TUGAS AKHIR - TF 181801

RANCANG BANGUN PENGHITUNG *SPARE PARTS* UNIT FOX-BABY MEREK ELITECH DENGAN METODE OBJECT COLOR TRACKING MENGGUNAKAN RASPBERRY PI 3 B

I Putu Dedi Semara Putra

NRP 5009201089

Dosen Pembimbing

Dr. Ir. Purwadi Agus Darwito, M.Sc.

NIP 196208221988031001

Departemen Teknik Fisika

Fakultas Teknologi Industri dan Rekayasa Sistem

Institut Teknologi Sepuluh Nopember

Surabaya

2022

Halaman ini sengaja dikosongkan

TUGAS AKHIR – TF 181801

RANCANG BANGUN PENGHITUNG *SPARE PARTS* UNIT FOX-BABY MEREK ELITECH DENGAN METODE *OBJECT COLOR TRACKING* MENGGUNAKAN *RASPBERRY PI 3 B*

I Putu Dedi Semara Putra
NRP 5009201089

Dosen Pembimbing
Dr. Ir. Purwadi Agus Darwito, M.Sc.
NIP 196208221988031001

Departemen Teknik Fisika
Fakultas Teknologi Industri dan Rekayasa Sistem
Institut Teknologi Sepuluh Nopember
Surabaya
2022

Halaman ini sengaja dikosongkan

FINAL PROJECT – TF 181801

DESIGN OF ELITECH BRAND FOX-BABY SPARE PARTS COUNTER WITH OBJECT COLOR TRACKING METHOD USING RASPBERRY PI 3 B

I Putu Dedi Semara Putra

NRP 5009201089

Advisor

Dr. Ir. Purwadi Agus Darwito, M.Sc.

NIP 196208221988031001

Departement of Engineering Physics

Faculty of Industrial Technology and System Engineering

Institut Teknologi Sepuluh Nopember

Surabaya

2022

Halaman ini sengaja dikosongkan

PERNYATAAN BEBAS PLAGIASI

Saya yang bertanda tangan di bawah ini.

Nama	: I Putu Dedi Semara Putra
NRP	: 5009201089
Departemen / Prodi	: Teknik Fisika / S1 Teknik Fisika
Fakultas	: Fakultas Teknologi Industri & Rekayasa Sistem (FTIRS)
Perguruan Tinggi	: Institut Teknologi Sepuluh Nopember

Dengan ini menyatakan bahwa Tugas Akhir dengan judul "**Rancang Bangun Penghitung *Spare Parts* Unit FOX-BABY Merek Elitech Dengan Metode *Object Color Tracking* Menggunakan *Raspberry Pi 3 B***" adalah benar karya saya sendiri dan bukan plagiat dari karya orang lain. Apabila di kemudian hari terbukti terdapat plagiat pada Tugas Akhir ini, maka saya bersedia menerima sanksi sesuai ketentuan yang berlaku.

Demikian surat pernyataan ini saya buat dengan sebenarnya-benarnya.

Surabaya, 02 Mei 2022

Yang membuat pernyataan,

I Putu Dedi Semara Putra

NRP. 5009201089

Halaman ini sengaja dikosongkan

LEMBAR PENGESAHAN

TUGAS AKHIR

RANCANG BANGUN PENGHITUNG *SPARE PARTS* UNIT FOX-BABY MEREK ELITECH DENGAN METODE *OBJECT COLOR TRACKING* MENGGUNAKAN *RASPBERRY PI 3 B*

Oleh:

I Putu Dedi Semara Putra

NRP. 5009201089

Surabaya,

Menyetujui,

Pembimbing I

Dr. Ir. Purwadi Agus Darwito, M.Sc.

NIP. 196208221988031001

Mengetahui,

Kepala Departemen

Teknik Fisika FT-IRS-ITS

Dr. Suyanto, S.T., M.T.

NIP. 197111131995121002

Halaman ini sengaja dikosongkan

LEMBAR PENGESAHAN

RANCANG BANGUN PENGHITUNG *SPARE PARTS* UNIT FOX-BABY MEREK ELITECH DENGAN METODE *OBJECT COLOR TRACKING* MENGGUNAKAN *RASPBERRY PI 3 B*

TUGAS AKHIR

Diajukan Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Teknik
pada
Progam Studi S-1 Departemen Teknik Fisika
Fakultas Teknologi Industri & Rekayasa Sistem (FTIRS)
Institut Teknologi Sepuluh Nopember

Oleh:

I Putu Dedi Semara Putra
NRP. 5009201089

Disetujui oleh Tim Penguji Tugas Akhir:

1. Dr. Ir. Purwadi Agus Darwito, M.Sc. ..... (Pembimbing I)
2. Prof. Totok Ruki Biyanto, S.T., M.T., Ph.D. ..... (Ketua Penguji)
3. Dr. Suyanto, S.T., M.T. ..... (Penguji I)

SURABAYA
Juli 2022

Halaman ini sengaja dikosongkan

RANCANG BANGUN PENGHITUNG *SPARE PARTS* UNIT FOX-BABY MEREK ELITECH DENGAN METODE *OBJECT COLOR TRACKING* MENGGUNAKAN *RASPBERRY PI 3 B*

Nama : **I Putu Dedi Semara Putra**

NRP : **5009201089**

Departemen : **Teknik Fisika FTIRS - ITS**

Dosen Pembimbing : **Dr. Ir. Purwadi Agus Darwito, M.Sc.**

ABSTRAK

Kesalahan perhitungan sparepart yang diakibatkan oleh *human error* dalam perhitungan manual serta sistem informasi yang belum terintegrasi merupakan permasalahan yang dialami PT. Sinko Prima Alloy sampai tahun 2022 ini. Maka dibuatlah rancang bangun penghitung *spare parts* unit FOX-BABY menggunakan metode *Object Color Tracing* dengan sistem informasi berbasis *website* serta penyimpanan data dilakukan pada database MySQL. Tempat instalasi rancang bangun adalah ruangan produksi E-9 dengan nilai kelembaban 72 %, Suhu 28.5^{o}C dan nilai intensitas cahaya ruangan 137,3 Lux. Konveyor yang digunakan untuk media penghitung memiliki kecepatan maksimal 76,570 RPM dan warna dasar sabuk hijau. Dilakukan modifikasi pada konveyor seperti penambahan kamera 1920p 2K merek *NEMESIS A96* sebagai sensor penghitung barang dan skat pembatas untuk mencegah barang tertumpuk yang lewat saat proses perhitungan. Sedangkan perangkat komputer yang digunakan sebagai pemroses gambar adalah *Raspberry Pi 3 B*. Website sitem informasi perhitungan sparepart dibuat dengan bahasa pemrograman *Python* dan *library OpenCV* serta Framework *Django* sebagai *Backend server program*. Percobaan perhitungan dilakukan pada 4 jenis *spare parts (Top Casing, Mainboard, Bottom Casing, dan Battrey Cover*) dan 6 jenis warna (Biru, Hitam, Hijau muda, Kuning, Orange dan Putih) dengan jumlah maksimal barang 20 pcs. Nilai akurasi perhitungan pada sparepart dengan warna dasar hitam, hijau muda dan putih kurang baik dengan nilai dibawah 95% sedangkan warna biru, kuning dan orange memiliki akurasi baik dengan pengukuran 100%. Faktor yang mempengaruhi akurasi perhitungan adalah *shadow* di sekitar konveyor dan warna dasar konveyor tesebut.

Kata Kunci: Penghitung *spare parts, Object Color Tracking, OpenCV, Django, Raspberry Pi 3 B*

Halaman ini sengaja dikosongkan

DESIGN OF ELITECH BRAND FOX-BABY SPARE PARTS COUNTER WITH OBJECT COLOR TRACKING METHOD USING RASPBERRY PI 3 B

Name	**:**	**I Putu Dedi Semara Putra**
NRP	**:**	**5009201089**
Department	**:**	**Engineering Physics FTIRS - ITS**
Supervisors	**:**	**Dr. Ir. Purwadi Agus Darwito, M.Sc.**

ABSTRACT

Spare parts calculation errors caused by human errors in manual calculations and information systems that have not been integrated are the problems experienced by PT. Sinko Prima Alloy until 2022. So the design of the FOX-BABY spare parts counter was made using the Object Color Tracing method with a website-based information system and data storage was carried out in a MySQL database. The design installation place is the E-9 production room with a humidity value of 72%, a temperature of 28.5oC and a room light intensity value of 137.3 Lux. The conveyor used for the counting media has a maximum speed of 76,570 RPM and the basic color of the belt is green. Modifications were made to the conveyor such as the addition of a 1920p 2K camera with the NEMESIS A96 brand as a sensor for counting goods and a barrier to prevent piled goods from passing during the calculation process. While the computer device used as an image processor is the Raspberry Pi 3 B. The spare parts calculation information system website is made with the Python programming language and the OpenCV library and the Django Framework as the Backend server program. Calculation experiments were carried out on 4 types of spare parts (Top Casing, Mainboard, Bottom Casing, and Battrey Cover) and 6 types of colors (Blue, Black, Light Green, Yellow, Orange and White) with a maximum number of 20 pcs. The value of calculation accuracy on spare parts with basic colors of black, light green and white is not good with a value below 95% while the colors blue, yellow and orange have good accuracy with a measurement of 100%. Factors that affect the accuracy of the calculation are the shadow around the conveyor and the base color of the conveyor.

Keywords: Spare parts counter, Object Color Tracking, OpenCV, Django, Raspberry Pi 3 B

Halaman ini sengaja dikosongkan

KATA PENGANTAR

Puji syukur senantiasa dipanjatkan kepada Tuhan Yang Maha Esa, karena atas berkat dan rahmat-Nya, penulis dapat menyelesaikan laporan Tugas Akhir dengan judul "Rancang Bangun Penghitung *Spare Parts* unit FOX-BABY Merk Elitech Dengan Metode *Object Color Tracking* menggunakan *Raspberry Pi 3 B*" dilakukan dalam rangka memenuhi salah satu syarat untuk mencapai gelar Sarjana. Penulis menyadari bahwa, tanpa bantuan dan bimbingan dari berbagai pihak. Oleh karena itu, penulis mengucapkan terima kasih kepada:

1. Bapak Dr. Ir. Purwadi Agus Darwito, M.Sc. selaku dosen pembimbing yang telah menyediakan waktu, tenaga, dan pikiran untuk mengarahkan penulis dalam penyusunan laporan tugas akhir ini.
2. Bapak Dedy Adi Nugroho selaku pembimbing selama di PT. Sinko Prima Alloy yang telah memberikan fasilitas, dukungan serta membimbing dalam usaha memperoleh data yang penulis perlukan untuk menyelesaikan tugas akhir ini.
3. *Stakeholder* Departemen Teknik Fisika FTIRS, ITS
4. Orang tua dan keluarga penulis yang telah memberikan bantuan dukungan material dan moral.
5. Teman-teman Teknik Fisika yang telah memberikan dukungan semangat, moral serta doa sehingga laporan tugas akhir ini dapat selesai.

Serta pihak-pihak lain yang tidak dapat disebutkan satu-persatu. Akhir kata, penulis berharap Tuhan Yang Maha Esa berkenan membalas segala kebaikan semua pihak yang telah membantu secara langsung maupun tidak langsung. Penulis menyadari dengan sepenuh hati bahwa dalam penulisan laporan Tugas Akhir ini masih terdapat banyak kekurangan, untuk itu penulis mengharapkan kritik dan saran yang membangun dari pembaca. Semoga tugas akhir ini membawa manfaat bagi pengembangan ilmu.

Surabaya, 02 Mei 2022

Penulis

Halaman ini sengaja dikosongkan

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

1.1 Latar Belakang

PT. Sinko Prima Alloy adalah salah satu perusahaan yang bergerak di bidang *electromedical device* di Indonesia dengan brand kebanggan yaitu "Elitech". PT. Sinko Prima Alloy terletak di Jl. Tambak Osowilangun No.61 Surabaya, Jawa Timur dan berdiri sejak tahun 1995 (*Elitech - PT. Sinko Prima Alloy*, n.d.). Sebagai perusahaan yang sedang berkembang, PT. Sinko Prima Alloy memiliki permasalahan terkait efisiensi kerja dan banyak human error dalam setiap proses produksi. Pada proses produksi unit FOX bayi terdapat permasalahan tentang proses perhitungan *spare parts* dan transfer informasi ke bagian *Production Planning and Inventory Control (PPIC).* Kesalahan perhitungan sparepart diakibatkan kesalahan operator dalam perhitungan manual. Kesalahan dikarenakan banyak nya jumlah sparepart dan warna unit (Ali Sukoco, Komunikasi Pribadi, 23 November 2021). Kemudian terdapat permasalahan tentang proses penyampaian data yang cukup lama ke bagian *PPIC*. Hal ini dikarenakan data harus diberikan secara estafet ke *PPIC* melalui beberapa ketua bagian (Farah Diska, Komunikasi Pribadi, 23 November 2021).

