People Counting menggunakan *Extended* CAMSHIFT dan Fitur *Haar-like*

People Counting using Extended CAMSHIFT and Haar-like Features

[1]Bobby Yuliandra **[2]Tjokorda Agung Budi W, ST., MT.**
[1,2]Fakultas Informatika – Telkom University
Jl. Telekomunikasi, Dayeuh Kolot Bandung 40257 Indonesia
[1]bobby.yuliandra@gmail.com [2]cokagung2001@gmail.com

ABSTRAK

Info jumlah orang yang masuk dalam suatu wilayah mempunyai info penting untuk beberapa tujuan seperti bisnis, keamanan, kebutuhan sumber daya yang harus dicukupi, dan lainnya. Ketika jumlah orang yang masuk terlalu banyak sehingga mengurangi akurasi perhitungan secara manual, maka keadaan tersebut membutuhkan sistem untuk menghitung secara otomatis dan *people counting* bisa melakukan tugas tersebut.

Perkembangan *computer vision* mampu memberikan akurasi yang baik untuk diterapkan pada *people counting*. Penggunaan metode *Face Detection* menggunakan haar-cascade *classifier* yang bisa menghasilkan akurasi 94%, *tracking* menggunakan *extended* CAMSHIFT yang bisa menanggulangi tabrakan *tracking* window dengan warna *background* yang sama dengan akurasi 88%. Metode yang bisa diterapkan ke dalam *people counting.*

Kata Kunci: *Computer Vision, People Counting,* Haar-Cascade *Classifier, Extended* CAMSHIFT

ABSTRACT

The information of visitor that entering in an area could be an important information for several purposes such as business, security, resources demand, and etc. While numbers of visitors are too excessive that could reduce accuration of manual counting, then this condition needs a system which could count automatically and people counting could do it.

The developing of computer vision could give good accuration when implementing it in people counting. Using Face Detection with haar-cascade classifier that can give accuration 94%, tracking with extended CAMSHIFT can avoid tracking window collision with the same-color object with accuration 88%. This method can be implemented to people counting.

Keyword: *Computer Vision, People Counting,* Haar-Cascade *Classifier, Extended* CAMSHIFT

1. Pendahuluan

1.1 Latar Belakang

Informasi jumlah pengunjung yang terdapat di suatu tempat memang dibutuhkan. Masalah keamanan, jumlah kepadatan pengunjung, atau masalalah yang lain bisa menjadi bahan acuan pertimbangan pemilik usaha atau tempat dalam mengelola keputusannya. Pengunjung dalam jumlah besar dan dalam aliran waktu yang sangat cepat akan menjadi sangat susah dihitung dengan cara manual dan memunculkan kesalahan perhitungan yang sangat besar. Oleh karena itu dibutuhkan alat secara otomatis menghitung jumlah pengunjung yang datang yang disebut *people counting.*

People counting menggunakan beberapa teknologi dalam implementasinya seperti *infrared beam* atau *computer vision.* Walaupun *infrared beam* mayoritas digunakan untuk *people counter*, metode ini tidak mempunyai akurasi yang tinggi jika jumlah orang yang melewati perangkatnya terlalu banyak. *Computer vision* menggunakan citra yang ditangkap kamera mampu memberikan akurasi yang lebih tinggi jika dibandigkan menggunakan *infrared beam*. Pengolahan citra yang dilakukan untuk *people counting* salah satunya dengan cara mendeteksi wajah yang ada kemudian wajah yang tertangkap akan dilakukan *tracking.* [1]

Setelah ditemukannya metode *tracking* CAMSHIFT oleh R.Bradsky [3], metode ini mampu lebih adaptif untuk melakukan *tracking* terhadap citra sekuensial (video) yang citranya berubah-ubah setiap urutannya. Metode ini lebih baik dari metode MEANSHIFT [4]. Walaupun CAMSHIFT mempunyai kelebihan seperti *real-time, robust*, mampu beradaptasi, dan lainnya, Lee et al [2] memperbaiki CAMSHIFT yang mudah kelihangan *tracking* ketika objeknya mempunyai warna yang sama dengan *background.* Selain itu Lee et al [2] juga menggunakan fitur Haar dan algoritma Adaboost untuk mempercepat deteksi wajah.

Dengan menggunakan motode dari Lee et al [2] dapat mempercepat dan memperbaiki *people counting* berdasarkan deteksi wajah dan *tracking* wajah. Jadi nantinya diharapkan dengan memakai metode ini akan lebih mampu *tracking* dengan kondisi *background* yang lebih mirip dengan warna kulit.

1.2 Perumusan Masalah

Permasalahan yang biasa terjadi dalam *people counting* antara lain masalah akurasi dan pengaruh background pada citra yang menyebabkan kesalahan pengenalan. *People counting* membutuhkan akurasi yang baik dalam berbagai kondisi. Untuk *people counting* dengan menggunakan pengenalan wajah dan *tracking*, pengenalan wajah dibutuhkan metode yang cepat dan tepat. Untuk *tracking* pun dibutuhkan metode yang baik mampu menghadapi *error* ketika objek bertemu dengan *background* yang warnanya sama.

Pembatasan tugas akhir kali ini perbaikan citra yang dipengaruhi *noise*, pencahayaan yang kurang terhadap warna kulit, dan *blur* pada citra tidak masuk dalam pembuatan tugas akhir kali ini. Selain itu bentuk pose muka yang tidak frontal juga termasuk batasan malasah

1.3 Tujuan

Tujuan tugas akhir kali ini antara lain :

1. Mendeteksi wajah menggunakan fitur *haar-like* dengan algoritma Adaboost
2. Melakukan *tracking* wajah dengan algoritma *Extended* CAMSHIFT
3. Melakukan *people counting* dari hasil *tracking* wajah
4. Analisa parameter yang mempengaruhi untuk menghasilkan akurasi *people counting* yang tinggi.

1.4 Hipotesa

Hipotesa dari tugas akhir ini sebagai berikut :

1. Dengan menggunakan metode ini, muka bisa dideteksi dengan akurasi 94% berdasarkan hasil literatur [2] jika dibandingkan dengan literatur [10].
2. Berdasarkan hasil studi literatur [2], *tracking* menghasilkan akurasi 95,3% pada *frame rate* 30-35 fps (*frame per second*) yang nantinya berdampak pada akurasi *people counting*
3. Metode ini mampu *tracking* objek ketika berdempatan dengan warna *background* yang sama.

