# BAB IV

# HASIL DAN PEMBAHASAN

## 4.1 Implementasi Sistem

Tahapan ini dilakukan setelah perancangan sistem selesai dilakukan dan selanjutnya diimplementasikan pada bahasa pemrograman yang digunakan. Tujuan implementasi sistem adalah untuk menerapkan perancangan yang telah dilakukan terhadap perangkat lunak sehingga nantinya maksud dan tujuan pembangunan perangkat lunak dapat tercapai.

### 4.1.1 Persiapan Kebutuhan dan Instalasi Program

Program yang telah dibuat melakukan proses deteksi dan pembelajaran. Proses deteksi pada gambar tidak bergerak pada umumnya tidak memerlukan proses yang berat karena sifatnya hanya mensimulasikan data input terhadap model matematik yang telah disusun. Proses pada suatu video sifatnya berat dan penting, karena deteksi dilakukan terus menerus sehingga akan menyebabkan video terasa lebih lambat apabila spesifikasi prosesor pengguna tidak memadai, oleh karena itu sebaiknya program dijalankan pada komputer yang memiliki kemampuan komputasi yang cukup cepat.

Adapun pelaksaan implementasi ini dilakukan pada komputer dengan spesifikasi seperti pada **tabel 4.1** Spesifikasi Perangkat Keras:

**Tabel 4.1 Spesifikasi Perangkat Keras**

| Spesifikasi Perangkat Keras | |
|---|---|
| Procesor | Intel Core 2 Duo @1.7 GHz |
| Memori | 4GB DDR3 |
| Harddisk | 320GB SATA |
| VGA | Intel Graphics Media Accelerator HD |

Spesifikasi perangkat lunak yang digunakan mempunyai spesifikasi seperti pada **tabel 4.2** sebagai berikut:

**Tabel 4.2 Spesifikasi Perangkat Lunak**

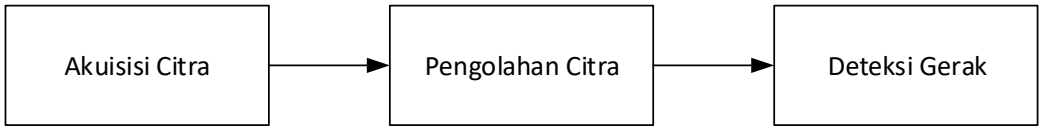
| Spesifikasi Perangkat Lunak | |
|---|---|
| Operating system | Windows10 |
| Bahasa pemrograman | Matlab 2015 A |

### 4.1.2 Instalasi Matlab 2015A

Dalam instalasi Matlab Jika menggunakan *Operating System Windows Xp* dan Vista program akan menghadapi masalah, dikarenakan versi matlab sudah memasuki versi terbaru yaitu 2015a, tetapi jika menggunakan OS Windows 7 ataupun windows 10 maka program akan berjalan dengan semestinya.

### 4.1.3 Proses Sistem Deteksi Gerak Berdasarkan Warna

Proses pendeteksian gerak pada sistem deteksi gerak berdasarkan warna ini secara garis besar dapat diliat pada **gambar 4.1** diagram blok system deteksi gerak.



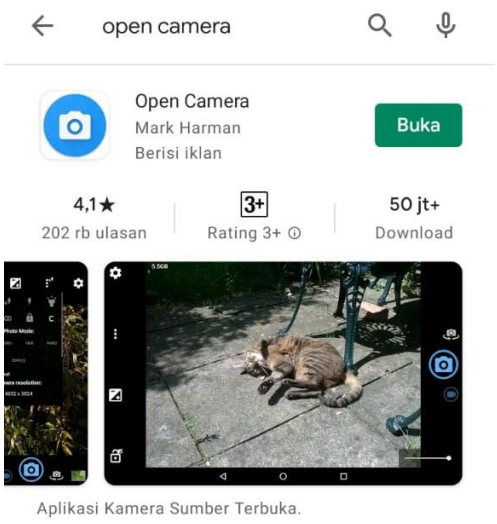

**Gambar 4.1 Diagram Blok Sistem Deteksi Gerak**

### 4.1.4 Akuisisi Citra

Akuisisi citra adalah pengambilan gambar dalam penelitian ini akuisisi citra menggunakan *camera handphone* android Redmi Note 9 pro yang setelah itu langsung di pindahkan ke laptop Toshiba Satelite c640.

