

**UAS COMPUTER VISION
MEMBUAT 3 RESUME JURNAL TERKAIT COMPUTER VISION**



Oleh

NAMA : KAFA NI'MAL MAULA

NIM : 18041037

KELAS: G

DOSEN: RASYID MUSTAFA M.Kom

**D III TEKNIK KOMPUTER
POLITEKNIK HARAPAN BERSAMA KOTA TEGAL
TAHUN 2020**

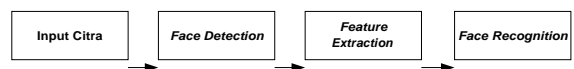
Sistem Pengenalan Wajah Secara *Real-Time* dengan *Adaboost*, *Eigenface* PCA & MySQL

Dodit Suprianto, Rini Nur Hasanah, Purnomo Budi Santosa

Abstrak – Aplikasi sistem pengenalan wajah secara *real time* dapat ditemukan pada sistem pengawasan, identifikasi dan keamanan yang berbasis pada pengenalan wajah. Pengamatan wajah secara langsung oleh manusia mempunyai kelemahan karena kelelahan dan kejenuhan yang mungkin terjadi dapat menyebabkan menurunnya ketelitian. Untuk itu penggunaan komputer dapat menjadi alternatif solusi. Pada penelitian ini pengenalan wajah dilakukan melalui tahap *face detection*, *feature extraction* dan *face recognition*, selanjutnya dicocokkan dengan data profil yang tersimpan di dalam *database*. Pendeteksian wajah menggunakan metode *Adaboost*, pengenalan wajah menggunakan metode *Eigenface PCA* dan *database MySQL* untuk menyimpan informasi profile. Penggunaan metode tersebut untuk pengenalan wajah pada kondisi real time dengan perbedaan jarak antara sensor dan wajah, posisi wajah, intensitas cahaya yang mengenai wajah, mimik muka dan atribut wajah dalam penelitian ini memberikan tingkat keberhasilan sebesar 80% dalam mengidentifikasi wajah.

Kata Kunci—*Face detection*, *Face recognition*, *Adaboost*, *Eigenfaces PCA*, *MySQL*.

Pengenalan wajah diperlukan sebagai alat pengawasan, penandaan otomatis (*automatic tagging*) dan interaksi robot dan manusia [3]. Berdasarkan beberapa penelitian sebelumnya maka peneliti mengangkat topik sistem pengenalan wajah secara *real-time*. Pada gambar 1 ditunjukkan blok diagram metode pengenalan wajah yang tersusun tiga bagian: *face detection*, *feature extraction* dan *face recognition*.

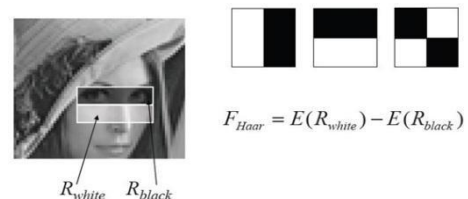


Gambar 1. Blok diagram pengenalan wajah

A. *Face Detection*

Face detection pada penelitian ini menggunakan metode yang dikemukakan oleh Viola & Jones [1], terbagi menjadi 4 komponen utama: *Haar Like Feature*, *Integral Image*, *Adaptive Boosting* dan *Cascade of Classifier*.

▪ *Haar Like Feature*



Gambar 2. Skema kerja *Haar Like Feature*

Gambar 2 menunjukkan skema *Haar Like feature* [4] yang memproses citra dalam wilayah kotak-kotak, berisi beberapa *pixel* dari bagian citra. Pixel-pixel dalam satu wilayah tersebut dijumlahkan dan dilakukan proses perhitungan (pengurangan rata-rata nilai *pixel* di bagian kotak yang terang dan gelap) sehingga diperoleh perbedaan nilai unik disetiap wilayah kotak-kotak tersebut.

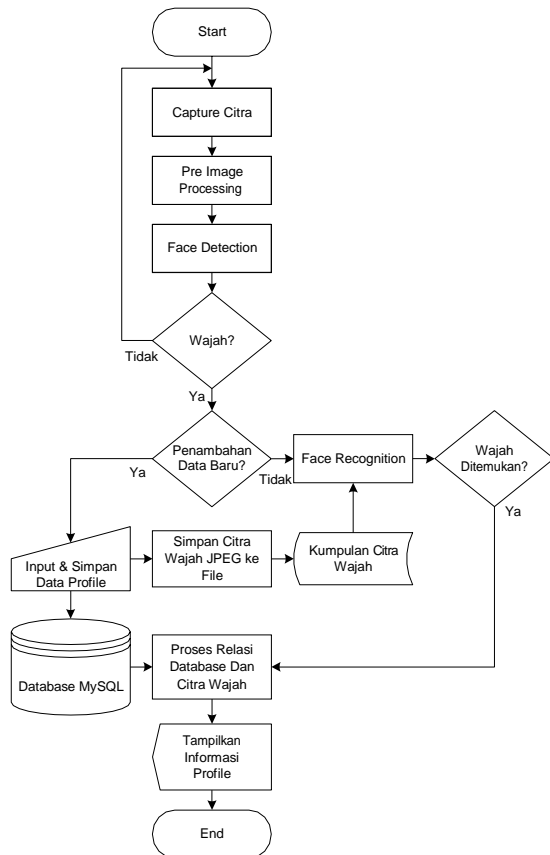
I. METODE PENGENALAN WAJAH SECARA REAL TIME Pada gambar 5 menunjukkan diagram alir aplikasi sistem pengenalan wajah yang dibangun, yang dapat dijelaskan sebagai berikut ini:

Citra diperoleh dari *sensor webcam*, citra warna RGB dikonversi menjadi citra *grayscale* untuk mengurangi kedalaman warna.

Dari citra *grayscale* dilakukan pendeteksian wajah dengan menggunakan metode *Adaboost*, jika citra tidak terdeteksi sebagai wajah maka lakukan penangkapan citra oleh sensor webcam secara berulang.

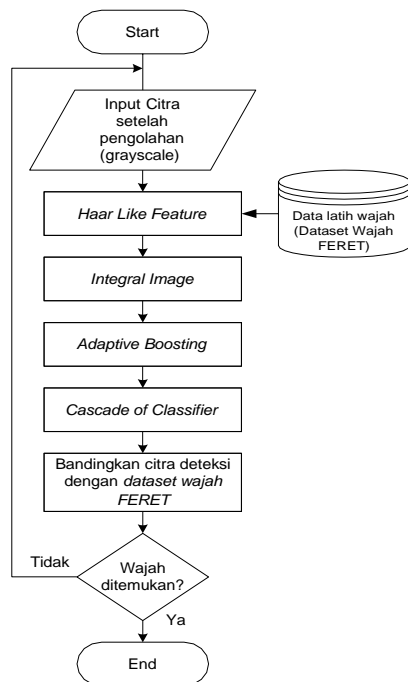
Jika bagian dari citra terdeteksi sebagai wajah oleh mesin *face detection* maka lanjutkan dengan dua opsi

- Jika sebelumnya citra wajah dan data profile telah tersimpan di dalam database maka lakukan identifikasi wajah dengan metode *Eigeface PCA*.

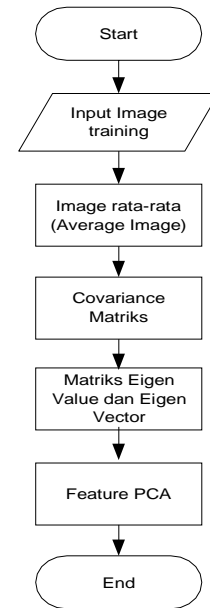


Gambar 5. Alur Kerja Pengenalan Wajah

Gambar 6 menunjukkan diagram alir pendeteksian wajah (*face detection*) dilakukan dengan mengklasifikasi sebuah citra, setelah sebelumnya sebuah pengklasifikasi dibentuk dari data latih.



Gambar 6. Algoritma Face Detection



Gambar 7. Algoritma Face Recognition

Gambar 7 memberikan algoritma *Face Recognition* [4], yang merupakan langkah pertama untuk mendapatkan satu set S dengan M citra wajah. Sebagai contoh: $M = 25$. Setiap gambar ditransformasi menjadi sebuah vektor berukuran N dan ditempatkan ke dalam set.

$$\square (4)$$

Setelah mendapatkan set, maka akan diperoleh gambar rata-rata (*mean image*) Ψ .

$$- (5)$$

Kemudian diperoleh perbedaan Φ antara citra input dan citra rata-rata.

$$(6)$$

Selanjutnya mencari set M vektor ortonormal, u_n , yang paling menggambarkan distribusi data. Vektor K_{th} , u_k , dipilih sedemikian rupa.

$$- (7)$$

Adalah maksimal untuk subject

$$(8)$$

Catatan: u_k and λ_k adalah *eigenvectors* dan *eigenvalues* dari kovarian matriks C .

Selanjutnya diperoleh matriks kovarians C dengan cara berikut

$$- (9)$$

$$(10)$$

$$(11)$$

$$(12)$$

Sehingga diperoleh *eigenvector*, v_l, u_l

$$(13)$$

Prosedur pengenalan wajah pada *Eigenfaces Principal Component Analysis* adalah sebagai berikut:

- Mengubah wajah baru menjadi komponen eigenface.

Pertama, gambar masukan dibandingkan dengan gambar rata-rata dan perbedaan mereka dikalikan dengan masing-masing vektor *eigen* dari matriks *L*. Setiap nilai akan mewakili bobot dan akan disimpan pada vektor Ω .

(14)

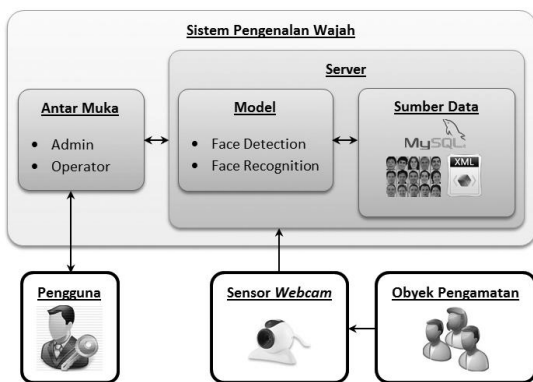
- Menentukan kelas wajah mana yang memberikan gambaran terbaik untuk citra masukan. Hal ini dilakukan dengan meminimalkan jarak *Euclidean*.

(15) Wajah input

mempertimbangkan penggolongan kelas. Jika ε_k berada di bawah ambang/*threshold* θ_ε maka citra merupakan wajah yang dikenali. Jika perbedaan berada di atas *threshold* yang diberikan, tetapi di bawah *threshold* kedua, maka citra merupakan wajah yang tak dikenali. Jika citra masukan bukan bagian dari kedua *threshold* di atas maka citra merupakan bukan wajah.