1.5 Metodologi Penyelesaian Masalah

1.5.1 Studi Literatur

Pada tahap ini dilakukan studi literatur yang berkaitan dengan tugas akhir ini antara lain :

1. Referensi tentang pengolahan citra
2. Referensi tentang *people counting*
3. Referensi tentang *face detection* menggunakan fitur *haar-like*
4. Referensi tentang *tracking* menggunakan *Extended* CAMSHIFT
5. Referensi tentang *library OpenCV*

1.5.2 Analisa dan Perancangan Sistem

Pada tahap ini dilakukan analisa terhadap perancangan sistem serta bentuk *dataset* yang digunakan untuk percobaan. Untuk perancangan sistem, sistem dipecah menjadi beberapa analisa. Analisa-analisa itu antara lain :

1. Analisa pembagian *imaging process* dan perancangan sistem
2. Analisa *code* dari *library OpenCV*
3. Analisa kerja *face detection* dan *tracking* menggunakan fitur *haar-like* dan *extended* CAMSHIFT
4. Analisa akurasi sistem *people counting* ini dengan berbagai kondisi

1.5.3 Implementasi Sistem

Pada tahap ini akan dilakukan *coding* atau implementasi berdasarkan rancangan yang telah dibuat sebelumnya.

1.5.4 Pengujian Sistem

Pada tahap ini akan dilakukan *testing* atau pengujian terhadap sistem yang telah diimplementasikan apakah sistem mampu mendeteksi dan *tracking* wajah dengan baik sehingga sistem *people counting* ini dapat menghasilkan akurasi yang tinggi. Dalam pengujian ini akan diukur berapa jumlah wajah yang masuk sampai berapa orang yang sudah dihitung sistem dengan berbagai kondisi lingkungan (pencahayaan, letak kamera, dan situasi objek).

1.5.5 Analisa Hasil Pengujian dan Pengambilan Keputusan

Sistem yang *people counting* berdasarkan *face detection* dan *tracking* menggunakan fitur *haar-like* dan *extended* CAMSHIFT ini diuji dengan menggunakan video. Kemudian hasil dari sistem ini akan dianalisa. Analisa difokuskan kepada akurasi yang dihasilkan sistem ini terhadap bentuk lingkungannya (pencahayaan, letak kamera, situasi objek). Hasil dari analisa ini akan didapatkan bentuk lingkungan yang paling baik untuk menghasilkan akurasi *people counting* yang tinggi.

1.5.6 Perumusan Kesimpulan dan Penyusunan Tugas Akhir

Pada tahap ini akan dilakukan perumusan kesimpulan berdasarkan analisa dari hasil implementasi sistem yang telah dilakukan pada tahap sebelumnya. Setelah diambil kesimpulan akan dilakukan penyusunan tugas akhir sesuai aturan-aturan yang telah ditetapkan oleh institusi.

2. Tinjauan Pustaka

2.1 Citra Digital

Benda-benda yang selama ini bisa kita lihat adalah pantulan gelombang cahaya putih ke mata

kita. Setiap benda memantulkan gelombang cahaya dengan panjang yang berbeda-beda. Mata kita hanya bisa merepresentasikan warna dengan panjang gelombang antara 380-780 nanometer[5]. Citra digital adalah representasi warna pada perangkat digital dan disajikan dalam bentuk yang model warna berbeda-beda, antara lain RGB (*red, green, blue*), *grayscale*, HSV(*hue, saturation, value*) dan YCbCr.

2.1.1 RGB (*Red, Green, Blue*)

RGB adalah pemodelan warna yang berasal dari tiga warna dasar yaitu merah (*red*), hijau (*green*), dan biru (*blue*). Setiap warna dasar ini memiliki nilai 8 bit dan *range* nilainya antara 0-255. Jadi ketika mendefinisikan sebuah warna contoh wahna hijau murni, maka nilai RGB nya akan didefinisikan R=0. G=255, B=0 atau ditulis RGB (0,255,0). Model warna ini biasanya terdapat dalam perangkat elektronik yang jika tidak diberi cahaya sama sekali RGB(0,0,0) akan berwarna hitam. Model RGB ini juga disebut warna additif karena jika digabungkan ketiga warna itu maka warna yang akan tampak berwarna putih, RGB(255,255,255).

2.1.2 *Grayscale*

Model warna *grayscale* merepresentasikan warna berdasarkan tingkat keabuannya [5]. Model warna *grayscale* biasanya digunakan untuk mempermudah analisa citra karena citra itu hanya akan mempunyai sebuah warna dengan rentang 0-255.

Sebagai contoh jika kita mempunyai citra RGB akan dikonversikan ke dalam bentuk citra *grayscale*, maka rumusnya sebagai berikut

$$grayscale = 0{,}299 * R + 0{,}587 * G + 0{,}114 * B$$

2.1.3 HSV (*Hue, Saturation, Value*)

Model warna HSV(*Hue, Saturation, Value*) mempunyai beberapa komponen antara lain :

- *Hue* mendefinisikan warna yang sebenarnya dari warna merah hingga warna biru. *Hue* digunakan untuk menentukan warna dasar sebuah warna.
- *Saturation* mendefinisikan kemurnian dari warna *hue* dan warna dasar putih. Sebagai contoh warna kuning dengan tingkat *saturation* 100% maka warna itu akan memberikan warna kuning murni. Jika *saturation* 50% maka warna kuningnya akan memudar sebanyak 50% dan jika *saturation* sebesar 0% maka warna kuning itu akan berwarna putih.
- *Value* mendefinisikan tingkat kecerahan dari model warna HSV. Sebagai contoh warna kuning tadi, jika mempunyai *value* yang lebih rendah maka warna yang dimunculkan akan semakin gelap. Jika sebuah warna mempunyai nilai *value* sebesar 0 maka warna itu didefinisikan sebagai warna hitam atau tidak ada warna sama sekali.

2.1.4 YCbCr

YCbCr adalah panel warna yang dipakai dalam menampilkan citra dari sistem video dan fotografi digital. YCbCr terdiri dari komponen kroma biru (Cb), kroma merah (Cr), dan luma (Y). YCbCr mengadopsi dari panel warna RGB.

Kroma adalah derajat antara warna dan luma. Sebagai contoh kroma biru adalah selisih nilai antara warna biru dengan komponen luma. Warna merah juga begitu. Ketika warna Cr dan Cb sama-sama 0, maka warna yang dihasilkan adalah hijau. Sedangkan ketika sama-sama 255, warna yang dihasilkan adalah magenta.

Luma memberikan *brightness* pada gambar. Ketika pada posisi puncak dan nilai antar kroma sama, maka warna yang dihasilkan berwarna putih. Begitu pula sebaliknya ketika nilai kroma sama tetapi nilai luma kecil maka warna yang dihasilkan warna hitam.

