ADAPTIVE BACKGROUND DENGAN METODE *GAUSSIAN MIXTURE MODELS* UNTUK *REAL-TIME TRACKING*

Silvia Rostianingsih[1], Rudy Adipranata[2], Fredy Setiawan Wibisono
Jurusan Teknik Informatika , Fakulas Teknologi Industri, Universitas Kristen Petra
Jl. Siwalankerto 121 – 131, Surabaya 60236
E-mail: silvia@peter.petra.ac.id[1], rudya@peter.petra.ac.id[2]

ABSTRAK: Saat ini, aplikasi *motion tracking* digunakan secara luas untuk banyak tujuan, seperti mendeteksi kemacetan dan menghitung berapa banyak orang yang masuk ke sebuah supermarket atau sebuah mall. Sebuah metode untuk memisahkan antara *background* dan obyek yang di-*track* dibutuhkan untuk melakukan *motion tracking*. Membuat aplikasi *tracking* pada *background* yang statis bukanlah hal yang sulit, namun apabila *tracking* dilakukan pada *background* yang tidak statis akan lebih sulit, dikarenakan perubahan *background* dapat dikenali sebagai area tracking. Untuk mengatasi masalah tersebut, dapat dibuat suatu aplikasi untuk memisahkan *background* dimana aplikasi tersebut dapat beradaptasi terhadap perubahan yang terjadi. Aplikasi ini dibuat untuk memisahkan background dengan menggunakan metode *Gaussian Mixture Models* (GMM). Metode GMM melakukan *cluster* data piksel dengan menggunakan warna *background* tiap piksel sebagai dasarnya. Setelah *cluster* dibentuk, dilakukan pencocokan input sebagai distribusi, dimana distribusi yang dominan dijadikan sebagai *background*. Aplikasi ini dibuat dengan menggunakan Microsoft Visual C++ 6.0. Hasil dari penelitian ini menunjukkan bahwa algoritma GMM dapat beradaptasi terhadap *background*. Hal ini dibuktikan dengan hasil pengujian yang sukses terhadap semua kondisi yang diberikan. Aplikasi ini dapat dikembangkan lebih lanjut supaya proses tracking dapat terintegrasi dengan *background* yang *adaptive*.

Kata kunci: *adaptive, background , Gaussian Mixture Models (GMM)*

ABSTRACT: Nowadays, motion tracking application is widely used for many purposes, such as detecting traffic jam and counting how many people enter a supermarket or a mall. A method to separate background and the tracked object is required for motion tracking. It will not be hard to develop the application if the tracking is performed on a static background, but it will be difficult if the tracked object is at a place with a non-static background, because the changing part of the background can be recognized as a tracking area. In order to handle the problem an application can be made to separate background where that separation can adapt to change that occur. This application is made to produce adaptive background using Gaussian Mixture Models (GMM) as its method. GMM method clustered the input pixel data with pixel color value as it's basic. After the cluster formed, dominant distributions are choosen as background distributions. This application is made by using Microsoft Visual C++ 6.0. The result of this research shows that GMM algorithm could made adaptive background satisfactory. This proofed by the result of the tests that succeed at all condition given. This application can be developed so the tracking process integrated in adaptive background maker process.

Keywords: adaptive, background , Gaussian Mixture Models (GMM)

PENDAHULUAN

Dewasa ini aplikasi *motion tracking* semakin banyak dipakai untuk berbagai tujuan, diantaranya adalah untuk mendeteksi kemacetan lalu lintas. Pada *motion tracking* diperlukan suatu metode untuk memisahkan antara *background* dengan obyek yang di-*tracking*, metode yang paling lazim digunakan adalah *background* substraction. Metode ini membutuhkan dua buah gambar yang memiliki *background* yang sama, dengan satu gambar berisi obyek yang di-*tracking*. Kemudian dilakukan *substraction* dan *threshold* antara kedua gambar tersebut. Hasilnya adalah suatu *foreground* yaitu gambar dimana hanya terdapat obyek yang bukan merupakan bagian dari *background*.

Metode di atas memiliki beberapa batasan, sebagai contoh jika terjadi perubahan pada *background* misalnya perubahan cahaya, penambahan dan/atau pengurangan obyek pada *background*. Hal-hal tersebut dapat menyebabkan *foreground* yang dihasilkan tidak optimal sehingga gambar *background* harus diatur ulang agar memenuhi kondisi sekarang.