II. HASIL DAN ANALISIS

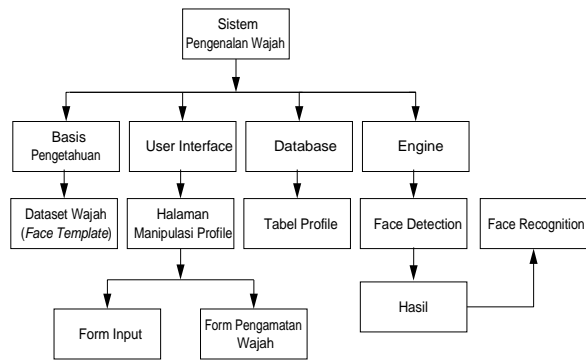
Pada Gambar 8 ditunjukkan arsitektur pengenalan wajah, yang terdiri dari dua bagian utama, yaitu antar muka dan server. Antar muka adalah tampilan awal program berupa otoritas hak akses bagi pengguna. Dalam *server* terdapat dua bagian utama yaitu model dan sumber data, model adalah suatu metode atau algoritma pengenalan wajah, terdiri dari *face detection* dan *face recognition* yang berhubungan dengan sumber data berupa *dataset* wajah *FERET* (berupa file *XML*) saat dilakukan pendeteksian wajah, data profil yang disimpan dalam *database MySQL* dan berkas wajah berupa citra wajah berformat *JPEG*.



Gambar 8. Arsitektur Sistem Pengenalan Wajah

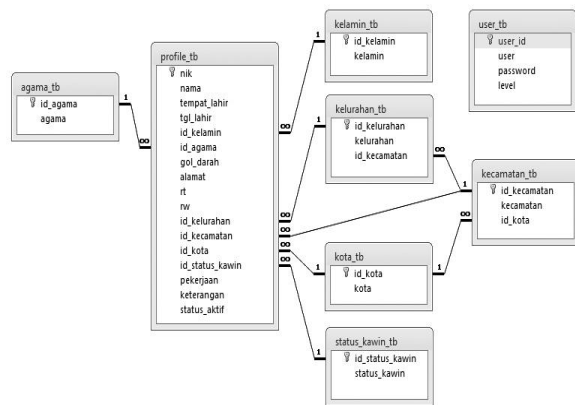
Pada Gambar 9 ditunjukkan sistem pengenalan wajah yang terdiri atas 4 lingkungan, yaitu: basis pengetahuan, *user interface*, *database* dan *engine*. Dalam Basis Pengetahuan terdapat suatu data wajah *template* yang bersumber dari *Database FERET*. *Database* berisi tabel profil. *Engine* terbagi menjadi 2 bagian, yaitu algoritma *Face Detection* dan algoritma *Face Recognition*. Terdapat hubungan antara *face detection* dan *face recognition* yang terintegrasi oleh data *input* dan *output*, dimana kedua proses tersebut harus berurutan. *User Interface* dapat berdialog langsung dengan sistem melalui Halaman Manipulasi Profile melalui *Form*

input dan *Form* Pengamatan Wajah.

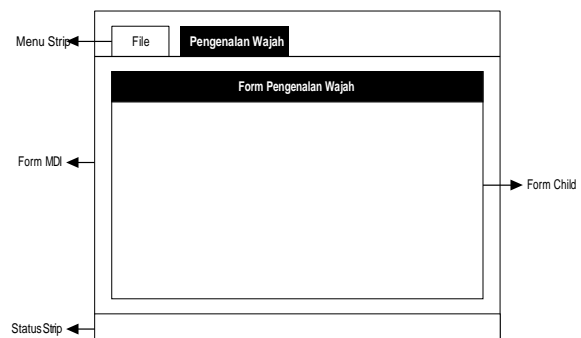


Gambar 9. Sub sistem model base

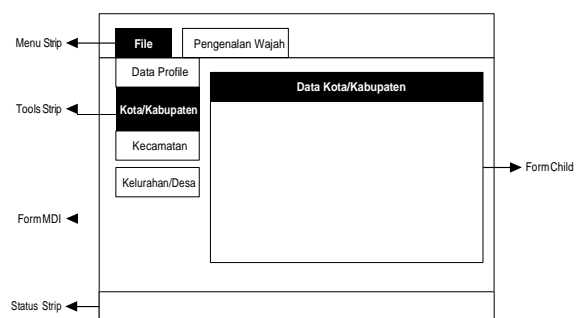
Gambar 10. merupakan desain fisis *database* profile yang sudah ternormalisasi dan berhubungan antara satu tabel dengan yang lain sebagai bentuk integritas data.



Gambar 10. Physical data model database pengenalan wajah



Gambar 12. Form utama untuk user Operator



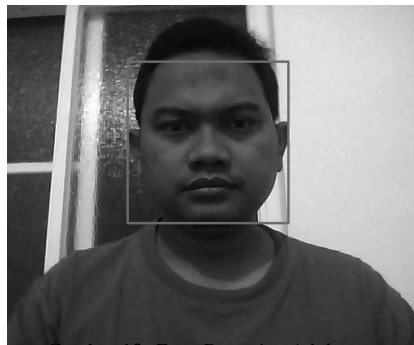
Gambar 11. Form utama untuk user Admin

Hasil pembangunan aplikasi sistem pengenalan wajah disusun atas: *Face Detection*, *Face Recognition* dan

manipulasi *Database*.

▪ *Face Detection*

Dari citra wajah *grayscale* akan dideteksi dengan *face detection* seperti tampak pada Gambar 13.



Gambar 13. *Face Detection Adaboost*

▪ *Face Recognition*

Langkah selanjutnya adalah melakukan *cropping* citra untuk diidentifikasi menggunakan metode *face recognition Eigenface PCA*. Citra wajah pada Gambar 14. akan menjadi citra data latih yang disimpan ke dalam *database* wajah.



Gambar 14. Citra diidentifikasi dengan *Eigenfaces PCA*

▪ *Antarmuka*

Antarmuka program berfungsi sebagai jembatan penghubung antara *user* dan sistem untuk berinteraksi.

Gambar 15. *Form data profile*

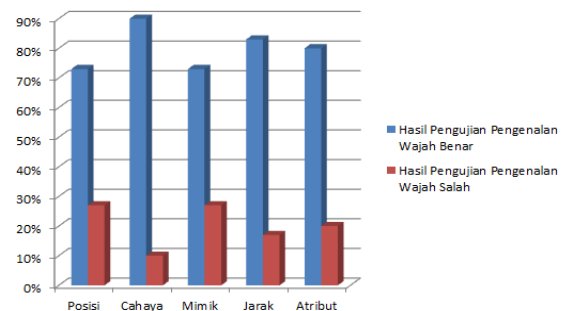
Gambar 15. adalah *form data profile*, di dalamnya terdapat beberapa masukkan berupa *textbox*, *option* dan *combobox* untuk diisi sesuai dengan data wajah yang sedang di-*capture*. Sedangkan pada bagian atas *form* terdapat *toolbar* navigasi yang berfungsi untuk menyimpan, mengubah dan menghapus data profile.

Pengujian dilakukan atas 10 responden yang terdiri dari 5 laki-laki dan 5 perempuan. Tabel I. adalah hasil dari pengujian yang dilakukan.

TABEL I
HASIL PENGUJIAN

Jenis Pengujian	Tingkat Kebenaran	Tingkat Kesalahan
Pengenalan wajah berdasarkan posisi wajah	73%	27%
Pengenalan wajah berdasarkan tingkat pencahayaan	90%	10%
Pengenalan wajah berdasarkan mimik muka	73%	27%
Pengenalan wajah berdasarkan jarak objek dan sensor webcam	83%	17%
Pengenalan wajah berdasarkan atribut berbeda, misalnya: kumis, kacamata, model rambut dan jenggot	80%	20%
Nilai rata-rata validitas	80%	20%

Dari pengujian di atas dapat disimpulkan sesuai dengan Gambar 16. berikut ini:



Gambar 16. Grafik validitas sistem pengenalan wajah

Dari pengujian sampel-sampel object citra wajah didapatkan hasil pemabahasan antara lain:

- Jika intensitas cahaya pada objek citra wajah di atas 1700 lux maka objek citra wajah dapat dikenali dengan baik, namun jika intensitas berkisar antara 0 sampai 1700 lux maka citra wajah tidak dikenali sama sekali atau salah dalam mendeteksi wajah karena kurangnya pencahayaan.
- Bila pencahayaan terhadap objek citra wajah pada kondisi normal (lebih dari 1700 lux) maka penambahan atribut wajah, antara lain: jilbab, cambang, model rambut dan jenggot, objek citra wajah masih dapat dikenali dengan baik. Sedangkan untuk penambahan atribut kacamata dan kumis dapat berpengaruh terhadap penurunan kualitas.
- Pengenalan wajah ditinjau dari posisi rotasi sangat berpengaruh terhadap akurasi pengenalan wajah. Rotasi posisi wajah terdapat 3 jenis kemungkinan, antara lain: menoleh ke kiri atau kanan, menggeleng ke kiri atau kanan, dan mendongak ke atas atau menunduk ke bawah. Secara ideal wajah yang dapat diidentifikasi dengan baik adalah wajah yang menghadap ke sensor kamera secara tegak lurus. Kemiringan yang bisa di tolerir berkisar 10 derajat dari posisi normal. Namun secara normal (tidak cacat fisik) kebanyakan wajah berada posisi tegak lurus dengan sensor kamera web sehingga aplikasi tetap mendeteksi wajah dengan baik.
- Pengenalan wajah juga dipengaruhi oleh mimik wajah, mimik yang dicoba adalah tersenyum (sudut bibir agak naik ke atas), tertawa (tampak

gigi), dan sedih (cemberut dan sudut bibir datar atau turun ke bawah). Dari uji mimik ternyata cukup signifikan mempengaruhi aplikasi dalam mengenali wajah.

Ditinjau dari jarak antara objek wajah dan sensor kamera web maka aplikasi secara efektif dapat mengenali wajah pada jarak 0,30 meter sampai 2,5 meter jika menggunakan kamera web standar yang terdapat pada laptop (webcam built-in). Pengenalan wajah dapat dilakukan lebih jauh lagi (kurang lebih 6 meter sampai 10 meter) tergantung dari kualitas resolusi citra sensor webcam dan kemampuan *zooming* ketika objek wajah berada sangat jauh.