Berdasarkan dua permasalahan tersebut, perlu dilakukan perubahan perhitungan sparepart secara manual oleh manusia menjadi perhitungan otomatis oleh mesin dengan metode *Machine Vision System*. Konveyor *horizontal* memiliki keunggulan dapat lebih mudah membawa barang dengan bentuk yang tidak beraturan (Thorat, 2015). Konveyor *belt* memiliki beberapa keunggulan seperti dapat melakukan perhitungan membawa barang dengan kapasitas yang cukup banyak dan memiliki keunggulan effisiensi waktu(Sachdeva, 2017). Efisiensi waktu dapat dicapai dengan pengaturan motor penggerak dari suatu sistem sesuai dengan kebutuhan(Chengrui Li et al., 2017). Bahkan penelitian dari (Aosoby et al., 2016) menjelaskan bahwa konveyor dengan berat diatas 2000 ton sampai 2700 ton dapat dibuat dengan harga yang dapat bersaing dan kontruksi 80% lokal. PT. Sinko Prima Alloy memiliki konveyor tipe horizontal berukuran 298 cm x 95 cm yang dibeli pada tahun 2019. Konveyor tersebut dapat di atur kemiringannya dan memiliki belt yang berwarna hijau. Namun konveyor ini memiliki beberapa kelemahan yaitu konveyor ini tidak dapat melakukan perhitungan jumlah barang yang terlewat secara otomatis dan konveyor ini tidak memiliki sistem informasi yang dapat menyalurkan data ke end user (PPIC) secara direct.

Dari penelitian yang dilakukan oleh (Bayangin and Karakose, 2017) penggunaan Machine Vision System memiliki beberapa kelebihan yaitu biaya yang cukup murah, tingkat presisi dan akurasi yang tinggi. Dimana Perangkat Computer yang digunakan pada penelitian tersebut adalah Raspberry Pi 3 B dengan metode Otsu Threshold. Metode Otsu Threshold ini merubah warna object menjadi hitam putih kemudian dilakukan penentuan luas daerah objek yang akan dihitung sampai dengan proses perhitungannya. Sedangkan pada penelitian (Singh et al., 2015) metode yang digunakan adalah Object Color Tracking dimana metode ini dapat digunakan untuk perhitungan dan pemilahan warna object secara bersamaan. Pemilahan warna dilakukan untuk mendeteksi warna yang berbeda sesuai dengan set point yang ditentukan kemudian dilanjutkan dengan pehitungan benda yang melewati titik hitung. Penelitian yang dilakukan oleh (Xu et al., n.d.) tentang perhitungan kendaraan menggunakan threshold OpenCV dan bounding *Yolo* menjadi solusi untuk sistem perhitungan barang secara otomatis menggunakan image processing, namun object yang dihitung terbatas sesuai dengan *library Yolo*. Isi dari *library Yolo* seperti identifikasi manusia, binatang, motor, mobil dan lain-lain (Objek umum bukan khusus). Identifikasi warna dan luasan dapat dilakukan menggunakan library OpenCV pyhton dimana dalam penelitian yang dilakukan oleh (Binti et al., n.d.) dimana dalam penelitan tesebut akurasi dari minimum dan maksimum warna sangat berpengaruh ke hasil pembacaan luasan area pixel data. Dari beberapa permasalahan lapangan dan beberapa penelitian sebelumnya dilakukan penelitian tentang proses perhitungan dengan *machine vision system* dengan judul "Rancang Bangun Penghitung Sparepart unit FOX-BABY Merk Elitech Dengan Metode *Object Color Tracking* menggunakan Raspberry Pi 3 B". Diharapkan penelitian ini dapat menyelesaikan permasalahan tentang human error pada proses perhitungan sparepart dan delay pada proses pengiriman data. Dimana penelitian ini telah disetujui oleh manager teknik dan direktur utama PT. Sinko Prima Alloy

1.2 Rumusan Masalah

Berdasarkan latar belakang diatas, maka rumusan masalah pada penelitian ini adalah:

a) Bagaimana merancang bangun penghitung sparepart FOX-BABY merek Elitech sesuai dengan *Bill of Material (BOM)* secara otomatis pada konveyor *belt* menggunakan metode *Object Color Tracking*?
b) Bagaimana membuat sistem informasi yang terintegrasi dengan website berbasis *database MySQL* pada *Raspberry Pi 3 B*?

1.3 Tujuan

Tujuan dilakukannya penelitian ini adalah:

a) Membuat rancang bangun penghitung sparepart FOX-BABY merek Elitech sesuai dengan Bill of Material (BOM) secara otomatis pada konveyor *belt* menggunakan metode *Object Color Tracking*
b) Membuat sistem informasi yang terintegrasi dengan website berbasis *database MySQL* pada *Raspberry Pi 3 B*

1.4 Lingkup Kajian

Batasan masalah pada penelitian ini adalah:

a) *Raspberry Pi 3 B* sebagai server dan pemroses data
b) Pengolahan citra yang digunakan berbasis warna
c) Perhitungan berfokus pada satu warna dan satu jenis *spare part*
d) *Image processing* berfokus pada pemilahan warna, bentuk, tulisan dan area pixel gambar
e) *Database* yang digunakan adalah *MySQL*
f) *User interface* yang digunakan adalah *website* dengan menggunakan *back-end Djanggo*
g) Bahasa pemrograman yang digunakan *Phyton, JavaScript, SQL, HTML* dan *CSS*

1.5 Sistematika Laporan

Laporan ini terdiri dari 5 BAB, antara lain:

a) BAB I Pendahuluan

   Menjelaskan mengenai latar belakang masalah, tujuan, manfaat, batasan masalah, dan sistematika laporan

b) BAB II Tinjauan Pustaka dan Dasar Teori

   Berisikan tentang *review* penelitian sebelumnya dengan materi terkait dan beberapa dasar teori dari berbagai sumber. Dasar teori menjelaskan tentang Unit FOX Bayi merek Elitech, *Citra, (Pixel, Resolusi, dan Intensitas)*, Model warna *RGB*, *Raspberry pi model 3 B*, *Conveyor belt* dan *Object Color Tracking*.

c) BAB III Metodologi

   Menjelaskan tentang metode penelitian, parameter penelitian, rincian kerja prosedur penelitian, serta alat dan bahan yang digunakan

d) BAB IV Pembahasan

   Menjelaskan dan memaparkan data-data hasil dari penelitian yang dilakukan.

e) BAB V PENUTUP

Menjelaskan mengenai kesimpulan akhir penelitian dan saran-saran yang direkomendasikan berdasarkan pengalaman di lapangan untuk perbaikan proses penelitian selanjutnya

BAB II
TINJAUAN PUSTAKA DAN SASAR TEORI

2.1 FOX-Baby Merk Elitech

Elitech adalah salah satu brand milik perusahaan *electromedical device* di Indonesia. PT. Sinko Prima Alloy berdiri pada tahun 1995 dan berdomisili tetap di Surabaya. Pada tanggal 26 September 2007, PT. Sinko Prima Alloy mendapatkan sertifikasi ISO 9001:2008. Penghargaan ini didapatkan melalui proses penilaian organisasi, produk dan pelayanan ke konsumen.

Salah satu produk yang terkenal di pasaran adalah FOX-BABY. FOX-BABY merupakan Pulse Oximeter untuk anak berusia 2 sampai 12 tahun. Ukuran yang cukup kecil dan ringan sangat memungkinkan untuk dibawa bepergian. Ukuran dari unit ini adalah panjang 46 mm x Lebar 40 mm x Tinggi 29 mm. Pada Tabel 2.1 akan dijelaskan spesifikasi dan nama enam *spare parts* yang akan dijadikan object penelitian.

Tabel 2.1 Bill of Material FOX-BAYI

No	Gambar	Nama	Spesifikasi
1.		Top Casing	Plastic ABS (P)45.5 X (L)39 X (T)22 mm (Hitam, Orange, Kuning, Biru)
2.		Battrey Cover	CMS7.820.136.01 (NZ) / 1.1 CMS7.820.135.02/1.0 (Hitam, Orange, Kuning, Biru)
3.		Bottom Casing	Plastic ABS (P)46 X (L)38.5 X (T)19 mm (Hitam dan Putih)
4		Mainboard	Plastic ABS (P)38 X (L)35 X (T)7.5 mm (Hijau Muda)

2.2 Citra

Pengertian citra secara umum ialah suatu gambar, foto ataupun berbagai tampilan dua dimensi yang menggambarkan suatu visualisasi objek. Citra bisa diwujudkan pada bentuk tercetak ataupun digital. Gambaran digital adalah barisan nomor-nomor secara dua dimensi. Gambaran digital tersimpan dalam suatu bentuk kumpulan angka digital yang akan terjadi kuantifikasi dari tingkat kecerahan masing-masing piksel penyusun citra tersebut. Tampilan dua dimensi mencerminkan beberapa *factor reflaksi* cahaya menggunakan warna Merah, Hijau, dan Biru yang dikombinasikan (Hestiningsih & Idhawati, 2018).

Dipandang berdasarkan sudut pandang matematis, citra adalah fungsi *continue* dari intensitas cahaya pada bidang dua dimensi. Sumber cahaya menerangi objek, objek memantulkan pulang sebagian berasal berkas cahaya tersebut. Pantulan cahaya ini ditangkap oleh alat-alat optik, contohnya mata pada mata, kamera, scanner dan lain sebagainya sebagai akibatnya bayangan objek yang didapatkan citra tersebut terekam (Hestiningsih & Idhawati, 2018). Citra digital yang tersimpan pada larik dua dimensi tersusun atas unsur-unsur kecil yang diklaim menggunakan piksel. Masing-masing piksel terkait secara spasial dengan area di permukaan bumi. Struktur array ini tersusun dalam urutan horisontal yang disebut baris serta urutan vertikal yang disebut kolom. Masing-masing piksel dalam raster gambaran menyimpan nilai taraf kecerahan piksel yang diujudkan menjadi suatu nomor digital. Susunan piksel dalam struktur kumpulan citra digital tersebut dianggap dengan data *raster*. menjadi suatu susunan asal nomor digital, beberapa bentuk operasi matematis dapat diberlakukan terhadap citra digital tadi.

Pengolahan citra digital merupakan manipulasi serta interprestasi digital berasal citra menggunakan bantuan komputer. Input berasal pengolahan citra merupakan gambar, sedangkan outputnya adalah gambaran hasil pengolahan (Prabowo et al., 2018a). Istilah pengolahan gambar digital secara umum didefinisikan sebagai pemrosesan gambar dua dimensi pada perangkat komputer dengan menggunakan *tools* tertentu. pada definisi yang lebih luas, pengolahan citra digital juga meliputi semua data dua dimensi. Citra digital merupakan barisan beberapa pixel komplek yang diwakili oleh bit-bit tertentu.

Operasi-operasi pada pengolahan citra diterapkan bila:

1. Perbaikan atau memodifikasi citra dilakukan untuk meningkatkan kualitas penampakan citra/menonjolkan beberapa aspek informasi yang terkandung dalam citra (image enhancement). Contoh: perbaikan kontras gelap/terang, perbaikan tepian objek, penajaman, pemberian warna semu, dll.

2. Adanya cacat pada citra sehingga perlu dihilangkan/ diminimumkan (image restoration). Contoh: penghilangan kesamaran (*debluring*) citra tampak kabur karena pengaturan fokus lensa tidak tepat / kamera goyang, penghilangan noise.
3. Elemen dalam citra perlu dikelompokkan, dicocokan atau diukur *(image segmentation*). Operasi ini berkaitan erat dengan pengenalan pola.
4. Diperlukannya ekstraksi ciri-ciri tertentu yang dimiliki citra untuk membantu dalam pengidentifikasian objek (*image analysis*). Proses segementasi kadangkala diperlukan untuk melokalisasi objek yang diinginkan dari sekelilingnya. Contoh: pendeteksian tepi objek.
5. Sebagian citra perlu digabung dengan bagian citra yang lain (*image reconstruction*). Contoh: beberapa foto rontgen digunakan untuk membentuk ulang gambar organ tubuh.
6. Citra perlu dimampatkan (*image compression*) Contoh: suatu file citra berbentuk BMP berukuran 258 KB dimampatkan dengan metode JPEG menjadi berukuran 49 KB.

