Penerapan dan Analisis Sistem Penghitungan Orang Menggunakan *Adaptive Gaussian Mixture Model* dan *Local Binary Pattern*

Implementation and Analysis of People Counting System Using Adaptive Gaussian Mixture Model and Local Binary Pattern

Reza Dwi Ansari[0], Bedy Purnama, S.Si, MT[1], Febryanti Sthevanie, ST, MT[2]

Fakultas Informatika, Universitas Telkom
Jl. Telekomunikasi No. 1 Dayeuhkolot, Bandung 40257
Telp (022) 7564108
ansariholic@gmail.com[0], bedy.purnama@gmail.com[1], febryantisthevanie@gmail.com[2]

Abstrak

Sistem penghitungan orang otomatis telah banyak digunakan di beberapa tempat umum, baik untuk aplikasi komersial maupun untuk keamanan. Jumlah orang yang didapat dalam periode waktu tertentu bermanfaat untuk sistem pengawasan, analisis pengunjung, strategi pemasaran, dan lain sebagainya.

Dalam tugas akhir ini metode deteksi wajah digunakan untuk penghitungan orang pada input data video. *Local binary pattern* (LBP) merupakan salah satu metode yang dapat digunakan dalam pendeteksian wajah. Metode ini membutuhkan jumlah dan varian sampel. Sampel tersebut berupa citra wajah dan citra bukan wajah yang selanjutnya akan dikomputasi untuk proses training dataset wajah. Untuk membantu meningkatkan akurasi deteksi, digunakan *Adaptive Gaussian Mixture Model* untuk segmentasi *background* dan *foreground.* Hasil training dan ekstraksi *foreground* akan digunakan dalam pendeteksian wajah. Proses penghitungan orang menggunakan garis hitung atau *Line of Interest* (LOI). Apabila muka orang yang terdeteksi berjalan melewati garis perhitungan dan mendekati kamera, maka akan dihitung oleh sistem.

Dengan metode-metode tersebut, tingkat akurasi sistem untuk penghitungan orang didapat 100% untuk lima dari tujuh video uji. Performansi sistem berjalan cukup baik untuk memproses video uji dengan rata-rata pemrosesan 18-29 fps.

Kata kunci : *fitur LBP, background subtraction, Adaptive Gaussian Mixture Model,* penghitungan orang

1. Pendahuluan

Kebutuhan informasi tentang jumlah orang yang berada pada suatu area dalam suatu periode waktu tertentu sangat bermanfaat dalam berbagai kebutuhan publik. Salah satu cara untuk mendapatkan informasi jumlah orang pada area publik adalah dihitung dari jumlah orang yang masuk ke suatu area. Pemantauan orang tersebut bisa secara manual dengan adanya petugas jaga atau secara otomatis dengan menggunakan kamera video atau sensor yang terhubung dengan komputer. Informasi orang yang telah dipantau tersebut bisa digunakan untuk membangun sistem penghitungan orang atau *People Counting System*. Saat ini *people counting* banyak digunakan dalam bidang keamanan, kebijakan pemasaran, dan pengawasan. Penghitungan orang dengan menggunakan komputer dapat menghasilkan hasil yang bagus dan sistem yang handal bergantung pada pemilihan metode deteksi orang dan perhitungan orang. Masalah yang sering terjadi pada sistem penghitungan orang adalah memastikan apakah objek yang dihitung adalah orang atau bukan. Salah satu cara untuk mengatasinya adalah dengan dilakukan proses ekstraksi ciri pada objek untuk memastikan apakah objek yang dideteksi untuk dihitung tersebut adalah orang atau bukan. Ciri yang digunakan sebagai representasi orang adalah ciri muka. Hasil deteksi orang yang baik sangat bergantung pada hasil ekstraksi ciri muka yang baik pula sehingga akan menghasilkan perhitungan orang yang baik juga. Penghitungan orang diukur berdasarkan akurasi penghitungan orangnya. Tingkat akurasi ini bergantung pada nilai parameter-parameter yang digunakan dalam membangun sistem, yaitu proses background subtraction menggunakan *Adaptive Gaussian Mixture Model* (GMM) dan proses deteksi muka orang menggunakan *Local Binary Pattern* (LBP). Parameter-parameter pada Adaptive GMM dan LBP selanjutnya akan dianalisis pengaruhnya terhadap tigkat akurasi sistem penghitungan orang.

Gaussian Mixture Model merupakan suatu metode yang digunakan untuk *background subtraction*. Proses *background subtraction* akan menghasilkan citra *foreground*. Metode GMM ini bersifat adaptif, yaitu dapat menangani masalah perubahan lingkungan dinamis, seperti perubahan intensitas cahaya, yang telah dilakukan pada percobaan [11,12,16,18,22]. Perubahan intensitas cahaya ini dapat mempengaruhi proses penghitungan orang. Ekstraksi *foreground* yang baik akan membuat penghitungan lebih akurat. Maka dari itu pada Tugas Akhir ini digunakan *Adaptive Gaussian Mixture Model* untuk ekstraksi *foreground* yang dapat mengatasi kondisi perubahan pada lingkungan dan bersifat adaptif.