Waktu yang dibutuhkan komputer untuk mengidentifikasi wajah (pengenalan sampai identifikasi/pelabelan wajah) secara real time sangat cepat, yaitu kurang dari 1 detik. Sebagai uji coba, aplikasi menggunakan 10 citra wajah sebagai data latih dan setiap objek memiliki 5 posisi citra wajah berbeda dan akurasi kebenaran mengenali wajah mencapai 80%.

KESIMPULAN

Dari hasil perancangan, analisis, implementasi dan pengujian sistem maka dapat disimpulkan hal-hal sebagai berikut :

- Perancangan dan implementasi pengenalan wajah dengan metode *Adaboost* dan *Eigenface PCA* telah berhasil dilakukan dalam penelitian.
- Rata-rata tingkat keberhasilan pengenalan wajah dengan metode *Adaboost* dan *Eigenfaces PCA* mencapai 80% pada berbagai kondisi berbeda (jarak objek dengan sensor, pencahayaan, posisi, atribut, dan mimik muka).
- Bila dibandingkan dengan metode-metode lainnya maka pengenalan wajah dengan metode *Adaboost* dan *Eigenfaces PCA* memiliki kelebihan pada proses kecepatan mengambil keputusan untuk mengenali wajah di kondisi *real time*.
- Pengenalan wajah yang diintegrasikan dengan data profil bermanfaat di berbagai sektor, misalnya bidang keamanan, pengawasan, kontrol akses, robotika, intelejen, militer, presensi dan lain-lain.
- Pengenalan wajah secara *real time* layak dikembangkan lebih lanjut oleh peneliti-peneliti lainnya dengan menggabungkan beberapa metode secara bertingkat untuk memperoleh akurasi yang lebih baik.

JURNAL 2

MENGHITUNG KECEPATAN MENGGUNAKAN COMPUTER VISION

Danny Agus Wahyudi; Iman H. Kartowisastro

Computer Engineering Department, Faculty of Engineering, Binus University
Jln. K.H. Syahdan No. 9, Palmerah, Jakarta Barat 11480
ihkartowisastro@binus.ac.id

ABSTRAK

Teknologi yang berada pada cangkupan sistem otomatis mengalami pengembangan di bidang vision. Tujuan dari pengembangan studi ini memanfaatkan computer vision untuk mendapatkan informasi perpindahan yang terjadi pada sebuah benda. Hasil yang didapat pada studi ini berupa informasi velocity benda yang diamati dan sudut elevasinya. Dari penelitian yang dilakukan penulis menarik kesimpulan bahwa penggunaan computer vision sangat bergantung terhadap cahaya. Agar studi ini lebih maksimal, algoritma harus disesuaikan dengan lingkungan di mana sistem ditempatkan.

Kata kunci: teknologi, vision, computer vision, velocity

PENDAHULUAN

Computer vision adalah bagian dari ilmu komputer yang membahas bagaimana sebuah komputer dapat ‘melihat’ seperti manusia, oleh karena sangat erat kaitannya dengan penglihatan, pencahayaan menjadi faktor yang juga penting dalam hal ini.

Pencahayaan sangat mempengaruhi kualitas hasil dari *computer vision*, karena seperti yang diketahui bahwa cahaya yang semakin banyak akan membuat kontras gambar semakin naik begitu juga kalau kekurangan cahaya, gambar akan semakin buruk.

Kemampuan di bidang *vision* dapat di kembangkan, contohnya sebuah robot menggunakan *vision* untuk seakan-akan dapat melihat, dan lengan robot di pabrik yang dapat membuat lingkaran presisi atau meletakkan sesuatu. Pengukuran jarak yang terintegrasi juga menggunakan bantuan *vision*. Dari pengembangan *vision* tersebut, penulis tertarik untuk membuat topik ini, yaitu menghitung kecepatan benda dengan bantuan *computer vision*. Studi ini dilakukan untuk mengetahui lebih jauh lagi kemampuan *computer vision*, dan mempelajari bagaimana proses memperkirakan kecepatan sebuah benda yang melaju dari satu titik ke titik lainnya. Topik ini dapat juga di kembangkan lebih lanjut dalam kehidupan sehari-hari seperti dalam memberikan informasi kedatangan kendaraan (estimasi waktu).

METODE

Kalibrasi dan Sampling Time

Kalibrasi adalah di mana terjadi penyesuaian ukuran sebenarnya terhadap ukuran yang berada pada gambar, kalibrasi digunakan untuk menghitung kecepatan, karena pada kecepatan digunakan ukuran sebenarnya bukan ukuran pada pixel. *Sampling time* digunakan untuk mengukur berapa lama sistem harus mengambil gambar, pada percobaan ini diberikan 0.5s untuk tiap pengambilan gambar.

Grayscale

Grayscale adalah proses mengubah derajat gambar yang mulanya RGB menjadi keabuan.

Thresholding

Thresholding adalah proses mengubah citra berderajat keabuan menjadi citra biner atau hitam putih sehingga dapat diketahui daerah mana yang termasuk obyek dan *background* dari citra secara jelas.

Reverse Pixel

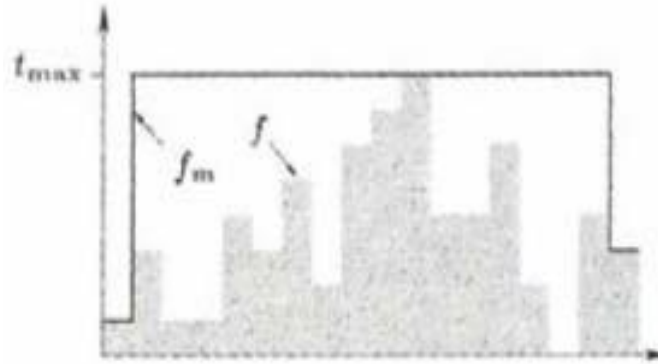
Mengubah nilai pixel dalam gambar, yang bernilai 1 menjadi 0, dan begitu juga sebaliknya. Dengan algoritma: $1 - \text{pixel awal} = \text{reverse pixel}$

Removing Pixel

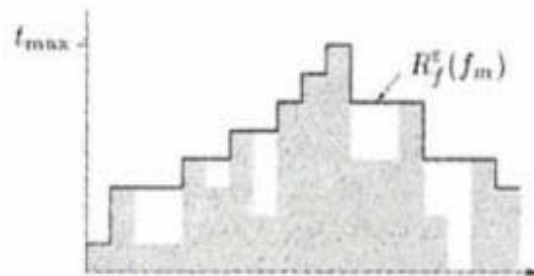
Removing pixel adalah teknik yang digunakan untuk menghilangkan pixel-pixel yang berhubungan dengan jumlah tertentu. Bila ingin menghilangkan pixel dengan jumlah 100, pixel yang berhubungan dengan jumlah 100 akan dihilangkan.

Image Filling Region

Teknik ini digunakan untuk mengisi sebuah daerah yang diinginkan dengan menyamakan jumlah pixel-nya. Proses pengisian daerah (*region*) dilakukan oleh sistem dengan cara mengisi daerah yang ditentukan dengan *pixel* yang sudah ditentukan juga, dengan syarat *pixel* yang diisi tidak melewati sisi luar (*edge*) dari objek tersebut, jadi batas pengisian *pixel* hanya sampai sisi luar objek (*edge*). Gambar 1 di bawah ini adalah bagaimana mengisi/mengubah nilai pixel untuk mengisi sebuah daerah, dengan cara menghilangkannya secara ‘paksa’ atau mengubah set nilai pixel (Gambar 2).



Gambar 1. Cara mengisi/mengubah nilai pixel.



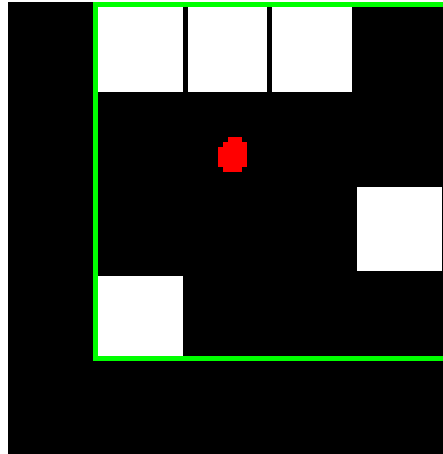
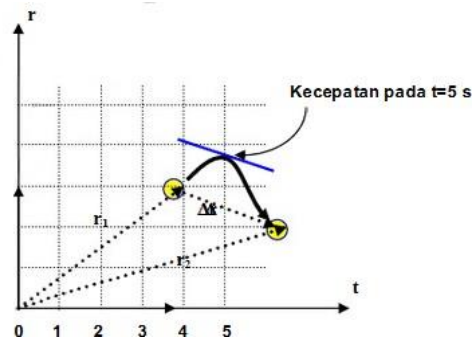
Gambar 2. Cara menghilangkan atau mengubah set nilai pixel.

Mencari Centroid

Untuk mendapatkan kecepatan sebuah benda yang bergerak, yang dibutuhkan adalah informasi posisi benda, melalui perubahan posisi benda baru dapat diberikan informasi kecepatannya. Oleh sebab itu, diperlu dicari informasi dari benda tersebut menggunakan algoritma untuk mencari titik tengah dari benda tersebut. Gambar 3 di bawah ini memperlihatkan bagaimana titik tengah adalah pixel merah, titik tengah di cari berdasarkan daerah yang sudah ditentukan. Pada gambar tersebut, daerah yang ditentukan adalah bounding box yang mempunyai tepi berwarna hijau.

Speed dan Velocity

Speed adalah perubahan jarak terhadap perubahan waktu, sementara *velocity* adalah perubahan posisi terhadap perubahan waktu. *Speed* menggunakan *distance*, sementara *velocity* menggunakan *displacement*. *speed* satuannya skalar jadi tidak ada informasi tentang arah di dalamnya, sementara *velocity* satuannya vektor sehingga mempunyai arah. Perhatikan gambar berikut ini.



Gambar 3. Titik tengah dan *bounding box*.

