TUGAS AKHIR

PROTOTIPE SISTEM PENGUSIR HAMA BURUNG BERBASIS *COMPUTER VISION*

Diajukan Untuk Memenuhi Salah Satu Syarat

Memperoleh Gelar Sarjana Teknik

Program Studi Teknik Elektro

Disusun oleh :

ANTHONIUS ADI NUGROHO

NIM : 145114022

JURUSAN TEKNIK ELEKTRO

FAKULTAS SAINS DAN TEKNOLOGI

UNIVERSITAS SANATA DHARMA

YOGYAKARTA

2018

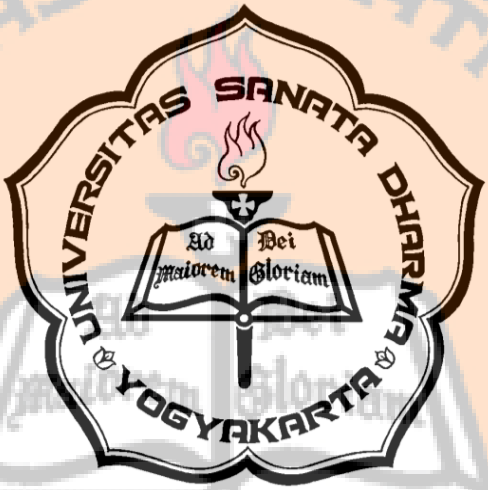
FINAL PROJECT

PROTOTYPE OF BIRD PEST CONTROL SYSTEM BASED ON COMPUTER VISION

In a partial fulfilment of requirements

for the degree of Sarjana Teknik

Department of Electrical Engineering



ANTHONIUS ADI NUGROHO

NIM : 145114022

DEPARTMENT OF ELECTRICAL ENGINEERING

FACULTY OF SCIENCE AND TECHNOLOGY

SANATA DHARMA UNIVERSITY

YOGYAKARTA

2018

HALAMAN PERSETUJUAN
TUGAS AKHIR

PROTOTIPE SISTEM PENGUSIR HAMA BURUNG BERBASIS *COMPUTER VISION*

Disusun oleh :

ANTHONIUS ADI NUGROHO
NIM : 145114022

Telah disetujui oleh:

Pembimbing

Dr.Linggo Sumarno

Tanggal : 4-09-18

HALAMAN PENGESAHAN

TUGAS AKHIR

PROTOTIPE SISTEM PENGUSIR HAMA BURUNG BERBASIS *COMPUTER VISION*

Disusun oleh :

ANTHONIUS ADI NUGROHO

NIM : 145114022

Telah dipertahankan di depan tim penguji

pada tanggal 27 Juli 2018

dan dinyatakan memenuhi syarat

Susunan Tim Penguji:

Nama lengkap	**Tanda Tangan**
Ketua : Ir. Th. Prima Ari Setyani, M.T.	
Sekretaris : Dr. Linggo Sumarno	
Anggota : Petrus Setyo Prabowo, S.T.,M.T.	

Yogyakarta 7 September 2018

Fakultas Sains dan Teknologi

Universitas Sanata Dharma

Dekan,

Sudi Mungkasi, S.Si., M.Math.Sc., Ph.D

PERNYATAAN KEASLIAN KARYA

Saya menyatakan dengan sesungguhnya bahwa tugas akhir ini tidak memuat karya atau bagian orang lain, kecuali yang telah disebutkan dalam kutipan dan daftar pustaka sebagaimana layaknya karya ilmiah.

Yogyakarta 27 Agustus 2018

Anthonius Adi Nugroho

HALAMAN PERSEMBAHAN DAN MOTTO HIDUP

MOTTO:

"Pengetahuan yang baik adalah pengetahuan

yang menyisakan pertanyaan"

Skripsi Ini Saya Persembahkan Untuk...

Sang Pencipta Allah Tri Tunggal Maha Kudus

Keluarga Dan Kerabat Terdekat

Seluruh Petani Indonesia

Teman Sejawat

LEMBAR PERNYATAAN PERSETUJUAN
PUBLIKASI KARYA ILMIAH UNTUK KEPENTINGAN AKADEMIS

Yang bertanda tangan di bawah ini, saya mahasiswa Universitas Sanata Dharma :

Nama : ANTHONIUS ADI NUGROHO

Nomor Mahasiswa : 145114022

Demi pengembangan ilmu pengetahuan , saya memberikan kepada perpustakaan Universitas Sanata Dharma karya ilmiah saya yang berjudul:

PROTOTIPE SISTEM PENGUSIR HAMA BURUNG BERBASIS *COMPUTER VISION*

Beserta perangkat yang diperlukan (bila ada). Dengan demikian saya memberikan kepada Perpustakaan Universitas Sanata Dharma hak untuk menyimpan, mengalihkan dalam bentuk media lain, mengelolanya dalam bentuk pangkalan data, mendistribusikan secara terbata, dan mempublikasikannya di Internet atau media lain untuk kepentingan akademis tanpa perlu meminta ijin dari saya maupun memberikan royalty kepada saya selama tetap mencantumkan nama sebagai penulis.

Demikian pernyataan ini yang saya buat dengan sebenarnya.

Yogyakarta, 27 agustus 2018

ANTHONIUS ADI NUGROHO

INTISARI

Computer vision merupakan rekayasa teknologi komputer yang dapat dimanfaatkan untuk membantu pekerjaan manusia. Sebuah computer yang dilengkapi kamera dapat diaplikasikan seperti mata manusia. Dengan algoritma tertentu, sebuah computer tersebut dapat mendeteksi objek, mengenali objek, memetakan ruangan dan sebagainya. Dalam segala aspek kehidupan secara tidak langsung kita menggunakan teknologi computer vision ini, diantaranya digunakan dalam bidang industri, robotika, automasi, keamanan, navigasi maupun dalam bidang medis.

Pemanfaatan computer vision pada penelitian ini sebagai pengusir hama burung menggunakan perangkat Raspberry Pi sebagai mini komputer. Raspbery pi akan dilengkapi dengan webcam untuk mendeteksi burung dengan metode *background subtraction*. Webcam akan menangkap setiap frame untuk diolah pada Raspberry Pi. Dengan *background subtraction*, setiap frame akan kurangi dengan frame sebelumnya. Perbedaan nilai yang terdapat pada setiap pixel akan terlihat dan di lakukan pengambangan. Proses pengolahan video dibantu dengan *library* python yaitu OpenCV yang telah terinstal pada operating sistem. Hasil dari proses pengolahan video akan menggerakan aktuator pada lahan sawah. Fungsi aktuator ini untuk mengusir burung yang tertangkap webcam.

Hasil penelitian menunjukan selama dilakukan uji coba 18 kali terdapat 2 kali *error* dan 16 kali keberhasilan. *Error* yang terjadi disebabkan oleh faktor dari luar yaitu angin yang menggerakan padi lahan sawah. Kehadiran sistem pengusir hama burung ini diharapkan dapat membantu petani dalam menghalau burung dari padi di lahan sawah.

Kata kunci : *Computer Vision*, Raspberry pi, *Background Subtraction*, Pengolahan Citra

ABSTRACT

Computer vision is an engineering computer technology that can be utilized to help human work. A computer equipped with a camera can be applied like a human eye. With certain algorithms, a computer can do such as object detection, object recognizing, room mapping and so on. In all aspects of life, we indirectly use this computer vision technology, including in the fields of industry, robotics, automation, security, navigation even in the medical field.

Computer vision utilized in this final project as a pest repellent using raspberry pi devices as mini computers. Raspbery pi will be equipped with a webcam to detect birds by background subtraction method. The webcam will capture each frame to be processed on raspberry pi. With background subtraction, each frame will subtract with the previous frame. Differences in the value contained in each pixel will be visible and continued on thresholding. The results of video processing will be assisted with python library that is OpenCV which has been installed on operating system. The results of video processing will move the actuator on the rice field. The actuator function is to repel the birds that caught on webcam.

The results showed during the trial 18 times there are 2 times error and 16 times success. Error that occurs caused by nature factors of the wind that moves paddy on field. The presence of this bird pest control system is expected to help farmers in driving away birds from rice fields

.

Keywords : Computer Vision, Raspberry pi, Background Subtraction, Image Processing

KATA PENGANTAR

Puji syukur dan terimakasih kepada Tuhan Yang Maha Esa atas segala yang telah diberikan melalui dinamika, pengalaman serta perjumpaan dengan setiap orang sehingga tugas akhir ini dapat diselesaikan dengan baik.

Dalam penulisan tugas akhir ini penulis menyadari banyak bimbingan dan dukungan yang didapatkan dari berbagai pihak. Oleh karena itu, pada kesempatan ini penulis mengucapkan banyak terimakasih kepada:

1. Bapak Dr. Linggo Sumarno, selaku Dosen Pembimbing tugas akhir yang telah membimbing penulis dalam setiap permasalahan yang dihadapi.
2. Bapak Petrus Setyo Prabowo, S.T,. M.T. dan Ibu Ir.Theresia Prima Ari Setyo, S.T,. M.T. selaku Dosen Penguji yang telah memberikan kritik dan saran terhadap penulis dalam mengerjakan tugas akhir.
3. Seluruh dosen Teknik Elektro Universitas Sanata Dharma yang memberikan ilmu pengetahuan kepada penulis.
4. Bapak Sis Bakir yang telah memperbolehkan peneliti melakukan pengujian di lahan sawah.
5. Setiap anggota keluarga yang selalu memberikan dukungan dalam hal materi maupun diluar materi.
6. Kerabat dekat yang berperan memberikan dukungan serta saran yang membantu penulis menyelesaikan tugas akhir.
7. Teman-teman yang telah membantu dalam melancarkan pengerjaan tugas akhir.
8. Teman-teman grup yang telah menghibur penulis dan bertukar pikiran.
9. Seluruh teman-teman Teknik Elektro 2014 Universitas Sanata Dharma yang telah berproses dan berdinamika bersama untuk menjadi lebih baik.
10. Teman teman kos “bu cicih” yang telah membantu dalam pengujian tugas akhir.
11. Serta semua pihak yang tidak bisa disebutkan satu per satu atas pengalaman yang diberikan kepada penulis untuk menjadi lebih baik

Dalam penyusunan tugas akhir ini penulis masih memiliki kekurangan dalam berbagai hal. Oleh karena itu, kritik dan saran dari semua pihak sangat diharapkan. Harapan dari penelitian ini dapat dikembangkan untuk peneliti selanjutnya.

Yogyakarta, 27 Agustus 2018

ANTHONIUS ADI NUGROHO

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I

PENDAHULUAN

1.1. Latar Belakang

Computer Vision mempunyai peranan penting terhadap perkembangan teknologi. Sampai saat ini masih terus dikembangkan, seiring dengan perkembangan teknologi *computer vision* difungsikan seperti mata manusia. Mata manusia adalah sebuah sistem canggih yang melakukan respon atas rangsangan visual. Secara fungsional, *computer vision* dan penglihatan manusia adalah sama, dengan tujuan menafsirkan data spasial yaitu data yang diindeks oleh lebih dari satu dimensi [1]. *Computer vision* dapat didefinisikan dengan pengertian pengolahan citra yang dikaitkan dengan akuisisi citra, pemrosesan, klasifikasi, penganan, dan pencakupan keseluruhan, pengambilan keputusan yang diikuti pengidentifikasian citra [2]. Tanpa disadari, *computer vision* telah digunakan dalam kehidupan sehari hari, sehingga masyarakat lebih terbantu hampir dalam segala bidang.

Dalam bidang pertanian kebutuhan teknologi dapat diterapkan dalam membantu kegiatan petani dalam bercocok tanam yang masih menggunakan cara konvensional. Sampai saat ini petani masih banyak yang belum menerapkan teknologi pada proses bercocok tanam, seperti pengusiran hama burung yang ada di sawah. Petani khususnya di Indonesia masih menggunakan alat yang terbilang konvensional. Peralatan konvensional tersebut seperti orang orangan sawah, tali yang dibentangkan di sawah dan masih banyak lagi. Akibatnya dalam pengusiran hama burung di sawah menjadi tidak maksimal dan berdampak pada hasil panen petani.

Berdasarkan permasalahan yang ada dalam bidang pertanian, penelitian ini bertujuan untuk membantu menyelesaikan salah satu permasalahan tersebut yaitu dengan merancang sebuah sistem pengusir hama burung berbasis *computer vision*. Dengan mengembangkan teknologi *computer vision* ini maka dalam pengusiran hama burung dapat dilakukan secara real time sehingga diharapkan membantu petani dalam mengusir hama burung disawah. Telah dilakukan penelitian tentang deteksi burung pada keadaan realtime. Pada penelitian sebelumnya untuk mendeteksi burung tersebut menggunakan metode *background subtraction*. Suatu deskripsi pendekatan yang telah ada tentang *background subtraction* adalah mendeteksi objek-objek *foreground* sebagai perbedaan yang ada antara frame sekarang dan gambar *background*

dari layar *static*[3]. Objek yang terdeteksi sudah menjadi citra biner sehingga untuk membedakan objek dan *background* dapat dilihat dari warnanya, *background* berwarna hitam dan objek burung berwarna putih. Namun metode *background subtraction* ini memiliki kelemahan terhadap benda apapun yang bergerak, karena akan terdeteksi sebagai objek atau *foreground*.

Pada penelitian awal ini menggunakan metode *binarization* untuk mendeteksi burung lalu diterapkan pada lahan sawah siap panen untuk pengusir hama burung. Pada lahan sawah siap panen warna padi sudah mulai menguning dan warna hama burung yang datang cenderung lebih gelap. Dengan *binarization* lebih mudah untuk memisahkan antara objek yang akan di deteksi dengan *background*.

1.2. Tujuan dan Manfaat Penelitian

Tujuan dari penelitian ini adalah:

1. Membuat prototipe sistem pengusir hama burung pada lahan sawah siap panen dengan perangkat *embedded system* berbasis *computer vision.*

Manfaat dari penelitian ini adalah:

1. Untuk Masyarakat:

   Penelitian awal ini diharapkan membantu masyarakat khususnya petani dalam menghalau hama burung di lahan sawah siap panen, sehingga petani tidak perlu standby di sekitar sawah untuk menghalau burung.

1.3. Batasan Masalah

Penelitian ini berfokus pada sistem pengusiran hama burung pada lahan sawah, sistem ini memiliki batasan masalah diantaranya:

a. Hanya mendeteksi burung di area sawah tanpa mengklasifikasi jenis burung.
b. Luasan sawah dalam pengujian dengan minimal cakupan area 4m x 4m. Penelitian dibatasi dengan ukuran sawah untuk pengujian karena keterbatasan webcam saat menangkap gambar pada jarak yang semakin jauh akan mengakibatkan resolusi citra biner semakin kecil saat pengujian berlangsung.
c. Pengujian dilakukan pada siang hari atau pada saat cuaca cerah.
d. Jika ada warna gelap lain misalnya tanah di sawah maka dapat ditutupi dengan warna terang.

e. Sistem dinyatakan berhasil apabila jumlah burung ≤ 10% dari jumlah keseluruhan pada lahan sawah yang tertangkap kamera, dengan minimal jumlah burung ≥ 6 pada lahan sawah.

f. Pengujian sistem *non realtime* dilakukan dengan mengambil tiga sampel umur padi yang berbeda pada lokasi sawah.

g. Pada saat pengujian *non realtime*, akan ditampilkan pada monitor berupa video hasil proses citra dan jumlah burung.

h. Perhitungan burung dilakukan secara manual dalam menentukan tingkat keberhasilan sistem.

1.4. Metodologi Penelitian

Berdasarkan pada tujuan yang ingin dicapai, metode-metode yang digunakan dalam penyusunan tugas akhir ini adalah:

a. Studi Literatur

Pengumpulan informasi dari berbagai sumber antara lain buku, jurnal, *datasheet* dan artikel di internet. Informasi yang dikumpulkan berkaitan dengan penelitian tugas akhir ini.

b. Survey Lahan

Survey lahan dilakukan untuk pemilihan lahan saat pengujian tugas akhir supaya kinerja sistem maksismal.

c. Perancangan *Hardware* dan *Software*

Tahap ini bertujuan untuk merancang desain aktuator pengusir hama burung dengan rangkaian pendukung dan pembuatan model sistem untuk program *python* di Raspberry Pi3.

d. Pembuatan *Hardware* dan *Software*

Tahap ini nerupakan pembuatan program dan aktuator untuk pengusir hama burung sesuai dengan hasil akhir perancangan *Hardware* dan perancangan *software*.

e. Pengambilan Data

Sebelum sistem diuji dilakukan pengambilan data berupa video untuk pengujian alat dan sebagai bahan analisis untuk dilakukan pengujian secara *realtime*. Maka perlu dilakukan pengambilan data secara *non realtime*. Pengambilan data *non realtime* sangat diperlukan untuk menguji coba program yang sudah dirancang. Pengambilan data *non realtime* ini dengan cara mengambil video yang akan di

capture oleh Webcam dan disimpan dalam *Micro* SD. Setelah pengujian program *non realtime* sudah dilakukan maka selanjutnya dilakukan pengujian keseluruhan sistem secara *realtime*.

f. Analisis dan Penyimpulan Hasil

Analisis dan penarikan kesimpulan diperoleh dari data hasil pengujian yang dilakukan secara *realtime* di lahan sawah untuk melihat tingkat keberhasilan sistem, maka dengan indikator jumlah burung ≤ 10% menjadi acuan keberhasilan sistem, dengan jumlah burung minimal ≥ 6 pada lahan sawah, sehingga nilai presentase yang tidak bulat akan dibuatkan. Dengan kata lain sistem dinyatakan tidak berhasil apabila aktuator tidak bergerak jika jumlah burung ≥ 6. Hasil pengujian tersebut dituliskan pada tabel pengujian *realtime*.

BAB II
DASAR TEORI

2.1. Hama burung

Hama adalah suatu penyebab kerusakan pada tanaman yang dapat dilihat dengan pancaindera (mata). Hama tersebut dapat berupa binatang. Hama dapat merusak tanaman secara langsung maupun tak langsung. Hama yang merusak tanaman secara langsung dapat dilihat bekasnya pada tanaman yang diserang, misalnya gerekan dan gigitan. Sedangkan hama yang merusak tanaman secara tidak langsung biasanya melalui penyakit[4].

Banyak sekali hama burung merusak tanaman padi pada lahan sawah siap panen. Salah satu hama padi adalah burung, sekelompok burung kecil pemakan biji bijian banyak menyebar di wilayah tropis. Karena biasanya burung akan mulai makan padi ketika padi berbuah sampai panen. Hama burung selalu berkelompok ketika terbang dan makan padi, sehingga jika jumlahnya banyak tentu bulir padi yang mereka makan akan banyak juga. Mereka tidak hanya satu hari saja, melainkan setiap hari[4]. Sehingga dapat mengakibatkan penurunan hasil produksi panen petani.

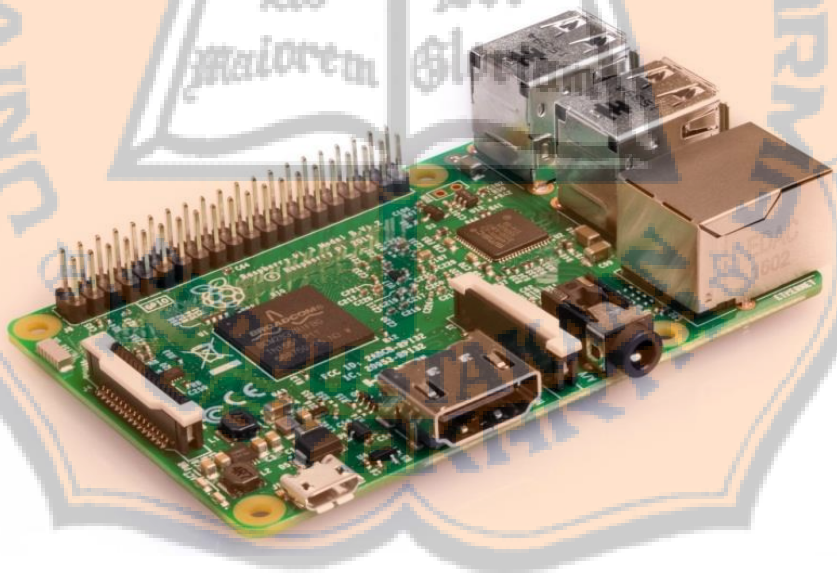
Gambar 2.1. Hama burung pada lahan sawah[4].

Pengendalian hama padi sawah untuk memperoleh produksi yang lebih meningkat terus dilakukan. Baik itu secara kimia maupun mekanik. Salah satu hama padi adalah burung yang biasa makan bulir padi. Petani biasa membuat alat untuk mengendalikan hama burung ini dengan beberapa untaian tali yang diberikan benda-benda yang dapat menimbulkan suara ketika tali ditarik. Hal itu akan menyebabkan burung kaget dan terbang meninggalkan padi, sehingga batal untuk makan[4].

2.2. Raspberry Pi 3

Raspberry Pi adalah komputer berukuran kartu kredit yang dihubungkan ke TV dan keyboard Anda. Ini adalah komputer kecil yang mumpuni yang bisa digunakan dalam proyek elektronik, dan untuk banyak hal yang dilakukan PC desktop Anda, seperti spreadsheet, pengolah kata, browsing internet, dan permainan. Ini juga memainkan video berdefinisi tinggi[5].

Raspberry Pi harus di *install* OS terlebih dahulu supaya dapat dioperasikan, salah satu OS yang lebih dikenal adalah raspbian. Raspbian merupakan sistem operasi gratis yang berbasis pada debian dan dioptimasi untuk perangkat keras Raspberry Pi[6]. OS tersebut di *install* dalam micro SD minimal 4GB, namun untuk sistem yang besar dapat menggunakan *microSD* diatas 8GB. Raspberry Pi3 ditenagai dengan tegangan 5 Volt dan arus rata rata 2.1 Ampere tergantung dari tambahan perangkat yang digunakan. *Input power supply* menggunakan *micro* USB. Fitur dari Raspberry Pi3 ini juga memiliki port USB yang digunakan untuk perangkat tambahan seperti webcam, selain itu memiliki 40 pin GPIO atau pin input dan output. Bentuk fisik dari Raspberry Pi3 dapat dilihat pada gambar 2.2



Gambar 2.2. Bentuk fisik Raspberry Pi 3[7].

Raspberry Pi3 memiliki spesifikasi sebagai berikut[7]:

1. *Quad Core 1.2GHz Broadcom BCM2837 64bit CPU.*
2. *1GB RAM.*
3. *BCM43438 wireless LAN and Bluetooth Low Energy (BLE) on board.*
4. *40-pin extended GPIO.*

5. *4 USB 2 ports.*
6. *4 Pole stereo output and composite video port.*
7. *Full size HDMI.*
8. *CSI camera port for connecting a Raspberry Pi camera.*
9. *DSI display port for connecting a Raspberry Pi touchscreen display.*
10. *Micro SD port for loading your operating system and storing data.*
11. *Upgraded switched Micro USB power source up to 2.5A.*

2.3. Webcam

Web camera (Webcam) adalah kamera video yang menyediakan aliran gambar secara real-time yang dihubungkan ke software pada komputer melalui port USB atau serial. Ketika diproses oleh komputer, aliran video dapat disimpan, dilihat atau dikirim ke perangkat lain melalui koneksi internet, *bluetooth*, port USB dan e-mail sebagai lampiran. Pada umumnya webcam terdiri dari sebuah lensa standar, cover dan kabel support. Spesifikasi dan gambar webcam Logitech C270 pada Tabel 2.1 dan gambar 2.3.

Tabel 2.1. Spesifikasi Logitech C270[8].

No	*High-Definition* (HD) *video calling*	HD 720 *pixels*
1	*Video capture*	*Up to* 1280 x 720 *pixels*
2	*Photo quality*	*Up to* 3.0 *Megapixels*
3	*Computer Interface*	USB 2.0 (*recommended*)
4	*Focus*	*Fixed Focus*
5	*Microphone*	*Yes*
6	*Hardware Support*	*Laptop, Monitor LCD or CRT*

Gambar 2.3. Logitech C270[8].

2.4. Python

Bahasa pemrograman *Python* adalah salah satu contoh dari beberapa *high-level-language*, contoh lainnya adalah C++, PHP dan masih banyak lainnya. *Python* banyak digunakan dalam pengembangan aplikasi perkantoran, aplikasi web dan lainna. *Python* juga dapat digunakan dalam pemrogrraman yang memerlukan dinamisme yang cukup tinggi, aplikasi dalam skala yang besar yang berorientasi pada objek [9]. *Python* juga mempunyai beberapa pustaka di dalamnya yang berguna untuk membantu proses pemrograman seperti NumPy dan SciPy.

1. *NumPy*: *NumPy* atau *Numerical Python* adalah salah satu pustaka pada *Python* yang menyediakan fungsi komputasi numerik seperti array dan fungsi fungsinya, sehingga sangat dibutuhkan pada pengolahan citra.
2. *SciPy*: *SciPy* atau *Science Python* adalah perpustakaan komputasi ilmiah yang terkait erat dengan *NumPy*. Pustaka ini berguna untuk memanipulasi data di dalam baik digunakan pada pengolahan citra maupun diluar itu. [10].

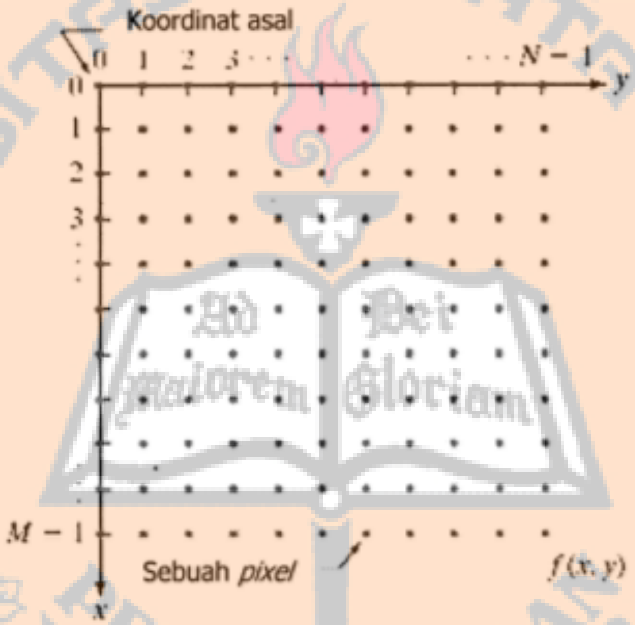
2.5. OpenCV

OpenCV (*Open Source Computer Vision*) adalah sebuah pustaka yang ditujukan untuk pengolahan citra. OpenCV dapat diprogram dengan bahasa C++, C, *Python* dan *java*. *Computer vision* sediri adalah cabang dari bidang ilmu pengolahan citra. Teknologi *Computer Vision* dapat membantu sebuah kamera yang dihubungkan pada komputer layaknya seperti mata manusia. Komputer dapat mengambil keputusan, mengenali suatu objek seperti mata manusia. Beberapa teknologi computer vision telah diterapkan pada kehidupan seperti face detection, face recognition dan object tracking[11]. Dalam OpenCV terdapat berbagai pustaka yang mendukung proses pengolahan citra.

