bola mata hitam. Perhitungan yang dilakukan untuk mendapatkan nilai titik tengah tersebut adalah dengan integral proyeksi yang sedikit dimodifikasi, yaitu mencari titik berat dari piksel-piksel yang berwarna hitam. Traking posisi ini dilakukan dengan menggunakan rumus berikut :

$$W_x = \frac{\sum_{x=0}^{k} x.p_x}{jp} \quad \text{dan} \quad W_y = \frac{\sum_{y=0}^{k} y.p_y}{jp} \qquad \text{.....(3)}$$

Dimana :

Wx , Wy adalah koordinat x, y pupil
x, y posisi x & y yang sedang dihitung
px, py jumlah piksel pada x & y
jp jumlah seluruh piksel pada *image* target

Pemetaan Koordinat *Pointer*

Penelitian Pamuji (2007) menyatakan bahwa resolusi area mata yang ditangkap kamera lebih kecil jika dibandingkan dengan resolusi layar komputer. Untuk itu perlu dilakukan pemetaan posisi *pointer* agar dapat mencakup seluruh area layar. Penskalaan tersebut dapat dilakukan dengan rumus berikut ini :

$$xl = \frac{rl}{rk}.xt \text{ dan } yl = \frac{rl}{rk}.yt \quad \text{.....(4)}$$

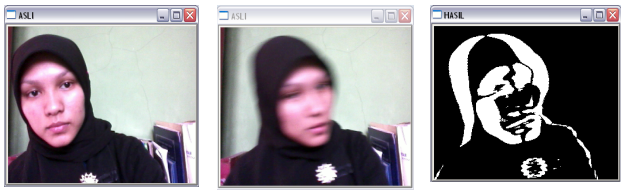
Dimana :

xl, yl	koordinat *pointer* pada layar
xt, yt	koordinat titik tengah mata hasil *tracking*
rl	resolusi layar
rk	resolusi area mata yang ditangkap kamera

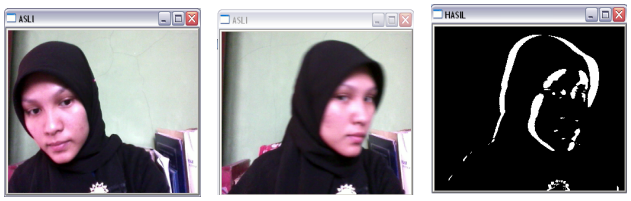
HASIL DAN PEMBAHASAN

Deteksi Pergerakan

Analisa deteksi pergerakan obyek sangat penting untuk mengetahui kecepatan gerakan obyek, apakah kecepatan mencukupi kondisi *real-time* pada *virtual pointer* atau tidak. Hasilnya adalah sebagai berikut:



Gambar 6. Deteksi pergerakan dengan *mean* = 30



Gambar 7. Deteksi pergerakan dengan *mean* = 70

Berikut adalah tabel perbandingan deteksi pergerakan dengan nilai mean yang berbeda:

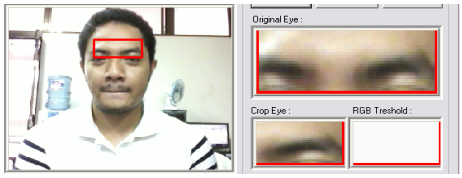
Tabel 1. Hasil Uji Deteksi Pergerakan terhadap Nilai Mean

Nilai Mean	Pergerakan yang terjadi	Presentase Deteksi Gerak	Selisih
10	12,2%	12,0%	0,2%
20	33,0%	21,7%	11,3%
30	34,1%	16,0%	18,1%
40	33,4%	12,5%	20,9%
50	35,2%	10,7%	24,5%
60	33,4%	6,9%	26,5%
70	32,1%	5,4%	26,7%
80	31,9%	3,8%	28,1%
90	36,4%	4,3%	32,1%
100	42,5%	5,2%	37,3%

Deteksi pergerakan yang diterapkan pada aplikasi ini dirancang untuk dapat diatur sesuai keinginan pengguna. Dari hasil uji coba yang dilakukan terhadap bagian deteksi pergerakan, nilai mean pembanding yang optimum pada kondisi cahaya normal adalah antara 5-10.