Karena pada studi ini gambar yang ditangkap dalam bentuk 2D, akan didapatkan informasi dari perubahan posisi tersebut. Untuk menghitung velocity-nya, kita akan menggunakan rumus berikut:

Di mana r berisi informasi x dan y , dan apabila dinyatakan dalam vektor satuan, maka:

$$\Delta \mathbf{r} = \mathbf{r}_2 - \mathbf{r}_1$$

$$\mathbf{v}_r = \frac{\Delta \mathbf{r}}{\Delta t} = \frac{\mathbf{r}_2 - \mathbf{r}_1}{t_2 - t_1}$$

$$\mathbf{v}_r = \frac{\Delta x \mathbf{i} + \Delta y \mathbf{j}}{\Delta t} = \frac{\Delta x}{\Delta t} \mathbf{i} + \frac{\Delta y}{\Delta t} \mathbf{j}$$

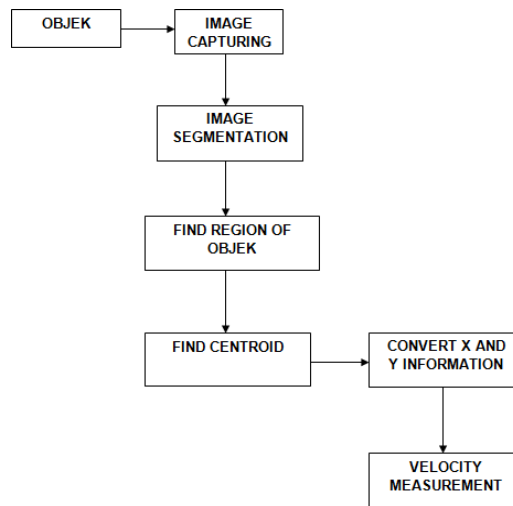
$$\mathbf{v}_r = v_{rx} \mathbf{i} + v_{ry} \mathbf{j}$$

dan besar kecepatan rata-rata dan arah kecepatan rata-rata dapat dihitung dengan rumus:

$$|\mathbf{v}_r| = v_r = \sqrt{v_{rx}^2 + v_{ry}^2} \quad \theta = \tan^{-1} \left(\frac{v_{ry}}{v_{rx}} \right)$$

Blok Diagram Sistem

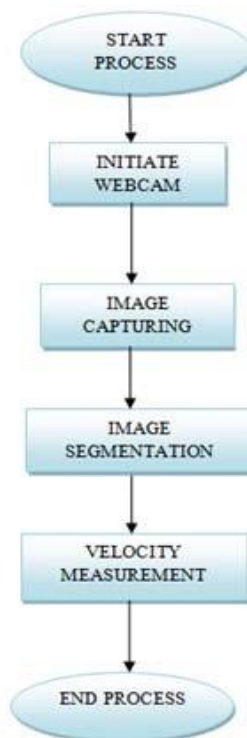
Gambar 4 di bawah ini menunjukkan alur dari sistem secara keseluruhan. Objek yang di tangkap oleh kamera, akan diproses dan sistem akan mengeluarkan informasi dari kecepatan benda tersebut.



Gambar 4. Blok diagram sistem.

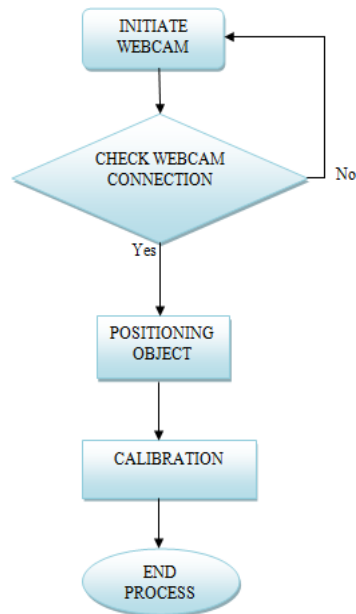
Diagram Alir Sistem

Gambar 5 berikut menunjukkan alir sistem secara keseluruhan.



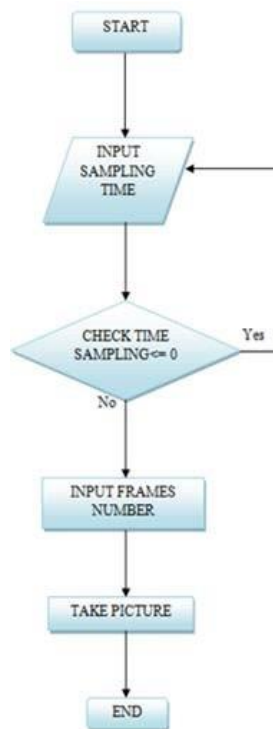
Gambar 5. Diagram alir sistem.

Gambar 6 menunjukkan diagram alir dari inisiasi *webcam* yang di pakai.



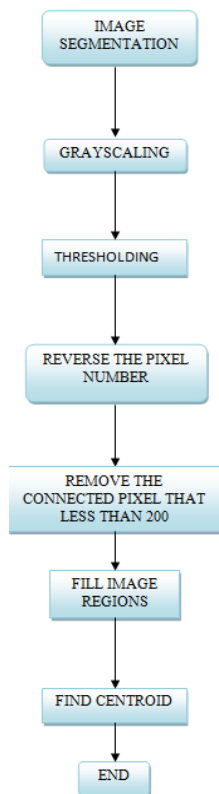
Gambar 6. Diagram alir *initiate webcam*.

Gambar 7 menunjukkan pengambilan gambar yang dilakukan oleh *webcam*. Dengan *interval* waktu yang dimasukkan, sistem akan mengambil gambar selama *t second*.



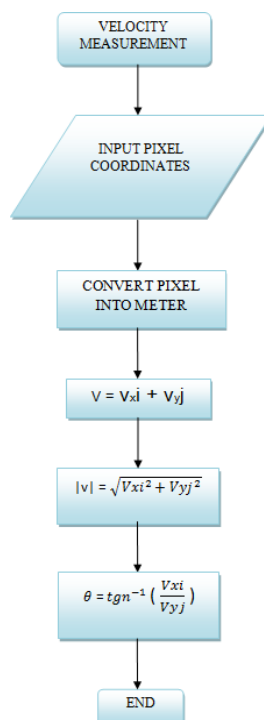
Gambar 7. Diagram alir pengambilan gambar.

Gambar 8 menunjukkan proses segmentasi yang dilakukan oleh sistem, bagaimana sistem memproses image sehingga menghasilkan titik tengah yang nantinya akan di analisis dan menghasilkan informasi kecepatan benda yang di amati.



Gambar 8. Diagram alir segmentasi citra.

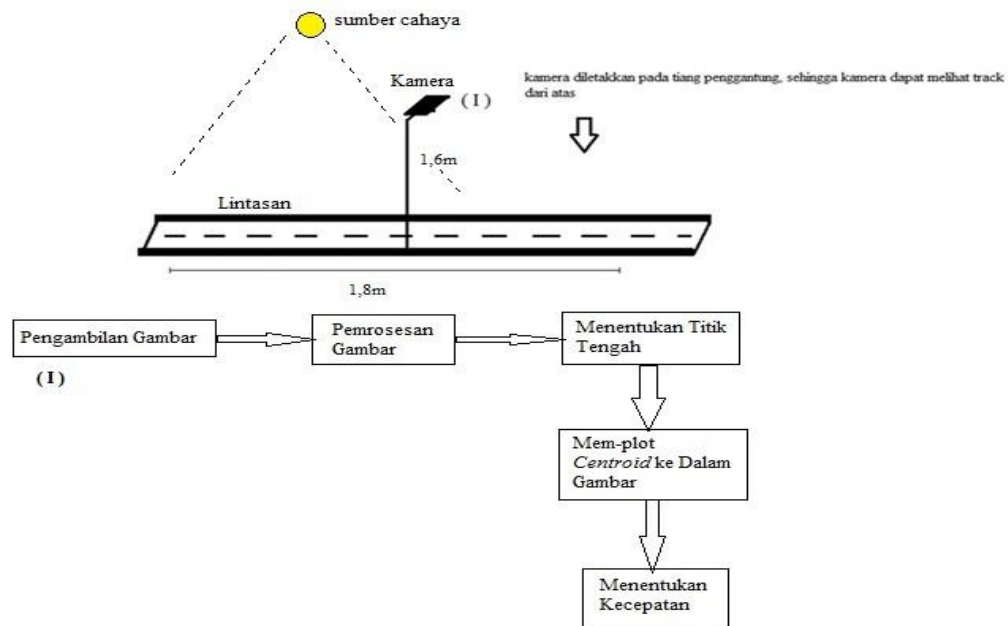
Gambar 9 menunjukkan proses perhitungan kecepatan terhadap informasi yang telah diterima, yaitu titik tengah, dan hasil dari perhitungan akan di tampilkan dan dapat di lihat oleh *user*.



Gambar 9. Diagram alir menghitung kecepatan.

Rancang Bangun

Berikut adalah rancang bangun dari sistem secara keseluruhan (Gambar 10).



Gambar 10. Rancang bangun dari sistem secara keseluruhan.

Spesifikasi Sistem

Spesifikasi sistem yang digunakan yaitu: webcam Logitech c270, *software* Matlab, *regionproperties* untuk mencari *centroid*, sistem operasi *Windows 7* pada PC, Tamiya dengan dimensi (14cm x 8cm), track lintasan dengan panjang 1,8m, *background* putih, lampu kamar sebagai sumber pencahayaan dengan spesifikasi lampu: 23watt (2 buah) dan 18watt (1 buah), penyangga dengan tinggi 2,20m, dan kamar dengan dimensi panjang 3,5m dan lebar 2,5m.

HASIL DAN PEMBAHASAN

Implementasi yang dilakukan ditujukan untuk menguji sistem yang telah dirancang, sistem dikatakan stabil apabila sistem dapat mempertahankan output yang stabil dan kinerja sistem yang stabil. Pada perancangan ini sistem diuji dengan pencahayaan yang masuk ke dalam sistem. Berikut adalah hasil percobaan dari sistem (Tabel 1 – 4).