2.6. Pengolahan Citra Digital

Pengolahan citra digital merupakan proses pengolahan setiap data 2 dimensi menggunakan komputer, sesuai dengan jumlah data dan jenis pengolahan. Citra digital merupakan sebuah *array* yang berisi nilai-nilai nyata maupun komplek yang direpresentasikan dengan deretan bit tertentu.

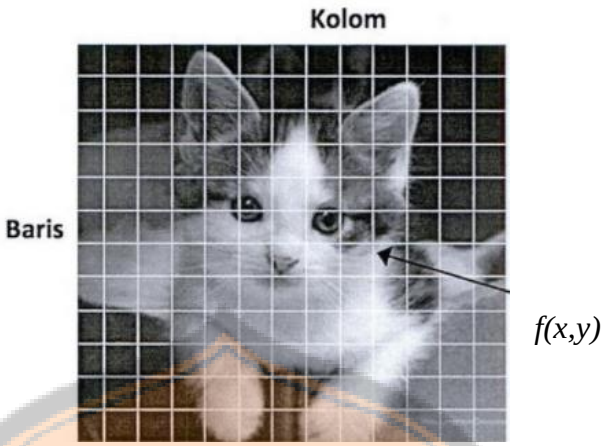
Citra dapat didefinisikan sebagai fungsi *f(x,y)* berukuran *M* baris dan *N* kolom, dengan *x* dan *y* adalah koordinat spasial, dan amplitudo *f* di titik koordinat *(x,y)* dinamakan tingkat keabuan dari suatu citra. Apabila nilai *x, y* dan *f* secara keselurahan berhingga dan bernilai diskrit maka citra tersebut adalah citra digital[12]. Citra digital dalam bentuk matrik dapat dilihat pada persamaan 2.1 dan posisi koordinat citra digital dapat dilihat pada Gambar 2.4.



Gambar 2.4. Koordinat Citra Digital [12].

$$f(x,y) = \begin{bmatrix} f(0,0) & f(0,1) & \cdots & f(0, N-1) \\ f(1,0) & f(1,1) & \cdots & f(1, N-1) \\ \vdots & \vdots & & \vdots \\ f(M-1,0) & f(M-1,1) & \cdots & f(M-1, N-1) \end{bmatrix} \quad (2.1)$$

Nilai pada posisi *(x,y)* sering disebut dengan piksel pada citra digital. Ilustrasi digitalisasi citra dengan *M* = 16 baris dan *N* = 16 kolom ditunjukan pada Gambar 2.5.



Gambar 2.5. Ilustrasi Citra (piksel koordinat x = 10, y = 7 memiliki nilai 110) [12].

2.6.1. Citra RGB

Citra RGB adalah sebuah citra yang mengandung informasi warna dan disimpan dalam array berukuran m x n x 3 yang mendefinisikan warna merah, hijau dan biru pada setiap piksel. Satu titik piksel merepresentasikan suatu warna yang terdapat 3 layer, sehingga warna pada sebuah piksel ditentukan dari kombinasi merah, hijau dan biru. Array pada setiap piksel dapat bertipe double, uint8 atau uint16 [13]. Gambar 2.6 menunjukan contoh citra RGB.

Gambar 2.6. Citra RGB [4].

2.6.2. Citra *Grayscale*

Citra *grayscale* adalah sebuah citra yang memiliki yang memiliki derajat keabuan. Warna yang dimiliki adalah warna hitam, keabuan dan putih. Tingkatan keabuan ini yang memiliki intensitas dari hitam hingga putih[12]. Citra *grayscale* ini memiliki kombinasi warna dalam bit. Sebagai contoh setiap piksel dari citra 8 bit jumlah maksimum nilai keabuan adalah 256. Contoh gambar citra *grayscale* dapat dilihat pada gambar 2.7.

Gambar 2.7. Citra *Grayscale*[14].

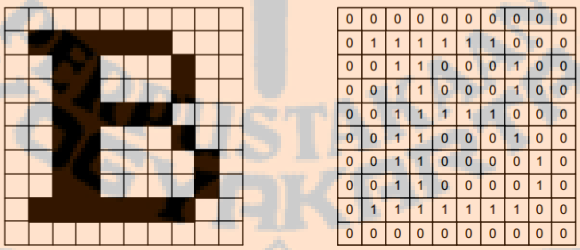
Secara umum rumus yang dipakai untuk mengubah citra RGB ke citra berskala keabuan dapat digunakan persamaan (2.2) [12] :

$$I = 0{,}2989xR + 0{,}5870xG + 0{,}1141B \qquad (2.2)$$

Dengan nilai R , G , B adalah 255, variabel I adalah nilai intensitas sebuah piksel tersebut.

2.6.3. Citra Biner

Citra biner adalah sebuah citra yang mempunyai dua nilai yaitu 0 atau 1 yang direpresentasikan putih dengan nilai 0 dan hitam bernilai 1. Beberapa proses pemrosesan citra mengacu pada citra biner. Pada sebuah citra memiliki suatu objek yang melekat pada fitur objek. Fitur bentuk merupakan suatu fitur yang diperoleh melalui bentuk objek dan dapat dinyatakan melalui kontur, area, transformasi dan lainnya. Fitur bentuk biasa digunakan untuk kepentingan identifikasi objek[15]. Representasi citra biner digambarkan pada gambar 2.8.



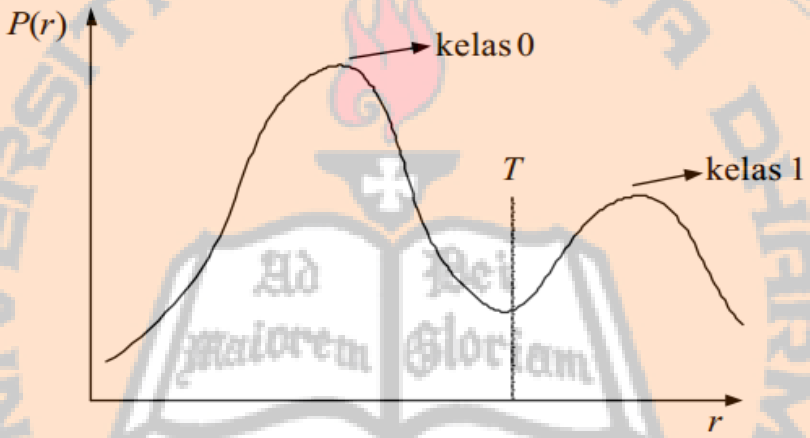
Gambar 2.8. Representasi Citra Biner [16].

Konversi dari citra keabuan ke citra biner dilakukan dengan operasi pengambangan (*thresholding*). Operasi pengambangan mengelompokkan nilai derajat keabuan setiap pixel ke dalam 2 kelas, hitam dan putih. Dua pendekatan yang digunakan dalam operasi pengambangan adalah pengambangan secara global dan pengambangan secara lokal[16].

Pada pengambangan global setiap pixel di dalam citra dipetakan ke dua nilai, 1 atau 0 dengan fungsi pengambangan persamaan (2.3):

$$f(i,j) = \begin{cases} 0, f(i,j) \leq T \\ 1, lainnya \end{cases} \quad (2.3)$$

yang dalam hal ini, fg(i, j) adalah citra hitam-putih, fb(i, j) adalah citra biner, dan T adalah nilai ambang yang dispesifikasikan. Dengan operasi pengambangan tersebut, objek dibuat berwarna gelap (1 atau hitam) sedangkan latar belakang berwarna terang (0 atau putih). Nilai ambang T dipilih sedemikian sehingga galat yang diperoleh sekecil mungkin. Cara yang umum menentukan nilai T adalah dengan membuat *histogram* citra. Jika citra mengandung satu buah objek dan latar belakang mempunyai nilai intensitas yang homogen, maka citra tersebut umumnya mempunyai *histogram* bimodal [16]. Bimodal adalah penggambaran pada sebuah *histogram* yang memiliki 2 puncak untuk merepresentasikan *foreground* dan *background* pada sebuah citra.

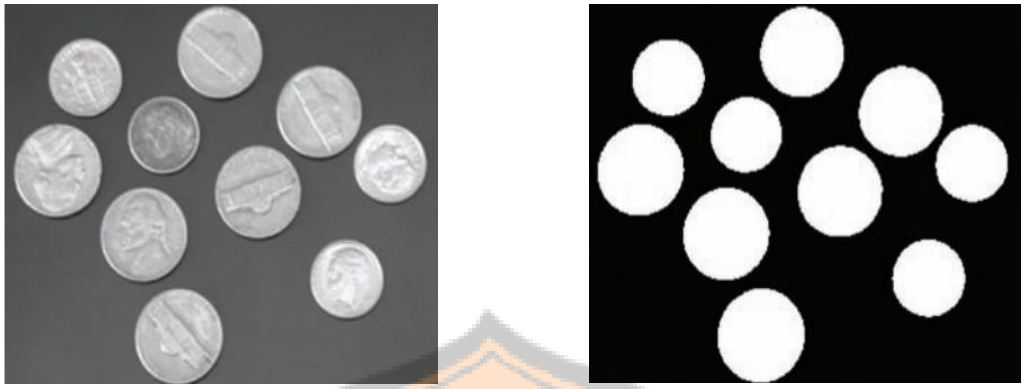


Gambar 2.9. Penentuan nilai ambang T [16].

Gambar 2.9 menunjukan penentuan nilai T pada *histogram* citra. Dengan cara seperti ini, kita tidak hanya mengkonversi citra hitam-putih ke citra biner, tetapi sekaligus melakukan segmentasi objek dari latar belakangnya[16].

2.6.4. Segmentasi

Segmentasi adalah satu dari banyak tugas yang paling penting dari pengolahan citra. Segmentasi citra adalah sebuah operasi yang menandai transisi antara *background* dan *foreground.* Sebagian besar algoritma segmentasi didasarkan pada salah satu dari dua nilai dasar yang di ekstraksi dari nilai piksel citra. Hasil segmentasi citra ditunjukan pada gambar 2.10.



(a) (b)

Gambar 2.10. (a) Citra grayscale sebelum segmentasi (b) Sesudah segmentasi[17].

2.6.4.1. Otsu *Threshold*

Thresholding adalah proses binerisasi dari sebuah citra. Secara umum, konversi citra *grayscale* menjadi citra biner, di mana nilai pixsel *grayscale* diantara 0 sampai 255. Contoh sederhana seperti memilih nilai piksel(p), dan mengatur semua nilai intensitas piksel di bawah p menjadi 0 dan semua nilai intensitas diatas p menjadi 255. Dengan cara ini dapat membuat representasi biner pada sebuah citra dapat dibuat dengan menggunakan *thresholding*. Secara umum, *thresholding* digunakan untuk mendapatkan objek tertentu pada area citra[14]. Sehingga objek tersebut tersegmentasi dari *background*.

Dalam penggunaan *threshold* dibagi menjadi *threshold* global dan lokal. Penelitian ini menggunakan *thresholding* global untuk segmentasi citra. Salah satu global *thresholding* diimplementasikan pada metode *otsu*. Otsu merupakan salah satu jenis *threshold adaptive* yang dapat menyesuaikan nilai *threshold* untuk mencari nilai ambang maksimal terhadap perubahan pencahayaan yang berubah-ubah.

Metode otsu terdiri dari beberapa tahapan. Tahapan pertama, probabilitas nilai intensitas *i* dihitung melalui persamaan (2.4)[18] :

$$p_i = n_i/N \tag{2.4}$$

Keterangan :

$\boldsymbol{n_i}$ = jumlah piksel dengan intensitas *i*.

$\boldsymbol{N}$ = jumlah semua piksel dalam citra (P x L)

Untuk menghitung nilai *zeroth-cumulative moment, first- order cumulative moment* dan nilai rerata dari gambar original dapat dilihat secara berturut pada persamaan (2.5) [18]:

$$\omega(k) = \sum_{i=1}^{k} p_i \tag{2.5}$$

$$\mu(k) = \sum_{i=1}^{k} i\, p_i$$

$$\mu_T = \mu(L) = \sum_{i=1}^{L} i.\, p_i$$

Keterangan :

$\omega(k)$ = *zeroth-cumulative moment*

$\mu(k)$ = *first- order cumulative moment*

$\mu(L)$ = nilai rerata

Untuk mendapatkan nilai *threshold* atau nilai ambang *k* dapat dihitung menggunakan persamaan (2.6) [18]:

$$\sigma_B^2(k) = \max \sigma_B^2(k) \tag{2.6}$$

dengan

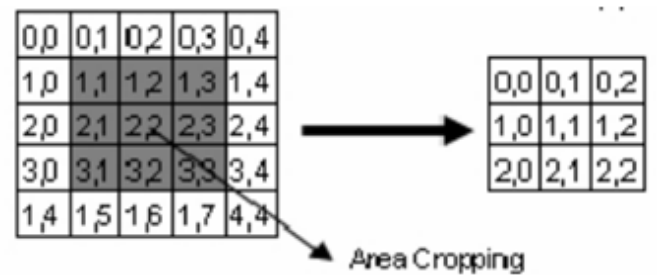
$$\sigma_B^2(k) = \frac{[\mu_T \omega(k) - \mu(k)]^2}{\omega(k)[1-\omega(k)]}$$

Keterangan :

σ_B^2 = nilai varian antar kelas maksimum

2.6.5. *Cropping*

Cropping adalah proses pemotongan citra pada koordinat tertentu pada area citra. Pemotongan bagian dari citra digunakan dua koordinat, yaitu koordinat awal yang merupakan titk awalrdinat bagi citra hasil pemotongan dan koordinat akhir yang merupakan titik koordinat akhir dari citra hasil pemotongan. Sehingga akan membentuk bangun segi empat yang mana tiap-tiap piksel yang ada pada area koordinat tertentu akan disimpan dalam citra yang baru[19]. Citra yang terbentuk itulah yang nantinya akan diproses lebih lanjut, proses *cropping* ditunjukan pada gambar 2.11.

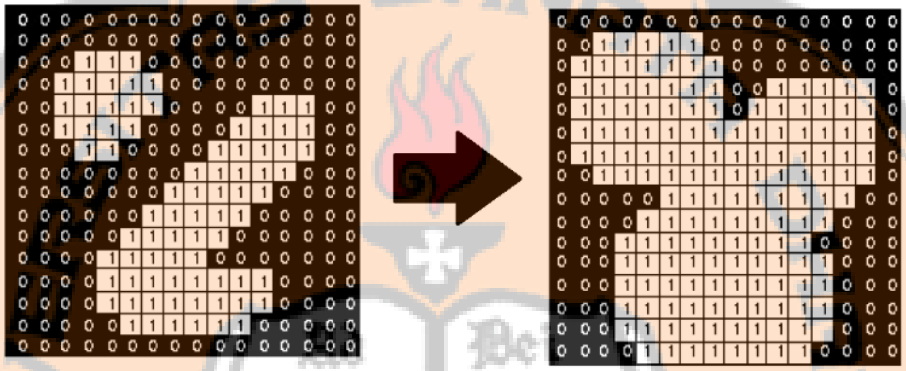


Gambar 2.11. Hasil *cropping* pada citra[19].

Dari gambar tersebut diatas dijelaskan bahwa terjadi proses pemotongan citra. Pada awalnya ukuran pixel dari citra asli adalah 5×5 piksel, setelah dilakukan proses pemotongan pada koordinat awal (1,1) dan koordinat akhir (3,3) atau dengan lebar 3 piksel dan tinggi 3 piksel akan terbentuk citra baru dengan ukuran 3×3 piksel. Citra baru ini berisi nilai pixel dari koordinat (1,1) sampai koordinat (3,3)[19].

2.6.6. Dilasi

Operasi dilasi digunakan untuk memperbesar sebuah citra dengan mendapatkan efek pelebaran struktur pada sekitar objek bernilai 1 [15]. Operasi morfologi dilasi ditunjukan pada gambar 2.12.

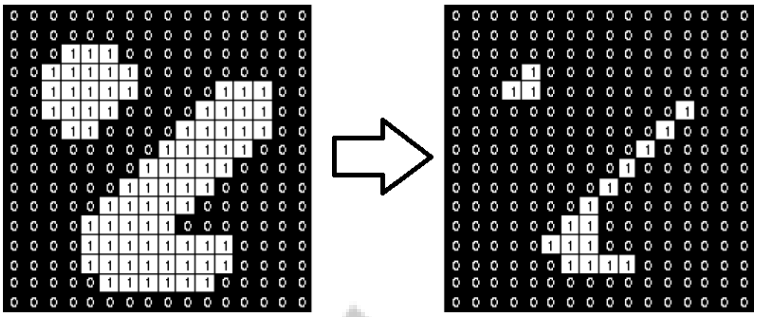


Gambar 2.12. Operasi morfologi dilasi[20].

Operai morfologi dinyatakan dengan rumus A⊕B. Dimana A adalah citra asli dan B adalah struktur elemen. Dengan menggunakan struktur elemen, sebuah citra akan dikenai operai dilasi. Sehingga objek akan semakin lebar. Struktur elemen adalah matrik yang berisi nilai. Struktur elemen ini yang akan di konvolusikan dengan citra biner.

2.6.7. Erosi

Erosi mempunyai efek memperkecil struktur dari sebuah citra. Operasi morfologi erosi mempunyai rumus A⊖B. Dimana A adalah citra asli dan B adalah struktur elemen Erosi adalah kebalikan dari dilasi yaitu teknik mengikis tepi objek dengan menjadi titik objek (1) yang bertetangga dengan titik latar(0) menjadi titik latar(0)[20]. Operasi ini digunakan untuk memisahkan objek yang terlihat berdekatan. Operasi morfologi dilasi ditunjukan pada gambar 2.13.



Gambar 2.13. Operasi morfologi erosi[20].

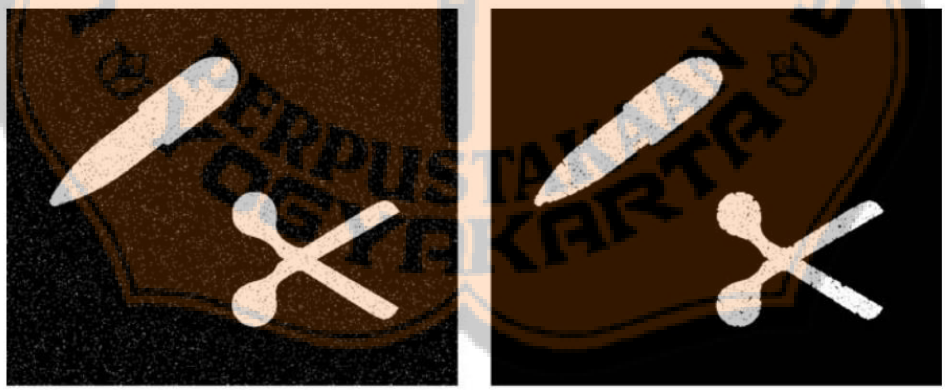
Dengan menggunakan struktur elemen, sebuah citra akan dikenai operai erosi. Sehingga setiap objek akan mengikis. Struktur elemen adalah matrik yang berisi nilai. Struktur elemen ini yang akan di konvolusikan dengan citra biner.

2.6.8. Opening

Opening adalah operasi morfologi citra pada suatu citra yang beraras keabuan atau pada citra biner. Operasi ini sangat berguna untuk menghilangkan *noise salt and pepper* yang terdapat pada citra, namun dengan menghilangnya *noise* pada citra, akan menurunkan ukuran suatu objek[19]. Sehingga objek yang terdeteksi mengalami pengikisan. Secara algoritma, opening dapat dituliskan dengan notasi:

Opening(Citra,SE) = Citra o SE=Dilasi(Erosi(Citra,SE),SE) [19].

Dalam operasi opening sebuah citra biner akan dikenai operasi erosi dengan struktur elemen. Selanjutnya hasil dari erosi akan dikenai operasi dilasi dengan struktur elemen tersebut.



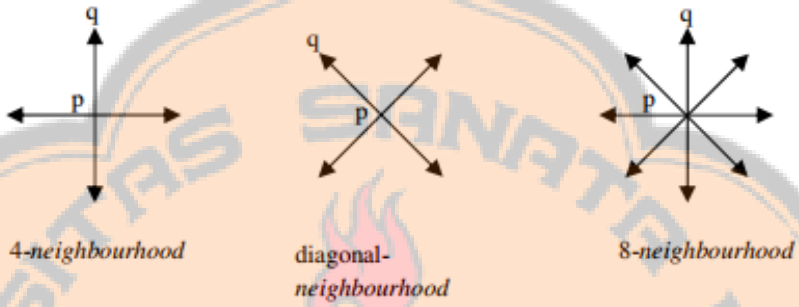
(a) (b)

Gambar 2.14. (a) Citra biner *salt and pepper noise*, (b) citra hasil *opening*[17].

Pada gambar 2.14 menunjukan dilihat bahwa *noise salt and pepper* hilang setelah dikenai operasi morfologi opening.

2.6.9. Labeling

Citra biner banyak dipakai pada pengolahan suatu citra yang akan diproses lebih lanjut. Pada citra biner akan memperlihatkan sejumlah objek melalui nilai nilai yang muncul. Objek yang muncul dapat dihitung menggunakan pelabelan[15]. Teknik ini memanfaatkan teori dari *connectivity* piksel pada citra. Terdapat aturan ketetanggaan yang menentukan sebuah piksel berada pada satu *region* ketetanggaan. Terdapat beberapa jenis aturan ketetanggaan, diantaranya 4-ketetanggaan, 8-ketetanggaan seperti gambar 2.15.



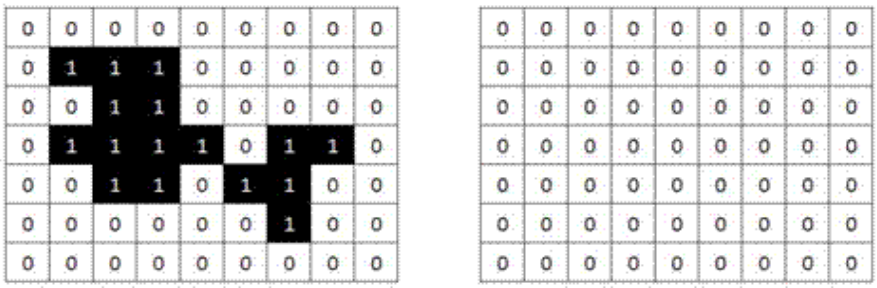
Gambar 2.15. Jenis ketetanggaan[23].

Pada dasarnya untuk menentukan objek yang diberi label pada piksel yang terkoneksi, perlu dilakukan proses *scaning* terhadap keseluruhan piksel secara berurutan. Scaning tersebut dilakukan dengan memperhatikan aturan ketetangaan yang akan dipakai. Setiap jenis ketetanggaan memiliki fungsi sendiri. Jenis 4-ketetanggaan digunakan untuk memisahkan objek yang memiliki nilai piksel antar diagonal, tetapi pada jenis 8-ketetanggaan wilayah yang tercakup lebih banyak karena terdapat 8 ketetanggaan untuk menentukan label. Gambar 2.16 menggambarkan sebuah matriks 4-ketetanggaan.

	P(x, y-1)	
P (x-1, y)	P (x,y)	P(x+1, y)
	P(x, y+1)	

Gambar 2.16. Pendekatan 4-ketetanggaan dalam piksel[23].

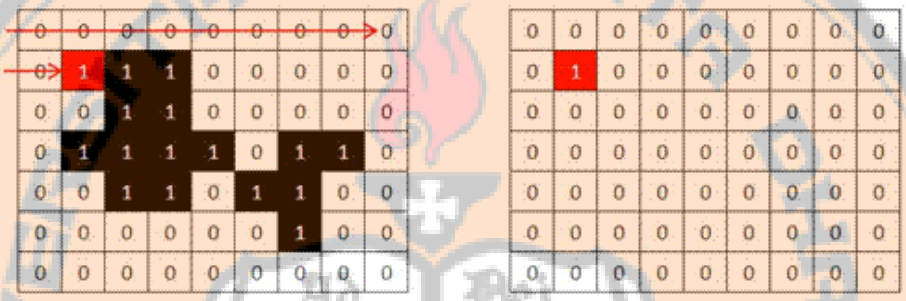
Piksel yang berdekatan secara horisontal dan vertikal dianggap memiliki hubungan 4-ketetanggaan sehingga piksel yang berdekatan secara diagonal akan dianggap sebagai objek selanjutnya.



(a) (b)

Gambar 2.17. (a) Matrik citra *original*, (b) matrik *mapping*[23]

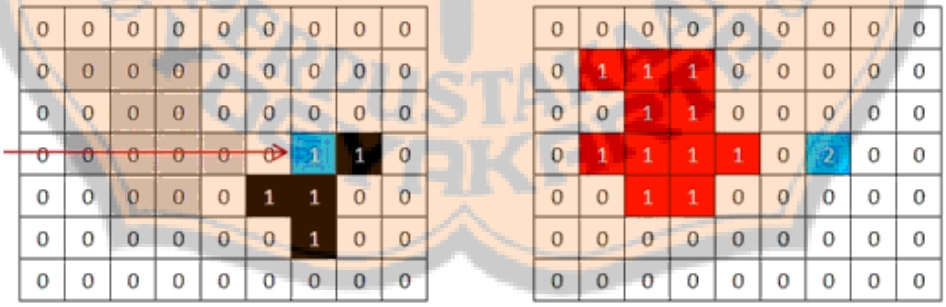
Gambar 2.17 adalah citra yang akan dikenai operasi *labeling* dengan pendekatan 4-ketetanggaan dan dipetakan, menggunakan proses scaning.



(a) (b)

Gambar 2.18. (a)*Scanning* citra setiap piksel, (b) matrik *mapping*[23].

Proses *scaning* ini dilakukan seperti gambar 2.18 dari sisi atas sebelah kiri ke kanan setelah itu beganti ke baris di bawahnya, proses ini dilakukan setiap piksel citra *foreground*.



(a) (b)

Gambar 2.19. (a) Proses *scanning* matrik citra, (b) Matrik *mapping*[23].

Jika saat proses *scaning* ditemukan piksel *foreground*, maka dilakukan pelabelan terhadap piksel tersebut. Pada gambar 2.19 matrik *mapping* bernilai 2 karena terpisah dengan piksel

bernilai 0, dengan menggunakan aturan 4-ketetanggaan maka piksel tersebut dianggap sebagai objek kedua dari sebuah citra.

0	0	0	0	0	0	0	0	0
0	1	1	1	0	0	0	0	0
0	0	1	1	0	0	0	0	0
0	1	1	1	1	0	2	2	0
0	0	1	1	0	2	2	0	0
0	0	0	0	0	0	2	0	0
0	0	0	0	0	0	0	0	0

Gambar 2.20. Hasil matrik proses *scaning*[23].

