KALBIS Scientia
Jurnal Sains dan Teknologi

ISSN 2356 - 4393

Aplikasi Perekam Citra Berdasarkan Pergerakan Objek Yang Nampak

Rizky Natanael Christianto[1)], Yulius Denny Prabowo[2)]

Teknik Informatika, Institut Teknologi dan Bisnis Kalbis
Jalan Pulomas Selatan kav.22, Jakarta 13210
[1)]*Email: rizkynatanael.c@gmail.com*
[2)]*Email: yulius.prabowo@kalbis.ac.id*

Abstract: *This research aims to build Image Record Application Based on Visualized the Object Movement, the application used to conserve storage capacity of the recording. Image Record Application Based on Visualized the Object Movement simulated with a camera on the notebook. This application is developed using prototype software development method. Computer vision used to perform image processing that appears on camera. Methods of computer vision used by author is background subtraction. Background subtraction method works by comparing the background image with the latest picture or new object captured by the camera. The results of the tests show that the memory capacity using the motion detection is smaller than not using motion detection.*

Keywords: *background subtraction, computer vision, motion detect, image processing*

Abstrak: *Penelitian ini bertujuan untuk membangun aplikasi perekam citra berdasarkan pergerakan objek yang nampak, aplikasi digunakan untuk menghemat penggunaan kapasitas media penyimpanan hasil perekaman. Aplikasi Perekam citra berdasarkan pergerakan objek yang nampak disimulasikan dengan kamera yang ada pada laptop. Aplikasi dikembangkan dengan metode pengembangan perangkat lunak prototipe. Pelihatan komputer digunakan untuk melakukan pengolahan pergerakan citra yang nampak pada kamera. Metode dari pelihatan komputer yang digunakan ialah background subtraction. Metode background subtraction bekerja dengan cara membandingkan background dengan gambar terbaru atau objek baru yang ditangkap oleh kamera. Hasil ujicoba menunjukkan bahwa kapasitas penyimpanan hasil perekaman yang menggunakan deteksi gerak lebih kecil dibandingkan dengan yang tidak menggunakan deteksi gerak.*

Kata kunci: *background subtraction, deteksi gerak, pelihatan komputer, pengolahan citra*

I. PENDAHULUAN

Penggunaan kamera pengawas sudah menjadi hal yang wajar dalam kehidupan manusia sekarang ini. Kamera pengawas digunakan untuk mengawasi serta merekam kegiatan-kegiatan atau aktivitas-aktivitas yang terjadi dalam suatu ruangan di tempat kamera tersebut diletakkan. Kamera pengawas yang biasa digunakan saat ini bekerja dengan cara melakukan perekaman secara terus-menerus, baik itu ditemukannya aktivitas-aktivitas dalam ruangan tersebut ataupun tidak terdapat aktivitas.

Cara kerja kamera pengawas yang demikian membuat besarnya penggunaan kapasitas media penyimpanan hasil perekaman. Dengan cara yang demikian sistem kamera pengawas dirasa kurang efektif dan membebani kapasitas media penyimpanan hasil perekaman.

Untuk mengatasi masalah tersebut, maka dibuat pemodelan kamera pengawas yang hanya akan merekam jika ditemukan adanya suatu aktivitas atau pergerakan, kamera tidak akan merekam jika tidak ditemukan pergerakan. Aplikasi ini dibuat menggunakan metode pada *computer vision, computer vision* digunakan untuk mengolah citra yang ditangkap oleh kamera pengawas. *Computer vision* merupakan proses otomatisasi yang akan mengintegrasikan sejumlah besar proses untuk persepsi visual yang bertujuan untuk mengkomputerisasi penglihatan manusia [1].

Dalam pemodelan aplikasi ini *computer vision* digunakan untuk memproses dan menganalisa citra yang ditangkap oleh kamera. Dengan demikian kamera tidak akan melakukan perekaman citra tanpa melalui pengolahan citra terlebih dahulu. Penggunaan aplikasi ini menghasilkan video hasil dari perekaman

citra yang akan bekerja jika hanya ditemukannya suatu aktivitas atau objek baru yang nampak pada kamera pengawas.

Aplikasi yang dikembangkan masih mempunyai beberapa batasan, objek yang diawasi harus memiliki pencahayaan yang cukup. Selain itu aplikasi masih belum dapat membedakan pergerakan objek maupun benda lain di sekitar objek. Hasil dari aplikasi ini adalah video dalam format *audio video interleave* (AVI). Aplikasi ini dibuat menggunakan WinPython 2.7.10. Hasil dari penelitian ini paling sesuai digunakan untuk memantau ruangan atau tempat dalam kondisi yang *statis* (minim pergerakan atau aktivitas), tanpa harus membebani media penyimpanan.

II. METODE PENELITIAN

Metode pengembangan perangkat lunak yang digunakan pada penelitian ini adalah metode *prototyping*. Metode *prototyping* merupakan suatu metode yang sering digunakan ketika pengguna hanya mampu membayangkan kebutuhan yang diinginkannya secara objektif dan tidak mampu menjelaskannya secara terperinci, baik itu mengenai masukan, proses, ataupun keluaran yang akan dihasilkan [2]. Dalam penggunaannya, metode *prototype* terdiri dari tahapan-tahapan yang dilakukan secara berulang, hal ini memungkinkan adanya interaksi dengan pengguna dalam upaya mencapai tujuan dari pengembangan aplikasi ini. Dalam pengembangannya, metode prototype ini terdiri dari 4 tahap [3],yaitu: (1) *Plan*; (2) *Specification*; (3) *Design*; dan (4) *Result*.