Tabel 1. Hasil Percobaan 1

No.	Lokasi	Pencahayaayan	Sistem – Velocity Measurement		Sistem Manual – (Stopwatch) (m/s)	Kondisi Baterai	Error rate (%)
			Magnitude (m/s)	Sudut Deviasi (°)			
1	Indoor	23 watt	1,09m/s	0,5	1,1m/s	10 – 15 kali pakai	0,9%
2	Indoor	23 watt	1,17m/s	0,8	1,2m/s	10 – 15 kali pakai	2,5%
3	Indoor	23 watt	0,75m/s	0,7	0,74m/s	10 – 20 kali pakai	1,35%
4	Indoor	23 watt	0,66m/s	0,6	0,7m/s	10 – 20 kali pakai	5,7%
5	Indoor	23 watt	0,72m/s	0,6	0,71	10 – 20 kali pakai	1%
6	Indoor	23 watt	0,73m/s	0,6	0,69	10 – 20 kali pakai	4%
7	Indoor	23 watt	0,7m/s	0,6	0,71	10 – 20 kali pakai	1%
8	Indoor	23 watt	0,73m/s	0,5	0,72	10 – 20 kali pakai	1%
9	Indoor	23 watt	0,7m/s	0,5	0,7	10 – 20 kali pakai	0%
10	Indoor	23 watt	0,72m/s	0,6	0,73	10 – 20 kali pakai	1%
Rata – rata			0,797m/s	0,6	0,8m/s		1,935%

Tabel 2. Hasil Percobaan 2

No.	Lokasi	Pencahayaayan	Sistem – Velocity Measurement		Sistem Manual – (Stopwatch) (m/s)	Kondisi Baterai	Error rate (%)
			Magnitudo (m/s)	Sudut Deviasi (°)			
1	Indoor	18 watt	2,25m/s	0,3	2,23m/s	1 – 10 kali pakai	0,8%
2	Indoor	18 watt	2,28m/s	0,2	2,3m/s	1 – 10 kali pakai	2%
3	Indoor	18 watt	2,3m/s	0,3	2,3m/s	1 – 10 kali pakai	0%
4	Indoor	18 watt	2,39m/s	0,3	2,36m/s	1 – 10 kali pakai	1,27%
5	Indoor	18 watt	2,33m/s	0,5	2,35m/s	1 – 10 kali pakai	0.8%
6	Indoor	18 watt	2,3m/s	0,4	2,3m/s	1 – 10 kali pakai	0%
7	Indoor	18 watt	2,4m/s	0,1	2,36m/s	1 – 10 kali pakai	1,69%
8	Indoor	18 watt	2,36m/s	0,2	2,3m/s	1 – 10 kali pakai	2,6%
9	Indoor	18 watt	2,29m/s	0,9	2,23m/s	1 – 10 kali pakai	2,6%
10	Indoor	18 watt	2,26m/s	0,5	2,2m/s	1 – 10 kali pakai	2,7%
Rata – rata			2,313m/s	0,37	2,293m/s		1,446%

Tabel 3. Hasil Percobaan 3

No.	Lokasi	Pencahayaayan	Sistem – Velocity Measurement		Sistem Manual – (Stopwatch) (m/s)	Kondisi baterai	Error rate (%)
			Magnititude (m/s)	Sudut Deviasi (°)			
1	Indoor	23 watt +18 watt	1,55m/s	0,3	1,56 m/s	10 - 15 kali pakai	0,6%
2	Indoor	23 watt +18 watt	1,5 m/s	0,2	1,49 m/s	10 - 15 kali pakai	0,6%
3	Indoor	23 watt +18 watt	1,6m/s	0,3	1,58 m/s	10 - 15 kali pakai	1,2%
4	Indoor	23 watt +18 watt	1,3m/s	0,3	1,3 m/s	10 - 15 kali pakai	0%
5	Indoor	23 watt +18 watt	1,4m/s	0,4	1,4m/s	10 - 15 kali pakai	0%
6	Indoor	23 watt +18 watt	1,4m/s	0,5	1.42m/s	1 – 10 kali pakai	1,4%
7	Indoor	23 watt +18 watt	2,4m/s	0,6	2,37m/s	1 – 10 kali pakai	1,26%
8	Indoor	23 watt +18 watt	2,36m/s	0,7	2,35m/s	1 – 10 kali pakai	0,42%
9	Indoor	23 watt +18 watt	2,3m/s	0,7	2,3m/s	1 – 10 kali pakai	0%
10	Indoor	23 watt +18 watt	2,4m/s	0,7	2,34m/s	1 – 10 kali pakai	2,5%
Rata – rata			1,821m/s	0,47	1,81m/s		0,858%

Tabel 4. Hasil Percobaan 4

No.	Lokasi	Pencahayaayan	Sistem – Velocity Measurement		Sistem Manual – (Stopwatch) (m/s)	Kondisi Baterai	Error rate (%)
			Magnititude (m/s)	Sudut Deviasi (°)			
1	Indoor	23 watt + 23 watt + 18 watt	1,63m/s	0,3	1,62 m/s	10 - 15 kali pakai	0,6%
2	Indoor	23 watt + 23 watt + 18 watt	1,6m/s	0,2	1,6 m/s	10 - 15 kali pakai	0%
3	Indoor	23 watt + 23 watt + 18 watt	1,6m/s	0,3	1,59 m/s	10 - 15 kali pakai	0,6%
4	Indoor	23 watt + 23 watt + 18 watt	1,62m/s	0,3	1,64 m/s	10 - 15 kali pakai	1,2%

5	Indoor	23 watt + 23 watt + 18 watt	1,45m/s	0,4	1,48m/s	10 - 15 kali pakai	2%
6	Indoor	23 watt + 23 watt + 18 watt	1,1m/s	0,6	1,15m/s	15 - 20 kali pakai	4,34%
7	Indoor	23 watt + 23 watt + 18 watt	1m/s	0,5	1m/s	15 - 20 kali pakai	0%
8	Indoor	23 watt + 23 watt + 18 watt	1,2m/s	0,6	1,18m/s	15 - 20 kali pakai	1,69%
9	Indoor	23 watt + 23 watt + 18 watt	1,14m/s	0,4	1,2m/s	15 - 20 kali pakai	5%
10	Indoor	23 watt + 23 watt + 18 watt	1,2m/s	0,3	1,2m/s	15 - 20 kali pakai	0%
Rata – rata			1,354m/s	0.39	1.366m/s		1,543%

Pada percobaan di ruang yang tidak terdapat cahaya, sistem tidak dapat mengambil gambar apapun. Pada keadaan ini sistem dianggap tidak dapat bekerja secara maksimal.

Terdapat beberapa poin yang didapatkan pada implementasi ini, yaitu: (1) perhitungan dari sistem dibandingkan dengan perhitungan sistem lain agar dapat membuktikan kestabilan sistem; (2) pencahayaan tidak merata/homogen mengakibatkan adanya *noise*; (3) pada ruangan yang gelap sistem tidak dapat bekerja karena tidak ada cahaya; (4) ruangan yang dipakai mempengaruhi pencahayaan, pemantulan cahaya terhadap latar dinding.

SIMPULAN

Pencahayaan yang tidak merata menyebabkan *noise*, dan tingkat pencahayaan yang tinggi ataupun rendah juga menyebabkan *noise*. *Noise* yang muncul akibat pencahayaan dapat dihilangkan dengan algoritma *remove pixel*. Akan tetapi, saat *noise* memiliki jumlah yang lebih besar dari obyek, sistem tidak dapat mengatasinya lagi, contohnya pada saat kamar gelap. Gambar tidak lagi dapat diproses karena tidak ada obyek apapun yang tertangkap.

JURNAL 3

SEGMENTASI CITRA UNTUK DETEKSI OBJEK WARNA PADA APLIKASI PENGAMBILAN BENTUK CITRA RECTANGLE

Asep Nana H^[1], M. Ichwan^[1], I Made Santika Putra

^[1]Jurusan Teknik Informatika, Fakultas Teknologi Industri Institut
Teknologi Nasional Bandung

asepnana@itenas.ac.id, ichwan@itenas.ac.id, imadesantikaputra@gmail.com

ABSTRAK

Pengambilan citra terkadang tidak sesuai dengan yang dibutuhkan, sehingga diperlukan proses pengolahan citra untuk mendapatkan bentuk citra yang diinginkan. Dengan menanamkan kecerdasan pada sebuah kamera, pengambilan suatu citra dengan bentuk tertentu dapat dilakukan sehingga mengurangi beban waktu dan memfasilitasi pengguna dalam proses pengambilan gambar. Segmentasi citra merupakan proses pengambilan informasi dari citra dalam pencarian citra yang serupa seperti warna. Warna dapat dijadikan input dalam penggambaran daerah yang diinginkan (*Region Of Interest*) melalui proses deteksi warna dan *tracking* warna, sehingga dapat dilakukan pengambilan gambar dalam bentuk tertentu. Object Tracking adalah proses mengikuti suatu objek yang bergerak dan berpindah posisi. Computer Vision didefinisikan sebagai salah satu cabang ilmu pengetahuan yang mempelajari bagaimana komputer dapat mengenali obyek yang diamati. Dalam mengenali obyek yang diamati dilakukan proses segmentasi citra dengan menggunakan euclidean color filtering, grayscale dan deteksi blob. Dalam implementasi segmentasi citra untuk deteksi objek warna untuk pengambilan bentuk citra rectangle, digunakan metode segmentasi dan center of gravity. Hasil dari pengujian sistem ini adalah objek warna dapat terdeteksi dan dijejaki dengan baik dalam jarak terbaik antara 40cm-80cm dan dengan intensitas cahaya terbaik antara 22lx-242lx.

Kata Kunci : *Pengolahan Citra, Computer Vision, Deteksi Warna, Segmentasi, Objek Tracking, Capture image, Center of Gravity.*

Latar Belakang

Webcam merupakan sebuah *device* yang dapat digunakan sebagai sensor dalam mendeteksi sebuah benda bergerak melalui proses pengolahan citra. Webcam juga dapat digunakan dalam pengambilan gambar (*capture image*).

Object Tracking adalah proses mengikuti suatu objek yang bergerak dan berpindah posisi. *Computer Vision* didefinisikan sebagai salah satu cabang ilmu pengetahuan yang mempelajari bagaimana komputer dapat mengenali objek yang diamati. Dalam proses pengenalan objek dan deteksi objek diperlukan pemisahan segmen tertentu pada suatu citra yang dikenal dengan proses segmentasi.

Segmentasi citra merupakan bagian dari proses pengolahan citra. Kegunaan segmentasi menurut Forsyth dan Ponce (2003) adalah pengambilan informasi dari citra seperti pencarian bagian mesin, pencarian manusia dan pencarian citra yang serupa. Secara umum pendekatan segmentasi citra yang sering digunakan adalah melalui pendekatan intensitas, pendekatan warna dan pendekatan bentuk (Rujikietgumjorn, 2008). Salah satu produk teknologi dalam proses pengolahan citra adalah *Aforge.Net Framework*.

Pada penelitian ini sistem yang dibangun adalah suatu aplikasi yang dapat mengambil sebuah bentuk citra *rectangle* dari hasil *Region Of Interest* (ROI) berdasarkan objek warna yang dideteksi dan dilacak. Pada Penelitian ini digunakan beberapa metode pelacakan (*tracking*) yakni dengan mengkombinasikan metode segmentasi dan *center of gravity* (COG) yang diharapkan dapat melacak pergerakan dari objek warna yang dideteksi

Tujuan Penelitian

Tujuan dari kegiatan penelitian ini adalah mendeteksi adanya warna yang bergerak dalam penentuan letak dengan menggunakan metode pelacakan yakni metode segmentasi warna dan *center of gravity* (COG) sehingga dapat menyeleksi suatu citra tertentu berdasarkan daerah yang diinginkan (*Region Of Interest*).