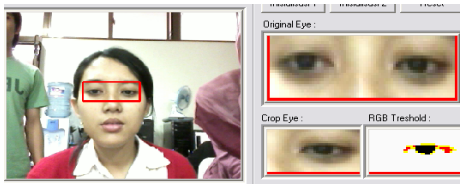
Object detection

Gambar 8 adalah hasil pendeteksian obyek yang salah :



Gambar 8. Hasil deteksi obyek yang salah

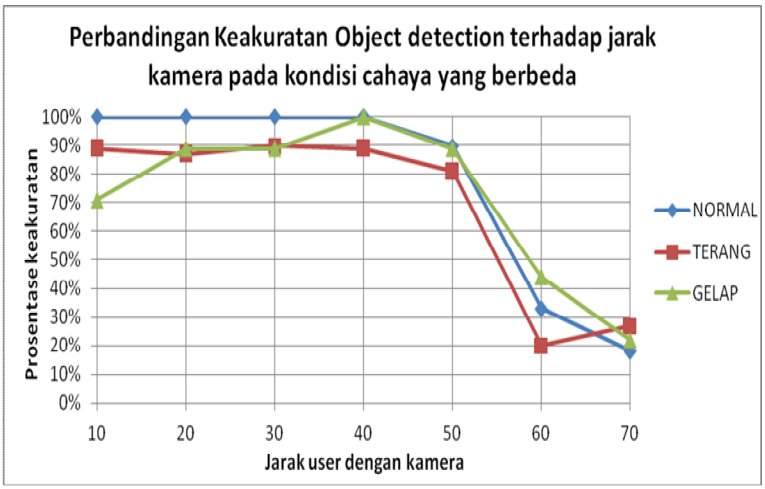
Gambar 9 yang berikut ini adalah hasil pendeteksian obyek yang benar :



Gambar 9. Hasil deteksi obyek yang benar

Dari 10 orang pengguna, 3 orang tidak dapat terdeteksi dengan baik, namun pada 7 orang lainnya obyek bisa terdeteksi dengan baik. Dari penilaian ini maka keakuratan pendeteksian mata pada sistem ini adalah sebesar 70%. Hal ini bisa diperbaiki dengan melakukan *training* ulang dengan menambah sampel serta jumlah stage pada *cascade classifier*.

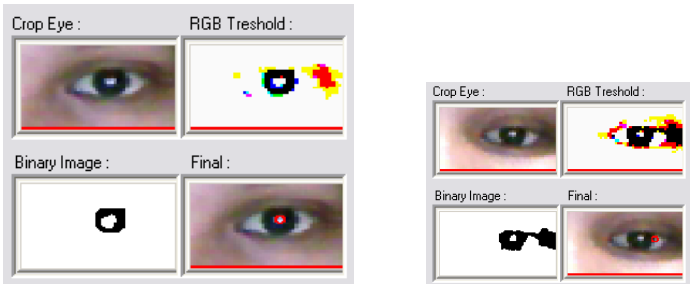
Selain pengguna, tolok ukur dari keberhasilan deteksi obyek adalah jarak kamera dengan pengguna. Berikut ini adalah grafik yang menunjukkan perbandingan keakuratan deteksi obyek berdasarkan jarak kamera.



Gambar 10. Grafik perbandingan keakuratan deteksi obyek terhadap jarak kamera dan cahaya.

Pupil detection

Parameter yang digunakan untuk menguji pendeteksian pupil adalah nilai *treshold* RGB yang di*setting* oleh pengguna. Di bawah ini merupakan contoh dari pendeteksian pupil yang benar dan yang menyimpang.