#### 4.1.4.1 Mengambil Citra Video

Untuk dapat mengakuisisi citra dalam pengimplementasiannya digunakan aplikasi pihak ketiga pada android yaitu aplikasi *Open Camera*. Bentuk *source* aplikasi yang digunakan dapat dilihat pada **gambar 4.2**:

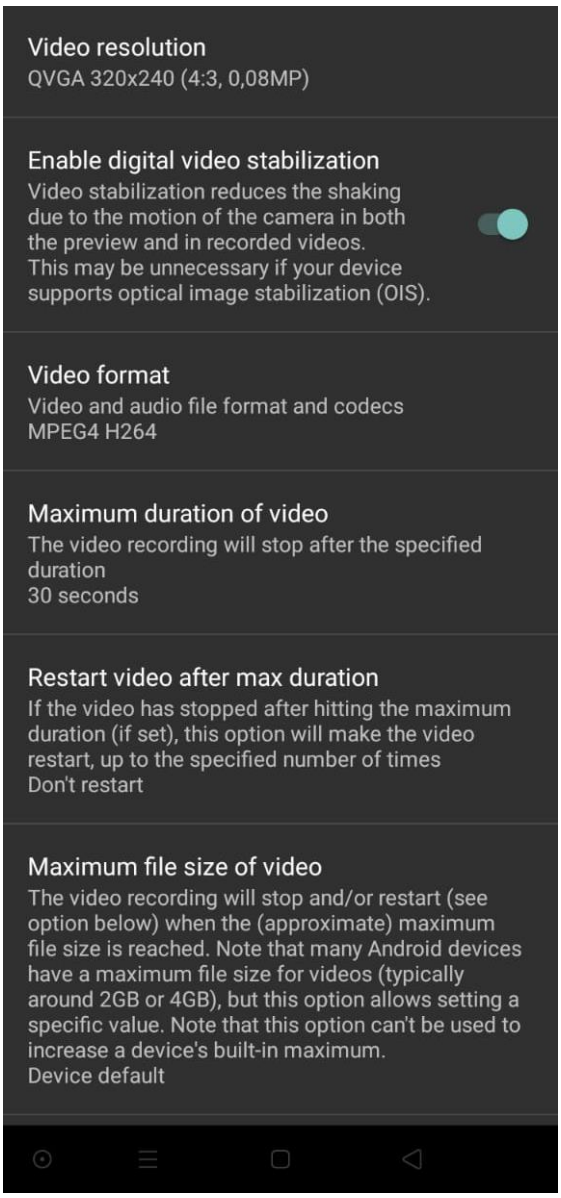



**Gambar 4.2 Aplikasi Open Camera**

Dengan aplikasi pada *open camera* pengambilan video dapat di sesuaikan dengan kebutuhan yang dimana data video menggunakan format MP4 durasi 15 detik dan menggunakan 16 frame/s.

#### 4.1.4.2 Pengaturan Properti Video

Pengaturan properti video dilakukan untuk mendapatkan video input sesuai dengan kebutuhan sistem. Seperti yang terlihat pada **gambar 4.3** *source code property* video objek.



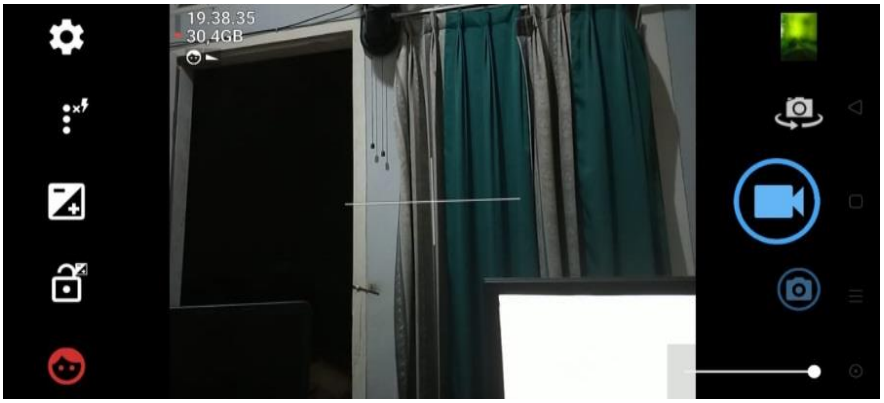

**Gambar 4.3 Setingan Properti Pengaturan Video**

Dengan *source code* diatas maka sistem akan mendapatkan video input yang terdiri dari *frame–frame* atau citra yang memiliki nilai RGB dan *frame* dari video input diambil secara terus menerus dan akan mengakuisisi 16 *frame* per

detiknya dengan maksimal durasi pada setiap video 15 detik dengan hasil video berformat MP4.