Framework Aforge.Net ^[6]

AForge.NET merupakan framework C# terbuka yang dirancang untuk pengembang dan peneliti di bidang Computer Vision dan Artificial Intelligence (Kecerdasan Buatan) - Pengolahan Gambar, Jaringan Saraf Tiruan, Algoritma Genetika, Logika Fuzzy, Pembelajaran Mesin, Robotika, dll. Kerangka ini terdirioleh set perpustakaan dan contoh aplikasi, yang

menunjukkan karakteristiknya:

- AForge.Imaging - library dengan rutinitas pengolahan citra dan filter;
- AForge.Vision - library *computer vision*;
- AForge.Video - set library untuk pemrosesan video;
- AForge.Neuro - library perhitungan jaringan saraf tiruan;
- AForge.Genetic - library pemrograman evolusi;
- AForge.Fuzzy - library perhitungan *fuzzy*;
- AForge.Robotics - library yang memberikan dukungan pada beberapa robotika kit;
- AForge.MachineLearning - library pembelajaran mesin; dan lain-lain.

Euclidean Color Filtering

Euclidean Color Filtering adalah metode yang berguna untuk menemukan sebuah warna yang terdapat pada sebuah gambar. Filter ini memfilter piksel-piksel pada gambar yang berada di dalam/di luar dari lingkup RGB (Red Green Blue) dengan pusat dan radius tertentu. Filter tersebut membiarkan piksel-piksel dengan warna yang berada di dalam/di luar dari lingkup yang telah ditentukan dan mengisi sisanya dengan warna tertentu.

Grayscale

GrayScale adalah suatu citra dimana nilai dari setiap pixel merupakan sampel tunggal. Citra yang ditampilkan adalah citra keabuan dimana intensitasnya berada pada interval 0 – 255, warna hitam

(0) pada bagian yang intensitasnya

terlemah dan warna putih(255) pada intensitas terkuat. Proses konversi dari citra berwarna menjadi Grayscale menggunakan koefisien dari ITU Recommendation BT.709.

$$\text{Grayscale} = 0.2125 * \text{red} + 0.7154 * \text{green} + 0.0721 * \text{blue}.$$

Persamaan Konversi Grayscale(1)

Blob [4]

Blob merupakan sekumpulan piksel-piksel yang memiliki hubungan tetangga. Proses perhitungan *blob* dapat dilakukan dengan melakukan analisis piksel yang bertetangga. Piksel bertetangga pada sebuah piksel ditentukan sebagai piksel yang berjarak 1 dari piksel asal. Proses perhitungan *blob* akan memanfaatkan relasi piksel *8-neighbors*. Gambar 1 merupakan gambaran sederhana dari relasi *8-neighbors*.

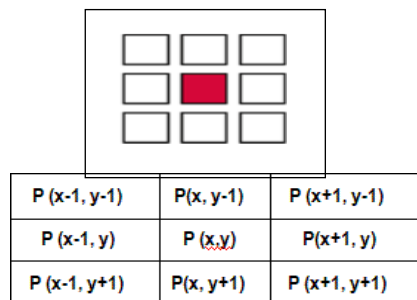
Gambar 1. Relasi 8-neighbors

Relasi 8-neighbors

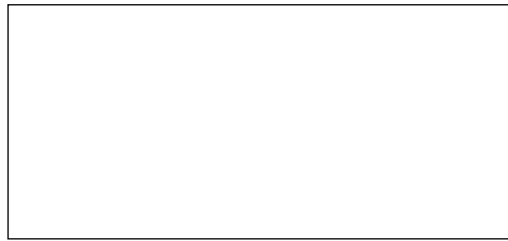
Proses pemetaan objek akan menelusuri tiap piksel pada setiap baris yang ada dan memberikan label pada piksel yang memiliki nilai warna selain hitam (RGB = 0 0 0). Setiap piksel yang memiliki relasi hubungan *8-neighbors* akan diberikan label yang sama.

Pusat Massa Obyek (Center Of Gravity)[9]

Metode Center of Gravity adalah metode yang dipergunakan untuk menentukan titik keseimbangan dari grafik yang merupakan hasil dari proses pengolahan citra. Persamaan



merupakan perumusan matematis dari metode center of gravity.



Persamaan COG (3)

Analisis Sistem

Kebutuhan sistem aplikasi deteksi objek warna untuk pengambilan bentuk citra berupa kebutuhan akan *hardware* dan *software*.

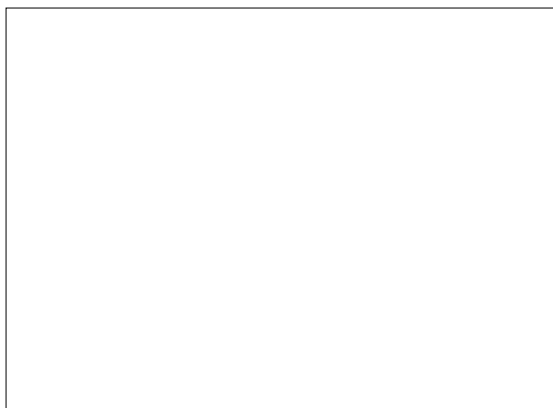
- Analisis fitur sistem

Dalam perancangan aplikasi deteksi objek warna untuk pengambilan bentuk citra, disusun sebuah analisis terhadap fitur yang akan diterapkan, seperti :

- Mendeteksi dan menjejaki objek warna.
- Menampilkan garis *rectangle* untuk pengambilan bentuk citra dari proses penguncuan poin 1 dan poin 2.

- Blok Diagram

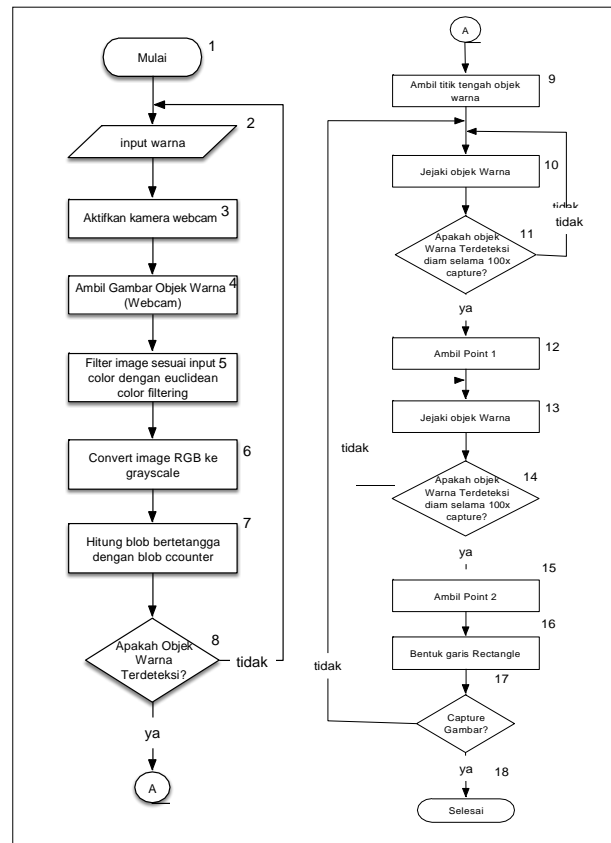
Gambar 2 adalah blok diagram atau alur kerja dari aplikasi deteksi objek warna untuk pengambilan bentuk citra.



Gambar 2. Blok Diagram

Sistem Kerja Aplikasi

Flowchart atau diagram alir adalah penggambaran secara grafik dari langkah- langkah dan urutan prosedur dari suatu program. Gambar 3 adalah perancangan flowchart aplikasi deteksi objek bergerak untuk pengambilan bentuk citra.



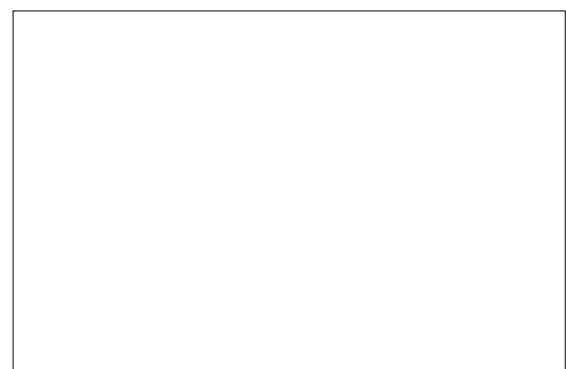
Gambar 3. FlowChart Sistem

Berdasarkan sistem kerja aplikasi seperti terlihat pada gambar 3, terdapat beberapa tahapan dalam implementasi segmentasi citra untuk deteksi objek warna pada pengambilan citra *rectangle*, berikut adalah tahapan-tahapan dari gambar 3 :

1. Mulai

Merupakan *state* awal memulai aplikasi.

2. Pemilihan Warna



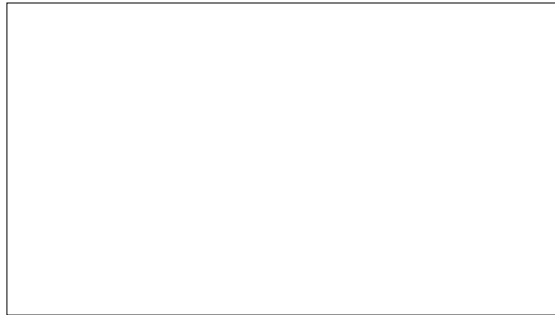
Gambar 4. *Pallate* Warna untuk pemilihan warna

Pada tahap pemilihan warna dilakukan sebagai masukan kepada sistem

untuk mendeteksi warna yang akan dideteksi dan dijejaki. Proses pemilihan warna dapat dilihat pada gambar 4.

3. Pengaktifan kamera webcam

Pada tahap proses pengaktifan perangkat kamera webcam dilakukan sebagai media untuk mendeteksi objek. Pengaktifan kamera webcam dilakukan dengan menekan tombol *start detection* pada aplikasi. Pada gambar 5 merupakan kondisi ketika media webcam aktif.