Gambar 11. (a) *treshold* = 60 (b) *treshold* =100

Dari hasil uji tersebut nilai optimum *treshold* pada cahaya normal adalah 60.

Pemetaan *pointer*

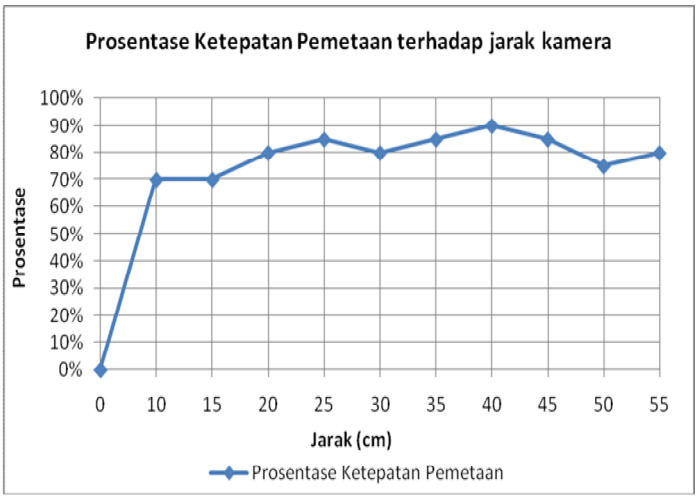
Bagian paling akhir dari proses sistem ini adalah pemetaan *pointer* dari titik tengah mata yang sudah terdeteksi terhadap layar monitor. Pengujian bagian ini menggunakan parameter jarak kamera dengan mata pengguna. Hal yang paling penting pada tahap ini adalah proses inisialisasi posisi mata terhadap layar monitor. Hal ini dimaksudkan untuk mendapatkan nilai panjang gerakan mata terhadap pergerakan pointer di layar monitor.

Tabel 2 menampilkan perbandingan ketepatan pemetaan *pointer* terhadap jarak kamera.

Tabel 2. Perbandingan ketepatan pemetaan *pointer* pada layar monitor dengan resolusi 1280x800

Jarak kamera	Panjang gerakan mata lebar (x)	Panjang gerakan mata tinggi (y)	%
10 cm	55	30	70 %
15 cm	55	27	70 %
20 cm	54	20	80 %
25 cm	53	22	85 %
30 cm	51	18	80 %
35 cm	46	27	85 %
40 cm	39	27	90 %
45 cm	34	17	85 %
50 cm	32	16	75 %
55 cm	27	18	80 %

Dari Tabel 2 terlihat akurasi ketepatan pemetaan *pointer* terhadap jarak kamera yang paling baik terdapat pada jarak 40 cm dari kamera, dan pada jarak 10 cm-15 cm tingkat akurasinya hanya 70%.



Gambar 12. Grafik perbandingan ketepatan pemetaan *pointer* pada layar monitor dengan resolusi 1280x800

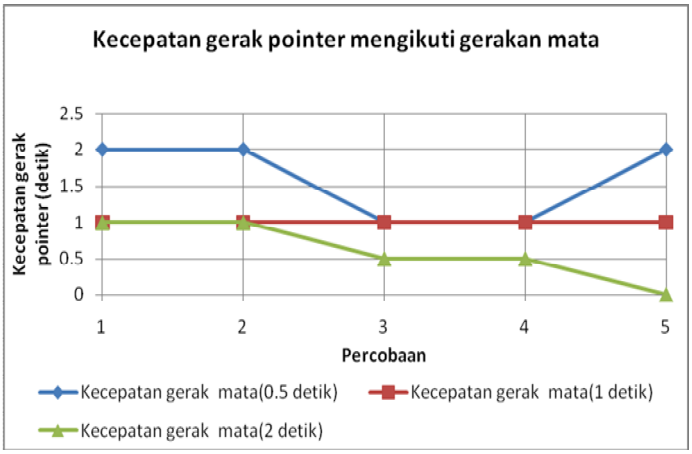
Dari gambar 12 dapat diambil kesimpulan bahwa rata-rata prosentase ketepatan atau keakuratan pemetaan *pointer* adalah sebesar 80%. Pemetaan *pointer* ini bisa bekerja dengan baik jika inisialisasi yang dilakukan baik pula. Jarak optimum yang menghasilkan pemetaan yang baik adalah pada jarak kamera antara 35 cm sampai 45 cm.