#### 4.1.4.3 Pengambilan Gambar

Setelah mengaktifkan dan mengatur properti video aplikasi siap untuk mengambil video berfungsi sebagai data yang akan kita deteksi. Dapat di lihat pada **gambar 4.4** berikut:




**Gambar 4.4 Source Code Memulai Video**

Dengan pengambilan video pada **gambar 4.4** maka proses pengambilan gambar akan dimulai secara terus menerus hingga sistem mengakuisisi 16 *frame* dan dengan waktu maksimal 15 detik per video yang kemudian akan diolah secara terpisah pada proses pengolahan citra seperti terlihat pada **gambar 4.5** tahapan proses pengolahan citra pada sistem indetifikasi orang pada video menggunakan metode *background subtraction*.

### 4.1.5 Proses Pengolahan Citra

Video adalah suatu citra atau *frame* yang putar secara berurutan dengan kecepatan tertentu sehingga terlihat seperti bergerak. Maka dalam sistem ini video akan diolah berdasarkan citra atau frame yang diakuisisi oleh sistem secara berurutan. Untuk itu di perlukan *code* untuk menjalankan video menggunakan matlab, adapun *source*

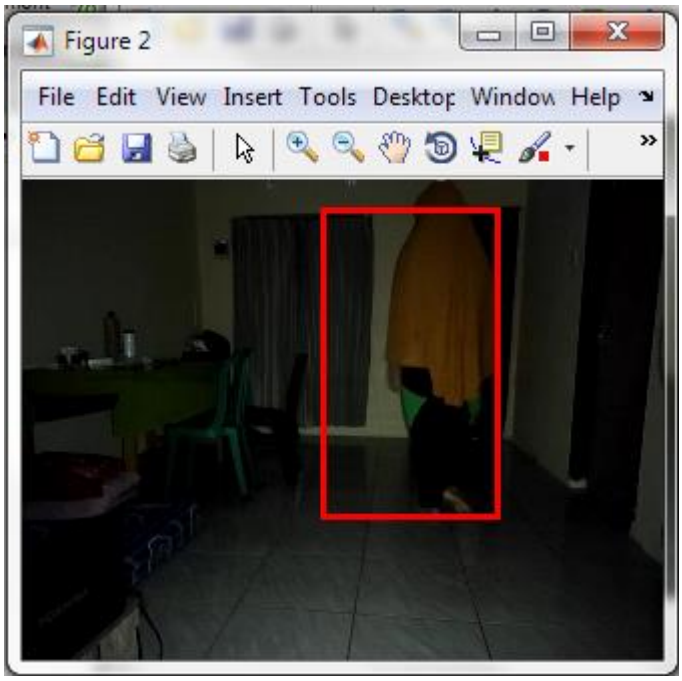
*code*nya sebagai berikut:

```
clc;clear;close all;

vid = VideoReader('SampleVideo.mp4');
```

**Gambar 4.5 Source Code Menjalankan File Video**

Dengan menggunakan *source code* diatas *file* video dengan nama *SampleVideo* dan dengan format MP4 akan di jalankan. Berikut gambar video yang di jalankan:




**Gambar 4.6 Screen Shot Video yang di Jalankan**

Untuk *file* video di atas menggunakan nilai *properties* video dengan nilai sebagai berikut:

Duration: 15 detik
Width: 320
Height: 240
FrameRate: 16.0000

BitsPerPixel: 424
VideoFormat: ‘Mp4’

Video tersebut memiliki durasi selama 15 detik dan *frame* rate sebesar 16 *frame* per detik sehingga banyaknya *frame* ketika diekstrak adalah 15x16=240 *frame*. Tampilan *frame* pada setiap detik ditunjukkan pada gambar di bawah ini:

**Gambar 4.7 Tampilan Gambar Pada Setiap Frame**

Video di ekstrak sebanyak jumlah *frame* yang ada (total frame = durasi video X *Framerate* video) dengan panjang durasi video 15 detik dan *framerate* sebesar 16 maka akan mengekstrak 240 *foto* yang berbeda pada video. Berikut coding untuk mengekstrak *frame* dari video:

```
vidWidth = vid.Width;
vidHeight = vid.Height;
mp4 = struct('cdata',zeros(vidHeight,vidWidth,240,'uint8'),...
    'colormap',[]);
frames = zeros(vidHeight,vidWidth,3,240);

k = 1;
while hasFrame(vid)
    frames(:,:,:,k) = readFrame(vid);
    k = k+1;
end
```

