ANALISIS DAN IMPLEMENTASI *OBJECT TRACKING* PADA KAMERA *WEBCAM* DENGAN *IMAGE PROCESSING* MENGGUNAKAN METODE *MEAN SHIFT*

Disusun Oleh:

N a m a : Fitry Yuliani

NIM : 15523183

PROGRAM STUDI INFORMATIKA – PROGRAM SARJANA

FAKULTAS TEKNOLOGI INDUSTRI

UNIVERSITAS ISLAM INDONESIA

2022

HALAMAN PENGESAHAN DOSEN PEMBIMBING

ANALISIS DAN IMPLEMENTASI *OBJECT TRACKING* PADA KAMERA *WEBCAM* DENGAN *IMAGE PROCESSING* MENGGUNAKAN METODE *MEAN SHIFT*

TUGAS AKHIR

Disusun Oleh:

N a m a	: Fitry Yuliani
NIM	: 15523183

Yogyakarta, 22 Juli 2022

Pembimbing,

(Arrie Kurniawardhani, S.Si., M.Kom.)

HALAMAN PENGESAHAN DOSEN PENGUJI

ANALISIS DAN IMPLEMENTASI *OBJECT TRACKING* PADA KAMERA *WEBCAM* DENGAN *IMAGE PROCESSING* MENGGUNAKAN METODE *MEAN SHIFT*

TUGAS AKHIR

Telah dipertahankan di depan sidang penguji sebagai salah satu syarat untuk memperoleh gelar Sarjana Komputer dari Program Studi Informatika – Program Sarjana di Fakultas Teknologi Industri Universitas Islam Indonesia

Yogyakarta, 22 Juli 2022

Tim Penguji

Arrie Kurniawardhani, S.Si., M.Kom. _____

Anggota 1

Sri Mulyati, S.Kom., M.Kom. _____

Anggota 2

Moh. Idris, S.Kom., M.Kom. _____

Mengetahui,

Ketua Program Studi Informatika – Program Sarjana

Fakultas Teknologi Industri

Universitas Islam Indonesia

(Dr. Raden Teduh Dirgahayu, S.T., M.Sc.)

HALAMAN PERNYATAAN KEASLIAN TUGAS AKHIR

Yang bertanda tangan di bawah ini:

Nama : Fitry Yuliani

NIM : 15523183

Tugas akhir dengan judul:

ANALISIS DAN IMPLEMENTASI *OBJECT TRACKING* PADA KAMERA *WEBCAM* DENGAN *IMAGE PROCESSING* MENGGUNAKAN METODE *MEAN SHIFT*

Menyatakan bahwa seluruh komponen dan isi dalam tugas akhir ini adalah hasil karya saya sendiri. Apabila di kemudian hari terbukti ada beberapa bagian dari karya ini adalah bukan hasil karya sendiri, tugas akhir yang diajukan sebagai hasil karya sendiri ini siap ditarik kembali dan siap menanggung risiko dan konsekuensi apapun.

Demikian surat pernyataan ini dibuat, semoga dapat dipergunakan sebagaimana mestinya.

Banjar, 22 Juli 2022

(Fitry Yuliani)

HALAMAN PERSEMBAHAN

بسم الله الرحمن الرحيم

Dengan mengucap syukur *alhamdulillah*, saya persembahkan karya ini untuk orang-orang yang saya sayangi:

Kedua orang tua,
Bapak Kusen dan Ibu Partini

Kedua orang tua yang senantiasa selalu memberikan doa, semangat, nasehat, dan motivasi untuk kelancaran dalam segala hal. Semoga dengan terselesaikannya tugas akhir yang menjadi salah satu syarat kelulusan pendidikan sarjana ini dapat menjadi kebanggaan dan membahagiakan kedua orang tua.

Kedua kakak,
Yayan Yulianto dan Dwi Kusmanto

Sebagai saudara laki-laki pengganti orang tua di kota perantauan yang selalu menyemangati dan menjadi tempat untuk berbagi cerita. Jika tidak ada kakak terutama kakak yang pertama, mungkin pendidikan ini tidak akan dapat berlanjut dikarenakan keterbatasan ekonomi kedua orang tua saya. Dan untuk kakak kedua saya yang rela meluangkan waktu dan tenaga untuk selalu membantu saya. Semoga dengan prestasi dan terselesaikannya pendidikan sarjana ini dapat membuat suatu kebanggaan dan semoga Allah SWT selalu membalas kebaikan dan rezeki yang lebih dari yang telah diberikan. Aamiin

Keponakan,
Alm. Alutfy Zulfiqly

Satu-satunya keponakan yang seperti adik sendiri karena sejak lahir selalu tinggal bersama. Keponakan yang selalu mengajarkan arti kehidupan, selalu berbagi apa pun itu, dan selalu menyemangati untuk lulus kuliah sampai akhir hayatnya. Semoga Allah SWT menempatkannya di Surga Firdaus. Aamiin

HALAMAN MOTO

"... dan jangan kamu berputus asa dari rahmat Allah. Sesungguhnya yang berputus asa dari rahmat Allah, hanyalah orang-orang kafir."

(QS. Yusuf: 87)

"... Allah akan mengangkat (derajat) orang-orang yang beriman di antaramu dan orang-orang yang diberi ilmu beberapa derajat. Dan Allah Mahateliti terhadap apa yang kamu kerjakan."

(QS. Al-Mujaadila: 11)

"Maka sesungguhnya bersama kesulitan ada kemudahan, sesungguhnya bersama kesulitan ada kemudahan. Maka apabila engkau telah selesai (dari sesuatu urusan), tetaplah bekerja keras (untuk urusan yang lain), dan hanya kepada tuhanmulah engkau berharap."

(QS. Al-Insyirah: 5-8)

KATA PENGANTAR

Assalamu'alaikum Warahmatullahi Wabarakaatuh.

Alhamdulillah, puji dan syukur penulis panjatkan kehadirat Allah SWT yang telah memberikan rahmat, hidayah dan karunia-Nya sehingga penulis dapat menyelesaikan laporan tugas akhir ini. Tak lupa *shalawat* serta salam senantiasa penulis haturkan kepada junjungan Nabi Muhammad SAW, yang telah membawa kita dari zaman jahiliyah menuju zaman yang terang benderang.

Tugas akhir merupakan salah satu syarat wajib yang harus dipenuhi untuk memperoleh gelar sarjana pada jenjang Strata Satu (S1) di Jurusan Informatika Fakultas Teknologi Industri Universitas Islam Indonesia. Pelaksanaan tugas akhir ini sebagai salah satu kuliah wajib dari Jurusan Informatika Fakultas Teknologi Industri Universitas Islam Indonesia dan menjadi sarana bagi penulis untuk menambah ilmu pengetahuan serta pengalaman dalam menerapkan keilmuan sesuai dengan bidang yang sebelumnya sudah didapatkan pada proses perkuliahan.

Adapun pembahasan terkait laporan tugas akhir ini adalah analisis dan implementasi *object tracking* pada kamera webcam dengan *image processing* menggunakan metode *Mean Shift*. Penulis menyadari bahwa jika tidak ada bimbingan, motivasi, dukungan serta bantuan dari berbagai pihak, penulisan laporan tugas akhir ini tidak akan terwujud. Oleh karena itu pada kesempatan ini penulis ingin menyampaikan rasa terimakasih kepada:

1. Allah SWT yang Maha pengasih, Maha penyayang dan Maha penolong yang selalu memberikan kemudahan setelah kesulitan dalam perkuliahan sehingga penulis dapat menyelesaikan laporan tugas akhir ini dengan baik.
2. Orang tua serta keluarga penulis atas segala doa dan dukungan yang diberikan.
3. Bapak Dr. Raden Teduh Dirgahayu, S.T., M.Sc. selaku Ketua Program Studi Informatika – Program Sarjana Fakultas Teknologi Industri Universitas Islam Indonesia.
4. Ibu Chanifah Indah Ratnasari, S.Kom., M.Kom. selaku dosen pembimbing akademik selama menjadi mahasiswa.
5. Ibu Arrie Kurniawardhani, S.Si., M.Kom. selaku dosen pembimbing penulis yang telah banyak memberikan waktu dan ilmu serta arahan pengerjaan Tugas Akhir.
6. Ibu Sri Mulyati, S.Kom., M.Kom. dan Bapak Moh. Idris, S.Kom., M.Kom selaku dosen penguji sidang tugas akhir penulis yang telah memberikan masukan serta arahan perbaikan tugas akhir ini.

7. Bapak dan Ibu dosen Jurusan Informatika yang telah memberikan ilmu kepada penulis yang telah memberikan ilmunya kepada penulis selama perkuliahan, semoga selalu dalam lindungan Allah SWT dan ilmu yang telah diajarkan dapat bermanfaat serta dapat menjadi amal jariah. Aamiin
8. Ade Rohmat Purnomo yang telah meluangkan waktu, bantuan, memberikan informasi dan ilmu yang sangat menunjang penulis untuk melakukan penelitian ini.
9. Uwa Sudarno dan keluarga, sebagai orang tua kedua penulis yang selalu mendukung dan memberikan doa kepada penulis.
10. Irvan Christiawan selaku rekan di Informatika Universitas Islam Indonesia yang selalu memberikan bantuan serta motivasi kepada penulis.
11. Pengurus KOSMIK #9 dan seluruh keluarga KOSMIK UII yang telah mendukung dan menyemangati selama perkuliahan.
12. Seluruh teman angkatan Metamorf Informatika Fakultas Teknologi Industri Universitas Islam Indonesia yang juga mendukung selama penelitian Tugas Akhir.

Penulis tentu menyadari bahwa laporan ini masih terdapat kekurangan, karena keterbatasan pengetahuan serta pengalaman di lapangan. Oleh karena itu, penulis mengharapkan kritik dan saran yang membangun untuk menyempurnakan laporan tugas akhir ini. Penulis berharap laporan ini dapat bermanfaat bagi semua pihak.

Wassalamu'alaikum Warahmatullahi Wabarakaatuh.

Banjar, 22 Juli 2022

(Fitry Yuliani)

SARI

Saat ini kebutuhan manusia terhadap keamanan lingkungan rumah, kantor, minimarket, dan lain sebagainya sangat meningkat. Pada kasus yang ada di sekitar lingkungan seperti *minimarket* atau toko kelontong, biasanya menggunakan kamera CCTV untuk mengamankan area kasir. Namun kamera CCTV pada umumnya tidak dapat bergerak atau statis sehingga tidak dapat melacak pergerakan target objek yang berada di area kasir dalam jakauan 180°.

Object tracking adalah suatu proses yang mengikuti perubahan posisi dari suatu objek pada suatu interval tertentu. *Object tracking* sudah pernah dikembangkan sebelumnya menggunakan metode *Mean Shift*, dengan tingkat keberhasilan yang tinggi yaitu 100% pada 6-7 *fps*. Namun pada sistem penelitiannya tidak menetapkan koordinat target objek, kemudian pada penelitiannya belum menambahkan metode *cropping* ROI (*Region of Interest*) untuk mengeliminasi daerah yang tidak diamati.

Dengan dikembangkannya sistem menggunakan metode *Mean Shift* pada kamera *webcam* dengan Raspberry Pi sebagai komputasi sistem dan penggerak motor servo yang terhubung dengan pin digital input / *output* yang sudah diregistrasi yang akan menggerakkan kamera sesuai dengan keberadaan target objek secara otomatis dalam jangkauan 180° dengan menambahkan metode *cropping* ROI agar komputasi menjadi semakin ringan.

Sebelum menerapkan metode *Mean Shift*, perlu mengatur lokasi target pelacakan dari *frame* pertama video yang ditangkap oleh kamera *webcam* NYK NEMESIS EVEREST secara langsung. Kemudian, dilakukan metode *cropping* ROI untuk segmentasi target objek bagian wajah. Ruang warna yang digunakan pada metode ini adalah HSV (*Hue, Saturation, Value*). Kemudian di-*masking* menggunakan batas ambang, dengan mengambil warna kulit bagian wajah yang dianggap sebagai target objek. Hasil segmentasi tersebut kemudian dijadikan parameter pada penerapan *Mean Shift* untuk *frame* selanjutnya. Kemudian motor servo akan bergerak sesuai dengan data masukkan koordinat x baru dari target objek pada setiap *frame* video. Hasil pelacakan sistem diuji dengan menggunakan metode IOU (*Intersection over Union*) untuk membandingkan hasil pelacakan dari sistem dengan hasil pelacakan yang seharusnya. Nilai IOU hasil pelacakan objek yang bergerak ke arah kiri dan kanan yaitu lebih dari 0.70 atau 70% yang berarti sistem dapat melacak target objek dengan baik karena kotak pelacakan mendekati kotak yang seharusnya.

Kata kunci: Raspberry Pi, *frame*, *object tracking*, *tracking window*, *masking*, *Mean Shift*, *histogram*, *intersection over union*.

GLOSARIUM

Array	larik yang berisi kumpulan data dengan tipe yang sama.
Error	kejadian program yang tidak sesuai yang diharapkan.
Firmware	sebuah perangkat lunak terprogram yang ditempatkan di perangkat keras.
Flowchart	diagram alur suatu proses.
Frame	satuan terkecil dalam video.
Library	kumpulan dari prodram atau fungsi yang sudah ada.
List	tipe data yang dapat memuat banyak nilai
Testing	menguji sistem.
Training	melatih sistem.
Website	kumpulan halaman dalam suatu domain yang berisi informasi.

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB I
PENDAHULUAN

1.1 Latar Belakang

Saat ini kebutuhan manusia terhadap keamanan lingkungan rumah, kantor, *minimarket*, dan lain sebagainya sangat meningkat. Pada kasus yang ada di sekitar lingkungan seperti *minimarket* atau toko kelontong, biasanya menggunakan kamera CCTV untuk mengamankan area kasir namun kamera CCTV pada umumnya tidak dapat bergerak atau statis sehingga tidak dapat melacak pergerakan target objek yang berada di area kamera CCTV dalam jakauan 180°. Agar kamera dapat melacak target objek dengan jarak jangkauan lebih luas, perlu pengembangan teknologi yang dapat melacak pergerakan target objek dalam jangkauan 180°.

Object tracking adalah suatu proses yang mengikuti perubahan posisi dari suatu objek pada suatu interval tertentu. Pada *object tracking*, proses ekstraksi ciri dari suatu objek merupakan hal yang sangat penting karena ciri-ciri yang didapat dari objek tersebut yang akan digunakan untuk perbandingan dalam melakukan pelacakan (Wijayana, W, & Sa'adah, 2015). Ada beberapa metode untuk mengimplementasikan pelacakan objek salah satunya yaitu metode *Mean Shift*. Menurut Wijayana et al. (2015) metode *Mean Shift* merupakan sebuah metode *tracking* yang sederhana dan *low cost* pada penerapannya.

Pada penelitian yang dilakukan oleh Purnomo (2016) hasil penelitiannya menggunakan metode *Mean Shift* yaitu sistem berjalan optimal mampu melacak target objek non-rigid atau bentuk dari objek yang berubah-ubah dengan memberikan tingkat keberhasilan 100% pada 6-7 fps dengan jumlah kepadatan intensitas piksel di bawah batas ambang 2275875 densitas pada setiap *frame*. Kemudian robot mampu melacak bola dengan kecepatan laju bola dengan kecepatan laju bola maksimal 0.15625 meter per detik. Namun pada sistem penelitiannya tidak menetapkan koordinat target objek sehingga untuk melacak target objek, pengguna harus secara manual memilih objek pelacakannya terlebih dahulu. Kemudian pada penelitiannya belum menambahkan metode *cropping* ROI (*Region of Interest*) untuk mengeliminasi daerah yang tidak diamati.

Maka dari itu pada tugas akhir ini penulis mengembangkan sistem menggunakan metode *Mean Shift* dengan Raspberry Pi sebagai komputasi sistem dan penggerak motor servo yang terhubung dengan pin digital input / *output* yang sudah diregistrasi yang akan

menggerakkan kamera sesuai dengan keberadaan target objek secara otomatis dalam jangkauan 180° dengan menambahkan metode *cropping* ROI (*Region of Interest*) agar komputasi menjadi semakin ringan.

1.2 Rumusan Masalah

Berdasarkan latar belakang masalah di atas, masalah yang dapat dirumuskan adalah sebagai berikut:

a. Bagaimana merancang sistem pelacakan target objek dengan menggunakan kamera *webcam*?
b. Bagaimana mengimplementasikan sistem pelacakan pergerakan target objek menggunakan metode *Mean Shift*?

1.3 Batasan Masalah

Adapun batasan masalah dalam penelitian ini yaitu:

a. Pergerakan kamera hanya dapat bergerak dalam rentang 0° – 180° secara horizontal.
b. Data citra berupa video yang diperoleh secara langsung menggunakan kamera webcam NYK NEMESIS EVEREST.
c. Target objek yang dilacak yaitu warna kulit wajah manusia.
d. Tidak ada cermin dalam jangkauan kamera, karena jika ada refleksi warna kulit wajah pada cermin maka sistem akan menganggapnya sebagai target objek.

1.4 Tujuan Penelitian

Tujuan dari penelitian ini adalah:

a. Merancang sistem pelacakan target objek dengan menggunakan kamera *webcam*.
b. Mengimplementasikan sistem yang dapat melacak pergerakan target objek menggunakan metode *Mean Shift*.

1.5 Manfaat Penelitian

Adapun manfaat yang didapatkan dari penelitian ini adalah:

a. Bagi Penulis

Penulis dapat mempelajari teori, metode, langkah pengerjaan dan implementasi untuk penyelesaian pelacakan objek melalui proses data secara digital agar pelacakan target objek dapat memaksimalkan jangkauan dalam melacak pergerakan manusia. Penelitian ini melatih

penulis dalam menyelesaikan permasalahan pada keterbatasan kamera CCTV yang digunakan di area kasir minimarket atau toko kelontong dengan memberikan solusi dalam bentuk pemrosesan secara digital melacak target objek dalam jangkauan 180 derajat.

b. Bagi Peneliti Lain

Peneliti lain dapat melakukan penelitian lanjutan yang lebih kompleks dan mendapatkan gambaran dari penelitian ini, sehingga peneliti lain dapat mengurangi resiko kegagalan dan *error* dari penelitian yang akan dilakukan. Selain itu penelitian ini dapat dijadikan sebagai referensi untuk ilmu informatika dan elektronika.

c. Bagi Masyarakat

Masyarakat dapat menggunakan hasil dari penelitian ini untuk memantau target objek di area kasir *minimarket* atau toko kelontong.

1.6 Sistematika Penulisan

Dalam penyusunan laporan tugas akhir, sistematika penulisan dibagi menjadi beberapa bab sebagai berikut:

Bab I Pendahuluan

Berisi latar belakang terkait permasalahan aktual yang mendasari dilaksanakannya penelitian ini, kekurangan kamera CCTV yang ada di *minimarket* atau toko kelontong, pelacakan objek, penelitian sebelumnya, serta solusi dari permasalahan tersebut. Berdasarkan latar belakang, disusun rumusan masalah sebagai acuan dalam merencanakan penyelesaian masalah, kemudian batasan masalah untuk membatasi ruang lingkup permasalahan yang akan diselesaikan, tujuan penelitian sebagai target yang akan dicapai pada penelitian ini, manfaat penelitian bagi penulis, peneliti lain, dan masyarakat, metodologi penelitian dan sistematika penulisan.

Bab II Landasan Teori

Bab ini berisi penjelasan terkait teori-teori yang berhubungan dengan topik penelitian sebagai dasar untuk melakukan penelitian. Penjelasan terkait dengan penelitian sejenis yang sudah dilakukan sebelumnya. Penjelasan yang ada pada bab ini mengambil referensi dari berbagai sumber. Topik penelitian yang berhubungan yaitu kamera CCTV, kamera *webcam*, Raspberry Pi, pengolahan citra digital, *object tracking*, metode *Mean Shift* dan pengujian sistem.

Bab III Metodologi Penelitian

Bab ini berisi penjelasan tahapan penelitian, gambaran umum sistem, solusi penyelesaian masalah, dan analisis kebutuhan yang diperlukan dalam penelitian ini. Pada bab ini terdiri dari analisis kebutuhan sistem, perancangan alat, perancangan sistem, analisis pengujian alat, analisis pengujian sistem dan pengujian kecepatan sistem.

Bab IV Hasil dan Pembahasan

Bab ini berisi hasil dan pembahasan dari setiap proses yang dijalankan dalam sistem, pengujian sistem, pengujian kecepatan sistem, pengujian alat.

Bab V Kesimpulan dan Saran

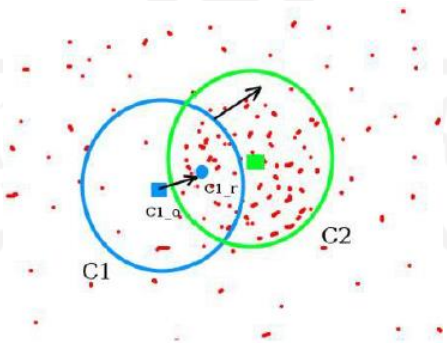
Bab ini berisi kesimpulan dari hasil penelitian, kesesuaian hasil penelitian dengan tujuan penelitian dan saran yang mendukung sehingga penelitian ini dapat dilanjutkan oleh peneliti lain, naik untuk memperbaiki kekurangan maupun mengembangkan keterbatasan dari penelitian ini.