Proses *scaning* dilakukan berulang ulang menggunakan pendekatan 4-ketetanggaan, sehingga seluruh seluruh piksel pada citra dapat dilabeli seperti gambar 2.20 yang memiliki dua objek citra.

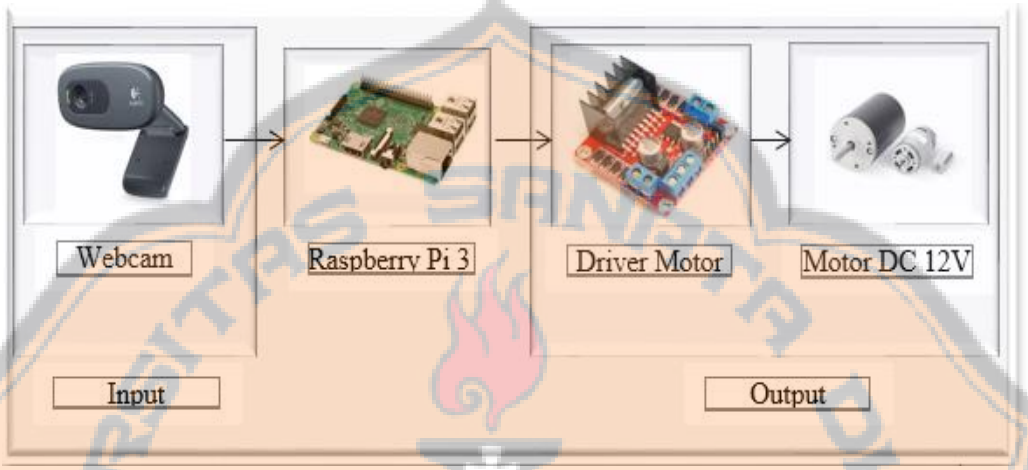
2.6.10. Perhitungan Objek

Proses *labeling* akan menentukan jumlah objek dengan melabeli setiap objek sesuai kaidah 4-ketetanggaan. Hasil dari *labeling* tersebut akan dihitung jumlah objek yang terdapat pada citra. Perhitungan objek citra dilakukan dengan cara melakukan scaning di setiap baris dari di setiap kolom secara berurutan, dengan menentukan nilai awal yang di ambil dari nilai matrik baris dan kolom untuk acuan nilai maksimal. Proses scaning nilai matrik dibandingkan dengan nilai acuan awal, jika nilai acuan kurang dari atau sama dengan nilai matrik maka nilai matrik tersebut dinyatakan sebagai nilai maksimmal sementara dan dijadikan sebagai nilai acuan baru. Proses tersebut dilakukan hingga scaning selesai pada baris dan kolom matrik terakhir. Setelah itu nilai yang menjadi acuan terakhir adalah nilai maksimal dari matrik tersebut. Citra yang sudah dilabel pada gambar 2.20 akan dilakukan perhitungan objek dan didapatkan jumlah objek sebanyak dua.

BAB III
RANCANGAN PENELITIAN

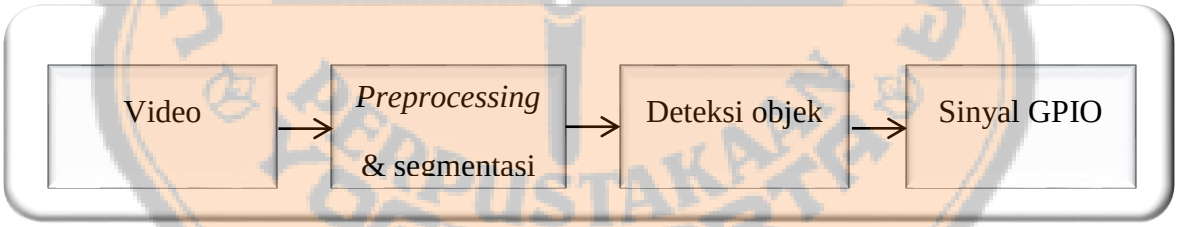
3.1. Perancangan Sistem Secara Umum

Perancangan sistem ini memiliki beberapa komponen utama seperti Gambar 3.1.



Gambar 3.1. Diagram I/O sistem.

Webcam menjadi input dari sistem dengan mendeteksi keberadaan burung, sehingga Raspberry Pi akan mengeluarkan sinyal untuk menggerakan motor DC sebagai aktuator sistem. Di dalam Raspberry Pi terjadi proses pengolahan citra seperti Gambar 3.2.



Gambar 3.2. Blok diagram proses pengolahan citra.

Selain itu memiliki komponen pendukung seperti monitor, keyboard, mouse. Sistem ini akan dibuat dengan bahasa pemrograman *python* pada Raspberry Pi3 yang sudah diinstall dengan operating system raspbian. Kamera yang digunakan untuk menangkap citra adalah Webcam. Untuk mengontrol program yang akan dijalankan pada Raspberry Pi3 dibutuhkan WiFi sebagai perantara komunikasi dengan pengguna melalui Laptop. Proses pengolahan citra dapat dilihat dalam blok diagram Gambar 3.2.

Pemrosesan citra video akan diolah dengan bantuan *Library* OpenCV. *Library* OpenCV ini dapat membantu proses pengolahan citra video, karena sudah terdapat fungsi fungsi yang mendukung proses pengolahan.

Perancangan sistem dibagi menjadi dua bagian seperti yaitu pengambilan data secara *non realtime* dan pengujian secara *realtime*. Pengambilan data dilakukan terlebih dahulu untuk menguji data secara *non realtime*. Pengambilan data ini dilakukan dengan cara merekam objek yang akan di teliti yaitu lahan sawah siap panen, pada lahan inilah hama burung menyerang tanaman padi. Proses pengambilan data *non realtime* ini bertujuan untuk mengambil citra hama burung di lahan sawah dengan cara merekam selama 40 detik supaya pergerakan hama burung tertangkap. Setelah itu data video disimpan di *microSD* dalam format MP4. Pengambilan data *non realtime* dilakukan hingga mendapatkan beberapa video untuk menambah variasi pergerakan burung. Data video yang di dapat digunakan sebagai bahan yang akan diolah sehingga mendapatkan program untuk diterapkan pada pengujian *realtime*.

Proses selanjutnya adalah pengujian *realtime*, setelah pembuatan program yang dilakukan pada pengambilan data *non realtime* selanjutnya program akan diterapkan pada pengujian *realtime* sehingga video yang diolah adalah video secara *realtime*. Pada saat pengujian *realtime*,burung akan terdeteksi oleh Webcam ,lalu akan diproses pada program sehingga saat terdeteksi Raspberry Pi akan mengirimkan sinyal untuk menggerakan aktuator berupa motor DC yang sudah di desain untuk menggerakan tali sebagai alat pengusir burung.

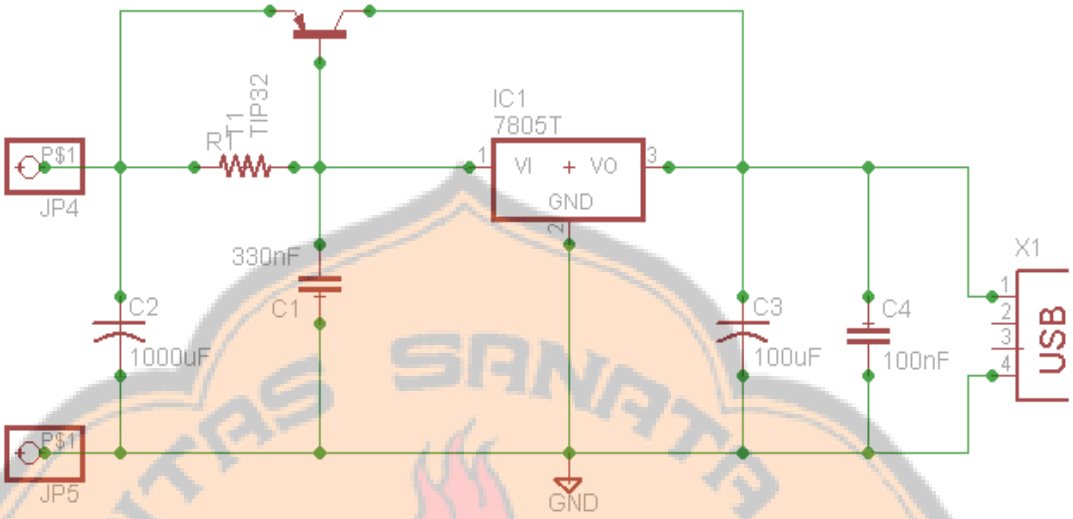
3.2. Perancangan Perangkat Keras (Hardware)

Perancangan perangkat keras dalam sistem ini dibagi menjadi 3 bagian. Bagian pertama adalah rangkaian catu daya yang berfungi sebagai sumber daya Raspberry Pi 3 dan driver motor. Sumber catu daya akan dicabangkan karena daya yang dibutuhkan Raspberry Pi 3 dan driver motor berbeda. Bagian kedua rangkaian driver motor yang berfungsi untuk menggerakan motor DC. Bagian ketiga bentuk fisik model aktuator untuk mengusir hama burung tersebut.

3.2.1. Rangkaian Catudaya

Sumber tegangan digunakan dalam sebuah sistem untuk memberikan supply tegangan pada setiap blok sistem. Pada penelitian ini sumber yang dibutuhkan berbeda, maka dari itu di buat rangkaian catu daya yang berfungsi untuk memparalelkan tegangan dan menurunkan tegangan. Sumber yang digunakan adalah baterai lipo 3S, sementara bagian

yang harus disupply adalah Raspberry Pi dan driver motor. Skematik rangkaian catu daya pada sistem dapat dilihat pada gambar 3.3.

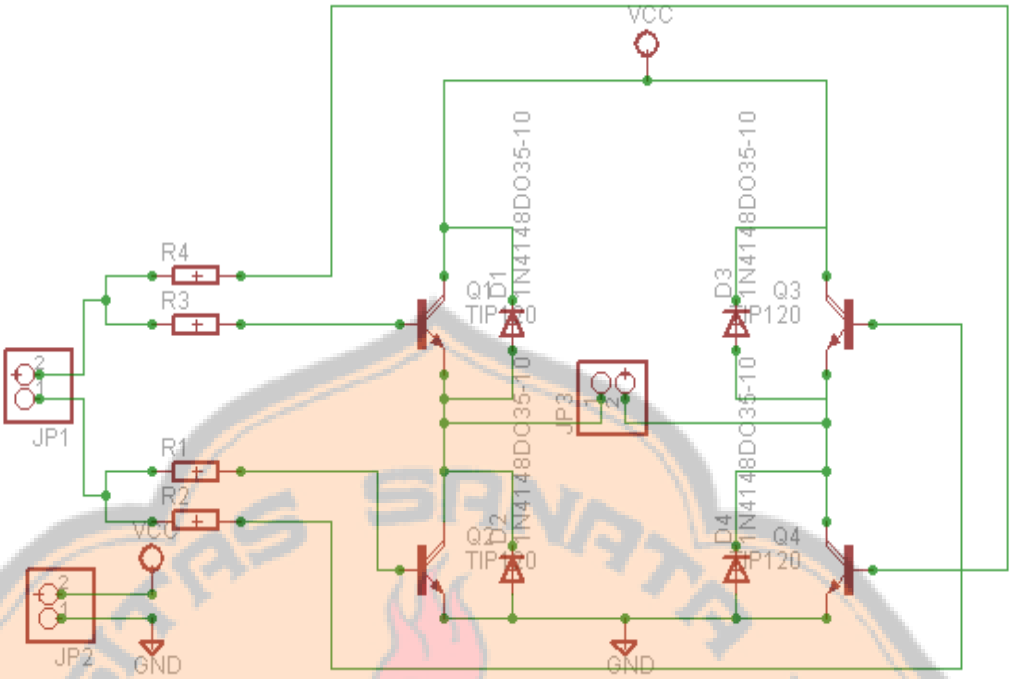


Gambar 3.3. Skematik rangkaian catu daya[21].

Raspberry Pi harus mendapatkan supply 5 volt dengan arus lebih dari 2 Ampere dengan *output* USB. Motor DC menggunakan tegangan ±12 volt dan untuk mengoperasikan sumber tersebut dihubungkan dengan driver motor. Skema rangkaian catu daya ini dapat menyuplai tegangan 5 Vdc dengan arus mencapai 3 Ampere, rangkaian ini menggunakan ic regulator 7805 untuk merubah tegangan input dari baterai lipo 3S menjadi 5 Vdc. TIP32 digunakan sebagai penguat arus, transistor jenis PNP ini mampu memberikan arus hingga 3 Ampere. *Output* dari rangkaian ini menggunakan USB sebagai penghubung *supply* ke Raspberry pi3.

3.2.2. Rangkaian Driver Motor

Driver motor DC digunakan untuk menggerakan motor supaya dapat berputar 2 arah putaran yang berbeda, arah putaran tersebut adalah serah jarum jam dan berlawanan arah jarum jam. Driver motor ini menggunakan logika 0 atau 1 untuk menggerakan motor, dengan menggunakan prinsip H-Bridge transistor maka arah putaran motor dapat diatur dengan mengaktifkan sepasang transistor dan me*non*aktifkan sepasang transistor lawannya. Masukan driver motor ini merupakan berupa logika TTL yaitu 0 volt dan 5 volt dengan input tegangan 12 volt dari catu daya.

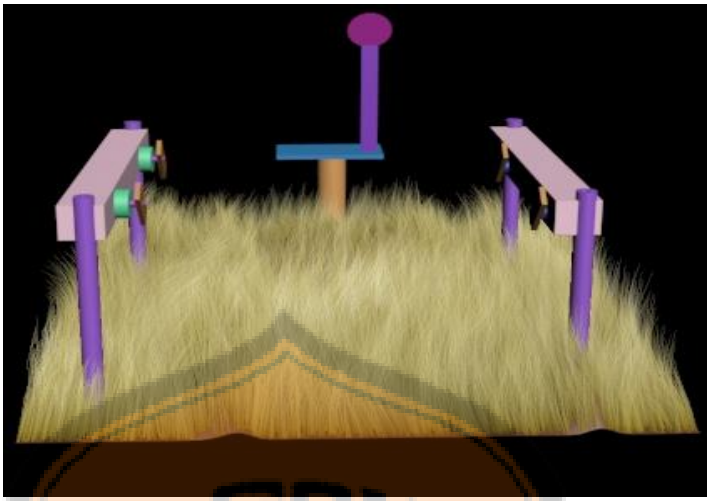


Gambar 3.4. Skematik driver motor[22].

Transistor TIP120 adalah jenis transistor NPN yang di dalamnya terdapat transistor darlington. TIP120 ini memiliki spesifikasi hfe 2500x pada arus IC 4 Ampere. Arus *collector*(Ic) mencapai 5 Ampere. Arus *collector* maksimum mencapai 8 Ampere. *Diode* dipasang sebagai proteksi pada transistor terhadap beban induksi dengan tipe 1N4007, sehingga arus maksimum yang dapat dilewati 7 Ampere.

3.2.3. Rancangan Bentuk Fisik

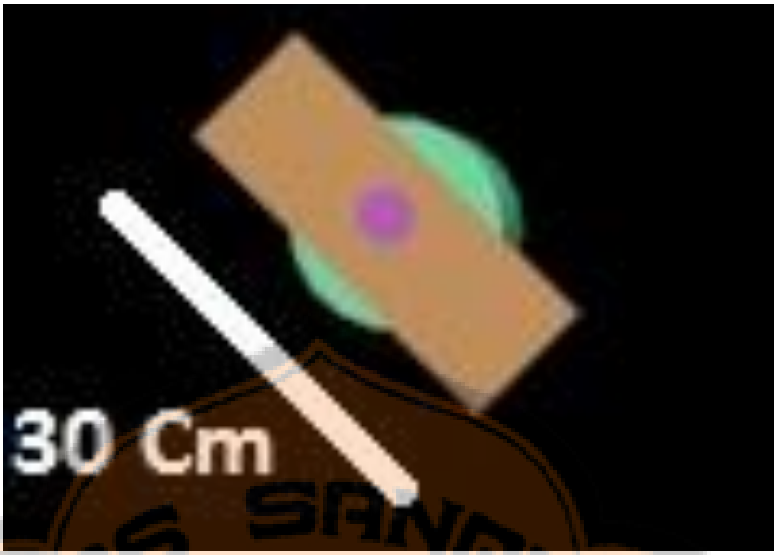
Pada proses pengusiran burung, aktuator yang digunakan adalah dua motor DC 12V. Pada lahan sawah 4 meter x 4 meter jarak antar motor DC adalah 2 meter. Motor DC diletakkan 1 meter dari tiap ujung penyangga. Putaran satu motor DC diasumsikan akan mencakup 2 meter x 2 meter. Peletakan tiang penyangga diletakkan 50 cm dari lebar area wilayah 4 meter x 4 meter. Dikarenakan area 4 meter x 4 meter akan dicrop sehingga supaya tidak mengganggu proses saat pengolahan maka tiang harus di jauhkan. Rancangan *hardware* dapat dilihat pada gambar 3.5 dan gambar 3.6.



Gambar 3.5. Rancangan *hardware* tampak depan.

Gambar 3.6. Rancangan *hardware* tampak samping kiri.

Pada aktuator motor DC dipasang penampang untuk tali diikat, sehingga saat motor berputar tali tersebut ikut berputar. Setiap tali ujungnya akan diikat pada ujung penampang lainnya yang memiliki penampang yang sama. Tali tesebut akan diberi bandul lonceng pemberat untuk memberikan bunyi bunyian, sehingga selain berputar tali tersebut akan menghasilkan bunyi yang akan membuat burung terkejut. Bentuk fisik aktuator dapat dilihat pada gambar 3.7 serta ukuran lahan sawah yang di ambil digambarkan pada gambar 3.8.



Gambar 3.7. Bentuk fisik penampang tali.

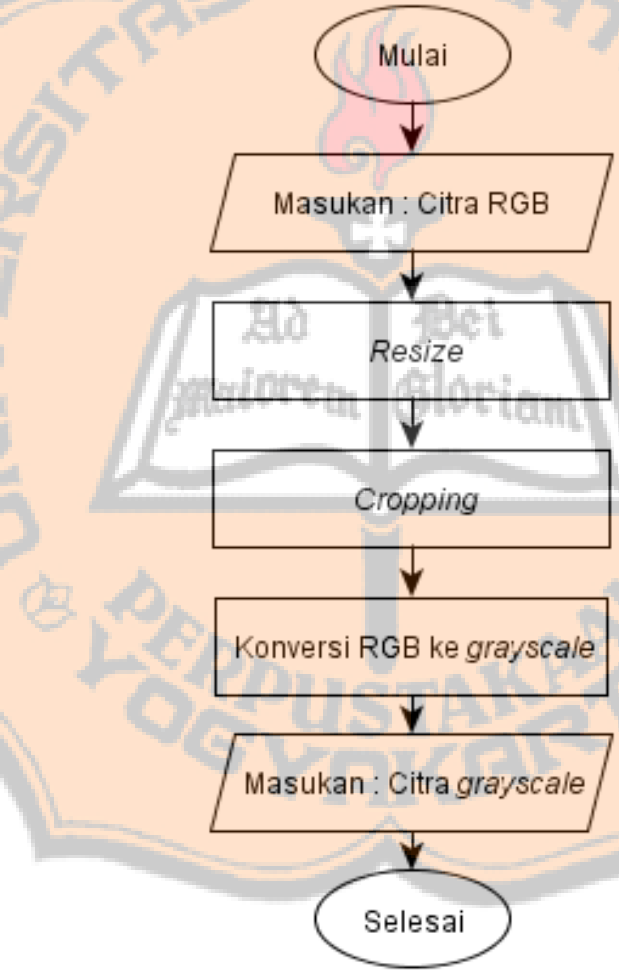
Gambar 3.8. Rancangan *hardware* tampak samping atas.

3.3. Perancangan Perangkat Lunak (Software)

Pada perancangan perangkat lunak akan dibuat program utama dan program sekunder unuk sistem pengusir hama dengan bahasa *python* yang akan di program pada Raspberry Pi3 dengan menambahkan *library* OpenCV untuk proses pengolahan video dan *library* webcam untuk mengakses webcam dalam pengambilan citra. Program utama tersebut adalah proses untuk melakukan pengujian, sementara program sekunder adalah proses pengambilan video *non realtime*. Perancangan program terdiri dari beberapa tahap diantaranya *preprocessing*, segmentasi, *labeling* dan proses akhir adalah penentuan keluaran yang akan mengirim sinyal untuk mengaktifkan aktuator.

3.3.1. Preprocessing

Preprocessing dilakukan untuk memilih citra RGB yang akan diproses menggunakan *cropping* dengan cara segmentasi koordinat yang dipilih, lalu mengubah citra RGB yang terpilih ke dalam citra biner dengan menggunakan *thresholding*. Proses ini memiliki beberapa tahap yaitu pengubahan citra RGB manjadi *grayscale*, citra yang diolah adalah citra yang ditangkap oleh Webcam. Citra *grayscale* tersebut akan diubah menjadi citra biner, untuk mengubah citra *grayscale* dalam citra biner menggunakan nilai *thresholding* tertentu. Jenis *threshold* yang digunakan adalah global *thresholding* Pada OpenCV untuk menjadikan citra RGB menjadi citra *grayscale* menggunakan perintah COLOR_BGR2GRAY. Proses *preprocessing* digambarkan pada Gambar 3.9.



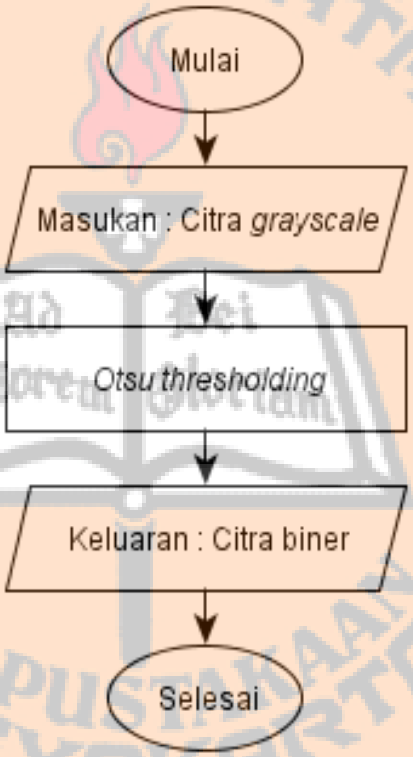
Gambar 3.9. *Flowchart preprocessing.*

3.3.2. Segmentasi

Segmentasi digunakan dalam pengolahan citra untuk memisahkan *foreground* dan *background*. Dalam penelitian ini *foreground* adalah burung dan *background* adalah lahan sawah. Untuk memisahkan *foreground* dari *background* maka citra tersebut harus dikenai

threshold supaya *foreground* dapat dipisahkan dari *background*. *Threshold* yang digunakan adalah jenis *threshold* global. Secara umum, proses *thresholding* merupakan citra masukan f(x,y) yang akan menjadi g(x,y) dengan menggunakan nilai *threshold* (T),Dimana nilai T digunakan untuk *threshold* semua piksel pada citra, maka disebut global *thresholding*[17].

Global *thresholding* akan lebih optimal jika pencahayaan pada citra merata. Salah satu jenis global *thresholding* adalah *Otsu thresholding*. Untuk kasus yang mempunyai banyak objek pada *background* maka perlu disegmentasi menggunakan *threshold* global[17]. Pada OpenCV untuk menjadikan citra *grayscale* menjadi citra biner menggunakan perintah THRESH_BINARY dan menambahkan perintah THRESH_OTSU pada program. Proses segmentasi digambarkan pada gambar 3.10.

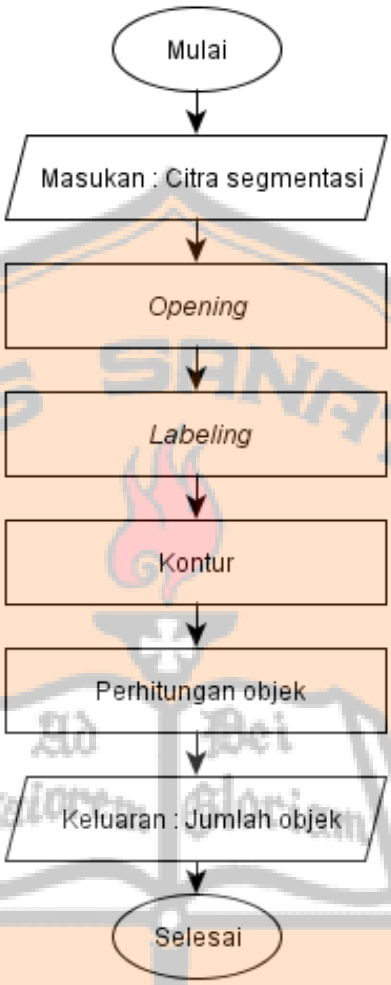


Gambar 3.10. *Flowchart* segmentasi

3.3.3. Deteksi objek

Deteksi objek adalah proses pengolahan sebuah citra yang dilakukan setelah tahap *preprocessing*, dengan *input* citra biner yang diperoleh dari *webcam* akan dideteksi objek yang terdapat pada citra biner tersebut. Objek yang terdeteksi adalah burung pada lahan sawah. Pada proses deteksi objek terdapat pelabelan dan perhitungan objek. Objek citra biner akan dilabel sehingga dapat diketahui burung yang ada di lahan sawah, setelah itu

perhitungan objek dilakukan untuk mengetahui berapa banyak objek pada lahan sawah tersebut. Proses deteksi objek digambarkan pada gambar 3.11.



Gambar 3.11. *Flowchart* deteksi objek.

Citra yang telah disegmentasi masih dapat mengalami kesalahan dalam perhitungan jumlah objek, kemungkinan *noise salt and pepper* yang terjadi sangat tinggi. Proses penghilangan *noise* tersebut dapat dilakukan menggunakan operasi morfologi opening. Dalam kasus opening, citra pertama tama dikenai operasi erosi lalu dilanjutkan dengan dilasi. Secara umum dilasi adalah pelebaran daerah citra dan erosi adalah pengikisan daerah citra[24]. Pada operasi morfologi sebuah citra dikonvolusi dengan struktur elemen yang telah di tentukan.