2.2 Hsu Skin Detection

Pengenalan wajah terdiri dari beberapa tahap. Untuk mengenali wajah pertama kali harus dilakukan adalah pendeteksian piksel warna kulit. Dari hasil menentukan piksel yang termasuk golongan warna kulit atau non-warna kulit.

Pendeteksian warna kulit adalah faktor fundamental untuk pendeteksian wajah. Biasanya proses pengenalan warna kulit berlangsung sangat cepat. Namun karena ada pengaruh iluminasi, *background* yang sama dengan warna kulit, atau gangguan lainnya, pendeteksian warna kulit mempunyai tingkat akurasi yang rendah. Akibatnya sering terjadi kesalahan dalam pendeteksian warna kulit.

Salah satu metode pendeteksian warna kulit dilakukan oleh Hsu et al[7]. Hsu menggunakan *skin color correction* sebelum melakuka deteksi warna kulit. Dalam metodenya, Hsu menggunakan model warna YCbCr.

Setelah mengkonversi RGB ke YCbCr, kemudian hasilnya dipakai untuk *skin color correction.*

Skin color correction adalah salah satu versi metode *white patch*. Prinsip dari *skin color correction* ini adalah mencari piksel dengan nilai luminasi yang termasuk top 5% dari nilai seluruh citra [2]. Piksel yang terdeteksi kita anggap sebagai bagian dari objek warna putih. Jika angka top piksel ini melebihi 100 dan artinya seluruh citra bukan warna kulit, maka *color correction* ini dijalankan.

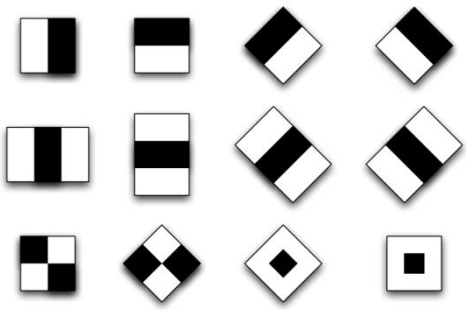
2.3 Deteksi wajah menggunakan fitur *haar-like*

Deteksi wajah banyak digunakan dalam bidang *computer vision.* Salah algoritma yang banyak digunakan adalah algoritm Viola-Jones [6]. Algoritma ini mampu mendeteksi wajah yang terdapat dalam citra atau terdapat dalam *frame* video. Untuk mengerti kerja algoritma ini, kita harus mengerti beberapa konsep penting antara lain :

- Citra integral untuk deteksi fitur *haar-like* untuk perhitungan lebih cepat
- *Classifier* wajah yang dibangun menggunakan algoritma *learning* Adaboost
- Kombinasi dari beberapa *classifier* dalam bentuk *cascade classifier*

Ada beberapa alasan kenapa lebih baik memakai fitur daripada langsung menghitung piksel. Salah satu alasan paling penting adalah sistem berbasis fitur bekerja lebih cepat daripada sistem berbasis piksel [6]. Selain itu fitur bisa berperan meng-*encode ad-hoc domain knowledge* yang susah dipelajari menggunakan jumlah *training* data yang terbatas.

Fitur sederhana yang digunakan mengingatkan kita kepada fungsi basis *haar* yang suah digunakan oleh Papageorgiou et al (1998). Sebagai contoh dua fitur persegi yang terdiri dari satu bagian berwarna gelap dan bagian yang lain berwarna terang. Bagian ini mempunyai ukuran persegi panjang yang sama dan berdekatan satu sama lainnya. Nilai fitur ini diambil dari selisih jumlah antara bagian gelap dan jumlah bagian yang terang. Untuk fitur dengan bentuk yang lain seperti fitur yang menggunakan 3 buah persegi atau dengan kemiringan 45 derajat, perhitungan nilai fitur juga sama yaitu antara selisih jumlah bagian yang gelap dengan jumlah bagian yang terang. Contoh fitur bisa dilihat di gambar berikut.

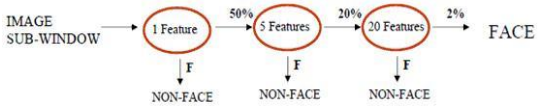


Gambar 2.4 Contoh fitur haar

Fitur *haar* yang dipakai Viola-Jones jumlahnya ada 160.000. Jumlah ini sangat besar dan membutuhkan waktu proses yang lama jika digunakan semuanya. Untuk mempercepat hanya dipilih fitur-fitur tertentu dengan tingkat pembeda yang tinggi antara wajah dan non-wajah. Untuk memilihnya dibutuhkan algoritma Adaboost dan selanjutnya kumpulan fitur-fitur ini disebut *classifier*. Algoritma Adaboost berfungsi untuk memilih fitur-fitur penting yang akan digunakan dalam *classifier* wajah. Algoritma Adaboost ini memiliki beberapa *classifier* dan nantinya *classifier-classifier* itu disusun bertingkat dalam memilih region wajah dan non-wajah. Adaboost disebut juga dengan *machine learning* dan fitur-fitur yang ada disebut juga dengan *learner.*

Jika menggunakan fitur yang termasuk wajah, maka jumlahnya akan lebih banyak ketimbang fitur non-wajah. Oleh karena itu lebih cepat menggunakan fitur non-wajah atau *weak learner* untuk seleksi karena jumlahnya yang sedikit. Tentu dengan hanya menggunakan *weak leraner* yang jumlahnya sedikit ini akan merpercepat proses pemilihan atau pengklasifikasian wajah.

Setelah proses menggunakan algoritma Adaboost tadi, kita mendapatkan beberapa fitur yang cocok dipilih untuk mengklasifikasikan wajah dan non-wajah yang di dalam sebuah *classifier.* Untuk lebih mempercepat komputasi, pengklasifikasian ini dilakukan dengan cara mengkombinasikan beberapa *classifier* secara bertingkat (*cascade classifier*). Contoh alur kerja sebuah *cascade classifier* dengan 3 tingkat (*stage*) bisa dilihat seabgai berikut



Gambar 2.7 Alur kerja algoritma Adaboost

2.4 Algoritma *Extended* CAMSHIFT

Ada beberapa algoritma untuk tracking seperti meanshift, CAMSHIFT, atau *extended* CAMSHIFT [7]. Algoritma meanshift secara luas dipakai untuk *real-time tracking* objek. Meanshift merupakan algoritma mencari rataan dalam distribusi padat peluang dalam distribusi citra (x,y). Hasil dari rataan distribusi padat peluang ini itulah yang nantinya menjadi *centroid* yang digunakan untuk *tracking. Centroid* mengikuti perpindahan gradient objek. Langkah-langkah algoritma ini sebagai berikut :