Oleh karena itu dibutuhkan suatu *background* yang dapat beradaptasi dengan perubahan yang terjadi. Metode *adaptive background* yang umum adalah merata-rata gabungan nilai piksel dari semua input yang berurutan. Namun metode ini tidak dapat

berjalan dengan baik jika diberi input yang memiliki banyak obyek bergerak. Untuk menutupi kekurangan ini terdapat metode *adaptive background mixture models* yang menggunakan *Gaussian Mixture Models*. Pada penelitian ini dibuat perangkat lunak untuk menghasilkan *adaptive background* dengan menggunakan metode *Gaussian Mixture Models*. Perumusan masalah yang dihadapi adalah bagaimana menghasilkan suatu *adaptive background* yang stabil terhadap perubahan cahaya dan perubahan *scene* jangka panjang dan bagaimana pengolahan input yang berasal dari webcam. Untuk proses *Gaussian Mixture Models* ini umumnya metode *clustering* yang digunakan adalah *Expectation Maximization* dan K-means. Metode *clustering* yang digunakan pada kasus ini adalah K-means karena metode ini memiliki kecepatan proses yang lebih cepat dan hasil yang cukup baik.

ADAPTIVE BACKGROUND

Sebuah *background* adalah suatu gambar yang di dalamnya terdapat obyek-obyek yang tidak bergerak (obyek static). Suatu gambar yang berisi obyek yang bergerak disebut *foreground*. Pada aplikasi-aplikasi yang membutuhkan input berupa *background* untuk memisahkan antara *background* dan *foreground* dibutuhkan suatu inisialisasi ulang apabila terdapat perubahan pada *background*. Bila tidak dilakukan inisialisasi ulang, mengakibatkan kesalahan pada *foreground* yang dihasilkan. Perubahan-perubahan yang mungkin terjadi pada *background* adalah perubahan intensitas cahaya dari siang hari ke sore hari, perubahan bayangan benda yang terdapat pada *background* yang diakibatkan oleh perubahan posisi matahari, perubahan posisi benda pada *background*, penambahan benda dalam *background*, dan sebagainya. Untuk mengurangi kesalahan yang terjadi karena perubahan *background* maka diperlukan suatu *adaptive background* yaitu *background* yang dapat menyesuaikan dengan perubahan-perubahan yang terjadi pada *background*.

Metode *adaptive background* yang umum adalah merata-rata gabungan nilai piksel dari semua input yang berurutan [1]. Namun metode ini tidak dapat berjalan dengan baik jika diberi input yang memiliki banyak obyek bergerak. Untuk menutupi kekurangan ini terdapat metode *adaptive background* mixture models yang menggunakan *Gaussian Mixture Models.*

Untuk proses *Gaussian Mixture Models* ini umumnya metode *clustering* yang digunakan adalah *Expectation Maximization* dan K-means. Dalam penelitian ini, sebagai alat bantu input dipakai kamera USB (*webcam*) dengan *frame rate* sekitar 10-20 frame/second, sehingga dibutuhkan metode yang cukup efisien dan cepat karena proses dilakukan pada tiap *frame* yang masuk. Karena tiap detik terdapat 10-20 *frame* yang diolah, maka Metode *clustering* yang digunakan pada kasus ini adalah K-means karena metode ini memiliki kecepatan proses yang lebih cepat dan hasil yang cukup baik.

GAUSSIAN MIXTURE MODEL (GMM)

Gaussian Mixture Model (GMM) adalah sebuah tipe *density model* yang terdiri dari komponen fungsi-fungsi Gaussian [2]. Komponen fungsi ini terdiri dari *weight* yang berbeda untuk menghasilkan *multi-model density*. Pada penelitian ini GMM digunakan untuk memodelkan warna-warna *background* dari tiap piksel.

Tiap piksel memiliki GMM-nya sendiri dan data yang diolah adalah warna piksel yang didapat dari input. Model-model GMM terbentuk dari data warna piksel berdasarkan waktu. Model yang terbentuk dibagi menjadi 2 bagian, model *background* dan model *non-background*. Model *background* adalah model yang mencerminkan *background*.

Jumlah model GMM yang digunakan mempengaruhi jumlah model *background*. Semakin besar jumlah model GMM yang dipakai semakin banyak model *background* yang dimiliki oleh suatu piksel.

Terdapat beberapa tahap proses untuk metode ini yaitu tahap pencocokan input terhadap distribusi dan tahap pemilihan distribusi yang mencerminkan *background*. Di dalam tahap pencocokan terdapat tahap *update* parameter.