Dilasi adalah operasi konvolusi citra dengan struktur elemen, struktur elemen disebut juga sebagai template. Saat dikonvolusikan struktur elemen akan tumpang tindih dengan citra mengakibatkan nilai piksel diganti dengan nilai maksimal dari struktur elemen tersebut. Sebaliknya erosi adalah kebalikan dari dilasi, menghitung nilai piksel minimal yang tumpang

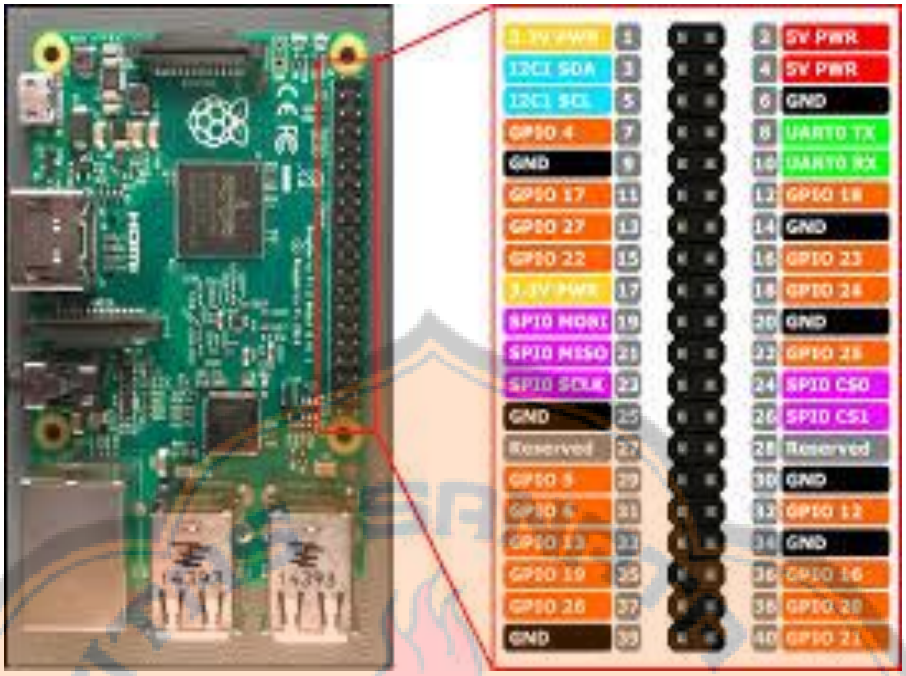
tindih struktur elemen dengan citra dan mengganti piksel citra dengan nilai minimum dari struktur elemen[24]. Penentuan struktur elemen akan berpengaruh terhadap citra, semakin besar struktur elemen jika objek mempunyai luasan yang kecil maka akan hilang. Sehingga struktur elemen ditentukan dengan melihat resolusi citra terlebih dahulu. Pada OpenCV untuk memberikan operasi morfologi opening pada citra dapat menggunakan perintah MORPH_OPEN

Labeling merupakan teknik yang digunakan untuk mengklasifikasikan region dalam citra digital, dari situ dapat diketahui keberadaan objek citra yang diambil oleh webcam sehingga sistem dapat menentukan keadaan lahan sawah terdeteksi adanya burung atau tidak. Objek citra yang telah dilakukan operasi morfologi akan dideteksi dengan cara pelabelan dalam bentuk matrik. Proses pelabelan matrik citra dilakukan seperti gambar 2.17 hingga 2.20. Pada OpenCV untuk memberikan label pada citra dapat menggunakan perintah labeled_array.

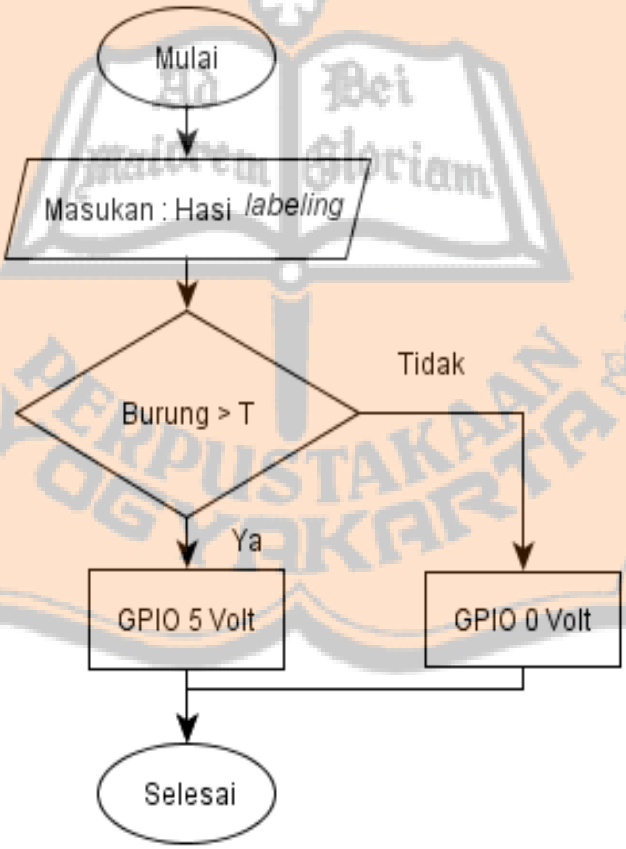
Objek citra dalam bentuk matrik tersebut akan di ketahui berapa jumlah burung yang terdapat pada lahan sawah. Sehingga keluaran dari citra yang telah dilabel akan dihitung jumlahnya. Pada OpenCV untuk perhitungan objek pada citra yang telah diberi label dapat menggunakan perintah num_features. Perhitungan jumlah ini adalah proses terakhir pada proses deteksi objek yang akan menentukan jumlah burung yang terdapat pada lahan sawah. Perhitungan objek ini dilakukan dengan menentukan berapa jumlah batasan yang diambil sebagai acuan dalam proses deteksi objek.

3.3.4. Penentuan Keluaran

Proses penentuan keluaran menjadi akhir dari pengolahan citra setelah di dapatkan hasil akhir dari deteksi objek. Tahap ini akan menghasilkan dua kondisi, yaitu kondisi pertama jika hasil deteksi objek tidak ditemukan objek maka pin GPIO bernilai 0 volt. Kondisi kedua yaitu saat *labeling* terdeteksi adanya burung maka akan terhitung jumlah burung. Sehingga dapat dituliskan jika jumlah burung ≥ 6 maka GPIO 5 volt dan jika jumlah burung < 6 maka GPIO 0 volt. Gambar 3.13 menunjukan diagram alur penentuan keluaran system. GPIO ini adalah pin I/O pada Raspberry Pi yang digunakan untuk memberikan sinyal pada driver motor. Keluaran pin GPIO bernilai $\pm$ 5 volt, pin GPIO tersebut akan menjadi input driver motor pada aktuator sistem untuk menggerakan motor DC. Gambar 3.12 adalah *mapping* pin GPIO pada Raspberry Pi 3.



Gambar 3.12. GPIO pin Raspberry Pi3[25]



Gambar 3.13. *Flowchart* penentuan keluaran.

3.3.5. Start Up Program

Start up program adalah sebuah instruksi untuk menjalankan program dalam sistem tanpa harus memberikan perintah dari luar. Raspberry Pi akan menjalankan sistem saat mendapat supply dari catu daya. Dalam Raspbian OS memiliki fitur *run program automatically* dengan cara menuliskan *script*. Pertama, buat *script init.ini* perlu dibuat di folder /etc/init.d/. setelah itu buat *file* baru menggunakan perintah sudo nano /etc/init.d/scriptnya, kemudian supaya *script* dapat dijalankan maka perlu diaktifkan dengan sudo chmod 755 /etc/init.d/scriptnya. Langkah terakhir adalah menjadikan *script* berjalan saat Raspberry selesai *booting* dengan memberikan perintah sudo update-rc.d scriptnya defaults[26]. Dalam script tersebut berisi tentang program untuk mengintegrasikan perintah menuju lokasi *script* program yang akan dieksekusi.

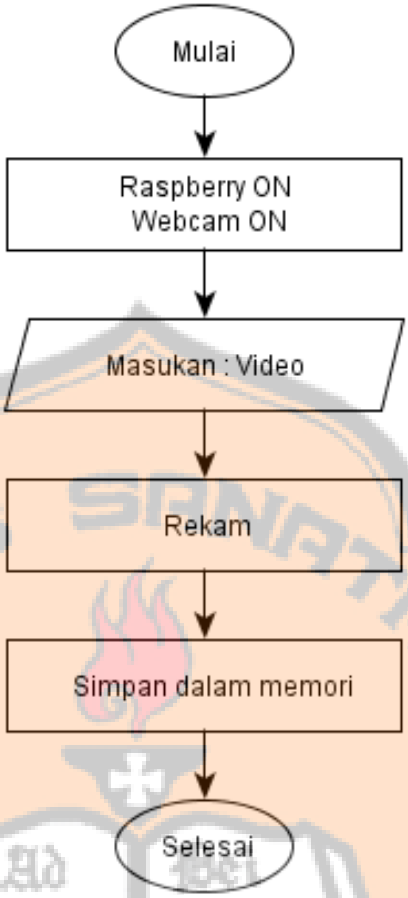
3.4 Proses Pengujian Sistem

Sebelum sistem siap digunakan maka harus dilakukan pengujian terlebih dahulu. Pengujian sistem dilakukan dengan 2 tahapan yaitu pengujian secara *realtime* dan *non realtime*. Pengujian *non realtime* bertujuan mempersiapkan sistem untuk pengujian *realtime* di lahan sawah. Pengujian *realtime* dilakukan untuk mengamati kinerja sistem sehingga didapatkan data untuk dilakukan analisis.

3.4.1. Proses Pengujian Video *Non Realtime*

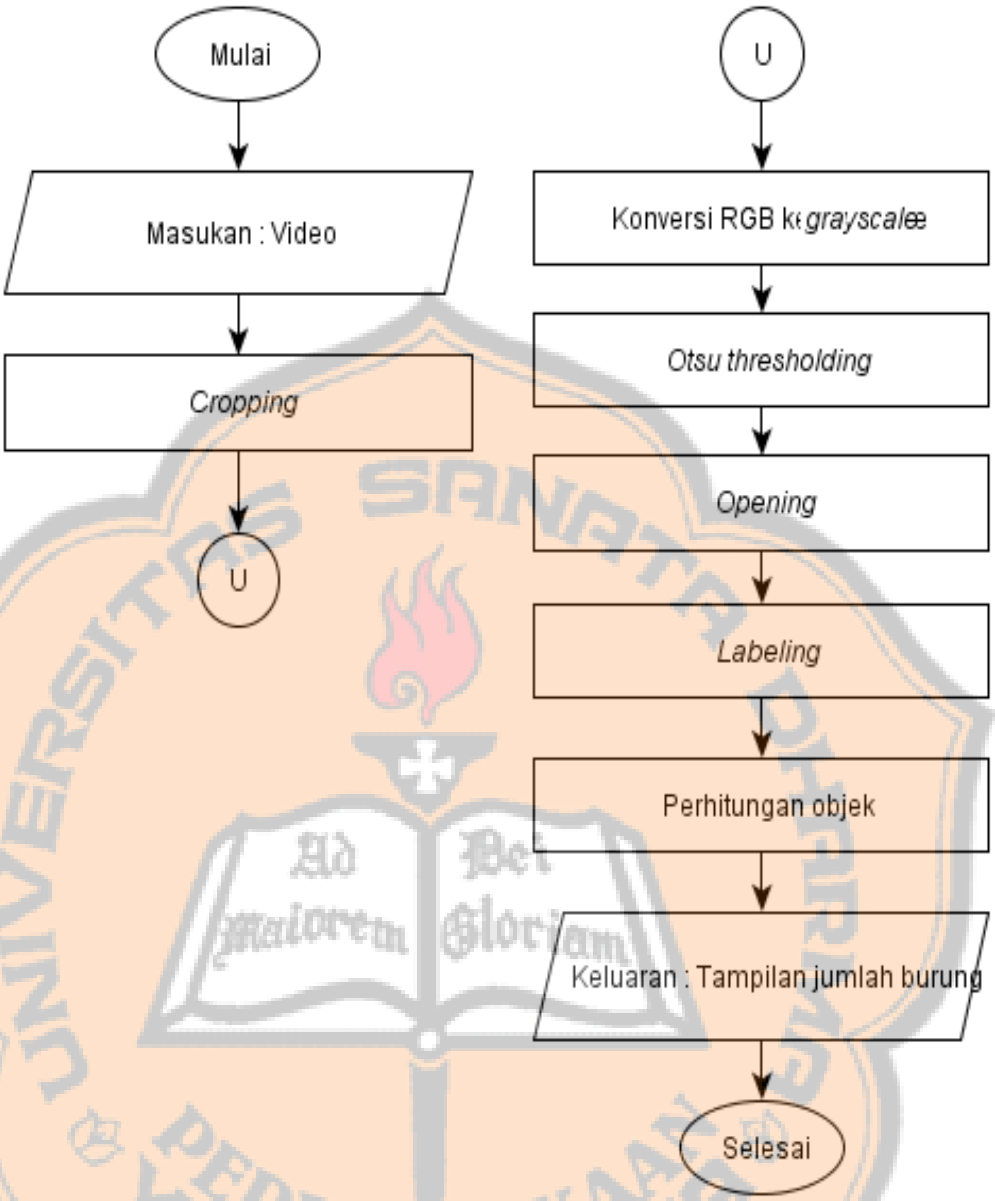
Proses pengambilan video *non realtime* dilakukan dengan menggunakan Webcam, yang akan untuk menangkap citra dan disimpan dalam *microSD*. Webcam akan dihubungkan dalam port USB Raspberry Pi sehingga dapat diterima oleh Raspberry Pi untuk selanjutnya diproses menggunakan *library* OpenCV. *Flowchart* pengambilan video *non realtime* dapat dilihat pada gambar 3.14.

Lama waktu perekaman dilakukan selama 5 menit. Lama waktu perekaman tersebut dapat ditentukan secara manual melalui program. Setelah citra selesai terekam, data yang dihasilkan berekstensi .avi. Data tersebut kemudian disimpan dalam micro SD. Perekaman video dilakukan sebanyak 3 kali. *Flowchart* pengujian *non realtime* dapat dilihat pada gambar 3.15.



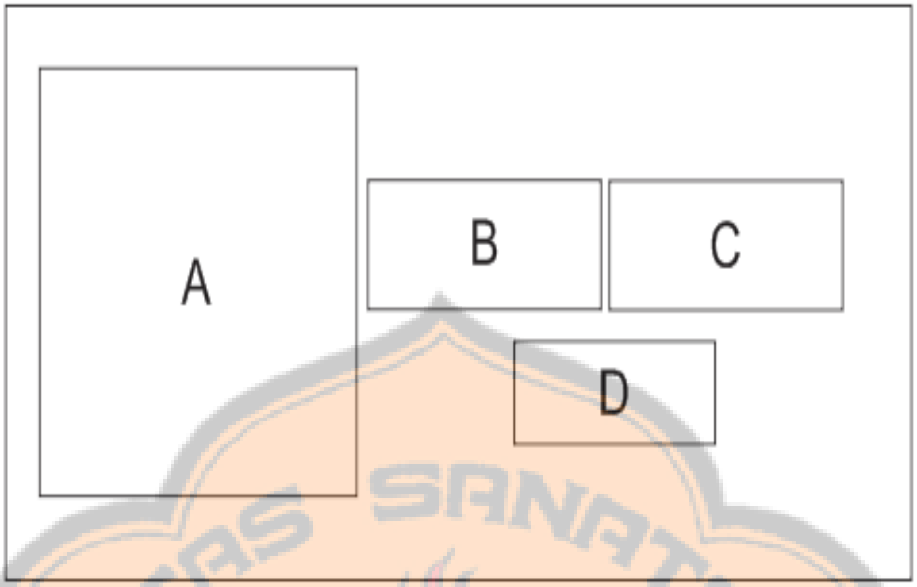
Gambar 3.14. *Flowchart* pengambilan video *non realtime*

Video yang sudah disimpan dalam *microSD* akan dilakukan pengujian yang bertujuan untuk memaksimalkan kerja sistem supaya dapat mengenali bagaimana sistem dapat mengenali burung dan memisahkan objek dari *background*, sehingga saat pengujian *realtime* sistem bekerja maksimal. Pengujian pada saan *non realtime* meliputi penentuan nilai *threshold*, nilai *labeling*, nilai kontur dan nilai learning rate. Penentuan nilai tersebut menggunakan uji coba *trial and error* dengan menggunakan contoh video dari hasil penyimpanan *non realtime*. Hasil dari beberapa nilai yang dimasukan dapat dibandingkan untuk melihat kinerja sistem yang maksimal.



Gambar 3.15. *Flowchart* penngujian *non realtime*

Pengujian *non realtime* ini akan ditampilkan jumlah burung yang terdeteksi yang akan ditampilkan pada monitor, sehingga proses analisis deteksi objek dapat dilihat secara program dan diandingkan dengan citra yang berbeda setiap nilainya. Tampilan proses pengolahan dan jumlah burung akan ditampilkan seperti gambar 3.16.



Gambar 3.16. Tampilan proses pengolahan dan jumlah burung pada monitor.

Keterangan:

A : Tampilan video (RGB) pada lahan sawah.

B : Tampilan video (Biner) hasil proses pengolahan..

C : Tampilan video (RGB) setelah melalui proses pengolahan

D : Tampilan jumlah burung.

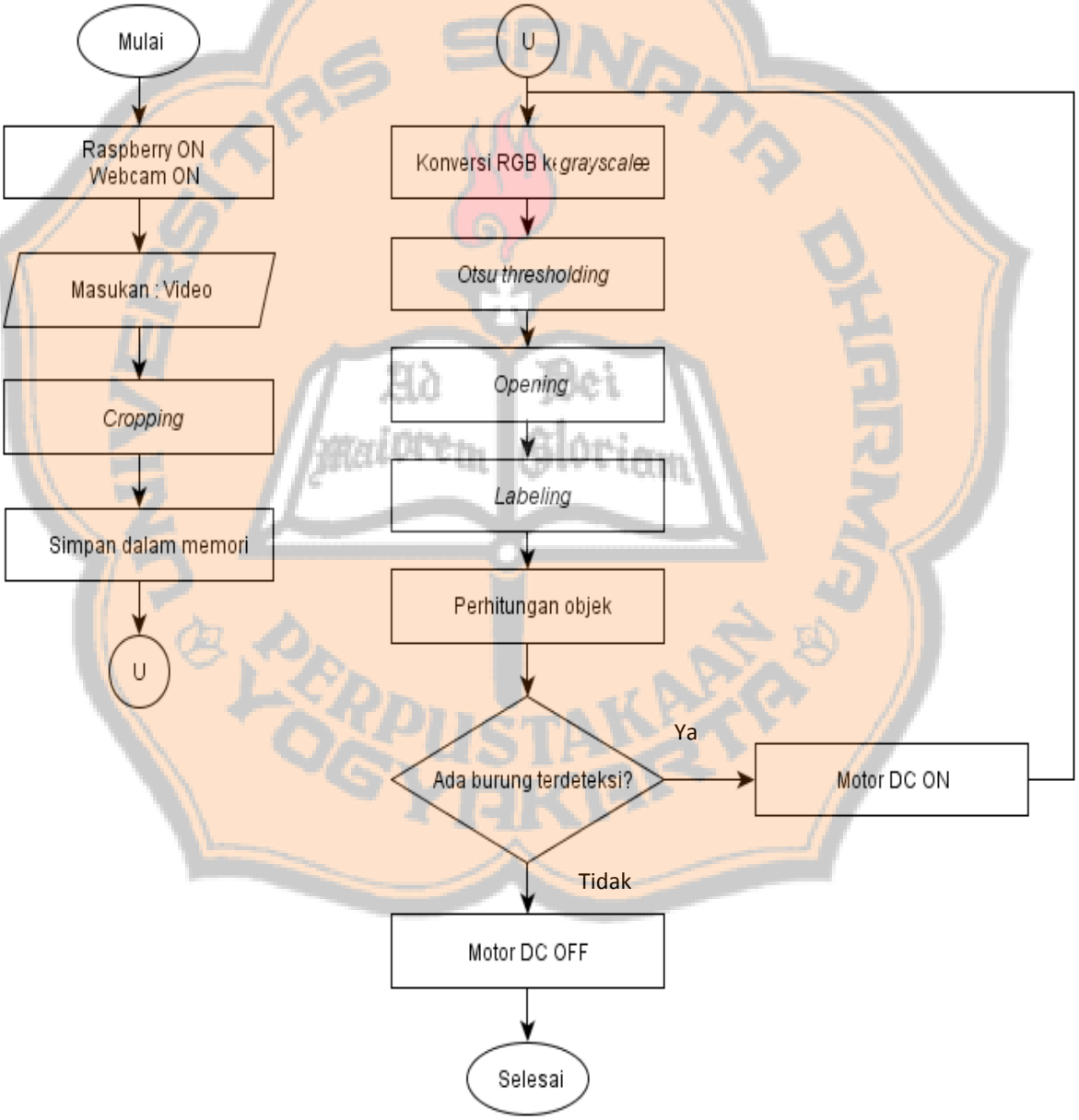
3.4.2. Proses Pengujian *Realtime*

Pengujian *realtime* adalah proses pengujian alat yang telah dirancang secara langsung, pengujian tersebut dilakukan di lahan sawah saat cuaca cerah tidak hujan.selama 5 menit. *Flowchart* pengujian seara *realtime* dapat dilihat pada gambar 3.17.

Pengujian ini dilakukan setelah memperoleh hasil akhir dari serangkaian uji coba pada video *non realtime*. Dengan menggunakan input citra RGB secara *realtime* maka akan di konversi menjadi citra biner. Dalam proses pengolahannya citra biner dilakukan *preprocessing* meliputi operasi morfologi dan *labeling*. Operasi morfologi yang dipakai pada operasi citra adalah *opening*. Morfologi *opening* dapat digunakan untuk menutupi lubang yang cacat pada objek citra. Citra *opening* akan dilakukan pemindaian menggunakan kernel yang telah ditentukan. Pemindaian ini merupakan operasi *labeling* yang digunakan untuk mengetahui objek yang ada di dalam citra tersebut. Objek akan terhitung secara keseluruhan. Objek yang terhitung dalam *labeling* adalah citra dengan piksel minimum sesuai dengan kernel. Hasil dari *labeling* berupa angka yang digunakan untuk proses selanjutnya. Proses selanjutnya akan terjadi pengambilan keputusan dengan menggunakan nilai keluaran dari

labeling, hasil dari pengambilan keputusan tersebut akan mengeluarkan sinyal agar motor DC bergerak untuk mengusir hama burung jika pada lahan sawah terdeteksi keberadaan burung.

Kinerja sistem akan diukur berdasarkan akurasi sistem dalam mengusir hama burung dengan menggerakan tali saat sistem mendeteksi adanya burung. Saat pengujian berlangsung dilakukan perekaman untuk mengamati kerja sistem serta pengambilan data dan analisis data.



Gambar 3.17. *Flowchart* pengujian video *realtime*.

Pengambilan data dilakukan dengan cara menghitung tingkat keberhasilan saat sistem bekerja selama 5 menit. Perhitungan burung dilakukan secara manual, dengan harapan sistem dapat mengusir burung. Selama sistem bekerja perekaman akan terus dilakukan. Proses pengujian *realtime* dilakukan satu kali pada lahan sawah.

BAB IV
HASIL DAN PEMBAHASAN

4.1. Implementasi program

Untuk mempermudah suatu program dalam sebuah sistem, python memiliki beberapa fitur untuk membantu proses pengolahan data. Fitur tersebut dapat diakses melalui *library* maupun package yang telah terinstal pada python. Untuk mengakses *library* maupun *package* yang digunakan, suatu program harus menginisialisasikan *header* pada awal program seperti pada gambar 4.1.

```
import cv2
import numpy as np
import scipy
from scipy import ndimage
import time
from time import sleep
import RPi.GPIO as GPIO
```

Gambar 4.1 Inisialisasi header pada program.

Library dan *package* yang digunakan antara lain adalah OpenCV yang dituliskan cv2. Numpy(*numerical python)* untuk membantu perhitungan array pada python diinisialkan sebagai np. Scipy*(science python)* digunakan untuk membantu perhitungan untuk pengolahan citra dalam perhitungan objek. Pada scipy terdapat *library* ndimage yang dimasukan pada inisialisasi selain itu untuk membuat *delay library* time dapat dimasukan pada program.

```
in1=27
in2=22
in3=20
in4=21
```

Gambar 4.2 Inisisalisasi pin raspberry pi.

Pin raspberry pi yang digunakan dalam sistem akan di dinisialisasikan seperti pada gambar 4.2. Setiap pin yang diinisialisasikan disimpan pada variabel yang berbeda.

```
GPIO.setmode(GPIO.BCM)
GPIO.setup(in1,GPIO.OUT)
GPIO.setup(in2,GPIO.OUT)
GPIO.setup(in3,GPIO.OUT)
GPIO.setup(in4,GPIO.OUT)
GPIO.output(in1,GPIO.LOW)
GPIO.output(in2,GPIO.LOW)
GPIO.output(in3,GPIO.LOW)
GPIO.output(in4,GPIO.LOW)
```

Gambar 4.3 Inisisalisasi nilai pin raspberry pi.

Raspberry juga harus dilakukan inisialisasi kondisi pin yang digunakan dalam penggunaanya. Kondisi pin yang di gunakan diinisialkan dengan nilai awal pada keluaran pin tersebut seperti pada gambar 4.3. Pin pada Raspberry memiliki dua mode, mode tersebut dapat dapat dipilih salah satu. BCM dan BOARD adalah mode yang berbeda, yang membedakan keduanya adalah perhitungan pin pada I/O Raspberry.

```
cap = cv2.VideoCapture('file.formatfile')
cap = cv2.VideoCapture(0)
```

Gambar 4.4 Setting webcam *realtime* dan *non realtime*.

Proses pengujian non *realtime* memerlukan suatu data yang harus diolah, data tersebut adalah video hasil sampling. Untuk menampilkan video tersebut file yang akan ditampilkan harus dituliskan beserta format file tersebut. Perbedaan dengan pengujian *realtime* adalah data yang ditampilkan pada webcam *realtime*, sehingga hanya perlu mengaktifkan webcam dengan menulis 0 seperti pada gambar 4.4.

```
while(1):
    ret, frame = cap.read()
    resize = cv2.resize(frame,(int(frame.shape[1]/2),int(frame.shape[0]/2)))
    crop=resize[90:320,0:320]
    gray = cv2.cvtColor(crop, cv2.COLOR_BGR2GRAY)
```

Gambar 4.5 *Preprocessing* citra.

Proses looping program terjadi setelah *while* seperti pada gambar 4.5, program akan berjalan terus menerus hingga perintah selanjutnya. Tahap ini adalah proses *preprocessing*,

citra yang akan diolah diresize terlebih dahulu supaya proses yang dilakukan tidak terlalu berat. Hasil resize akan dicrop pada *region* yang akan dipilih menggunakan koordinat. Hasil citra *cropping* tersebut akan diubah menjadi citra *grayscale*.