Menyembunyikan data rahasia (berupa teks/citra) pada citra sehingga keberadaan data rahasia tersebut tidak diketahui orang (*steganografi* dan *watermaking*).

2.3 *Pixel, Resolusi, Intensitas (value)*

Suatu gambar yang ada di pada komputer sesungguhnya berasal ribuan titik yang sangat kecil dan tiap-tiap titik tersebut memiliki rona eksklusif. Kotak-kotak kecil itulah yang disebut *pixel*, ukuran suatu citra dinyatakan pada titik atau *pixel* (Prabowo et al., 2018a). Setiap *pixel* mempunyai satu warna serta bergabung menggunakan beberapa *pixel* lainnya sehingga menghasilkan suatu pola dan membentuk gambar.

Jumlah *pixel* per wilayahnya dianggap dengan resolusi. Resolusi itulah yang menentukan kualitas dari gambar. Jika suatu gambar diperbesar, maka resolusi gambar akan sebagai mungil serta gambar menjadi tidak tajam. Semakin tinggi resolusi gambar, maka akan meningkat kemampuan perbesarannya.

Pixel yang membentuk suatu gambar mempunyai warna-rona eksklusif. Jumlah rona yang dimiliki suatu gambar diklaim intensitas. Intensitas gambar mempunyai beberapa jenis data yaitu 256 warna, *high color*, 16 juta rona (*true color*), gradasi abu-abu (*grayscale*), serta hitam-putih (*black & white*).

Semakin banyak jumlah warna pada suatu gambar maka akan semakin indah. Jumlah warna maksimum berasal gambar dapat dicermati dari jenis eksistensi filenya. Arsip gambar

berformat (.jpg) memiliki jumlah rona maksimum 16 juta rona, arsip gambar berekstensi (.gif) mempunyai jumlah rona maksimum 265 rona.

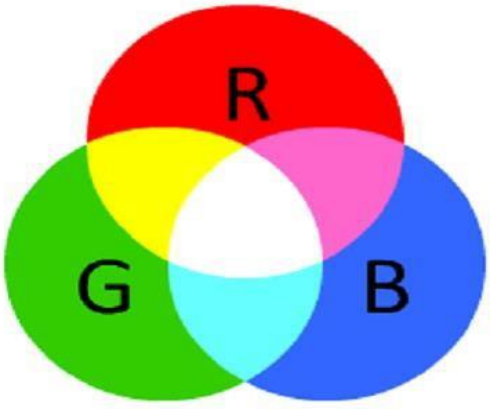
2.4 Model warna *Red, Green, Blue (RGB)*

RGB adalah suatu model warna yang terdiri atas 3 buah warna: merah *(red)*, hijau *(green)*, dan biru *(blue)*, dan jika ditambah proses pencampuran dengan berbagai cara dapat menghasilkan bermacam-macam warna (Prabowo et al., 2018a).

Model warna RGB adalah model warna berdasarkan konsep penambahan kuat cahaya primer yaitu *red, green* dan *blue*. Dalam suatu ruang yang sama sekali tidak ada cahaya, maka ruangan tersebut adalah gelap total. Tidak ada signal gelombang cahaya yang diserap oleh mata kita atau *RGB* (0, 0, 0). Apabila kita menambahkan cahaya merah pada ruangan tersebut, maka ruangan akan berubah warna menjadi merah misalnya *RGB* (255, 0, 0), semua benda dalam ruangan tersebut hanya dapat terlihat berwarna merah. Demikian apabila cahaya kita ganti dengan hijau atau biru.

Seperti yang diketahui tahu bahwa *RGB* atau *Red, Green, Blue* merupakan sistem pewarnaan untuk *digital appearance* dan banyak sekali digunakan untuk monitor komputer, video, layar ponsel dll. Sistem warna *RGB* terdiri dari 100% Red, 100% Green dan 100% Blue yang menghasilan 100 % putih. Tidak ada hitam di *RGB*.

Apabila kita melanjutkan percobaan memberikan 2 macam cahaya primer dalam ruangan tersebut seperti (merah dan hijau), atau (merah dan biru) atau (hijau dan biru), maka ruangan akan berubah warna masing-masing menjadi kuning, atau magenta atau cyan. Warna-warna yang dibentuk oleh kombinasi dua macam cahaya tersebut disebut warna sekunder.



Gambar 2.1 Warna *RGB* (Hestiningsih & Idhawati, 2018)

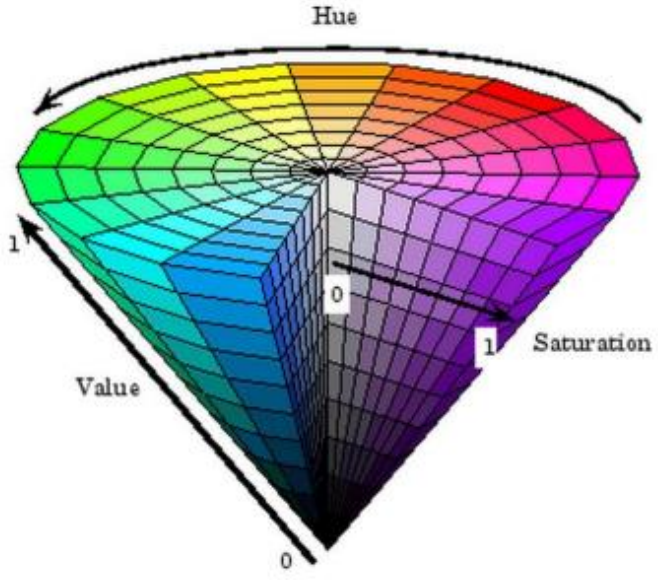
2.5 *Color conversion RGB to HSV*

Tidak seperti warna *RGB, HSV* lebih dekat dengan bagaimana manusia memandang warna. Terdapat tiga komponen: *Hue, Saturation, and Value. Hue/Tint* adalah ruang warna yang menggambarkan warna tersebut *(Hue or Tint)* dalam ketentuannya *(Saturation* adalah jumlah dari *gray*) dan *brightness value nya.* Berikut adalah penjelasan dari nilai *HSV*:

a) *Hue* adalah bagian warna model. Dinyatakan sebagai angka dari 0 hingga 360 derajat. Berikut adalah pembagiannya
   - *Red* jatuh diantara 0 dan 60 derajat.
   - *Yellow* jatuh diantara 61 dan 120 derajat.
   - *Green* jatuh diantara 121 dan 180 derajat.
   - *Cyan* jatuh diantara 181 dan 240 derajat.
   - *Blue* jatuh diantara 241 dan 300 derajat.
   - *Magenta* jatuh diantara 301 dan 360 derajat.

b) *Saturation* di deskripsikan sebagai nilai abu-abu dari warna particular, dimulai dari 0 sampai 100. Mengurangi komponen menuju titik Nol menghasilkan lebih banyak abu-abu dan membuat sebuah efek pudar. Sesekali, Saturasi muncul sebagai jarak dari 0 sampai 1, dimana 0 adalah abu-abu dan 1 adalah warna asli.

c) *Value (Brightness)* bekerja dalam konjungsi dengan saturasi dan mendeskripsikan *brightness* atau intensitas dari warna, dari 0 sampai 100 persen, dimana 0 adalah warna hitam pekat dan 100 adalah paling terang dan dan kecerahan mengungkapkan warna terbanyak.



Gambar 2.2 Ruang warna *HSV*(*OpenCV: Changing Colorspaces*, n.d.)

Konversi nilai *RGB to HSV* digunakan dalam menentukan pandangan warna oleh manusia sendiri. Berikut adalah rumus untuk mendapatkan nilai *HSV*(*OpenCV: Changing Colorspaces*, n.d.).

Persamaan dalam mencari nilai *Value*

$$V = \max(R, G, B) \tag{2.1}$$

Persamaan dalam mencari nilai Saturasi (S)

$$S = \begin{cases} \frac{V - \min(R,G,B)}{V} & if\ V \neq 0 \\ 0 & Otherwise \end{cases} \tag{2.2}$$

Persamaan dalam mencari nilai *Hue*

$$H = \begin{cases} \frac{60(G-B)}{(V - \min(R,G,B)} & if\ V = R \\ (120 + 60)\frac{(B-R)}{(V - \min(R,G,B))} & if\ V = G \\ (240 + 60)\frac{R-G}{(V - \min(R,G,B)} & if\ V = B \end{cases} \tag{2.3}$$

Pada *image processing* menggunakan Open CV 8-bit maka nilai akan berubah formatnya menjadi

V = 255V, S = 255S, H = 180 (Menjadi nilai yang cocok dengan 0 sampai 255)

2.6 *Object Color Tracking*

Metode *Tracking* dapat dibagi menjadi dua kelas utama yang ditentukan sebagai pendekatan bottom-up atau *top-down*. Dalam pendekatan *bottom-up*, gambar umumnya tersegmentasi menjadi objek yang kemudian digunakan untuk pelacakan. pendekatan *top-down* menghasilkan hipotesis objek dan mencoba memverifikasinya menggunakan gambar (Nummiaro et al., n.d.).

Object color tracking adalah salah satu metode tercepat dan termudah untuk melacak objek dari satu *frame* gambar ke *framre* gambar berikutnya. Kecepatan teknik ini membuatnya sangat menarik untuk aplikasi yang hampir *real-time*. Metode ini menggunakan pengolahan citra berbasis warna dimana akan didapatkan contour area yang sesuai dengan objek yang dikehendaki (Michalko et al., n.d.).

Citra digital tidak lebih dari matriks nilai skalar atau vektor (tergantung pada apakah gambar itu monokromatik atau berwarna) dan dapat diproses dengan berbagai cara. Di situlah perpustakaan *OpenCV* dapat digunakan, karena dioptimalkan untuk operasi semacam itu. *Masking* objek ditentukan oleh siluetnya. Representasi siluet tersebut dalam format

digital disebut citra biner. Biner berarti, bahwa elemen terkecil, dalam hal ini piksel, dapat diwakili oleh 0 atau 1 (atau 0 dan 255).

Kontur objek ditentukan oleh kumpulan titik, yang menggambarkan tepi objek, garis besar. Misalnya, kontur bola tenis adalah lingkaran. Banyak algoritma untuk menemukan kontur dalam citra digital diusulkan dan salah satu yang pertama diusulkan oleh Theo Pavlidis (Theodosios Pavlidis, 1982), di mana algoritmanya dianggap sebagai dasar dari semua yang lain. Dokumentasi *OpenCV* menawarkan implementasi algoritma pencarian kontur yang sangat efisien, yang berisi fitur tambahan seperti ekstraksi kontur dalam hierarki dan perkiraan kontur yang ditemukan. Perkiraan kontur garis dengan kumpulan titik adalah fitur yang sangat berguna dan secara signifikan mempercepat lebih lanjut pengolahan kontur.

Tracking memiliki arti mengkuti jalan, atau dalam arti bebasnya ialah suatu kegiatan untuk mengikuti jejak suatu objek (Prabowo et al., 2018b). Sistem pelacakan adalah suatu sistem yang mampu melacak atau mencari suatusuatu hal dengan memberikan informasi tentang hal tersebut. Beberapa faktor yang sering kali mengganggu pelacakan objek menggunakan metode visual adalah sebagai berikut.

1. Hilangnya informasi dikarenakan projeksi 3 dimensi dalam citra 2 dimensi.
2. *Noise* pada citra.
3. Nonrigid atau artikulasi alami pada objek.
4. Objek terhalang suatu benda.
5. Bentuk objek yang rumit.
6. Perubahan drastis pencahayaan.
7. Persyaratan pengolahaan secara Real-Time.

Dasar dari teknik pelacakan objek memerlukan suatu pembeda pada objek yang ingin dilacak yang akan menjadi suatu acuan pelacakan. Seperti warna, bentuk, ataupun perbandingan frame 1 ke frame yang berikutnya.