Tahap Pencocokan Input terhadap Distribusi

Pada tahap ini input dicocokkan dengan semua distribusi sampai ditemukan distribusi yang paling cocok. Suatu piksel dikatakan masuk dalam suatu distribusi jika nilai piksel tersebut masuk dalam jarak 2.5 standar deviasi dari sebuah distribusi.

$$\mu_k - 2.5 * \sigma_k < X_t < \mu_k + 2.5 * \sigma_k \quad (1)$$

dimana Xt adalah vector dari warna piksel (R,G,B) [3] pada waktu t, μk adalah vector nilai *mean* (R,G,B) dari Gaussian ke k^{th}, dan σ k sebagai standar deviasi dari Gaussian ke k^{th} [4]. Apabila piksel tidak cocok dengan semua distribusi yang ada maka piksel tersebut dianggap sebagai *foreground* dan dibuat suatu distribusi baru dengan menggantikan distribusi yang paling tidak mencerminkan *background*. Distribusi baru memiliki nilai *mean* sesuai dengan nilai piksel, nilai *varian*s yang tinggi, dan nilai *weight* yang kecil.

Nilai awal yang diberikan pada variabel tersebut sangat mempengaruhi performa dari algoritma GMM.

Tahap Update Parameter

Pada tahap ini dilakukan *update* terhadap nilai dari parameter-parameter GMM yang nantinya digunakan untuk mengolah input selanjutnya. Nilai yang di-*update* terdiri dari *weight*, *mean*, dan *varian*. Nilai *weight* di-*update* tiap waktu. Untuk meng-*update* nilai *weight* digunakan rumus [1]:

$$\omega_{k,t} = (1-\alpha)\omega_{k,t-1} + \alpha(M_{k,t}) \quad (2)$$

dimana ωk,t adalah *weight* dari Gaussian ke kth pada waktu t, α adalah *learning rate* dan nilai Mk,t adalah 1 untuk model yang cocok dan 0 untuk model lainnya. Setelah nilai *weight* di-*update* dilakukan normalisasi sehingga total *weight* dari semua distribusi tidak lebih dari 1.

Nilai *mean* dari suatu distribusi di-*update* setiap ada nilai piksel yang cocok dengan distribusi tersebut. Untuk meng-*update* nilai *mean* digunakan rumus [1]:

$$\mu_t = (1-\rho)\mu_{t-1} + \rho X_t \quad (3)$$

dimana

$$\rho = \alpha\eta(X_t|\mu_k, \Sigma_k) \quad (4)$$

dimana η adalah fungsi Gaussian Probability Density (GPD)

$$\eta(X \mid \mu, \Sigma) = \frac{1}{(2\pi)^{\frac{n}{2}}|\Sigma|^{\frac{1}{2}}} e^{-\frac{1}{2}(X-\mu)^T \Sigma^{-1}(X-\mu)} \quad (5)$$

dimana Σ adalah co*varian*ce matrix, $|\Sigma|$ adalah *determinant* dari co*varian*ce, pangkat T adalah *transpose* matrix, pangkat -1 adalah invers matrix, e adalah exponen, π adalah phi, dan n adalah ukuran vector X (R,G,B) dimana co*varian*ce didapat dari [1]:

$$\Sigma_{k,t} = \sigma_k^2 \mathrm{I} \quad (6)$$

dimana I adalah matrix identitas dan σ_k^2 adalah *varian*s dari Gaussian ke kth.

Nilai standar deviasi dari suatu distribusi di-*update* setiap ada nilai piksel yang cocok dengan distribusi tersebut. Untuk meng-*update* nilai standar deviasi digunakan rumus [1]:

$$\sigma_t^2 = (1-\rho)\sigma_{t-1}^2 + \rho(X_t - \mu_t)^T(X_t - \mu_t) \quad (7)$$

Tahap Pemilihan Distribusi Background

Pada tahap ini dipilih model-model yang mencerminkan *background*. Pertama model-model diurutkan berdasarkan ω/σ2 sehingga distribusi yang paling mencerminkan *background* tetap di atas dan yang tidak mencerminkan *background* ada di bawah yang nantinya digantikan oleh distribusi yang lain. Untuk memilih B distribusi pertama yang dijadikan distribusi *background* digunakan rumus:

$$B = \arg\min_b \left(\sum_{k=1}^{b} \omega_k > T\right) \quad (8)$$

dimana T adalah proporsi terkecil dari data yang sebaiknya dihitung sebagai *background* [1].