Plan merupakan tahap awal yang akan membantu peneliti dalam pembuatan *prototype* sesuai dengan kebutuhan yang diinginkan dan membantu dalam perencanaan pembuatan *prototype*. Berdasarkan tahap *plan* ini maka diperoleh hasil mengenai rencana atau rancangan yang ingin diperoleh dalam penelitian ini, yaitu bahwa penelitian ini bertujuan untuk membuat aplikasi yang dapat melakukan pengawasan terhadap suatu ruangan dan juga aplikasi ini hanya akan merekam jika ditemukannya suatu pergerakan atau objek baru.

Specification merupakan tahap lanjutan dari tahap *plan*. Pada tahap ini akan dilakukan pengambilan keputusan yang akan dibuat sesuai dengan hasil yang ada pada tahap *plan*. Pada tahap ini akan dilakukan pendefinisian karakteristik dari *prototype*, pemilihan metode yang akan digunakan, serta pemilihan alat yang akan digunakan dalam pembuatan *prototype*. Pada tahap ini pula ditemukan bahwa aplikasi melakukan pendeteksian ketika ditemukannya pergerakan atau objek baru.

Aplikasi ini menggunakan metode *background subtraction*, metode ini digunakan untuk melakukan pendeteksian gerak. Cara kerja dari metode ini adalah dengan cara mengambil citra dari citra yang nampak pada kamera sebagai citra acuan atau citra awal dan membandingkan citra yang dijadikan sebagai citra acuan itu dengan citra terbaru yang ditangkap oleh kamera. Dari hasil pembandingan itu akan diketahui ada atau tidaknya pergerakan atau objek baru. Jika ditemukan perbedaan antara citra acuan dengan citra terbaru maka citra terbaru akan dilihat sebagai objek atau benda yang bergerak atau adanya perubahan dibandingkan dengan gambar awal, sehingga aplikasi akan melakukan perekaman. Aplikasi ini dibuat dengan menggunakan bahasa pemrograman python yang diimplementasikan pada WinPython 2.7.10.

Design merupakan tahap lanjutan dari tahap *specification*, tahap ini berfokus pada penetapan desain dari *prototype* yang digunakan. Pada tahap ini diperoleh bahwa aplikasi ini akan bekerja dengan cara mengambil citra awal untuk dijadikan sebagai citra acuan dan juga akan mengambil citra terbaru yang nampak di depan kamera. Setelah itu akan dilakukan pembandingan antara citra acuan dengan citra terbaru. *Result* merupakan tahap di mana *prototype* telah selesai dikerjakan, namun pada *prototype* tersebut masih akan dilakukan evaluasi. Evaluasi ini dilakukan untuk mengetahui cara kerja dari prototype, apakah telah memenuhi kebutuhan yang telah didefinisikan pada tahap-tahap sebelumnya atau belum sesuai dengan kebutuhan.

A. Citra

Citra merupakan gambar yang ada pada bidang *dwimatra* atau dapat juga dikatakan sebagai gambar dua dimensi. Citra merupakan salah satu komponen multimedia yang memegang peranan penting sebagai informasi visual, serta memiliki karakteristik yang tidak dimiliki oleh data teks [4]. Keluaran citra yang berupa gambar atau foto disebut juga sebagai citra diam. Citra diam adalah citra tunggal yang tidak bergerak, selain itu juga terdapat istilah citra bergerak (*moving images*) yang artinya ialah rangkaian citra diam yang ditampilkan secara beruntun (*sekuensial*) sehingga memberi kesan pada mata kita sebagai gambar bergerak. Setiap citra di dalam rangkaian itu disebut *frame* [4].

Citra dibagi menjadi tiga jenis [5], yaitu:

* Citra Berwarna

Citra berwarna biasa dikenal sebagai citra RGB merupakan citra yang tersusun atas tiga komponen

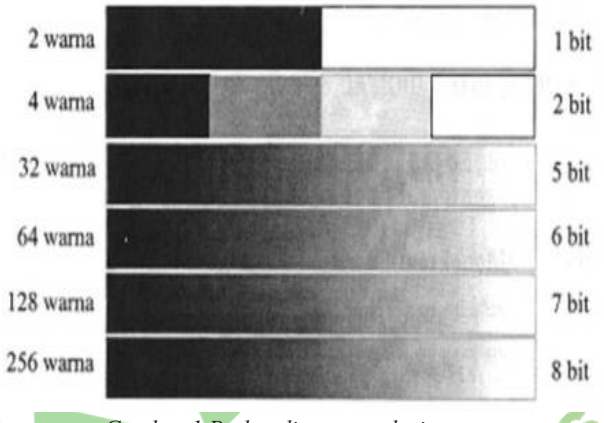
warna, yaitu komponen merah (*red* atau R), komponen hijau (*green* atau G), dan komponen biru (*blue* atau B). Setiap *pixel* dalam citra berwarna diwakili oleh tiga komponen tersebut (*red,green,blue*). Setiap warna dasar akan menggunakan penyimpanan sebesar 8 bit atau 1 *byte* [5]. Dengan demikian maka dalam setiap citra berwarna akan menggunakan penyimpanan sebesar 3 *byte*[6]. Setiap warna dasar memiliki gradasi sebanyak 255 warna, yang berarti *pixel* kombinasi warna sebanyak 2^8. 2^8. $2^8 = 2^{24} = 16$ juta warna lebih [6].