BAB II
LANDASAN TEORI

2.1 *Mean Shift*

Mean Shift merupakan metode pelacakan objek visual, biasanya diaplikasikan pada aplikasi *real-time* atau secara langsung seperti pengawasan, ruangan pintar dan lain-lain untuk melacak objek yang sedang bergerak (Purnomo, 2016). Objek tersebut dapat bermacam-macam, dapat berupa barang seperti bola, kendaraan, manusia dan lain-lain. Menurut (Wijayana et al., 2015), *Mean Shift* adalah sebuah metode pelacakan target objek berbasis kepadatan atau *density* sebuah ciri. Pada penelitian ini, penulis menggunakan *Mean Shift* menggunakan warna kulit manusia sebagai acuan pelacakan objeknya. *Mean Shift* banyak digunakan untuk melakukan pelacakan terhadap objek yang bersifat non-rigid atau bentuk dari objek yang berubah-ubah. Pada metode *Mean Shift*, objek non-rigid dapat dideskripsikan dengan histogram warna, sehingga target objek dapat tetap terdeteksi walaupun objek berubah bentuk.

Pada *website* dokumentasi OpenCV (*Open Source Computer Vision Library*) tertulis, secara sederhana jika terdapat satu set poin yang berupa sebaran piksel yang disebut dengan histogram *backprojection*, kemudian ada kotak pelacakan kecil di mana jendela kecil itu harus berpindah ke area kepadatan piksel. Hal ini diilustrasikan pada Gambar 2.1.



Gambar 2.1 Ilustrasi pencarian area kepadatan yang maksimum

Sumber: *website* dokumentasi OpenCV

Inisialisasi jendela ditampilkan dalam bentuk lingkaran berwarna biru dengan nama C1, pusat asli dari lingkaran biru ditandai dengan persegi panjang berwarna biru yang bernama

C1_o. Tetapi jika yang dicari adalah titik pusat yang ada di dalam citra tersebut, maka titik pusat yang ditemukan adalah titik C1_r ditandai dengan bentuk lingkaran biru yang kecil. Kemungkinan C1_r itu tidak selalu identik dengan kepadatan piksel target objek, maka dari itu diperlukan iterasi untuk menemukan pusat objek dengan kepadatan piksyang maksimal. perlu memindahkan jendela agar lingkaran jendela baru dapat sesuai dengan *centroid* atau titik tengah sebelumnya. Jika masih belum sesuai, iterasi dilanjutkan sampai pusat jendela dan pusatnya berada pada posisi yang sama. Dengan demikian posisi jendela berada pada posisi dengan distribusi piksel yang maksimum, ditandai dengan lingkaran hijau bernama C2.

Pada metode *Mean Shift*, proses kerjanya yaitu melakukan konvolusi (mengalikan sebuah citra dengan sebuah *mask* atau kernel) gambar atau *frame* video dalam *window* yang dipilih menggunakan sebuah *window* yang dinamakan sebagai *parzen window*. *Parzen window* disebut juga *kernel density estimation* yang merupakan sebuah *kernel* digunakan untuk perhitungan *density*, di mana sebelumnya dilakukan *smoothing* atau perataan histogram warna. *Smoothing* histogram warna ini dilakukan agar pada saat pencarian *density* tidak didapatkan kepadatan yang salah. Konvolusi pada *parzen window* dapat menggunakan banyak fungsi, di antaranya yaitu *Gaussian kernel*, *Uniform kernel*, *Epanechnikov kernel*, dan lain sebagainya. Pada penelitian ini konvolusi yang digunakan yaitu *Gaussian kernel*. Terdapat nilai *density* pada metode *Mean Shift*, di mana nilai *density* merupakan parameter *Mean Shift* yang dapat melakukan iterasi menggunakan Persamaan (2.1), di mana K di sana merupakan *kernel density* dan h merupakan ukuran *window* yang digunakan (Wijayana et al., 2015).

$$f(x) = \frac{1}{nh^d} + \sum_{i=1}^{n} K\left(\frac{x - xi}{h}\right) \qquad (2.1)$$

Kemudian setelah mendapatkan nilai *density* pada *frame* tersebut, selanjutnya adalah menghitung nilai gradien *density* yang dilakukan pada *frame* selanjutnya. Setelah nilai gradien *density* pada kedua *frame* (*frame* pertama dan *frame* selanjutnya) didapatkan, kedua nilai tersebut dihitung kemiripannya (*similarity*). Untuk menghitung kemiripannya menggunakan Persamaan (2.2), di mana nilai *pu* merupakan nilai *gradient density* pada *frame* kedua dan nilai *qu* merupakan nilai gradien *density* pada *frame* pertama (Wijayana et al., 2015).

$$p(y) = \cos\theta_y = \sum_{u=1}^{m} \sqrt{pu(y)qu} \qquad (2.2)$$

Untuk menentukan gradien *density* pada kedua *frame* tersebut mirip atau tidak, menggunakan jarak *Bhattacharyya Coefficient* ($p(y)$). Di mana semakin kecil jarak yang dihasilkan maka akan semakin mirip kedua gradien *density* tersebut. Untuk menghitung jarak *Bhattacharyya* dapat menggunakan Persamaan (2.3) (Wijayana et al., 2015).

$$d(y) = \sqrt{1 - p(y)} \qquad (2.3)$$

Jika nilai jarak *Bhattacharyya* kurang dari *threshold* yang digunakan, maka posisi pelacakannya akan diperbaharui ke posisi *similarity* yang terdekat, proses tersebut akan berjalan terus menerus hingga pada *frame* terakhir dari suatu video (Wijayana et al., 2015).

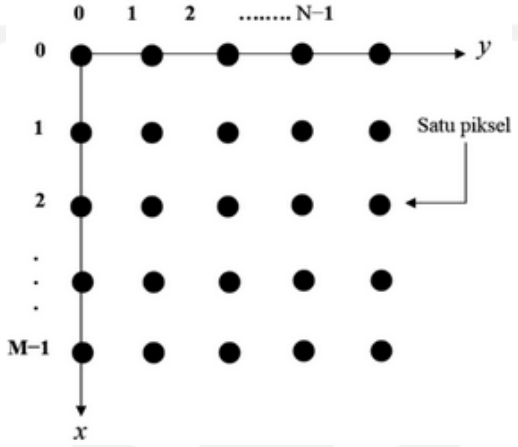
2.2 Pengolahan Citra Digital

Menurut Putra (2010), pengolahan citra digital secara umum yaitu mengolah atau memproses suatu gambar 2 dimensi menggunakan komputer. Di mana citra digital itu sendiri merupakan sebuah larik (*array*) yang berisi nilai-nilai yang *real* maupun kompleks sekalipun, nilai-nilai tersebut direspresentasikan dengan deretan bit tertentu.

Secara langsung komputer tidak dapat merepresentasikan citra analog, sehingga citra analog harus dikonversi terlebih dahulu menjadi citra digital, karena citra digital dapat diproses oleh komputer. Komputer dapat memproses secara langsung gambar/citra yang dihasilkan dari peralatan digital seperti kamera DSLR karena pada peralatan digital terdapat sampling dan kuantisasi sedangkan pada peralatan analog tidak dilengkapi sistem sampling dan kuantisasi (Andono, T.Sutojo, & Muljono, 2017).

Sistem sampling adalah sitem yang dapat mengubah citra kontinu menjadi citra digital, di mana citra kontinu tersebut diperoleh dari sistem optik yang menerima sinyal analog seperti kamera-kamera analog. Cara mengubah citra kontinu menjadi citra digital yaitu dengan cara membagi citra analog tersebut menjadi M baris dan N kolom (citra diskrit), semakin besar nilai M dan nilai N (*MxN*) maka semakin halus citra digital yang dihasilkan. Pertemuan dari baris dan kolom tersebut disebut dengan piksel. Sedangkan sistem kuantitasi adalah sistem yang mengubah intensitas analog menjadi intensitas diskrit (Andono et al., 2017).

Pada proses dapat dimungkinkan membuat gradasi warna yang sesuai dengan kebutuhan. Karenanya sistem sampling bertugas untuk memotong-motong citra menjadi M baris dan N kolom yang secara bersamaan sistem kuantitasi menentukan besar intensitas yang terdapat pada titik tersebut. Proses tersebut menghasilkan resolusi citra yang kita inginkan. Secara umum sistem koordinasi yang digunakan dalam teori pengolahan citra diilustrasikan pada Gambar 2.2 (Andono et al., 2017).



Gambar 2.2 Sistem digunakan untuk mewakili citra

Sumber: (Andono et al., 2017)

Artinya, sebuah citra digital dapat ditulis dalam bentuk matriks berikut:

$$f(x,y) \approx \begin{bmatrix} f(0,0) & f(0,1) & \dots & f(0,M-1) \\ \dots & \dots & \dots & \dots \\ f(N-1,0) & f(N-1,1) & \dots & f(N-1,M-1) \end{bmatrix} \quad (2.4)$$

Dari Persamaan (2.4), didapatkan fungsi matematis:

$$\begin{aligned} 0 \le x \le M-1 \\ 0 \le y \le N-1 \\ 0 \le f(x,y) \le G-1 \end{aligned} \quad (2.5)$$

Pada Persamaan (2.4), secara matematis citra digital ditulis dengan *f*(*x*,*y*) sebagai fungsi intensitas di mana nilai *x* (baris) dan nilai *y* (kolom) merupakan koordinat posisi dan fungsi intensitas *f*(*x*,*y*) merupakan nilai dari fungsi setiap titik (*x*,*y*). Fungsi intensitas *f*(*x*,*y*) tersebut menyatakan besar tingkat derajat keabuan, intensitas citra maupun warna dari piksel pada titik tersebut.

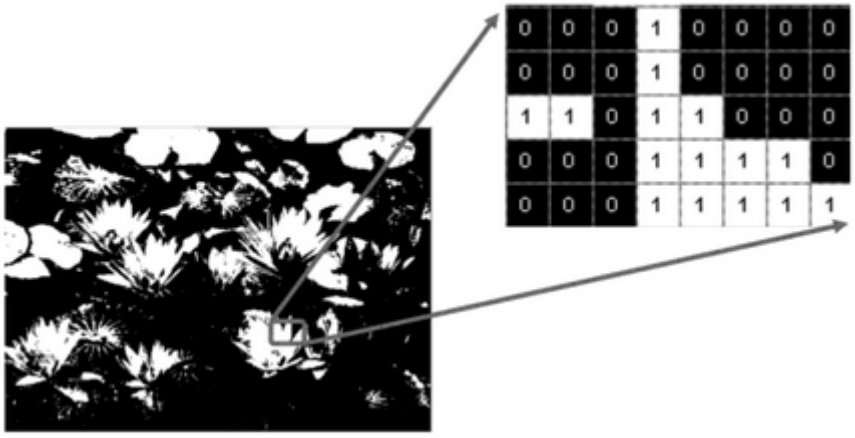
Untuk nilai *M*, *N*, *G* pada Persamaan (2.5) secara umum merupakan bilangan positif dari hasil perpangkatan bilangan 2 dan nilai *G* ini tergantung pada proses digitalisasi citra yang biasanya nilai 0 menyatakan intensitas hitam dan nilai 1 menyatakan intensitas putih.

2.2.1 Tipe Citra

Adapun tipe citra yang biasa digunakan peneliti untuk melakukan penelitian, berikut diantaranya:

a. Citra Biner

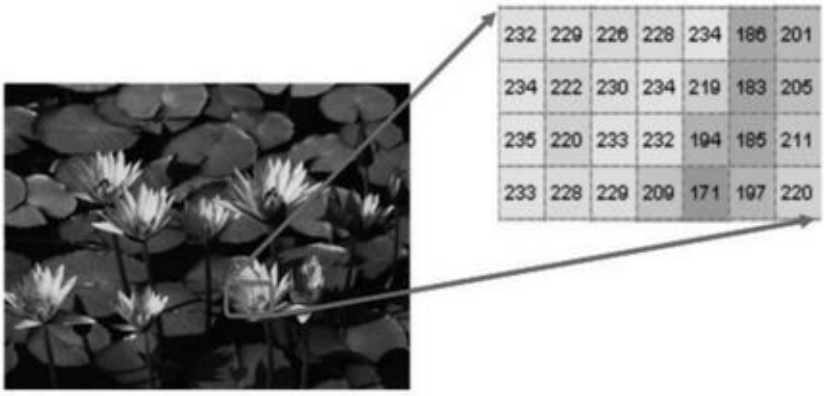
Pada citra biner setiap piksel hanya membutuhkan 1 bit memori maka setiap piksel hanya mempunyai 2 kemungkinan nilai intensitas yaitu nilai 0 atau 1. Gambar 2.3 menunjukkan bahwa citra biner yang dilihat dari dekat memiliki beberapa nilai intensitas piksel (Andono et al., 2017).



Gambar 2.3 Citra biner dengan nilai piksel 1 atau 0.
Sumber: (Andono et al., 2017)

b. Citra *Grayscale*

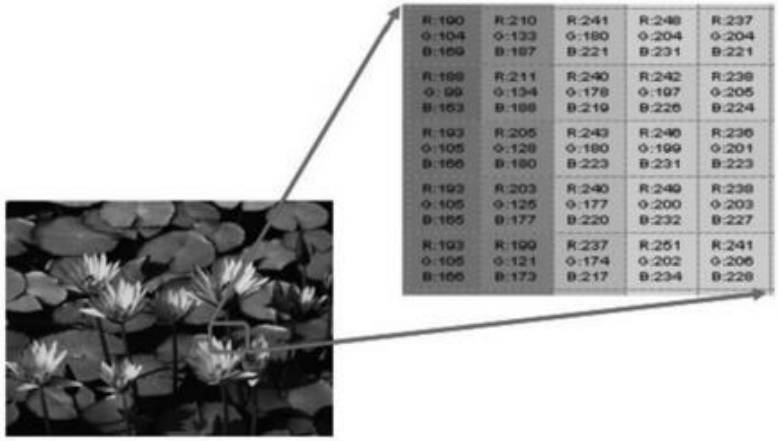
Citra *grayscale* adalah matriks data yang nilai-nilainya mewakili intensitas setiap piksel berkisar antara 0 sampai 255. Setiap piksel membutuhkan 8 bit memori (Andono et al., 2017). Gambar 2.4 menunjukkan citra *grayscale* yang dilihat dari dekat dengan beberapa nilai intensitas piksel.



Gambar 2.4 Citra *grayscale* dengan nilai piksel antara 0 sampai dengan 255.
Sumber: (Andono et al., 2017)

c. Citra Warna

Citra warna adalah citra yang memiliki tiga komponen warna yang spesifik di setiap pikselnya, warna tersebut diantaranya yaitu merah (*Red*), hijau (*Green*) dan biru (*Blue*). Warna-warna setiap piksel ini ditentukan oleh kombinasi dari intensitas warna RGB yang disimpan pada bidang warna di lokasi piksel tersebut. Setiap intensitas warna pada citra warna ini membutuhkan 8 bit, jadi karena citra memiliki 3 komponen warna maka format *file* grafis menyimpan citra warna sebanyak 24 bit. Hal tersebut menyebabkan citra warna memiliki 24 juta kemungkinan warna dari hasil kombinasi warna RGB. Gambar 2.5 menunjukkan citra warna yang dilihat dari dekat dengan beberapa nilai intensitas piksel.



Gambar 2.5 Citra warna dengan komponen warna R (*Red*), G (*Green*) dan B (*Blue*)
Sumber: (Andono et al., 2017)

2.2.2 Elemen-Elemen Citra Digital

Suatu gambar/citra digital memiliki sejumlah elemen-elemen dasar yang akan dimanipulasi dalam proses pengolahan citra. Elemen-elemen tersebut adalah:

a. Warna

Warna merupakan elemen dasar dari gambar digital yang paling penting, untuk setiap gambar harus melalui digitalisasi yang direpresentasikan dalam bentuk matriks warna (*matrix of color*). Setiap warna memiliki panjang gelombang yang berbeda-beda, panjang gelombang cahaya tersebut dihasilkan dari cahaya yang dipantulkan oleh sebuah objek. Warna yang memiliki panjang gelombang tertinggi yaitu warna merah, sedangkan warna yang memiliki panjang gelombang terendah yaitu warna ungu. Untuk setiap manusia memiliki persepsi yang berbeda-beda tergantung dari yang dirasakan oleh sistem visual manusia itu sendiri terhadap panjang gelombang yang dipantulkan oleh sebuah objek yang dilihat, warna yang diterima oleh manusia merupakan gabungan dari cahaya dengan panjang gelombang yang berbeda-beda. Kombinasi warna RGB (*Red, Green,* Blue) merupakan kombinasi warna yang memiliki rentang warna paling besar dengan nilai R, G, dan B masing-masing 8 bit dengan rentang 0 sampai 255 (Handoyo, 2006).

Terdapat beberapa ruang warna pada citra digital, sebagai berikut:

1. RGB (Red, Green, Blue)

Ruang warna ini terdiri dari dimensi warna merah, hijau dan biru dengan tujuan untuk merepresentasikan dan menampilkan suatu gambar pada sistem elektronik seperti kamera CCTV, kamera *webcam*, televisi, kamera DSLR dan alat elektronik lainnya.

2. HSV (Hue, Saturation, Value)

Ruang warna HSV (*Hue, Saturation, Value*) merupakan pemetaan warna dari warna primer RGB (*Red, Green, Blue*) menjadi dimensi yang lebih mudah dipahami oleh mata manusia.

3. YCrCb

Ruang warna YCrCb terdiri dari komponen Y yang merepresentasikan intensitas rendah sedangkan untuk komponen Cr dan Cb merepresentasikan komponen yang kromatis. Biasanya ruang warn aini digunakan pada video dan sistem fotografi.

4. L*a*b

Ruang warna L*a*b ditetapkan oleh CIE (*Commision Internationale de l'Eclairge*) dengan komponen L* sebagai terang, a* sebagai koordinat warna merah atau warna hijau dan komponen b* sebagai koordinat warna kuning atau warna biru.

b. *Brightness*

Brightness atau tingkat kecerahan atau intensitas cahaya menyatakan banyaknya cahaya yang diterima oleh mata manusia. Cahaya ini dirasakan seperti lampu penerang yang berwarna putih di mana objek akan terlihat semakin putih jika cahaya lampu putih tersebut tingkat kecerahan atau intensitas cahayanya tinggi. Sebaliknya, objek tersebut akan terlihat semakin gelap jika tingkat kecerahannya rendah dan jika tingkat kecerahannya 0 maka objek tersebut akan terlihat berwarna hitam (Handoyo, 2006).

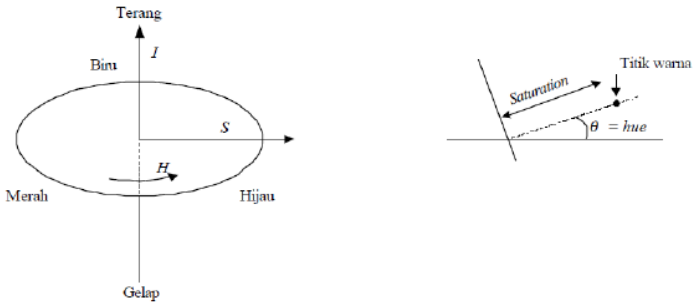
c. *Saturation*

Saturation merupakan tingkat kepekatan atau konsentrasi suatu objek. Objek akan terlihat lebih pekat warnanya karena semakin tinggi nilai saturasinya maka semakin pekat warna benda tersebut. Warna dengan nilai saturasi yang tinggi yaitu warna merah, warna dengan nilai saturasi yang rendah yaitu warna merah muda dan warna dengan tingkat saturasi 0 yaitu warna putih (Handoyo, 2006).

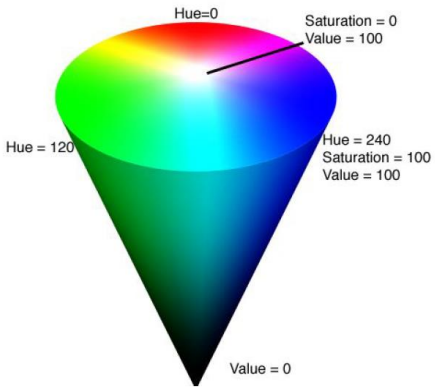
d. *Hue*

Hue merupakan nilai yang menyatakan corak atau warna suatu objek, di mana *hue* berasosiasi dengan gelombang cahayanya (Handoyo, 2006).