1. S*earch window W* dipilih dengan ukuran s.
2. *Initial search window* dipilih dengan p_k sebagai *centroid* awal.
3. Hitung *centroid* dalam *search window* itu dengan menggunakan rumus

$$P_k(W) = \frac{1}{|W|}\sum_{j=W}^{\infty}(Pj)$$

   Lalu hitung juga meanshift mengikuti gradient dari *f(p)*

$$P_k(W) - P_k \approx \frac{f'(P_k)}{f(P_k)} \qquad (5)$$

4. Set *centroid search window* yang dihasilkan pada point $P_k(W)$.
5. Ulangi langkah ke-3 dan ke-4 sampai konvergen, yaitu ketika $f'(p) \cong 0$

Kamera video menggunakan distribusi diskrit, oleh sebab itu *centroid* pada *search window* pada rumus (3) dan (4) ditambahkan aturan berikut :

Tentukan *zeroth moment*

$$M_{00} = \sum_x \sum_y I(x,y) \quad (6)$$

Tentukan moment pertama untuk *x* dan *y*

$$M_{10} = \sum_x \sum_y xI(x,y) \quad (7)$$

$$M_{01} = \sum_x \sum_y yI(x,y)$$

Kemudian *centroid* dari *search window* diperoleh melalui rumus

$$Xc = \frac{M_{10}}{M_{00}} \qquad Yc = \frac{M_{01}}{M_{00}} \quad (8)$$

I(x,y) adalah intensitas piksel pada posisi *(x,y) plane* ekstraksi fitur, di mana *x* dan *y* adalah range sepanjang *search window.*

Konvergensi iterasi algoritma meanshift dari distribusi diskrit dapat dirumuskan sebagai berikut.

$$dx = x_c - Px \quad (9)$$

$$dy = y_c - Py$$

$$dx^2 + dy^2 \leq 1 \quad (10)$$

(Px, Py) adalah koordinat pusat dari *search window* yang lama. Rumus (10) berkoresponden dengan konvergensi pada distribusi *Euclidean* yaitu $f'(p) \cong 0$. Berikut ini penjabaran proses algoritma meanshift jika dilakukan terhadap citra :

1. Dapatkan ukuran dari *input search window*
2. Dapatkan lokasi kalkulasi dari *plane* fitur yang berkoresponden dengan ukuran dan lokasi dari *input search window*
3. Tentukan lokasi *centroid* dalam *input search window*
4. Geser atau *shifting input search window* sehingga akan dihasilkan *output search window* yang memiliki posisi *centroid* sesuai dengan hitungan pada langkah ke-3
5. Ulangi langkah ke-3 dan ke-4 sampai *search window* konvergen, yaitu sampai *search window* tersebut berpindah dengan jarak yang kurang dari nilai *threshold* yang sudah ditentukan dalam rumus (10).

Algoritma meanshift didesain untuk distribusi gambar statis. Distribusi gambar yang statis tidak mengalami perubahan nilai piksel setiap saat, padahal saat kondisi *real*, kamera selalu mengalami perubahan warna karena adanya pergerakan objek, *noise*, atau perbedaan lingkungan. Pada saat *tracking*, objek mengalami perubahan ukuran dan lokasi sehingga distribusi probabilitas ikut berubah dalam setiap waktunya. Karena ada perubahan ini maka dibutuhkan algoritma yang dinamis dan adaptif terhadap perubahan. Algoritma CAMSHIFT mengambil probabilitas *hue* sebagai dasar pencarian objek *tracking*.

Ide dari algoritma CAMSHIFT ini didasari dari pencarian *centroid* yang dilakukan oleh algoritma meanshift. Tambahannya adalah algoritma CAMSHIFT ini menggunakan probabilitas dari *hue* untuk proses *tracking.* Warna *hue* didapat dari panel warna HSV. Dalam hal ini wajah yang dideteksi pada bagian sebelumnya mempunyai *hue* yang terdeteksi sebagai warna kulit. Distribusi probabilitas warna kulit ini kita ambil. Setelah itu warna *hue* dipetakan dalam histrogram dan dilihat distribusi probabilitasnya. Probabilitas yang ada dibuah dikembalikan dalam bentuk citra *grayscale* sebagai *backprojection*. Citra *backprojection* itu dipakai untuk *tracking* biasa menggunakan meanshift.

Gambaran lengkap dari CAMSHIFT bisa dilihat dalam proses berikut :

1. Input citra RGB dari *image sequence*
2. Inisialisasi ukuran dan lokasi *search window*
3. Konversi panel warna RGB ke HSV
4. **While** (# iterasi) **do**
5. Hitung distribusi probabilitas warna *hue* yang ada di dalam *search window*
6. Temukan lokasi *centroid* dari *search window*
7. Jalankan algoritma meanshift dan dapatkan ukuran dan lokasi baru dari *search window*
8. Dalam *next frame* dari *sequence image*, inisialisasi ukuran dan lokasi *search window* yang nilainya didapat dari langkah ke-7
9. **Endwhile**
10. pergi ke langkah ke-3

Algoritma CAMSHIFT selalu melakukan perubahan ukuran *search window* dalam setiap iterasinya. CAMSHIFT akan menyesuaikan ukuran dan sudut dari objek setiap kali *search window* berpindah, tidak seperti meanshift. Hal ini terjadi karena dalam citra bergerak, ukuran wajah akan berubah setiap waktunya dan algoritma CAMSHIFT selalu meng-*set* ukuran *search window.*

Perubahan yang terjadi tidak hanya pada ukuran dari *search window* saja, namun juga terjadi pada sudut orientasi objek. Objek yang bergerak bisa berubah ukurannya setiap waktu dan dibutuhkan algoritma yang adaptif untuk bisa melakukan hal ini.