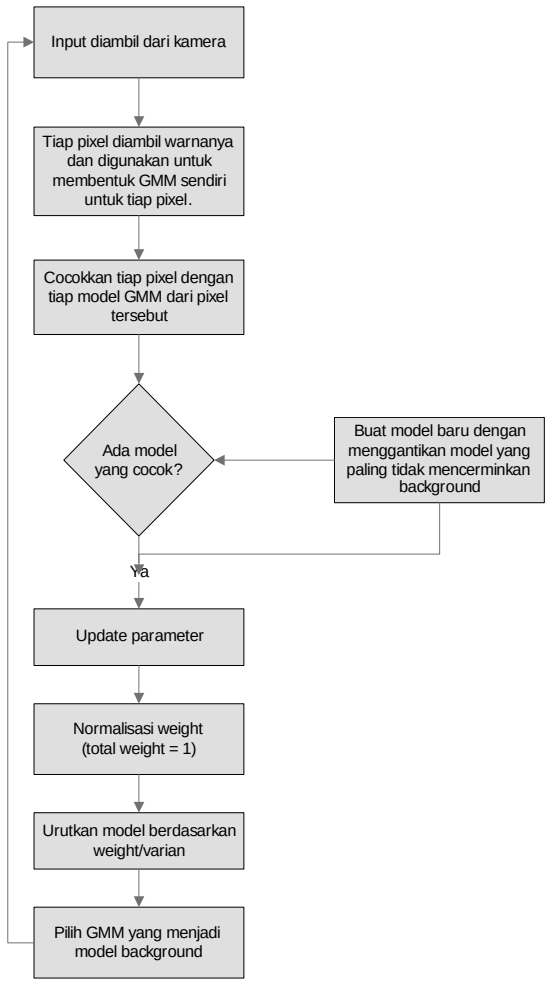
PENELITIAN

Perencanaan sistem aplikasi pembuatan *adaptive background* menggunakan metode GMM terdiri dari beberapa bagian yang meliputi:

- Perencanaan sistem input
  Pengambilan *input* hanya dapat dilakukan bila terdapat kamera yang terhubung dengan perangkat lunak. Perangkat lunak mengambil gambar yang ditangkap oleh kamera dan menyimpannya dalam sebuah *class* CImage yang tersedia dalam *library* Highgui [5] untuk selanjutnya dilakukan proses pada gambar tersebut. Berhubung perangkat lunak berjalan secara *real-time* maka diperlukan sebuah kamera (*webcam*) untuk pengambilan data gambar [6]. Jadi bila tidak terdapat kamera yang terhubung dengan perangkat lunak maka tidak terdapat pengambilan data gambar sama sekali. Setelah data gambar didapatkan dan disimpan maka data gambar tersebut ditampilkan.
- Perencanaan sistem pencocokkan terhadap distribusi
  Input yang masuk dicocokkan dengan semua distribusi yang ada sampai ditemukan distribusi yang cocok atau semua distribusi sudah dicocokkan. Apabila terdapat distribusi yang cocok maka dilakukan proses *update*, bila tidak terdapat distribusi yang cocok maka dibuat distribusi baru kemudian dilanjutkan dengan proses *update*. Input dikatakan cocok atau masuk suatu distribusi bila input masuk dalam jarak 2.5 standar deviasi dari sebuah distribusi. Untuk input yang tidak cocok dengan semua distribusi maka input tersebut masuk sebagai *foreground*. Kemudian dibuat distribusi baru yang menggantikan distribusi yang paling tidak mencerminkan *background*.
  Untuk input yang cocok dengan salah satu distribusi namun distribusi tersebut bukan distribusi *background* maka input tersebut juga masuk sebagai *foreground*.

- Perencanaan sistem *update* parameter
  Input yang sudah dicocokkan dengan distribusi digunakan untuk meng-*update* parameter-parameter GMM yang digunakan untuk input berikutnya. Jika suatu distribusi cocok dengan input maka *mean* dan *varian* dari distribusi tersebut di-*update* dan untuk distribusi yang tidak cocok, *mean* dan *varian*nya tidak di-*update*. *Weight* dari semua distribusi di-*update* setiap kali ada input, baik untuk distribusi yang cocok maupun yang tidak cocok.
- Perencanaan sistem pemilihan distribusi *background*
  Setelah dilakukan proses pencocokkan dan *update* parameter maka yang tersisa adalah proses pemilihan distribusi yang mencerminkan *background*. Distribusi *background* yang dipilih bisa lebih dari satu namun yang ditampilkan hanya yang berada di urutan teratas.

Gambar 1 adalah blok diagram dari sistem kerja perangkat lunak.