* Citra Berskala Keabuan

Citra berskala keabuan atau yang biasa dikenal dengan *grayscale* adalah citra yang menggunakan gradasi warna abu-abu yang merupakan kombinasi warna hitam dan warna putih.

Citra berskala keabuan ini menggunakan nilai bulat antara 0 sampai dengan 255. Secara sistematis rumus perhitungan untuk mengubah citra berwarna menjadi citra *grayscale* adalah sebagai berikut:

$$F_o(x,y) = (f_i^R(x,y) + f_i^G(x,y) + f_i^B(x,y))/3 \quad [6]$$



Gambar 1 Perbandingan gradasi warna grayscale

* Citra Biner

Citra biner biasa dikenal sebagai citra monokrom atau citra hitam-putih. Citra biner memiliki nilai *pixel* berupa angka 0 dan 1 saja. Citra ini digunakan untuk kepentingan segmentasi yang memisahkan objek dengan latar belakangnya.

B. Pengolahan Citra

Pengolahan citra merupakan suatu pemrosesan citra yang secara khusus menggunakan mesin atau komputer untuk mendapatkan citra dengan kualitas yang lebih baik atau dapat juga dibilang sebagai kegiatan untuk memperbaiki kualitas citra agar mudah diinterpretasikan oleh manusia ataupun komputer, sehingga dapat disimpulkan bahwa pengolahan citra memiliki masukkan berupa citra dan keluaran berupa citra namun dengan kualitas citra yang lebih baik [4].

C. Video

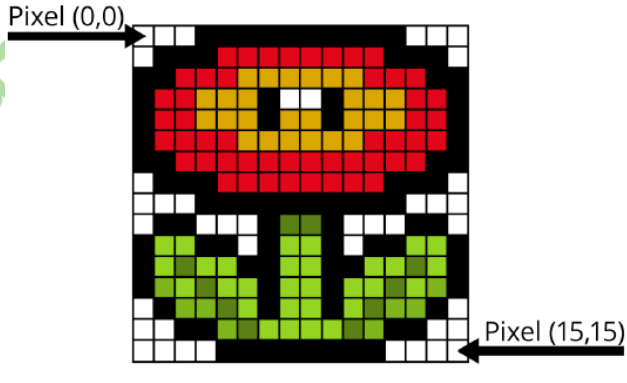
Video merupakan citra diam yang di tangkap oleh kamera dalam rentang waktu yang konstan secara terus menerus atau dapat juga dikatakan sebagai kumpulan citra yang diputar secara berurutan dengan kecepatan waktu tertentu, sehingga citra terlihat seperti bergerak. Citra yang merupakan bagian dari video disebut *frame* [7].

D. Frame Rate

Frame Rate merupakan kecepatan dalam pengambilan citra atau bisa juga sebagai kecepatan dalam membaca citra. *Frame rate* memiliki satuan yang dikenal sebagai *frames per second (fps)* [7]. Untuk menghasilkan gambar yang halus, diperlukan minimal 10 *fps* [8].

E. Picture Element

Picture element atau yang biasa dikenal dengan *pixel* merupakan satu titik dalam suatu grid yang berbentuk persegi, atau ribuan titik individual yang menyusun suatu citra. *Pixel* merupakan unit terkecil dari perangkat keras pencetak atau *display* [9]. Sebuah citra yang memiliki resolusi 1024x768 merupakan grid yang terdiri dari 1024 kolom dan 768 baris, sehingga diperoleh *pixel* sebesar 786,432 *pixel* (1024x768 = 786,432 *pixel*) [10].



Gambar 2 Contoh citra yang memiliki ukuran 15 x 15 pixel [10]

F. Kedalaman Warna

Kedalaman warna atau *bit depth* merupakan ukuran untuk mengukur halus atau kasarnya pembagian tingkat gradasi warna. Kedalaman warna berfungsi untuk menentukan berapa banyaknya informasi warna yang tersedia untuk ditampilkan dalam setiap *pixel*. Semakin besar nilai kedalaman warna maka akan semakin bagus juga kualitas dari

citra yang dihasilkan [6], seperti pada Tabel 1.

Tabel 1 Hubungan kedalaman warna dengan resolusi warna

Kedalaman Warna	Resolusi Warna
1 bit	2 warna
4 bit	16 warna
8 bit	256 warna
16 bit	65.563 wanra
24 bit	16.777.216 warna

G. *Video Size*

Video *size* merupakan ukuran dari video yang dihasilkan dan di simpan di dalam memori. Untuk mengetahui perkiraan besaran memori yang dibutuhkan dari video selama waktu tertentu, maka digunakan cara perhitungan [7] sebagai berikut: (1) Hitung total pixel. Tinggi frame * Lebar frame; (2) Hitung besaran bit per frame. Total pixel * kedalaman warna (*bit depth*); dan (3) Hitung besaran bit selama waktu tertentu. Bit per frame * frame rate * waktu.