Gambar 5. Kamera Webcam Aktif

4. Input Objek Warna

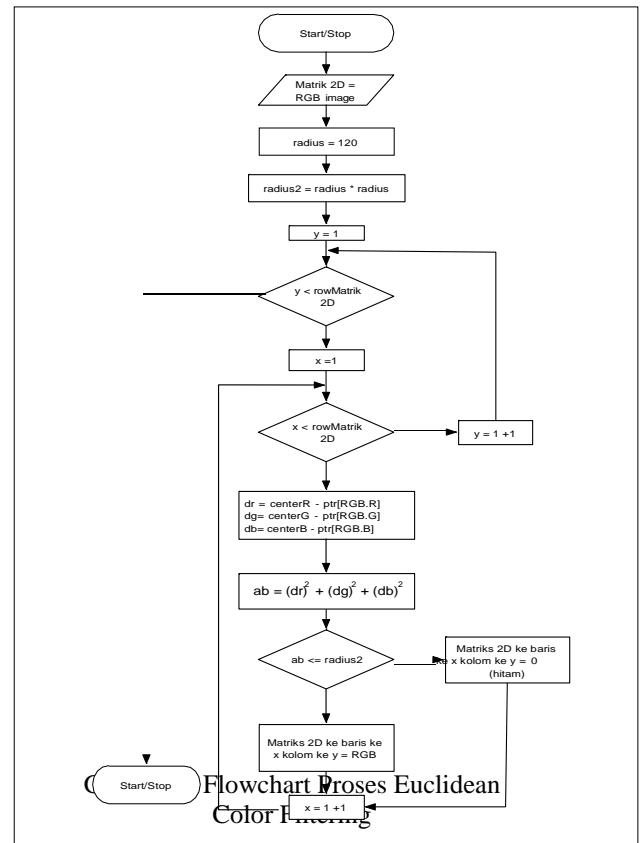
Pada tahap ini dilakukan input objek warna dari warna yang sebelumnya dipilih pada tahap 1. Gambar 6 adalah proses input objek warna.



Gambar 6. Input Objek Warna

5. Segmentasi objek warna terhadap ruang warna RGB dengan Euclidean Color Filtering

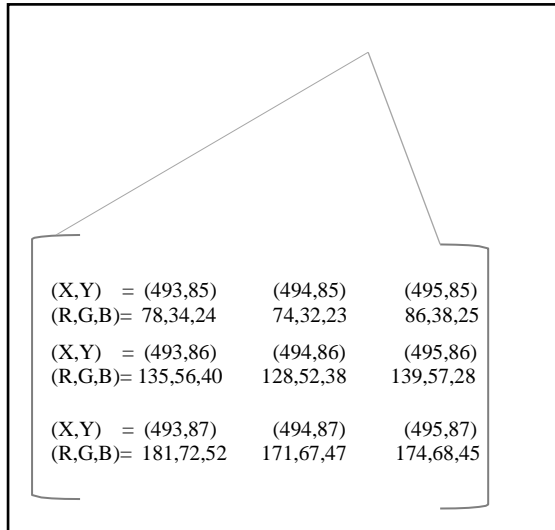
Pada tahap ini dilakukan proses *filtering* warna menggunakan *euclidean color filtering* berdasarkan pemilihan warna pada tahap 1. Proses *filtering* menggunakan *euclidean color filtering* ini adalah memisahkan objek warna dari background dimana background akan dijadikan warna hitam seperti terlihat pada gambar 8. Proses euclidean dapat dilihat pada gambar 7.



Gambar 8. Input citra setelah dilakukan euclidean color filtering

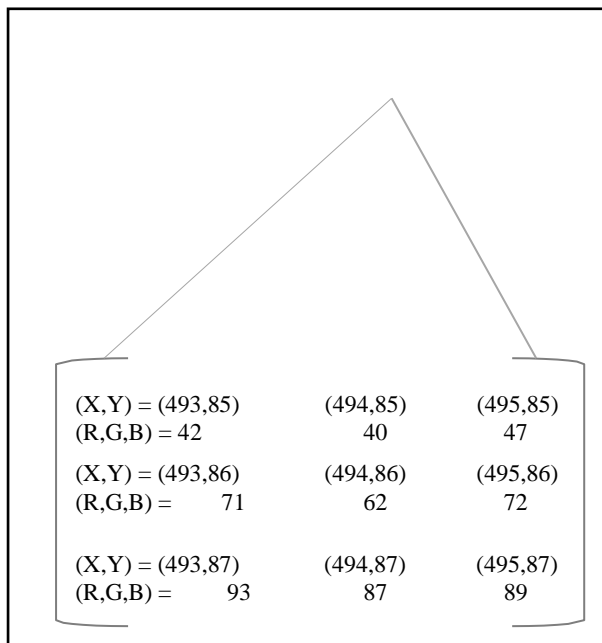
6. Konversi Warna RGB ke Grayscale

Setelah dilakukan pemisahan objek warna dengan menggunakan *euclidean color filtering*, *state* selanjutnya adalah mengubah nilai warna citra rgb ke dalam ruang warna *grayscale* (keabuan). Proses mengubah nilai warna citra rgb ke dalam ruang warna *grayscale* menggunakan persamaan 1. ilustrasi perubahan nilai warna citra rgb dapat dilihat pada gambar 9 dan gambar10.



(X,Y) = (493,85)	(494,85)	(495,85)
(R,G,B)= 78,34,24	74,32,23	86,38,25
(X,Y) = (493,86)	(494,86)	(495,86)
(R,G,B)= 135,56,40	128,52,38	139,57,28
(X,Y) = (493,87)	(494,87)	(495,87)
(R,G,B)= 181,72,52	171,67,47	174,68,45

Gambar 9. Ilustrasi RGB pada piksel citra sebelum dikonversi



(X,Y) = (493,85)	(494,85)	(495,85)
(R,G,B) = 42	40	47
(X,Y) = (493,86)	(494,86)	(495,86)
(R,G,B) = 71	62	72
(X,Y) = (493,87)	(494,87)	(495,87)
(R,G,B) = 93	87	89

Gambar 10. Ilustrasi RGB pada piksel citra setelah dikonversi

Berikut contoh proses perhitungan nilai rata-rata citra pada titik koordinat (493,85), (494,86), (495,87) dengan persamaan 1.

a. Koordinat (493,85)

$$\begin{aligned}
 \text{Grayscale} &= (0.2125 \times R) + (0.7154 \times G) + (0.0721 \times B) \\
 &= (0.2125 \times 78) + (0.7154 \times 34) + (0.0721 \times 24) \\
 &= 16.575 + 24.3236 + 1.7304 \\
 &= 42,629 = 42
 \end{aligned}$$

b. Koordinat (494,86)

$$\begin{aligned}
 \text{Grayscale} &= (0.2125 \times R) + (0.7154 \times G) + (0.0721 \times B) \\
 &= (0.2125 \times 128) + (0.7154 \times 52) + (0.0721 \times 38) \\
 &= 27,2 + 37,2008 + 2,7398 \\
 &= 62,1406 = 62
 \end{aligned}$$

c. Koordinat (495,87)

$$\begin{aligned}
 \text{Grayscale} &= (0.2125 \times R) + (0.7154 \times G) + (0.0721 \times B) \\
 &= (0.2125 \times 174) + (0.7154 \times 68) + (0.0721 \times 45) \\
 &= 36,975 + 48,6472 + 3,2445 \\
 &= 88,867 = 89
 \end{aligned}$$

7. Pemetaan dan Perhitungan Blob dengan BlobCounter

Proses perhitungan *blob* akan memanfaatkan relasi piksel *8-neighbors*. Menurut pustaka AForge.Net langkah- langkah perhitungan blob adalah sebagai berikut.

- Proses pemetaan objek akan menelusuri tiap piksel pada setiap baris yang ada dan memberikan label pada piksel yang memiliki nilai warna selain hitam (RGB = 0 0 0). Setiap piksel yang memiliki relasi hubungan *8-neighbors* akan diberikan label yang sama.
- Proses pengumpulan informasi *blob*, akan mengumpulkan dan mengolah informasi tiap piksel yang bertetangga berdasarkan letak dan label yang dihasilkan oleh proses pemetaan objek. Letak dan label piksel yang bertetangga tersebut digunakan untuk membentuk suatu *blob* dan informasi pendukungnya seperti luas area, tingkat kepenuhan, titik pusat dan area kotak *blob*.

Gambar 11 adalah hasil dari pemetaan dan perhitungan *blob*.

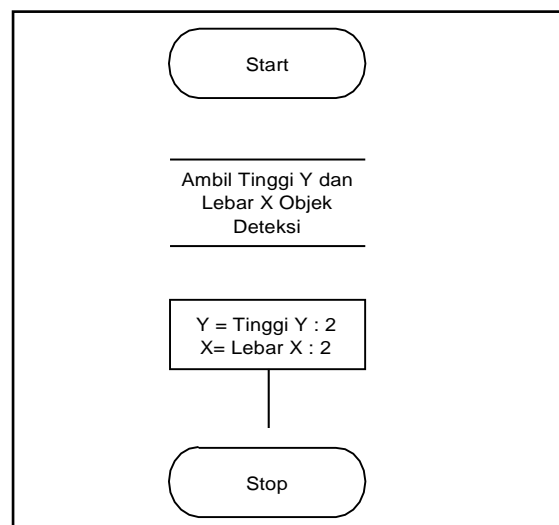


Gambar 11. hasil deteksi blob pada ruang warna RGB

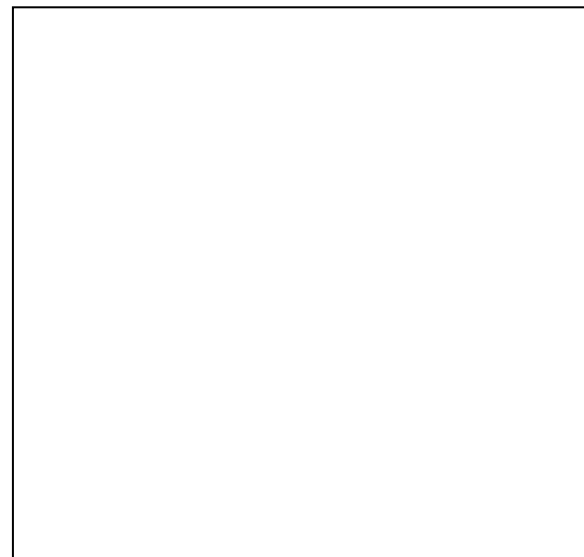
8. Penentuan Titik Pusat (*Center of Gravity*)

Setelah objek warna terdeteksi melalui proses segmentasi warna, grayscale dan blob, maka proses selanjutnya adalah penentuan titik pusat (*center of gravity*) dari objek yang telah dideteksi. Penentuan titik pusat ini nantinya dapat digunakan dalam pembentukan garis *rectangle* dalam proses pengambilan bentuk citra. Proses penentuan titik pusat ini menggunakan persamaan 3 seperti terlihat pada gambar