Untuk perbandingan seberapa cepat *pointer* mengikuti gerakan mata ditunjukan pada Tabel 3.

Tabel 3. Perbandingan kecepatan gerak *pointer* mengikuti gerakan mata

Kecepatan gerak mata (detik)	Jumlah frame tertangkap	Lama pergerakan pointer (detik)
0,5	5	2
0,5	5	2
0,5	5	1
0,5	4	1
0,5	6	2
1	11	1
1	10	1
1	9	1
1	9	1
1	10	1
2	18	1
2	19	1
2	19	0,5
2	20	0,5
2	21	0

Dari Tabel 3 dapat dibuat grafik seperti pada Gambar 13 berikut ini:



Gambar 13. Perbandingan kecepatan gerak *pointer* mengikuti gerakan mata

Dari Tabel 3 dapat diambil kesimpulan bahwa rata-rata prosentase ketepatan atau keakuratan pemetaan *pointer* adalah 80%. Pemetaan pointer ini bisa bekerja dengan baik jika inisialisasi yang dilakukan baik pula. Jarak optimum yang menghasilkan pemetaan yang baik adalah pada jarak kamera antara 35 cm sampai 45 cm.

Untuk kecepatan rata-rata gerak *pointer* mengikuti gerakan mata berdasarkan hasil uji coba pada Gambar 13 adalah sekitar 1,06 detik. Pergerakan mata yang bergerak secepat 0,5 detik dapat diikuti oleh *pointer* dengan lama waktu pergerakan kurang lebih 1,6 detik, untuk pergerakan mata secepat 1 detik, pointer mampu mengikutinya dengan proses pergerakan rata-rata 1 detik. Sedangkan untuk pergerakan mata yang bergerak dengan kecepatan gerak 2 detik, rata-rata proses *pointer* untuk mengikutinya adalah 0,6 detik.

Dari penjabaran ini, bisa kita ketahui bahwa pergerakan *pointer* akan semakin *real-time* jika gerakan mata semakin lambat. Sebenarnya proses ini pun sangat terkait erat dengan rata-rata jumlah *frame* yang ditangkap kamera tiap detiknya. Jika *frame* per *second* (Fps) suatu kamera tinggi, maka pergerakan *pointer* akan semakin *real-time* dalam mengikuti pergerakan mata.

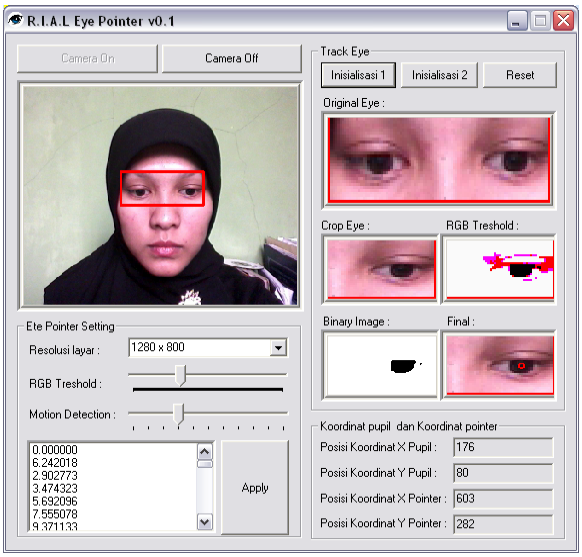
Analisa Keseluruhan Sistem

Pengujian keseluruhan sistem dilakukan dengan kondisi cahaya normal. Nilai optimum dari parameter-parameter terhadap keseluruhan sistem adalah sebagai berikut :