2.7 *Gaussian filter*

Gaussian Filter adalah *Low Pass Filter* yang digunakan untuk mengurangi noise (komponen frekuensi tinggi) dan mengaburkan area gambar. Filter diimplementasikan sebagai matrik Simetris berukuran ganjil (versi DIP dari Matriks) yang dilewatkan melalui setiap piksel Wilayah yang Diinginkan untuk mendapatkan efek yang diinginkan. Matrik tidak sulit untuk berubah warna secara drastis karena piksel ke arah tengah matrik memiliki bobot lebih ke arah nilai akhir daripada periferal. *Gaussian Filter* dapat dianggap sebagai aproksimasi dari Fungsi *Gaussian*. (*OpenCV: Smoothing Images*, n.d.)

Dalam proses menggunakan *gaussian filter* pada sebuah gambar, pertama-tama kita menentukan ukuran Matrik yang akan digunakan untuk menghilangkan gambar. Ukuran umumnya angka ganjil, yaitu hasil keseluruhan dapat dihitung pada piksel pusat. Matrik juga simetris & karena itu memiliki jumlah baris dan kolom yang sama. Nilai di dalam kernel dihitung dengan fungsi *Gaussian*, yaitu sebagai berikut:

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \quad (2.4)$$

Dimana,

$x = X\ koordinat$

$y = Y\ koordinat$

$\pi = 3.14$

$\sigma = Standar\ Deviation$

OpenCV menyediakan fungsi cv2.gaussianblur() untuk menerapkan Gaussian Smoothing pada gambar sumber input. Berikut ini adalah sintaks dari fungsi GaussianBlur():

dst = cv2.GaussianBlur(src, ksize, sigmaX[, dst[, sigmaY[, borderType = BORDER_DEFAULT]]])

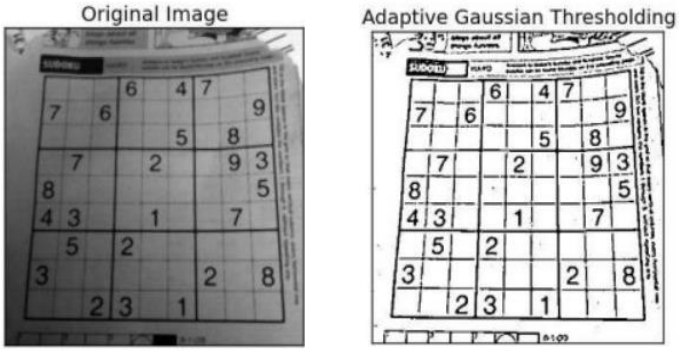
Tabel 2.2 Parameter dalam syntax OpenCV

No	Parameter	Deskripsi
1	*scr*	*input image*
2	*dst*	*output image*
3	*ksize*	*Gaussian Kernel Size. [height width]. height and width should be odd and can have different values. If ksize is set to [0 0], then ksize is computed from sigma values.*
4	*sigmaX*	*Kernel standard deviation along X-axis (horizontal direction).*
5	*sigmaY*	*Kernel standard deviation along Y-axis (vertical direction). If sigmaY=0, then sigmaX value is taken for sigmaY*
6	*borderType*	*Specifies image boundaries while kernel is applied on image borders. Possible values are:cv.BORDER_CONSTANT*

		cv.BORDER_REPLICATE *cv.BORDER_REFLECT* *cv.BORDER_WRAP* *cv.BORDER_REFLECT_101* *cv.BORDER_TRANSPARENT* *cv.BORDER_REFLECT101* *cv.BORDER_DEFAULT* *cv.BORDER_ISOLATED*

2.8 *Otsu Thresholding*

Otsu theresholding adalah teknik yang memungkinkan untuk melakukan binarisasi gambar berdasarkan nilai piksel. Biasanya, jika nilai piksel lebih besar dari ambang batas, itu diatur ke nilai maksimum (seringkali 255 - putih), jika tidak, diatur ke 0 (hitam). Ada berbagai metode untuk menghitung nilai ambang batas. Salah satunya adalah metode *Otsu* adalah algoritma ambang batas global, di mana nilai ambang tunggal diterapkan untuk seluruh gambar (*OpenCV: Image Thresholding*, n.d.).



(a) (b)

Gambar 2.3 (a) Gambar *original* (b) Hasil *Otsu Thresholding* (*Otsu's Thresholding Technique | LearnOpenCV*, n.d.)

OpenCV menawarkan fungsi ambang batas untuk melakukan ambang batas gambar. Kita dapat menentukan flag THRESH_OTSU sebagai argumen untuk menerapkan metode *Otsu* dalam menghitung nilai ambang. Berikut adalah persamaan untuk mencari nilai *thresholding image.*

$$\sigma_B^2 = W_b W_f (\mu_b - \mu_f)^2 \quad (2.5)$$

Dimana nilai:

σ_B^2 = Nilai maksimum Thresholding

$W_b W_f$ = Nilai pixel backround dan foreground

$\mu_b \mu_f$ = Nilai intensitas (value) dari backround dan foreground

2.9 *Sobel Edge Detection*

Sobel edge detection adalah teknik pemrosesan citra, yang digunakan untuk mengidentifikasi batas atau tepi objek, atau wilayah dalam suatu citra. Tepi adalah salah satu fitur terpenting yang terkait dengan gambar. Kami mengetahui struktur yang mendasari gambar melalui tepinya. Oleh karena itu, pipeline pemrosesan visi komputer secara ekstensif menggunakan deteksi tepi dalam aplikasi. (*OpenCV: Sobel Derivatives*, n.d.) Metode ini bekerja dengan menghitung gradien intensitas gambar pada setiap piksel dalam gambar. Metode ini menemukan arah kenaikan terbesar dari terang ke gelap dan laju perubahan ke arah itu. Hasilnya menunjukkan seberapa mulus gambar berubah pada masing-masing piksel, dan oleh karena itu seberapa besar kemungkinan piksel tersebut mewakili tepi. Ini juga menunjukkan bagaimana tepi itu cenderung berorientasi. Hasil penerapan filter ke piksel di wilayah konstan intensitas adalah vektor nol. Hasil penerapannya pada piksel di tepi adalah vektor yang menunjuk melintasi tepi dari nilai yang lebih gelap ke nilai yang lebih terang. **Gambar 2.4** adalah gambar original dan gambar yang telah dilakukan deteksi tepi.

(a) (b)

Gambar 2.4 (a) Gambar *original* (b) Hasil dari *Sobel Edge Detection* (*OpenCV: Sobel Derivatives*, n.d.)

2.10 *Raspberry Pi 3 B*

Raspberry Pi merupakan komputer dalam satu *singleboard*. *Operating System* (OS) pada *Raspberry Pi* yaitu *Linux*. *Raspberry Pi* memiliki beberapa seri seperti *Raspberry Pi* 1, 2, 3, model A, model A+, model B, model B+. Seri yang akan digunakan dalam penelitian kali ini adalah *Raspberry Pi 3* model B yang merupakan seri terbaru. Berikut ini adalah spesifikasi dari *Raspberry Pi* 3 model B (Christian, 2017)

a. *Prosesor*

   System on a Chip (SoC) berupa jenis chip jenis *Broadcom BCM2837R*. CPU *4x ARM Cortex-A53,* kecepatan prosesor 1.2 *GHz*, GPU berupa *Broadcom VideoCore IV*, dengan RAM 1 GB LPDDR2 (900 *MHz*).

b. *Slot Secure Digital Card (SD Card)*

Raspberry Pi 3 model B dilengkapi dengan slot SD *card* sebagai *hard drive* untuk menyimpan seluruh data.

c. *Port* USB

Raspberry Pi 3 model B mempunyai 4 port USB tipe 2.0.

d. *Bluetooth*

Raspberry Pi 3 model B mempunyai jenis *Bluetooth* 4.1 *Classic* yang berfungsi sebagai media penghubung komunikasi dengan perangkat komunikasi lainnya.

e. Konektor *HDMI*

Raspberry Pi 3 model B mempunyai *port HDMI* sebagai perantara *audio/video* yang akan ditampilkan pada sebuah monitor.

f. *Output Audio Analog*

Port audio analog berfungsi sebagai penyedia keluaran *audio analog* untuk disambungkan pada perangkat *speaker* dengan *jack* sebesar 3,5 mm.

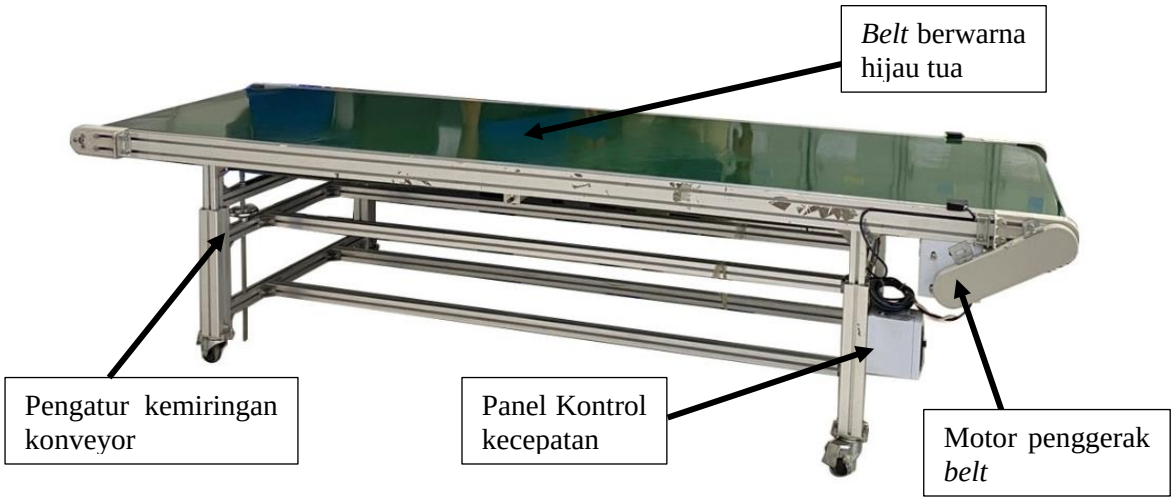
Gambar 2.5 Raspberry Pi 3 B+ (*Raspberry Pi*, 2021)

Raspberry Pi 3 model B terdapat 40 buah pin. Pin pada *Raspberry Pi 3 B*+ terdiri dari beberapa bagian yaitu bagian *VCC, GND* dan *GPIO* (*General Purpose Input/Output*). Terdapat 3 pin *VCC* dan 8 pin *GND*. Pin GPIO mulai dari GPIO2 hingga GPIO27, pada pin GPIO terdapat fungsi lainnya.

2.11 Konveyor Belt

Conveyor belt atau konveyor sabuk adalah media pengangkutan yang digunakan untuk memindahkan muatan dalam bentuk satuan atau tumpahan, dengan arah horizontal atau membentuk sudut inklinasi dari suatu sistem operasi yang satu ke sistem operasi yang lain dalam suatu jalur proses produksi, yang menggunakan sabuk *(Belt)* sebagai penghantar muatannya (Thorat, 2015).

