BAB 2 LANDASAN TEORI

Pada bab ini menjelaskan mengenai berbagai teori yang digunakan untuk melakukan penelitian ini. Bab ini terdiri dari penjelasan mengenai penghitung pengunjung, lalu penjelasan mengenai haar-like features dan gentle adaboost yang digunakan dalam proses pelatihan datanya. Selanjutnya juga dijelaskan mengenai nilai jarak *Euclidian* yang digunakan pada penjejakan objek kepala.

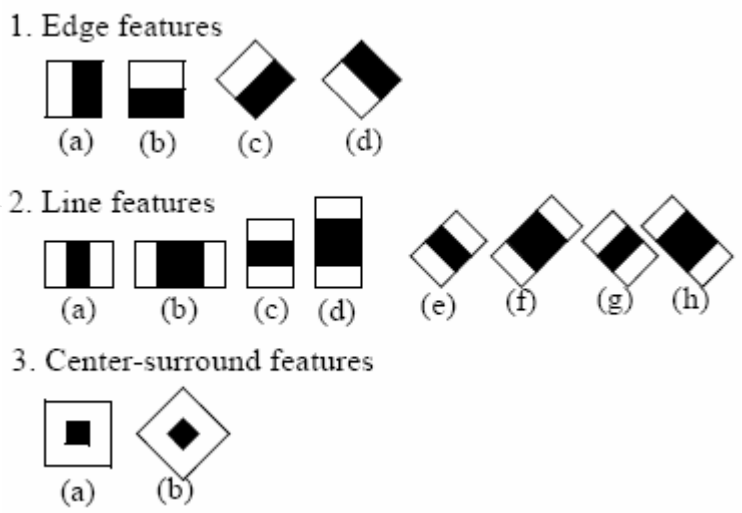
2.1 PENGHITUNG PENGUNJUNG

Penghitungan pengunjung secara otomatis terbukti mampu menghasilkan penghitungan yang memuaskan dan mempermudah pemilik pusat keramaian [15]. Operator pusat keramaian tidak perlu disibukkan lagi oleh hal-hal monoton yang tidak seharusnya dilakukan manusia.

2.2 HAAR-LIKE FEATURES

Pemrosesan data gambar dengan intensitas gambar (piksel per piksel dengan aturan RGB) membutuhkan resource yang mahal, baik waktu maupun komputasi. Papageorgiou et. al.[11] telah membahas mengenai penggunaan fitur gambar saat pemrosesan daripada intensitas gambar. Suatu set dari fitur ini mempertimbangkan wilayah atau *region* persegi panjang dari gambar dan menjumlahkan tiap piksel pada *region* ini. Hasil penjumlahan tersebut digunakan untuk mengkategorisasikan gambar-gambar [5].

Setiap *haar-like features* terdiri dari gabungan kotak-kotak hitam dan putih[7, 12]:



Gambar 2. 1 Satu Set *Extended Haar-like Features* (telah diolah kembali) [7, 16, 17]

1. Bagaimana Menghitung *Haar-like Features*

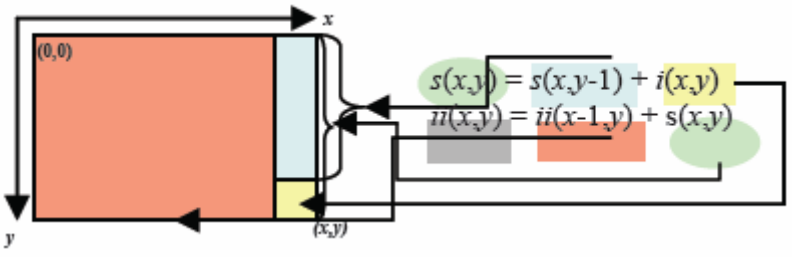
Nilai dari *haar-like features* adalah perbedaan antara jumlah nilai-nilai piksel level abu-abu ke dalam daerah kotak hitam dan daerah kotak putih. [7, 16, 17].

$$f(x) = Sum_{blackrectangle}(pixelraylevel) - Sum_{whiterectangle}(pixelraylevel)$$

Kotak *rectangular haar-like features* dapat dihitung secara cepat menggunakan *integral image*. Integral image pada lokasi x,y terdiri dari jumlah nilai piksel diatas dan dikiri dari lokasi x,y[7, 16, 17].