Proses selanjutnya adalah proses deteksi objek. Objek dalam hal ini adalah orang. Untuk proses deteksi orang, dalam Tugas Akhir ini digunakan metode *Local Binary Pattern*. LBP awalnya diimplementasikan untuk deskripsi tekstur [10]. LBP berbasis fitur digunakan dalam Tugas Akhir ini untuk proses ekstraksi ciri. Hasil dari ekstraksi ciri tersebut selanjutnya akan digunakan untuk menentukan fitur suatu objek apakah objek tersebut adalah orang

atau bukan. Ciri yang akan diekstraksi dari orang adalah bagian muka, yang kemudian akan dideteksi. Muka dianggap sebagai komposisi dari tekstur-tekstur mikro tergantung pada situasi lokal [17].

2. Landasan Teori

2.1 Adaptive GMM

Gaussian Mixture Model (GMM) merupakan salah satu metode dalam *background subtraction*. Metode ini digunakan untuk mendeskripsikan piksel dari *background*. *Gaussian Mixture Model* telah menjadi terkenal karena efisiensinya dalam pemodelan distribusi *multi-modal* dari beberapa *background*, seperti pergerakan yang terjadi pada pohon, ombak, refleksi cahaya, dan lain sebagainya [21].

Tahapan dari metode GMM ini antara lain tahap pembentukan model background, tahap pencocokan input terhadap distribusi dan tahap pemilihan distribusi yang termasuk model *background* [16]. Gambaran umumnya, pada GMM, setiap data piksel dalam sebuah frame video memiliki model sendiri yang diolah dari input warna R, G, B. Suatu piksel X pada koordinat (□,□) akan ditentukan jenisnya. Pertama-tama, diperlukan sebuah masukan data piksel pada koordinat tersebut. Nilai data ini merupakan *mean* dari distribusi gaussian dengan nilai varian yang telah diinisialisasi. Pada waktu berikutnya, akan dilakukan proses *update* nilai pembentuknya (*mean* dan varian) pada distribusi ini sampai terbentuk beberapa distribusi. Distribusi-distribusi ini selanjutnya akan dipilih sebagai model yang merepresentasikan *background*. Data piksel X kemudian dicocokkan dengan model *background* tersebut berdasarkan probabilitas Gaussian-nya. Apabila termasuk dalam model *background*, maka piksel inputan tersebut merupakan piksel *background*, sebaliknya merupakan piksel *foreground* [12].

Jika data yang digunakan merupakan skalar, gaussian distribusi didefinisikan sebagai:

$$\eta(x,\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{1(x-\mu)^2}{2\sigma^2}} \quad (2.2)[18]$$

Sedangkan jika data yang digunakan adalah vektor, maka didefinisikan sebagai berikut.

$$\eta(\vec{X}_t,\vec{\mu}_k,\Sigma_k) = \frac{1}{(2\pi)^{\frac{n}{2}}|\Sigma|^{\frac{1}{2}}} e^{-\frac{1}{2}(\vec{X}_t-\vec{\mu}_t)\Sigma^{-1}(\vec{X}_t-\vec{\mu}_t)} \quad (2.3)[18]$$

Dalam pemodelan ini, GMM yang digunakan adalah GMM yang telah dimodifikasi pada [22]. Pada tahap pemodelan GMM ini, komponen pembentuk distribusi Gaussian (*mean* dan varian) didapatkan dengan mengestimasi nilainya. Tiap piksel baru akan digunakan untuk mengestimasi nilai *mean* dan varian distribusi yang baru. Distribusi yang terbentuk tidak hanya satu, tetapi bisa terbentuk beberapa distribusi. Distribusi yang terbentuk akan dibatasi sebesar K komponen.

Pertama-tama akan dicari nilai vektor *mean* ($\vec{\mu}$) dan *matrix covariance* (Σ) yang akan menentukan probabilitas persebaran data pada piksel tersebut dengan persamaan

$$p(\vec{X}_t) = \sum_{k=1}^{K} \vec{\omega}_{k,t} * \eta(\vec{X}_t,\vec{\mu}_{k,t},\Sigma_{k,t}) \quad (2.4)[22]$$

dimana K adalah jumlah distribusi Gaussian, $\vec{\omega}_{k,t}$ adalah *weight* pada Gaussian ke-i pada waktu t, $\vec{\mu}_{k,t}$ adalah vektor *mean* pada Gaussian ke-i pada waktu t, $\Sigma_{k,t}$ adalah *matrix covariance* pada Gaussian ke-i pada waktu t, $\vec{X}_t$ adalah *matrix* input data pada waktu t, dan η adalah *probability density function* seperti persamaan (3). *Matrix covariance* didapat dengan cara mengalikan varian semua distribusi dengan matriks identitas (I).

$$\Sigma_{k,t} = \vec{\sigma}_t^2 I \quad (2.5)[18]$$

Setelah model terbentuk, akan dipilih model *background* dari model-model distribusi yang telah terbentuk.