Contoh : Ukuran frame = 640 x 480, kedalaman warna = 24 bit/pixel , durasi 1 detik, dan *frame rate* = 20 fps. maka: (1) Total pixel = 640 x 480 = 307.200 pixel/frame; (2) Bit per frame = 307.200 pixel/frame * 24 bit/pixel = 7.372.800 bit/frame; (3) Ukuran bit selama 1 detik = 7.372.800 bit/frame * 20 frame/second * 1 second = 147.456.000 bit; dan (4) Ukuran video dalam bytes = 147.456.000 bit / (8 bit/byte) = 18.432.000 byte = 18 megabyte

H. Audio Video Interleave

Audio Video Interleave (AVI) merupakan format video yang tidak dikompresi. AVI merupakan format video standar tertua yang sudah ada sejak Windows 3.1 dan menjadi format standar video untuk Microsoft Windows. Penggunaan video dengan format ini akan menghasilkan ukuran *file* yang besar, hal ini dikarenakan resolusi yang dipakai sesuai dengan resolusi asli dari sumber videonya [11].

I. Python

Python merupakan bahasa pemorgraman yang dikembangkan pada akhir tahun 1980-an di National Research Institute oleh Guido van Rossum. Python merupakan bahasa pemrogra-man yang dinilai fleksibel, hal ini dikarenakan bahasa pemrograman python sebagian besar menggunakan bahasa yang dapat dikenali (bahasi Inggris) oleh manusia. Bahasa python tidak seperti bahasa assembler yang hanya dapat dimengerti oleh orang-orang yang sudah memiliki pengalaman saja. Python dipublikasikan dalam lisensi *open-source* dan tersedia untuk sistem operasi Linux, OS X, dan Windows [12].

J. Background Subtraction

Background subtraction merupakan cara yang sering digunakan untuk pemrosesan awal aplikasi yang berbasis pada penggunaan penglihatan atau *vision*, salah satunya seperti yang digunakan pada sebuah kasir, di mana kamera statis digunakan untuk menghitung pengunjung yang datang ataupun pergi. Oleh karena penggunaan yang sederhana dan keberadaan lokasi kamera yang tetap, maka *background subtraction* digunakan sebagai operasi pengolahan gambar yang mendasar untuk aplikasi video pengaman [13].

Dalam penggunaannya, *background subtraction* akan membandingkan antara latar belakang (*background*) dengan latar depan (*foreground*). Latar belakang yang dimaksud adalah latar belakang yang bersifat tetap dan tidak bergerak di depan kamera, sedangkan latar depan merupakan semua objek baru yang terdapat di depan kamera selain latar belakang.

$$S(n) = \left| F(n) - B \right|_{[14]}$$

S(n) : hasil selisih frame ke-n dengan *background*
F(n) : frame ke-n dari masukan citra
B : latar belakang atau *background*

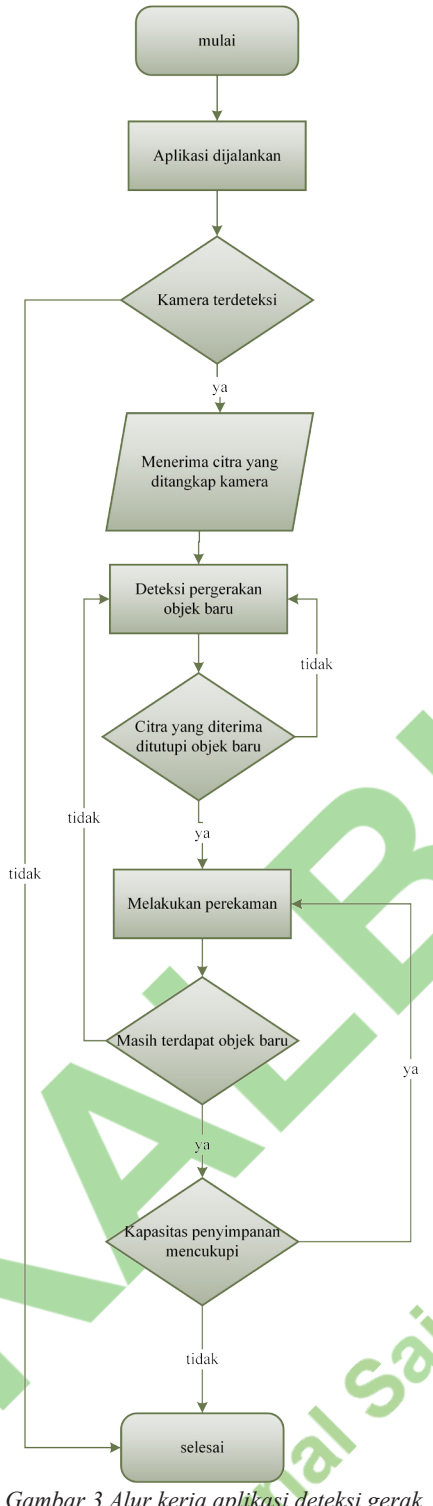
K. Thresholding

Thresholding dilakukan dengan menggunakan nilai mutlak atau *absolut*. *Thresholding* merupakan cara yang digunakan dalam pengolahan citra, untuk mengkonversi citra *grayscale* atau derajat keabuan menjadi citra biner.