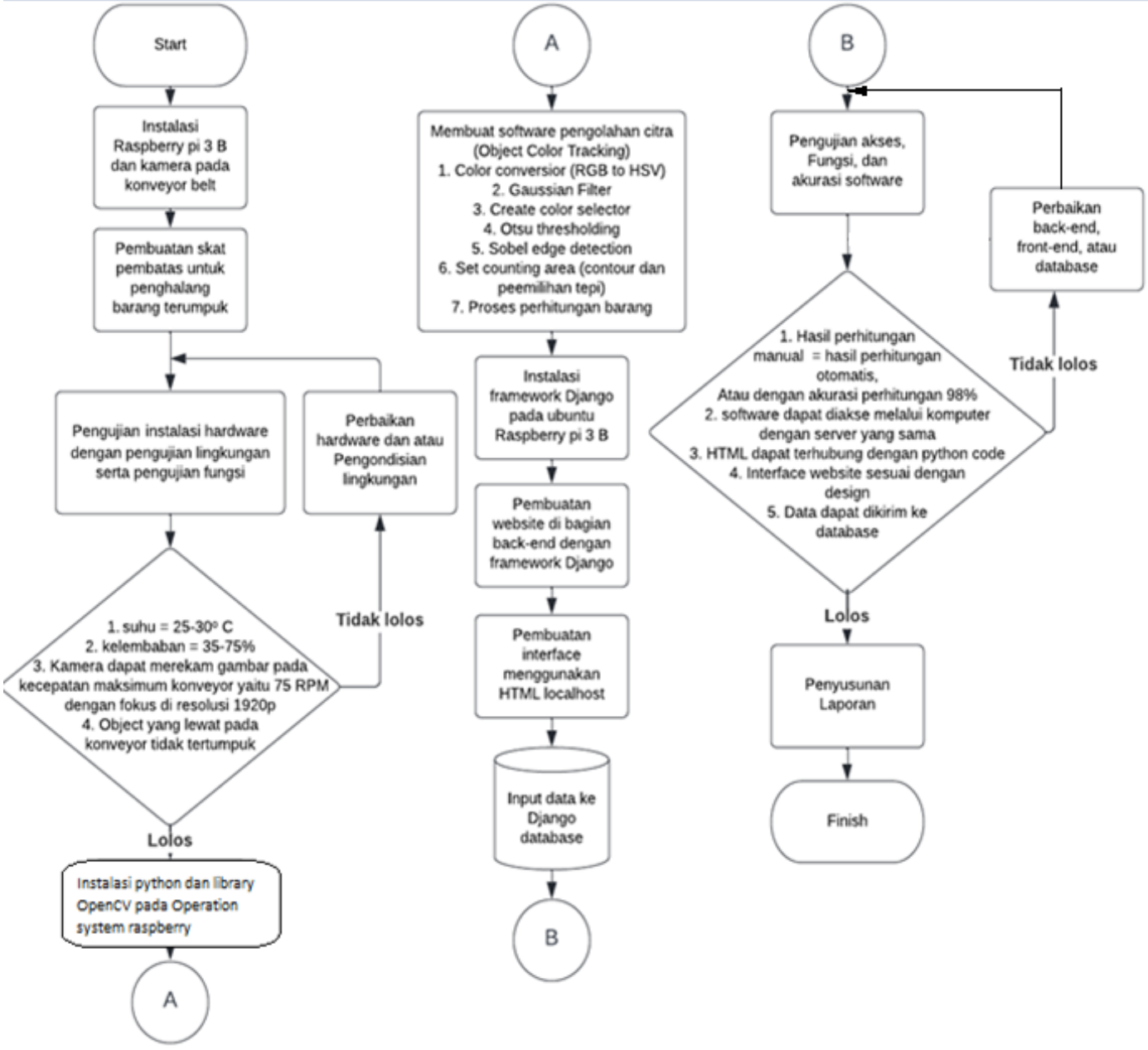
Kelebihan dari transportasi dengan Belt Conveyor antara lain bekerja secara otomatis, mudah dalam memulai operasi dan terus beroperasi secara terus menerus. Konveyor sabuk (belt conveyor) memiliki komponen utama berupa sabuk yang berada diatas roller-roller penumpu. Sabuk digerakkan oleh motor penggerak melalui suatu pulley, sabuk bergerak secara translasi dengan melintas datar atau miring tergantung kepada kebutuhan dan perencanaan. Konveyor yang dimiliki oleh PT. Sinko Prima Alloy memiliki belt yang berwarna hijau. Ukuran dari konveyor tersebut dalam satuan centimeter P x L: 298 x 95. **Gambar 2.6** merupakan penampakan konveyor belt yang berada di PT. Sinko Prima Alloy



Gambar 2.6 Konveyor *Belt* PT. Sinko Prima Alloy

BAB III
METODOLOGI PENELITIAN

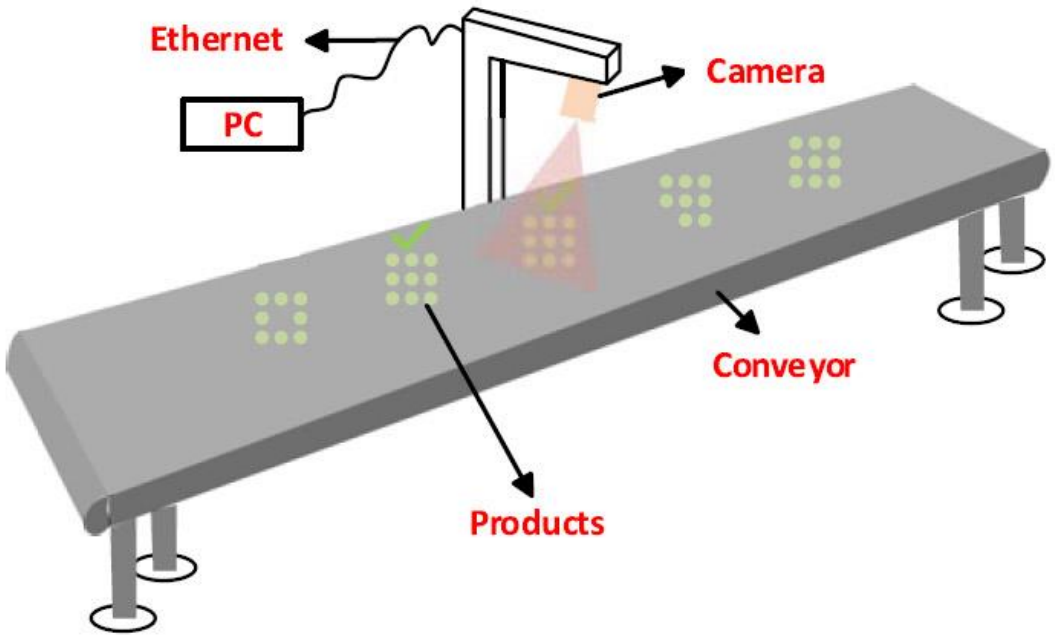
Rancang bangun penghitung sparepart unit FOX-Baby merek elitech dengan metode *object color tracking* menggunakan *Raspberry pi 3 B* dibuat dengan melakukan beberapa tahapan penelitian. Pada **Gambar 3.1** adalah beberapa kegiatan yang harus dilakukan dalam perancangan bangun tersebut.



Gambar 3.1 Diagram alir penelitian

3.1 Instalasi *Raspberry pi 3 B* dan kamera pada konveyor belt

Instalasi dilakukan berdasarkan posisi konveyor di ruangan dan keadaan lingkungan sekitar. Faktor yang diperhatikan pada pemasangan adalah kelembaban ruangan, suhu dan intensitas cahaya di sekelilingnya dengan cara melakukan pengukuran dengan alat ukur. Hal ini bertujuan untuk meningkatkan *lifetime dari raspberry pi 3 B* tersebut. Sedangkan untuk instalasi kamera dilakukan berdasarkan jarak pandang kamera, lebar dari konveyor belt, dan Kekuatan focus dari kamera. Kamera yang digunakan pada penelitian ini adalah Kamera dengan resolusi 1920P 2K 18 MegaPixel dengan merk *Nemesis*. **Gambar 3.2** adalah rancangan singkat dari pemasangan raspberry pi 3 b dan kamera pada konveyor belt.



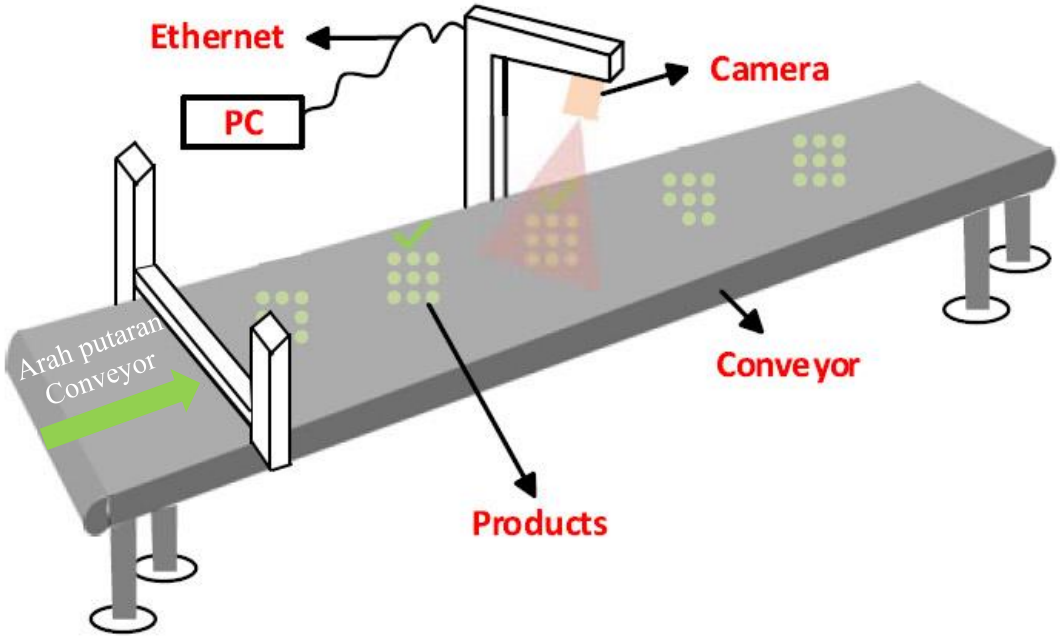
Gambar 3.2 Rancangan pemasangan *raspberry pi 3 B* dan kamera.

3.2 Instalasi pemasangan skat untuk meencegah barang tertumpuk

Instalasi pemasangan skat ini berujuan untuk mengindari barang yang tertumpuk. Barang yang tertumpuk jika lewat pada kamera akan menghasilkan error yang signifikan di aplikasi. Dengan pemasangan skat ini diharapkan barang yang tertumpuk akan tertata dan dapat dihitung sesuai dengan prosedur. Bahan yang digunakan untuk pemasangan adalah alumium profile dengan lebar 40 cm x 40 cm. Aluminium profile adalah jenis aluminium yang berbentuk batangan dengan berbagai macam bentuk penampang. Aluminium profil hadir dalam macam-macam bentuk padat dan berongga yang tak terbatas, seperti sudut,

tabung bundar, tabung persegi, dan panjang slot-t, dan lain sebagainya. Aluminium profil mengacu pada bentuk penampang dari produk aluminium ekstrusi.

Keunggulan *aluminium profile* yaitu gampang sekali pengaplikasiannya karena merakitnya melalui *knockdown system*. Yang dimaksud *knockdow system* adalah merakit seperti halnya main lego jadi hanya disambung dengan *connector* yang sudah ada dan tidak perlu di las. *Aluminium profile* memiliki keunggulan yaitu tidak cepat berkarat dan proses perakitannya yang cukup mudah dikarenakan dapat di sambung sesuai dengan kebutuhan. Pada **Gambar 3.3** merupakan sketsa dari pemasangan skat penghalang tumpukan barang.



Gambar 3.3 Rancangan pemasangan skat pembatas untuk barang tertumpuk

3.3 Instalasi *Python* dan *library OpenCV* pada *Operation System Raspberry*

Instalasi python (PIP) dibutuhkan sebelum menjalankan program yang mengambil data dari library OpenCV dengan code. Instalasi pip python merupakan Langkah awal dalam proses instalasi paket-paket library yang dimiliki oleh program python tersendiri. Berikut adalah beberapa code yang harus dijalankann untuk proses instalasi PIP python dan Library OpenCV.