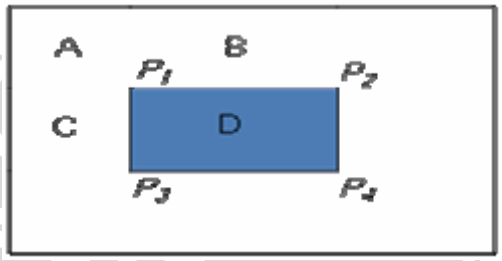
Gambar 2. 2 *Integral Image* (telah diolah kembali) [7, 16, 17]

$$P(x,y) = \sum_{x' \le x, y' \le y} i(x', y') \tag{2.1}$$



Gambar 2. 3 Perhitungan *Integral Image* (telah diolah kembali) [7, 16, 17]

Pada gambar diatas, $i(x,y)$ adalah nilai piksel dari image pada posisi x,y. $s(x,y)$ adalah kumulatif jumlah kolom , Kita dapat menghitung integral image dengan sekali jalan (single pass).



Gambar 2. 4 Perhitungan *Integral Image* (telah diolah kembali) [7, 16, 17]

Dengan menggunakan integral image, nilai jumlah piksel *rectangular* dapat dihitung dalam waktu yang konstan. Sebagai contoh jumlah nilai piksel di dlam kotak D, dapat dihitung sebagai berikut:

$$ii(P4) + ii(P1) - ii(P2) - ii(P3) \quad (2.2)$$

Berikut adalah contoh bagaimana menghitung *haar-like features*:

Misalkan sebuah citra memiliki nilai *grayscale* sebagai berikut:

i(x,y)

2	5	7
11	20	4
6	5	3

Maka untuk menghitung *integral image*-nya dapat dilakukan dengan menggunakan rumus:

$$ii(x,y) = ii(x-1,y) + s(x,y) \quad (2.3)$$

$$s(x,y) = s(x-1,y) + i(x,y) \quad (2.4)$$

Maka dengan rumus tersebut akan didapatkan

s(x,y)

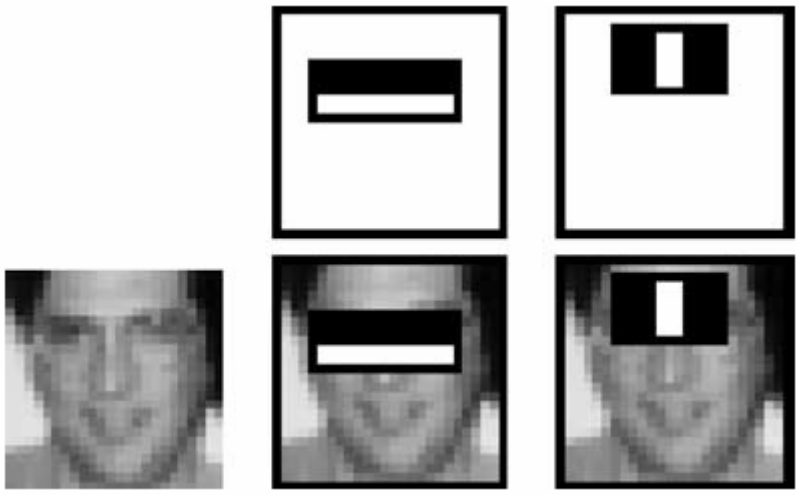
2	5	7

13	25	11
19	30	14

Dan akan menghasilkan *integral image* berupa:

ii(x,y)		
2	5	14
13	38	49
19	49	63

Berikut adalah contoh ekstraksi *haar-like features* pada wajah manusia oleh Viola[17].



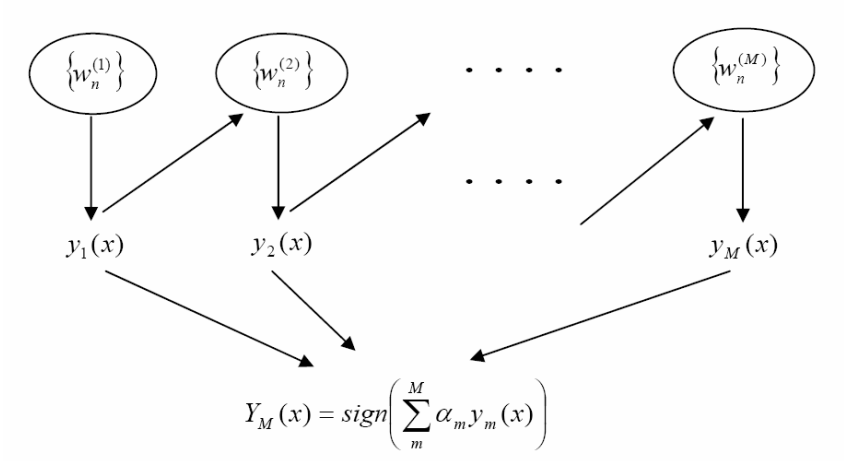
Gambar 2. 5 Contoh Hasil Ekstraksi *Haar-like Features* pada wajah manusia (telah diolah kembali)[17].