Operasi ini dilakukan dengan mengelompokkan nilai derajat keabuan pada setiap *pixel* ke dalam 2 kelas, hitam dan putih. Pengkonversian ini dilakukan dengan tujuan untuk mengidentifikasi keberadaan suatu objek, dengan cara memisahkan objek dari gambar latar belakangnya. *Pixel-pixel* pada objek akan dinyatakan dengan nilai 1, sedangkan pada pixel latar dinyatakan dengan nilai 0, pengelompokkan ini mengacu pada nilai *threshold* atau nilai batas yang telah ditetapkan [4].

L. Alur Kerja Aplikasi

Alur kerja dari aplikasi perekam citra ini dapat dilihat pada Gambar 3. Cara kerja aplikasi yang ditunjukkan pada Gambar 3 ini diawali dengan melakukan pendeteksian ada atau tidaknya kamera yang terhubung dengan aplikasi, jika tidak ditemukan adanya kamera, maka aplikasi ini akan berhenti berjalan. Namun jika ditemukan adanya kamera yang dapat terbaca oleh aplikasi, maka alur kerja aplikasi



Gambar 3 Alur kerja aplikasi deteksi gerak

akan berlanjut ke tahap pengambilan citra awal untuk dijadikan sebagai citra acuan.

Setelah itu kamera akan terus menerus melakukan pengambilan citra, di mana pada tahap ini aplikasi mulai melakukan pendeteksian gerak berdasarkan perbandingan antara citra acuan dengan citra terbaru. Bila terdapat perbedaan antara citra acuan dengan citra terbaru dan perbedaan ini melebihi nilai *threshold* yang telah ditentukan, maka perbedaan itu dianggap sebagai pergerakan / objek baru, dengan begitu kamera akan melakukan perekaman citra secara terus-menerus sampai tidak ditemukannya perbedaan antara citra acuan dengan citra terbaru dan selama kapasitas penyimpanan memori mencukupi.

Terjadi pengolahan citra ketika menjalankan deteksi pergerakan objek baru yang ada pada Gambar 3. Pengolahan citra yang terjadi itu antara lain: (1) Mengubah citra berwarna menjadi citra berskala keabuan. Warna dari citra yang direkam oleh kamera sebelum melalui pengolahan ialah RGB, namun untuk dilakukan pendeteksian gerak, maka warna RGB tersebut dikonversi menjadi warna berskala keabuan atau *grayscale*; (2) Menghilangkan *noise*. Pada citra yang telah mengalami proses konversi kemudian akan dilakukan proses penghilangan *noise-noise* yang ada untuk memudahkan dalam pendeteksian gerak; (3) Pembandingan citra. Pada tahap ini akan dilakukan pembandingan antara citra acuan dengan citra terbaru yang nampak pada kamera. Tahap ini dilakukan dengan cara menghitung selisih antara citra terbaru dengan citra acuan (citra terbaru dikurang dengan citra acuan); dan (4) *Thresholding*. Pada tahap dilakukan pengecekkan nilai dari hasil tahap sebelumnya dengan nilai *threshold*. Jika nilai hasil pada tahap sebelumnya lebih besar dibandingkan dengan nilai *threshold*, maka akan terdeteksi sebagai pergerakan atau objek baru.

M. Perangkat Pengembangan Aplikasi

Perangkat yang digunakan dalam pengembangan aplikasi perekam citra ini dapat dilihat pada Tabel 2.

Tabel 2 Perangkat yang digunakan

Perangkat Keras	Perangkat Lunak
Kamera	WinPython 2.7.10
Laptop/Notebook	
	OpenCV 3.1.0

N. Pengujian

Pengujian dalam aplikasi dilakukan untuk mengetahui ketercapaian dari tujuan aplikasi ini, yaitu menghemat penggunaan kapasitas memori penyimpanan. Pengujian ini dilakukan dengan cara menjalankan aplikasi dengan menggunakan dua algoritma yang dibuat dengan menggunakan bahasa pemrograman python, di mana algoritma yang satu dengan menggunakan deteksi gerak dan yang satu lagi tanpa menggunakan deteksi gerak.

III. HASIL DAN PEMBAHASAN

Aplikasi perekam citra berdasarkan objek yang nampak ini dibuat dengan menggunakan bahasa

pemrograman python yang diimplementasikan dengan menggunakan WinPython 2.7.10, hasil dari perekaman yang disimpan berupa video dengan format avi. Untuk melakukan pendeteksian gerak maka dilakukanlangkan pengambilan citra, pengolahan citra dan pendeteksian gerak. Ketiga hal tersebut diterapkan dalam tahap inmplementasi. Penjelasan setiap tahapan adalah sebagai berikut:

A. Pengambilan Citra

Pada tahap ini aplikasi akan melakukan perintah kepada kamera untuk melakukan pengambilan citra yang kemudian akan dijadikan sebagai citra awal (citra acuan) maupun sebagai citra terbaru (citra terbaru baru akan diambil setelah citra awal sudah didapati). Hasil dari pengambilan citra ini dapat dilihat pada Gambar 4.