Gambar 1. Blok Diagram Sistem

PENGUJIAN SISTEM

Pengujian dilakukan dengan menggerakkan obyek yang ada pada *background*, menambahkan obyek pada *background*, mengurangi obyek pada *background*, menambah intensitas cahaya, mengurangi intensitas cahaya, memberikan bayangan pada *background*, dan memberi *foreground*. Pengujian sistem di sini terdiri dari pengujian terhadap kemampuan beradaptasi dan akurasi. Akurasi berhubungan dengan ketepatan hasil *foreground* dan *background* yang dihasilkan. Berikut ini adalah beberapa pengujian yang telah dilakukan.

Pengujian dengan Menggerakkan Obyek

Pengujian dilakukan dengan menggerakkan obyek yang ada pada *background* dan obyek merupakan obyek yang dominan pada *background*. Pada Gambar 2, Gambar 3, dan Gambar 4 dapat dilihat bahwa *background* dapat beradaptasi dengan baik walaupun perubahan yang dilakukan terhadap *background* cukup significant yaitu memindahkan obyek yang mendominasi *background*.

Pengujian dengan Menambahkan Obyek pada *Background*

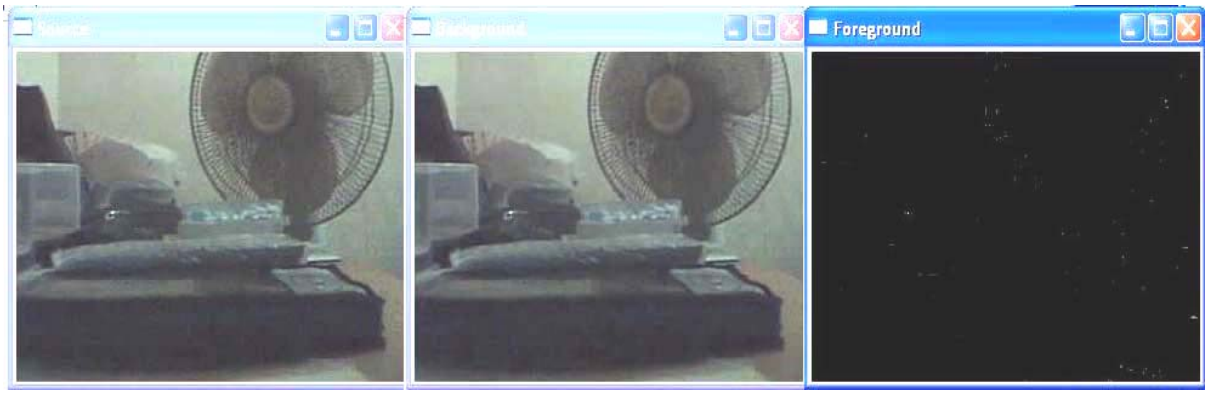
Pengujian dilakukan dengan menambahkan obyek pada *background*. Pada Gambar 5, Gambar 6, dan Gambar 7 dapat dilihat bahwa *background* dapat beradaptasi dengan baik terhadap penambahan obyek pada background. Foreground yang dihasilkan pada Gambar 6 kurang sesuai dengan benda yang ditambahkan disebabkan oleh kemiripan warna pada bagian-bagian tertentu sehingga pada pixel dimana terjadi kemiripan warna, piksel tersebut dianggap sebagai *background*.

Pengujian dengan Mengurangi Obyek pada *Background*

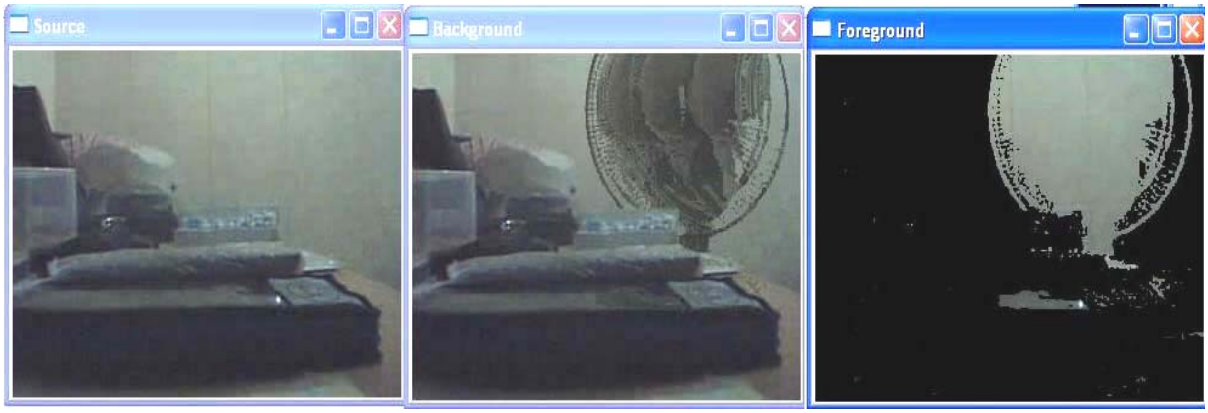
Pengujian dilakukan dengan mengurangi obyek pada *background*. Pada Gambar 8, Gambar 9, dan Gambar 10 dapat dilihat bahwa *background* dapat beradaptasi dengan baik terhadap pengurangan obyek pada background. Foreground yang dihasilkan pada Gambar 9 diakibatkan karena obyek yang tadinya *background* dipindahkan sehingga piksel dimana obyek tersebut sebelumnya berada dianggap sebagai *foreground* untuk beberapa waktu.