2.3 BOOSTING

Algoritma yang digunakan untuk melakukan pelatihan data dalam mengenali pola kepala pengunjung tampak atas adalah *boosting.* Lienhart[7] mendefiniskan sekumpulan *classifier* seperti sebuah pohon keputusan dimana setiap langkah atau level keputusan dilatih untuk mendeteksi hampir semua bagian pada objek dan menolak objek yang tidak memenuhi kriteria.

Boosting merupakan teknik yang ampuh untuk mengkombinasikan banyak *classifier* untuk membentuk suatu gabungan yang performanya lebih baik dibanding performa tiap *classifier* dasar tersebut[3, 4, 15]. Bentuk *boosting* yang banyak digunakan adalah *AdaBoost,* singkatan dari *adaptive boosting*, yang dikembangkan oleh Freund dan Schapire[4]. Performa boosting dapat menghasilkan klasifikasi yang bagus, meskipun tiap *classifier* dasar-nya hanya

sedikit lebih bagus daripada algoritma *random*[1]. Satuan *classifier* ini dapat disebut *weak learner*.



Gambar 2. 6 Algoritma Boosting (telah diolah kembali) [1]

Pada gambar, setiap *classifier* $y_m(x)$ dilatih dan diberi bobot, dimana bobot-bobonya $w_n^{(m)}$ bergantung pada performa dari *classifier* dasar sebelumnya $y_{m-1}(x)$. Ketika semua *classifier* dasar sudah dilatih, semuanya akan dikombinasikan untuk menghasilkan *classifier* akhir $Y_M(x)$.

Algoritma *AdaBoost:*

1. Inisiasi data koefisien bobot $\{w_n\}$ dengan menset $w_n(1) = 1/N$ for n = 1, …, N.
2. For m=1,…,M:

   (a) Cocokkan *classifier* ym(x) dengan data pelatihan dengan meminimalisasi bobot fungsi eror

$$J_m = \sum_{n=1}^{N} w_n^{(m)} I(y_m(x_n) \neq t_n \tag{2.5}$$

   Dimana $I(y_m(x_n) \neq t_n)$ sebagai fungsi indicator dan akan bernilai 1 ketika $Y_m(x_n) \neq t_n$ dan 0 sebaliknya.

   (b) Evaluasi kuantitas

$$\varepsilon_m = \frac{\sum_{n=1}^{N} w_n^{(m)} I(y_m(x_n) \neq t_n)}{\sum_{n=1}^{N} w_n^{(m)}} \tag{2.6}$$

   Lalu gunakan ini untuk mengevaluasi

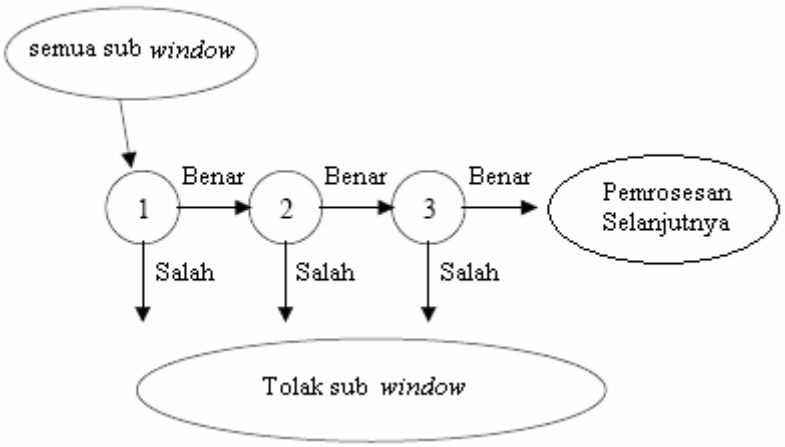
$$\alpha_m = \ln\left\{\frac{1-\varepsilon_m}{\varepsilon_m}\right\} \quad (2.8)$$