Gambar 4 Citra awal

Kode program untuk pengambilan gambar seperti yang ada pada Gambar 4, yaitu:

```
while (kam.isOpened):
    (ambil_gambar, frame) = kam.read()
```

Cuplikan kode program tersebut digunakan untuk membaca kamera dan mengambil gambar yang kemudian akan disimpan pada variabel *frame*.

B. Mengolah Citra

Pada tahap ini, citra yang telah diambil oleh kamera kemudian akan diolah. Citra yang semula memiliki warna RGB akan dikonversi menjadi citra berskala keabuan. Setelah diubah menjadi citra berskala keabuan, kemudian pada citra tersebut akan dilakukan penghilangan *noise* dengan menggunakan *Gaussian Blur*. Berikut ini merupakan kode program yang digunakan untuk melakukan konversi tersebut.

```
frame_sekarang = cv2.cvtColor(frame,
                    cv2.COLOR_BGR2GRAY)
frame_sekarang=cv2.GaussianBlur(frame_sekarang, (21,21),0)
```

Pada cuplikan kode program tersebut, variabel frame_sekarang digunakan untuk menampung citra hasil dari dikonversi warna RGB menjadi citra berskala keabuan atau *grayscale*. cv2.cvtColor merupakan kode program yang digunakan untuk melakukan konversi citra RGB yang disimpan di dalam variabel frame menjadi citra berskala keabuan. Kode cv2.GaussianBlur digunakan untuk menghilangkan *noise* yang masih ada pada citra setelah dilakukannya pengkonversian warna.

C. Mendeteksi Gerak

Pada tahap ini akan dilakukan pendeteksian gerak dengan cara membanding antara citra acuan dengan citra terbaru. Pembandingan ini dilakukan dengan cara melihat nilai selisih antara citra acuan dengan citra terbaru. Jika hasil selisih yang terjadi melebihi nilai threshold yang ditetapkan, maka hal itu akan dianggap sebagai pergerakan. Berikut ini merupakan cuplikan program yang digunakan untuk melakukan hal tersebut.

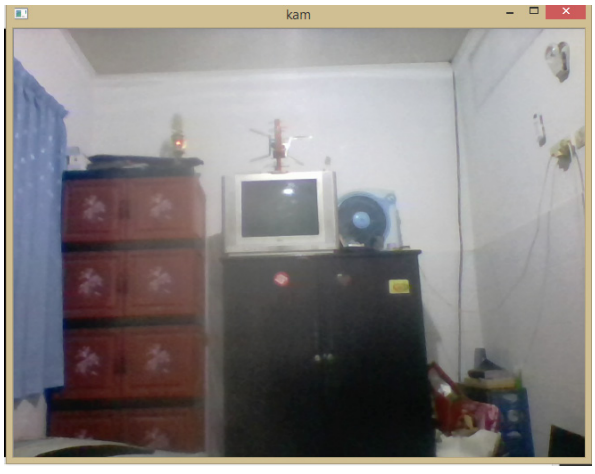
```
banding = cv2.absdiff(f_awal,
                    frame_sekarang)
thresh = cv2.threshold(banding, 80, 255,
                    cv2.THRESH_BINARY)[1]
```

Pada cuplikan kode program di atas, variabel banding digunakan untuk menampung hasil dari pembandingan yang dilakukan antara citra acuan (di simpan di dalam variabel f_awal) dengan citra terbaru (di simpan dalam variabel frame_sekarang). Pembandingan ini dilakukan dengan menggunakan kode program cv2.absdiff.

Setelah dilakukan pembandingan, maka hasil dari pembandingan tersebut akan dibandingkan kembali dengan nilai ambang batas atau *threshold* yang telah ditentukan. Bila mana nilai tersebut berada di bawah dari nilai *threshold,* maka nilai *pixel* dari citra itu akan berubah menjadi 0 dan jika berada di atas nilai *threshold,* maka nilai *pixel* dari citra tersebut berubah menjadi 1. *Pixel* yang memiliki nilai 0 merupakan *pixel* dari citra acuan atau *background,* sedangkan *pixel* dengan nilai 1 merupakan *pixel* dari objek baru yang bukan merupakan bagian dari *background.* Kode untuk melakukan *thresholding* ini adalah cv2.threshold. *Thresholding* ini akan menghasilkan citra biner atau citra hitam-putih.

D. Implementasi

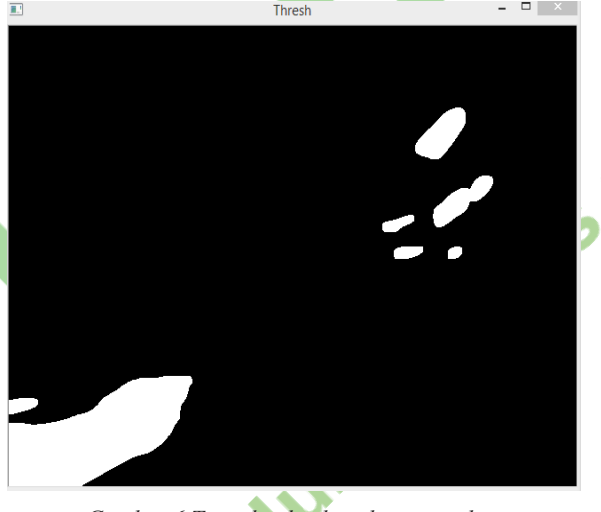
Pada tahap ini akan dilakukan pengujian terhadap cara kerja dari aplikasi deteksi gerak yang telah dibuat.