Pengujian dengan Menambah Intensitas Cahaya

Berikut ini merupakan hasil pengujian dengan menambahkan intensitas cahaya, dalam kasus ini



Gambar 2. Kondisi Sebelum Pergerakan Obyek *Background*



Gambar 3. Kondisi Saat Pergerakan Obyek *Background*

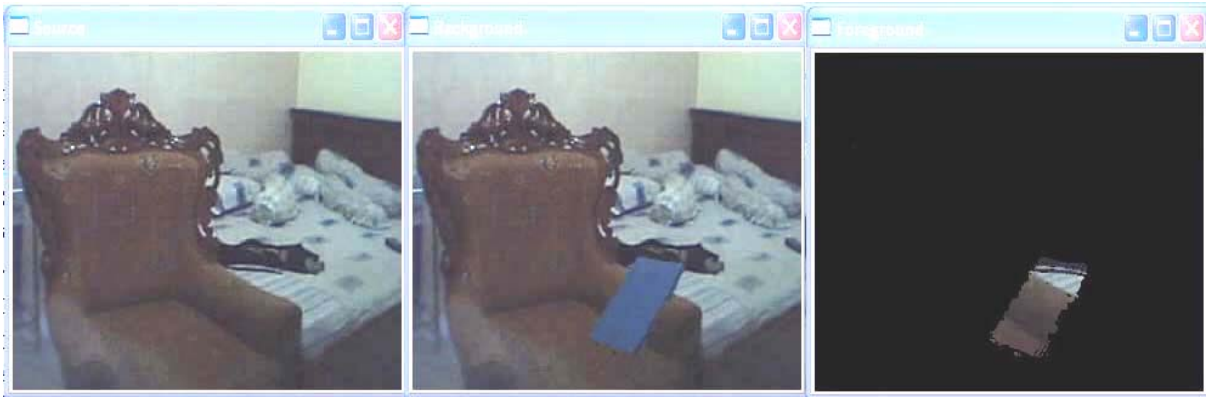
Gambar 4. Kondisi Setelah Pergerakan Obyek *Background*