Gambar 2.6 merupakan ilustrasi HSV yang dimodelkan sebagai dimensi yang berputar di sekitar sumbu vertical pusat yang mewakili seluruh nilai komponen warna *hue*, *saturation*, *value*. Di mana komponen *hue* dikuantisasi dengan setiap perubahan nilai menyatakan warna yang berbeda. Di mana komponen *hue* merupakan warna dari sudut 0 – 360 derajat, warna merah ada pada 0 – 60 derajat, kuning 60 – 120 derajat, hijau 120 – 180 derajat, cyan 180 – 240 derajat, biru 240 – 300 derajat, dan magenta 300 – 360 derajat. Komponen *saturation* mendefinisikan gradasi warna dari warna jenuh atau abu-abu pada sumbu vertikal, jumlah abu-abu pada ruang warna yang berkisar antara 0 – 100%. Komponen *value* merupakan kecerahan intensitas cahaya atau disebut dengan *brightness*, yang bergerak dari gelap ke terang.



Gambar 2.6 Model HSV



Gambar 2.7 HSV *cone*

Sumber: (Cardani, 2001)

2.3 Histogram Citra

Histogram citra merupakan grafik yang menggambarkan penyebaran nilai-nilai intensitas piksel pada suatu citra atau bisa juga pada bagian tertentu di dalam citra tersebut. Kita dapat mengetahui frekuensi kemunculan nisbi (*relative*) dari intensitas citra dengan histogram citra dan histogram citra dapat menunjukkan *brightness* atau tingkat kecerahan dan kontras (*contrast*) suatu citra atau gambar. Maka dari itu histogram citra merupakan alat bantu yang sangat berharga untuk mengerjakan pengolahan citra secara kualitatif maupun kuantitatif (Purnomo, 2016).

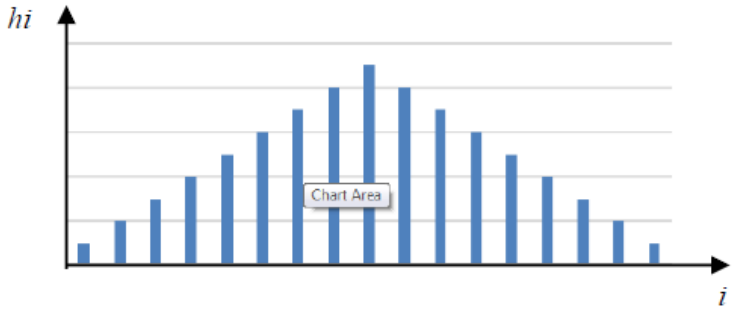
Misalkan suatu citra digital memiliki L derajat keabuan dengan nilai 0 sampai $L - 1$. Secara matematis, histogram citra dihitung menggunakan Persamaan (2.6) di mana nilai *ni* merupakan jumlah piksel dari suatu citra yang memiliki derajat keabuan *i* dan *n* merupakan jumlah dari seluruh piksel dari suatu citra. Nilai *ni* tersebut telah dinormalisasikan dengan membagi nilai *ni* dengan nilai n, sehingga nilai *hi* (histogram) berada dalam rentang 0 – 1. Histogram citra secara grafis ditampilkan dengan diagram batang seperti pada Gambar 2.8.

$$hi = \frac{ni}{n}, i = 0,1 \ldots \qquad (2.6)$$

Di mana,

ni merupakan jumlah piksel yang memiliki derajat keabuan *i*

n merupakan jumlah seluruh piksel di dalam citra



Gambar 2.8 Histogram citra

Sumber: (Purnomo, 2016)

(a) Kapal 512 x 512, 8 bit (b) Histogram citra kapal

Gambar 2.9 Citra kapal (512 x 512) dan histogramnya

Sumber: (Purnomo, 2016)

2.4 Normalisasi MIN MAX

Normalisasi *min-max* merupakan metode normalisasi yang mengubah nilai pada suatu data secara linier, normalisasi *min-max* juga dapat mengurangi nilai yang telalu besar (Reynaldo, Adikara, & Wihandika, 2020). Metode normalisasi *min-max* dapat menggunakan Persamaan (2.7).

$$d' = \frac{d - min\,(p)}{max\,(p) - min\,(p)} \qquad (2.7)$$

Di mana,

d' merupakan nilai dari hasil normalisasi

d merupakan nilai sebelum normalisasi

p merupakan nilai keseluruhan dari suatu atribut

max(p) merupakan nilai maksimum dari suatu atribut

min(p) merupakan nilai minimum dari suatu atribut

2.5 Kamera

2.5.1 Kamera CCTV (*Closed Circuit Television*)

Kamera CCTV adalah suatu alat yang dapat mengirimkan data berupa video melalui transmisi kabel maupun tanpa kabel ke lokasi tertentu dengan menampilkan citra berupa video dari kamera yang dipasangkan pada suatu ruangan yang ingin dipantau, direkam atau dianalisa. Teknologi CCTV ini sudah ada sejak tahun 1940 sejalan dengan perkembangan kamera pada umumnya, namun sekitar pada tahun 1970 kamera baru digunakan dan diaplikasikan untuk keamanan (Hadiwijaya, Darjat, & Zahra, 2014).

Menurut Hadiwijaya, Darjat, & Zahra (2014), keberhasilan sistem CCTV ditentukan oleh kualitas elemen-elemen yang mendukung sistem tersebut diantaranya:

a. Kamera

Kamera CCTV terbagi menjadi 2 macam bentuk yaitu *fixed camera* (posisi kamera tidak dapat berubah-ubah) dan PTZ (*Pan Tilt Zoom*) (posisi kamera dapat berubah-ubah dan dapat diperbesar).

b. Media Transmisi

Media transmisi pada kamera CCTV menggunakan kabel koaksial dan berupa *wireless* yang menggunakan *access point* berupa *router*.

c. Monitor

Menampilkan suatu objek yang dapat dijangkau dan ditangkap oleh kamera CCTV.

d. Aplikasi Piranti Lunak

Suatu aplikasi yang dapat mengontrol kamera CCTV dari suatu tempat dan dapat diintegrasikan dengan *server* penyimpanan video.

e. Media Penyimpanan

DVR (*Digital Video Recorder*) atau *Hardisk.*

2.5.2 Kamera *Webcam*

Kamera *webcam* adalah sebuah periferal (perangkat tambahan) berupa kamera pengambil gambar, video dan mikrofon sebagai pengambil audio yang dikendalikan oleh komputer atau jaringan komputer (Mauko & Tunliu, 2016). Media transmisi pada kamera *webcam* biasanya menggunakan kabel USB, sehingga mempermudah pengguna untuk menggunakannya dengan sistem *plug and play*.

2.6 Raspberry Pi

Raspberry Pi adalah perangkat PC yang berukuran sebesar kartu kredit. Raspberry Pi ini memiliki sistem *Broadcom* BCM2835 *chip* (SoC) yang mencakup ARM1176JZF-S 700 MHz *processor* (*firmware* yang termasuk dengan sejumlah mode "Turbo" sehingga pengguna dapat mencoba *overlocking* hingga 1 GHz tanpa mempengaruhi garansinya), VideoCore IV GPU, dan pada awalnya dikirim dengan ukuran RAM 256 MB yang kemudian ditingkatkan menjadi 512 MB. Raspberry Pi ini juga memiliki *built-in hard disk* atau *solid-state drive*, namun untuk melakukan *booting* harus menggunakan kartu SD dan dapat digunakan untuk penyimpanan jangka panjang (Hakim & Putra, 2013).

Pada Raspberry Pi terdapat GPIO (*General Purpose Input / Output*), di mana GPIO merupakan pin *generic* pada *chip* yang dapat dikontrol atau diprogram melalui perangkat lunak baik dikonfigurasi sebagai pin input maupun *output*. Raspberry Pi memiliki GPIO dengan 26 pin yang berukuran 2,54 mm. Konektor GPIO memiliki fitur sebagai berikut:

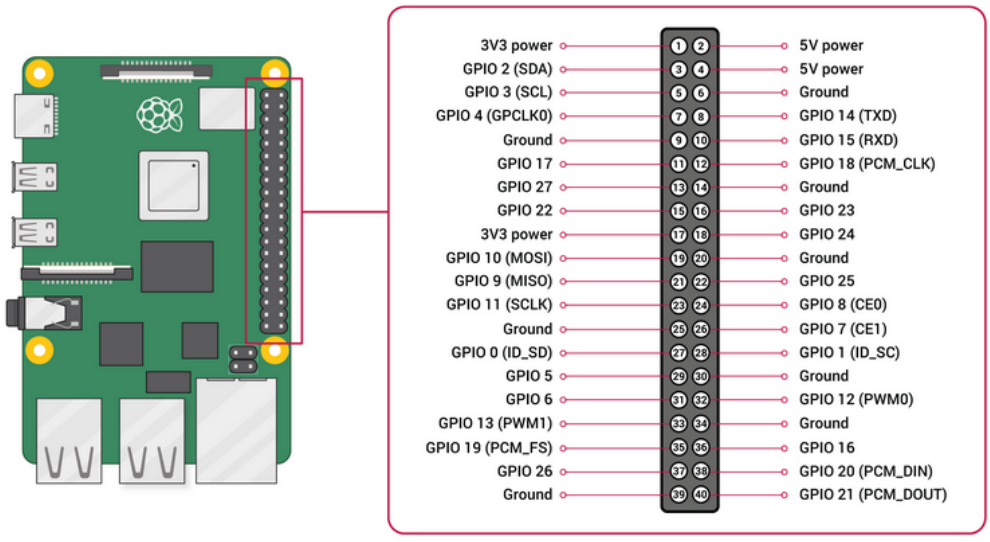
a. Pin antarmuka IIC atau I2C, pin ini digunakan untuk menghubungkan modul perangkat keras dengan hanya dua pin kontrol.
b. SPI (*Serial Peripheral Interface*), pin ini memiliki konsep yang mirip dengan pin antarmuka 12C namun memiliki standar yang berbeda.
c. Serial Rx dan Tx, pin ini dapat digunakan untuk berkomunikasi dengan perangkat serial.
d. Pin PWM (*Pulse Width Modulation*), pin ini digunakan untuk mengontrol daya.
e. Pin PPM (*Pulse Position Modulation*), pin ini dapat digunakan untuk mengendalikan motor servo.

Tegangan yang disediakan oleh GND (*Ground*) ada 2 yaitu 3.3V dan 5V, pin yang berlabel SCL (*Serial Clock*) dan SDA (*Serial Data*) dapat digunkan untuk pin antarmuka 12C dan untuk pin yang berlabel MOSI (*Master Output Slave In*), MISO (*Master Input Slave Output*) dan SCKL (*Serial Clock*) dikontrol dari master yang dapat digunakan untuk

menghubungkan perangkat SPI dengan kecepatan yang tinggi. Semua pin GPIO memiliki tingkat logika sebesar 3.3V, sehingga tingkat *output* atau keluarannya yaitu 0 – 3.3V dan input atau masukannya tidak boleh lebih tinggi dari 3.3V (Hakim & Putra, 2013).

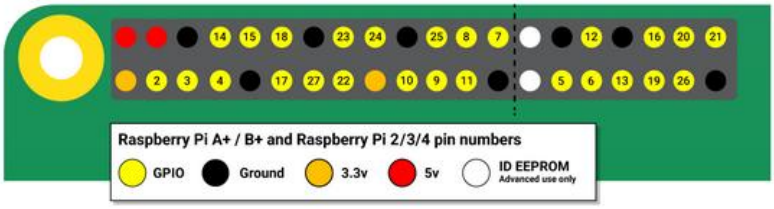
Raspberry Pi 4 Model B yang digunakan penulis merupakan produk terbaru dari Raspberry Pi. Raspberry Pi 4 Model B ini menawarkan peningkatan kecepatan prosesor, performa multimedia, memori, dan konektivitas dibandingkan dengan generasi yang sebelumnya yaitu Raspberry Pi 3 Model B+, dengan mempertahankan kompatibilitas *backwards* dan konsumsi daya yang sama. Raspberry Pi 4 Model B ini memberikan performa desktop yang sebanding dengan sistem PC x86. Fitur utama dari produk ini mencakup *quad-core* 64-bit dengan performa prosesor yang tinggi, mendukung *dual-display* pada resolusi hingga 4K melalui *port micro-HDMI*, perangkat keras *video decode* hingga 4Kp60, RAM hingga 8GB, *dual-band* 2.4/5.0 GHz *wireless* LAN, *Bluetooth* 5.0, *Gigabit Ethernet*, USB 3.0, dan kemampuan PoE (melalui *add-on* PoE HAT yang terpisah) (Raspberry Pi, n.d.).

Konfigurasi GPIO (*General-Purpose Input/Output*) pada Raspberry Pi 4 Model B memiliki 40 pin (ditunjukkan pada Gambar 2.10 yang diambil dari *website* dokumentasi Raspberry Pi).



Gambar 2.10 Daftar GPIO Raspberry Pi 4 Model B

Sumber: (Raspberry Pi, n.d.)



Gambar 2.11 Nomor pin GPIO Raspberry Pi 4 Model B

Sumber: (Raspberry Pi, n.d.)

Pada penelitian Purnomo (2016), dijelaskan nama-nama pin pada Raspberry Pi 2 Model B yang sama dengan pin Raspberry Pi 4 Model B. Berikut ini merupakan penjelasan dari setiap pin GPIO pada Raspberry Pi 4 Model B:

a. Pin 1 dan pin 17 merupakan *power* 3v3, pin ini memiliki arus maksimal yang tersedia sekitar 50 mA. Sumber daya tersebut cukup digunakan untuk menyalakan beberapa LED atau mikroprosesor.
b. Pin 2 dan pin 4 merupakan *power* 5V, pin ini terhubung langsung ke tegangan input pada Raspberry Pi yang mampu memberikan arus penuh dari tegangan adaptor yang terhubung.
c. Pin 3 merupakan BCM2 (I2C *data*), di mana SDA (*Serial Data*) merupakan salah satu pin IIC atau I2C (*Inter-Integrated Circuit*) pada Raspberry Pi. Pin SDA ini menetapkan nilai resistor tetap sebesar 1,8 kΩ *pull-up* ke 3.3V yang tidak sesuai jika digunakan sebagai tujuan umum dari I/O karena tidak adanya *pull-up* resistor yang diinginkan.
d. Pin 7 merupakan BCM 3 (I2C *clock*), di mana SCL (*Serial Clock*) merupakan salah satu pin I2C pada Raspberry Pi. Seperti pin SDA, pin SCL juga memiliki nilai tetap resistor 1,8 kΩ *pull-up* ke 3.3V.
e. Pin 8 merupakan pin UART (*Universal Asynchronous Receiver/Transmitter*) sebagai *transmitter* (TXD), sedangkan pin 10 merupakan pin UART sebagai *receiver* (RXD).
f. Pin 11, 13, 15, 16, 18, 22, 29, 31, 36, 37 hanya dapat digunakan sebagai input maupun *Output*.
g. Pin 6, 9, 14, 20, 25, 30, 34, 39 berfungsi sebagai *grounding*.
h. Pin 12, 32 dan 33 merupakan pin PWM (*Pulse Width Modulation*) dan dapat digunakan sebagai input / *output*.
i. Pin 19 dan pin 35 merupakan pin MOSI (*Master Output Slave Input*) pada SPI0 dan SPI1. Terdapat 3 *controller* SPI (*Serial Peripheral Interface*) pada BCM2711 Raspberry

Pi 4, namun yang ada pada *header* Raspberry Pi 4 ini hanya ada *controller* SPI0 dan SPI1.

j. Pin 21 dan pin 38 merupakan pin MISO (*Master Input Slave Output*) pada *controller* SPI0 dan SPI1.

k. Pin 23 dan pin 40 merupakan pin SLCK (*Serial Clock*) pada *controller* SPI0 dan SPI1.

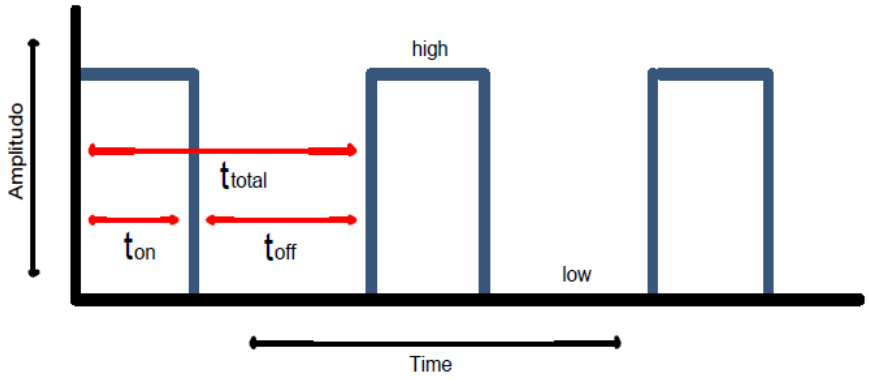
l. Pin 24 merupakan pin CE0 (*Chip Enable* atau *Chip Select*).

m. Pin 26 merupakan pin CE1 (*Chip Enable* atau *Chip Select*).

n. Pin 27 dan pin 28 pada umumnya dicadangkan untuk IIC/I2C agar dapat berkomunikasi dengan EEPROM.

2.7 PWM (*Pulse Width Modulation*)

Menurut Akbar & Riyadi (2019), PWM merupakan suatu proses perbandingan antara sinyal *carrier* dengan sinyal modulasi yang menghasilkan sinyal kotak dengan *width pulse* (lebar pulsa) yang berbeda-beda. *Width pulse* ini diatur dengan menggunakan *duty cycle* (siklus kerja), *duty cycle* itu sendiri merupakan presentase periode sinyal *low* dan *high*. Presentase *duty cycle* akan berbanding lurus dengan tegangan rata-rata yang dihasilkan. Sinyal PWM memiliki *width pulse* yang bervariasi sesuai dengan *duty cycle*. Penjelasan sinyal PWM dijelaskan pada Gambar 2.12.



Gambar 2.12 Sinyal PWM (*Pulse Width Modulation*)

Sumber: (Akbar & Riyadi, 2019)

Di mana,

t_{on} merupakan waktu di mana tegangan keluaran berada pada posisi *high* (tinggi).

t_{off} merupakan waktu di mana tegangan keluaran berada pada posisi *low* (rendah).

t_{total} merupakan waktu satu siklus atau waktu total dari penjumlahan antara t_{on} dan t_{off}.

Duty cycle sebuah gelombang didefinisikan pada Persamaan (2.8).

$$d = \frac{t_{on}}{(t_{on} + t_{off})} \times 100\% \qquad (2.8)$$

Tegangan keluaran dapat bervariasi dengan *duty cycle* dan dapat dirumuskan dengan Persamaan (2.9).

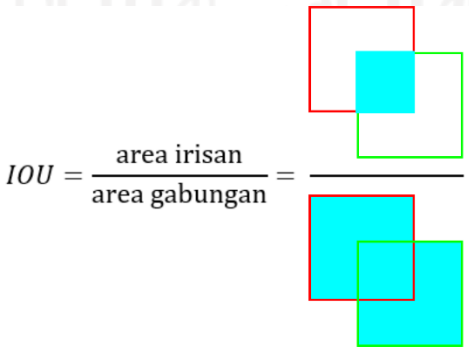
$$\mathrm{V_{out}} = d \times \mathrm{V} \qquad (2.9)$$

2.8 IOU (*Intersection over Union*)

Menurut Salim (2020), IOU merupakan nilai berdasarkan statistik kesamaan dan keberagaman set data sampel yang bertujuan untuk mengevaluasi area yang beririsan antara dua *bounding box*. *Bounding box* ini terdiri dari boks hasil prediksi dari sistem dan *bounding box ground truth* yang berarti boks yang seharusnya. IOU dapat dicari menggunakan Persamaan (2.10).

$$IOU = \frac{\mathrm{area}(BB_{prediksi} \cap BB_{groundTruth})}{\mathrm{area}(BB_{prediksi} \cup BB_{groundTruth})} \qquad (2.10)$$

Dapat dilihat pada Gambar 2.13, persamaan untuk mendapatkan nilai IOU adalah perbandingan dari daerah area yang beririsan dibagi dengan area gabungan. Kemudian setelah mendapatkan nilai IOU, semakin beririsan jarak antara boks prediksi dengan boks sebenarnya maka semakin baik nilai IOU-nya.



Gambar 2.13 Ilustasi persamaan IoU

Sumber: (Salim, 2020)

2.9 Penelitian Sejenis

Pada penelitian yang dilakukan oleh Romzi (2018) hasil penelitiannya yaitu metode *Mean Shift* dapat diimplementasikan pada kamera pengawas semi otomatis dengan mengatur nilai variabel dengan jumlah iterasi 2 – 4, ukuran *template* target model berbentuk persegi atau persegi panjang yang lebih condong panjang ke arah horizontal, dan luasan *increment area* dengan perbesaran 2 – 4 kali. Sistem pelacakan objek dapat diterapkan pada kamera pengawas dengan menggunakan 2 buah motor. Pada penelitian ini, pelacakan objek berhasil diterapkan dengan keberhasilan yang tinggi dan pada penelitiannya objek yang dilacak yaitu manusia.