Gambar 5 Tampilan awal

Pada saat pertama kali aplikasi dijalankan, aplikasi akan memberikan perintah terhadap kamera untuk melakukan pengambilan citra yang akan dijadikan sebagai citra awal atau citra acuan. Gambar 5 merupakan citra awal yang disimpan untuk dijadikan sebagai citra acuan. Citra acuan ini nantinya akan digunakan untuk membandingkan dengan citra terbaru yang ditangkap oleh kamera.

Tampilan yang ada pada Gambar 6, merupakan tampilan hasil pengolahan terhadap citra awal atau citra acuan yang ditangkap oleh kamera, dengan membandingkan citra acuan tersebut dengan citra terbaru. Warna putih pada Gambar 6 menandakan adanya pergerakan yang terdeteksi oleh aplikasi ini.



Gambar 6 Tampilan ketika ada pergerakan

E. Uji Coba dengan Deteksi Gerak

Pada tahap ini akan dilakukan uji coba perekaman citra yang akan dilakukan dengan menggunakan algoritma deteksi gerak. Berikut ini merupakan algoritma yang akan digunakan dalam pengujicobaan.

```
while (kam.isOpened):
 (ambil_gambar, frame) = kam.read()
  if not ambil_gambar:
                break
 frame = imutils.resize(frame, width=640,
            height=480)
  frame_sekarang = cv2.cvtColor(frame,
             cv2.COLOR_BGR2GRAY)
  frame_sekarang = cv2.GaussianBlur(frame_sekarang,
        (21, 21), 0)
    if f_awal is None:
      f_awal = frame_sekarang
      continue
  banding = cv2.absdiff(f_awal,
             frame_sekarang)
  thresh = cv2.threshold(banding, 80, 255,
            cv2.THRESH_BINARY)[1]
     thresh = cv2.dilate(thresh, None,
            iterations=2)
  (_, kontur, _) = cv2.findContours
            (thresh.copy(),
             cv2.RETR_EXTERNAL,
          cv2.CHAIN_APPROX_SIMPLE)
  for k in kontur:
    if cv2.contourArea(k) < 50:
        continue
    out.write(frame) #rekam
```

Cuplikan kode yang tersebut merupakan kode yang digunakan untuk melakukan perekaman citra bila ditemukannya suatu pergerakan yang dapat dilihat oleh kamera. Cara kerja kode ini diawali dengan dapat atau tidaknya citra yang ditangkap kamera yang disimpan di variabel frame. Bila citra sudah dapat tersimpan dalam variabel frame, maka citra awal ini akan dijadikan sebagai cira acuan atau citra *background* yang akan disimpan dalam variabel f_awal. Sebelum disimpan dalam variabel f_awal, citra tersebut akan diolah terlebih dahulu dengan cara mengkonversi warna RGB yang ada pada citra tersebut menjadi citra berskala keabuan atau grayscale, kemudian menghilangkan noise yang ada dengan menggunakan GaussianBlur.

Setelah mendapatkan citra yang dijadikan sebagai background, maka aplikasi ini akan terus-menerus mengambil citra untuk dibandingkan dengan citra background tersebut. Aplikasi akan melakukan perekaman secara terus-menerus selama ditemukannya perbedaan antara citra terbaru dengan citra background dan hal itu melebihi nilai threshold yang telah ditentukan, perekaman akan terus dilakukan sampai tidak ditemukannya perbedaan antara citra terbaru dengan citra background. Dengan menggunakan algoritma deteksi gerak, maka

dilakukan pengujicobaan sebanyak 10 kali dengan rentang waktu 10 detik yang dilakukan dalam sebuah ruangan. Pada Tabel 3, dapat dilihat rata-rata dari hasil uji coba yang dilakukan oleh penulis, berdasarkan 10 kali pengujicobaan.

Tabel 3 Hasil percobaan dengan menggunakan deteksi gerak

No	Lama Merekam (detik)	Frame Size Rekaman	Ukuran File Rekaman
1	10	160 x 120	490,291 bytes
2	10	240 x 180	1,074,381 bytes
3	10	320 x 240	2,079,539 bytes
4	10	640 x 480	3,903,898 bytes

Pada Tabel 3 menunjukkan bahwa rata-rata hasil perekaman citra yang dilakukan dengan *frame size* 160 x 120 menggunakan penyimpanan memori sebesar 490,291 *bytes*, *frame size* 240 x 180 sebesar 1,074,381 *bytes*, *frame size* 320 x 240 sebesar 2,079,539 *bytes*, dan *frame size* 640 x 480 sebesar 3,903,898 *bytes*.