(c) Perbaharui koefisien-koefisien bobot

$$W_n^{(m+1)} = w_n^{(m)} \exp\{\alpha_m I(y_m(x_n) \neq t_n)\} \quad (2.9)$$

3. Buat prediksi dengan menggunakan model final, yaitu

$$Y_M(x) = sign\left(\sum_{m=1}^{M} \alpha_m y_m(x)\right) \quad (2.10)$$



Gambar 2. 7 Contoh Proses *learning* atau pelatihan dengan 3 tahap oleh Viola[16] (telah diolah kembali).

Sesuai dengan yang disebutkan pada latar belakang, bahwa penelitian Kuranov[6] menyimpulkan bahwa *Gentle Adaboost* adalah algoritma boosting yang paling baik, maka algoritma Adaboost yang akan digunakan dalam proses pelatihan adalah *Gentle Adaboost.*

2.3.1 Gentle Adaboost

Gentle Adaboost menghasilkan performa yang lebih baik dari *Discreet Adaboost* dan *Real Adaboost* dari sisi akurasi untuk proses deteksi karena membutuhkan komputasi yang lebih ringan[6].

Algoritma Gentle Adaboost:

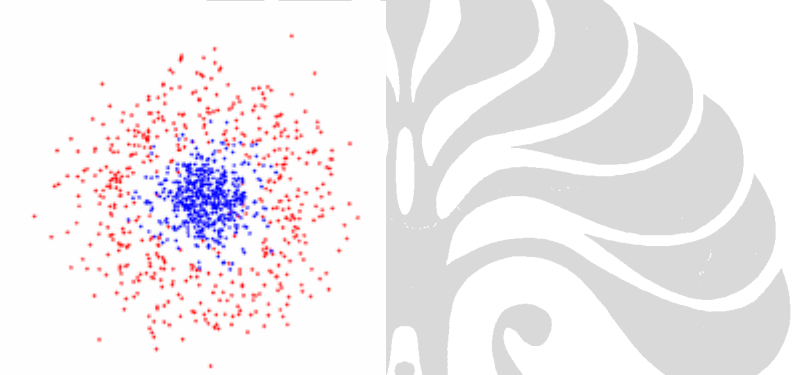
1. Terdapat N contoh $(x_1, y_1),...,(x_N, y_N)$ dengan $x \in R^k, y_i \in \{-1,1\}$
2. Mulai dari bobot w = 1/N, i=1, …, N. (2.11)

3. Ulangi untuk m=1, …, M
4. Hasil dari *classifier* adalah $sign\left[\sum_{m=1}^{M} f_m(x)\right]$ (2.12)

2.3.1 Bagaimana Adaboost Bekerja

Berikut adalah bagaimana Adaboost bekerja yang dijelaskan oleh Sochman[14]. Sesuai algoritma Adaboost, cara kerja adaboost adalah sebagai berikut:

1. Terdapat: $(x_1, y_1),...,(x_m, y_m); x_i \in x, y_i \in \{-1,+1\}$

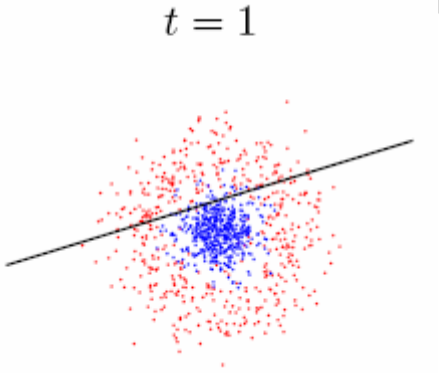


Gambar 2. 8 Adaboost Bekerja (telah diolah kembali) [14]

2. Inisiasi bobot $D_1(i) = 1/m$

   Untuk $t = 1,...,T$:

   Cari $h_t = \arg\min_{h_j \in \mathcal{H}} \epsilon_j = \sum_{i=1}^{m} D_t(i)[y_i \neq h_j(x_i)]$ (2.13)