- Kondisi cahaya = normal, dengan asumsi pencahayaan ruangan harus cukup atau tidak gelap.
- Jarak kamera dengan mata = 40 cm, sesuai dengan Tabel 1, perbandingan ketepatan pemetaan *pointer* pada layar monitor dengan resolusi 1280x800.
- RGB batas ambang = 60, sesuai hasil uji nilai optimum *treshold* pada cahaya normal.
- Kecepatan gerakan bola mata = 2 detik, rata-rata proses *pointer* untuk mengikutinya adalah 0,6 detik.

Jika kondisi–kondisi parameter seperti yang telah dijelaskan sebelumnya dipenuhi, maka hasil akhir dari aplikasi *Eye Pointer* ini adalah pergerakan *mouse pointer* pada sistem operasi serta dapat berjalan di atas aplikasi apapun.

Apabila inisialisasi telah dilakukan namun tidak ada obyek yang terdeteksi, maka penggerak *pointer* secara otomatis pindah pada *mouse* yang tersedia. Namun apabila obyek (mata) terdeteksi, maka *pointer* secara otomatis akan digerakan oleh arah gerakan bola mata. Untuk inisialisasi panjang gerakan mata terhadap layar, dapat dilakukan pengguna secara berulang kali, sesuai keinginan pengguna, apabila pergerakan sudah dianggap kurang akurat. Hasil program secara keseluruhan seperti pada Gambar 14.



Gambar 14 . Program Pengendali *Pointer*

KESIMPULAN

Berdasarkan pengujian dan analisa pada sistem *virtual pointer*, maka dapat diambil beberapa kesimpulan. Sistem *virtual pointer* yang dikembangkan mampu mengendalikan *mouse pointer* dengan baik sesuai dengan arah pergerakan mata dengan prosentase keakuratan sebesar 80%.

Semakin kecil nilai *mean*, maka deteksi pergerakan akan semakin sensitif. Sedangkan nilai *mean* yang terlalu besar dapat mengakibatkan gerakan susah dideteksi bahkan dapat tidak terdeteksi sama sekali. Nilai *mean* pembanding untuk deteksi pergerakan obyek dalam area kamera yang optimal adalah pada nilai antara 5 sampai 10.

Semakin besar jumlah data *training* serta jumlah *stage* pada *cascade classifier* pada *object detection* (*Haar Cascade*), maka semakin akurat pendeteksian yang dilakukan, namun hal ini akan memakan waktu proses yang sangat lama. Jarak optimum untuk pendeteksian obyek adalah 40 cm dari kamera. Semakin jauh posisi pengguna terhadap kamera, maka pendeteksian akan semakin sulit. Sistem mampu menangkap obyek yang bergerak dengan baik pada kecepatan kurang 1-2 detik. Nilai optimum *treshold* pada kondisi cahaya normal adalah 60, untuk kondisi cahaya terang nilai *treshold* optimumnya adalah 90, sedangkan untuk kondisi cahaya gelap nilai *treshold* optimumnya adalah 40.

REFERENSI

Dian,S.K.& Satriyanto,E. (2006). *Tugas akhir vrtual pointer sebagai media presentasi*. Politeknik Elektronika Negri Surabaya.

Irawan, M. & Satriyanto, E. (2008). Virtual pointer untuk identifikasi isyarat tangan sebagai pengendali gerakan robot secara real-time. *Jurnal Informatika*,9. Petra Christian Universitas.

J.J. Magee, Scott, M.R. Waber, B.N. & Betke. M. E. (2008). *A real-time vision interface based on gaze detection from a low-grade video camera:* Proceedings of the IEEE Workshop on Real-Time Vision for Human-Computer Interaction (RTV4HCI).

Pamuji (2007).*Tugas akhir permainan shooter dengan kendali mata sebagai penggerak arah senjata.* Politeknik Elektronika Negeri Surabaya.