Kombinasi *ASIFT-Mean Shift* yang dilakukan oleh Wijayana et al. (2015) hasil penelitian menunjukkan bahwa kondisi lingkungan objek berpengaruh terhadap banyaknya kemungkinan objek terdeteksi dengan tepat. Kondisi lingkungan terkontrol cenderung memiliki hasil yang lebih baik dari lingkungan yang tidak terkontrol. Selain itu, nilai *threshold* dengan nilai 0,9 dan radius 10% memiliki kecenderungan dapat mendeteksi objek dengan tepat. Hasil penelitian ini juga menunjukkan bahwa metode *ASIFT-Mean Shift* dapat mengatasi permasalahan perubahan sudut pandang atau *point of view* yang terjadi pada pelacakan objek di mana nilai akurasi yang dihasilkan sama dengan 30%. Pada penelitian Wijayana et al. (2015) menerapkan penelitiannya sebagai alat pengamanan berbasis video yang dapat bekerja secara otomatis. Dalam penelitiannya, *Mean Shift* menggunakan warna sebagai acuan *tracking*-nya.

Hasil penelitian yang dilakukan oleh Setyawan & Purwanto (2012) yaitu dari hasil percobaan yang dilakukan dapat diketahui bahwa penjejakan objek dapat dilakukan dengan menggunakan kamera *pan tilt*. Dengan menggunakan metode *Mean Shift* dapat dihasilkan titik koordinat tengah dari objek dengan baik. Algoritma *Mean Shift* yang digunakan dapat melakukan penjejakan objek memiliki *error* kurang dari 15 piksel. Kontrol PID (*Propotional, Integral, Derivative*) yang digunakan dapat mencapai *setting point* dengan waktu kurang yang dari 3 detik.

Penelitian yang dilakukan oleh Mahali & Harjoko (2014) dengan judul pelacakan benda bergerak menggunakan *Mean Shift* dengan perubahan skala dan orientasi, tingkat keberhasilan dalam mengenali objek yang menjadi target meningkat ketika parameter masukan *increment area* dinaikan, namun waktu pelacakannya juga akan bertambah.

Semakin besar wilayah objek yang akan dilacak maka semakin bertambah juga waktu dalam melakukan pelacakan objek tersebut.

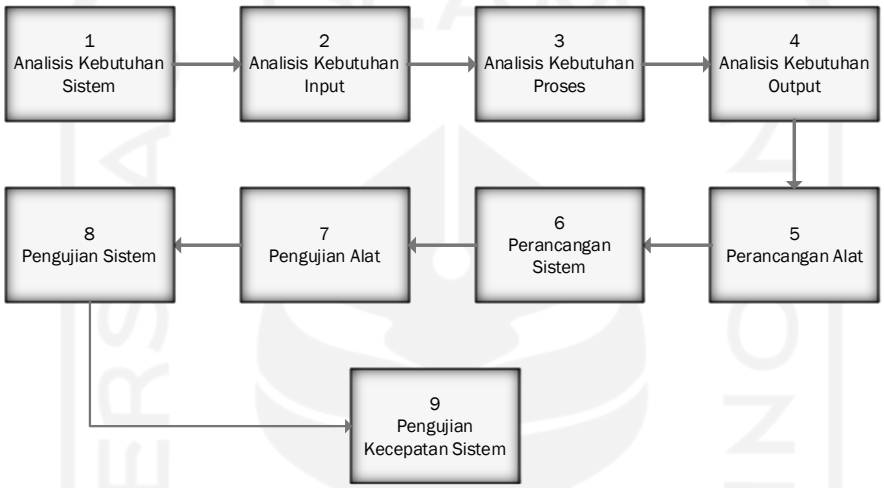
Penelitian yang dilakukan oleh Purnomo (2016) menggunakan bahasa pemrograman C++, hasil penelitiannya menggunakan metode *Mean Shift* yaitu sistem berjalan optimal mampu melacak target objek non-rigid atau bentuk dari objek yang berubah-ubah dengan memberikan tingkat keberhasilan 100% pada 6-7 fps dengan jumlah kepadatan intensitas piksel di bawah batas ambang 2275875 densitas pada setiap *frame*. Kemudian robot mampu melacak bola dengan kecepatan laju bola dengan kecepatan laju bola maksimal 0.15625 meter per detik.

Dari berbagai sumber penelitian yang sudah dilakukan sebelumnya, maka penelitian ini juga memilih metode dengan mengacu pada hasil yang didapatkan. Pada proses *pre-processing* dilakukan percobaan dengan menggunakan metode *cropping* ROI sesuai dengan saran penelitian yang dilakukan oleh Purnomo (2016) dan menggunakan warna kulit manusia sebagai acuan pelacakan target objeknya. Penerapan *Mean Shift* menggunakan batasan kriteria dengan maksimal 10 iterasi atau berpindah ke wilayah kepadatan piksel yang lebih tinggi/maksimal setidaknya 1 poin. Kemudian pada proses penggerakan kamera, motor servo mengacu pada koordinat x dari kotak pelacakan.

Apabila dibandingkan dengan penelitian Purnomo (2016), terdapat beberapa perbedaan. Sistem ini dibuat dengan menggunakan bahasa pemrograman Python, sedangkan sebelumnya menggunakan bahasa pemrograman C++. Pada penelitian sebelumnya target objek yang dilacak yaitu bola ping pong sedangkan penelitian yang dilakukan oleh penulis target objek yang dilacak yaitu manusia dengan warna kulit wajah manusia sebagai acuan pelacakannya. Selain itu, pengujian yang dilakukan pada sistem ini berbeda dengan sebelumnya. Maka dari itu, proses yang dijalankan pada sistem ini sebagian besar berbeda dengan sistem yang sudah dibuat sebelumnya.

BAB III
METODOLOGI PENELITIAN

Tahapan penelitian ditunjukkan pada Gambar 3.1 yang terdiri dari proses yang dilakukan untuk melakukan penelitian ini.



Gambar 3.1 Tahapan penelitian

Penelitian ini diawali dengan melakukan analisis kebutuhan sistem, analisis kebutuhan input, analisis kebutuhan proses dan analisis kebutuhan *output* yang dibutuhkan. Setelah hasil analisis didapatkan, kemudian pada tahap selanjutnya yaitu melakukan perancangan alat dan perancangan sistem di mana alat yang akan digunakan berfungsi sesuai dengan proses dari sistem yang akan dijalankan agar dapat menyelesaikan permasalahan. Setelah alat dan sistem dirancang, maka dilakukan pengujian alat dan pengujian sistem, kemudian setelah itu sistem diuji kecepatannya untuk mengetahui seberapa baik sistem yang dikembangkan.

3.1 Analisis Kebutuhan Sistem

Analisis kebutuhan sistem merupakan tahapan di mana tahapan ini bertujuan untuk mengidentifikasi kebutuhan dan langkah-langkah apa saja untuk memecahkan permasalahan sistem pelacakan objek dengan *image processing* menggunakan metode *Mean Shift*. Harapannya penulis dapat mengetahui permasalahan yang dihadapi agar solusi yang

ditawarkan dapat sesuai dengan kebutuhan. Tahapan analisis kebutuhan sistem ini terdiri dari analisis kebutuhan input, analisis kebutuhan proses dan analisis kebutuhan *output*.

3.2 Analisis Kebutuhan Input

Analisis kebutuhan input atau masukan yang dibutuhkan oleh sistem yang akan dikembangkan yaitu:

1. Citra objek yang berwarna, target objeknya yaitu wajah manusia yang ingin dilacak keberadaannya.
2. Video *real-time* yang didapatkan dari kamera *webcam* secara langsung dari *frame* pertama video diproses oleh sistem hingga frame terakhir video dengan ukuran 640 x 480.
3. Masing-masing input video berjumlah 1 orang di dalam frame
4. Citra tersebut akan digunakan sebagai data *training* dan data *testing*.

3.3 Analisis Kebutuhan Proses

Analisis kebutuhan proses bertujuan untuk mengidentifikasi langkah dari sistem yang akan dijalankan berdasarkan data input yang didapatkan dari kamera *webcam* secara langsung. Kebutuhan proses ini adalah sebagai berikut:

a. Prapemrosesan

Pada tahap ini, citra akan diperbaiki yang kemudian dapat digunakan sebagai masukan untuk proses selanjutnya. Pada tahap prapemrosesan ini terdapat metode ROI (*Region of Interest*), dilakukan untuk menentukan target objek dan bukan objek dengan warna kulit wajah manusia sebagai acuan pelacakan target objeknya.

b. Kalkulasi histogram *backprojection*

Tahap ini bertujuan untuk mengetahui sebaran piksel warna kulit target objek pada *frame* selanjutnya sampai dengan *frame* terakhir video.

c. Penerapan *Meanshift*

Tahap ini bertujuan untuk melacak wajah target objek baru pada *frame* selanjutnya sampai dengan *frame* terakhir dari video.

d. *Marking* target objek

Tahap ini bertujuan untuk menandai daerah warna kulit target objek baru yang sudah terlacak sebelumnya agar pengguna dapat mengetahui lokasi keberadaan target objek.

e. Kontrol penggerak kamera

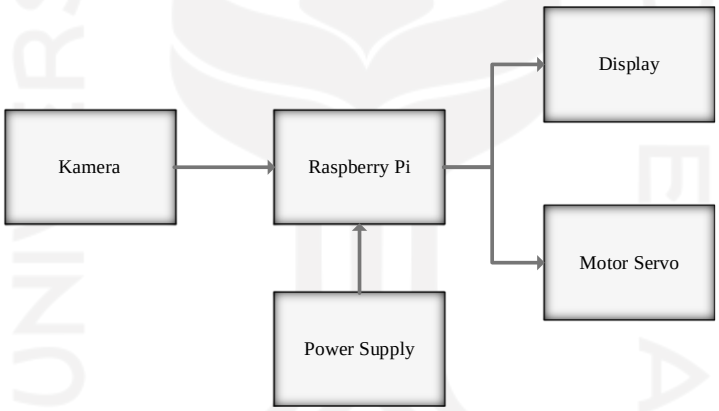
Tahap ini dilakukan untuk dapat menggerakkan kamera agar target objek tetap terlacak oleh sistem dalam jangkauan 0 – 180 derajat.

3.4 Analisis Kebutuhan *Output*

Analisis kebutuhan *output* berisi informasi yang didapatkan dari proses yang sudah dijalankan oleh sistem. Informasi ini berupa video dengan target objek yang sudah ditandai, kemudian akan ditampilkan pada layar monitor. Setelah itu kamera akan bergerak mengikuti target objek secara horizontal dalam jangkauan 180 derajat, sehingga dapat digunakan sebagai alat keamanan pada area kasir *minimarket* atau toko kelontong.

3.5 Perancangan Alat

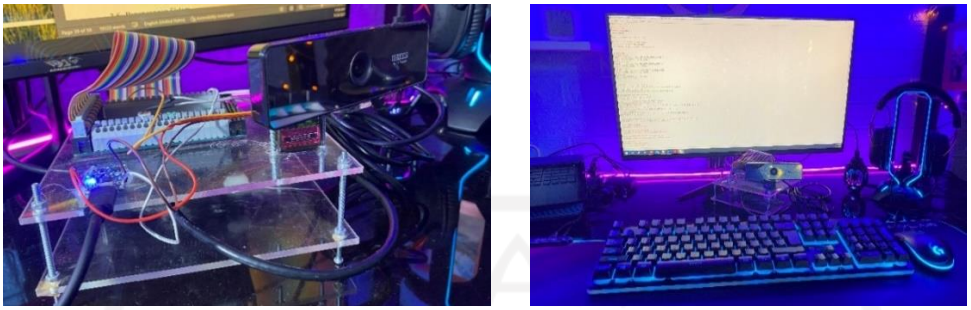
Penulis merancang alat yang akan digunakan untuk penelitian. Gambar 3.2 merupakan blok diagram perancangan perangkat keras pada penelitian ini.



Gambar 3.2 Blok diagram perancangan perangkat keras

Kamera mengirim data citra berupa video secara *real-time* ke Raspberry Pi *frame* pertama sistem akan memilih target objek dengan posisi kotak deteksi sudah diinisialisasi sebelumnya. Kemudian penulis melakukan pemrograman pada Raspberry Pi menggunakan perangkat lunak Geany (perangkat lunak bawaan Linux) dengan bahasa pemrograman Python serta menggunakan *library* OpenCV untuk mengolah citra berupa video yang dikirimkan oleh kamera dan objek yang telah dipilih oleh pengguna. Hasil pengolahan citra dijadikan parameter oleh Raspberry Pi sebagai data penentu untuk dapat menggerakkan kamera menggunakan motor servo dalam rentang 0° – 180° derajat secara horizontal. *Display*

berguna untuk memonitor objek yang dipilih dan pengguna juga dapat memilih target pada *display* tersebut. Rancangan alat pada penelitian ini dapat dilihat pada Gambar 3.3.

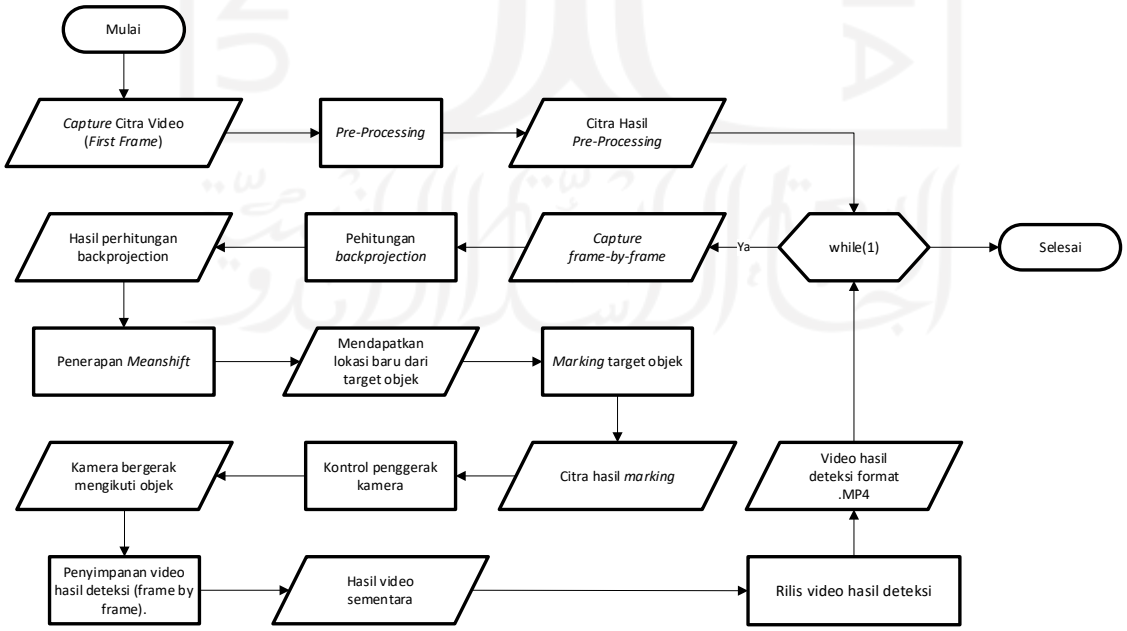


(a) (b)

Gambar 3.3 Rancangan alat

3.6 Perancangan Sistem

Perancangan sistem dilakukan untuk menggambarkan perencanaan pengembangan sistem agar dapat lebih terstruktur dan mempermudah peneliti untuk mengimplementasikan sistem yang akan dikembangkan. Perancangan sistem pada penelitian ini menggunakan *flowchart* untuk menggambarkan atau mengilustrasikan setiap proses yang dilakukan oleh sistem itu sendiri. *Flowchart* tersebut terdiri dari gambaran input, proses dan *output* dari sistem yang sudah dirancang.



Gambar 3.4 *Flowchart* perancangan sistem

Seperti pada Gambar 3.4, *flowchart* pada penelitian ini memiliki tujuh proses yang dijalankan oleh sistem yaitu *pre-processing*, perhitungan *backprojection*, penerapan *Meanshift*, *Marking* target objek, kontrol penggerak kamera, penyimpanan video hasil deteksi (*frame by frame*), dan rilis video hasil deteksi. Proses ini diawali dengan input atau masukan citra *frame* pertama dari video, hasil *output* atau keluarannya berupa kamera yang dapat bergerak mengikuti objek yang ditandai dan tersimpannya video hasil rekaman deteksi sistem dengan format .MP4.

3.6.1 *Pre-Processing*

Pre-processing dilakukan untuk memperbaiki citra agar bagian-bagian yang tidak diperlukan pada citra yang dimasukan akan dihilangkan. Kamera yang terhubung dengan Raspberry Pi akan merekam daerah sekitar dan akan menghasilkan citra beresolusi 640 x 480 piksel. *Frame* pertama dari video tersebut diambil, kemudian citra yang telah diperoleh akan dijadikan data input. Target objek yang ada pada *frame* pertama video akan dijadikan parameter oleh sistem dengan ciri target objek yang dipilih yaitu berwarna putih. Hal ini dilakukan agar sistem dapat melacak target objek pada *frame* selanjutnya.

A. Menentukan ROI

ROI adalah bagian dari citra yang dipilih menjadi target objek. Untuk menentukan ROI ada beberapa proses yang dilalui sistem agar target objek memiliki ciri yang dapat digunakan menjadi parameter pada *frame* selanjutnya, yaitu:

a. Memotong Citra

Setelah lokasi awal target objek ditetapkan, untuk menentukan ROI hal pertama yang dilakukan adalah memotong citra dengan ukuran yang disesuaikan dengan resolusi *frame* video. Pada penelitian ini ukuran ditentukan yaitu pada input video *real-time* yang didapatkan dari kamera *webcam* secara langsung dengan resolusi 640 x 480, target objek harus berada pada koordinat x 240, koordinat y 105, dengan lebar 169 dan tinggi 270. Sehingga citra yang dihasilkan yaitu bagian yang dianggap menjadi target objek.

b. Konversi Citra

Citra yang sudah dipotong kemudian dikonversi dari citra yang berwarna RGB menjadi HSV, hal ini sering disebut dengan *color space*. *Color space* merupakan sebuah metode agar warna yang terdapat pada citra dapat dikhususkan, diciptakan dan divisualisasikan. *Output* atau keluaran dari proses konversi citra ini berupa citra berwarna HSV. Intensitas cahaya

pada area kasir *minimarket* atau toko kelontong tidak selalu terang atau tidak selalu redup, maka ruang warna HSV ini digunakan karena masih dapat merepresentasikan objek dengan jelas pada intensitas cahaya terang dan redup dibandingkan dengan ruang warna BGR.

c. Segmentasi

Proses selanjutnya yaitu segmentasi, pada OpenCV proses segmentasi disebut dengan *masking* digunakan untuk membedakan antara target objek dan bukan target objek dengan menggunakan fungsi *inRange()*. Pada penelitian ini *threshold* yang digunakan adalah batas atas dan batas bawah dari warna kulit wajah manusia. Keluaran dari proses segmentasi ini adalah citra dengan warna monokrom (hitam dan putih) di mana untuk warna hitam bukan target objek dan warna putih adalah target objek.

d. Kalkulasi Histogram

Kalkulasi histogram pada OpenCV menggunakan fungsi *calcHist()* untuk menghitung histogram dengan mengambil parameter dari citra HSV. Kemudian salah satu komponen HSV yaitu *hue* diambil, karena komponen *hue* memiliki rentang warna yang dapat membedakan antara daerah warna kulit dan bukan warna kulit. Komponen *hue* ini akan dihitung histogramnya, karena yang dihitung hanya sebagian dari citra maka proses *mask* yang sudah dilakukan sebelumnya dijadikan parameter dengan ukuran histogram 180. Hal ini bertujuan agar tidak ada kepadatan pada objek lain yang ada pada citra yang diolah, maka harapannya tidak terjadi kesalahan pengenalan objek.

e. Normalisasi Nilai

Setelah hasil kalkulasi histogram didapatkan, maka nilainya dinormalisasikan dengan normalisasi *min-max*. Normalisasi MIN MAX pada sistem ini digunakan untuk memperkecil rentang nilai terkecil dan nilai terbesar agar dapat mempercepat proses pembelajaran pada suatu klasifikasi. Jumlah piksel diperkecil rentang nilainya menjadi ukuran 1 byte yaitu jangkauan antara 0 – 255.

3.6.2 Kalkulasi Histogram *Backprojection*

Proses penerapan kalkulasi histogram *backprojection* ini berawal dari pengambilan citra *frame by frame*. Pada setiap *frame* dilakukan konversi citra dari BGR menjadi HSV. Kemudian dilakukan kalkulasi *histogram backprojection* untuk mengetahui sebaran piksel dari target objek pada *frame* selanjutnya. Hasil dari citra *pre-processing* sebelumnya dijadikan parameter pada proses kalkulasi histogram *backprojection*.

3.6.3 Penerapan *Meanshift*

Penerapan *Meanshift* di sini digunakan untuk melacak lokasi dari target objek. *Meanshift* memproses pelacakan target objek dengan mengambil hasil dari kalkulasi *histogram backprojection*. Kemudian target objek yang sudah diinisialisasi sebelumnya juga dijadikan parameter pada penerapan *Meanshift*. Kriteria penghentian pencarian target objek yaitu dengan maksimal 10 iterasi atau berpindah ke wilayah kepadatan piksel yang lebih tinggi/maksimal setidaknya 1 poin.