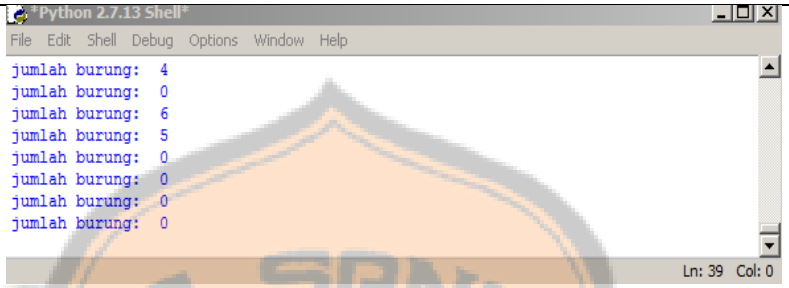
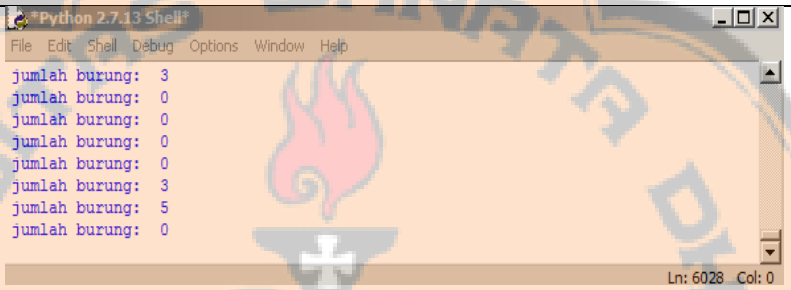
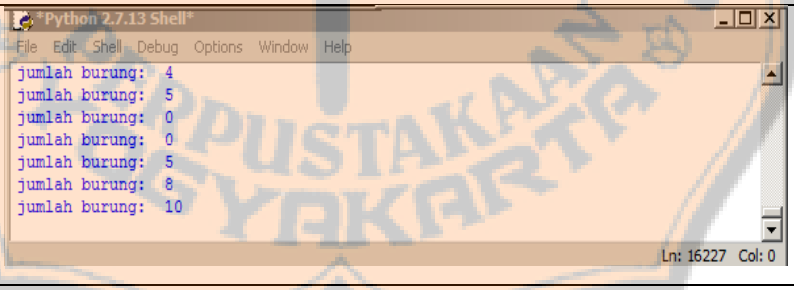
```
if count == 0:
    background = gray
    count += 1
else:
##################avrage background#########################
    background = ( gray * α )+ background * (1 - α)
    foreground = cv2.absdiff(background.astype(np.uint8), gray)
    ret,th = cv2.threshold(foreground,15,255,cv2.THRESH_BINARY)
```

Gambar 4.6 Algoritma *background subtraction.*

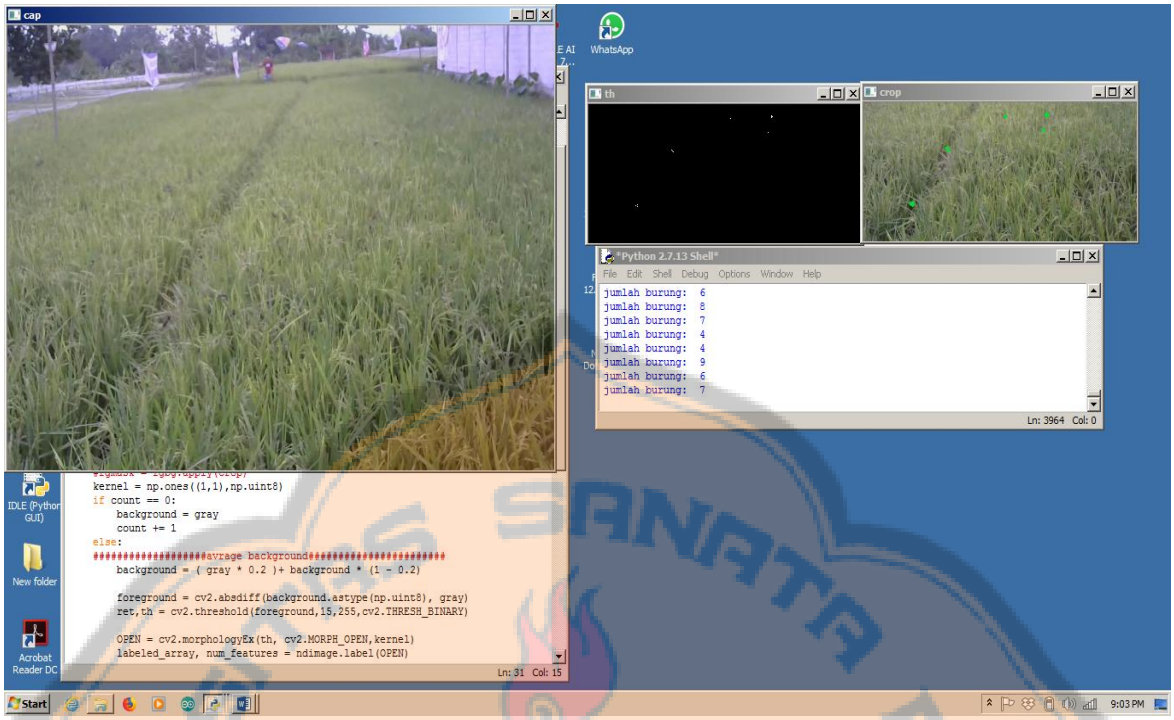
Setelah tahap *preprocessing* citra akan dilanjutkan pada *background subtraction* seperti pada gambar 4.6. Tahap ini merupakan metode yang digunakan untuk melihat pergerakan burung dilahan sawah. Frame pertama akan menjadi *background* yang akan dikurangi dengan frame selanjutnya untuk mendapatkan *foreground*. Proses pengurangan ini dilanjutkan terus menerus hingga sistem berhenti. Dengan metode running averaging frame akan dikalikan dengan α sebagai learning rate. Learning rate ditentukan melalui *trial and error* pada pengujian *non realtime*. Gambar 4.7 menjadi acuan pengujian untuk nilai *learning rate, threshold , labeling* dan kontur.

Gambar 4.7 Citra acuan pengujian.

Tabel 4.1. Hasil percobaan nilai *learning rate*

No	Nilai learning rate	Tampilan Output	Jumlah burung
1	0.6		0
2	0.5		3
3	0.2		7
4	0.1		10

Dilakukan beberapa percobaan penentuan nilai learning rate seperti tabel 4.1 menggunakan nilai learning rate 0.6, 0.5, 0,2, 0,1. Pada proses *background subtraction* dengan nilai learning rate 0.6 menghasilkan nilai output 0. Pergerakan objek tidak terdeteksi, sehingga menyebabkan sistem tidak dapat mendeteksi objek yang bergerak. Percobaan dengan nilai learning rate 0.5 menghasilkan nilai output 3. Sistem tidak mengalami perubahan yang signifikan terhadap pergerakan objek.



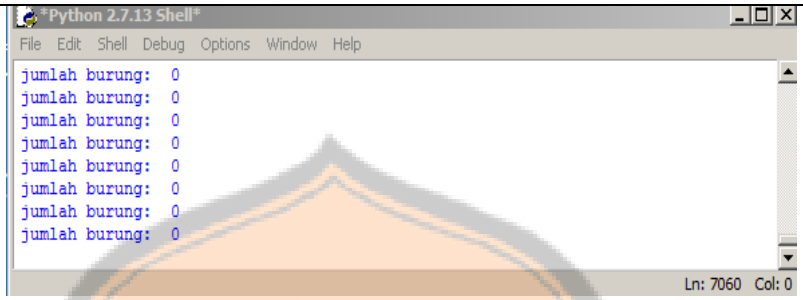
Gambar 4.8 Pengujian nilai learning rate 0.2.

Percobaan ketiga dilakukan dengan nilai learning rate 0.2 seperti gambar 4.8. Hasil output pada frame terlihat adanya pergerakan, sehingga dapat mendeteksi objek yang bergerak. Percobaan keempat dilakukan dengan menggunakan nilai learning rate 0.1. Nilai learning rate yang semakin kecil akan meningkatkan sensitifitas deteksi pada pergerakan objek. Angin yang disebabkan kepakan sayap burung yang menggerakan padidapat terdeteksi sebagai objek. Dari pengujian *trial and error* learning rate mendapatkan nilai ideal 0.2.

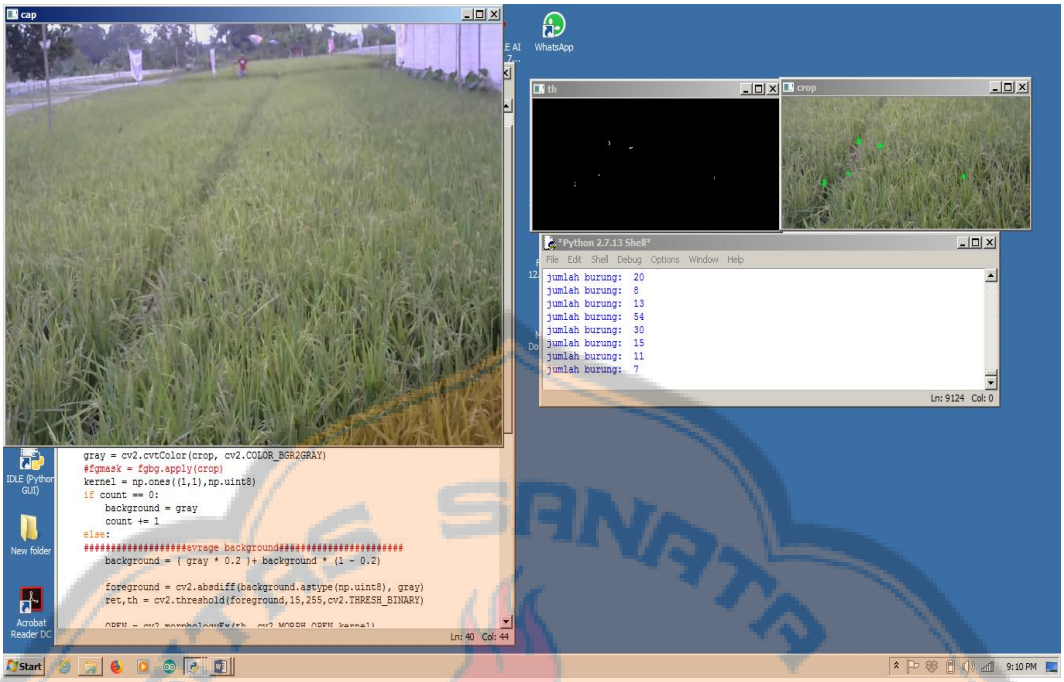
Setelah itu dengan fungsi pada OpenCV frame akan dikurangi sehingga mendapatkan *foreground*. Hasil pengurangan masih berupa citra *grayscale* 8 bit, sehingga untuk memisahkan *background* dengan *foreground* dilakukan pengambangan dengan *threshold* dengan nilai rendah yang ditentukan oleh user.

Dilakukan beberapa percobaan dengan nilai *threshold* berbeda pada pengujian *non realtime*. Percobaan dilakukan dengan nilai *threshold* 100, 50, 15, 7 seperti tabel 4.2. Nilai *threshold* 100 menghasilkan nilai output 0 yang menandakan tidak ada pergerakan objek pada lahan sawah. Nilai *threshold* yang terlalu tinggi menyebabkan banyak objek yang bergerak bernilai 0 atau hitam.

Tabel 4.2. Hasil percobaan nilai *threshold*.

No	Nilai *threshold*	Tampilan output	Jumlah burung
1	100		0
2	50		4
3	15		7
4	7		19

Percobaan kedua dengan nilai *threshold* 50 menghasilkan output 4. Hasil output yang dihasilkan tidak terlalu signifikan. Nilai *threshold* 50 mengolah data tidak terlalu sensitive karena masih terlalu tinggi. Percobaan nilai threshold ini menggunakan acuan citra pada gambar 4.7.



Gambar 4.9 Pengujian nilai *threshold* 15.

Percobaan ketiga dengan nilai *threshold* 15 menunjukan pergerakan objek. Hal ini menunjukan hasil pengurangan pada *background subtraction* sangat kecil, sehingga nilai *threshold* yang ditetapkan harus rendah. Percobaan keempat dengan nilai *threshold* 7 menghasilkan output yang sangat sensitif terhadap pergerakan sehingga akan banyak objek terdeteksi apabila menggunakan nilai *threshold* yang rendah. Berdasarkan keempat percobaan nilai *threshold trial and error* dapat disimpulkan nilai *threshold* yang ideal bernilai 15 dan hasil citra dapat dilihat pada gambar 4.9.

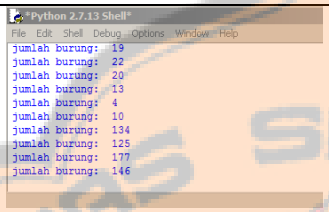
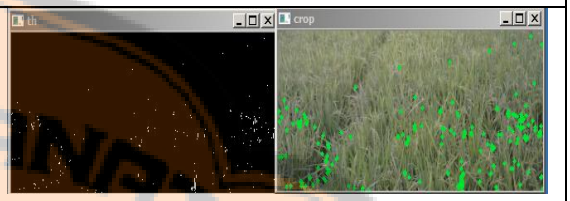
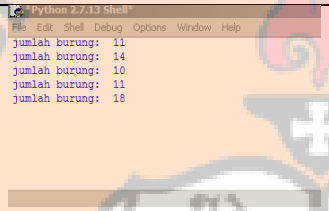
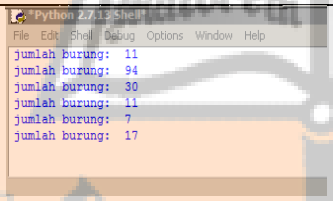
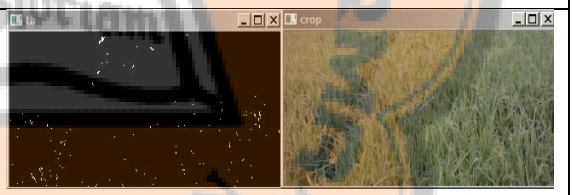
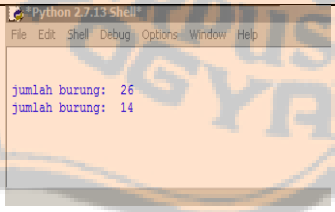
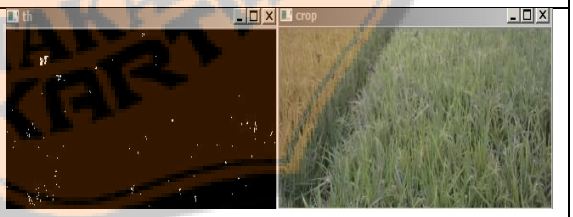
```
kernel = np.ones((1,1),np.uint8)
OPEN = cv2.morphologyEx(th, cv2.MORPH_OPEN,kernel)
labeled_array, num_features = ndimage.label(OPEN)
if num_features < 180:
        kontur()
```

Gambar 4.10 Operasi morfologi citra.

Keluaran pada *background subtraction* adalah citra biner. Untuk mengurangi noise yang disebabkan oleh pergerakan padi, maka dilakukan operasi morfologi seperti gambar 4.10. Objek yang lolos dari proses morfologi akan dihitung menggunakan *labeling*. Proses *labeling* ini untuk penentuan suatu citra akan dilakukan deteksi burung atau tidak. Pada implementasinya, sawah memiliki gangguan dari luar yang dapat menyebabkan *error* pada

perhitungan burung. Gangguan tersebut adalah angin, untuk mengatasi gangguan angin pada lahan sawah digunakan proses *labeling* untuk meminimalisir tingkat kesalahan dalam deteksi burung. Penentuan nilai *labeling* dilakukan dengan *trial and error*.

Tabel 4.3. Hasil percobaan nilai *labeling*.

No	Nilai *labeling*	Tampilan output	Tampilan output citra
1	250		
2	180		
3	110		
4	40		

Percobaan pertama dilakukan dengan menggunakan nilai *labeling* 250 seperti gambar 4.10. Nilai *labeling* yang telalu tinggi menyebabkan sawah yang terganggu angin terdeteksi sebagai burung. Hasil perhitungan burung memiliki *error* yang sangat tinggi. Percobaan kedua dilakukan dengan nilai *labeling* 180 . Gambar pada tabel 4.3 tersebut menunjukan keadaan sawah yang terganggu akibat angin.

Apabila nilai *labeling* lebih dari 180 maka dilanjutkan proses deteksi burung menggunakan kontur. Nilai *labeling* yang lebih dari 179 tidak akan dilakukan deteksi burung. Percobaan ketiga dengan nilai 110 saat keadaan sawah tidak terlalu terganggu angin namun proses tidak dilanjutkan pada tahap kontur. Dilakukan beberapa percobaan nilai *labeling*, percobaan keempat dilakukan dengan nilai 40.

Hasil perbandingan output dari proses dapat dilihat pada gambar tabel 4.3. Nilai *labeling* yang terlalu kecil akan mengakibatkan sistem tidak mendeteksi burung, sehingga sistem tidak dapat bergerak ketika ada burung. Berdasarkan percobaan *trial and error*, dapat disimpulkan nilai 180 menjadi nilai yang ideal pada pengujian *realtime*. Perhitungan burung dilakukan menggunakan deteksi kontur. Luas area objek yang terdeteksi akan dihitung luasan objek, apabila memenuhi syarat maka akan dilanjutkan dengan menggambar keliling dari objek terdeteksi tersebut. Jumlah kontur pada citra akan dihitung dan hasil tersebut menjadi keluaran jumlah burung pada lahan sawah.

```
def kontur():
im,contours,hierarchy=cv2.findContours(OPEN,cv2.RETR_TREE,cv2.CHAIN_APPROX_
NONE)
  totalContours = 0
  for contour in contours:
    area = cv2.contourArea(contour)
    for i in xrange(len(contours)):
      if hierarchy[0][i][1] == ROOT_NODE and area >= 1:
        cv2.drawContours(crop, contour, -1, (50, 225, 0), 3)
        totalContours += 1
  print 'jumlah burung: ', totalContours
```

Gambar 4.11 Program kontur pada citra.

Jumlah burung digunakan untuk membuat keputusan apakah aktuator akan bergerak atau diam seperti pada gambar 4.11. Burung yang masuk pada nilai batas yang ditetapkan akan menggerakan aktuator. Pada bagian ini, nilai batas yang ditetapkan diperoleh dari pengujian *non realtime* yang dilakukan dengan melihat nilai kontur minimal pada burung yang terdeteksi.

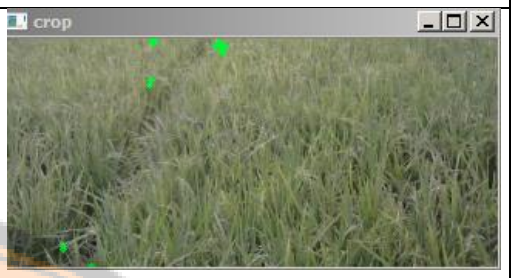
```
  if totalContours >=8 and totalContours<=40:
######Statement################
        GPIO.output(in1,GPIO.HIGH)
        GPIO.output(in2,GPIO.HIGH)
        GPIO.output(in3,GPIO.HIGH)
        GPIO.output(in4,GPIO.HIGH)
        sleep(1)
######Set to Low###############
        GPIO.output(in1,GPIO.LOW)
        GPIO.output(in2,GPIO.LOW)
        GPIO.output(in3,GPIO.LOW)
        GPIO.output(in4,GPIO.LOW)
        sleep(0.3)

   elif totalContours >=41:
        print (“kontur 0”)
   return;
```

Gambar 4.12 Pengambilan keputusan aktuator.

Percobaan nilai batas kontur dilakukan dengan nilai kontur yang bervariasi, percobaan pertama dengan nilai 4. Batas kontur dengan nilai rendah mengakibatkan aktuator bergerak dengan mudah. Pada gambar 4.12 lahan sawah tidak terdapat burung, hanya mendapat gangguan dari angin. Nilai kontur yang sangat kecil dapat menyebabkan sistem yang sensitif dalam menggerakan aktuator. Aktuator akan bergerak apabila jumlah kontur bernilai minimal 4. Ganguan angin ringan dapat menyebabkan deteksi kontur bernilai 4, sehingga nilai yang rendah tidak dapat diterapkan.

Percobaan kedua dilakukan dengan nilai batas kontur 16. Pada lahan sawah terdeteksi 8 burung, namun aktuator tidak bergerak, karena nilai kontur tidak mencapai batas. Aktuator akan bergerak apabila ada pergerakan objek lebih banyak lagi. Percobaan ketiga dilakukan dengan nilai batas kontur 8 pada lahan sawah terdapat 2 burung, aktuator bergerak ketika jumlah kontur mencapai angka 8.

Tabel 4.4. Hasil percobaan nilai kontur.

No	Nilai kontur	Tampilan output	Tampilan citra
1	4	*Python 2.7.13 Shell* File Edit Shell Debug Options Window Help jumlah burung: 0 jumlah burung: 40 motor on jumlah burung: 25 motor on jumlah burung: 5 motor on jumlah burung: 10 motor on jumlah burung: 4 motor on	
2	16	*Python 2.7.13 Shell* File Edit Shell Debug Options Window Help jumlah burung: 0 jumlah burung: 4 jumlah burung: 5 jumlah burung: 7 jumlah burung: 8 jumlah burung: 3 jumlah burung: 4 jumlah burung: 0 jumlah burung: 3 jumlah burung: 4 jumlah burung: 4	
3	8	*Python 2.7.13 Shell* File Edit Shell Debug Options Window Help jumlah burung: 4 jumlah burung: 3 jumlah burung: 3 jumlah burung: 0 jumlah burung: 3 jumlah burung: 0 jumlah burung: 4 jumlah burung: 5 jumlah burung: 7 jumlah burung: 8 motor on	

. Batas nilai kontur ditetapkan 8 untuk membuat akatuator bergerak. Nilai batas tersebut diterapkan pada pengujian *realtime* dan digunakan untuk mengusir burung pada saat lahan sawah terdeteksi burung dengan jumlah seminimal mungkin.

```
cv2.imshow('th',OPEN)
cv2.imshow('crop',crop)
cv2.imshow('cap',frame)
```

Gambar 4.13 Menampilkan citra pada frame.

Proses terakhir adalah menampilkan frame pada monitor. Untuk melihat frame menggunakan perintah imshow seperti gambar 4.13, diikuti dengan nama frame dan variable frame pada program. Menapilkan frame digunakan untuk melihat citra sebelum dan sesudah diproses.

Untuk menjalankan program pada Raspberry Pi terlebih dahulu menuliskan perintah pada terminal untuk masuk pada package OpenCV. Dengan menuliskan source ~/.profile kemudian workon cv. Setelah masuk pada package OpenCV, program dalam folder dijalankan dengan perintah python file.py

4.2. Bentuk rangka pengusir hama

Pada implementasinya terjadi beberapa perubahan rancangan pada pengusir hama saat pengujian realtime. Gambar 4.14 adalah rancangan yang dibuat sebelum melakukan pengujian.

Gambar 4.14 Rancangan awal pengujian realtime.

Dengan posisi tali terkangkap webcam dapat menyebabkan kesalahan perhitungan pada sistem. Tali yang bergerak akan mengakibatkan sistem mendeteksi burung dengan kontur. Rancangan awal tidak dapat diterapkan pada pengujian realtime menggunakan konfigurasi seperti gambar 4.14.

Gambar 4.15 Pemasangan alat saat pengujian realtime.

Pemasangan prototype alat dapat dilakukan seperti gambar 4.21 dengan tujuan tali aktuator tidak terdeteksi webcam. Selain itu dapat dilakukan seperti gambar 4.22 dengan mengatur posisi webcam diluar jangkauan tali aktuator.

Gambar 4.16 Pemasangan alat dengan cara kedua.

Pengujian ini dilakukan untuk melihat tingkat keberhasilan burung yang terdeteksi pada lahan sawah. Pada implementasinya untuk menerapkan pengusir hama burung pada lahan sawah tali aktuator diikat pada ujung sawah yang terdapat tali pengusir yang saling terhubung. Pemasangan pada lahan sawah dilakukan seperti gambar 4.23 dengan posisi webcam diluar jangkauan tali.

Gambar 4.17 Pemasangan tali aktuator pada lahan sawah.

Pada rancangan awal aktuator yang digunakan adalah dua motor dc. Untuk implementasinya aktuator yang digunakan dirubah menjadi satu motor DC dengan tambahan satu buzzer. Buzzer diletakan pada tengah sawah dengan tujuan untuk membantu dalam pengusiran burung pada lahan sawah.

4.3. Perubahan Perancangan Software

Rancangan awal software pada penelitian ini menggunakan metode binerisasi untuk mendeteksi burung, namun pada implementasi program hasil keluaran jumlah burung tidak dapat terlihat. Metode binerisasi ini sangat sensitif terhadap perubahan cahaya karena menggunakan *threshold* sebagai acuan nilai citra, sehingga hasil deteksi burung menjadi tidak terlihat. Setelah melakukan beberapa percobaan dengan metode lain, deteksi burung pada penelitian ini dapat menggunakan metode *background subtraction*.

Metode *background subtraction* bekerja dengan melihat nilai perbedaan frame. Perbedaan frame ini dapat terjadi apabila ada sebuah pergerakan dari sebuah objek, sehingga objek bergerak tersebut dapat menandakan burung di area sawah. Perubahan perancangan software dilakukan pengujian non realtime untuk mengamati perbedaan pada setiap metode.

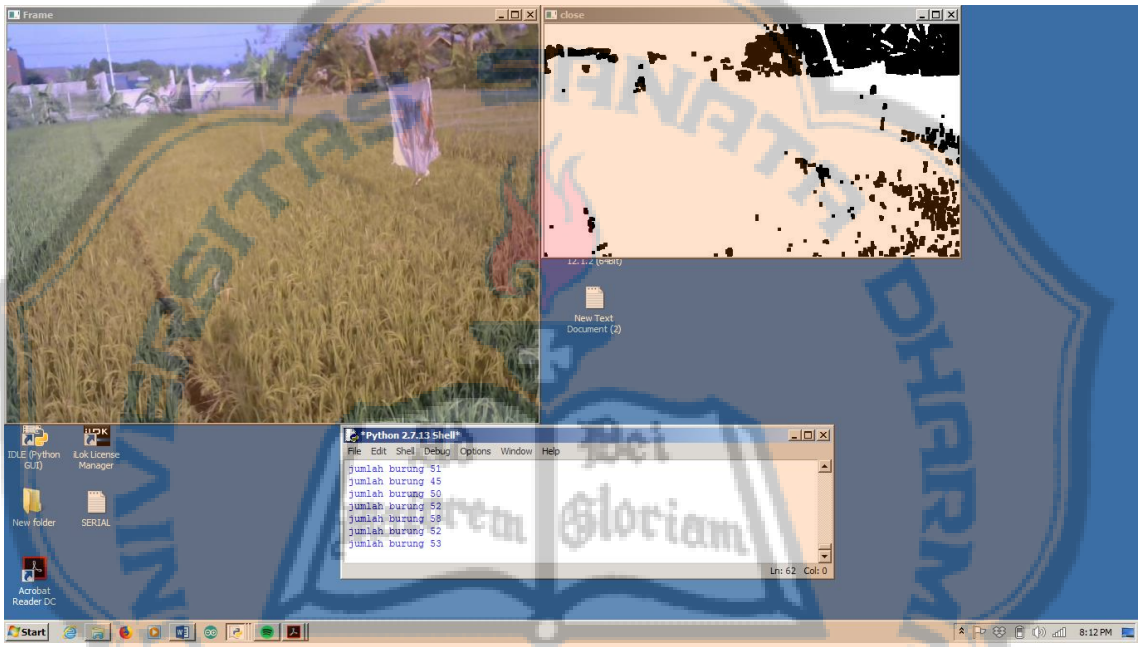
4.4. Pengujian Non Realtime

Pengujian non realtime bertujuan untuk mengetahui kinerja sistem dalam menghitung jumlah burung pada lahan sawah yang tertangkap webcam. Jumlah burung yang terdeteksi akan digunakan untuk pengambilan keputusan untuk menggerakan motor dc pada saat pengujian realtime.Pengujian ini menggunakan data lahan sawah pada tiga kondisi

sawah yang berbeda, yaitu ada saat 3 minggu sebelum panen , 2 minggu sebelum panen dan 1 minggu sebelum panen.