Gambar 5. Kondisi Sebelum Penambahan Obyek

Gambar 6. Kondisi Saat Penambahan Obyek

Gambar 7. Kondisi Setelah Penambahan Obyek

Gambar 8. Kondisi Sebelum Pengurangan Obyek

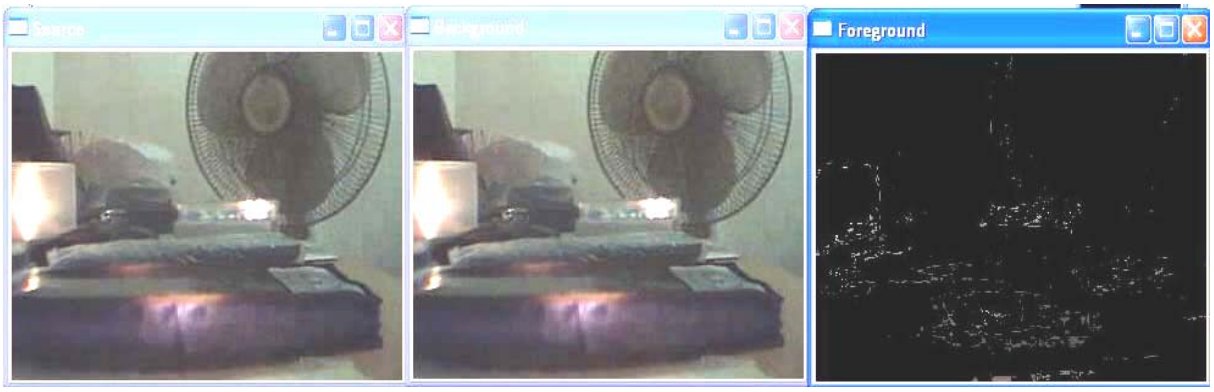


Gambar 9. Kondisi Saat Pengurangan Obyek

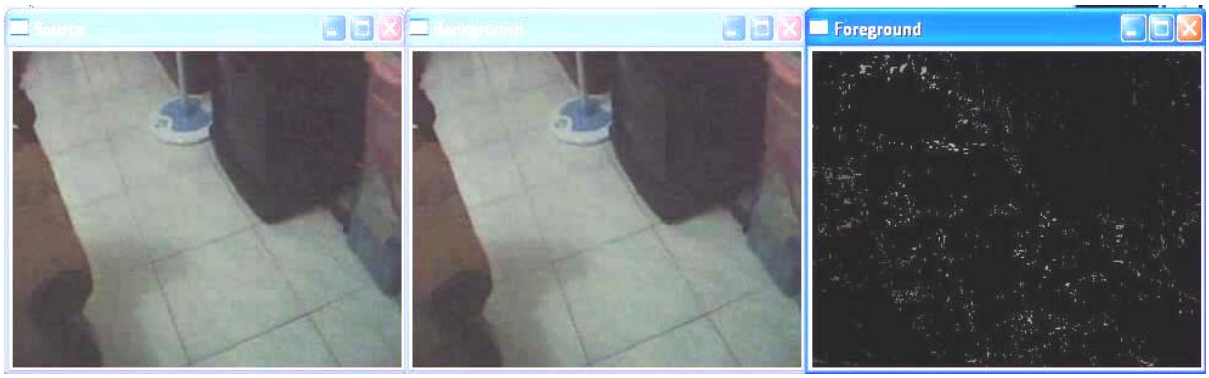
Gambar 10. Kondisi Setelah Pengurangan Obyek

Gambar 11. Kondisi Sebelum Penambahan Intensitas Cahaya

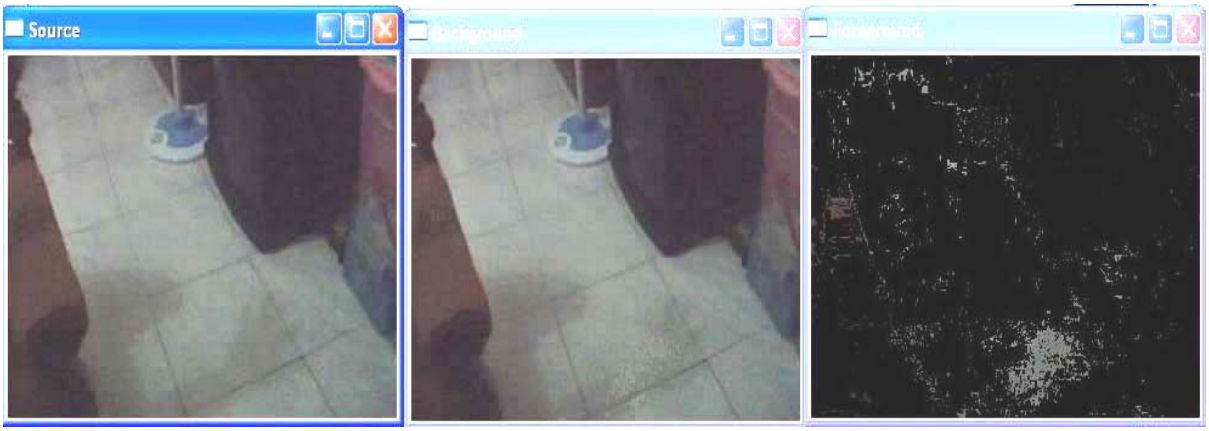
Gambar 12. Kondisi Saat Penambahan Intensitas Cahaya



Gambar 13. Kondisi Setelah Penambahan Intensitas Cahaya



Gambar 14. Kondisi Sebelum Pemberian Bayangan



Gambar 15. Kondisi Saat Pemberian Bayangan

Gambar 16. Kondisi Setelah Pemberian Bayangan

Gambar 17. Kondisi Sebelum Pemberian foreground I

Gambar 18. Kondisi Saat Pemberian foreground I

Gambar 19. Kondisi Setelah Pemberian foreground I

untuk menambah intensitas cahaya digunakan cahaya dari lampu senter yang dinyalakan dan diarahkan ke tempat DVD yang berada pada daerah kiri. Pada Gambar 11, Gambar 12, dan Gambar 13 dapat dilihat bahwa *background* dapat beradaptasi dengan baik terhadap penambahan intensitas cahaya yang dilakukan dengan lampu senter. *Foreground* yang dihasilkan pada Gambar 6 diakibatkan karena penambahan intensitas cahaya pada piksel-piksel tertentu mengakibatkan warna dari piksel tersebut keluar dari distribusi *background* sehingga dianggap sebagai *foreground*.