3.6.4 *Marking* Target Objek

Hal pertama yang dilakukan adalah inisialisasi koordinat x dan y yang baru, ukuran lebar dan tinggi yang sama dengan inisialisasi target objek pada *frame* pertama menjadi kotak pelacakan baru pada *frame* selanjutnya. Kemudian setelah itu sistem menggambar/membuat kotak pelacakan hasil pelacakan target objek pada proses penerapan *Meanshift*. Keluaran dari proses ini yaitu persegi panjang atau kotak pelacakan dapat berpindah sesuai dengan pergerakan target objek beserta keluaran nilai koordinat x dan y yang baru.

3.6.5 Kontrol Penggerak Kamera

Hasil pelacakan yang sudah dilakukan pada proses *marking*, diambil nilai koordinat x dari target objek yang baru untuk dijadikan acuan gerak motor servo. Aturan gerak servo ini ditentukan titik tengahnya, titik tengah ini didapatkan dari inisialisasi target objek pada *frame* pertama. Keluaran dari proses ini yaitu kamera dapat bergerak secara horizontal mengikuti pergerakan target objek dalam jangakauan 0° – 180°.

3.6.6 Penyimpanan Video Hasil Deteksi

Hasil deteksi target objek dari setiap *frame* video dilakukan proses penyimpanan video dalam format .MP4, 30 FPS dengan ukuran resolusi video 640 x 480 ketika pengguna menekan tombol “r” pada *keyboard*. Keluaran dari proses ini yaitu berupa video sementara, karena selama pengguna belum menekan tombol “e” pada *keyboard*, maka video akan disimpan sementara dan tidak dapat diputar.

3.6.7 Rilis Video Hasil Deteksi

Setelah pengguna menekan tombol “e” pada *keyboard*, hasil deteksi sistem akan disimpan pada *folder* yang sama dengan keberadaan *source code* dari program penelitian ini.

3.7 Pengujian Alat

Pengujian alat dilakukan untuk memastikan bahwa perangkat keras yang digunakan sudah sesuai.

a. Kamera *webcam* diuji pada intensitas cahaya terang dan pada intensitas cahaya redup dengan mengeluarkan hasil pengujian yaitu hasil foto, peringkat kualitas, *frame rate*, jenis aliran, mode gambar, *webcam MegaPixels*, resolusi, standar video, rasio aspek, ukuran *file* PNG, kecepatan bit, jumlah warna, keringanan, kecerahan, warna, dan kejenuhan.
b. Motor servo diuji seberapa kecil dan besar sudut putarnya. Motor servo yang digunakan pada penelitian ini dapat digunakan, apabila motor servo mampu berputar sesuai dengan perintah sistem. Di mana sistem mengatur derajat servo menjadi *pulse* atau pulsa dengan nilai 0 derajat sama dengan 50 *pulse*.

3.8 Pengujian Sistem

Agar pada pengujian sistem ini ada perbedaan jika menggunakan motor servo dengan tidak, maka penulis menggunakan 2 data input dengan masing-masing target objek berada pada jarak sekitar 50 cm sampai dengan 300 cm, yaitu:

a. Video pengujian format .mp4, dengan ukuran 1280 x 720 dan
b. Video *real-time* yang didapatkan dari kamera *webcam* NYK NEMESIS yang terhubung dengan Raspberry Pi dengan ukuran 640 x 480.

Hal ini dilakukan untuk mengetahui kinerja dari sistem yang dikembangkan menggunakan metode IOU. Seperti yang sudah dijelaskan sebelumnya, IOU memanfaatkan kotak pelacakan hasil prediksi sistem dan kotak pelacakan yang seharusnya. Hasil dari pelacakan sistem *frame by frame* disimpan dengan format .png. Kemudian penulis memasukkan citra hasil pelacakan tersebut, setelah itu dilakukan inisialisasi koordinat x dan koordinat y dari target objek yang seharusnya, ukuran panjang dan lebar yang sama dengan *frame* pertama. Kemudian menggambar/membuat kotak pelacakan yang seharusnya, setelah itu hasilnya disimpan dalam format .png.

Untuk mendapatkan nilai IOU, kotak pelacakan hasil prediksi sistem dan kotak pelacakan yang seharusnya diiris. Kemudian kotak pelacakan hasil prediksi sistem dan kotak pelacakan yang seharusnya digabung. Setelah itu, area irisan dibagi dengan area gabungan sesuai dengan persamaan IOU. Hasil pelacakan sistem dikatakan baik jika nilai IOU sama dengan 0.7330 seperti yang dikatakan Cowton, Kyriazakis, & Bacardit (2019).

3.9 Pengujian Kecepatan Sistem

Pengujian kecepatan sistem pada penelitian ini menggunakan 3 data input yaitu:

a. Video pengujian format .mp4 dengan ukuran 1280 x 720,
b. Video *real-time* yang didapatkan dari kamera *webcam* NYK NEMESIS yang terhubung dengan Raspberry Pi dengan ukuran 640 x 480, dan
c. Video *real-time* yang didapatkan dari kamera *webcam* bawaan dari laptop ROG GL552VX dengan ukuran 640 x 480.

Pengujian kecepatan sistem dilakukan pada 2 perangkat yaitu, laptop dengan sistem operasi Windows dan Raspberry Pi dengan sistem operasi bawaan yaitu Linux. Hal ini dilakukan untuk mengetahui perbedaan performa kecepatan sistem yang dikembangkan dalam menjalankan setiap proses yang ada. Untuk data input video pengujian dan video *real-time* yang didapatkan dari kamera *webcam* bawaan laptop, pengujian kecepatan sistem dimulai dari sistem mengakses video sampai dengan proses penerapan *Meanshift*. Kemudian untuk data input video *real-time* yang didapatkan dari kamera *webcam* NYK NEMESIS yang terhubung dengan Raspberry Pi, pengujian kecepatan sistem dimulai dari sistem mengakses video sampai dengan proses kontrol penggerak kamera, yang kemudian akan didapatkan total waktu keseluruhan proses.

BAB IV
HASIL DAN PEMBAHASAN

4.1 Implementasi Sistem

Proses yang telah dirancang sebelumnya menggunakan *flowchart* pada Gambar 3.4 diimplementasikan ke dalam kode program dengan menggunakan bahasa pemrograman Python, pada bab ini akan dijelaskan fungsi dan hasil yang didapatkan dari setiap kode program pada setiap proses yang dijalankan oleh sistem.

Sebelum menjalankan sistem, penulis melakukan *import library* terlebih dahulu untuk memproses seluruh program yang ada pada sistem. Berikut penjelasan setiap *library* yang digunakan pada sistem:

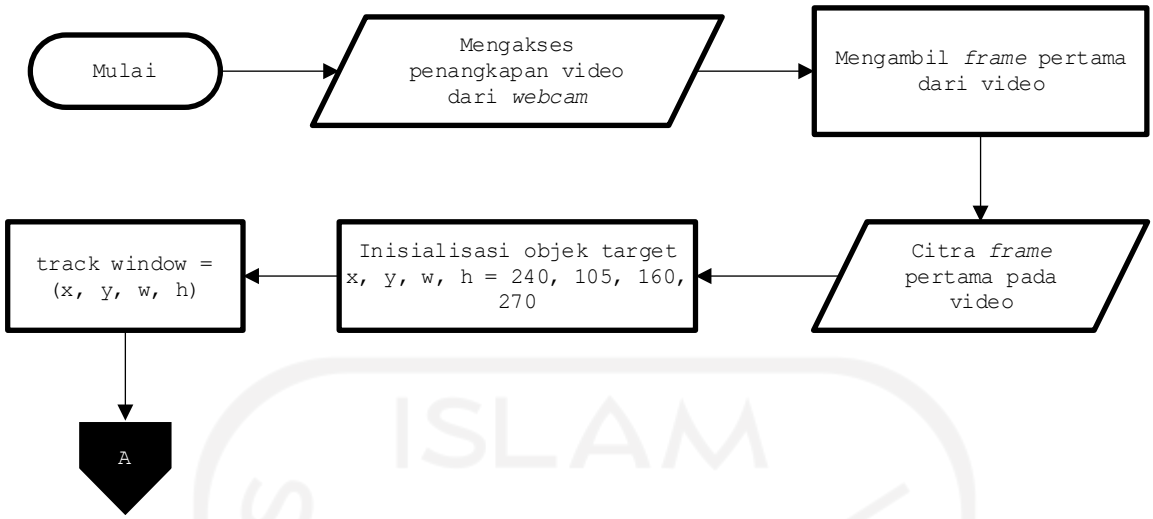
a. *NumPy* (*Numerical Python*) merupakan *library* Python berfokus pada *scientific computing* yang memiliki kemampuan membentuk objek *ndarray* (*n-dimmensional array*) yang mirip dengan *list*, namun konsumsi memorinya lebih kecil dan *runtime* yang lebih cepat dibandingkan dengan *list*.
b. CV2 merupakan *library* OpenCV versi 2, *library* utama yang digunakan oleh penulis untuk membuat sistem pelacakan objek. *Library* ini dibangun untuk menyediakan infrastruktur umum untuk aplikasi visi komputer.
c. WiringPi merupakan *library* yang digunakan penulis untuk mengakses GPIO, agar servo yang terhubung dengan Raspberry Pi dapat diakses dan berjalan.

```
import numpy as np
import cv2 as cv
import wiringpi
```

Gambar 4.1 *Library* yang digunakan pada sistem

4.1.1 Inisialisasi Lokasi Pelacakan

Objek target pada penelitian ini yaitu wajah manusia yang bergerak di depan kamera, maka tahap pertama pada penelitian ini harus mempersiapkan data input atau masukan. Di mana data masukan pada penelitian ini yaitu berupa video *real-time*, maka yang akan diambil yaitu data citra per-*frame* selama sistem melakukan *capture*.



Gambar 4.2 *Flowchart* inisialisasi lokasi pelacakan

Pada Gambar 4.2 menunjukkan bahwa sebelumnya kamera diakses terlebih dahulu untuk dilakukan pengambilan video yang kemudian *frame* pertama dari video tersebut. Dikarenakan pengguna menggunakan data input dari kamera *webcam* dengan sistem *plug and play* maka penulis mengisi nilainya dengan angka 0 pada baris kode `cap = cv.VideoCapture(0)`. Jika data input berupa *file* video dengan format AVI atau MP4 pada sistem operasi Linux diisi dengan nama dan format *file* yang akan digunakan beserta lokasi *file*-nya, namun jika menggunakan sistem operasi *windows*, hanya perlu menulis nama dan format *file*, kemudian *file* video harus berada satu folder dengan *source code*.

Setelah data masukan didapatkan, objek target yang akan dilacak harus dipilih terlebih dahulu dengan cara inisialisasi dengan nilai *hardcoded* (data atau parameter yang tetap, sehingga nilai dari target tidak dapat diubah tanpa memodifikasi program). Data input video *real-time* yang didapatkan dari kamera *webcam* NYK NEMESIS yang terhubung dengan Raspberry Pi dengan ukuran 640 x 480, didapatkan titik tengah yaitu nilai variabel x (titik koordinat x) 240, y (titik koordinat y) 105, w (lebar) 160 dan variabel h (tinggi) 270. Variabel x, y, w, h ini merupakan penentu lokasi dari objek yang ingin dilacak oleh sistem yang kemudian didefinisikan dengan nama "*track_window*".

Gambar 4.3 merupakan cara mencari nilai variabel x, y, w dan h. Untuk menetapkan lokasi pertama objek pelacakan dengan cara menampilkan nilai koordinat x dan y pada titik yang akan dilacak. Kemudian setelah nilai x dan y didapatkan, penulis mencari nilai w dan h menggunakan titik koordinat kedua. Berikut cara penulis untuk mendapatkan nilai w dan h:

$x_1 = 472$

$y_1 = 37$

$x_2 = 764$

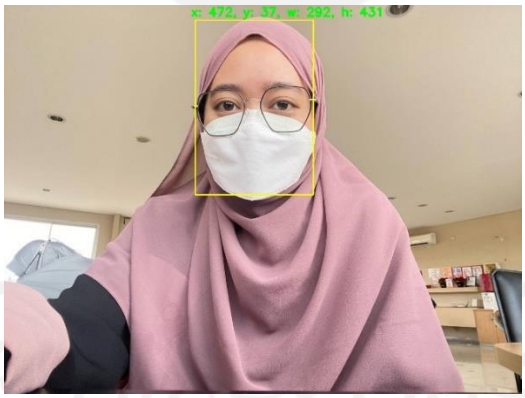
$y_2 = 468$

$w = x_2 - x_1 = 764 - 472 = 292$

$h = y_2 - y_1 = 468 - 37 = 431$

Gambar 4.3 Hasil pencarian titik koordinat x dan y

Setelah nilai x, y, w, h didapatkan, sistem memproses nilai tersebut menjadi parameter target pelacakan seperti pada Gambar 4.4.



Gambar 4.4 Hasil *setup* lokasi pelacakan

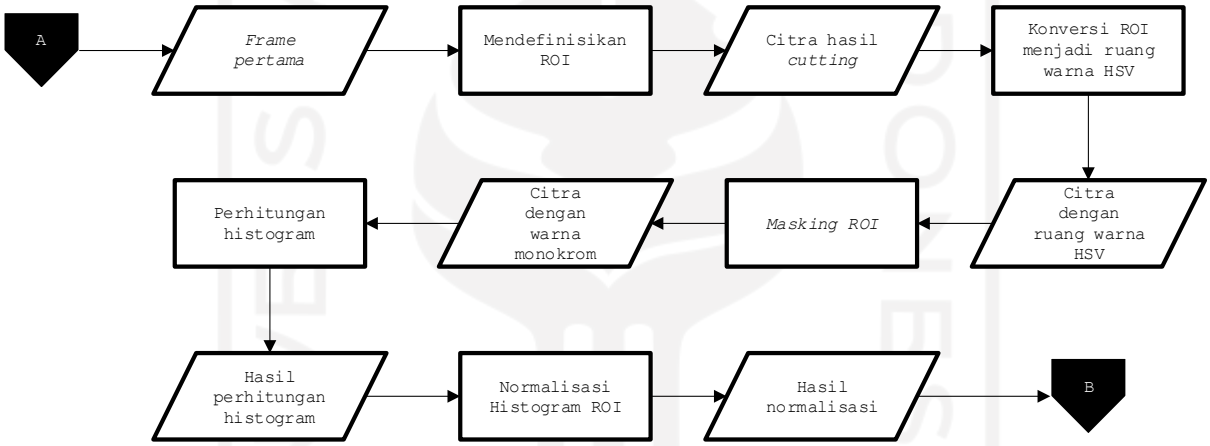
```
# Opens a camera for video capturing.
cap = cv.VideoCapture(0)
# Take first frame of the video
ret, frame = cap.read()
# Setup initial location of window
x, y, w, h = 240, 105, 160, 270
track_window = (x, y, w, h)
```

Gambar 4.5 Kode program inisialisasi lokasi pelacakan

Gambar 4.6 Hasil pengambilan *frame* pertama berupa citra warna RGB

4.1.2 Menentukan ROI (*Region of Interest*)

Berikut ini merupakan *flowchart* pada proses untuk menentukan ROI untuk pelacakan target:



Gambar 4.7 *Flowchart* untuk menentukan ROI (*Region of Interest*)

A. Mendefinisikan Posisi Pelacakan

Setelah *frame* pertama dari video didapatkan dan posisi target objek sudah diinisialisasi, maka tahap selanjutnya yaitu memotong citra pada bagian yang sudah diinisialisasi sebelumnya. Proses ini bertujuan untuk menentukan wilayah objek target agar hanya pada bagian yang dipotong saja yang diproses pada tahap penentuan ROI. Dengan cara mendefinisikan terlebih dahulu posisi dari objek target dengan variabel ROI di mana *frame* pertama dari data input video *real-time* diambil diisi dengan `[y:y+h, x:x+w]` yang merupakan posisi dari objek target atau *window* yang akan dipotong.

```
# set up the ROI for tracking
roi = frame[y:y+h, x:x+w]
```

Gambar 4.8 Kode program mendefinisikan posisi pelacakan

Gambar 4.9 Hasil *setup* ROI sebagai objek pelacakan

B. Konversi ROI Menjadi Ruang Warna HSV

Kode program pada Gambar 4.10 menunjukkan proses konversi ROI dengan ruang warna BGR menjadi ruang warna HSV, pada kondisi perubahan intensitas cahaya hasil dari konversi ruang warna HSV dapat membedakan citra objek yang dipilih dengan objek yang lain melalui kriteria tertentu yang dapat dilacak oleh sistem. Kriteria yang digunakan pada penelitian ini yaitu berupa warna kulit manusia bagian wajah. Pada *library* OpenCV, format citra berwarna bukan dalam bentuk RGB melainkan dalam bentuk BGR. Di sini sistem mengubah warna BGR (*Blue, Green, Red*) ke dalam ruang warna HSV. Hal ini dilakukan karena pada proses selanjutnya masukan data citra harus berwarna HSV untuk diekstraksi salah satu komponennya yaitu komponen *hue*. Alasan lain menggunakan ruang warna HSV ini karena dapat dilihat pada Gambar 4.13 citra dengan intensitas cahaya redup, ruang warna HSV masih dapat merepresentasikan objek dengan jelas sedangkan pada ruang warna BGR objek tidak terlalu jelas.

```
# convert ROI to the HSV color space
hsv_roi = cv.cvtColor(roi, cv.COLOR_BGR2HSV)
```

Gambar 4.10 Kode program konversi BGR menjadi HSV

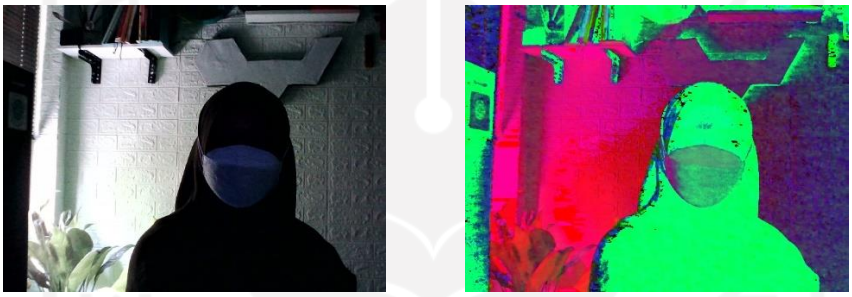
(a)

(b)

Gambar 4.11 Hasil konversi citra ROI; (a) citra BGR (b) citra HSV

(a) (b)

Gambar 4.12 Konversi citra dengan intensitas cahaya terang; (a) RGB (b) HSV



(a) (b)

Gambar 4.13 Konversi citra intensitas cahaya redup; (a) RGB (b) HSV

C. Segmentasi

Setelah warna BGR dikonversi menjadi warna HSV, sistem menjalankan proses segmentasi di mana pada OpenCV proses segmentasi disebut dengan *masking*. Variabel *mask_roi* diisi menggunakan fungsi `inRange` untuk membedakan antara target objek dan bukan target objek. Parameter pertama yaitu citra dengan ruang warna HSV, kemudian parameter kedua dan ketiga akan menjadi batas bawah dan batas atas dalam bentuk *topple* dengan menggunakan *library* NumPy.

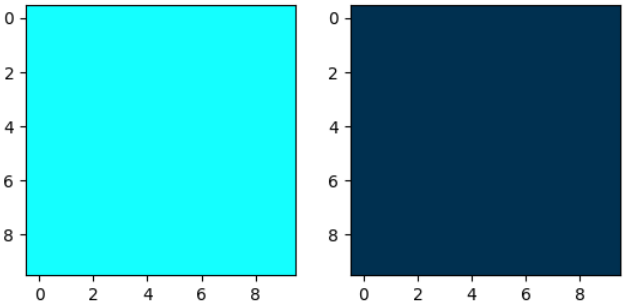
Hal ini dilakukan karena pada proses ini akan menentukan target objek yang ditandai dengan warna putih dan bukan target objek ditandai dengan warna hitam. *Masking* di sini mengandalkan nilai *threshold*, angka *threshold* tersebut didapatkan dari hasil penelitian sebelumnya yang dilakukan oleh Rosebrock (2014), di mana ketiga nilai tersebut terdiri dari nilai masing-masing komponen HSV yaitu *hue*, *saturation*, *value* dengan warna kulit wajah manusia yang dijadikan warna putih. Namun untuk studi kasus ini hasilnya di-*tunning* kembali sampai dengan mendapatkan hasil segmentasi warna kulit wajah yang dominan

dibandingkan dengan warna lain. Gambar 4.17 merupakan warna yang diambil oleh penulis.