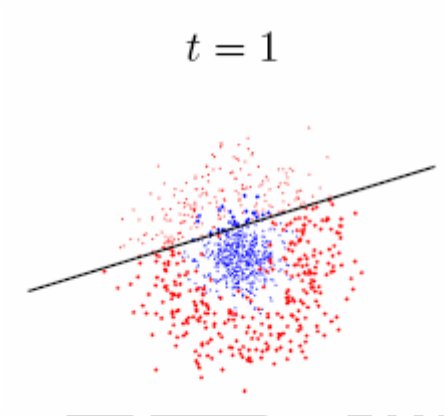
Gambar 2. 9 Adaboost Bekerja (telah diolah kembali) [14]

3. Jika $\epsilon_t \geq 1/2$ maka berhenti

Set $\alpha_t = \frac{1}{2}\log(\frac{1-\epsilon_t}{\epsilon_t})$ (2.14)

Update nilai $D_{t+1}(i) = \frac{D_t(i)\exp(-\alpha_t y_i h_t(x_i))}{Z_t}$ (2.15)

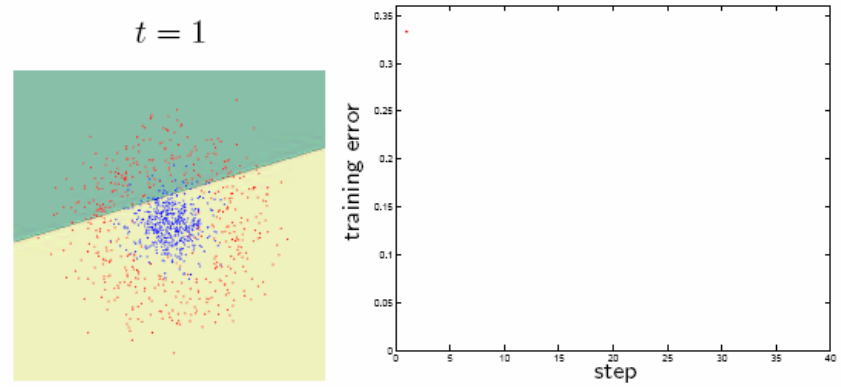
Z_t adalah faktor normalisasi.



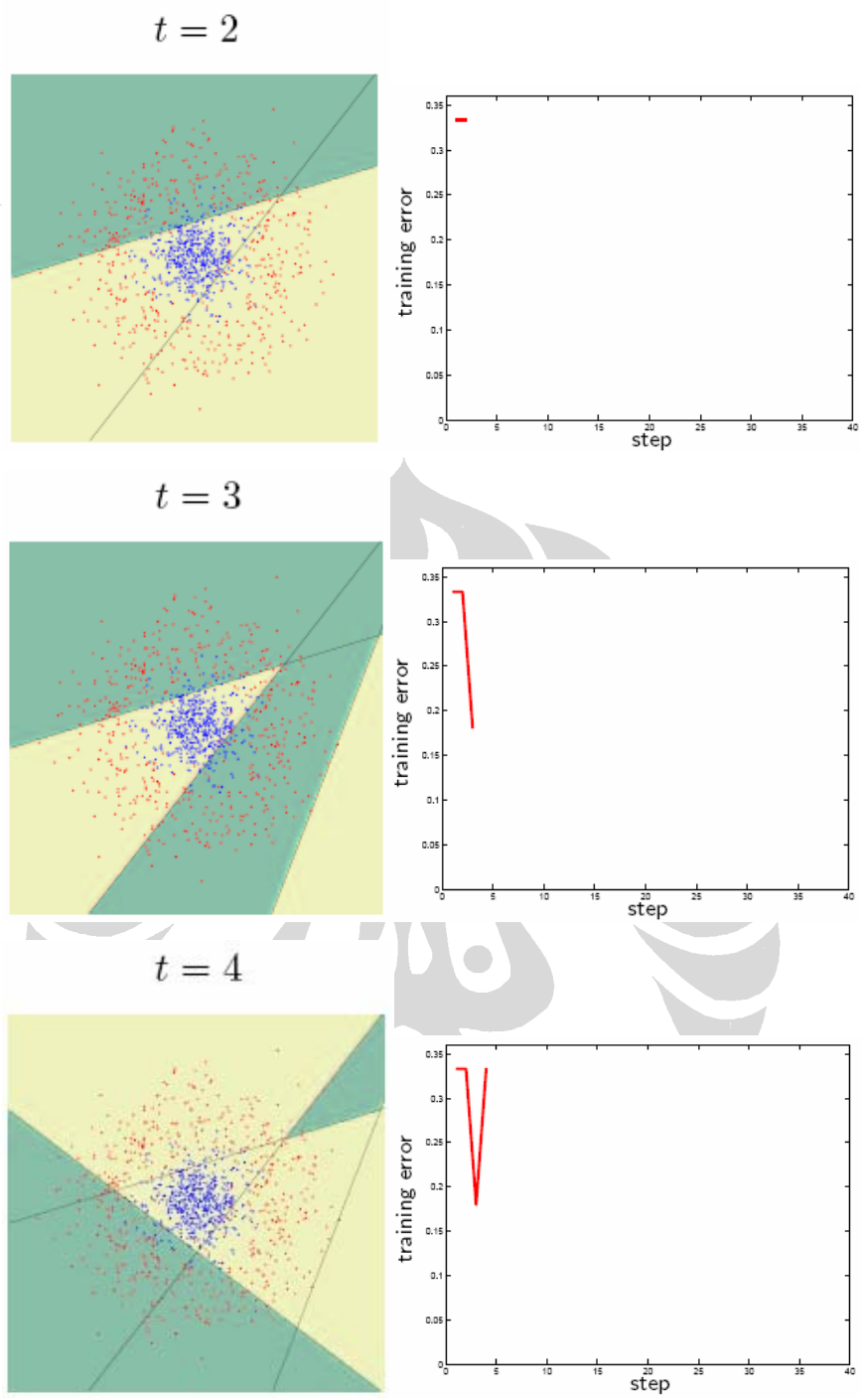
Gambar 2. 10 Adaboost Bekerja (telah diolah kembali) [14]