1. Pertama dilakukan proses check python3 pada Raspberry pi dengan cara berikut

   $ python3

2. Jika versi dari python telah muncul maka dilanjukan dengan install pip3, jalankan pada terminal,

   $ sudo python3-pip

3. Dikarenakan os yang digunakan adalah *Linux*, harus dilakukan penambahkan *GUI* agar *OpenCV* dapat diinstall secara *full feature, install GUI PIXEL ubuntu* dengan cara berikut,

   $ sudo apt-get install raspberrypi-ui-mods

4. Selanjutnya dilakukan instalasi *library* pendukung untuk *OpenCV*, jalankan pada terminal,

   sudo apt-get -y install libatlas-base-dev libqtgui4 python3-pyqt5 libqt4-test libilmbase-dev libopenexr-dev libgstreamer1.0-dev

   sudo apt-get -y install libopencv-dev

   sudo apt-get -y install build-essential checkinstall cmake pkg-config yasm

   sudo apt-get -y install libtiff4-dev libjpeg-dev libjasper-dev

   sudo apt-get -y install libavcodec-dev libavformat-dev libswscale-dev libdc1394-22-dev libxine-dev libgstreamer0.10-dev libgstreamer-plugins-base0.10-dev libv4l-dev

   sudo apt-get -y install libqt4-dev libgtk2.0-dev

   sudo apt-get -y install libfaac-dev libmp3lame-dev libopencore-amrnb-dev libopencore-amrwb-dev libtheora-dev libvorbis-dev libxvidcore-dev

   sudo apt-get -y install x264 v4l-utils ffmpeg

5. Selanjutnya dilakukan instalasi *OpenCV* dengan cara berikut,

   $ pip3 install opencv-python==3.4.6.27

6. buka python REPL untuk melakukan checking hasil instalasi

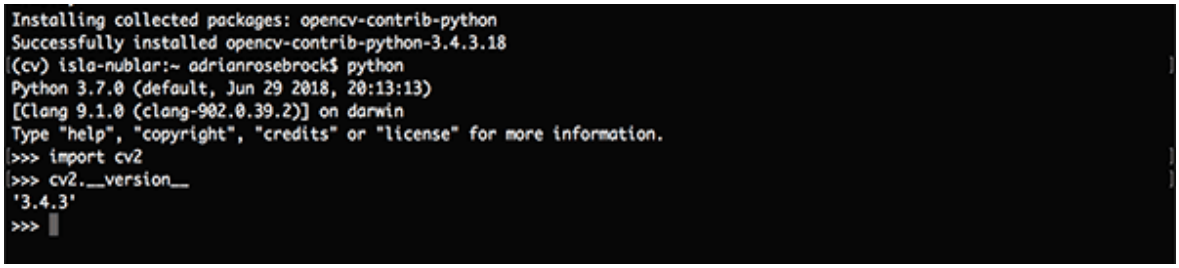
   $ python3

7. import opencv dan check version,

   >>> import cv2

   >>> cv2.__version__

Hasil yang muncul dari proses check version pada step terakhir instalasi haruslah memunculkan versi dari library tersebut secara detail.



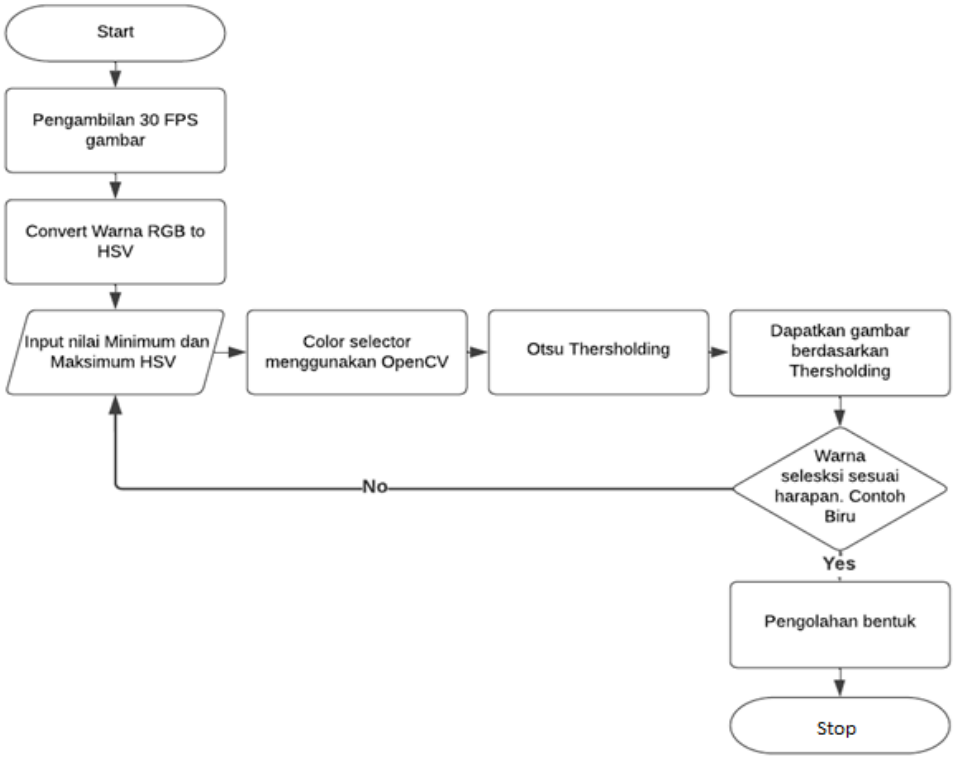
Gambar 3.4 Contoh hasil *check version* instalasi library OpenCV (*Pip Install OpenCV - PyImageSearch*, n.d.)

3.4 Rancangan metode *Object Color Tracing* menggunakan *Library OpenCV*

Terdapat beberapa tahapan dalam penyusunan program *Object Color Tracking.* Terdapat dua bagian besar dalam pembuatan yaitu proses Pemilahan warna dan Deteksi bentuk. Pada dasarnya semua perhitungan tersebut menggunakan Bahasa python yang dimplementasikan pada Lokal server. Setiap prosesnya akan dijelaskan sesuai dengan algoritma berikut

3.2.1 Kalibrasi Nilai Maksimum dan Minimum *HSV*

Pada proses pemilahan warna terdapat beberapa proses yang akan dijabarkan sesuai flowchart berikut.



Gambar 3.5 Diagram alir kalibrasi nilai maksimum dan minimum warna

Proses pengolahan citra dilakukan dengan menggunakan Bahasa pemrograman *python* dengan *library OpenCV*. Dimana akan terdapat permainan nilai *range* warna *HSV* dari nilai minimum dan maksimumnya. **Gambar 3.5** menunjukan bahwa terdapat beberapa proses yang dilakukan untuk mendapatkan warna seleksi dari gambar yang dilakukan pengolahan citra.

Berikut adalah penjelasan dari beberapa prosedur pengerjaan.

1. (Pengambilan 30 FPS gambar) Rekam gambar yang digunakan yaitu menggunakan nilai 30 *Frame per Second* dengan menggunakan fungsi *OpenCV* sebagai berikut.

Import cv2

Import numpy as np

camera = cv2.VideoCapture(0)

#camera adalah variable yang digunakan untuk menyimpan tangkapan *frame OpenCV*

2. (*Convert warna RGB to HSV*) Segmentasi warna merupakan proses segmentasi dengan pendekatan daerah yang bekerja dengan menganalisis nilai warna dari tiap piksel pada citra dan membagi citra tersebut sesuai dengan fitur yang diinginkan. Citra digital menggunakan model warna RGB sebagai standar acuan warna, oleh karena itu proses awal pada metode ini memerlukan konversi model warna RGB ke HSV.

   hsv = cv2.cvtColor(src, cv2.COLOR_BGR2HSV)

   #hsv adalah variable yang digunakan untuk menyimpan frame yang telah di ubah menjadi warna HSV

3. (Input nilai minimum dan maksimum HSV) input nilai diperuntukan membentuk segmen sesuai dengan warna yang diinginkan maka ditentukan nilai toleransi pada setiap dimensi warna HSV, kemudian nilai toleransi tersebut digunakan dalam perhitungan proses adaptive threshold. Hasil dari proses threshold tersebut akan membentuk segmen area dengan warna sesuai toleransi yang diinginkan.

   l_b = np.array([l_h, l_s, l_v])

   u_b = np.array([u_h, u_s, u_v])

   #l_b dan u_b adalah variable yang digunakan untuk menyimpan nilai array dari nilai maksimum dan minimum HSV

4. (Otsu Thresholding) penambahan fungsi thresholding digunakan untuk memisahkan pixel gambar sesuai dengan background dan foreground. Background akan berwarna hitam dan foreground (Warna seleksi) akan berwarna putih.

   mask = cv2.inRange(gaussian_blur, l_b, u_b)

   #mask adalah variable yang digunakan untuk menyimpan frame yang telah di ubah menjadi warna Threshold

5. (Dapatkan gambar berdasarkaı resholding) Gambar didapatkan dengan menumpuk gambar sesuai dengan sumbu x dan sumbu y dengan gambar original dengan gambar thresholding. Kemudian, foreground dari thresholding dijadikan sebagai selector.

warna_seleksi = cv2.bitwise_and(src, src, mask=mask)

#warna adalah variable yang digunakan untuk menyimpan frame yang telah di ubah menjadi warna seseksi hasil pemotongan threshold.

6. (Warna seleksi sesuai dengan warna yang diharapkan. contoh = biru) Jika warna yang dihasilkan berdasarkan hasil seleksi sudah tepat biru maka akan dilanjutkan dengan pengolahan bentuk. Namun jika hasil yang didapatkan kuning, merah atau warna lain selain warna biru object maka akan dilanjutkan ke proses input nilai minimum dan maksimum HSV

Setelah didapatkan nilai minimum dan maksimum data tersebut akan dijadikan acuan saat melakukan proses seleksi gambar berdasarkan warna. Dengan ketentuan nilai minimum dan nilai maksimum HSV warna yang dicari adalah warna Orange, Kuning, Biru, Hitam, Putih dan Hijau muda. Selain menghitung nilai minimum dan maksimum HSV warna, Harus dilakukan perhitungan nilai intensitas cahaya pada bidang yang akan dijadikan lahan proses perhitungan. Dimana di PT. Sinko Prima Alloy tempat yang digunakan adalah ruangan produksi kecil (Blok E9). Untuk data hasil kalibrasi akan disimpan dengan format table sebagai berikut.

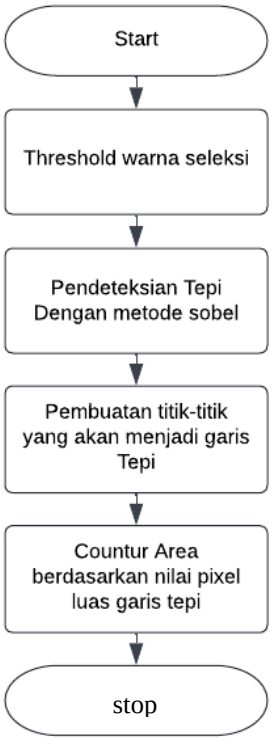
Tabel 3.1. Format hasil kalibrasi nilai minimum dan maksimum HSV setiap warna

No	Warna	Min Hue	Min Saturation	Min Value	Max Hue	Max Saturation	Max Value
1	Biru	….	….	….	….	….	….
2	Hitam	….	….	….	….	….	….
3	Kuning	….	….	….	….	….	….
4	Orange	….	….	….	….	….	….
5	Hijau muda	….	….	….	….	….	….
6	Putih	….	….	….	….	….	….

3.2.2 Deteksi bentuk dan luas area menggunakan *sobel edge detection*

Dalam proses pendeteksian bentuk dan luas area diperlukan beberapa tahapan meliputi pendeteksian tepi dengan menggunakan metode *sobel edge detection*. Edge detection adalah proses untuk menemukan perubahan intensitas yang berbeda nyata dalam sebuah bidang citra. Deteksi tepi berfungsi untuk memperoleh tepi objek, deteksi tepi memamfaatkan perubahan nilai intensitas yang drastis pada batas dua area Digunakan pendeteksian dengan

filter jenis konvolusi pada proses ini. Sehingga didapatkan luasan area di dalam tepi menggunakan metode *contour area*. Dimana sumbu X diangga sebagai arah vertical image sedangkan sumbu Y adalah arah horizontal image (Baygin & Karakose, 2017). **Gambar 3.6** adalah diagram alir dari proses pendeteksian bentuk dan luas area.