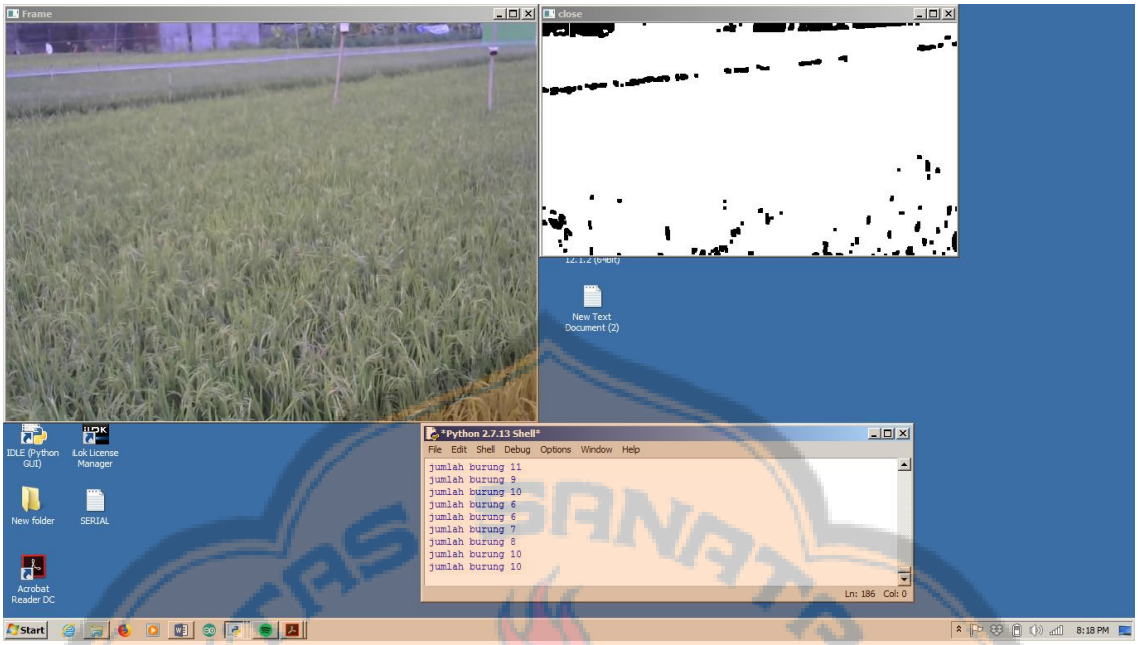
4.4.1. Hasil pengujian metode binerisasi

Pengujian binerisasi dilakukan beberapa kali untuk melihat kemungkinan *error* pada metode. Perhitungan jumlah objek dengan metode binerisasi terlihat seperti gambar 4.18. Lahan sawah terlihat kondisi tidak ada burung, namun citra akan terkena *threshold* sehingga menyebabkan nilai yang melebihi *threshold* akan terdeteksi sebagai burung.



Gambar 4.18 Perhitungan jumlah burung dengan binerisasi sampel satu.

Citra biner yang berwarna hitam adalah citra yang akan dihitung menggunakan *labeling*. Hasil citra *labeling* tersebut adalah gambaran dari deteksi burung pada lahan sawah. Python shell menunjukan jumlah burung pada lahan sawah berjumlah antara 45 hingga 58. Metode ini hanya menggunakan nilai *threshold* untuk proses deteksi burung, sehingga menyebabkan *error* pada perhitungan burung tinggi. Metode ini memiliki kelebihan pada kecepatan proses pengolahan citra. Output proses pengolahan citra lebih cepat daripada menggunakan metode *background subtraction*.



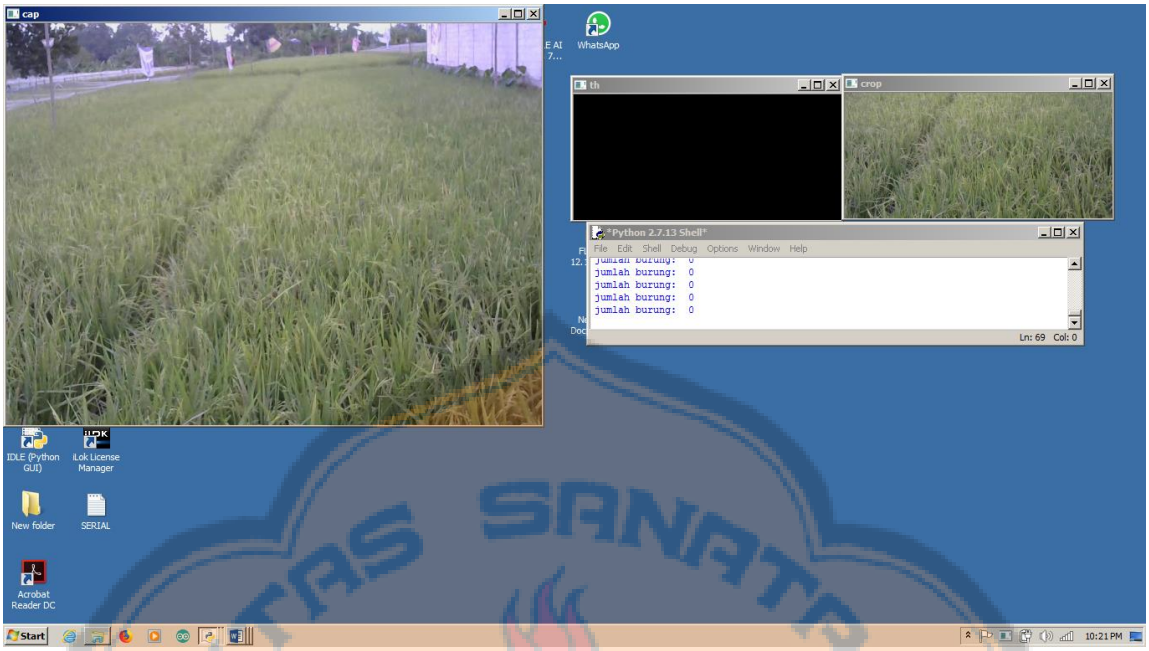
Gambar 4.19 Perhitungan jumlah burung dengan binerisasi sampel dua.

Pengujian dilakukan dengan binerisasi pada tempat yang berbeda terlihat pada gambar 4.19 masih memiliki *error* yang tinggi pada perhitungan jumlah burung. Pencahayaan pada lahan sawah sangat berpengaruh terhadap nilai *threshold*. Metode binerisasi ini tidak dapat diterapkan pada sistem karena dari hasil pengujian non realtime tidak dapat menentukan jumlah burung dengan baik.

4.4.2. Hasil pengujian metode *background subtraction*

Pengujian *background subtraction* dilakukan pada tiga kondisi yang berbeda. Gambar 4.20 menunjukan pengujian pada kondisi 1 minggu sebelum panen. Gambar 4.20 terlihat bahwa frame cap adalah citra hasil yang terekam webcam dengan ukuran piksel 640x480.

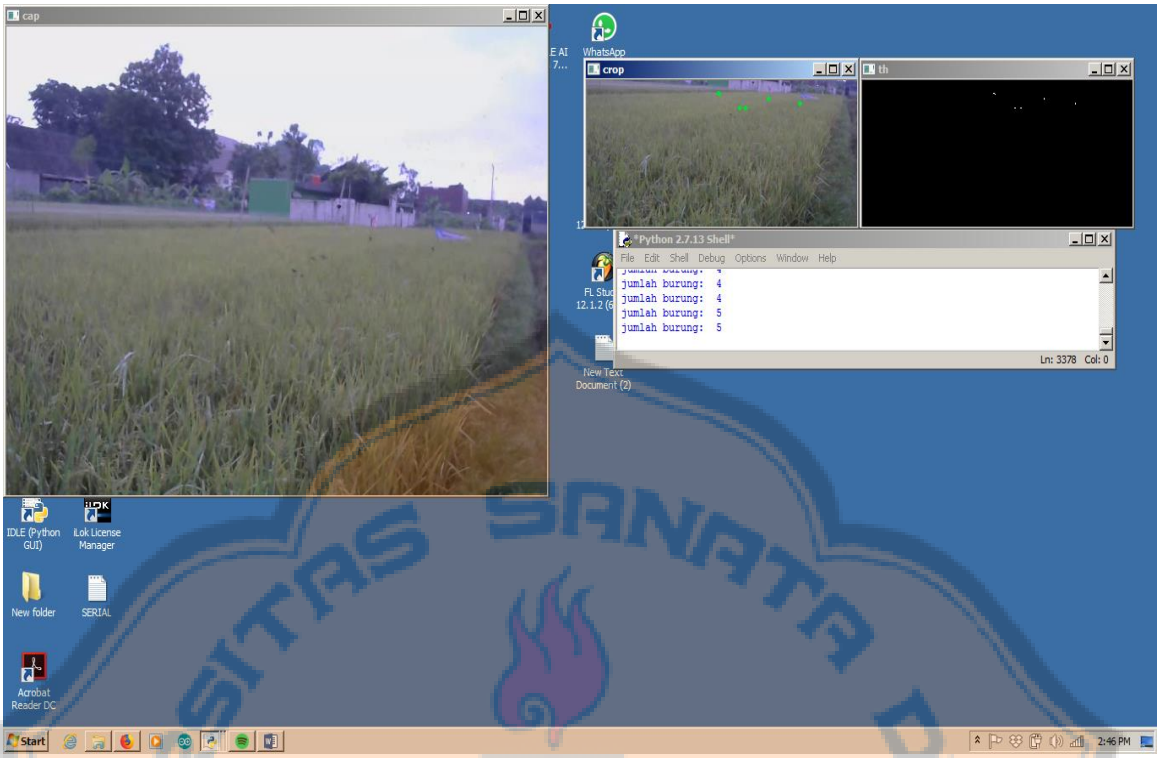
Frame crop adalah hasil frame cap yang telah dilakukan proses resize dan *cropping*. Proses resize digunakan untuk mengubah skala citra mencadi ukuran piksel yang baru. Ukuran piksel citra dirubah menjadi setengah dari piksel frame cap, sehingga ukuran piksel menjadi 320x240. Frame dengan piksel baru tersebut akan di crop agar mendapatkan ukuran yang diinginkan. Proses menentukan ukuran *cropping* menggunakan titik koordinat piksel.



Gambar 4.20 Kondisi sawah 1 minggu sebelum panen.

Frame th adalah hasil citra proses *background subtraction* yang akan dideteksi jumlah burung dan tertampil pada python shell. *Background subtraction* dapat melihat objek bergerak pada lahan sawah, objek bergerak tersebut akan terlihat sebagai citra bernilai 1 atau berwarna putih. Objek berwarna putih tersebut akan dilakukan proses *labeling*, yang digunakan untuk melihat banyaknya objek citra berwarna putih. Citra berwarna putih yang melebihi batas dapat disimpulkan bahwa keadaan sawah sedang mengalami gangguan yang diakibatkan oleh angin. Batas pada *labeling* ditentukan melalui *trial and error* seperti dijelaskan pada implementasi program. Hasil dari proses *labeling* menentukan sebuah citra akan dilakukan deteksi burung atau tidak. Hasil objek terdeteksi yang melebihi batas tidak dilakukan deteksi burung, sementara hasil yang kurang dari batas akan dilakukan deteksi burung.

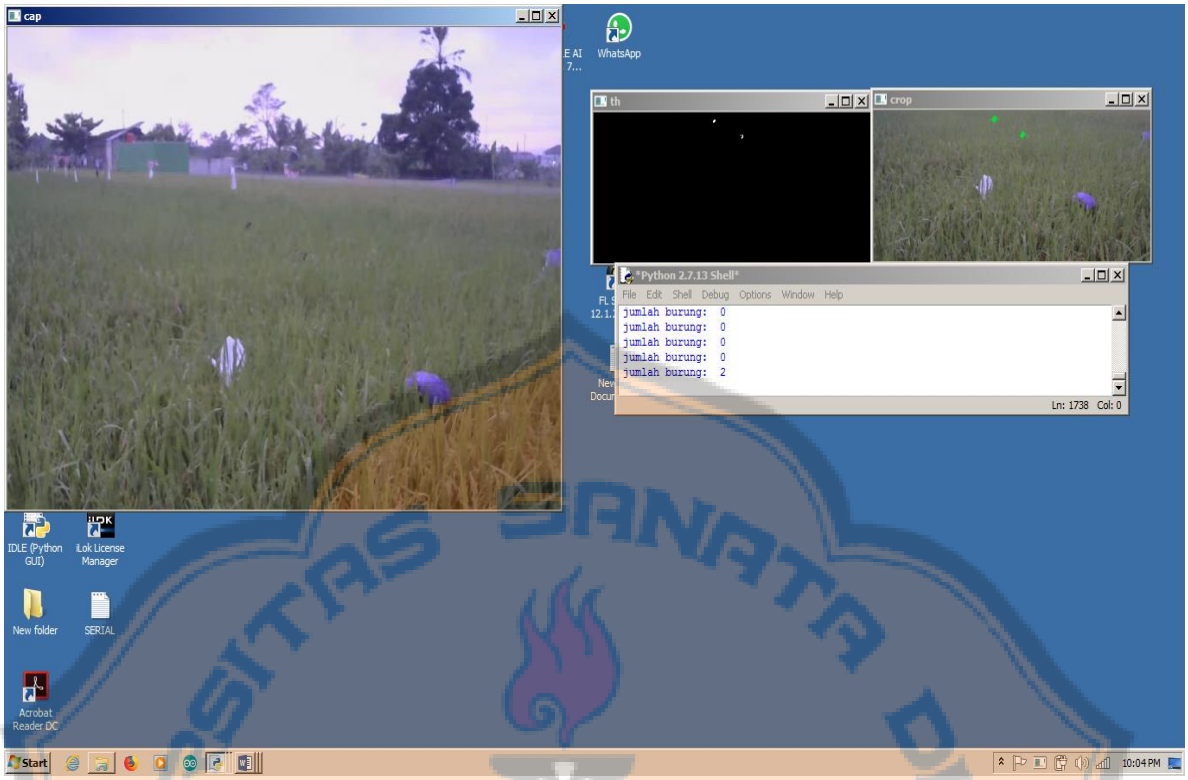
Proses deteksi burung akan dilakukan menggunakan kontur, dari burung yang terdeteksi tersebut dapat dihitung jumlahnya. Citra berwarna putih tersebut akan dilihat luasan areanya, dalam hal ini luasan burung yang terlihat lebih besar apabila keadaan sawah tenang. Citra yang memiliki luasan di atas batas akan dilakukan deteksi kontur. Citra terkontur akan dihitung jumlahnya, sehingga jumlah burung dapat dilihat melalui output dari python shell. Terlihat pada gambar 4.20 tidak ada burung pada lahan sawah sehingga python shell akan menampilkan jumlah burung 0. Jumlah burung 0 menunjunkan keadaan sawah tidak terdeteksi burung.



Gambar 4.21 Kondisi sawah 2 minggu sebelum panen.

Pengujian *background subtraction* dilakukan pada beberapa kondisi sawah untuk melihat pengaruh warna terhadap perhitungan burung. Gambar 4.22 terlihat burung pada lahan sawah dapat terdeteksi oleh webcam. Terdapat 5 burung yang tertangkap webcam, burung tersebut dihitung menggunakan deteksi kontur dan akan digambarkan bentuk konturnya pada frame crop. Hasil dari deteksi burung ditampikan pada python shell. Deteksi burung tersebut dilakukan secara cepat pada sistem sehingga akan menghitung secara terus menerus dan tertampil pada python shell.

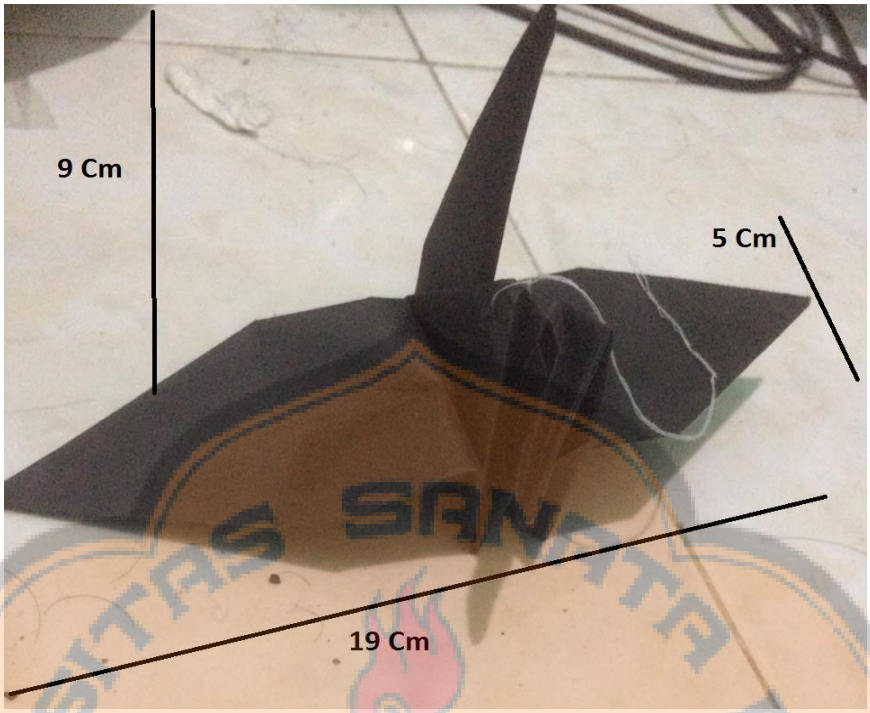
Lahan sawah dengan kondisi 3 minggu sebelum panen sistem dapat mendeteksi burung dengan baik. Terlihat pada gambar 4.22 terdapat 2 burung dilahan sawah dan sistem mampu mendeteksi keberadaan burung sesuai dengan jumlah sesungguhnya. Burung dapat terdeteksi dengan jelas apabila burung tersebut bergerak.



Gambar 4.22 Kondisi sawah 3 minggu sebelum panen.

4.5. Pengujian realtime

Sistem pengusir hama dilakukan pengujian secara realtime setelah melakukan analisis pada pengujian non realtime. Pengujian realtime objek burung digantikan menggunakan origami burung kertas berwana hitam. Penggantian objek burung dikarenakan kesulitan dalam mendatangkan burung pada lahan sawah. Origami burung menggunakan kertas berwarna hitam dengan tujuan ukuran dan warna yang menyerupai hama burung pada lahan sawah. Dimensi burung yang digunakan dalam pengujian seperti pada gambar 4.23 dengan ukuran panjang 19 cm dan lebar 5 cm dan tinggi 9 cm. Origami tersebut akan di ikat pada tongkat menggunakan benang. Tongkat digunakan saat pengujian agar webcam hanya menangkap burung pada area sawah.



Gambar 4.23 Dimensi burung yang digunakan.

Pengujian realtime hanya melihat kinerja sistem saat burung tertangkap webcam. Perhitungan burung tidak dilakukan secara program atau dengan kata lain hanya dilakukan secara manual untuk menyesuaikan mengetahui tingkat keberhasilan sistem. Hasil deteksi burung pada program dengan perhitungan manual dapat mengalami perbedaan, sehingga untuk menentukan keberhasilan sistem dilakukan perhitungan manual.

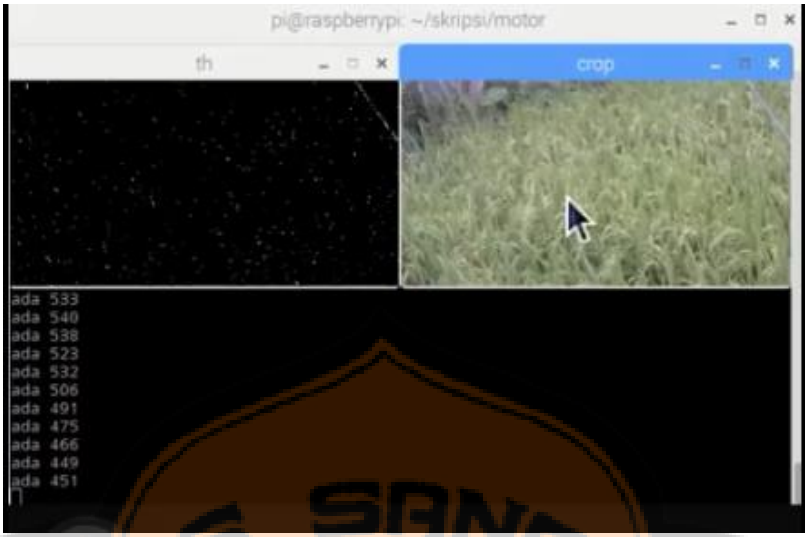
Gambar 4.24 Kondisi sistem saat lahan sawah tidak ada burung.

Lahan sawah yang tidak ada burung aktuator akan diam seperti pada gambar 4.24. Aktuator tidak bergerak karena pada lahan sawah objek yang tidak terdeteksi sebagai burung kurang dari atau sama dengan 8. Kondisi lahan sawah yang terdeteksi ada burung aktuator akan bergerak seperti pada gambar 4.25. Pengujian dengan objek burung dilakukan pada jarak ± 3 meter dari webcam seperti pada gambar 4.25.

Gambar 4.25 Pengujian realtime saat lahan sawah ada burung.

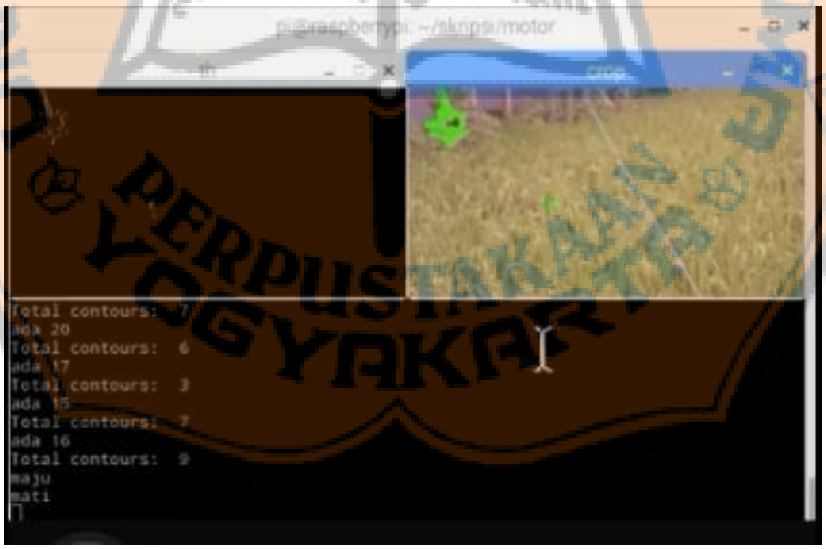
Aktuator bergerak ketika objek terdeteksi sebagai burung pada lahan sawah lebih 7. Batas untuk menentukan aktuator bergerak dilakukan saat pengujian non realtime dengan *trial and error*. Batas yang terlalu kecil mengakibatkan kondisi aktuator bergerak saat tidak ada burung sangat tinggi sehingga menyebabkan *error*. Keadaan ini disebabkan oleh faktor angin yang terjadi sewaktu waktu pada lahan sawah.

Meminimalisir aktuator yang bergerak apabila kondisi lawah terdapat objek yang bukan burung dapat dilakukan penambahan pada program pengambilan keputusan pertama dengan proses *labeling*. Nilai *labeling* yang melebihi batas mengakibatkan aktuator tidak akan bergerak apabila kondisi sawah terjadi pergerakan akibat angin. Nilai *labeling* yang melebihi batas akan menghiraukan perhitungan kontur seperti pada gambar 4.26, sehingga keadaan sawah dianggap sedang mengalami gangguan angin .



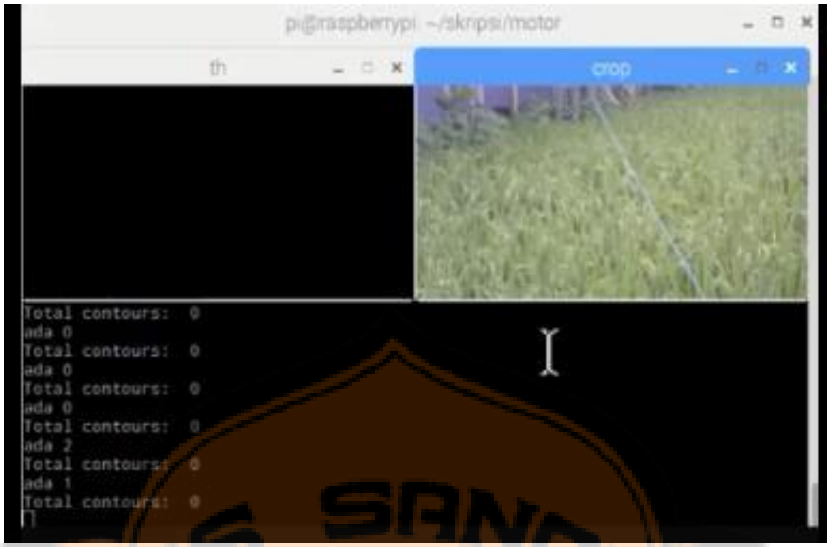
Gambar 4.26 Kondisi sawah saat mengalami gangguan angin.

Nilai *labeling* kurang dari batas maka sistem akan melanjutkan proses perhitungan burung dengan kontur seperti pada gambar 4.26. Keadaan sawah masih mengalami sedikit gangguan angin, namun sistem dapat masih dapat melakukan perhitungan burung. Sehingga burung dapat terdeteksi, selanjutnya apabila proses deteksi secara program menunjukan jumlah lebih dari 7 maka aktuator akan bergerak seperti gambar 4.27. Itu sebabnya untuk menentukan keberhasilan sistem perhitungan objek dilakukan secara manual.