4. Hasil dari *classifier* akhir

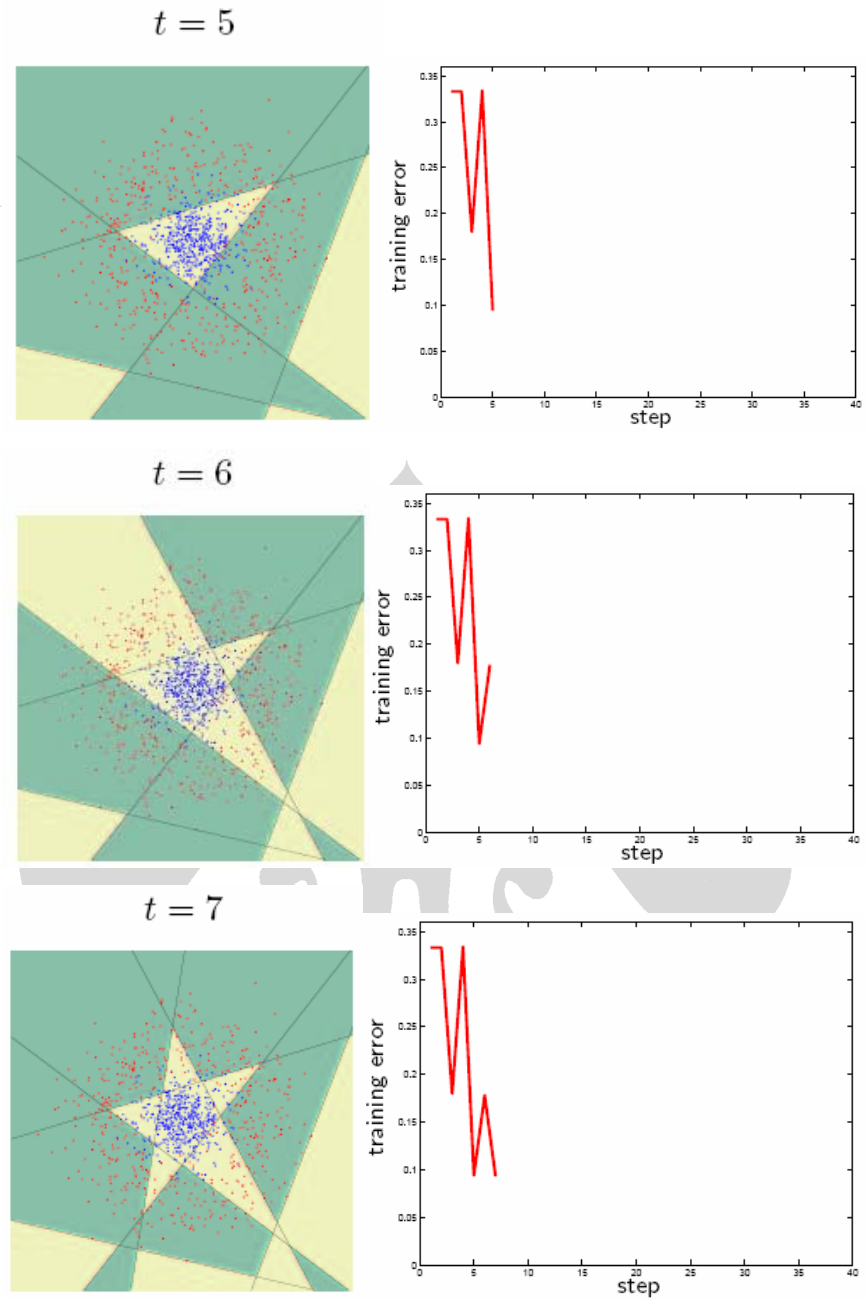
$$H(x) = sign\left(\sum_{t=1}^{T}\alpha_t h_t(x)\right) \quad (2.16)$$