Pengujian dengan Memberi Bayangan pada *Background*

Pengujian dilakukan dengan memberi bayangan pada *background*. Pada Gambar 14, Gambar 15, dan Gambar 16 dapat dilihat *background* beradaptasi dalam waktu kurang lebih 1 detik dari saat munculnya bayangan, hal ini dikarenakan bayangan hanya memberikan perubahan sedikit pada intensitas cahaya sehingga *background* dapat beradaptasi dengan sangat cepat

Pengujian dengan Memberi Fore*ground*

Pengujian dilakukan dengan memberikan *foreground* pada input. *Foreground* yang diberikan adalah obyek berupa *map* yang dimasukkan pada layar input. Obyek ditempatkan pada *background* yang memiliki warna yang mirip.

Pada Gambar 17, Gambar 18, dan Gambar 19 dapat dilihat bahwa *foreground* yang dihasilkan tidak terlalu baik, hal ini disebabkan obyek yang dilewati oleh *foreground* memiliki warna yang mirip yaitu warna biru. Sehingga piksel-piksel dimana terdapat obyek tersebut tidak masuk sebagai *foreground* melainkan masuk pada distribusi *background*.

Pengujian Kecepatan Adaptasi terhadap K

Berikut ini adalah pengujian kecepatan adaptasi terhadap k, k yang diuji coba adalah 3 dan 5.

k = 3 beradaptasi pada frame ke-20

k = 5 beradaptasi pada frame ke-35

Berdasarkan hasil pengujian yang dilakukan, semakin sedikit nilai k yang digunakan maka semakin cepat adaptasi terhadap suatu perubahan namun semakin sedikit model *background* yang dimiliki.

Berdasarkan hasil pengujian yang dilakukan dapat dilihat bahwa algoritma ini memiliki kelebihan yaitu dapat beradaptasi terhadap semua jenis perubahan yang ada pada *background*. Namun setelah perangkat lunak berjalan kurang lebih 30 detik terdapat *noise* pada *foreground* yang kemungkinan penyebabnya adalah hasil tangkapan dari kamera yang kurang akurat dalam hal perubahan intensitas cahaya.

KESIMPULAN

Berdasarkan hasil pengujian dapat disimpulkan beberapa hal sebagai berikut:

1. *Foreground* yang dihasilkan baik selama warna *background* tidak mirip dengan warna *foreground*.
2. Berdasarkan semua hasil pengujian yang dilakukan *background* dapat beradaptasi dengan baik terhadap semua pengujian yang dilakukan.
3. Walaupun bayangan dapat diadaptasi oleh *background* dalam kurun waktu kurang dari 1 detik namun bila bayangan tersebut berubah bentuk atau bergerak secara terus-menerus maka bayangan tersebut mengganggu *foreground* yang dihasilkan.

DAFTAR PUSTAKA

1. Stauffer, C., Grimson, W.E.L (1999). *Adaptive Background Mixture Model for Real Time Tracking*. 10 Desember 2007, http://www.ai.mit.edu/projects/vsam/Publications/stauffer cvpr98_track.pdf.
2. Gu, Juan, Jun Chen, Qiming Zhou, Hongwei Zhang. *Gaussian Mixture Model of Texture for Extracting Residential Area from High-Resolution Remotely Sensed Imagery*, ISPRS Workshop on Updating Geo-spatial Databases with Imagery & The 5th ISPRS Workshop on DMGISs, China, Agustus 2007, pp.157-162.
3. Jacinto, N., Jorge, S.M. (2004). *New Performance Evaluation Metrics for Object Detection Algorithms*. 20 Maret 2008, http://homepages. inf.ed.ac.uk/rbf/ CAVIAR/PAPERS/WECCV.pdf.
4. Rui, T., Hong, H., Jin, Q., Tao, F. (2006). *Traffic Video Segmentation Using Adaptive-k Gaussian Mixture Mode*l. 15 April 2008, http://www.cs.cityu.edu.hk/~tanrui/pub/akgmm.pdf.
5. Chien, Chao C., 2002 *Professional Software Development with Visual C++ 6.0 & MFC*, Massachusetts: Charles River Media Inc.
6. Wren, Christopher, R., Ali , A., Trevor, D., Alex, P. (1997). *Pfinder: Real-time Tracking of The Human Body*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 19 (7), 780-785.