Selama berjalan algoritma ini, pergerakan wajah yang terjadi dengan arah kanan-kiri (sumbu X), atas-bawah (sumbu Y), maju-mundur (sumbu Z), dan patah kanan-kiri (*roll*). Algoritma ini pun butuh rumus untuk pergerakan sudut orientasi dan penskalaan *search window* sebagai berikut :

first moment untuk *x* dan *y*

$$M_{11} = \sum_x \sum_y xyI(x,y) \quad (11)$$

second moment untuk *x* dan *y*

$$M_{20} = \sum_x \sum_y x^2 I(x,y) \quad (12)$$

$$M_{02} = \sum_x \sum_y y^2 I(x,y) \quad (13)$$

Sudut orientasinya wajah ketika melakukan *roll* adalah

$$\theta = \frac{\arctan\left(\frac{2(\frac{M_{11}}{M_{00}} - x_c y_c)}{(\frac{M_{20}}{M_{00}} - x_c^2) - (\frac{M_{02}}{M_{00}} - y_c^2)}\right)}{2} \quad (14)$$

Dengan adanya pergerakan dari wajah pada sumbu x, y, dan z, maka akan terjadi perubahan ukuran *search window.* Ukuran panjang l dan lebar *w search window* dihitung menggunakan rumus :

$$a = \frac{M_{20}}{M_{00}} - x_c^2 \quad (15)$$

$$b = 2(\frac{M_{11}}{M_{00}} - x_c y_c) \quad (16)$$

$$c = \frac{M_{02}}{M_{00}} - y_c^2 \quad (17)$$

$$l = \sqrt{\frac{(a+c) + \sqrt{b^2+(a-c)^2}}{2}} \quad (18)$$

$$w = \sqrt{\frac{(a+c) - \sqrt{b^2+(a-c)^2}}{2}} \quad (19)$$

Lae-Kyoung Lee at el [2] [8] mengembangkan algoritma CAMSHIFT yang telah ada. Metode yang asli yang hanya menggunakan nilai dari komponen *hue* dalam panel warna HSV, tidak cukup untuk mengekpresikan warna objek. Sebagai contoh jika warna objek sama dengan warna *background*, tentu akan sangat mudah kehilangan *tracking* pada objek. Berdasarkan hal kekurangan ini, mereka menggunakan multi-dimensional histogram dengan mengambil nilai *hue* dan *saturation.* Multi-dimensional histogram digunakan untuk komputasi probabilitas *back projection* yang berkoresponden selama *tracking* nantinya.

Setiap perubahan citra yang terjadi akan dihitung dan ditempatkan dalam skala [0;1] untuk setiap histogramnya. Nantinya histogram *hue* atau *saturation* yang berada pada satu objek akan digabung, dihitung kembali dan hasilnya akan ditempatkan kembali dalam skala [0;1].

Selanjutnya dalam banyak kasus, histogram yang men-*tracking* objek dipengaruhi oleh perubahan iluminasi, *background* yang kacau dengan warna objek, dan perubahan ukuran geometri dari objek. Untuk itulah sangat diperlukan untuk meng-*update* model histogram objek agar *tracking* menjadi lebih baik.

Kita memberikan nilai update $\tau \in (0,1)$ yang nantinya akan menjadi koefisien dalam rumus pencarian histogram $\widehat{H}_t$ baru yang adaptif. Rumus nilai update sebagai berikut

$$\widehat{\mathrm{H}}_t = (1-\tau) * \mathrm{H}_t + \tau\, \mathrm{H}_{t-1} \quad (20)$$

Update koefisien τ dijalankan sesuai dengan parameter yang mengontrol ukuran dari target model histogram *adaptation* dari *candidate* model. Berdasarkan pendekatan histogram *generalized adaptation*, kita akan membagi *tracking window* menjadi bagian yang blok-blok kecil. Setiap blok itu akan dihitung setiap komponen histogram *hue* dan *saturation* dan menemukan *sub-region* yang mirip dengan distribusi histogram yang sedang dipakai.

Dengan menggunakan hasil ini, kita akan mengukur (*weighting*) histogram *sub-region* dan hasilnya akan dimasukkan ke dalam histogram baru dengan menggunakan *weighting function* dengan *exponential preference* untuk setiap *region* yang mirip dengan *region* serupa.

Kita andaikan ada beberapa blok sejumlah M yang terdiri dari $B_1, B_2 .. B_M$ dengan S_i piksel dalam blok i^{th}. Setiap pembagian *tracking window* ditunjukkan oleh cara berubah-ubah, yang secara khusus diberikan *tracking window* untuk memisahkan ke dalam blok-blok persegi yang diatur oleh sebuah *regular grid* secara kolektif.

Untuk setiap blok, normalisasinya E_i ada diatas *threshold*. Kami menganggap sebagai sebuah *outlier* dan mengeluarkannya dari komputasi histogram

$$E^i{}_{block\ i} = \frac{1}{S_i} \sum_{x \in B_i} [\hat{H}_{ij}^t - \hat{H}_{ij}^{t-1}]^2 \quad (21)$$

2.5 *People Counting*

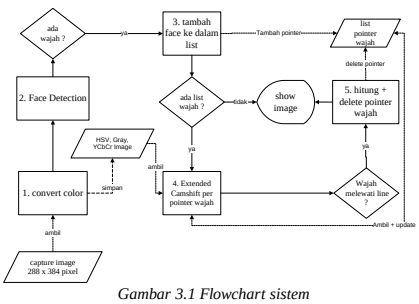
People counting menjadi bidang riset yang yang telah memperoleh banyak perhatian beberapa tahun ini. Sistem ini mampu untuk menghitung jumlah orang ketika masuk atau keluar sebuah ruangan. *People counting* saat berguna untuk memantau jumlah orang di sebuah ruangan dengan tujuan pengamanan, mengatur kapasitas, merancang strategi marketing terhadap jumlah pengunjung, atau yang lainnya.

Secara tradisional *people counting* sudah dapat dilakukan menggunakan *turnstile*, laser atau sensor yang lainnya. Tapi alat-alat ini mengganggu pergerakan manusia yang melewatinya. Sebagai pembandingnya, *people counting* memanfaatkan *computer vision* mampu memberikan solusi yang tidak mengganggu dan murah. Selain itu, banyak kamera pemantau yang banyak dipasang dan bisa dimanfaatkan untuk *people counting.*

3. Perancangan Sistem

3.1 Perancangan People Counting

People Counting adalah gambaran umum dari sistem ini. Sistem ini akan menggunakan *counting line* untuk mengetahui jumlah orang yang dihitung. Ketika wajah orang yang terdeteksi kamera, wajah itu disimpan dalam list pointer, wajah itu kemudian dilakukan *tracking* pada frame yang dilewati, dan ketika wajah itu melewati *counting line* maka wajah itu akan dihitung. Gambaran sistem (per frame yang masuk) sebagai berikut :



Gambar 3.1 Flowchart sistem

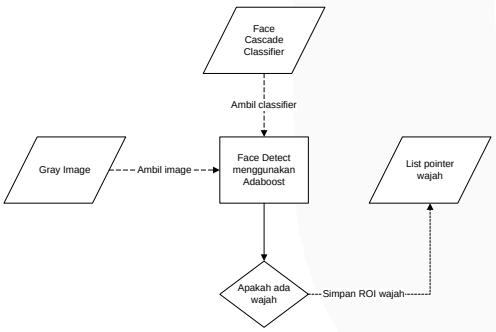
3.2 Perancangan Face Detection – Haar Cascade Classifier