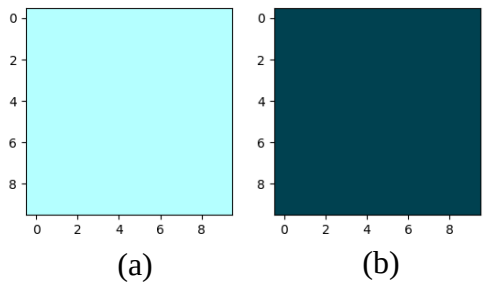
```
# calculate the mask
mask_roi = cv.inRange(hsv_roi, np.array((0., 65., 50.)), np.array((180., 255., 255.)))
```

Gambar 4.14 Kode program konversi BGR menjadi HSV

Gambar 4.15 Hasil proses segmentasi warna kulit wajah



Gambar 4.16 Warna yang diambil; (a) warna batas atas (0., 48., 80.) (b) warna batas bawah (20., 255., 255.)



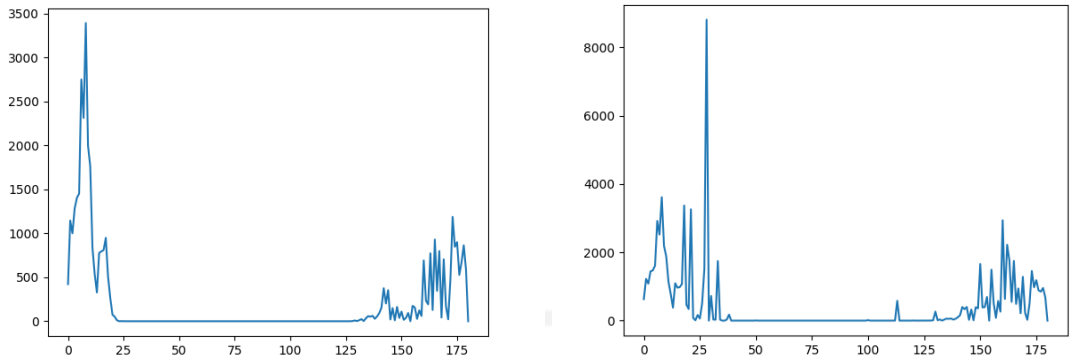
Gambar 4.17 Warna yang diambil; (a) warna batas atas (0., 65., 50.) (b) warna batas bawah (180., 255., 255.)

Penulis menggunakan 2 nilai *threshold* yaitu nilai *threshold* yang digunakan pada penelitian yang dilakukan oleh Rosebrock (2014) pada Gambar 4.16 dan Gambar 4.17 hasil *threshold* yang di-*tunning* oleh penulis, untuk pembanding dan mencari nilai yang terbaik untuk mengeliminasi warna yang bukan merupakan objek target.

Gambar 4.18 Hasil segmentasi dengan warna batas atas (0., 48., 80.), warna batas bawah (20., 255., 255.)

Gambar 4.19 Hasil segmentasi dengan warna batas atas (0., 65., 50.), warna batas bawah (180., 255., 255.)

Pada studi kasus ini, sistem menentukan objek berdasarkan warna kulit wajah manusia yang digunakan objek dengan mengabaikan latar belakang sekitar objek, karena warna kerudung yang digunakan objek mirip dengan warna kulit maka itu termasuk objek pelacakan. Pada Gambar 4.18 keseluruhan warna masker tereliminasi, namun pada bagian warna kulit wajah target objek hanya dapat dideteksi sebagian. Pada Gambar 4.19, hampir keseluruhan warna kulit wajah terdeteksi. Dari kedua ambang batas di atas penulis memilih ambang batas yang kedua dengan batas atas (0., 65., 50.) dan batas bawah (180., 255., 255.), karena ciri yang terdeteksi dan dibutuhkan oleh sistem lebih banyak dari ambang batas dengan nilai batas atas (0., 48., 80.) dan batas bawah (20., 255., 255.).



(a) (b)

Gambar 4.20 Hasil kalkulasi histogram; (a) menggunakan *mask* (b) tidak menggunakan *mask*

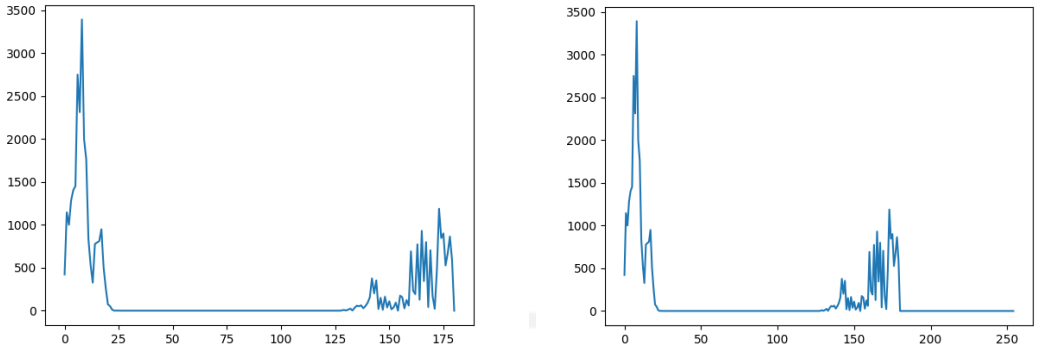
Dapat dilihat pada Gambar 4.20, hasil kalkulasi histogram yang telah melalui proses *mask*, sebaran piksel dalam rentang 0 – 21 dan 132 – 180 lebih sedikit sebaran pikselnya dibanding dengan hasil kalkulasi yang tidak melalui proses *mask*, sebaran pikselnya lebih banyak dari rentang 0 – 180.

D. Kalkulasi Histogram

Kalkulasi histogram pada OpenCV menggunakan fungsi `calcHist`. Variabel `roi_hist` diisi dengan menggunakan fungsi `calcHist` dengan parameter pertama yaitu citra dengan ruang warna HSV, parameter kedua merupakan nilai dari *channel* yang akan diambil yaitu *channel hue* di mana itu adalah *channel* pertama dari ruang warna HSV maka ditulis angka 0, parameter ketiga yaitu *mask* yang diambil dari proses segmentasi yang sudah dilakukan sebelumnya, kemudian pada parameter keempat merupakan ukuran histogram yang dimulai dari 0 sampai dengan 180, karena setelah 180 tidak ada lagi kemunculan intensitas piksel seperti pada Gambar 4.22 (b). Hal ini bertujuan agar tidak ada kepadatan pada objek lain yang ada pada citra yang diolah, maka harapannya tidak terjadi kesalahan pengenalan target objek.

```
# calculate histogram a value
roi_hist = cv.calcHist([hsv_roi], [0], mask, [180], [0, 180])
```

Gambar 4.21 Kode program proses perhitungan histogram



(a) (b)

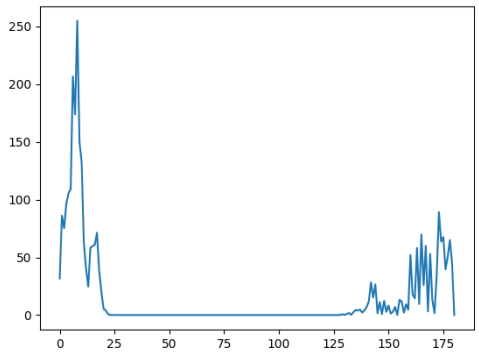
Gambar 4.22 Hasil histogram citra ROI komponen *hue*

E. Normalisasi Nilai

Tahap selanjutnya yaitu menormalisasikan nilai dengan fungsi `normalize`, fungsi ini membutuhkan beberapa nilai dari sumbernya. Di mana sumbernya itu adalah variabel `roi_hist`, tujuan (tujuan ini sama karena hanya ingin mengganti nilai dari `roi_hist`), nilai alfa (nilai 0), nilai beta (nilai 255), dan parameter terakhir yaitu tipe dari normalisasi di sini menggunakan normalisasi *min-max*. Normalisasi *min-max* pada sistem ini digunakan untuk memperkecil rentang nilai terkecil dan nilai terbesar agar dapat mempercepat proses pembelajaran pada suatu klasifikasi dengan mengubah data terkecil dan data terbesar secara linier (Reynaldo et al., 2020). Dalam hal ini rentang diperkecil menjadi ukuran 1 byte yaitu jangkauan antara 0 – 255.

```
# normalize these value
roi_hist_norm = cv.normalize(roi_hist, roi_hist, 0, 255, cv.NORM_MINMAX)
```

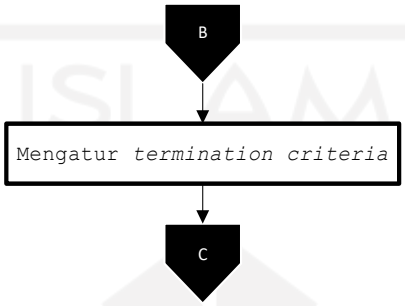
Gambar 4.23 Kode program proses normalisasi



Gambar 4.24 Histogram hasil normalisasi min max

Pada Gambar 4.22 nilai maksimum dari data hasil kalkulasi histogram pada citra yang sudah melalui proses *mask* adalah 3400 dan pada Gambar 4.24 nilai maksimumnya adalah 255. Dari proses normalisasi *min-max* ini sistem dapat mengurangi nilai-nilai yang terlalu besar pada fitur yang memiliki pengaruh terbesar pada proses klasifikasi.

4.1.3 Menentukan Batasan Kriteria



Gambar 4.25 *Flowchart* untuk mengatur kriteria penghentian iterasi

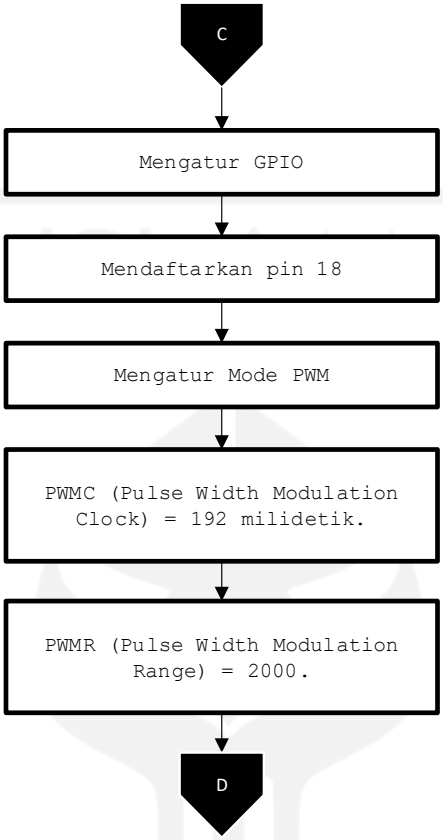
Pada tahap ini proses penerapan *Meanshift* untuk mendeteksi target objek pada *frame* selanjutnya dibatasi dengan maksimal 10 iterasi atau berpindah ke wilayah kepadatan piksel yang lebih tinggi/maksimal setidaknya 1 poin. Dengan cara mendefinisikan terlebih dahulu dengan nama `term_crit`.

```
# Setup the termination criteria, either 10 iteration or move by atleast 1 point
term_crit = ( cv.TERM_CRITERIA_EPS | cv.TERM_CRITERIA_COUNT, 10, 1 )
```

Gambar 4.26 Kode program batasan kriteria

4.1.4 Mengatur Penamaan GPIO

Gambar 4.27 Merupakan *flowchart* untuk mengatur GPIO untuk dapat menggerakan motor servo:



Gambar 4.27 *Flowchart* untuk mengatur kriteria penghentian iterasi

Agar kamera dapat bergerak sesuai dengan pergerakan objek, maka perlu mengatur GPIO dengan mendaftarkan pin yang terhubung dengan servo. Untuk mengakses pin GPIO yang ada pada Raspberry Pi, di sini penulis menggunakan *library* WiringPi dengan mengatur penamaannya terlebih dahulu. Kemudian menetapkan pin 18 yang digunakan PCM (*Pulse Code Modulation*) untuk memberikan sinyal *clock* ke perangkat audio eksternal. Output PWM0 dari GPIO 18 ini sangat berguna untuk mengontrol perangkat dengan pengaturan waktu yang sangat spesifik. GPIO 18 di sini menjadi *output* atau keluaran dari PWM (*Pulse Width Modulation*) yang kemudian mode PWM diatur menjadi tipe milidetik.

Penulis mengatur waktu 192 milidetik dengan jarak 2000 milidetik. Angka tersebut berasal dari perhitungan frekuensi dari PWM yang ditunjukkan pada Persamaan (4.1), di mana servo berjalan pada frekuensi 50Hz maka dari itu PWMC (*Pulse Width Modulation*

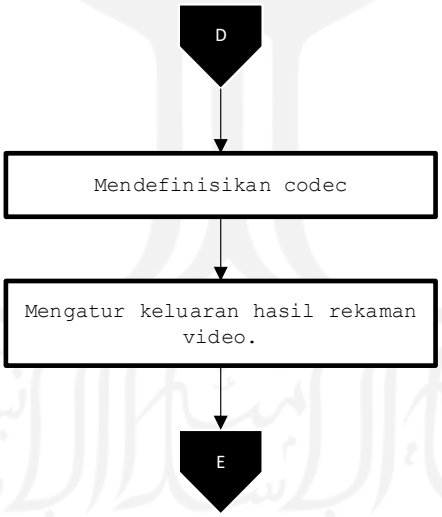
Clock) yaitu 192 dan PWMR (*Pulse Width Modulation Range*) yaitu 2000, berdasarkan frekuensi 19.2 MHz ke Hz menjadi 19.2 x 10^6.

$$PWM\ Frequency = \frac{19.2\ \text{MHz}}{\text{PWMC x PWMR}} \qquad (4.1)$$

```
# use 'GPIO naming'
wiringpi.wiringPiSetupGpio()
# set #18 to be a PWM output
wiringpi.pinMode(18, wiringpi.GPIO.PWM_OUTPUT)
# set the PWM mode to milliseconds stype
wiringpi.pwmSetMode(wiringpi.GPIO.PWM_MODE_MS)
# divide down clock
wiringpi.pwmSetClock(192)
wiringpi.pwmSetRange(2000)
```

Gambar 4.28 Kode program mengatur penamaan objek

4.1.5 Mengatur Format Penyimpanan Video



Gambar 4.29 *Flowchart* untuk mengatur kriteria penghentian iterasi

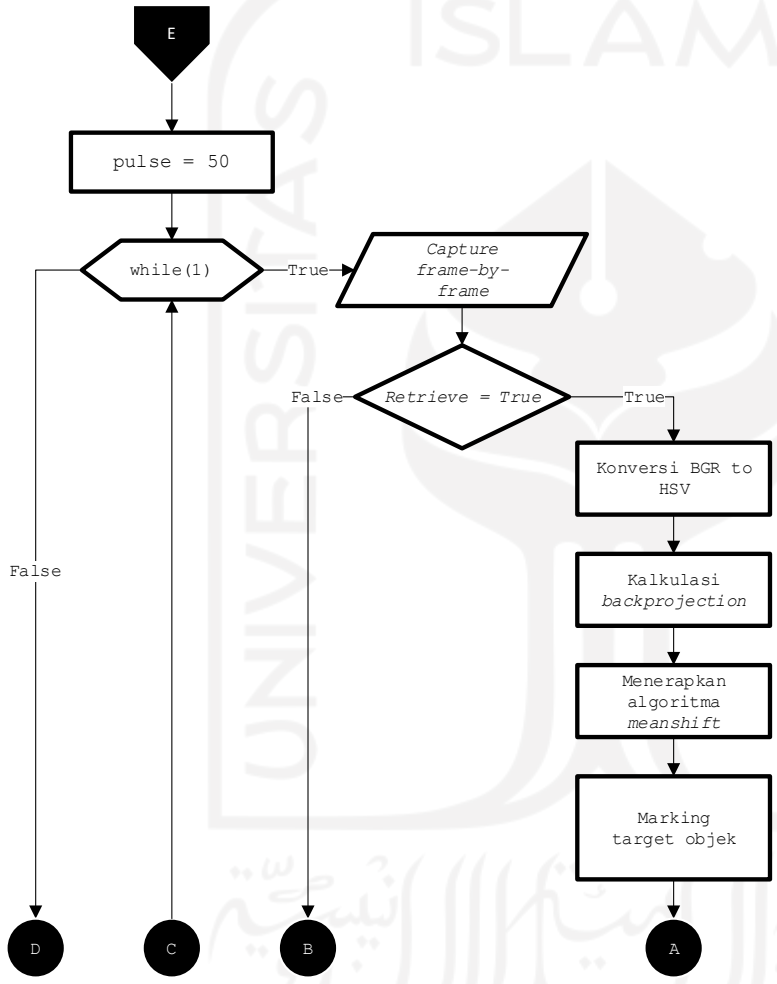
Agar kamera dapat menyimpan video hasil rekaman deteksi target objek, maka perlu mengatur format penyimpanan video dengan cara mendefinisikan *codec* / *codec-decoder* (*compressor-decompressor* yang berguna untuk menyimpan data video dalam ukuran sekecil mungkin) dengan format *file* video yaitu MP4. Kemudian mengatur keluaran hasil rekaman video dengan isi nama *file* video, *codec*, kecepatan FPS video, dan ukuran resolusi video.

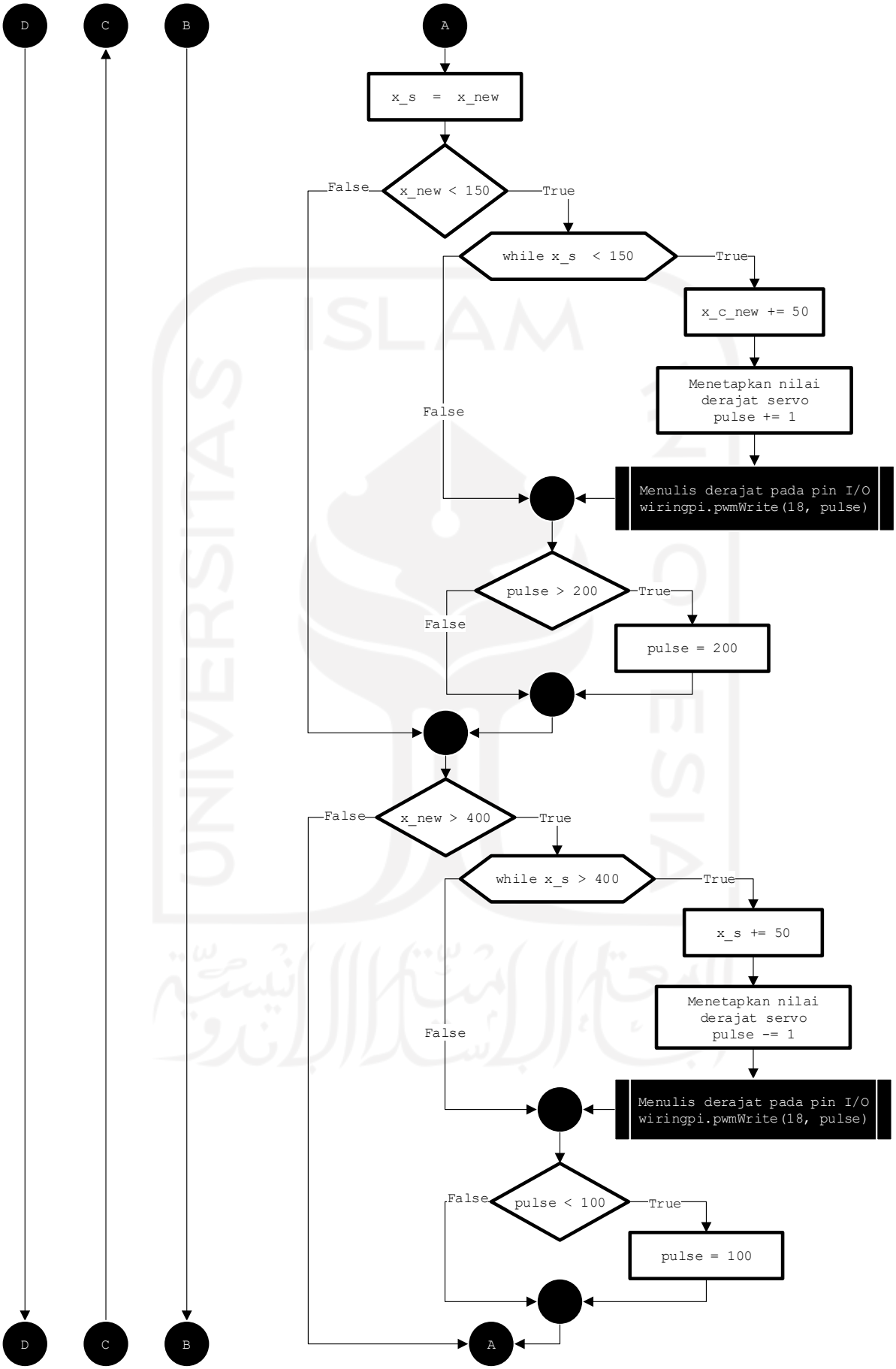
```
# Define the codec and create VideoWriter object
fourcc = cv.VideoWriter_fourcc(*'MP4V')
out = cv.VideoWriter('HASIL PENGUJIAN.mp4', fourcc, 30.0, (640, 480))
```