Pada pencocokan setiap data piksel inputan dengan model distribusi yang telah dibentuk. Pencocokan ini dilakukan untuk menentukan apakah piksel tersebut termasuk ke dalam *background* atau *foreground*. Piksel dianggap cocok dengan distribusi apabila jarak Mahalanobis-nya kurang dari jarak T kali dari standar deviasi sebuah distribusi atau dalam bentuk persamaannya sebagai berikut:

$$\sqrt{(\vec{X}_t-\vec{\mu}_{k,t})^T\Sigma^{-1}(\vec{X}_t-\vec{\mu}_{k,t})} \le \vec{\sigma}_t \quad (2.6)[22]$$

Apabila data piksel tidak ada yang cocok dengan semua model distribusi, maka piksel tersebut dianggap sebagai *foreground* dan dibuat distribusi baru dengan menggunakan distribusi yang merepresentasikan *background*. Sedangkan apabila ada model distribusi yang cocok dengan data piksel inputan, maka bobot dari semua distribusi diatur sebagai berikut:

$$\vec{\omega}_{k,t} = (1-\alpha)\vec{\omega}_{k,t-1} + \alpha(M_{k,t}) \quad (2.7)[22]$$

Dimana α adalah *learning rate*, $M_{k,t}$ bernilai 1 jika model distribusi cocok dengan data piksel inputan dan 0 jika tidak cocok. Setelah itu nilai *weight* dinormalisasi [17]. Parameter μ dan σ untuk distribusi yang tidak cocok tetap sama sedangkan untuk distribusi yang cocok akan diperbarui seperti berikut:

$$\vec{\mu}_t = \vec{\mu}_t + \rho(\frac{\alpha}{\vec{\omega}_{k,t}})\vec{\delta}_t \quad (2.8)[22]$$

$$\vec{\sigma}_t^2 = \vec{\sigma}_t^2 + \rho(\frac{\alpha}{\vec{\omega}_{k,t}})(\vec{\delta}_t^T\vec{\delta}_t - \vec{\sigma}_t^2) \quad (2.9)[22]$$

dimana

$$\vec{\delta}_{\blacksquare} = \vec{\blacksquare}^{\blacksquare} - \vec{\mu}^{\blacksquare} \quad (2.10)[22]$$

Setelah semua distribusi di-*update*, berikutnya lakukan normalisasi *weight* hingga penjumlahan dari *weight* semua distribusi sebanyak K, nilainya adalah 1.

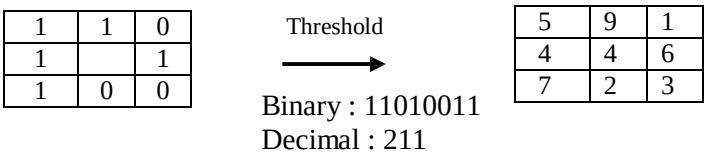
Untuk menentukan distribusi mana yang merepresentasikan *background*, dilakukan pengurutan distribusi berdasarkan nilai ω dari distribusi yang paling merepresentasikan *background* sampai distribusi yang tidak merepresentasikan *background*. Distribusi B yang pertama dipilih sebagai model *background* menggunakan persamaan:

$$\blacksquare = \blacksquare\blacksquare\blacksquare\blacksquare\blacksquare\blacksquare(\Sigma_{\blacksquare=1}^{\blacksquare}\omega_{\blacksquare} > 1 - \blacksquare_{\blacksquare}) \quad (2.11)[22]$$

dimana $\blacksquare_{\blacksquare}$ adalah ukuran maksimal dari porsi data yang merupakan bagian dari objek *foreground* yang tidak mempengaruhi *background model*[22].

2.2 Local Binary Pattern

Local binary pattern (LBP) adalah sebuah deskriptor *nonparametric* yang secara efisien merangkum struktur lokal dari gambar dengan cara membandingkan setiap piksel dengan piksel *neighbor* [5]. Properti paling penting dari LBP adalah ketahanannya pada perubahan iluminasi monoton dan kesederhanaan komputasionalnya. Operator asli LBP memberikan label piksel dari sebuah gambar dengan nomer desimal, yang disebut *LBPs* atau *LBP codes* yang melakukan *encode* struktur lokal di sekitar tiap piksel. Setiap piksel dibandingkan dengan delapan piksel tetangga pada 3x3 *neighborhood* dengan cara substraksi nilai piksel tengah. Nilai-nilai negatif dikodekan dengan 0, dan selainnya dengan 1. Untuk setiap piksel yang diambil, bilangan biner didapat dari penggabungan semua nilai biner dalam searah jarum jam, yang dimulai dari kiri atas *neighbor*. Nilai desimal dari bilangan biner yang dihasilkan kemudian digunakan untuk pemberian label piksel yang diambil. Bilangan asal biner dianggap sebagai LBPs atau LBP *codes* seperti diilustrasikan pada gambar 1.



Gambar 2-1 Contoh operator basic LBP

Secara formal, piksel (x_c,y_c), hasil LBP dapat ditunjukkan dalam bentuk desimal sebagai berikut:

$$\blacksquare\blacksquare_{\blacksquare\blacksquare\blacksquare\blacksquare\blacksquare\blacksquare} = \sum_{\blacksquare=0}^{\blacksquare-1}\blacksquare(\blacksquare_{\blacksquare} - \blacksquare_{\blacksquare})2^{\blacksquare} \quad (2.12)[3]$$

dimana i_c dan i_p adalah nilai *gray-level* pada piksel tengah dan P mengellilingi piksel dalam *circle neighborhood* dengan radius R, dan fungsi s(x) didefinisikan :

$$\blacksquare(\blacksquare) = \begin{cases} 1, \blacksquare \geq 0 \\ 0, \blacksquare < 0 \end{cases} \quad (2.13)[3]$$

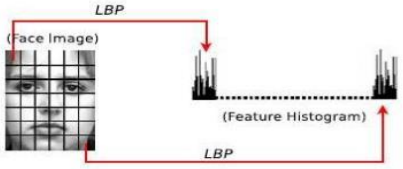
Dengan melakukan ekstraksi piksel menggunakan LBP, akan didapatkan nilai *matrix* baru yang akan dirubah ke histogram untuk memperoleh fitur vektor wajah. Seringkai distribusi LBP *code* dalam sebuah gambar digunakan untuk mendeskripsikan tekstur sebagai sebuah histogram dari sebuah gambar [3].