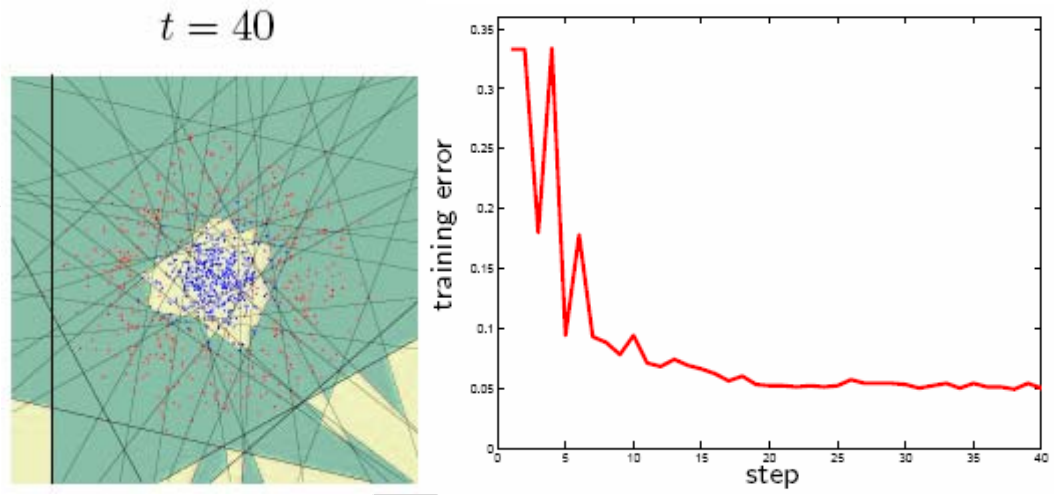
Gambar 2. 11 Adaboost Bekerja (telah diolah kembali) [14]



Gambar 2. 12 Adaboost Bekerja (telah diolah kembali) [14]



Gambar 2. 13 Adaboost Bekerja (telah diolah kembali) [14]



Gambar 2. 14 Adaboost Bekerja (telah diolah kembali) [14]

2.4 NILAI JARAK *EUCLIDIAN*

Penghitungan pengunjung menggunakan jarak *euclidian* terbukti lebih cepat dan menghasilkan penghitungan yang relatif memuaskan[15].

Metode jarak *euclidian* akan digunakan dalam proses *tracking.* Jarak *Euclidian* adalah jarak terpendek antara dua buah titik. Jika terdapat dua buah titik, maka jarak terpendek tersebut didapatkan dengan cara menarik garis lurus yang menghubungkan kedua titik tersebut. Dalam ruang Euclidian berdimensi n, R^n, jarak antara titik x dan y dapat dirumuskan sebagai berikut [2, 15]:

$$D = |x-y| = \sqrt{\sum_{i=1}^{n} |x_i - y_i|^2} \quad (2.17)$$

Dimana *n* adalah jumlah titik dalam R^n.

Karena sistem yang dikembangkan bekerja dalam ruang *Euclidian* dengan dimensi dua, maka jarak *euclidian* antara dua titik p(x1,y1) dan q(x2,y2) dapat dihitung dengan persamaan sebagai berikut[12]:

$$D(p,q) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \quad (2.18)$$

Jarak Euclidian(jarak antar objek pada citra) maksimum yang ditoleransi dalam berbagai video dapat beragam, tergantung dari letak kamera[15].