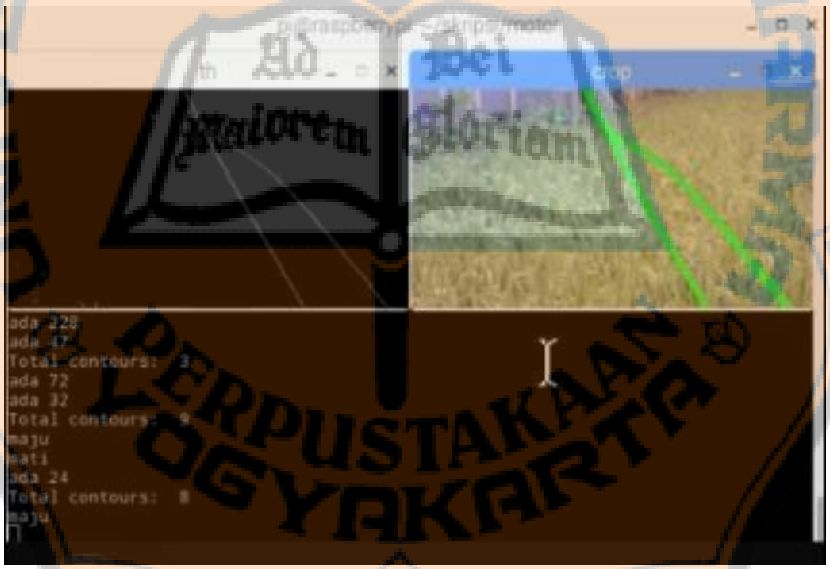
Gambar 4.27 Aktuator bergerak saat kontur terdeteksi 9.

Sistem dapat bekerja sangat baik apabila keadaan sawah tidak mengalami gangguan seperti pada gambar 4.28. Keadaan lahan sawah yang kosong dan tidak ada gangguan dari angin dapat membuat sistem bekerja dengan baik, perhitungan burung dapat sesuai dengan kondisi realtime.



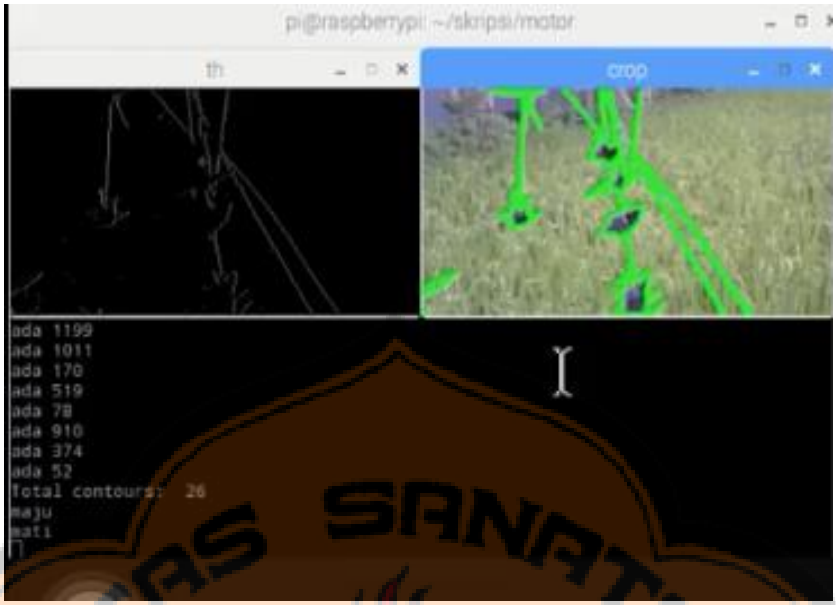
Gambar 4.28 Kondisi lahan sawah tidak ada burung.

Pemasangan tali pada aktuator tidak dapat dilakukan seperti gambar 4.29. Pemasangan tali yang terlihat pada lahan sawah dapat menyebabkan *error* pada perhitungan burung. Pergerakan tali dapat mempengaruhi perhitungan burung.



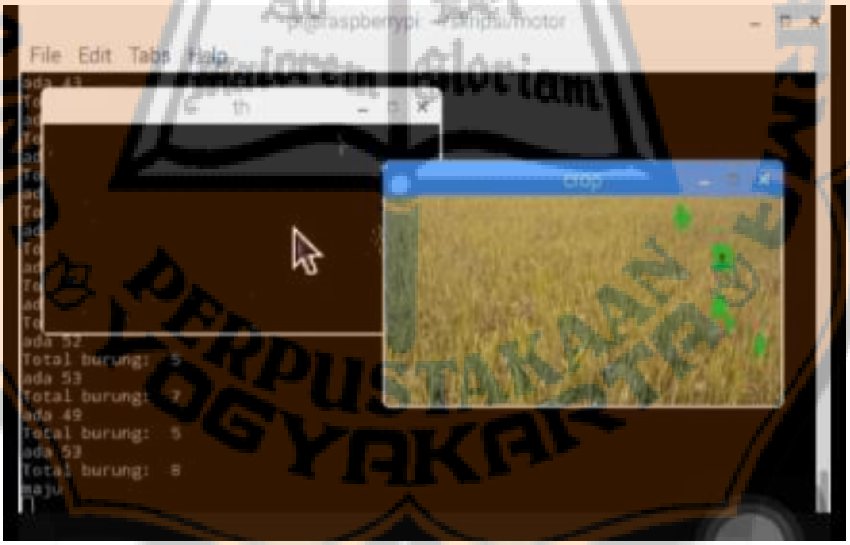
Gambar 4.29 Kondisi lahan sawah dengan tali saat bergerak.

Perhitungan burung menjadi lebih banyak apabila keadaan lahan sawah terdapat burung dan tali aktuator bergerak terlihat seperti gambar 4.30.



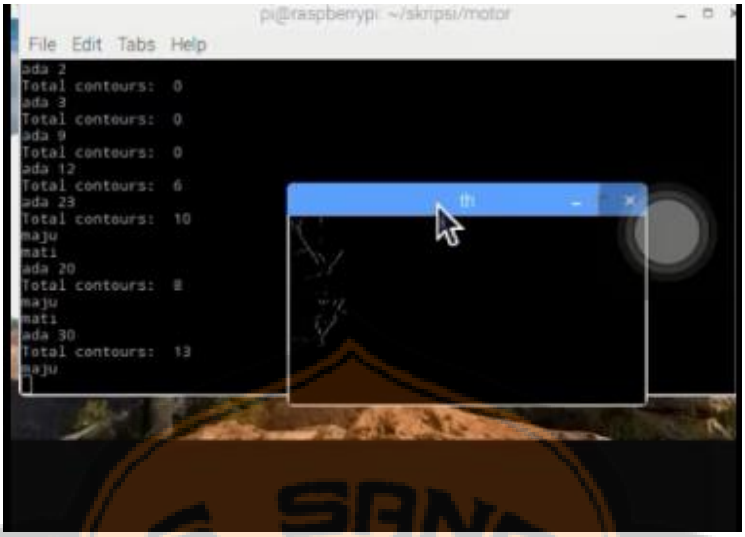
Gambar 4.30 Kondisi lahan sawah dengan dengan tali dan burung.

Hasil pengujian yang telah dilakukan, penulis dapat menyimpulkan pengujian realtime dapat dilakukan dengan meletakan tali aktuator pada sisi sawah. Posisi webcam tersebut diarahkan pada area sawah akan diluar lingkup tali aktuator.



Gambar 4.31 Aktuator bergerak saat kontur terdeteksi 8.

Pengujian realtime dapat dilakukan seperti gambar 4.31, dengan posisi tali aktuator diluar lingkup webcam. Cara ini dapat mengurangi *error* pada perhitungan burung pada lahan sawah. Pada gambar 4.31 dilakukan pengujian burung dengan jumlah 3 burung, dengan keadaan yang sedikit terganggu angin masih dapat menghitung burung pada lahan sawah dan menggerakan tali aktuator.

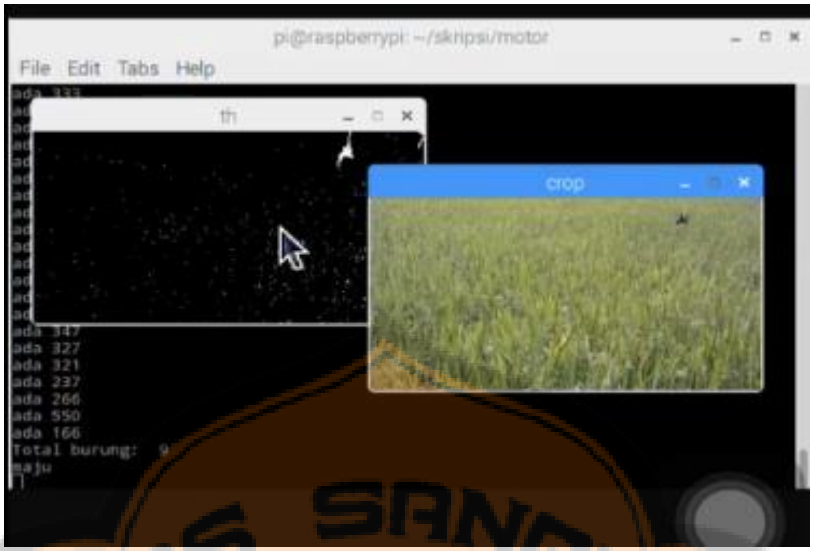


Gambar 4.32 Aktuator bergerak saat kontur terdeteksi 10.

Gambar 4.32 menunjukan aktuator bergerak saat sistem mendeteksi 10 lalu aktuator bergerak dan kembali menghitung jumlah burung. Aktuator bergerak saat sistem mendeteksi 8 kemudian kembali menghitung dan bergerak karena terdeteksi 13.

Gambar 4.33 Aktuator bergerak saat kontur terdeteksi 9.

Percobaan dilakukan dengan jumlah burung 2 seperti pada gambar 4.33. Aktuator bergerak ketika sistem mendeteksi 8. Kemudian saat aktuator diam sistem kembali bergerak ketika sistem mendeteksi 9.

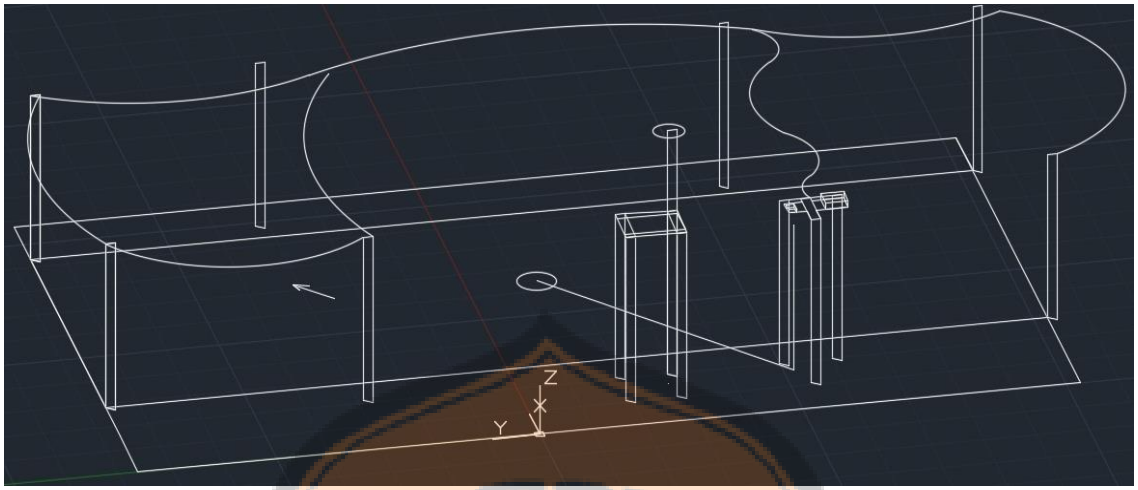


Gambar 4.34 Aktuator bergerak saat kontur terdeteksi 9.

Percobaan dilakukan dengan 1 burung seperti gambar 4.34. Sistem mendeteksi 9 dan aktuator bergerak. Gambar 4.34 terlihat sawah mengalami gangguan angin, ketika nilai *labeling* kurang dari 180 sistem akan mendeteksi burung.

Gambar 4.35 Aktuator bergerak saat kontur terdeteksi 19.

Percobaan dilakukan dengan jumlah burung 3 seperti pada gambar 4.35. Sistem mendeteksi burung ketika nilai *labeling* 170. Jumlah burung yang terdeteksi pada sistem terdapat 19 dan menggerakan aktuator. Sistem kembali menghitung *labeling*, jumlah *labeling* setelah aktuator bergerak 332 dan menyebabkan tidak terjadi perhitungan burung.



Gambar 4.36 Implementasi pada lahan sawah.

Implementasi pada lahan sawah dapat dilakukan dengan cara memasang ujung tali pada sisi ujung sawah seperti gambar 4.36. Tali yang terhubung pada sisi sawah dapat menggerakan tali tali yang terhubung lainya.

Gambar 4.37 Buzzer sebagai aktuator.

Implementasinya untuk mengusir hama burung pada lahan sawah digunakan buzzer sebagai alat tambahan untuk mengusir burung. Tali tali yang terhubung akan ikut bergerak dibantu dengan bunyi buzzer untuk mengusir burung. Buzzer dengan spesifikasi 12V/1A dapat dilihat pada gambar 4.37.

Gambar 4.38 Aplikasi sound meter pada handphone.

Pengukuran dilakukan untuk mengukur tingkat kebisingan menggunakan aplikasi *sound level meter* pada *handphone* seperti pada gambar 4.38, dengan cara mengukur pada jarak 1 meter saat buzzer dibunyikan. Pengukuran dilakukan 3 detik sebanyak 3 kali mendapatkan hasil seperti pada tabel 4.5.

Tabel 4.5. Pengukuran tingkat kebisingan buzzer

No	Pengukuran	Hasil (dB)
1	1	90.6
2	2	97.6
3	3	85.2

Berdasarkan tabel 4.5 setelah dilakukan 3 pengukuran memperoleh hasil 90.6 dB, 97.6 dB dan 85.2 dB.

4.6. Perubahan perancangan hardware

Perancangan hardware pada penelitian mengalami perubahan rangkaian driver dan catu daya. Perancangan awal driver motor menggunakan 1 keluaran output, kemudian diganti modul driver motor dengan keluaran 2 output. Driver motor pada sistem digunakan untuk menggerakan output buzzer dan motor DC. Driver motor pada rangkaian dilakukan pengukuran tegangan saat aktuator bergerak dan saat aktuator diam. Tabel 4.6 menunjukan hasil pengukuran output driver pada sistem.

Tabel 4.6. Tegangan aktuator pada sistem.

No	Output (Volt)	Kondisi aktuator
1	10.5	Bergerak
2	0.16	Diam

Catu daya pada perancangan awal menggunakan IC7805 dengan penguatan arus menggunakan transistor. Catu daya yang digunakan untuk menurunkan tegangan dan menguatkan arus diganti menggunakan modul step down LM2596. Rangkaian stepdown LM2596 sebagai modul catu daya dilakukan pengukuran tegangan pada saat sistem bekerja. Tabel 4.7 menunjukan hasil pengukuran input dan output modul pada sistem. Modul LM2596 dapat dilakukan adjustment tegangan output untuk menyesuaikan kebutuhan sistem.

Tabel 4.7. Tegangan catu daya pada sistem.

No	Input (Volt)	Output (Volt)
1	12.01	5.06

Perubahan catu daya dikarenakan kesalahan rangkaian pada perancangan. Transistor pada rangkaian tidak dapat menguatkan arus yang dibutuhkan pada beban, sehingga menyebabkan transistor meletus.

Catu daya pada sistem pengusir hama ini telah dilakukan uji coba ketahanan umur baterai. Selama sistem hidup diperlukan waktu selama ± 6 jam untuk menghabiskan kapasistas baterai. Baterai yang digunakan pada penelitian ini berkapasitas 2200 mAh 3 cell dengan tegangan 12.6 V. Setiap cell baterai berisi 4.2 V dan cel minimum tiap cel 3.7 V, sehingga baterai tersebut akan habis apabila tegangan 11.1 V Beban yang digunakan pada rangkaian ini adalah Raspberry Pi, motor DC dan buzzer. Tabel 4.8 menunjukan pengujian ketahanan baterai pada sistem.

Tabel 4.8. Pengujian ketahanan catu daya sistem.

No	Lama waktu (Menit)	Tegangan (Volt)	Kondisi sistem (ON/OFF)
1	60	12.6	ON
2	120	12.3	ON
3	180	12	ON
4	240	11.7	ON
5	300	11.3	ON
6	360	11.1	OFF

Rangkaian elektronik keseluruhan dan rangkaian hardware keseluruhan pada sistem ini seperti dapat dilihat pada gambar 4.39 dan gambar 4.40 dengan keterangan komponen sistem dapat dilihat pada tabel 4.9.

Gambar 4.39 Rangkaian elektronik keseluruhan.

Tabel 4.9. Keterangan komponen pada sistem

No	Keterangan
1	Baterai lipo
2	Step down LM2596
3	Driver motor
4	Buzzer
5	Motor DC
6	Raspberry Pi 3

Gambar 4.40 Rangkaian hardware keseluruhan.

4.7. Hasil pengamatan dan pembahasan

Pengujian realtime dilakukan di lahan sawah pada satu lokasi. Hasil pengamatan pada lahan sawah dilakukan dengan jumlah burung bervariasi. Pengujian dilakukan sebanyak 18 kali dengan mengamati jumlah burung serta aktuator. Hasil pengamatan sistem dapat dilihat pada tabel 4.10.

Tabel 4.10. Hasil pengamatan sistem pada pengujian *realtime*

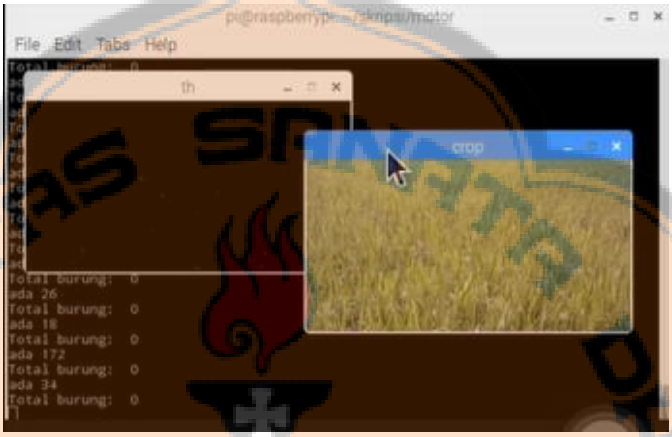
Pengujian	Jumlah burung yang diuji	Aktuator		Nilai *labeling*
		Bergerak/bunyi	Diam	
1	2		×	317
2	4	√		< 180
3	4	√		< 180
4	6	√		< 180
5	6	√		< 180
6	7	√		< 180
7	7	√		< 180
8	7	√		< 180
9	3	√		< 180
10	1	√		< 180
11	6	√		< 180
12	6	√		< 180
13	1	√		< 180
14	8	√		< 180
15	9	√		< 180
16	9		×	333
17	9	√		< 180
18	4	√		< 180

Berdasarkan tabel 4.10 selama 18 kali pengujian sistem mengalami kegagalan 2 kali. Saat pengujian pertama dengan jumlah 2 burung, aktuator tidak bergerak. Aktuator tidak bergerak dikarenakan *labeling* yang terdeteksi pada sistem lebih dari 180. Kegagalan kedua terjadi pada pengujian ke 16 dengan jumlah 9 burung. Aktuator tidak bergerak ketika burung masih pada lahan sawah. hal ini dikarenakan pada lahan sawah terjadi gangguan angin, gangguan angin tersebut menyebabkan nilai *labeling* diatas 180. Jumlah burung yang terdeteksi diluar range batas kontur, proses kontur akan dilewati sehingga aktuator akan diam.

Percobaan lainnya ketika sistem mendeteksi burung aktuator dapat bergerak. Aktuator dapat bergerak dikarenakan burung yang terdeteksi pada sistem melebihi batas kontur. Batas kontur yang ditetapkan pada sistem adalah 8. Burung dengan jumlah kurang

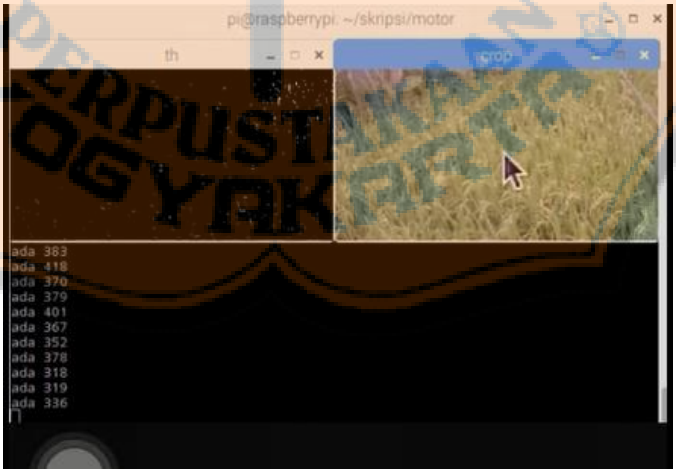
dari 8 secara rasional seharusnya tidak bergerak, namun karena gangguan angin selalu ada pada lahan sawah tersebut mengakibatkan perhitungan burung pada sistem mencapai batas kontur.

Tingkat gangguan angin dapat disimpulkan dengan melihat nilai *labeling* pada saat pengujian. Nilai *labeling* kurang dari 180 sistem masih dapat mendeteksi burung dengan kontur.



Gambar 4.41 Keadaan sawah mengalami sedikit gangguan angin.

Dengan nilai *labeling* dibawah 180, keadaan sawah dinyatakan sedikit mengalami gangguan angin seperti gambar 4.41. Apabila nilai *labeling* lebih dari atau sama dengan 180, maka lahan sawah dinyatakan sedang mengalami gangguan angin yang kencang seperti pada gambar 4.42.



Gambar 4.42 Keadaan sawah mengalami gangguan angin kencang.

4.8. Kelebihan dan kekurangan sistem

Selama proses pengujian, sistem pengusir hama berbasis computer vision memiliki kelebihan dan kekurangan.

4.8.1. Kelebihan

1. Perbedaan umur padi pada lahan sawah tidak mempengaruhi tingkat keberhasilan sistem, sehingga sistem dapat diterapkan pada kondisi apapun.
2. Sistem ini tidak terganggu pada pencahayaan sesuai dengan batasan masalah saat pengujian pada cuaca cerah.
3. Warna gelap pada lahan sawah yang terdeteksi webcam tidak mempengaruhi kerja sistem.

4.8.2. Kekurangan

1. Sistem tidak dapat bekerja dengan baik apabila ada gangguan angin pada lahan sawah. gangguan angin tersebut dapat menyebabkan aktuator bergerak sendiri.
2. Sistem dapat mendeteksi pergerakan selain burung, sehingga pergerakan objek lain dapat mengakibatkan aktuator bergerak.
3. Webcam hanya bisa mendeteksi burung pada jarak tertentu, apabila jarak burung terhadap webcam terlalu jauh mengakibatkan tidak terdeteksinya burung pada sistem.

BAB V
KESIMPULAN DAN SARAN

5.1. Kesimpulan

Selama proses perancangan dan pengujian sistem pengusir hama berbasis computer vision dapat disimpulkan sebagai berikut:

1. Sistem dapat mendeteksi burung bergerak saat keadaan lahan sawah tidak terganggu angin.
2. Sistem dapat menggerakan aktuator ketika satu burung terdeteksi pada lahan sawah.
3. Sistem ini dilakukan percobaan pada jarak ± 3 meter dari webcam dengan 18 kali uji menghasilkan 16 keberhasilan dan 2 *error*.
4. Sistem pengusir hama dapat diimplementasikan pada lahan sawah dengan memasang tali aktuator pada ujung tali yang saling terhubung di lahan sawah.
5. Sistem dapat beroperasi selama ± 6 jam dengan catu daya 2200 mAh.

5.2. Saran

1. Untuk mengurangi kegagalan sistem dari gangguan angin, dapat ditambahkan metode baru sebagai pengambilan keputusan saat aktuator bergerak.
2. Webcam dengan resolusi tinggi dapat mendeteksi burung pada jarak lebih jauh.
3. Pengenalan burung pada pengolahan citra dapat menambahkan ciri-ciri yang lebih spesifik untuk memaksimalkan deteksi burung.
4. Sistem dapat dikembangkan dengan memasang webcam lebih dari satu, sehingga cakupan lahan sawah dapat lebih luas.

DAFTAR PUSTAKA

[1] Masithoh ,Rudiati Evi ,dkk . 2011. *Pengembangan Computer Vision System Sederhana Untuk Memenentukan Kualitas Tomat.* Yogyakarta: Program Studi Elektronika dan Instrumentasi, Fakultas Matematika Ilmu Pengetahuan Alam Universitas Gajah Mada.

[2] Fadliansyah, 2007, *Computer Vision* dan Pengolahan Citra, Yogyakarta: ANDI.

[3] Shakeri, Moein dan Hong Zhang.2012.*Real-Time Bird Detection Based on Background Subtraction*.Canada: University of Alberta Edmonton

[4] Modjo, Ardiyanto Saleh.,2012.,*Rancang Bangun Alat Pengendali Hama Burung Pemakan Bulir Padi Sawah (Oryza Sativa L.) Sistem Mekanik Elektrik*., Program Studi Teknologi Hasil Pertanian Jurusan Agroteknologi Fakultas Ilmu-Ilmu Pertanian Universitas Negeri Gorontalo.

[5]---,---,*Raspberry Pi3 Model B*, https://www.raspberrypi.org/help/faqs/#introWhatIs . diakses 9 desember 2017.

[6]---,---,*Welcome to Raspbian,* https://www.raspberrypi.org/. diakses 9 Desember 2017.

[7]---,---,*Raspberry Pi3 Model B*, https://www.raspberrypi.org/products/raspberry-pi-3-model-b/ . diakses 9 desember 2017

[8] *Logitech*, ---, *HD Webcam C270*, http://www.logitech.com/id-id/product/hd-webcam-c270h?crid=34, diakses 11 Desember 2017.

[9] Herianto, Lawrence., 2014, *Demung Elektronik Berbasis Raspberry Pi*, Tugas Akhir, Jurusan Teknik Elektro, FST, Universitas Sanata Dharma Yogyakarta.

[10] Howse, Joseph. 2013,OpenCV *Computer Vision with Python*,Birmingham UK: Packt Publishing.

[11] Stiefanus, Fendish Cakrawala., 2014, *Pengontrol Robot Soccer Beroda Berbasis Raspberry Pi 3 Sebagai Prototype ERSBI 2017*, Tugas Akhir, Jurusan Teknik Elektro, FST, Universitas Sanata Dharma Yogyakarta.

[12] Prof. Dr. Madenda, Sarifuddin, 2015, *Pengolahan & Video Digital, Erlangga* : Jakarta.