Gambar 3.6 Diagram alir pendeteksian bentuk dan luas area

Berikut adalah algoritma yang digunakan untuk mencari garis tepi dari setiap bagian warna seleksi.

scale = 1

delta = 0

ddepth = cv2.CV_16S

grad_x = cv2.Sobel(mask, ddepth, 1, 0, ksize=3, scale=scale, delta=delta, borderType=cv2.BORDER_DEFAULT)

grad_y = cv2.Sobel(mask, ddepth, 0, 1, ksize=3, scale=scale, delta=delta, borderType=cv2.BORDER_DEFAULT)

abs_grad_x = cv2.convertScaleAbs(grad_x)

abs_grad_y = cv2.convertScaleAbs(grad_y)

grad = cv2.addWeighted(abs_grad_x, 0.5, abs_grad_y, 0.5, 0)

contours, hierarchy = cv2.findContours(mask, cv2.RETR_TREE, cv2.CHAIN_APPROX_NONE)

x, y, w, h = cv2.boundingRect(contours[i])

print("area", i, ":", cv2.contourArea(contours[i]))

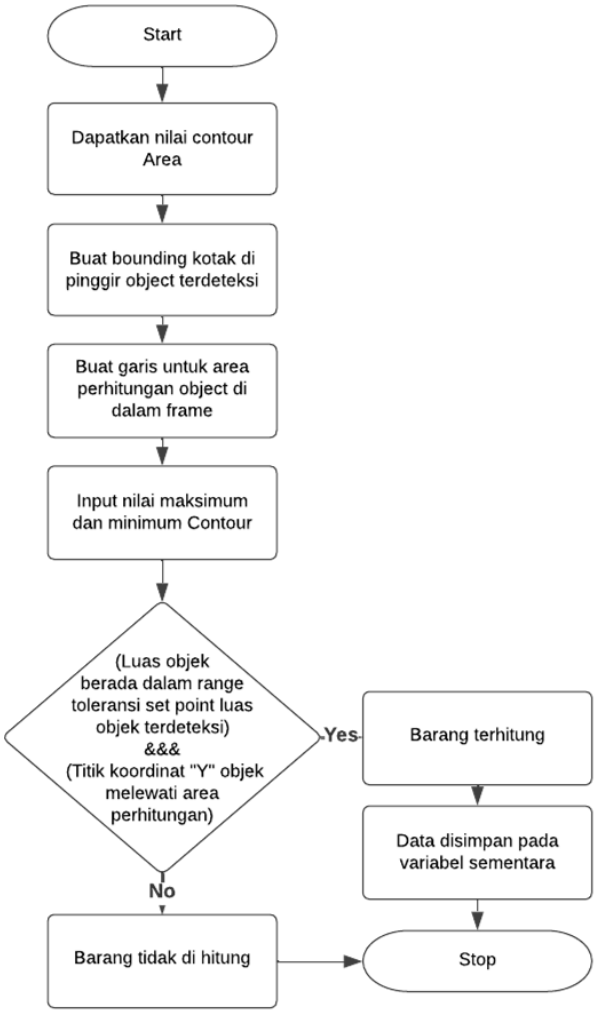
#kata yang terdapat di depan tanda sama dengan adalah variable dari masing-masing fungsi.

Dilanjutkan dengan melakukan deteksi jumlah pixel dari luasan area yang telah terdeteksi sehingga nilai tersebut akan dapat dibandingkan pada saat proses pendeteksian nilai. Berikut adalah persamaan yang digunakan

cv2.contourArea(contours[i])

3.5 Perhitungan objek yang melewati konveyor

Pada proses perhitungan objek yang melewati konveyor dibutuhkan algoritma yang dapat mengetahui kapan benda tersebut dikatakan lewat dan juga kapan benda tersebut dikatakan menghilang.

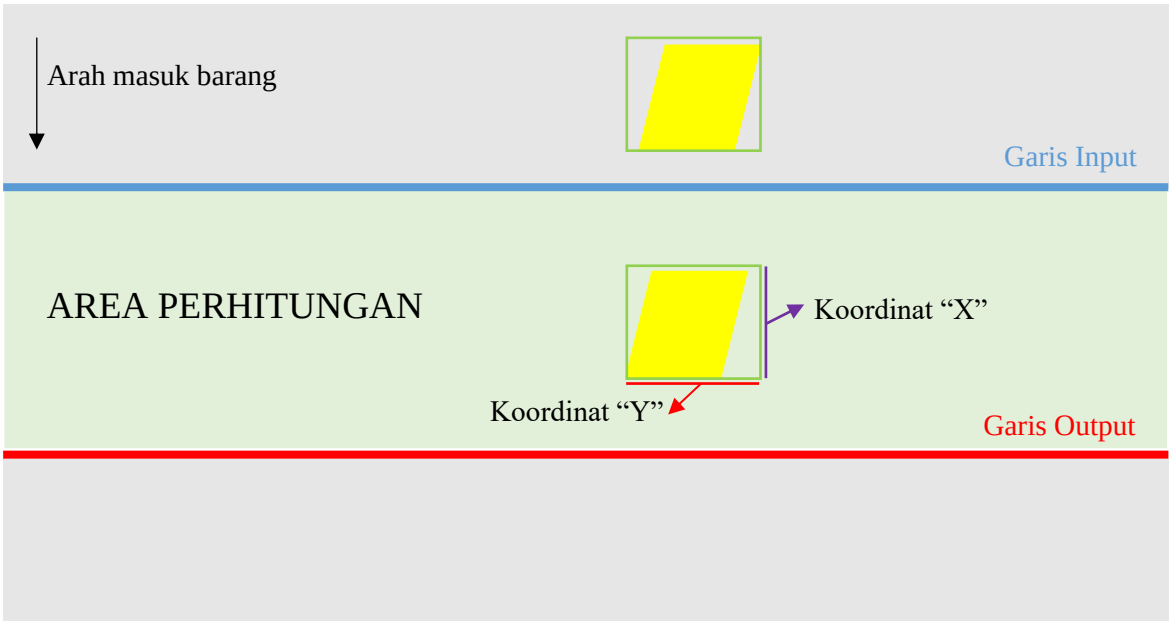


Gambar 3.7 Diagram alir perhitungan benda yang melewati konveyor

Fungsi if pada **Gambar 3.7** memiliki 2 kondisi agar benda/objek dapat dihitung dan kondisi tersebut menggunakan medote AND (&). Berikut adalah kedua kondisi tersebut.

1. Ketika *contour area* benda (Luas benda dalam satuan pixel) berada di antara batas maksimum dan minimum *set point contour area* perhitungan objek yang terdeteksi.
2. Ketika koordinat Y benda/objek melewati garis perhitungan input dan output. Nilai koordinat benda dideteksi menggunakan fungsi *boundingrect* (Kotak hijau yang muncul saat mendeteksi benda) perhatikan **Gambar 3.8**. sedangkan nilai koordinat Y input dan output area didapatkan dengan nilai vertical resolusi gambar.

Jika kedua kondisi tersebut terpenuhi, maka benda/objek dapat dihitung dan akan disimpan pada variable sementara terlebih dahulu, dikarenakan nilai tersebut akan terus bertambah Ketika objek menghilang dari frame video akibat berjalannya konveyor dan akan muncul objek baru saat proses perhitungan selanjutnya. Simulasi perhitungan dapat diasumsikan seperti **Gambar 3.8**



Gambar 3.8 ilustrasi sistem perhitungan

Dilakukan proses perhitungan dengan memperhatikan beberapa komponen sebagai berikut:

1. (Dapatkan nilai contour area) Nilai ini adalah jumlah pixel yang ada di dalam luasan area seleksi. Nilai ini didapatkan dengan menggunakan persamaan sebagai berikut.

   cv2.contourArea(contours[i])

2. (Buat bounding kotak di pinggir object terdeteksi) proses ini bertujuan untuk mengetahui nilai sumbuk y dari object yang akan dihitung. Berikut adalah persamaan yang digunakan

   ```
   contours, hierarchy = cv2.findContours(mask, cv2.RETR_TREE, cv2.CHAIN_APPROX_NONE)
   x, y, w, h = cv2.boundingRect(contours[i])
   ```

3. (Buat garis untuk area perhitungan object di dalam frame) proses ini bertujuan agar proses perhitungan dapat dilakukan secara terstruktur pada daerah frame yang ditunjukan. Berikut adalah persamaan yang digunakan.

   ```
   width = src.shape[1] ;height = src.shape[0]
   CoorYEntranceLine = int((height / 2)-150)
   CoorYExitLine = int((height / 2)+150)
   cv2.line(warna_seleksi, (0,CoorYEntranceLine), (width,CoorYEntranceLine), (255, 0, 0), 2)
   cv2.line(warna_seleksi, (0,CoorYExitLine), (width,CoorYExitLine), (0, 0, 255), 2)
   ```

4. (Input nilai maksimum dan minimum Contour) proses ini bertujuan untuk menentukan object mana yang dipilih dan dihitung. Berikut adalah algoritma yang digunakan

   ```
   cv2.namedWindow("Tracking")
   cv2.resizeWindow("Tracking", 400, 400)
   cv2.createTrackbar("Upper", "Tracking", 0, 100000, nothing)
   cv2.createTrackbar("Lowwer", "Tracking", 0, 100000, nothing)
   Up_CA = cv2.getTrackbarPos("Upper", "Tracking")
   low_CA = cv2.getTrackbarPos("Lowwer", "Tracking")
   ```

5. Perhitungan object berdasarkan fungsi yang telah tertera. Berikut adalah persamaan yang digunakan dalam proses ini.

   ```
   start = False ; flag = False ; count_total = 0 ; max_count = 0 ; count = 0
   for i in range(len(contours)):
           x, y, w, h = cv2.boundingRect(contours[i])
                   if Up_CA > cv2.contourArea(contours[i]) > low_CA and y > CoorYEntranceLine and y < CoorYExitLine:
                    # print("area", i, ":", cv2.contourArea(contours[i]))
   ```

```
cv2.rectangle(warna_seleksi, (x, y), (x + w, y + h), (0, 255, 0),
2)
# print("area", cv2.contourArea(contours[i]))
count += 1
# print(count)
if count > max_count:
 max_count = count
flag = True
 if start and flag and count == 0:
count_total += max_count
 max_count = 0
flag = False
```

Seluruh nilai dari hasil perhitungan akan dimasukkan ke database dan juga akan diolah datanya untuk mengetahui nilai akurasi dari proses perhitungan. Data ini dibutuhkan untuk melakukan Analisa pengaruh warna, bentuk dan kondisi pada saat melakukan proses perhitungan barang. Untuk mempermudah proses perhitungan maka akan dibuatkan table bantuan dengan format sebagai berikut.

Tabel 3.2. Tabel bantuan perhitungan nilai akurasi

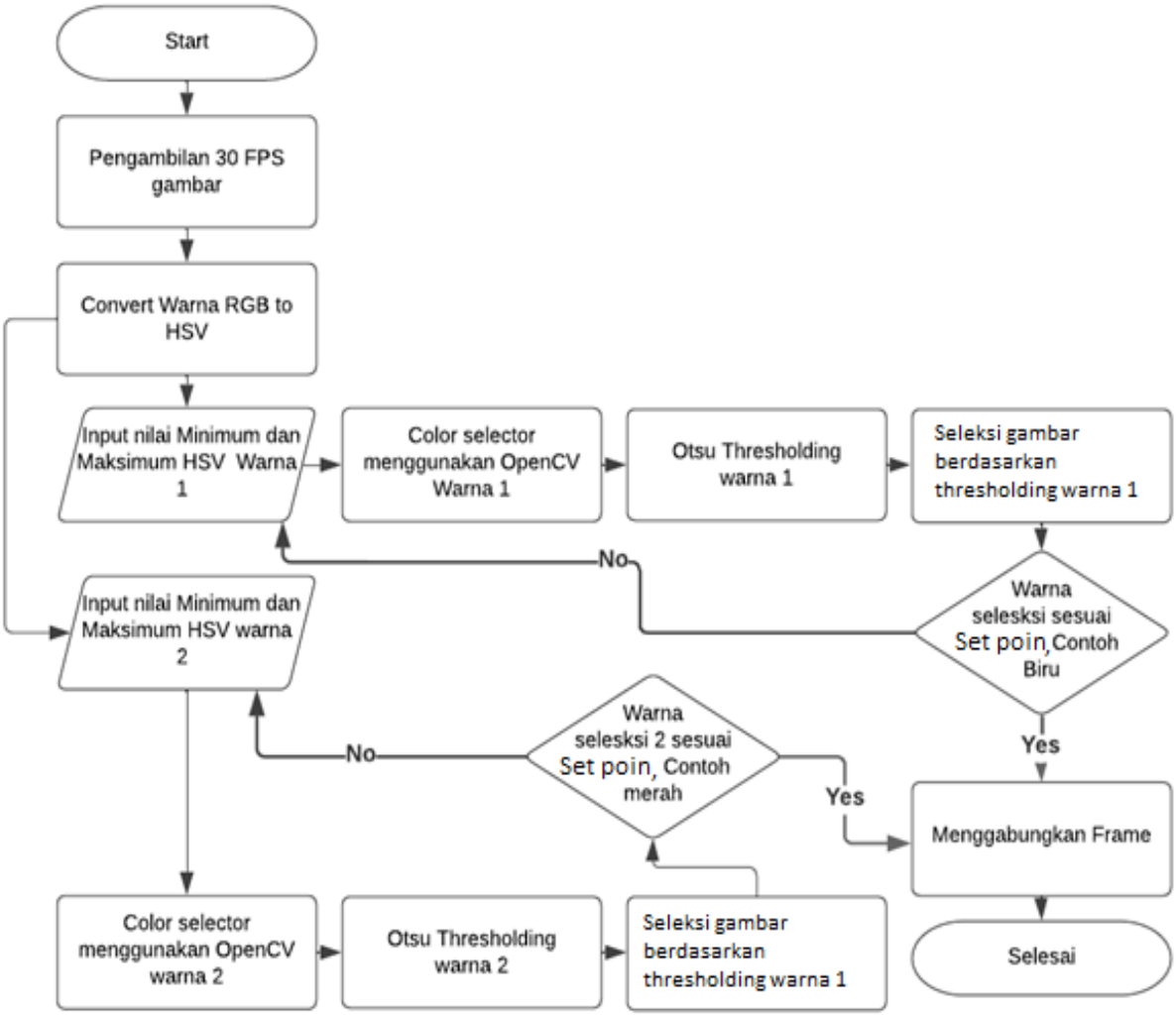
No	Nama Sparepart	Jumlah perhitungan pada sistem	Jumlah perhitungan manual	Nilai Akurasi
1.	Top Casing	….	….	….
2	Mainboard	….	….	….
3	Bottom Casing	….	….	….
4	Battrey Cover	….	….	….