2.2.1 Ekstraksi Fitur dengan LBP

Setiap gambar wajah dapat dianggap sebagai komposisi dari *micro-pattern* yang bisa dideteksi secara efektif oleh LBP *operator* [3]. Sebuah histogram LBP dapat dibangun dan kemudian dihitung pada gambar wajah. Pada [17] dikenalkan *LBP based face representation* untuk *face recognition*. Untuk mempertimbangkan informasi bentuk dari wajah, gambar wajah dibagi ke dalam M *non-overlapping region* R_o, R_1, ..., R_M seperti pada gambar 2-5. Histogram LBP kemudian diekstraksi dari setiap *sub-region* untuk dikonkatenasi dalam fitur histogram tunggal yang ditingkatkan secara spasial yang merepresentasikan gambar wajah, seperti persamaan:

$$\blacksquare_{\blacksquare\blacksquare} = \sum_{\blacksquare\blacksquare}\blacksquare(\blacksquare\blacksquare\blacksquare\blacksquare = \blacksquare)\blacksquare(\blacksquare\blacksquare\blacksquare_{\blacksquare}) \quad (2.14)[16]$$

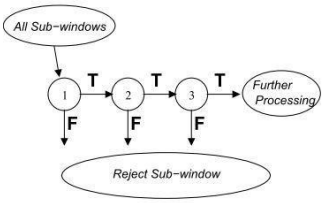
dimana i=0, ..., L-1 dan j=0,..., M-1



Gambar 2-1 LBP-*based face description* [17]

2.2.2 Cascade Classifier dan Boosting

Cascade classifier adalah metode *boosted classifier*. *Cascade* terdiri dari sejumlah tahapan berturut-turut yang dilatih untuk menolak sampel yang tidak cocok atau sampel negatif dan tetap mempertahankan hampir seluruh sampel positif dan meloloskan sampel positif tersebut ke tahapan selanjutnya. Apabila sampel berhasil melewati tahap *cascade* terakhir tanpa ada penolakan, maka sampel tersebut dikatakan terdeteksi[21].



Gambar 2-3 Skema proses cascade of classifier [21]

Tahapan classifiers dibangun menggunakan *Adaptive Boosting* atau AdaBoost. Metode ini digunakan untuk menggabungkan beberapa *weak classifier* sehingga terbentuk *strong classifier* dengan tingkat akurasi yang bagus.

3. Perancangan Sistem

3.1 Spesifikasi Sistem

Sistem yang dibangun menggunakan input berupa data video yang akan diproses tiap *frame*. Untuk setiap frame dilakukan proses pemisahan *background* dan *foreground*. Selanjutnya *foreground* yang telah terpisah tersebut akan diproses untuk dicari muka orang dalam frame tersebut. Setelah muka orang terdeteksi, dicari titik tengah dari muka tersebut yang akan digunakan dalam penghitungan orang. Apabila titik tengah dari orang tersebut telah melewati garis, maka orang tersebut akan dihitung.

Sistem yang dibangun terdiri dari beberapa pemrosesan. Pertama sistem akan meminta inputan berupa video atau *image sequence*, setelah itu *frame* video akan disimpan dalam bentuk matrik. Isi dari matrik tersebut adalah informasi piksel pada frame tersebut. Informasi perubahan piksel *frame* sebelum dan *frame* setelahnya akan digunakan sebagai acuan menentukan objek. Setelah itu, proses *background subtraction* akan menentukan piksel mana yang termasuk dalam *background* atau termasuk dalam objek (*foreground*).

Setelah mendapatkan sekumpulan piksel objek, maka didapatkan *Region of Interest* (ROI) yang akan dideteksi apakah objek tersebut orang atau objek selain orang. Penentuan apakah objek tersebut adalah orang atau bukan dengan cara ekstraksi ciri menggunakan LBP. Hasil ekstraksi ciri yang didapat pada frame saat ini (*current frame*) akan dibandingkan dengan frame selanjutnya. Apabila *feature-feature* pada ciri orang tersebut sama (*match*), maka sistem akan menganggap orang tersebut orang yang sama. Setiap objek dan orang yang terdeteksi tiap frame ditampilkan sebagai output. Objek atau orang yang terdeteksi akan ditandai dengan kotak. Jika objek yang terdeteksi sebagai orang tersebut melewati garis perhitungan (LOI) maka sistem akan menghitung orang tersebut. Hasil perhitungan akan digunakan untuk tahapan analisis dan membantu perhitungan apakah objek yang terdeteksi orang yang benar (*true posisitve*) atau objek selain orang terdeteksi sebagai orang (*false positive*).

4. Pengujian dan Analisis

Sistem yang sudah dibangun akan diuji tingkat akurasi dalam penghitungan orang dan performa sistem. Akurasi diukur dari jumlah orang yang terhitung sistem dibandingkan dengan jumlah orang yang melewati garis hitung (LOI). Sedangkan performa sistem diukur dari rata-rata jumlah *frame* yang dapat diproses dalam satu detik. Namun sebelum dilakukan pengujian pada sistem secara keseluruhan, akan terlebih dahulu dilakukan pengujian pada paramater-paramater dari tiap bagian sistem yang berpengaruh dan akan dibandingkan tingkat akurasinya. Pengujian sistem keseluruhan menggunakan tujuh video uji dengan jumlah orang pada setiap video uji berbeda.