Untuk perancangan fitur haar *cascade classifier*, diperlukan data wajah yang akan di-*training* dan hasilnya berupa classifier akan digunakan untuk mengenal region yang mana wajah dan non-wajah. Data *training* akan berformat .xml dan nantinya data training ini akan dipakai sebagai *classifier* untuk *people counter.*

Classifier yang telah didapat akan dipakai dalam pendektesian wajah selanjutnya. *Classifier* akan menghasilkan nilai positif dan nilai negatif untuk wajah. Data *classifier* .xml hasil *training* wajah bisa didapat dari beberapa sumber seperti hasil *training* dari beberapa dataset [11] atau dari OpenCV.org [12]. Aplikasi *Face detection* pada tugas akhir ini memanfaat *classifier* hasil *training* untuk mendeteksi wajah yang ada pada setiap *frame image.*

Untuk pengujian deteksi muka nantinya akan dicari parameter terbaik pada bab selanjutnya untuk digunakan ketika aplikasi *People Counter*. *Face Detection* ini termasuk bagian yang paling penting karena aplikasi ini pembuka untuk aplikasi lainnya. Jika tidak ada wajah yang terdeksi, maka tidak ada wajah bisa di-*tracking* dan dihitung. Jika ada wajah yang terdeteksi, maka ROI wajah itu disimpan.

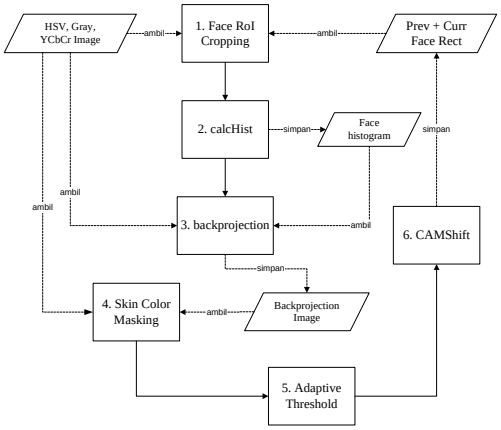
Gambaran *Face Detection* per frame bisa dilihat di *flowchart* di bawah ini :



Gambar 3.2 Flowchart Face Detection

3.3 Perancangan *Extended* CAMSHIFT *Face Tracking*

Aplikasi *Face Tracking* akan dilakukan setelah pendeteksian wajah. Setiap wajah yang terdeteksi yang ada di dalam list pointer akan dilakukan *tracking* dalam setiap frame. Untuk aplikasi *Face Tracking* juga dilakukan *pre-processing* untuk setiap wajahnya. Tahapan *tracking* untuk setiap pointer dalam wajah (*Face Rect*) per frame (*HSV Image*) bisa dilihat di flowchart diagram di bawah ini



Gambar 3.3 Flowchart Extended CAMSHIFT

Wajah yang telah terdeteksi akan dilakukan *tracking* sesuai kebutuhan. *Tracking* wajah dimulai dengan menyimpan wajah yang telah terdeteksi menjadi *intial tracking window* dan selanjutnya setiap koordinat *tracing window* disimpan dalam list berbentuk pointer.

4. Analisa dan Pengujian Sistem

4.1 Pengujian Sistem

Pengujian sistem tugas akhir ini akan terdiri dari tiga bagian dan harus bertahap. Tahapan ini dilakukan karena jika ada salah satu bagian belum berhasil maka bagian selanjutnya belum bisa diuji. Pengujiannya antara lain :

1. Pengujian *Face Detection* menggunakan algoritma Viola-Jones
2. Pengujian *Face Tracking* menggunakan *Extended* CAMSHIFT dari hasil *Face Tracking.*
3. Pengujian *People Counting*

4.2 Tujuan Pengujian

Tujuan dilakukannya pengujian adalah

1. Menganalisa pengaruh *scaleFactor* dan *minimum Neighbors* pada saat Viola-Jones *Face Detection.*
2. Menganalisa pengaruh nilai update coeffisien τ dan nilai normalisasi E *Saturation* pada saat *tracking* menggunakan algoritma *Extended Camshift.*
3. Mengukur keefektifan sistem ini ketika menghitung orang.

4.3 Hasil dan Analisa Pengujian Sistem

4.3.1 Hasil dan Analisa Pengujian Skenario A

Pengujian skenario A mencari nilai *scale factor* terbaik menggunakan nilai 1.2, 1.3, 1.4. Rumus akurasi deteksi sebagai berikut $akurasi = \frac{waja\ h\ terdeteksi}{total\ waja\ h}$

Hasil ujicoba pendeteksian wajah dengan nilai minimal *neighbours* sementara 5 bisa dilihat tabel di bawah. Nilai error nantinya akan dipakai untuk memisahkan parameter mana yang tidak dipakai. Jika ada nilai error maka paramater itu tidak akan

dipakai. Pada tabel berikut data yang error akan diberi tanda merah :

nama	jumlah terdeteksi	jumlah real	error	keterangan	akurasi
1	2	3	0	indoor terang	67%
2	2	3	0	indoor terang	67%
3	1	2	1	indoor terang	50%
4	1	2	2	outdoor gelap	50%
5	2	2	2	outdoor gelap	100%
6	3	3	0	outdoor gelap	100%
7	1	1	1	indoor gelap	100%
8	1	1	0	indoor gelap	100%
9	1	1	1	indoor gelap	100%
10	1	3	1	outdoor terang	33%
11	5	9	0	outdoor terang	56%
12	4	7	0	outdoor terang	57%

Tabel 4.3 Hasil Face Detection dengan Scale Factor 1.4

nama	jumlah terdeteksi	jumlah real	error	keterangan	akurasi
1	2	3	0	indoor terang	67%
2	2	3	0	indoor terang	67%
3	1	2	0	indoor terang	50%
4	1	2	0	outdoor gelap	50%
5	2	2	0	outdoor gelap	100%
6	3	3	0	outdoor gelap	100%
7	1	1	0	indoor gelap	100%
8	1	1	0	indoor gelap	100%
9	1	1	0	indoor gelap	100%
10	2	3	0	outdoor terang	67%
11	6	9	0	outdoor terang	67%
12	5	7	0	outdoor terang	71%

Tabel 4.4 Hasil Face dengan scale factor 1.3

nama	jumlah terdeteksi	jumlah real	error	keterangan	akurasi
1	2	3	0	indoor terang	67%
2	3	3	0	indoor terang	100%
3	1	2	0	indoor terang	50%
4	1	2	0	outdoor gelap	50%
5	2	2	0	outdoor gelap	100%
6	3	3	0	outdoor gelap	100%
7	1	1	0	indoor gelap	100%
8	1	1	0	indoor gelap	100%
9	1	1	0	indoor gelap	100%
10	2	3	0	outdoor terang	67%
11	6	9	0	outdoor terang	67%
12	5	7	0	outdoor terang	71%

Tabel 4.5 Hasil Face Detection dengan scale factor 1.2

Berdadarkan hasil tersebut wajah masih bisa diberikan nilai *scale factor* 1.2 dan deteksinya mampu berjalan di kondisi outdoor terang, outdoor gelap, indoor terang, maupun indoor gelap. Akan tetapi untuk penentuan minimal *neighoburs* yang tepat harus dicoba lagi. Percobaan minimal neighbour yang tepat akan diujicobakan dengan nilai 3,4, dan 5.