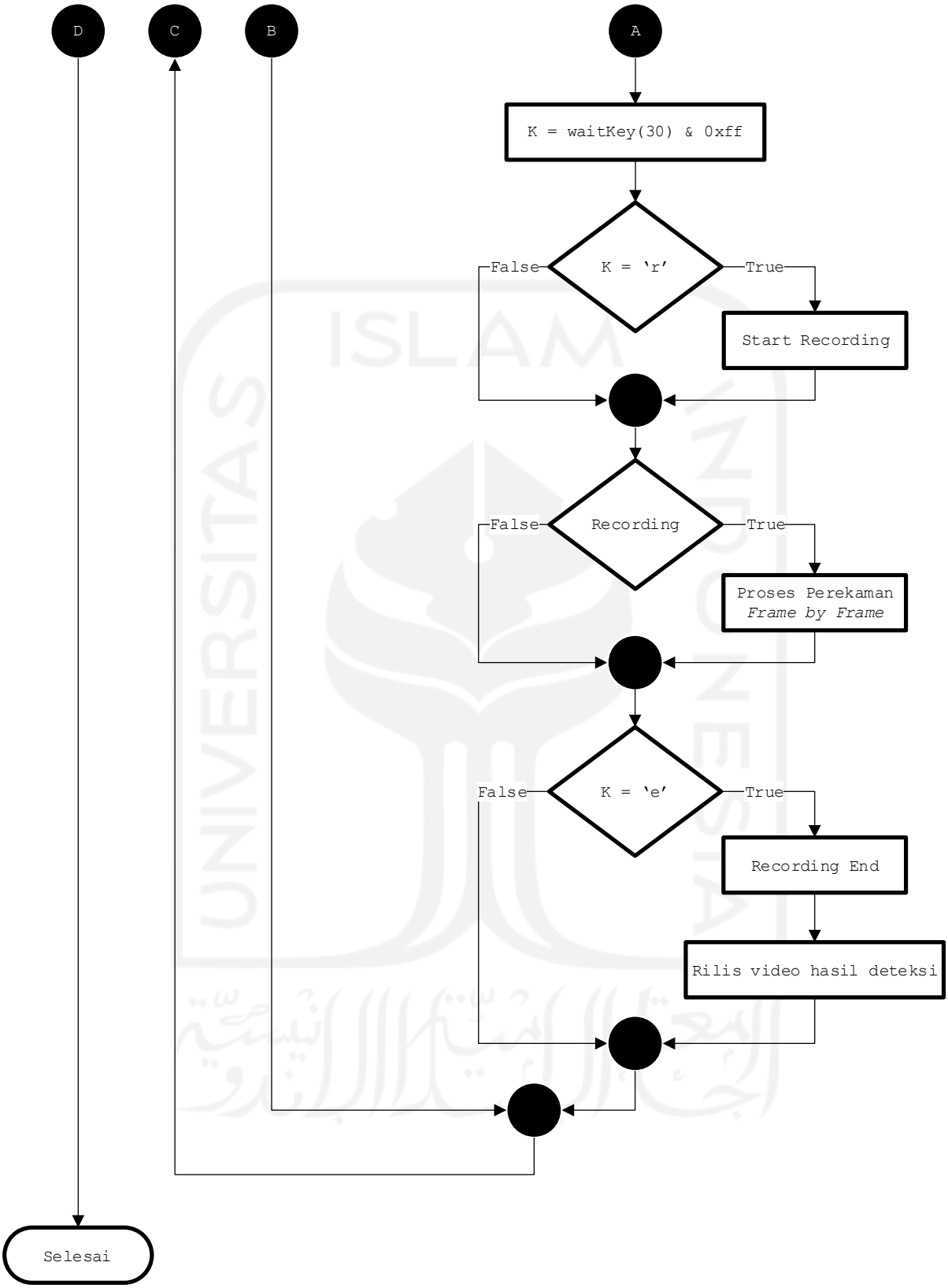
Gambar 4.30 *Flowchart* untuk mengatur kriteria penghentian iterasi

4.1.6 Pelacakan Objek

Gambar 4.31 merupakan *flowchart* pada proses pelacakan objek.




D
C
B
A
x_s = x_new
x_new < 150
False
True
while x_s < 150
True
x_c_new += 50
Menetapkan nilai derajat servo pulse += 1
False
Menulis derajat pada pin I/O wiringpi.pwmWrite(18, pulse)
pulse > 200
True
False
pulse = 200
x_new > 400
False
True
while x_s > 400
True
x_s += 50
Menetapkan nilai derajat servo pulse -= 1
False
Menulis derajat pada pin I/O wiringpi.pwmWrite(18, pulse)
pulse < 100
False
True
pulse = 100
D
C
B
A



Gambar 4.31 Diagram alur program pelacakan objek dengan menerapkan *Mean Shift*

Sebelum menerapkan algoritma *Mean Shift*, penulis menetapkan nilai *pulse* yang akan digunakan pada proses kontrol gerak servo. Nilai *pulse* sama dengan 50 yang artinya 0 derajat servo adalah 50 *pulse*. Kemudian dilakukan perulangan `while(1)`, baris pertama sistem membaca *capture frame by frame*. Karena sebelumnya sudah ditetapkan target objeknya, jadi selama sistem membaca informasi dari kamera maka proses akan terus berjalan sampai sistem sudah tidak mengambil *frame* atau tidak menerima informasi dari video yang direkam oleh kamera.

Untuk menerapkan algoritma *Mean Shift*, sistem harus memiliki masukan data berupa citra yang memiliki ruang warna HSV, maka diperlukan melakukan proses konversi terlebih dahulu untuk setiap *frame* yang dibaca dari warna BGR menjadi ruang warna HSV. Setelah proses konversi selesai, maka perlu dihitung citra dengan menggunakan kalkulasi *backprojection* yang didefinisikan dengan nama `dst`. Hal ini dilakukan dengan mengambil komponen atau *channel hue* dari ruang warna HSV.

Kemudian setelah proses kalkulasi selesai maka diperlukan untuk menerapkan algoritma *Mean Shift* dengan membaca informasi dari kamera (`ret`) dan membaca tanda pelacakan target `track_window_new` yang kemudian memproses kalkulasi *backprojection*, `track_window` ini merupakan lokasi *window* yang sudah diatur sebelumnya pada proses inisialisasi lokasi pelacakan yang kemudian diterapkan pada *frame* yang baru dengan batasan kriteria *Mean Shift* yaitu mendeteksi target objek dengan maksimal 10 iterasi atau berpindah ke wilayah kepadatan piksel yang lebih tinggi/maksimal setidaknya 1 poin. Algoritma *Mean Shift* di sini dilakukan untuk mendapatkan lokasi target yang baru, karena objeknya manusia jadi lokasi target selalu berpindah walau pun hanya 1 poin saja.

Agar servo dapat bergerak ke kiri dan ke kanan harus diketahui titik tengah servo yang ditentukan dari ukuran piksel kamera yang digunakan. Input video *real-time* dari kamera *webcam* NYK NEMESIS yang terhubung dengan Raspberry Pi berukuran 640 x 480, yang berarti titik tengahnya yaitu koordinat x 320 dan koordinat y 240. Inisialisasi target objek pada *frame* pertama yaitu x 240, y 105, w 160 dan h 270, kemudian nilai variabel w dan h diambil. Untuk mendapatkan nilai koordinat x dan y titik tengah dari kotak deteksi, penulis mencarinya dengan cara:

$mid\ x = 320$

$mid\ y = 240$

$\frac{w}{2} = \frac{160}{2} = 80$

$$\frac{h}{2} = \frac{270}{2} = 135$$

$$mid\ x + 80 = 400$$

$$mid\ x - 80 = 240$$

$$mid\ y + 135 = 375$$

$$mid\ y - 135 = 105$$

Jadi nilai titik tengah servo adalah 240 < x < 400, di mana jika nilai *x* kurang dari 240 maka servo akan bergerak ke kanan 0 – 90 derajat, kemudian jika nilai *x* lebih dari 400 maka servo akan bergerak ke arah kiri 91 – 180 derajat.

X_s merupakan perkiraan langkah servo untuk mengikuti target objek, nilai awal x_s berupa nilai piksel yang didapatkan dari referensi x_new. Penggunaan x_s untuk mempermudah motor servo melacak target objek. Motor servo akan bergerak ke kanan atau ke kiri secara linier tergantung dengan nilai referensi awal x_new, jika kurang dari 240 servo mengarah ke kanan, kemudian jika lebih besar dari 400 servo akan mengarah ke kiri. Angka 50 dari x_s ditentukan dari hasil *trial*, di mana setiap servo bergerak 1 derajat maka jarak piksel yang dapat dijangkau oleh servo 0 – 50 piksel. Ketika nilai x_s keluar dari *range* 240 < x_s < 400, maka nilai x_new akan di*copy* ke nilai x_s sebagai referensi. Proses ini akan berulang sampai dengan nilai 240 < x_s < 400.

Untuk kode program `wiringpi.pwmWrite(18, pulse)` digunakan untuk men-*generate pulse* sesuai dengan karakteristik *pulse* servo, di mana *pulse* dapat disesuaikan menjadi 0 - 10% dari 20 milidetik panjang *pulse* atau 0 – 2 ms *pulse on*, 2 – 20 ms *pulse off*. Variabel pulse 50 – 250 untuk membangkitkan pulse on 0 – 2 ms pada rentang *pulse* 20 ms. Untuk dapat mencapai maksimal *pulse* sebanyak 10%, maka dilakukan pembatasan dengan *pulse* maksimal 250 dan untuk menghilangkan nilai *error* pada variabel *pulse* maka dilakukan pembatasan minimal nilai *pulse* = 50.

Untuk menyimpan video hasil deteksi, pengguna harus menekan tombol r pada *keyboard* untuk memulai *recording* atau perekaman. Karena hasil video rekaman pada OpenCV terbalik, maka perlu menggunakan fungsi `flip` dengan *flip code* yaitu +180 yang berarti membalik sumbu x dan nilai positif pada setiap *frame* video. Kemudian jika ingin menghentikan perekaman maka pengguna cukup menekan tombol e pada *keyboard*. Setelah itu, hasil rekaman video pelacakan target objek akan dirilis pada *folder* yang sama dengan keberadaan *source code* program. Deteksi target objek akan terus berlanjut sampai dengan pengguna menutup atau *close* program yang dijalankan.

```
pulse = 50
while(1):
    # Capture frame-by-frame
    ret, next_frame = cap.read()

    if ret == True:
        hsv = cv.cvtColor(next_frame, cv.COLOR_BGR2HSV)
        dst = cv.calcBackProject([hsv],[0],roi_hist,[0,180],1)

        # apply mean shift to get the new location
        ret, track_window_new = cv.mean shift(dst, track_window, term_crit)

        # Draw it on image
        x_new, y_new, w, h = track_window_new
        mean shift_tracking = cv.rectangle(next_frame, (x_new , y_new), (x_new +
w, y_new + h), (0, 255, 255), 2)
        cv.imshow('MEAN SHIFT TRACKING', mean shift_tracking)

        #Servo Tracking, untuk mengintegrasi antara koordinat object dengan alat.
        # 0-180 degree = 50 - 250 degree
        x_s = x_new
        if (x_new < 240):
                while x_s < 240:
                        x_s += 50
                        pulse += 1 #nilai derajat servo
                        wiringpi.pwmWrite(18, pulse)
                if pulse > 200:
                        pulse = 200
        if (x_new > 400):
                while x_s > 400:
                        x_s -= 50
                        pulse -= 1
                        wiringpi.pwmWrite(18, pulse)
                if pulse < 100:
                        pulse = 100

        k = cv.waitKey(30) & 0xff
        # SAVE VIDEO
        if k == ord('r') and recording is False:
            print('recording start')
            recording = True

        if recording:
```

```
            frame = cv.flip(meanshift_tracking, +180)
            out.write(frame)

        if k == ord('e'):
            print('recording end')
            recording = False
            out.release()
    else:
        break
```

Gambar 4.32 Kode program pelacakan objek

4.2 Pengujian Alat

4.2.1 Pengujian Kamera

Kamera yang digunakan pada sistem ini menggunakan kamera *webcam* NYK NEMESIS EVEREST, pengujian kamera dilakukan secara daring pada *website* https://id.webcamtests.com/ dalam keadaan instensitas cahaya redup dan terang, hasil informasi yang didapatkan terdapat pada Tabel 4.1.

Tabel 4.1 Pengujian kamera *webcam* NYK NEMESIS EVEREST

Informasi Webcam		
	NYK NEMESIS EVEREST	
Informasi	**Intensitas Cahaya Terang**	**Intensitas Cahaya Redup**
Hasil Foto		
Peringkat Kualitas	1335	2471
Frame rate	30 FPS	30 FPS
Jenis aliran	Video	Video
Mode Gambar	RGB	RGB
Webcam MegaPixels	2.07 MP	2.07 MP
Resolusi Webcam	1920x1080	1920x1080
Standar Video	FHD	FHD
Rasio Aspek	1.78	1.78
Ukuran File PNG	1.73 MB	1.65 MB

Kecepatan bit	25.13 MB/s	24.11 MB/s
Jumlah Warna	141716	124691
Keringanan	25.88%	22.55%
Kilau	25.28%	21.19%
Kecerahan	26.14%	22.61%
Warna	290°	273°
Kejenuhan	4.55%	9.57%

Tabel 4.2 Pengujian kamera *webcam* laptop ROG GL552VX

Informasi	**Webcam Laptop ROG GL552VX**
Peringkat Kualitas	832
Frame rate	30 FPS
Jenis aliran	Video
Mode Gambar	RGB
Webcam MegaPixels	0.92 MP
Resolusi Webcam	1280x720
Standar Video	HD
Rasio Aspek	1.78
Ukuran File PNG	229.55 KB
Kecepatan bit	4.69 MB/s
Jumlah Warna	10434
Keringanan	2.94%
Kilau	1.03%
Kecerahan	2.22%
Warna	235°
Kejenuhan	86.67%

Pada hasil pengujian kamera pada Tabel 4.1, kamera dapat menampilkan video secara langsung. Pada intensitas cahaya redup peringkat kualitas kamera tinggi yaitu mencapai 2471 yang artinya kamera ini memiliki kualitas kamera yang baik karena pada *website* pengujian *webcam* ini tertulis semakin baik kameranya maka semakin tinggi penilaian kualitasnya. *Webcam* ini memiliki kamera dengan sensor 2.07 MP, resolusi *webcam* 1920 x 1080, standar video FHD (*Full* HD), frame *rate* 30 FPS, jumlah warna yang dihasilkan pada intensitas

cahaya terang lebih banyak dibandingkan pada intensitas cahaya redup. Jika dibandingkan dengan hasil pengujian *webcam* laptop ROG GL552VX pada Tabel 4.2, hasilnya jauh lebih baik *webcam* NYK NEMESIS EVEREST.

4.2.2 Pengujian Servo

Pengujian perangkat keras servo dilakukan berdasarkan aturan dasar dari servo, yaitu dengan menguji derajat putarnya. Servo mengirim sinyal ke GPIO dari Raspberry Pi, kemudian sistem mengubah derajat servo menjadi *pulse*, dengan nilai 0 derajat menjadi 50 *pulse*, 180 derajat menjadi 250 *pulse*. Nilai per 1 derajat servo adalah 1,111 *pulse*, maka untuk jarak 0 – 180 derajat sama dengan 50 – 250 *pulse*. Pengujian dilakukan dengan 3 nilai derajat yaitu 0 derajat, 90 derajat dan 180 derajat. Nilai derajat servo didapatkan dari:

$0° = 50\ pulse$

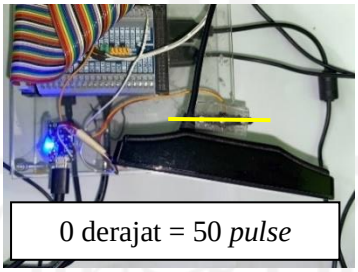
$180° = 250\ pulse$

$90° = 250 - 50 = 200$

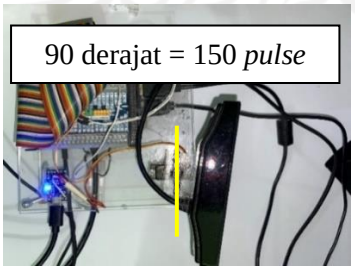
$= \frac{200}{2} + 50 = 150\ pulse$

$1° = \frac{200}{2} = 100$

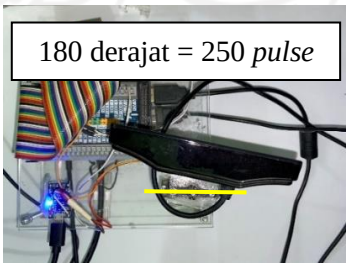
$= \frac{100}{90} = 1{,}111 + 50 = 51{,}111\ pulse$



(a)

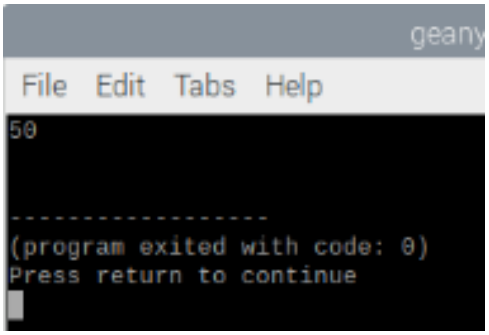


(b)

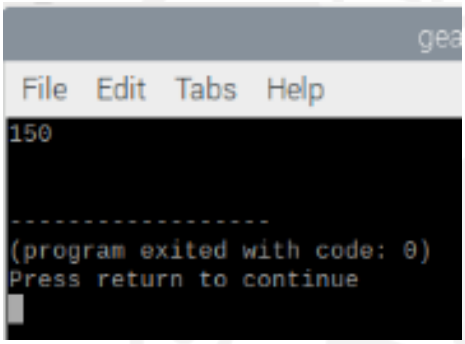


(c)

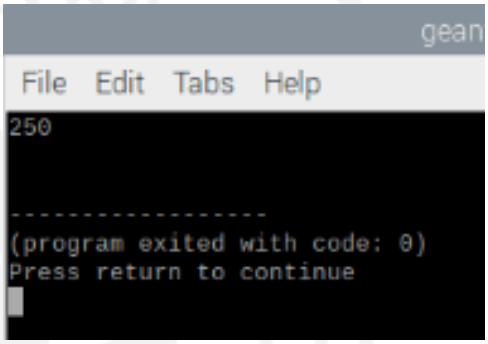
Gambar 4.33 Hasil pengujian sudut servo; (a) 0 derajat (b) 90 derajat (c) 180 derajat



(a)



(b)



(c)

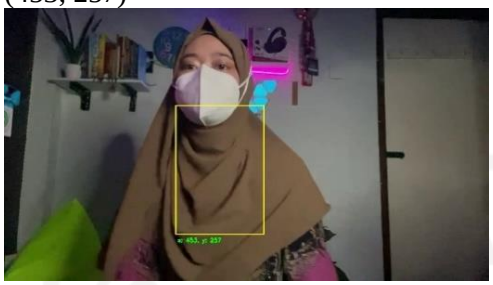
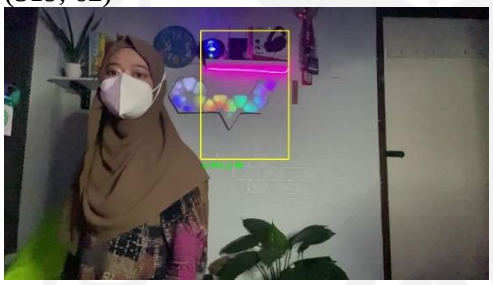
Gambar 4.34 Hasil keluaran sistem; (a) 50 *pulse* (b) 150 *pulse* (c) 150 *pulse*

Pengujian yang telah dilakukan menunjukkan bahwa respons motor servo sesuai dengan perintah sistem, di mana jika masukkan *pulse* pada sistem adalah 50 *pulse* servo bergerak pada arah 0 derajat, jika masukkan *pulse* 150 servo bergerak pada arah 90 derajat dan jika masukkan *pulse* 250 maka servo bergerak pada arah 250 derajat.