4.1 Pengujian LBP

Pada proses ini dilakukan pengujian parameter ukuran *sliding window* (*width* x *height*) pada saat proses *training* data. Ukuran *sliding window* ini selanjutnya akan digunakan untuk proses pendeteksian orang berdasarkan deteksi muka. Ukuran *sliding window* (*width* x *height*) yang diuji adalah 10x10 piksel, 12x12 piksel, dan 16x16 piksel. Pada pengujian ini digunakan delapan citra uji untuk proses deteksi orang berdasarkan akurasi deteksi sliding window. Perhitungan akurasi deteksi berdasarkan hasil deteksi objek per *frame* untuk setiap video menggunakan rumus :

$$\text{Akurasi} = \frac{\text{Jumlah orang terdeteksi}}{\text{Jumlah orang yang lewat}} \times 100\% \quad (4.1)[12]$$

Setiap citra uji akan dilakukan proses pendeteksian orang berdasarkan ukuran *sliding window* yang telah ditetapkan. Pada pengujian ini belum dilakukan proses *background subtraction*. Pada pengujian ini dilakukan pendeteksian wajah pada citra inputan untuk dihitung akurasi deteksi berdasarkan jumlah sliding window yang teridentifikasi (*True Positive, True Negative, False Positive,* dan *False Negative*). Pengujian ini bertujuan untuk memperoleh ukuran sliding window yang menghasilkan akurasi deteksi terbaik dari delapan data uji. Akurasi deteksi diukur menggunakan *confusion matrix* berdasarkan persamaan (4.1). Nilai *sliding window* terbaik akan digunakan untuk pengujian sistem keseluruhan.

Tabel 4-1 Tabel Pengujian LBP

	TP			TN			FN			FP		
	10x10	12x12	16x16	10x10	12x12	16x16	10x10	12x12	16x16	10x10	12x12	16x16
Citra 1 / background	0	0	0	768	540	300	0	0	0	0	0	0
Citra 2	0	1	0	59	26	23	1	0	1	0	0	0
Citra 3	0	0	0	43	39	23	1	1	1	0	0	0
Citra 4	1	1	1	49	35	31	0	0	0	0	0	0
Citra 5	0	1	0	90	44	44	1	0	1	0	0	0
Citra 6	0	1	1	64	26	23	1	0	0	0	0	0
Citra 7	0	0	0	83	54	35	1	1	1	0	0	0
Citra 8	0	0	0	78	66	32	0	0	0	0	0	0
Jumlah *sliding window*	1	4	2	1234	830	511	5	2	4	0	0	0

Berdasarkan tabel 4-1 dan persamaan (4.1) di atas, didapatkan akurasi deteksi dengan ukuran sliding window 10x10 menghasilkan akurasi sebesar 99,60%, akurasi deteksi dengan ukuran sliding window 12x12 menghasilkan akurasi sebesar 99,76%, dan akurasi deteksi dengan ukuran sliding window 16x16 menghasilkan akurasi sebesar 99,23%. Dari hasil tersebut didapat ukuran sliding window yang menghasilkan akurasi deteksi terbaik adalah 12x12 piksel.

4.2 Pengujian *Adaptive* GMM

Pada pengujian *Adaptive* GMM ini, parameter-parameter yang akan diuji adalah *Threshold* (TH) dan *Learning Rate* (LR). Pengujian TH dan LR ini akan dianalisis nilai TH dan LR berdasarkan tingkat kesalahan deteksi piksel dalam membedakan *background* dan *foreground*. Tingkat kesalahan tersebut didapat dengan membandingkan satu citra acuan dengan citra hasil *background subtraction* menggunakan *Adaptive* GMM. Citra acuan ini merupakan modifikasi manual pada capture frame ke-n dan akan dibandingkan dengan citra hasil ekstraksi *foreground* pada frame ke-n juga. Kesalahan deteksi dapat diukur dengan persamaan:

$$\text{Kesalahan deteksi} = \frac{\text{jumlah piksel salah terdeteksi}}{\text{jumlah piksel foreground}} \times 100\% \quad (4.2)$$

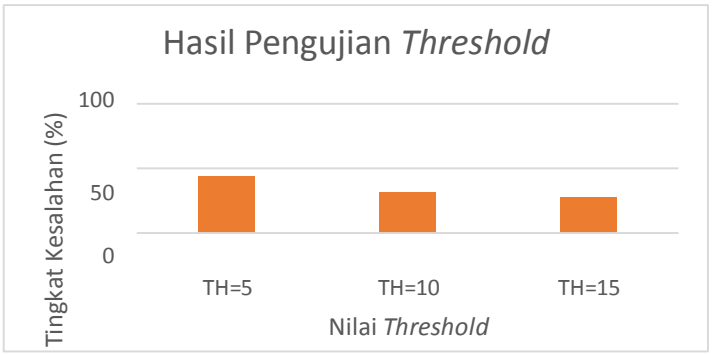
Piksel salah terdeteksi merupakan piksel-piksel yang salah dalam penentuannya (*miss*). Maksudnya adalah piksel yang seharusnya adalah piksel foreground tetapi malah terdeteksi sebagai piksel background dan juga sebaliknya.