[13] Prijono, Agus dan Marvin Ch. Wijaya, *Pengolahan Citra Digital Menggunakan Matlab*, 2007,informatika : bandung

[14] Rosebrock, Adrian.2014.*Practical Python and OpenCV: An Introductory, Example Driven Guide to Image Processing and Computer Vision*,pysearchimage

[15] Kadir, A., Susanto, A., 2013, *Teori dan Aplikasi Pengolahan Citra*, Andi, Yogyakarta.

[16] Rinaldi, Munir.,2004.,*Pengolahan Citra Digital*.,Bandung.,Informatika.

[17] Marque, Oge.2011.*Practical Image and Video Processing Using MATLAB*.--: IEEE.

[18] Putra, Darma, 2010, *Pengolahan Citra Digital*, Andi, Yogyakarta

[19]---,---,*Operasi Cropping*,http://informatika.web.id/operasi-cropping.htm,diakses 12 Desember 2017

[20]---,---,*Operasi Morfologi Pada Pengolahan Citra*, https://devtrik.com/opencv/operasi-morfologi-pada-pengolahan-citra/ ,diakses 11 Januari 2018

[21]---,---,*Rangkaian power supply+5Vdc TIP32*,http://skemarangkaianpcb.com/rangkaian-power-supply-5-vdc-tip32/,diakses pada 25 Januari 2018.

[22] ---,---, *Driver Motor DC H-Bridge Transistor*, http://elektronika-dasar.web.id/driver-motor-dc-h-bridge-transistor/,diakses pada 11 Januari 2018.

[23]---,---,*Segmentasi Citra Biner Menggunakan Algoritma Connected Component Labeling*,https://salamilmu.wordpress.com/2014/07/02/segmentasi-citra-biner-menggunakan-algoritma-connected-component-labeling/, diakses pada 25 Januari 2018.

[24] Bradski, Gary and Adrian Kaehler.2008.*Learning OpenCV*.United State of America: O'Reilly Media, Inc.

[25] ---,--- *Raspberry Pi 2 & 3 Pin Mappings*, https://docs.microsoft.com/en-us/windows/iot-core/learn-about-hardware/pinmappings/pinmappingsrpi, diakses pada 25 Januari 2018.

[26] Monk, Simon.2014.*Raspberry Pi Cookbook*.United State of America:O'Reilly Media, Inc.

[27] Saharia, Sarat and Deepjoy Das, 2014, *Implementation And Performance Evaluation Of Background Subtraction Algorithms*, Tezpur University, India

[28] Martin Cerman,2014, *Background Subtraction Using Running Gaussian Average: A Color Channel Comparison*, Seminar aus Bildverarbeitung und Mustererkennung.

[29]---,---,*Contours:GettingStarted*, https://docs.opencv.org/3.0beta/doc/py_tutorials/py_imgproc/py_contours/py_contours_begin/py_contours_begin.html#contours-getting-started, diakses pada 5 februari 2018.

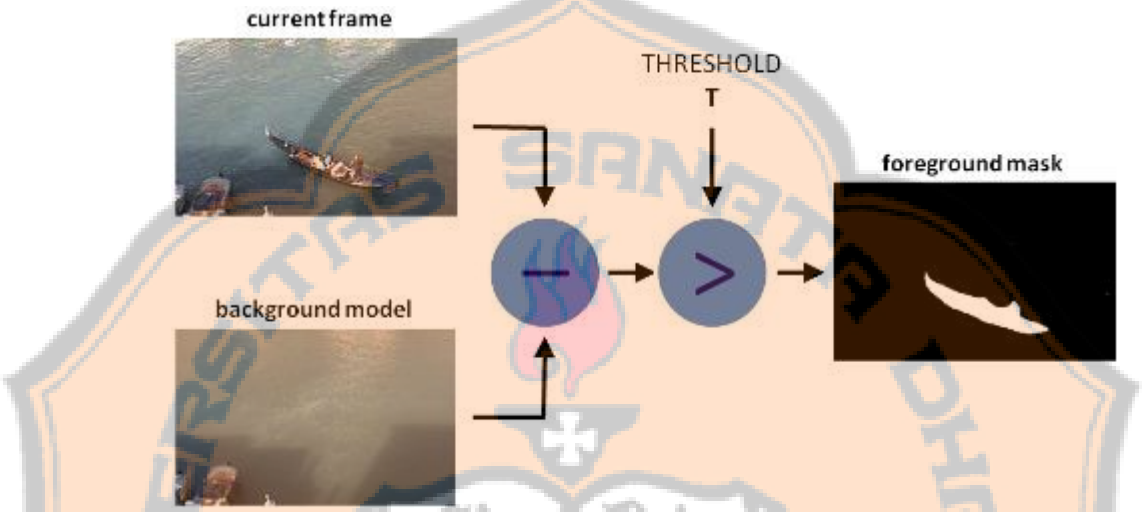
[30]Tjahyadi, Christianto, *Driver Motor L298*, http://christianto.tjahyadi.com/belajar-robotik/driver-motor-l298.html, diakses pada 25 Juli 2018.

[31]Wesiyadi, *LM2956 voltage regulator switching*. http://pulangsore.com/elektronika/komponen/lm2956-voltage-regulator-switching/, diakses pada 25 juli 2018

LAMPIRAN

LAMPIRAN 1. *Background Subtraction*

Background subtraction secara umum mendeteksi objek bergerak dalam video menggunakan kamera statis. Dasar dalam pendekatan mendeteksi suatu objek bergerak dari perbedaan antara frame sekarang dan frame refrensi. *Background* pada suatu citra merepresentasikan sebuah citra yang statis, *background* yang bergerak akan mempengaruhi hasil akhir pengurangan antar citra.



Gambar L.1. Hasil *background subtraction*[27].

Gambar L.1 adalah hasil citra dari *background subtraction*. Metode *background subtraction* bekerja pada citra *grayscale*. Perbedaan frame menjadi bagian penting dalam *background subtraction*. Frame terbaru akan dikurangkan dengan frame sebelumnya sehingga hasil dari pengurangan akan diabsolutkan dan dibandingkan dengan nilai *threshold*, persamaan L.1 menunjukan algoritma dari *background subtraction*[27].

$$|frame(i) - frame(i-1)| > Threshold \text{ [27]} \quad \text{.....(L.1)}$$

Pendekatan ini dibagi menjadi tiga bagian yaitu, memodelkan *background*, melakukan pengurangan frame dan memperbarui model *background*. Latar belakang dimodelkan secara terpisah untuk masing-masing saluran warna dalam ruang warna (R, G, B) dan sekali untuk gambar intensitas. Frame yang dikurangi menghasilkan perbedaan nilai yang akan menunjukan *foreground*. Model *background* harus segera menampilkan citra yang terjadi perubahan,

sehingga memungkinkan deteksi hanya pada objek bergerak. Dengan begitu piksel diklasifikasikan sebagai latar depan yang rendah nilai untuk harus dipilih untuk mencegah integrasi objek bergerak ke latar belakang model. Pada setiap piksel running average akan dihitung pada setiap frame menggunakan persamaan L.2[28].

$$\mu_t = \propto \mu_t + (1-\propto)\mu_{t-1} - 1[28] \qquad \ldots(L.2)$$

Keterangan :

μ_t = frame terbaru pada video

μ_{t-1} = frame sebelumnya pada video

α = *learning rate*

Dengan menggunakan persamaan L.2 maka *background* model akan diperbarui sehinga dapat di lanjutkan pada pengurangan frame pada persamaan L.1. Pada proses pengurangan frame, nilai absolut akan dibandingkan dengan *threshold*.

LAMPIRAN 2. Deteksi Kontur

Kontur adalah keadaan yang ditimbulkan oleh perubahan intensitas pada pixel-pixel yang bertetangga karena adanya perubahan intensitas maka tepi tepi objek citra dapat terdeteksi. Untuk mendeteksi tepi dari suatu objek, perlu dilakukan deteksi tepi pada suatu objek. Pada praktiknya, tepi suatu citra memiliki noise dan blur.

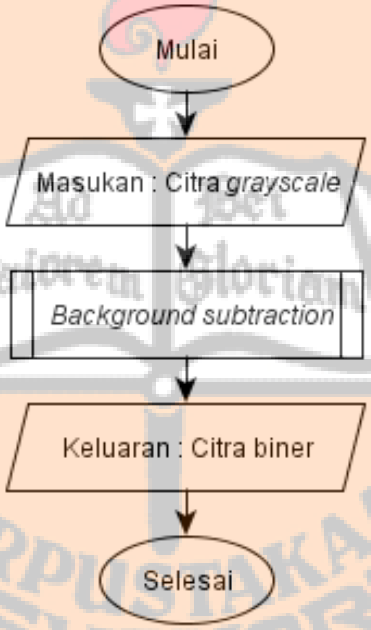
Gambar L.2. Hasil kontur citra [29].

Kontur dapat dijelaskan sebagai kurva yang menggabungkan semua titik kontinyu yang memiliki nilai intensitas seragam. Gambar L.2 menunjukan hasil citra biner yang terkontur. Kontur merupakan salah satu fungsi yang berguna untuk menganalisis bentuk dan deteksi objek maupun pengenalan. Untuk menemukan kontur dari suatu citra, objek harus berwarna putih dan *background* berwarna hitam. Maka sebelum melakukan kontur, citra diubah menjadi citra biner menggunakan *threshold* terlebih dahulu. Untuk menentukan tepi yang terdeteksi dapat menggunakan ketetanggaan horizontal, vertikal dan diagonal pada setiap piksel. Setiap titik titik citra yang terdeteksi tepi akan terkontur. Untuk menghasilkan citra terkontur dengan hasil yang lebih baik dapat menggunakan operasi deteksi tepi *canny edge*[29]. Dalam kontur, citra biner yang terkontur dapat dicari luasan di dalamya untuk memudahkan dalam memisahkan objek yang akan dipili

LAMPIRAN 3. Segmentasi

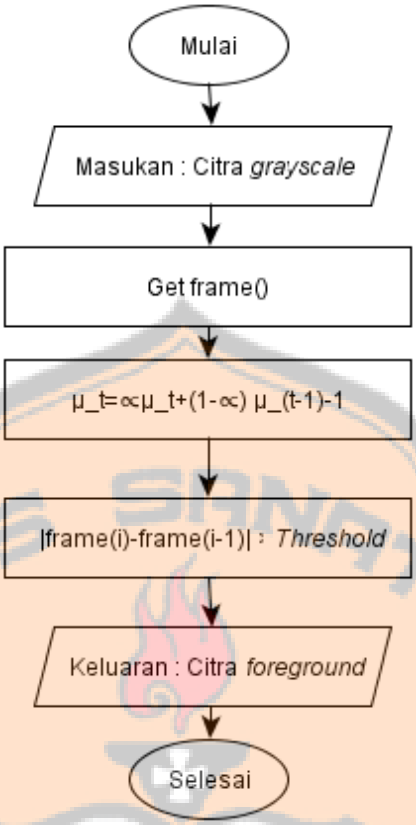
Segmentasi digunakan dalam pengolahan citra untuk memisahkan *foreground* dan *background*. Dalam penelitian ini *foreground* adalah burung dan *background* adalah lahan sawah. Untuk memisahkan *foreground* dari *background* maka citra tersebut harus dikenai *threshold* supaya *foreground* dapat dipisahkan dari *background*. *Threshold* akan ditetntukan melalui uji coba terlebih dahulu untuk memaksimalkan segmentasi citra. Secara umum, proses *thresholding* merupakan citra *grayscale* masukan f(x,y) yang akan menjadi citra biner g(x,y) dengan menggunakan nilai *threshold* (T),Dimana nilai T digunakan untuk *threshold* semua piksel pada citra.

Pada OpenCV untuk menjadikan citra *grayscale* menjadi citra biner menggunakan perintah THRESH_BINARY pada program OpenCV. Proses segmentasi digambarkan pada gambar L.3.



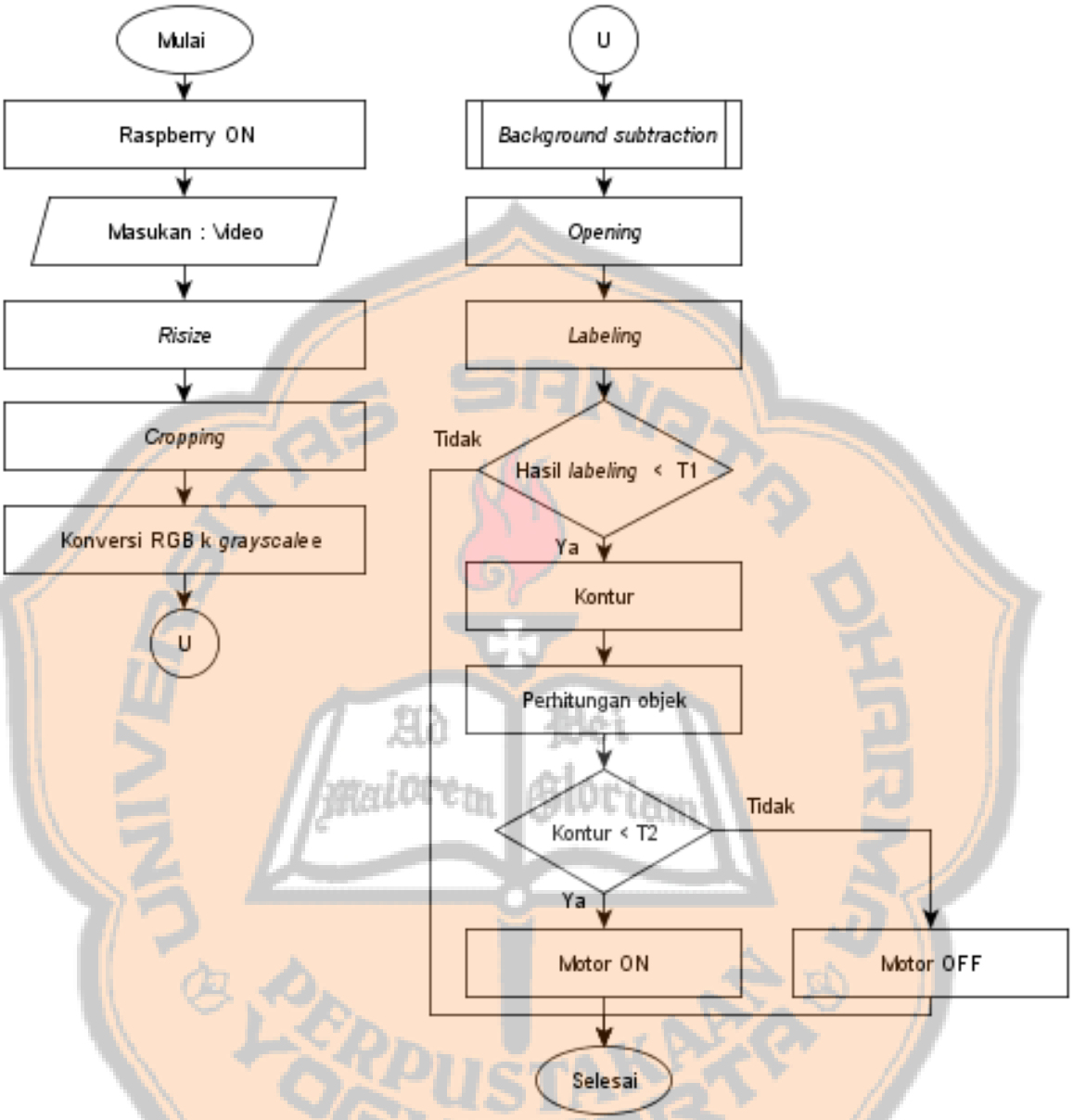
Gambar L.3. *Flowchart* segmentasi

Pada metode *background subtraction* terdapat subrutin seperti pada gambar L.4. Masukan citra *grayscale* akan dilihat nilainya dan di olah pada rumus *background* averaging. Perubahan nilai pada hasil pengurangan frame akan dilakukan pengambangan dengan *threshold*. Nilai yang melebihi *threshold* akan bernilai 0 dan nilai dibawah *threshold* akan bernilai 1. Keluaran dari proses *background subtraction* berupa citra biner.



Gambar L.4. *Flowchart background subtraction.*

LAMPIRAN 4. *Flowchart* Sistem Keseluruhan



Gambar L.5. *Flowchart* sistem keseluruhan.

LAMPIRAN 5. Program Pengambilan Video pada Raspberry pi dengan bahasa pemrograman Python

```
import numpy as np
import cv2
cap = cv2.VideoCapture(0)
# Define the codec and create VideoWriter object
fourcc = cv2.VideoWriter_fourcc(*'XVID')
out = cv2.VideoWriter('rekam.avi',fourcc, 20.0, (640,480))
while(cap.isOpened()):
    ret, frame = cap.read()
    if ret==True:
        out.write(frame)
        cv2.imshow('frame',frame)
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
    else:
        break
# Release everything if job is finished
cap.release()
out.release()
cv2.destroyAllWindows()
```

LAMPIRAN 6. Program Non-Realtime dengan bahasa pemrogram Python pada Windows

```
import cv2
import numpy as np
import scipy
from scipy import ndimage
import time
ROOT_NODE = 1
cap = cv2.VideoCapture('filevideo.format')
count = 0
#################### KONTUR#########################
def kontur():
im,contours,hierarchy=cv2.findContours(OPEN,cv2.RETR_TREE,cv2.CHAIN_APPROX
_NONE)
    totalContours = 0
    for contour in contours:
        area = cv2.contourArea(contour)
        for i in xrange(len(contours)):
            if hierarchy[0][i][1] == ROOT_NODE and area >= 0:
                cv2.drawContours(crop, contour, -1, (50, 250, 0), 3)
                totalContours += 1
    print 'jumlah burung: ', totalContours
    if totalContours >=8 and totalContours<=40:
        print 'motor on'
        time.sleep(1)
    return;
while(1):
    ret, frame = cap.read()
    resize = cv2.resize(frame,(int(frame.shape[1]/2),int(frame.shape[0]/2)))
```

```
    crop=resize[90:320,0:320]
    gray = cv2.cvtColor(crop, cv2.COLOR_BGR2GRAY)
    kernel = np.ones((1,1),np.uint8)
    if count == 0:
        background = gray
        count += 1
    else:
    ####################avrage background#########################
        background = ( gray * 0.2 )+ background * (1 - 0.2)
        foreground = cv2.absdiff(background.astype(np.uint8), gray)
        ret,th = cv2.threshold(foreground,15,255,cv2.THRESH_BINARY)
        OPEN = cv2.morphologyEx(th, cv2.MORPH_OPEN,kernel)
        labeled_array, num_features = ndimage.label(OPEN)
        if num_features > 180:
            print' '
        else:
            kontur()
        cv2.imshow('th',OPEN)
        cv2.imshow('crop',crop)
        cv2.imshow('cap',frame)
        k = cv2.waitKey(30) & 0xff
        if k == 27:
            break
cap.release()
cv2.destroyAllWindows()
```

LAMPIRAN 7. Program Realtime Raspberry pi dengan bahasa pemrograman Python

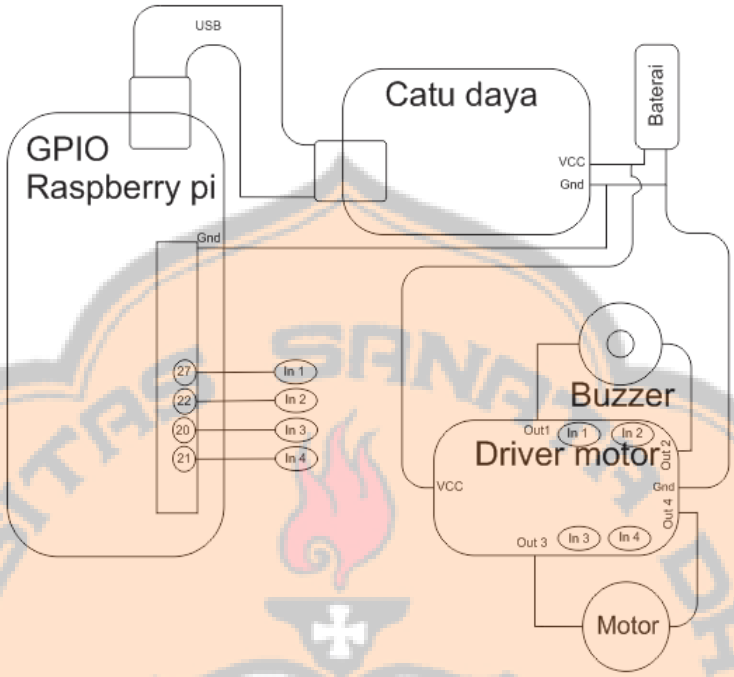
```
import cv2
import numpy as np
import scipy
from scipy import ndimage
import time
import RPi.GPIO as GPIO
from time import sleep
GPIO.cleanup()
in1 = 27
in2 = 22
in3 = 20
in4 = 21
GPIO.setmode(GPIO.BCM)
GPIO.setup(in1,GPIO.OUT)
GPIO.setup(in2,GPIO.OUT)
GPIO.setup(in3,GPIO.OUT)
GPIO.setup(in4,GPIO.OUT)
#######pin setup#########
GPIO.output(in1,GPIO.LOW)
GPIO.output(in2,GPIO.LOW)
GPIO.output(in3,GPIO.LOW)
GPIO.output(in4,GPIO.LOW)
GPIO.setwarnings(False)
ROOT_NODE = 1
count = 0
cap = cv2.VideoCapture(0)
def kontur():
```

```
im,contours,hierarchy=cv2.findContours(OPEN,cv2.RETR_TREE,cv2.CHAIN_APPROX
_NONE)
    totalContours = 0
    for contour in contours:
        area = cv2.contourArea(contour)
        for i in xrange(len(contours)):
            if hierarchy[0][i][1] == ROOT_NODE and area > 1:
                cv2.drawContours(crop, contour, -1, (50, 225, 0), 3)
                totalContours += 1
    print 'Total burung: ', totalContours
    if totalContours >= 8 and totalContours<=40:
########################MOTOR##################
        GPIO.output(in1,GPIO.HIGH)
        GPIO.output(in2,GPIO.LOW)
########################BUZER##################
        GPIO.output(in3,GPIO.LOW)
        GPIO.output(in4,GPIO.HIGH)
        print("maju")
        sleep(1)
        GPIO.output(in1,GPIO.LOW)
        GPIO.output(in2,GPIO.LOW)
        GPIO.output(in3,GPIO.LOW)
        GPIO.output(in4,GPIO.LOW)
        print("mati")
        sleep(0.3)
```

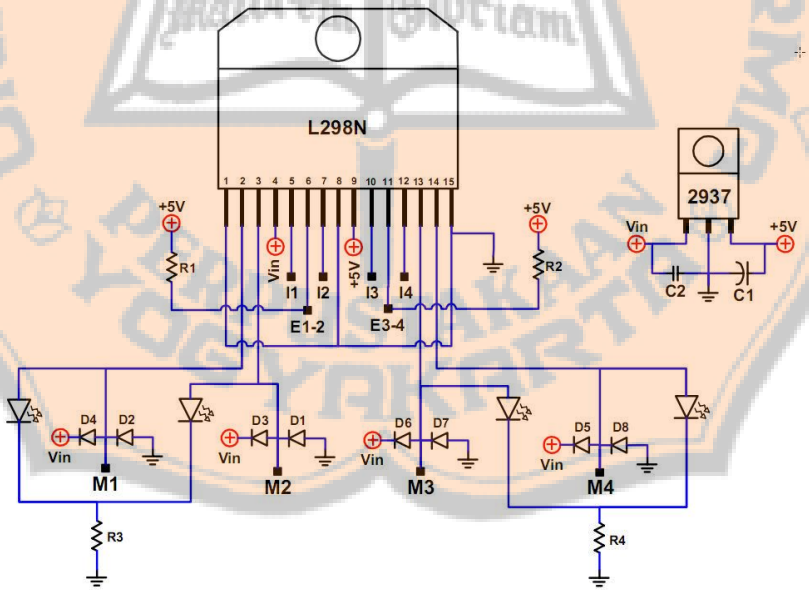
```
    elif totalContours >= 41:
        print("kontur 0")
    else :
        print("kontur 0")
while(1):
    ret, frame = cap.read()
    resize = cv2.resize(frame,(int(frame.shape[1]/2),int(frame.shape[0]/2)))
    crop = resize[70:320,0:320]
    gray = cv2.cvtColor(crop, cv2.COLOR_BGR2GRAY)
    kernel = np.ones((1,1),np.uint8)
    if count == 0:
      background = gray
      count += 1
    else:
    ####################avrage background#########################
      background = ( gray * 0.2 )+ background * (1 - 0.2)
      foreground = cv2.absdiff(background.astype(np.uint8), gray)
      ret,th = cv2.threshold(foreground,15,255,cv2.THRESH_BINARY)
      OPEN = cv2.morphologyEx(th, cv2.MORPH_OPEN,kernel)
      labeled_array, num_features = ndimage.label(OPEN)
      print 'ada',(num_features)
      if num_features <180:
         kontur()
      else:
          print 'ada',(num_features)
    #cv2.imshow('frame',resize)
    #cv2.imshow('fr',frame)
      cv2.imshow('th',OPEN)
      cv2.imshow('crop',crop)
```

```
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
GPIO.cleanup()
cap.release()
cv2.destroyAllWindows()
```

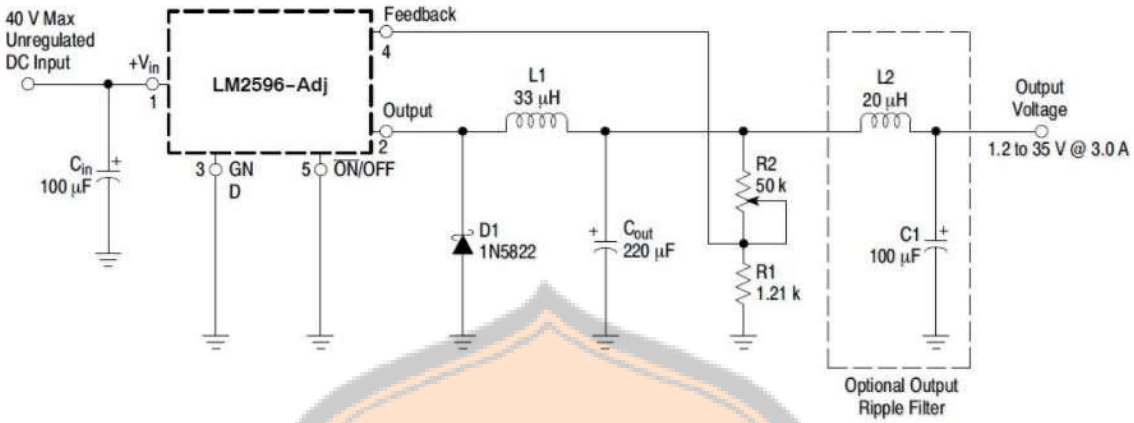
LAMPIRAN 8. Rangakaian Hardware dan Elektronik Keseluruhan



Gambar L.6. Rangkaian Hardware



Gambar L.7. Rangkaian Modul Driver [30]



Gambar L.8. Rangkaian Catu daya LM2596[31]