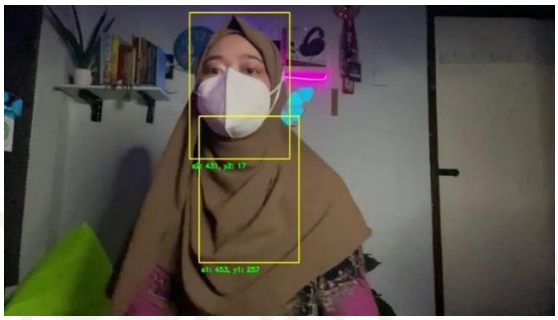
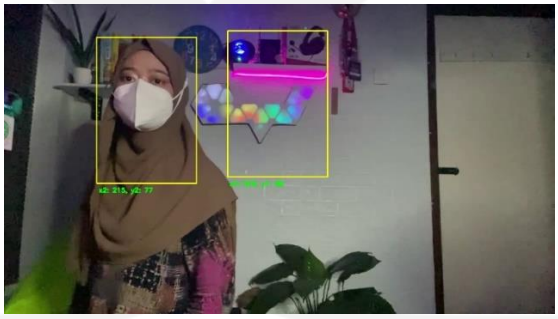
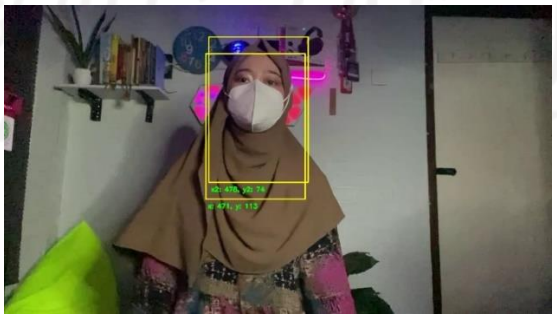
4.3 Pengujian Sistem

Pada pengujian sistem ini penulis mengambil 4 *frame* dari video pengujian yaitu *frame* ke 73, *frame* ke 288, *frame* ke 474 dan *frame* ke 835. Untuk mengevaluasi model dari deteksi objek penulis menggunakan metode IOU, di mana IOU di sini memanfaatkan *bounding box* yang terdapat pada *frame* hasil deteksi sistem. Tujuannya adalah untuk membandingkan dan mencari nilai kecocokan atau *similarity* hasil deteksi dari sistem dengan hasil deteksi yang seharusnya. Hasil deteksi yang seharusnya ini dilakukan dengan menandai target objek dengan menampilkan *window* atau kotak deteksi pada koordinat target objek.

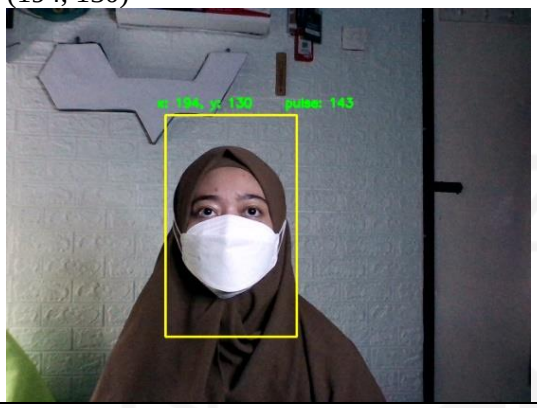
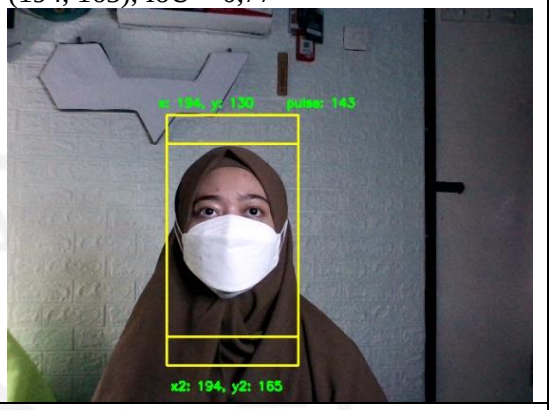
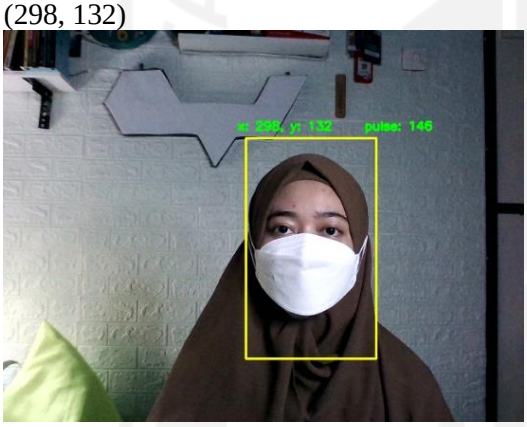
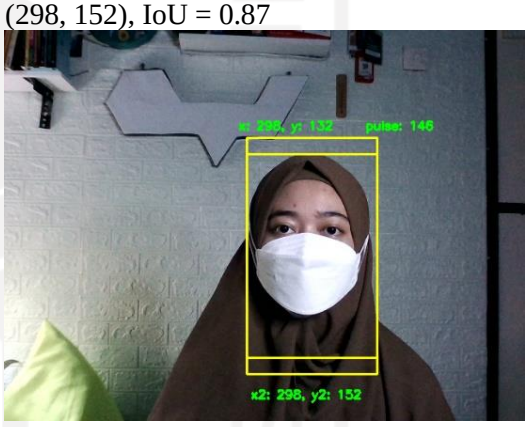
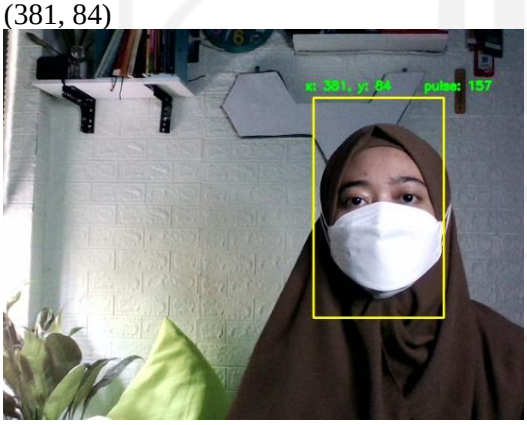
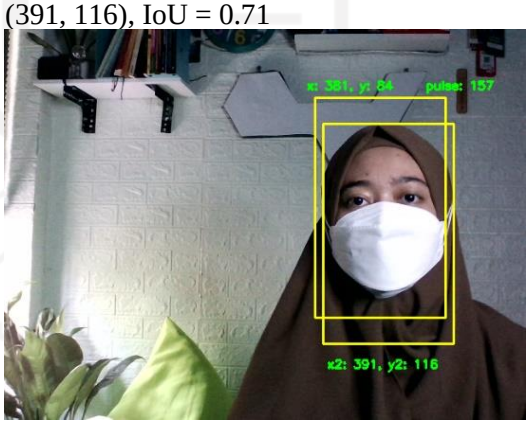
Tabel 4.3 Pengujian sistem menggunakan video pengujian

***Frame* ke-**	**Menggunakan Video Pengujian (Kamera Tidak Bergerak)**	
	Prediksi Sistem	***Ground Truth***
73	(453, 257) 	(431, 17) 
288	(500, 189) 	(825, 156) 
474	(519, 62) 	(215, 77) 
835	(471, 113) 	(478, 74) 

Tabel 4.4 Pengujian sistem menggunakan video pengujian

***Frame* ke-**	***Intersection over Union***
73	IoU = 0.15 
288	IoU = 0.0 
474	IoU = 0.0 
835	IoU = 0.75 

Tabel 4.5 Pengujian sistem menggunakan servo

No	Menggunakan Kamera Secara Langsung	
	Prediksi Sistem	**IoU**
1	(194, 130) 	(194, 165), IoU = 0,77 
2	(298, 132) 	(298, 152), IoU = 0.87 
3	(381, 84) 	(391, 116), IoU = 0.71 

Ketika nilai IoU sama dengan 0, artinya *bounding box* hasil prediksi sistem dengan bok yang seharusnya tidak menghasilkan area yang cocok. Jika nilai IoU kurang dari 0.4, maka kecocokan gagal atau sistem tidak dapat mendeteksi target objek. Kemudian jika nilai IoU

lebih dari 0.4, berarti prediksi sistem dengan boks yang seharusnya dapat dikatakan sistem dapat mendeteksi target objek (Nasution, 2020).

Pada penelitian yang dilakukan oleh Cowton, Kyriazakis, & Bacarit (2019), semakin tinggi nilai IoU, maka semakin banyak keakuratan antara kotak prediksi sistem dengan kotak yang seharusnya. Prediksi sistem dapat dikatakan akurat dalam mendeteksi objek, mereka menggunakan *threshold* dengan nilai IoU lebih dari sama dengan 0.6. Di mana pada Gambar 4.35, hasil IoU 0.4034 prediksi dikatakan buruk untuk mendeteksi target objek. Kemudian prediksi dikatakan baik jika nilai IoU 0.7330, dan prediksi sistem dengan performa yang luar biasa untuk memprediksi target objek jika nilai IoU mencapai 0.9264.

Gambar 4.35 Contoh kalkulasi IoU

Sumber: (Cowton, Kyriazakis, & Bacardit, 2019)

Dapat dilihat pada Tabel 4.3 dan Tabel 4.4, *frame* ke 73, 288 dan 474 jika target objek bergerak ke kiri atau ke kanan sistem tidak dapat mendeteksi target objek dengan nilai IoU 0.0 yang berarti sangat buruk, namun pada *frame* ke 835 target objek target berada di tengah-tengah *frame* kotak deteksi dapat mendeteksi target objek dengan nilai IoU 0.75. Pada pengujian tersebut penulis menggunakan video pengujian dengan format *file* .mp4. Terlihat perbedaannya jika sistem menggunakan servo pada Tabel 4.5, objek dapat dideteksi meskipun objek bergerak ke kiri atau ke kanan dengan nilai IoU lebih dari 0.7 yang berarti prediksi sistem dengan menggunakan servo ini lebih baik dibandingkan dengan prediksi sistem tidak menggunakan servo.

4.4 Pengujian Kecepatan Sistem

Pengujian yang dilakukan selanjutnya yaitu pengujian kinerja dari sistem berdasarkan kecepatan pada setiap pemrosesan yang dilakukan. Tujuan dilakukannya pengujian kecepatan sistem ini untuk mengetahui seberapa cepat sistem dapat memproses masukan sampai dengan menghasilkan keluaran. Masukan data berupa keseluruhan *frame* dari video pengujian dengan format *file* .mp4, resolusi 1280 x 720, kecepatan video 240 fps dengan jumlah *frame*

sebanyak 1040, masukan data dari kamera *webcam* internal laptop dengan menggunakan sistem operasi windows dengan ukuran 640 x 480 dan masukan data dari kamera yang terpasang pada Raspberry Pi ukuran 640 x 480, dengan data masukan masing-masing sebanyak 1040 *frame*.

Tabel 4.6 Pengujian kecepatan sistem

Perang-kat	**Akses Kamera / Video**		**Mengambil Citra *frame* Pertama**	**Pre-processing**	**Pelacakan Target Objek**		**Total**
					Kamera Tidak Begerak	**Kamera Bergerak 180°**	
Laptop	Kamera *webcam*	10.06	00.50	00	01:13.38	-	01:23.94
	Video pengujian	00.03	00.03	00	00:48.88	-	00:48.94
Raspberry Pi	Kamera *webcam*	00.10	00.20	00.22	00:46:11	00:46.44	00:46.96
	Video pengujian	00.06	00.07	00.18	01:12.04	-	01:12.35

Dari Tabel 4.6 dapat diketahui kecepatan sistem untuk memproses pelacakan objek menggunakan kamera *webcam* internal laptop adalah 1 menit 23.94 detik, kemudian jika menggunakan video pengujian membutuhkan waktu 48.94 detik. Namun beda halnya jika menggunakan Raspberry Pi, pelacakan target objek menggunakan kamera *webcam* secara langsung hanya membutuhkan waktu selama 46.63 detik. Kemudian jika pelacakan target dilakukan menggunakan kamera *webcam* secara langsung dalam jangkauan 180°, waktu yang dibutuhkan yaitu hanya 46.44 detik dengan total keseluruhan proses dari awal sampai akhir yaitu 46.96 detik. Jika menggunakan video pengujian pada Raspberry Pi (video pengujian ini sama dengan yang ada pada laptop penulis), membutuhkan waktu 1 menit 12.35 detik. Proses yang paling lama tentunya pada penerapan *Mean Shift* karena pada proses tersebut sistem memproses sebanyak 1039 *frame* untuk melacak target objek pada setiap *frame* tersebut.

Jadi dapat disimpulkan dari pengujian kecepatan sistem ini, bahwa total waktu yang dibutuhkan sistem untuk melacak pergerakan target objek menggunakan Raspberry Pi dengan kamera *webcam* secara langsung lebih cepat dibandingkan dengan yang lainnya seperti pada Tabel 4.6.

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Kesimpulan yang didapatkan dari penelitian ini adalah sebagai berikut:

a. Berhasil merancang sistem pelacakan target objek dengan menggunakan kamera *webcam* yang terhubung dengan Raspberry Pi sebagai komputasi sistem dan penggerak motor servo yang terhubung dengan pin digital input / *output* yang sudah diregistrasi untuk menggerakkan kamera sesuai dengan keberadaan target objek secara otomatis dalam jangkauan 180°.
b. Sistem yang diajukan sudah mampu mendeteksi target objek dengan menggunakan metode *Mean Shift*. Dengan dilakukannya konversi dari warna BGR menjadi HSV pada bagian ROI, komponen *hue* dapat diambil dan dijadikan parameter pada proses segmentasi. Pada proses segmentasi, sistem dapat menentukan target objek dan bukan target objek dengan warna putih dan hitam. Di mana untuk warna putih dianggap sebagai objek dan warna hitam dianggap bukan objek. Dikarenakan target objek yang dilacak adalah manusia, maka yang menjadi patokan warna adalah warna kulit wajah manusia. Dengan menggunakan *mask* warna kulit wajah manusia yang dipilih menggunakan rentang tertentu berdasarkan nilai ambang (*thresholding*).
c. Kamera yang digunakan pada penelitian ini dapat menampilkan video secara langsung. Pada intensitas cahaya redup peringkat kualitas kamera tinggi yaitu mencapai 2471 yang artinya kamera ini memiliki kualitas kamera yang baik dengan resolusi *webcam* 1920x1080, standar video FHD (*Full* HD), frame *rate* 30 FPS, jumlah warna yang dihasilkan pada intensitas cahaya terang lebih banyak dibandingkan pada intensitas cahaya redup.
d. Motor servo yang digunakan pada penelitian ini mampu bergerak secara horizontal 0° – 180°.
e. Sistem yang diajukan sudah mampu melacak target objek yang bergerak pada jarak 0° – 180° derajat dengan nilai IOU lebih dari 0.7 yang berarti prediksi sistem dengan menggunakan servo ini lebih baik dibandingkan dengan sistem yang tidak menggunakan servo, karena jika tidak menggunakan servo target objek yang bergerak ke kiri atau ke kanan nilai IOU-nya yaitu 0.0 yang berarti prediksi sistem sangat buruk.

f. Kecepatan sistem untuk memproses pelacakan objek menggunakan kamera *webcam* internal laptop adalah 1 menit 23.94 detik, kemudian jika menggunakan video pengujian membutuhkan waktu 48.94 detik. Namun beda halnya jika menggunakan Raspberry Pi, pelacakan target objek menggunakan kamera *webcam* secara langsung hanya membutuhkan waktu selama 46.63 detik. Kemudian jika pelacakan target dilakukan menggunakan kamera *webcam* secara langsung dalam jangkauan 180°, waktu yang dibutuhkan yaitu hanya 46.44 detik dengan total keseluruhan proses dari awal sampai akhir yaitu 46.96 detik. Jika menggunakan video pengujian pada Raspberry Pi (video pengujian ini sama dengan yang ada pada laptop penulis), membutuhkan waktu 1 menit 12.35 detik. Proses yang paling lama tentunya pada penerapan *Mean Shift* karena pada proses tersebut sistem memproses sebanyak 1039 *frame* untuk melacak target objek pada setiap *frame* tersebut.

5.2 Saran

Pada penelitian ini masih terdapat banyak kekurangan yang harus diperbaiki, maka dapat disarankan untuk mengembangkan penelitian lebih lanjut tentang proyek tugas akhir ini agar kekurangan dari penelitian ini dapat diminimalisir. Berikut merupakan saran yang dapat dipertimbangkan selanjutnya:

a. Untuk penelitian selanjutnya, parameter ciri dari target objek perlu ditambahkan atau diperbaiki dengan mengembangkan parameter ciri yang sudah ada.
b. Untuk penelitian selanjutnya, diharapkan dapat menghapus hasil rekaman pelacakan target objek sebelum memori Raspberry Pi penuh.
c. Servo yang digunakan pada penelitian ini hanya dapat bergerak ke kiri dan ke kanan dalam rentang 0° – 180°, jadi jika objek ada pada jarak lebih dari 180° sistem tidak dapat melacak objek. Kemudian jika objek berada lebih tinggi atau lebih rendah dari kamera, kamera hanya dapat menampilkan sebagian dari objek. Saran untuk penelitian selanjutnya menggunakan servo *pan tilt* dengan gerak putaran 360°.

DAFTAR PUSTAKA

Akbar, D., & Riyadi, S. (2019). Pengaturan Kecepatan Pada Motor Brushless Dc (Bldc) Menggunakan Pwm (Pulse Width Modulation), 255–262. Retrieved from https://doi.org/10.5614/sniko.2018.30

Andono, P. N., T.Sutojo, & Muljono. (2017). *Pengolahan Citra Digital*. (A. Pramesta,Ed.). Yogyakarta: ANDI (Anggota IKAPI). Retrieved from https://books.google.co.id/books?hl=en&lr=&id=zUJRDwAAQBAJ&oi=fnd&pg=PR3&dq=Pengolahan+citra+biasa+digunakan+dalam&ots=CiEjK8EZ2M&sig=-9muxho7eJKt6E2TFi_9TbyEe_k&redir_esc=y#v=onepage&q=Pengolahan citra biasa digunakan dalam&f=false

Cardani, D. (2001). Adventures in hsv space. *... de Robótica, Instituto Tecnológico Autónomo de ...*, 1–10. Retrieved from http://132.68.58.138/labs/anat/hsvspace.pdf

Cowton, J., Kyriazakis, I., & Bacardit, J. (2019). access-2933060-pp.pdf - Automated_Individual_Pig_Localisation_Tracking_And.pdf. Newcastle.

Hadiwijaya, B., Darjat, & Zahra, A. A. (2014). Perancangan Aplikasi CCTV Sebagai pemantau Ruangan Menggunakan IP Camera. *Transient*, 3 No.2(Perancangan Aplikasi CCTV), 1–6.

Hakim, M. A. I., & Putra, Y. H. (2013). PEMANFAATAN MINI PC RASPBERRY PI SEBAGAI PENGONTROL JARAK JAUH BERBASIS WEB PADA RUMAH, 6.

Handoyo, E. (2006). Perancangan Mini Image Editor Versi 1.0 sebagai Aplikasi Penunjang Mata Kuliah Digital Image Processing. *Jurnal Informatika*, 2(2), 145–154.

Mauko, I. C., & Tunliu, S. (2016). Kontrol Arah Gerak Web Kamera (Webcam) Berbasis Web. *Jurnal Ilmiah Flash*, 2(2), 106. Retrieved from https://doi.org/10.32511/jiflash.v2i2.31

Nasution, H. (2020). Implementasi Logika Fuzzy pada Sistem Kecerdasan Buatan. *ELKHA: Jurnal Teknik Elektro*, 4(2), 4–8. Retrieved from https://jurnal.untan.ac.id/index.php/Elkha/article/view/512

Purnomo, A. R. (2016). ROBOT PENGIKUT BOLA DENGAN SENSOR KAMERA MENGGUNAKAN METODE MEAN SHIFT PADA IMAGE PROCESSING.

Raspberry Pi. (n.d.). GPIO. Retrieved 7 October 2021, from

https://www.raspberrypi.org/documentation/usage/gpio/

Reynaldo, J., Adikara, P. P., & Wihandika, R. C. (2020). Analisis Sentimen Mengenai Produk Toyota Avanza Menggunakan Metode Learning Vector Quantization Versi 3 (LVQ 3) dengan Seleksi Fitur Chi Square, Lexicon-Based Features serta Normalisasi Min-Max, 4(3), 830–839. Retrieved from http://j-ptiik.ub.ac.id

Rosebrock, A. (2014). Skin Detection: A Step-by-Step Example using Python and OpenCV. Retrieved from https://pyimagesearch.com/2014/08/18/skin-detection-step-step-example-using-python-opencv/%0A

Salim, A. (2020). Intersection Over Union. Retrieved 7 June 2022, from https://medium.com/bisa-ai/intersection-over-union-a8d1532899b3

Wijayana, A., W, T. A. B., & Sa'adah, S. (2015). Analisis dan Implementasi Object Tracking Menggunakan Metode ASIFT dan Mean Shift. *E-Proceeding of Engineering*, 2(1), 1166–1176.

LAMPIRAN

FORM-TA/TF-A3

UNIVERSITAS ISLAM INDONESIA
Program Studi Informatika FTI

SARAN/USULAN PRESENTASI KEMAJUAN TUGAS AKHIR

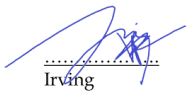
Nama Mhs. : **Fitri**
No. Mhs. : _____
Judul TA : **CCTV** _____

1. Perkuat justifikasi ilmiah pada pemilihan metode meanshift
2. Berhati-hati dalam pengujian kecepatan
3. Ukuran kesuksesan perlu diperjelas

Nilai kemajuan Tugas Akhir: _____ **(0 – 100)**
(studi pustaka, perancangan, penguasaan materi, ketepatan)

Yogyakarta, 21 Juni 2022

Dosen,



.....
Irving

Dilampirkan pada Laporan TA yang diajukan untuk pendadaran