Parameter *Threshold* (TH) berpengaruh terhadap hasil ekstraksi *foreground*. Nilai *Threshold* ini akan menentukan apakah suatu piksel termasuk model *background* atau *foreground*. Nilai *Threshold* dapat ditentukan dengan cara mencoba-coba beberapa nilai untuk dapat menghasilkan hasil ekstraksi *foreground* yang paling baik. Setelah itu citra hasil ekstraksi foreground tersebut akan dibandingkan dengan citra acuan pada setiap pixel untuk diukur tingkat kesalahan deteksi piksel *foreground*. Nilai parameter *Threshold* yang diuji adalah TH=5, TH=10, dan TH=15.

Pada citra hasil perbandingan, warna biru menunjukkan piksel yang merupakan piksel *foreground*, warna putih menunjukkan piksel *background* yang terdeteksi sebagai *foreground*, warna merah menunjukkan piksel yang seharusnya *foreground* tetapi terdeteksi sebagai piksel *background*, dan warna hitam merupakan piksel *background*. Tingkat kesalahan deteksi piksel *foreground* didapat dari perbandingan jumlah pixel putih ditambah jumlah piksel merah dengan jumlah piksel warna putih, merah, dan biru. Dari citra hasil perbandingan di atas didapat:

Tabel 4-2 Data hasil pengujian *Threshold*

	Jumlah Piksel Biru	Jumlah Piksel Putih	Jumlah Piksel Merah	Kesalahan Deteksi (%)
TH=5	9758	7427	98	43,54
TH=10	9557	4266	250	32,09
TH=15	9362	3164	398	27,56



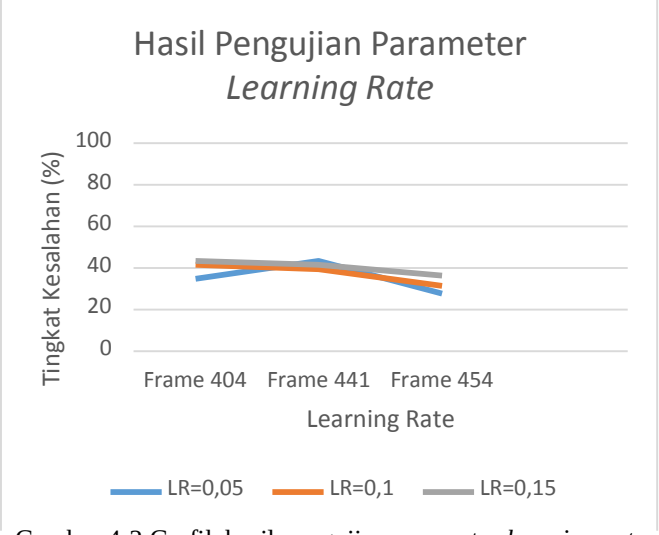
Gambar 4-1 Grafik hasil pengujian parameter *threshold*

Berdasarkan grafik di atas, nilai TH=5 menghasilkan tingkat kesalahan sebesar 43,54%, nilai TH=10 menghasilkan tingkat kesalahan sebesar 32,09%, dan nilai TH=15 menghasilkan tingkat kesalahan sebesar 27,56%. Nilai TH=15 menghasilkan tingkat kesalahan yang paling sedikit, tetapi kalau dilihat pada citra hasil ekstraksi *foreground*, terlihat ada piksel yang seharusnya merupakan piksel *foreground* tetapi terdeteksi sebagai piksel *background* (terlihat bolong pada bagian badan). Sedangkan pada nilai TH=10 *blob* yang dihasilkan lebih solid. Untuk nilai *threshold* di atas TH=15 masih terlihat ada kesalahan penentuan piksel seperti pada *threshold* = 15. Maka nilai threshold yang digunakan adalah TH=10.

Parameter yang diuji selanjutnya adalah parameter *Learning rate* (LR) yang berpengaruh pada kecepatan proses *update* suatu model. Untuk dapat menentukan nilai *learning rate* yang tepat, bergantung pada kondisi dinamis atau perubahan yang terjadi pada *background*. Perubahan yang cepat pada *background* membutuhkan nilai *learning rate* yang besar, sebaliknya perubahan yang lambat pada *background* membutuhkan *learning rate* yang kecil.

Pada video uji 3 ini, terdapat perubahan yang terjadi pada *background* yang disebabkan oleh terjadinya perubahan intensitas cahaya. Pada pengujian ini dilakukan percobaan dengan nilai LR=0,05, LR=0,1, dan LR=0,15. Nilai parameter TH=10 yang telah diujikan sebelumnya digunakan pada pengujian ini. Untuk pengujian *learning rate* sama seperti pengujian pada *threshold*. Pada pengujian ini akan dihitung tingkat kesalahan deteksi piksel foreground dan akan ditentukan nilai learning rate terbaik apabila memiliki tingkat kesalahan deteksi yang rendah.

Pada kondisi cahaya yang stabil pada frame 404, nilai *learning rate* 0,05 menghasilkan kesalahan deteksi piksel *foreground* yang paling sedikit, yaitu sebesar 34,93%. Pada kondisi cahaya yang berubah pada frame 441, nilai LR=0,1 menghasilkan kesalahan deteksi piksel foreground yang paling rendah, yaitu sebesar 39,53%. Pada saat kondisi cahaya sudah mulai stabil pada frame 454, nilai LR=0,05 menghasilkan tingkat kesalahan deteksi *foreground* yang paling rendah, yaitu sebesar 27,86%. Berdasarkan hasil pengujian, maka nilai parameter terbaik untuk parameter *learning rate* adalah LR=0,1 dapat dilihat dari grafik berikut.