**Gambar 4.8 Source Kode Menampilkan Setiap Frame Pada Video**

Adapun tahapan proses pengolahan citra yang harus dilalui suatu citra agar dapat terdeteksi oleh sistem deteksi gerak berdasarkan gerakan tahapan proses pengolahan citra pada sistem deteksi gerak berdasarkan warna pada file citra RGB. adapun pengolahan yang harus dilalui agar dapat mendeteksi objek bergerak adalah sebagai berikut :

1. Citra yang pertama adalah Inputan awal yang berupa video RGB yang didapat dari salah satu *frame* video input sistem deteksi objek yang didapat dengan menggunakan *source code* **gambar 4.9** *source code* untuk menampilkan gambar.

```
clc;clear;close all;

vid = VideoReader('SampleVideo.mp4');
```

**Gambar 4.9 Source Code Menjalankan File Video**

2. Setelah input citra video dapat ditampilkan maka, langkah berikutnya dapat lihat pada **gambar 4.10** yaitu mengekstrak semua *frame* pada video. Perhitungan *frame* video merupakan perkalian antara durasi video dan *frame rate* video dengan

menggunakan *source code* pada **gambar 4.10** untuk mengekstrak setiap *frame* pada video.

```
vidWidth = vid.Width;
vidHeight = vid.Height;
mp4 = struct('cdata',zeros(vidHeight,vidWidth,240,'uint8'),...
    'colormap',[]);
frames = zeros(vidHeight,vidWidth,3,240);

k = 1;
while hasFrame(vid)
    frames(:,:,:,k) = readFrame(vid);
    k = k+1;
end
```

**Gambar 4.10 Source Kode Menampilkan Setiap Frame Pada Video**

3. Mencari *Background Frame* secara otomatis dengan cara menghitung nilai modus pada setiap *frame* menggunakan *source code* **gambar 4.11** untuk mencari *Background Frame* secara otomatis.

```
R = squeeze(frames(:,:,1,:));
G = squeeze(frames(:,:,2,:));
B = squeeze(frames(:,:,3,:));


R_back = uint8(mode(R,3));
G_back = uint8(mode(G,3));
B_back = uint8(mode(B,3));

Background = cat(3,R back,G back,B back);
```

**Gambar 4.11 Mencari Background Frame Secara Otomatis**

sehingga diperoleh *Background Frame* seperti ditunjukkan pada gambar di bawah sebagai berikut:

**Gambar 4.12 Background Dari Video**

4. Langkah berikutnya adalah Melakukan operasi *background subtraction* dengan metode pengurangan citra *grayscale* pada setiap *frame*. Sebelum itu di lakukan, kita harus membaca *frame* dari setiap video yang ada agar dapat di pisahkan antara *background* dan objeknya. *Source code* dapat di lihat pada gambar di bawah:

```
for x = 1:240
    CurrentFrame = uint8(frames(:,:,:,x));
```

**Gambar 4.13 Source Code Membaca Citra Frame**

5. Setelah didapat citra hasil dari setiap *frame* video pada sistem, maka langkah selanjutnya adalah dengan mengkonversi citra *frame* menjadi citra *grayscale* dengan menggunakan fungsi RGB *to grayscale* dengan *source code* pada **gambar 4.14** source code RGB *to grayscale*.

```
% Mengkonversi citra menjadi grayscale
Background_gray = rgb2gray(Background);
CurrentFrame_gray = rgb2gray(CurrentFrame);
```

**Gambar 4.14 Source Code *RGB* ke Grayscale**

Dengan *source code* diatas kita dapat megubah citra video RGB menjadi *grayscale*. Proses pengubahan dari citra RGB ke *grayscale* di bagi menjadi 2 proses pertama mengubah

*background* RGB ke *grayscale*. Dengan perintah *rgb2gray*, dapat di lihat pada gambar sebagai berikut:

**Gambar 4.15 Mengubah Video Background dari RGB ke Grayscale**

Proses pengubahan dari citra RGB ke grayscale yang kedua merupakan perubahan citra RGB yang ada objeknya menjadi citra *grayscale*. Dengan perintah *rgb2gray*. Dapat di lihat pada gambar berikut:

**Gambar 4.16 Mengubah Video Objek dari RGB ke Grayscale**

6. Langkah selanjutnya adalah pengolahan citra dengan melakukan pengurangan antara *CurrentFrame* dengan *BackgroundFrame*. Citra di ubah ke *grayscale*, pengurangan antara *CurrentFrame* dengan *BackgroundFrame* pada **gambar 4.17** *source code* sebagai berikut:

```
Subtraction = (double(Background_gray)-double(CurrentFrame_gray)
Min_S = min(Subtraction(:));
Max_S = max(Subtraction(:));
Subtraction = ((Subtraction-Min_S)/(Max_S-Min_S))*255;
Subtraction = uint8(Subtraction);
```

**Gambar 4.17 Pengurangan Antara *Currentframe* Dengan *Backgroundframe***

7. Langkah berikutnya adalah mengkonversi citra menjadi biner menggunakan metode Otsu dimana pada proses ini sistem akan metransformasi citra dari citra *grayscale* untuk

membentuk citra biner, sebuah citra digital yang hanya memiliki dua kemungkinan warna *pixel*nya hitam dan putih, Jika nilainya berada antara dua nilai *threshold* berisikan *threshold outside* dimana adalah kebalikan dari *threshold inside*. Biasanya *pixel* objek diberi nilai 1 sementara *pixel background* diberi nilai 0. *Source code* **gambar 4.18** dapat di lihat pada gambar di bawah ini:

```
Subtraction = ~im2bw(Subtraction,graythresh(Subtraction));
```

**Gambar 4.18 Mengkonversi Citra Menjadi Biner Menggunakan Metode Otsu**

8. Langkah selanjutnya adalah Operasi Morfologi. Operasi Morfologi merupakan teknik pengolahan citra berdasarkan bentuk segmen citra. Yang bertujuan untuk memperbaiki hasil segmentasi. Teknik morfologi biasanya digunakan pada citra biner atau untuk beberapa kasus juga bisa diterapkan pada citra keabuan (*grayscale*). Untuk source kode dapat di lihat pada **gambar 4.19** di bawah ini:

```
bw = imfill(Subtraction,'holes');
bw = bwareaopen(bw,10);
```

**Gambar 4.19 Source Code Operasi Morfologi**

9. Langkah berikutnya adalah Pembuatan *masking* dan proses *cropping*. Proses *cropping* bertujuan untuk memotong citra yang akan digunakan dan membuang citra yang tidak digunakan. Citra akan dipotong menjadi bentuk persegi panjang dengan fungsi *imrect*. Ukuran serta posisi citra yang akan dipotong akan disesuaikan menurut citra itu sendiri karena ukuran mata pada setiap orang berbeda. Setelah

penyesuaian posisi, maka citra akan di potong dengan fungsi *imcrop*. Adapun proses masking adalah mebuat kotak atau persegi untuk objek yang akan di deteksi. *Source code* dapat di lihat pada **gambar 4.20** berikut:

```
[row,col] = find(bw==1);
h_bw = imcrop(CurrentFrame,[min(col) min(row) max(col)-min(col) max(row)-min(row)]

[a,b] = size(bw);
mask = false(a,b);
mask(min(row):max(row),min(col):max(col)) = 1;
mask =  bwperim(mask,8);
mask = imdilate(mask,strel('square',3));

R = CurrentFrame(:,:,1);
G = CurrentFrame(:,:,2);
B = CurrentFrame(:,:,3);

R(mask) = 255;
G(mask) = 0;
B(mask) = 0;

RGB = cat(3,R,G,B);

mp4(x).cdata = RGB;
```

**Gambar 4.20 Source Code Pembuatan Masking dan Proses Cropping**

10. Langkah 1 sampai 9 adalah tahap–tahap pengolahan citra yang harus dilalui suatu *frame* dalam video input agar dapat mendeteksi objek yang sedang bergerak yang ada dalam *frame* tersebut. Setelah objek bergerak terdeteksi maka objek yang terdeteksi tersebut akan diberi tanda dan diberi koordinat untuk menunjukan posisi objek. Seperti yang terdapat pada **gambar 4.21** *source code regionprops*.

```
figure, imshow(bw);

hf = figure;
set(hf,'position',[320 240 vidWidth vidHeight]);

movie(hf,mp4,1,vid.FrameRate);
close
```