Tabel berikut hasil ujicoba wajah dengan minimal *neighbours* 3 dan 4 dengan menggunakan *scale factor* 1.2. Minimal *neighbours* 5 sudah diujicobakan pada tahap pertama dan hasilnya bisa diambil dari tabel 4.3

nama	jumlah terdeteksi	jumlah real	error	keterangan	akurasi
1	2	3	0	indoor terang	67%
2	3	3	0	indoor terang	100%
3	1	2	0	indoor terang	50%
4	1	2	0	outdoor gelap	50%
5	2	2	0	outdoor gelap	100%
6	3	3	0	outdoor gelap	100%
7	1	1	0	indoor gelap	100%
8	1	1	0	indoor gelap	100%
9	1	1	0	indoor gelap	100%
10	2	3	0	outdoor terang	67%
11	6	9	0	outdoor terang	67%
12	5	7	0	outdoor terang	71%

Tabel 4.6 Hasil Face Detection dengan min. Neighoburs 4

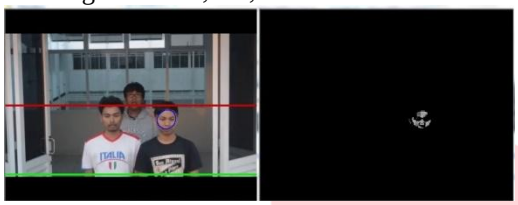
nama	jumlah terdeteksi	jumlah real	error	keterangan	akurasi
1	2	3	0	indoor terang	67%
2	3	3	0	indoor terang	100%
3	1	1	1	indoor terang	100%
4	1	2	0	outdoor gelap	50%
5	2	2	1	outdoor gelap	100%
6	2	3	2	outdoor gelap	67%
7	1	1	0	indoor gelap	100%
8	1	1	0	indoor gelap	100%
9	1	1	0	indoor gelap	100%
10	2	3	0	outdoor terang	67%
11	6	9	1	outdoor terang	67%
12	5	7	1	outdoor terang	71%

Tabel 4.7 Hasil Face Detection dengan min. Neighoburs 3

4.3.2 Hasil dan Analisa Pengujian Skenario B

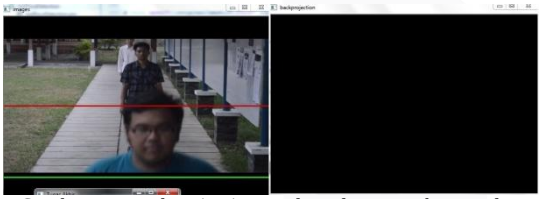
Sebelum skenario B dilakukan, *tracking* diujicoba di berbagai kondisi seperti outdoor dengan kondisi cahaya redup (sekitar jam 5 sore dan tertutup bayangan), outdoor dengan cahaya cukup terang, indoor gelap, dan indoor dengan cahaya cukup terang. Pada skenario B ini hanya

diujicobakan data dengan jumlah orang yang sedikit dan jalannya sudah diskenarionkan. Tujuan data seperti ini agar kita tahu hasil *tracking* saja. Apakah ini *tracking* ini berjalan normal dan bisa dilihat lebih teliti hasilnya. Berdasarkan hasil percobaan dari data yang ada *backprojection* hanya berjalan pada kondisi terang baik outdoor maupun indoor. Berdasarkan hal ini *tracking* skenario B hanya dilakukan di kondisi terang. Hasil *backprojection* sangat mempengaruhi *tracking*. Hasil *backprojection* dengan berbagai kondisi bisa dilihat di gambar 4.4, 4.5, dan 4.6.



Gambar 4.5 Backprojection indoor dengan cahaya cukup terang

Gambar 4.6 Backprojection outdoor dengan cahaya cukup terang



Gambar 4.7 Backprojection outdoor dengan cahaya redup

Pada kondisi gelap, probabilitas warna pada *backprojection* tidak muncul sama sekali sehingga objek tidak bisa dilakukan *tracking*. Jadi skenario B dilakukan di kondisi terang.

Pengukuran akurasi hasil *tracking* sama seperti deteksi wajah. Jika *tracking* gagal dengan asumsi *tracking* tidak mencapai *counting line* atau tertahan di tengah-tengah maka *tracking* dikategorikan tidak berhasil. Selain itu asumsi lainnya jika hanya wajah yang terdeteksi bisa dilakukan *tracking* yang dihitung dalam akurasi *tracking ini. Tracking* dilkukan dengan kondisi cahaya terang (outdoor) dan ada menggunakan hambatan seperti wajah berdempetan dan tanpa hampatan seperrti jalan sendirian atau jalan bersamaan. Dari hasil pengaruh nilai update tidak berpengaruh banyak. Untuk perbedaan hasil 1 orang pada video 2 di tabel hasil, bukan termasuk hasil signifikan. Hipotesa akurasi di atas 93.4% tidak benar karena itu tergantung kondisi pencahayaan dan kemampuan kamera. Rata-rata akurasi tertinggi sebesar 88 %. Hasil *tracking* bisa dilihat pada tabel di bawah ini dengan rata-rata akurasi tertinggi :

video	deskripsi video	jumlah orang	nilai update video		
			0.25	0.5	0.75
1	satu orang berjalan sendiri-sendiri	3	66%	66%	66%
2	dua orang berjalan bersama-sama	4	75%	50%	75%
3	Ada orang dempet dengan lainnya	3	100%	100%	100%
4	Salah satu orang melakukan *crossing*	2	100%	100%	100%
5	orang terhalang benda	1	100%	100%	100%
rata-rata akurasi			88%	83%	88%

Tabel 4.8 Hasil tracking outdoor terang

4.3.3 Hasil dan Analisa Pengujian Skenario C

Skenario ini adalah data real yang diujicobakan untuk *people counting. Counting line* mempunyai fungsi untuk mengetahui siapa saja diantara objek yang mengalami deteksi dan *tracking* yang termasuk perhitungan. *Counting line* memberikan arti ruang gerak apakah objek itu berjalan sesuai perhitungan. Karena data berbasis video, maka *counting line* diambil 10 dan 25 persen dari bawah. 10 persen mempunyai arti sangat dekat dengan batas bawah sedangkan 25 persen lebih agak ke atas. Nilai 25 diset agar wajah segera terhitung. Untuk akurasi *people counting* ini, nilai yang didapat dari jumlah di sistem orang yang wajah frontalnya terlihat melewari sampai *counting line* per jumlah *real*-nya.