Gambar 4-2 Grafik hasil pengujian parameter *learning rate*

4.3 Pengujian Sistem Keseluruhan

Tabel 4-3 Hasil Penghitungan Orang

Video Uji	Jumlah orang yang terhitung sistem	Jumlah orang yang melewati garis hitung	Akurasi Penghitungan Orang
1	1	1	100%
2	2	2	100%
3	3	3	100%
4	3	3	100%
5	3	5	60%
6	5	6	83,33%
7	6	6	100%

Sistem menghasilkan akurasi penghitungan orang sebesar 100% pada pengujian dengan video uji 1,2,3,4, dan 7. Pada video-video uji tersebut, jumlah orang yang terhitung sistem berjumlah sama dengan jumlah orang yang melewati garis hitung. Pada video uji keempat dan ketujuh, satu orang berjalan keluar area kamera sebelum garis hitung. Hal ini mengakibatkan tidak terhitungnya orang tersebut dalam sistem, padahal orang tersebut terdeteksi pada area sebelum garis hitung.

Selanjutnya akan dianalisis pengaruh jumlah orang terhadap jumlah frame yang bisa diproses dalam satu detik. Sistem masih dapat berjalan dengan baik pada saat diuji untuk video tujuh orang. Pada video lima orang, jumlah frame yang bisa diproses dalam satu detik turun sampai rata-rata frame rate sebesar 18,28 fps. Hal tersebut dikarenakan terdapat banyak kesalahan deteksi muka (*false alarm*) dan juga terjadi perubahan cahaya pada video. Pada video enam orang, jumlah frame yang bisa diproses dalam satu detik naik menjadi 21,56 fps karena kesalahan deteksi muka pada sistem menurun. Sistem dapat memproses data uji dengan sangat baik dan dapat berjalan secara *realtime* karena fps terendah yang didapat pada saat memproses data uji tidak di bawah batas minimal sistem untuk berjalan secara *realtime* yaitu 12 fps.

Tabel 4-4 Rata-rata fps pada video uji

Video Uji	Jumlah Orang	*Frame R*ate (fps)
1	1	29,03
2	2	26,56
3	3	24,20
4	4	25,97
5	5	18,28
6	6	21,56
7	7	21,42

Kelemahan dari sistem penghitungan orang yang dibangun ini adalah muka yang dideteksi untuk kemudian dihitung sebagai orang adalah pada proses deteksi muka sensitif pada pose muka orang pada video dan sistem belum dapat menangani muka yang tidak terlihat semua atau memiliki atribut yang menutupi sebagian muka, seperti penggunaan *masker*. Pengujian penghitungan orang dengan menggunakan atribut *masker* pada muka dilakukan. Pengujian ini menggunakan satu video uji baru yang di dalamnya terdapat orang yang menggunakan atribut *masker* yang menutupi muka bagian bawah sehingga muka tidak terlihat semua bagiannya. Skenario pengambilan video uji sama dengan pada tujuh video uji sebelumnya. . Muka yang tidak terdeteksi tersebut tidak akan bisa dihitung sebagai orang oleh sistem karena syarat untuk dapat dihitung sebagai orang adalah muka yang akan dihitung sebagai orang tersebut harus terdeteksi sebelum dan sesudah garis perhitungan. Pada gambar di atas, terlihat *counter* masih bernilai 0 pada setelah garis perhitungan, yang artinya sistem tidak menghitung adanya orang pada video uji tersebut.

Ketidakmampuan sistem dalam mendeteksi muka yang menggunakan atribut *masker* atau muka yang tidak terlihat semua bagiannya dikarenakan muka pada setiap frame pada video uji tersebut tidak lolos setiap tahapan *cascade*-nya sehingga histogram yang dihasilkan tidak sesuai dengan *threshold* yang telah ditentukan pada proses training citra positif dan negatif. Selain itu ciri muka yang menggunakan *masker* tidak sama dengan hasil ekstraksi ciri muka yang telah dilakukan pada proses *training* ciri muka sebelumnya.

5. Kesimpulan dan Saran

Berdasarkan pengujian yang telah dilakukan, dapat disimpulkan:

1. Berdasarkan pengujian deteksi wajah menggunakan fitur LBP, ukuran *sliding window* terbaik yang didapatkan berdasarkan pengujian adalah 12x12 dimana nilai *width*=12 piksel dan *height*=12 piksel. Akurasi deteksi yang didapat sebesar 99,76%.
2. Berdasarkan pengujian *background subtraction dengan Adaptive Gaussian Mixture Model* didapatkan nilai parameter terbaik antara lain parameter *Threshold* = 10 dan parameter *Learning Rate* = 0.1. Pada saat terjadi perubahan cahaya, nilai learning rate = 0,1 mampu menghasilkan blob yang baik. Untuk nilai threshold = 10 dipilih karena menghasilkan blob yang lebih solid daripada blob yang dihasilkan dari threshold = 15.