**Gambar 4.21 Source Code Menampilkan Objek yang Bergerak pada Suatu Video**

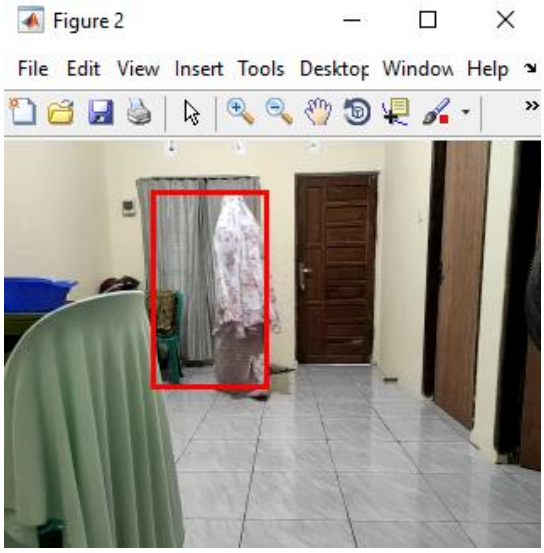
## 4.2 Pengujian Sistem

Proses pengujian suatu program bertujuan untuk memastikan bahwa program yang telah dibuat dapat di implementasikan secara akurat dan tepat, sehingga pengguna dapat menggunakan program ini dengan baik sesuai dengan fungsi program tersebut dibuat.

Sesuai dengan tujuan dari sistem ini yaitu menghasilkan aplikasi yang dapat mendeteksi objek bergerak pada *file* video, maka pengujian sistem deteksi gerak berdasarkan *file* video menggunakan video yang terekam menggunakan kamera *Handphone* dengan aplikasi *Open Camera* dengan pengaturan 16 Frame/S dan durasi video 15 detik.

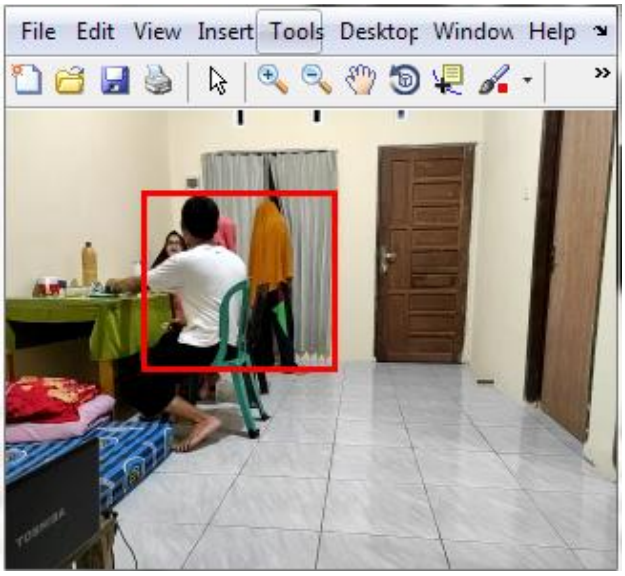
### 4.2.1 Pengujian 1 (Kondisi Terang)

Pengujian pertama dilakukan untuk mendeteksi objek yang dilakukan pada ruangan yang terdapat dilantai 1 gedung yang dilengkapi dengan jendela kaca. Pengujian pertama ini dilakukan pada siang hari sehingga intensitas cahaya pada pada ruang pengujian cukup terang, saat sistem mulai dijalankan objek yang akan dideteksi berada cukup jauh dengan kamera yakni kurang lebih 2 meter, pada pengujian pertama ini dapat menghasilkan kesimpulan bahwa objek dapat diterima dengan baik oleh kamera, dan gambar dapat di proses dengan baik sehingga objek akan dideteksi dapat terdeteksi sesuai dengan harapan yakni munculnya *boundingbox* yang mengikuti gerakan objek yang terdekteksi dan koordinat objek yang menunjukkan perubahan posisi objek selama sistem mendeteksi objek tersebut. Seperti yang terlihat dalam **gambar 4.22** sebagai berikut:



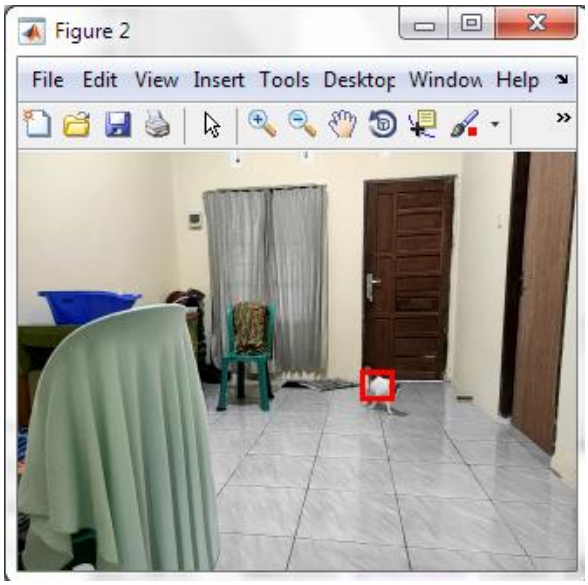

**Gambar 4.22 Hasil Uji Menampilkan Objek Yang Bergerak Pada Suatu Video**

Pengujian pertama yang dilakukan menggunakan beberapa objek orang, pada video pertama ini dapat dilihat bahwa sistem dapat mendekteksi orang yang bergerak. Objek yang bergerak akan di tandai dengan tanda kotak (*mask*). Selengkapnya dapat dilihat pada **gambar 4.23** dibawah ini:




**Gambar 4.23 Hasil Uji Menampilkan Beberapa Objek Yang Bergerak Pada Suatu Video**

Pengujian pertama yang dilakukan menggunakan objek hewan, disini kita menggunakan kucing sebagai objeknya tempat dan ruangannya masih sama. Pada video kedua dapat di lihat bahwa sistem dapat mendekteksi kucing yang bergerak. Objek yang bergerak akan di tandai dengan tanda kotak (*mask*). Selengkapnya dapat dilihat pada **gambar 4.24** dibawah ini:



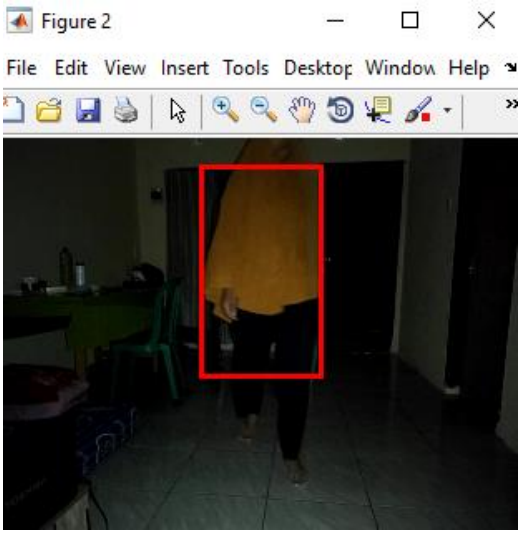

**Gambar 4.24 Hasil Uji Menampilkan Objek Yang Bergerak Pada Suatu Video (Hewan)**

### 4.2.2 Pengujian 2 (Kondisi Gelap)

Pengujian kedua dilakukan dalam mendeteksi objek pada ruangan yang tertutup dan tidak dilengkapi oleh penerangan yang cukup. Saat sistem mulai dijalankan objek yang akan dideteksi berada cukup dekat dengan kamera, namun hasil pengujian pada pengujian pertama ini berbeda dengan hasil pengujian kedua.

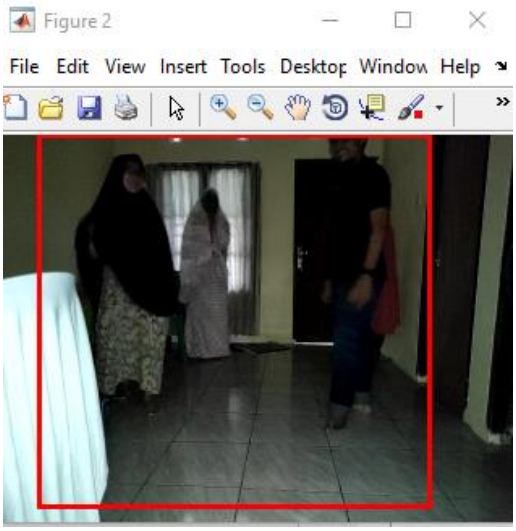
Pada pengujian yang kedua ini objek dapat terdeteksi dengan baik,

seperti yang terlihat pada **gambar 4.25**. Namun saat objek semakin menjauh maka area yang terdeteksi yang ditandai dengan *boundingbox* pun berkurang tidak mencakup seluruh area objek yang terdeteksi.