Kesimpulan untuk skenario ini jumlah orang melewati terlalu banyak dan dempet maka data yang dihasilkan tidak akurat. Dari hasil ujicoba ternyata yang lebih baik *counting line* diset mendekati batas bawah video. Tetapi ketika *tracking* nya buruk yang dipengaruhi cahaya maka hasilnya *counting* nya juga buruk. Data hasil *people counting* dengan nilai *counting line* 10 bisa dilihat sebagai berikut :

nama	jumlah terdetek	jumlah real
video 1	5	11
video 2	5	6
video 3	4	4

Tabel 4.9 Akurasi dengan counting line 10

Ketika *counting line* di-set sebesar 25, maka hasil *counting line* menjadi berkurang karena wilayah deteksi muka semakin diperkecil. Semakin diperkecil wilayah deteksi muka maka berakibat ke hasil deteksi.

nama	jumlah terdetek	jumlah real
video 1	6	11
video 2	2	6
video 3	4	4

Tabel 4.9 Akurasi dengan counting line 25

5. Kesimpulan dan Saran

Dalam tugas ini penulis menyimpulkan

1. Data *training* untuk *cascade classifier* deteksi wajah harus jelas apakah nantinya *people counting* berbentuk menghadapat wajah frontal, miring, atau yang lainnya. Untuk metode ini *cascade classifier* harus sesuai dengan kondisi cahaya dan letak kamera. Letak kamera yang salah dan cahaya sekitara yang kurang berpengaruh terhadap deteksi wajah.
2. Cahaya sekitar juga perlu diperhatikan karena berdampak pada *tracking*. Cahaya yang gelap terhadap warna kulit akan membuat *backprojection* tidak terlihat sehingga ketika *Extended* CAMSHIFT berjalan menghasilkan akurasi yang jelek.
3. *Counting line* berpengaruh terhadap jumlah objek. Ketika *counting line* digeser ke bawah, maka masih tersedia ruang yang besar terhadap deteksi da *tracking.* Ketika terlalu ke atas maka bisa jadi wajah tidak terdeteksi karena terhalang objek lain atau piksel wajah kurang memenuhi syarat *cascade classifier.*

Saran dari penulis untuk tugas akhir

1. Untuk mengaplikasikan *people counting* harus dalam kondisi cahaya yang terang menghadap area warna kulit agar bisa melakukan *tracking* yang baik. Perlu ada metode deteksi kulit pada kondisi cahaya yang gelap.
2. *Agar face detection* dan saat perhitungan orang menggunakan *counting line* menghasilkan perhitungan yang cepat di resolusi yang besar, perlu algoritma atau strategi khusus agar bisa melakukan *people counting* di resolusi kamera yang besar.
3. Perlu ada sistem yang bisa automatis memberikan parameter-parameter yang digunakan pada setiap lingkungan sehingga jika ada lingkungan baru tidak usah men-set parameter. Dalam kasus ini contoh system bisa automatis men-set lingkungan gelap atau terang

Daftar Pustaka

[1] Chen, Tsong-Yi, Chao-Ho Chen, Da-Jinn Wang, and Yi-Li Kuo. "A People Counting System Based on Face-Derection." *Fourth International Conference on Genetic and Evolutionary Computing*, 2010: 699-702.

[2] Lee, Lae-Kyoung, Su-Yong An, and Oh Se-young. "Efficient Face Detection and Tracking with Extended CAMSHIFT and Haar-Like Features." *IEEE International Conference on Mechatronics Automation*, August 2011: 507-513.

[3] R. Bradski, Gary. "Computer Vision Face Tracking For Use in a Perceptual User Interface." *Intel Technology Journal Q2*, 1998: 1-15.

[4] Comaniciu, Dorin, Ramesh Visvanathan, dan Pete Meer. "Real-Time Tracking of Non-Rigid Objects using Mean Shift." *Computer Vision and Pattern Recognition* 2 (2000): 142-149.

[5] Putra, Septioadi Anggara. "Hand Tracking dengan Menggunakan Metode CAMShift dan Kalman Filter Pada Augment Reality". Institut Teknologi Telkom. Bandung. 2012

[6] Viola, Paul and Michael J. Jones. "Robust Real-Time Face Detection". *International Journal of Computer Vision 57(2)*, 2004:137-154

[7] R.L Hsu, M. Abdel-Mottaleb, and A.K. Jain. "Face Detection in Color Images," *IEEE Transactions on Pattern Analysis and Machine Intelligence 24(5)*,2002: 900-903.

[8] Lee, Lae-Kyoung, Su-Yong An, and Oh Se-young. "Robust Visual Object Tracking with extended CAMSHIFT in Complex Environments," *IEEE Industrial Elextronics Society*, Melbourne, 2011 : 4536-4542

[9] Cong, Yang, Haifeng Gong, Song-Chun Zhiu, Yandong Tang. "Flow Mosaicking: Real-time Pedestrian Counting without Scene-specific Learning,"*Proceedings of IEEE Computer Vision and Pattern Recognition*, Ezhou, 2009 : 1093-1100

[10] Hakimi, Yogi Fahmi. "Pendeteksian Wajah Menggunakan *Normalized Color Coordinates* dan *Skin Ratio* untuk *People Counting*". Institut Teknologi Telkom. Bandung. 2013

[11] Seo, Naotoshi. "Tutorial: OpenCV haartraining (Rapid Object Detection with A Cascade of Boosted Classifiers Based on Haar-Like Features)", pp. http://note.sonots.com/SciSoftware/haartraining.html, Aug 2015

[12] OpenCV Documentation, "Haar Feature-based Cascade Classifier", pp. http://docs.opencv.org/doc/tutorials/objdetect/cascade_classifier/cascade_classifier.html, Aug 2015

[13] Lienhart, Rainer, Alexander Kuranov, and Vadim Pisarevsky. "Empirical Analysis of Detection Cascades of Boosted Classifiers for Rapid Object Detection". MRL Technical Report, 2002

[14] Sourceforge, "OpenCV Open Source Computer Vision Library", pp. http://sourceforge.net/projects/opencvlibrary/, Aug 2015