3. Untuk akurasi penghitungan orang pada sistem didapat 100% untuk video uji satu, dua, tiga, empat, dan tujuh, 60% untuk video uji lima, dan 83,33% untuk video uji enam.
4. Sistem yang dibangun dapat berjalan secara *realtime* sampai kasus tujuh orang berjalan bersama menuju kamera.

Saran dari penulis berdasarkan pengujian yang telah dilakukan adalah :

1. Perlu penambahan citra latih untuk variasi pose muka agar sistem dapat mendeteksi dengan lebih baik lagi.
2. Perlu penambahan metode yang dapat mendeteksi muka yang tidak terlihat semua atau memiliki atribut yang menutupi sebagian muka seperti masker dan lain sebagainya.

Daftar Pustaka

[1]	Berg, Roy-Erland. 2008. *"Real-Time People Counting System Using Video Camera"*. Norway: Gjøvik University College.
[2]	Bilainuk Olexa, Ehsan Fazl-Ersi, Robert Lagani'ere, Christina Xu, Daniel Laroche, Craig Moulder. 2014. *"Fast LBP Face Detection on Low-Power SIMD Architecture"*. CVPRW IEEE Conference 2014.
[3]	Brahnam Sheryl, Lakhmi C., Jain Loris Nanni, Alessandra Lumini. 2014. *"Local Binary Pattern: New Variants and Applications"*. Berlin: Springer
[4]	Hadid Abdenour, Pietikainen Matti, Ahonen Timo. 2004. *"A Discriminative Feature Space for Detecting and Recognizing Faces"*. Finland: University of Oulu.
[5]	Huang Di, Shan Caifeng, Ardabilian Mohsen, Wang Yunhang, Chen Liming. 2011. *"Local Biary Patterns and Its Application to Facial Image Analysis: A Survey"*. IEEE Transaction on Systems, Man, and Cybernetics-Part C: Applications and Reviews, Vol. 41, No. 6.
[6]	Kanan, Christopher dan Cottrell, Garrison W. 2012. *"Color-to-Grayscale: Does the Method Matter in Image Recognition?"*. California USA: PloS ONE.
[7]	Kumar Rakesh, Parashar Tapesh, Verma Gopal. 2012. *"Background Modeling and Substraction Based People Counting for Real Time Video Surveillance"*. International Journal of Soft Computing and Engineering (IJSCE).
[8]	Lowe, David G. 2004. "Distinctive Image Features from Scale-Invariant Keypoints". International Jurnal of Computer Vision
[9]	Munir, Rinaldi. 2004. "*Pengolahan Citra Digital*". Bandung: Informatika.
[10]	Ojala Timo, M. Pietikäinen, T.Mäenpaä. 2002. *"Multiresolution Gray-scale and Rotation Invariant Texture Classification with Local Binary Patterns"*. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 24, no. 7, pp. 971-987
[11]	Prakoso, Danu Hary. 2015. "*Perhitungan Orang dengan Metode Gaussian Mixture Model dan Human Presence Map*". Bandung: Universitas Telkom.
[12]	Primanda, Rodeztyan. 2013. "Penerapan dan Analisis Sistem Perhitungan Orang menggunakan Gaussian Mixture Model dengan Penghapusan Bayangan untuk Ekstraksi Foreground dan Pendekatan Model Energi Potensial". Bandung: Institut Teknologi Telkom.
[13]	Powers, David M.W. 2007. "Evaluation: From Precision, Recall and F-Factor to ROC, Informedness, Markedness & Correlation". Adelaide: School of Informatics and Engineering Flinders University.
[14]	Raheja J. L., Sishir Kalita, Pallab Jyoti Dutta, Solanski Lovendra. 2012. *"An Insight into The Algorithms on Real-Time People Tracking and Counting System"*. International Journal of Computer Applications (0975 – 8887) Volume 46– No.5.
[15]	Rodriguez, Yann. 2006. "Face Detection and Verification using Local Binary Patterns". Swiss: Ecole Polytechque Federale De Lausanne.
[16]	Rostianingsih Silvia, Rudy Adipranata, dan Fredy Setiawan Wibisono. 2008. *"Adaptive Background Dengan Metode Gaussian Mixture Models Untuk Real-Time Tracking"*. Jurnal Informatika Vol. 9 No. 1.
[17]	Sánchez López, Laura. 2010. *"Local Binary Patterns applied to Face Detection and Recognition"*. Spanyol: Universitat Politecnica De Catalunya.
[18]	Stauffer, Christ dan W.E.L., Grimson. 1999. "*Adaptive Background Mixture Model for Real-Time Tracking*". Pada Proceeding IEEE CVPR 1999, pp. 246-252.
[19]	Viola, P., Jones, M., Snow, D. 2003. "*Detecting Pedestrians Using Patterns of Motion and Appearance*". USA: Mitsubishi Electric Research Laboratories.
[20]	Viola, Paul dan Jones, Michael. 2001. "*Rapid Object Detection using a Boosted Cascade of Simple Features*". Accepted Conference on Computer Vision and Pattern Recognition.
[21]	Wang, Hanzi dan Sutter, David. 2005. "*A Re-evaluation of Mixture-of-Gaussian Background Modeling*". International Conference on Acoustics, Speech, and Signal Processing (ICASSP), Pennsylvania, USA, pages 1017-1020.
[22]	Zivkovic, Zoran. 2004. "*Improved Adaptive Gaussian Mixture Model for Background Subtraction*". Pada Proceeding ICPR.