F. Uji Coba Tanpa Deteksi Gerak

Pada tahap ini akan dilakukan uji coba perekaman citra yang akan dilakukan tanpa menggunakan algoritma deteksi gerak. Berikut ini merupakan algoritma yang akan digunakan dalam pengujicobaan.

```
while (kam.isOpened):
  (ambil_gambar, frame) = kam.read()
    if not ambil_gambar:
      break
   frame = imutils.resize(frame, width=640
                ,height=480)
   out.write(frame)
```

Cuplikan kode di atas, merupakan kode yang digunakan untuk menjalankan aplikasi tanpa menggunakan deteksi gerak. Cara kerja kode ini diawali dengan penyimpanan citra yang ditangkap oleh kamera ke dalam variabel frame, jika tidak dapat melakukan penyimpanan citra, maka aplikasi ini akan berhenti bekerja. Setelah berhasil melakuan penyimpanan citra, maka kode program ini akan terus melakukan perulangan secara terus-menerus dan menyimpan citra yang ditangkap oleh kamera ke dalam viarabel frame untuk nantinya disimpan ke dalam memori.

Dengan menggunakan algoritma yang tidak menggunakan deteksi gerak, maka dilakukan pengujicobaan sebanyak 10 kali dengan rentang waktu 10 detik yang dilakukan dalam sebuah ruangan. Pada Tabel 4, dapat dilihat rata-rata dari hasil uji coba yang dilakukan oleh penulis, berdasarkan 10 kali pengujicobaan yang dilakukan.

Tabel 4 Hasil percobaan tanpa menggunakan deteksi gerak

No	Lama Merekam (detik)	Frame Size Rekaman	Ukuran file Rekaman
1	10	160 x 120	1,126,400 bytes
2	10	240 x 180	2,058,240 bytes
3	10	320 x 240	2,959,770 bytes
4	10	640 x 480	8,165,376 bytes

Pada Tabel 4 menunjukkan bahwa rata-rata hasil perekaman citra yang dilakukan dengan *frame size* 160 x 120 menggunakan penyimpanan memori sebesar 1,126,400 *bytes*, *frame size* 240 x 180 sebesar 2,058,240 *bytes*, *frame size* 320 x 240 sebesar 2,959,770 *bytes*, dan *frame size* 640 x 480 sebesar 8,165,376 *bytes*.

IV. SIMPULAN

Berdasarkan hasil dari analisa dan pengujian yang telah dilakukan pada penelitian ini, maka dapat diperoleh kesimpulan sebagai berikut: (1) Perekaman citra berdasarkan pergerakan objek dapat berjalan dengan dengan sangat baik. Aplikasi ini dapat membedakan antara citra acuan dengan citra terbaru yang tedeteksi sebagai pergerakan dan hanya akan melakukan perekaman jika ditemukannya pergerakan; (2) Metode *background subtraction* dapat digunakan dalam pendeteksian objek baru atau deteksi pergerakan; (3) Tahapan-tahapan pengolahan citra yang digunakan untuk mendeteksi pergerakan; dan (4) Seperti terlihat pada Tabel 3 dan Tabel 4, ukuran file hasil dari perekaman citra dengan menggunakan deteksi gerak lebih kecil daripada yang tidak menggunakan deteksi gerak.

V. DAFTAR RUJUKAN

[1] D. Putra, Pengolahan Citra Digital, Yogyakarta: Andi, 2010, pp. 1-2.

[2] V. Yasin, Rekayasa Perangkat Lunak Berorientasi Objek, Jakarta: Mitra Wacana Media, 2012, hlm. 19-20.

[3] J. Arnowitz, M. Arent and N. Berger, "Effective Prototyping For Software Makers," San Francisco, Dianne Cerra, 2007, hlm. 21-25.

[4] F. A. Hermawati, Pengolahan Citra Digital : Konsep dan Teori, Yogyakarta: Andi, 2013, hlm. 1-3.

[5] A. Kadir, Dasar Pengolahan Citra dengan Delphi, Yogyakarta: Andi, 2013.

[6] T. Sutoyo, E. Mulyanto, V. Suhartono, O. D. Nurhayati and Wijanarto, Teori Pengolahan Citra Digital, Yogyakarta: Andi, 2009, hlm. 18.

[7] Y. L. Wong, Digital Media Primer: Digital Audio, Video, Imaging and multimedia Programming,

Upper Saddle River: Pearson Education, 2013, hlm. 142;159;175-176.

[8] A. Setiawan, "Deteksi Gerak Satu Obyek pada Video AVI," *Jurnal Sains dan Teknologi,* vol. 4, hlm. 133-137, 2011.

[9] J. Simarmata and T. Chandra, Grafika Komputer, Yogyakarta: Andi, 2007, hlm. 16-18.

[10] K. Demaagd, A. Olive, N. Oostendorp and K. Scott, Practical Computer Vision, Sebastopol: O'Reilly Media, 2012, hlm. 51;54.

[11] Heriansyah, Febriani and Willy, "Rancang Bangun Pendeteksian Gerakan Tangan," *Seminar Perkembangan dan Hasil Penelitian Ilmu Komputer,* hlm. 334-343, 2014.

[12] E. Upton and A. Halfarcee, Raspberry Pi User Guide, Chicester: John Wiley & Sons, 2012, hlm. 104.

[13] G. Bradski and A. Kaehler, Learning OpenCV, Sebastopol: O`Reilly Media, 2008, hlm. 265-266.

[14] J. K. Candra, I. K. Timotius and I. Setyawan, "Sistem Pendeteksi Orang Tergeletak berbasis sebuah Kamera Pengawas dengan menggunakan metode Template Matching," *Jurnal Cybermatika,* vol. 1, hlm. 1-5, 2013.