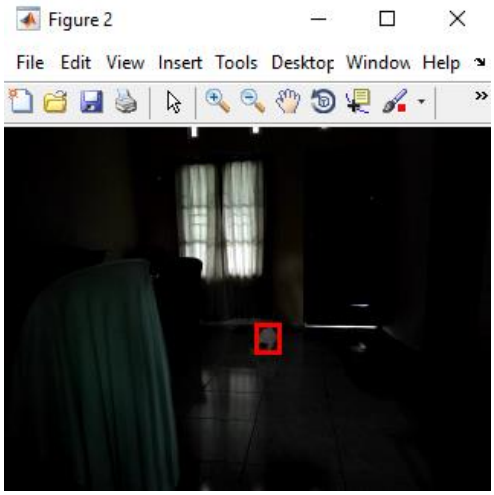

**Gambar 4.25 Hasil Uji Menampilkan Objek Yang Bergerak Pada Suatu Video**

Pengujian kedua yang dilakukan menggunakan beberapa objek orang, pada video kedua dapat di lihat bahwa sistem dapat mendekteksi orang yang bergerak. Objek yang bergerak akan di tandai dengan tanda kotak (*mask*). Selengkapnya dapat dilihat pada **gambar 4.26** dibawah ini:




**Gambar 4.26 Hasil Uji Menampilkan Beberapa Objek Yang Bergerak Pada Suatu Video**

Pengujian kedua yang dilakukan menggunakan hewan, disini kita menggunakan kucing sebagai objeknya tempat dan ruangannya masih sama. Pada video kedua dapat di lihat bahwa sistem dapat mendekteksi kucing yang bergerak. Objek yang bergerak akan di tandai dengan tanda kotak (*mask*). Selengkapnya dapat dilihat pada **gambar 4.27** dibawah ini:




**Gambar 4.27 Hasil Uji Menampilkan Objek Yang Bergerak Pada Suatu Video (Hewan)**

Berdasarkan penjabaran diatas mengenai dua kondisi ruangan yang berisikan 12 video dapat disimpulkan sebagai berikut:

**Tabel 4.3 Hasil Pengujian Background Subtraction**

| No | Nama Video | Kondisi Ruangan | Objek | Terdeteksi/Tidak Terdeteksi |
|---|---|---|---|---|
| 1 | SampleVideo1 | Terang | Orang | Terdeteksi |
| 2 | SampleVideo2 | Terang | Orang | Terdeteksi |
| 3 | SampleVideo3 | Terang | Beberapa Orang | Terdeteksi |
| 4 | SampleVideo4 | Terang | Beberapa Orang | Terdeteksi |
| 5 | SampleVideo5 | Terang | Hewan | Terdeteksi |
| 6 | SampleVideo6 | Terang | Hewan | Tidak Terdeteksi |
| 7 | SampleVideo7 | Gelap | Orang | Terdeteksi |
| 8 | SampleVideo8 | Gelap | Orang | Tidak Terdeteksi |
| 9 | SampleVideo9 | Gelap | Beberapa Orang | Terdeteksi |
| 10 | SampleVideo10 | Gelap | Beberapa Orang | Tidak Terdeteksi |
| 11 | SampleVideo11 | Gelap | Hewan | Terdeteksi |
| 12 | SampleVideo12 | Gelap | Hewan | Tidak Terdeteksi |

Pada tabel diatas menjelaskan bahwa dari 12 data *SampleVideo* terdapat 8 video yang terdeteksi sedangkan 4 video lainnya tidak terdeteksi dikarenakan terdapat beberapa kendala pada video dari bagian noise, pergerakkan objek, dan pengekstrak *background* yang tidak sesuai.

## 4.3 Kesimpulan Pengujian

Pengujian yang telah dilakukan dapat diambil kesimpulan, adalah sebagai berikut:

1. Pengujian pada 12 *SampleVideo* yang berisikan ruangan kosong, objek orang, objek beberapa orang, dan objek hewan menghasilkan 7

video yang terdiri dari 4 video pada kondisi terang sedangkan 3 video dengan kondisi gelap.

2. Pengujian membuktikan keakuratan dalam mendeteksi objek berkisar kurang dari 2 meter dari *camera*, namun saat objek semakin menjauh maka area yang terdeteksi yang ditandai dengan *boundingbox* pun berkurang tidak mencakup seluruh area objek yang terdeteksi.
3. Pengujian sebelumnya yang dilakukan oleh Umam,dkk memiliki video dengan 5 Frame/S berdurasi 10 detik yang memiliki akurasi sebesar 30%, sedangkan pengujian ini memiliki 16 Frame/S dan durasi video 15 detik memiliki akurasi sebesar 66,7%.