12. Realisasi dari perhitungan persamaan 3 terlihat pada gambar 13.



Gambar 12. Flowchart penentuan titik pusat

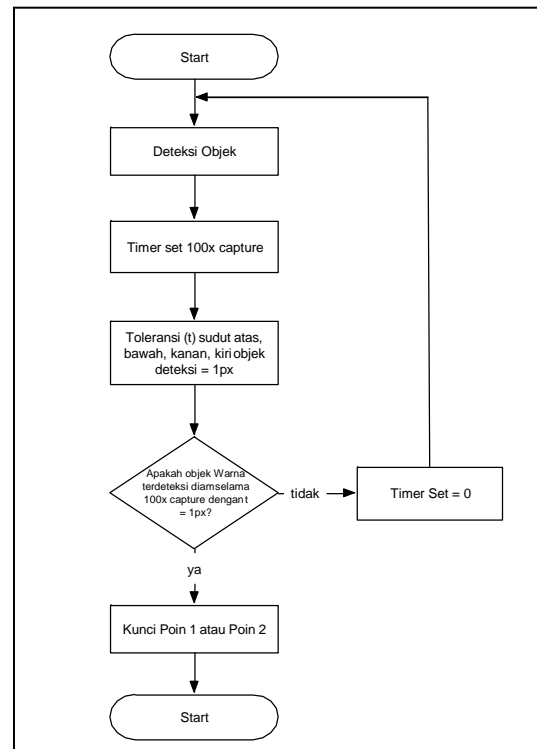


Gambar 13. Titik pusat dari deteksi objek warna

9. Penguncian Poin 1 dan Poin 2

Dalam proses pengambilan bentuk citra *rectangle* dibutuhkan penentuan poin 1 dan poin 2 oleh user. Setelah objek terdeteksi dan telah ditentukan titik pusat, proses selanjutnya adalah penguncian poin

1 dan poin 2. Poin 1 dan poin 2 dapat terkunci jika objek yang terdeteksi diam selama 100x capture dalam toleransi 1px atas, 1px bawah, 1px kanan dan 1px kiri. Gambar 14 adalah *flowchart* proses penguncian poin 1 dan poin 2. Gambar 15 adalah realisasi dari *flowchart* penguncian poin 1 dan poin 2.



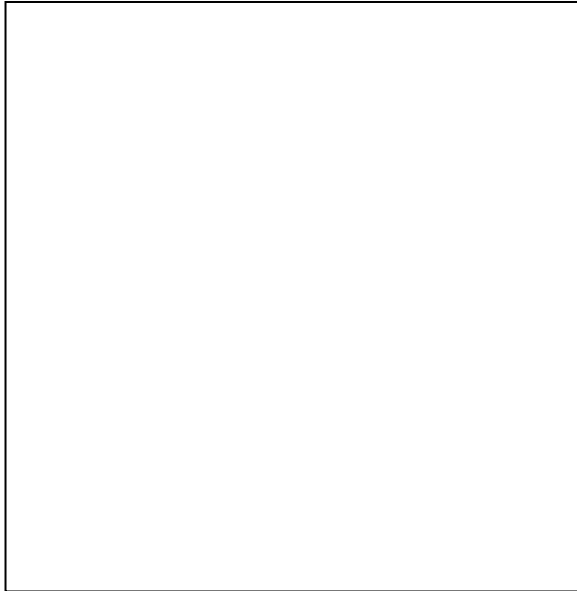
Gambar 14. Flowchart Penguncian Poin 1 dan poin 2



Gambar 15. Realisasi penguncian poin 1 dan poin 2

10. Pembentukan *rectangle*

Setelah penguncian poin 1 dan poin 2, maka sistem secara otomatis akan membentuk garis *rectangle*. Proses pembentukan *rectangle* seperti terlihat pada Gambar 16. Realisasi proses pembentukan *rectangle* seperti terlihat pada Gambar 17.



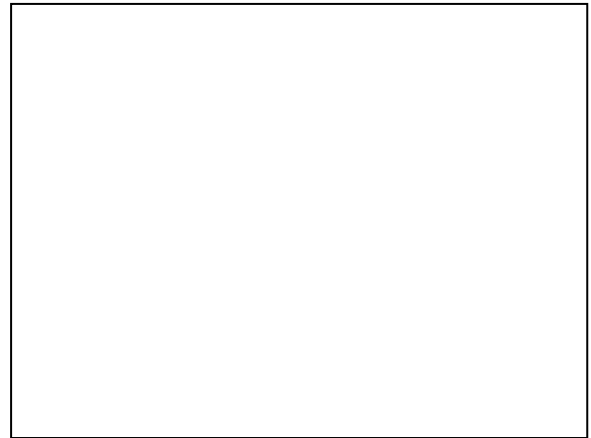
Gambar 16. Proses Pembentukan *Rectangle*



Gambar 17. Realisasi Pembentukan
Rectangle

11. Pengambilan Citra *Rectangle*

Setelah pembentukan garis *rectangle* maka untuk pengambilan bentuk citra *rectangle* dilakukan penekanan pada tombol *capture* yang terdapat pada aplikasi. Hasil proses *capture* seperti terlihat pada Gambar 18. Format gambar dari hasil pengambilan citra berdasarkan pembentukan garis *rectangle* adalah berformat .jpg.



Gambar 18. Hasil Pengambilan gambar

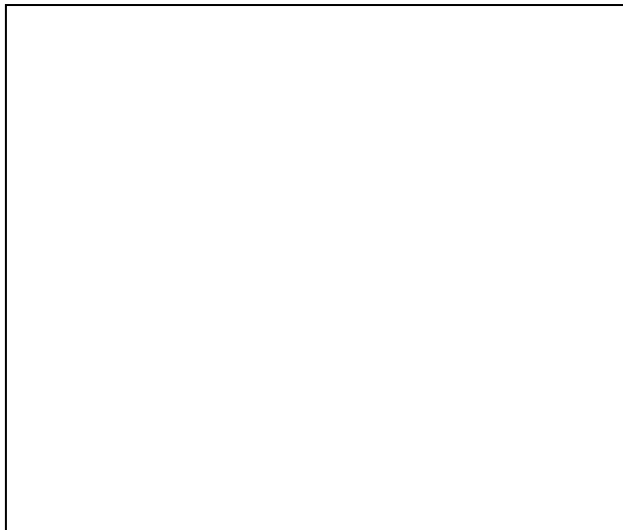
Pengujian Deteksi Terhadap Cahaya dan Jarak

Pengujian pada aplikasi deteksi objek bergerak untuk pengambilan bentuk citra ini meliputi pengujian deteksi objek warna terhadap cahaya, pengujian deteksi objek warna terhadap jarak, dan pengujian blackbox fitur sistem. Hasil pengujian deteksi objek warna terhadap cahaya dan jarak terlihat pada tabel 1.

Tabel 1 tingkat keberhasilan deteksi objek warna terhadap cahaya

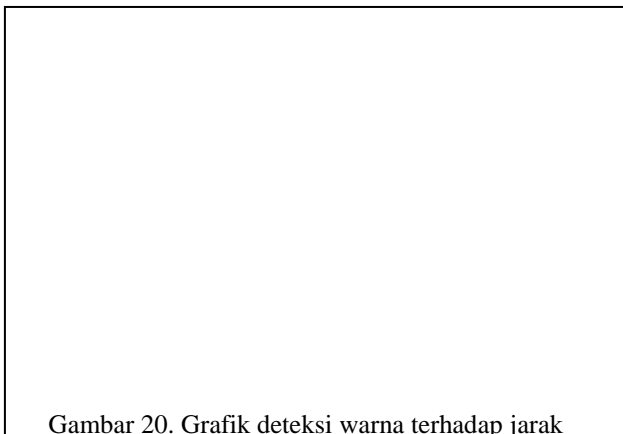
Jarak	Nilai Cahaya									
	20	40	60	80	100	120	140	160	180	200
Pengujian Deteksi Objek Warna Memori	X	X	X	X	X	X	X	X	X	X
	X	X	X	X	X	X	X	X	X	X
	X	X	X	X	X	X	X	X	X	X
	X	X	X	X	X	X	X	X	X	X
	X	X	X	X	X	X	X	X	X	X
	X	X	X	X	X	X	X	X	X	X
	X	X	X	X	X	X	X	X	X	X
	X	X	X	X	X	X	X	X	X	X
	X	X	X	X	X	X	X	X	X	X
	X	X	X	X	X	X	X	X	X	X
0% = 0 lx										
10% = 22 lx										
20% = 33 lx										
30% = 44 lx										
40% = 68 lx										
50% = 104 lx										
60% = 156 lx										
70% = 242 lx										
80% = 310 lx										
90% = 387 lx										
100% = 545 lx										

Pada gambar 19 merupakan grafik pengujian deteksi objek warna terhadap cahaya dari 3 objek warna yang diuji yaitu merah, biru dan orange



Gambar 19. Grafik deteksi warna terhadap intensitas cahaya


Pada gambar 20 merupakan grafik pengujian deteksi objek warna terhadap jarak dari 3 objek warna yang diuji yaitu merah, biru dan orange



Gambar 20. Grafik deteksi warna terhadap jarak

2. Deteksi objek warna
3. Penguncian poin
4. Pembentukan *Rectangle*
5. Pengambilan Gambar

Tabel 2 Pengujian *Blackbox* fitur sistem

NO	Hasil Pengujian	Keterangan
1		Media pilihan warna muncul/berhasil
2		Deteksi objek warna berhasil
3		Penguncian poin 1 dan poin 2 berhasil
4		Pembentukan garis <i>rectangle</i> berhasil
5		Pengambilan gambar bentuk <i>rectangle</i> berhasil

Pengujian Blackbox Fitur Sistem

Pengujian fitur sistem ini dilakukan dengan menggunakan metode *blackbox* agar dapat diketahui fitur sistem dapat berfungsi dengan baik seperti terlihat pada tabel 2. Pengujian fitur sistem yang dilakukan adalah:

1. Pemilihan warna

Kesimpulan

Berdasarkan penelitian yang telah dilakukan, didapat kesimpulan bahwa :

1. Segmentasi Citra dan pengambilan titik pusat untuk deteksi objek bergerak dapat diimplementasikan pada aplikasi deteksi objek warna untuk pengambilan citra bentuk *rectangle*.
2. Dari hasil pengujian yang telah dilakukan, dapat ditarik kesimpulan bahwa dari proses segmentasi citra untuk deteksi objek warna diketahui jarak terbaik adalah antara 40cm – 80cm. Sedangkan untuk deteksi warna terhadap intensitas cahaya, nilai lumiance terbaik adalah antara 22lx – 242lx.
3. Dari hasil pengujian pada aplikasi deteksi objek warna untuk pengambilan bentuk citra *rectangle*, didapatkan bahwa warna pada objek warna yang dideteksi tidak boleh sama atau terdapat pada latar *background* atau objek yang akan di *capture*.
4. Dari percobaan 3 warna yang dideteksi, warna merah adalah warna yang dapat terdeteksi dengan baik.