Untuk perhitungan nilai akurasi akan digunakan persamaan yang didapatkan dari buku Teknik pengukuran oleh (John P Bentley, 2005). Berikut adalah persamaan untuk menghitung nilai akurasi pengukuran.

$$Akurasi = \left(1 - \left(\frac{Hasil\ perhitungan\ manual - Hasil\ perhitungan\ Otomatis}{Hasil\ Perhitungan\ manual}\right)\right) \times 100\% \qquad (3.1)$$

3.6 Pembuatan sistem penghitung nilai dengan jenis barang yang sama namun dengan warna yang berbeda

Sistem perhitungan ini hamper sama dengan Sistem perhitungan yang dilakukan pada kondisi satu warna dan satu jenis yang sama. Proses perhitungannya pun dapat dilakukan scara berkala sesuai keinginan user. Namun demi mempersingkat pemilihan nilai minimum dan maksimum HSV setiap nilai maka hasil kalibrasi akan disimpan agar data dapat diolah dengan metode yang telah ditentukan. Perhatikan **Gambar 3.9**



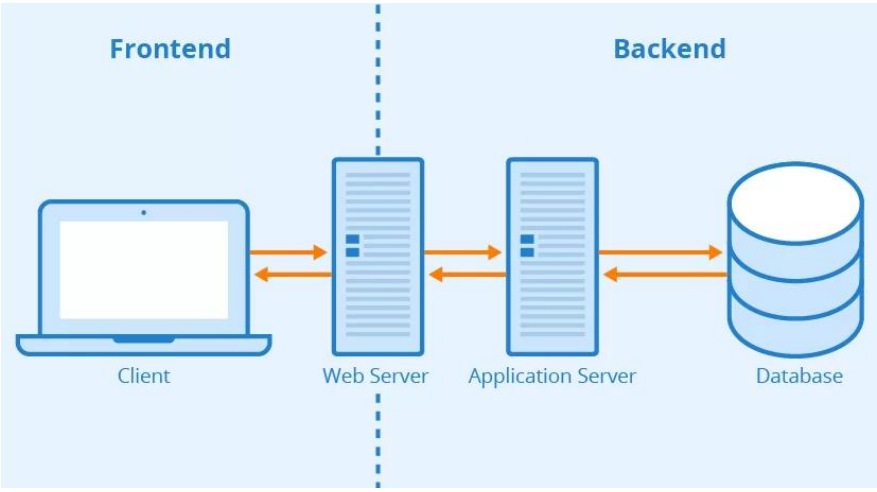
Gambar 3.9 Pengolahan warna berbeda

3.7 Pembuatan website di bagian *back-end* dengan *framework Django*

Django adalah sebuah framework full stack untuk membuat aplikasi web dengan bahasa pemrograman Python. Framework akan membantu dalam membuat web lebih cepat, dibandingkan menulis kode dari nol.

Full-stack artinya, django meliputi sisi front-end dan juga back-end. Front-end adalah sisi depan yang akan dilihat oleh pengguna, sedangkan back-end adalah sisi belakang yang

berhubungan dengan database dan logika bisnis **Gambar 3.10** merupakan ilustrasi singkat dari pengolahan data frontend dan backend.



Gambar 3.10 Django framework(*Django Overview | Django*, n.d.)

3.8 Pembuatan interface menggunakan HTML Localhost

Lokalhost digunakan sebagai server yang akan mengirimkan data di setiap client server yang mengaksesnya. HTML yang digunakan menggunakan Bahasa HTML sesuai dengan ketentuan framework Django. Dalam pembuatan sebuah website ada beberapa hal yang harus diperhatikan sebagai berikut ("Principles of User Interface Design: Important Rules That Every Designer Should Follow," 2015).

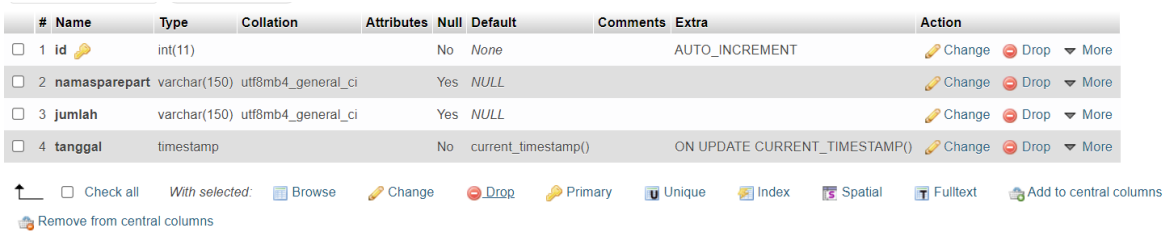
1. **"Jelas**" berarti sebuah UI harus memiliki tujuan yang jelas. Tujuan tersebut dapat disampaikan oleh sebuah desain web, agar user dapat menggunakan dan berinteraksi dengan sistem dengan mudah.
2. **"Familiar"** yaitu dengan salah satu kata yang sering didengar maka desain web adalah intuitif. Intuitif artinya layout yang digunakan bisa dimengerti dengan natural dan hanya dengan insthink user dapat memahaminya. Cara membuat layout yang intuitif adalah menggunakan desain web yang familiar, menggunakan icon yang sudah sering dilihat sebelumnya.
3. **"Responsif"** artinya UI melakukan respon terhadap tindakan user dengan cepat. Jika user menunggu sebuah website melakukan loading yang cukup lama maka user akan malas, dan jika loadingnya cepat maka user experience juga akan semakin baik.
4. **"Konsisten"** pada desain web UI dapat membantu user mengerti pola dari sebuah desain web UI, user dapat mempelajari apa kegunaan tombol, tabs, icon dan

berbagai elemen yang ada pada UI tersebut. Dengan begitu user dapat mengerjakan sesuatu lebih cepat dan mempelajari fitur baru dengan lebih cepat.

5. **"Menarik"** Sebuah desain web UI jika dibuat dengan menggunakan tampilan yang menarik, user akan lebih senang menggunakannya. Tentunya apa yang dianggap menarik untuk website perlu disesuaikan dengan user, namun harus tetap memperhatikan fungsi website agar tetap berfungsi dengan baik.
6. **"Efisien"** yaitu UI yang baik harus memperhatikan bahwa website bisa digunakan dengan efisien. Agar dapat membuat UI yang efisien terlebih dahulu mengetahui apa yang ingin dicapai oleh user dan perlu mengidentifikasi bagaimana website bekerja. Apa saja fungsinya dan apa saja kegunaanya.

3.9 Input data ke MySQL database

Melakukan integrasi program dengan melakukan penggabungan database MySQL dengan framework Django. Dimana hasil yang harus didapatkan adalah data dapat di simpan di database tanpa lost yang dapat berpengaruh ke dalam sistem. Struktur yang harus dimiliki meliputi Id, Nama Sparepart, Warna, Jumlah dan Tanggal Input data. Berikut adalah gambaran struktur table yang digunakan untuk data penyimpanan.



#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	id	int(11)			No	None		AUTO_INCREMENT	Change Drop More
2	namasparepart	varchar(150)	utf8mb4_general_ci		Yes	NULL			Change Drop More
3	jumlah	varchar(150)	utf8mb4_general_ci		Yes	NULL			Change Drop More
4	tanggal	timestamp			No	current_timestamp()		ON UPDATE CURRENT_TIMESTAMP()	Change Drop More

Check all With selected: Browse Change Drop Primary Unique Index Spatial Fulltext Add to central columns Remove from central columns

Gambar 3.11 Struktur Table MySQL

Halaman ini sengaja dikosongkan

BAB IV
HASIL DAN PEMBAHASAN

Pada BAB IV dijelaskan mengenai hasil dan pembahasan perancangan hardware dan hasil perancangan software sesuai dengan kondisi di lapangan. Pada proses pembuatan rancang bangun hardware dilakukan dua tahapan yaitu instalasi *raspberry pi 3 B* dan kamera pada konveyor *belt* serta pemasangan skat penghalang barang tertumpuk melewati konveyor belt. *Software* penghitung barang memiliki beberapa tahapan diantaranya proses pencarian nilai *set point* warna setiap *spare parts* untuk menentukan nilai minimal dan maksimal *Hue, Saturation* dan *Value (HSV).* Dilanjutkan dengan pendeteksian Tepi menggunakan metode *Sobel edge detection* dan proses identifikasi luas wilayah. Setelah didapatkan seluruh nilai tersebut maka dilanjutkan dengan perhitungan objek menggunakan metode *object color tracking* dan proses upload data menuju *database MySQL*.

4.1 Hasil instalasi *Raspberry Pi 3 B* dan kamera pada konveyor *belt*

Proses pengukuran suhu dan kelembaban ruangan menggunakan alat ukur Thermohygrometer dengan dengan hasil 28.5ºC dan nilai kelembaban adalah 72 %. Berdasarkan spesifikasi yang diberikan oleh pabrikan raspberry pi 3 b nilai tersebut termasuk dalam range operasi produk tersebut. Oleh karena itu pemasangan raspberry pi 3 B dapat dilakukan tanpa perlu perlindungan lebih terhadap suhu maupun kelembaban. Dilanjutkan dengan proses pengukuran intensitas cahaya. Dimana nilai ini akan berpengaruh pada proses identifikasi barang menggunakan metode *object color tracking*. Dimana didapatkan nilai intensitas cahaya di ruangan produksi PT. Sinko Prima Alloy sebesar 137,3 Lux. Nilai tersebut telah memenuhi syarat sebagai ruangan kerja berdasarkan standard ISO 9001 dimana nilai minimum pencahayaan di ruangan kerja adalah 100 Lux.

Gambar 4.1 (a) Hasil pengukuran suhu dan kelembaban ruangan (b) Hasil pengukuran intensitas cahaya

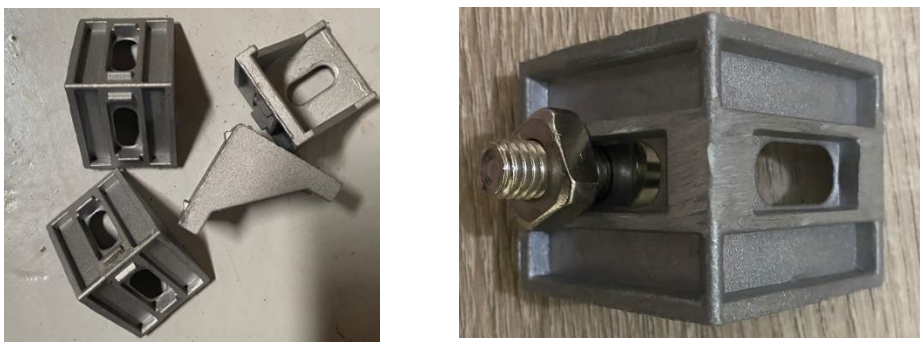
Gambar 4.1 (a) *Thermohygrometer* (b) Tampak hasil dari pembacaan alat ukur *Luxmeter* merek *TES 1339R* buatan Taiwan. Pada pemasangan kamera Nemesis dengan tipe NYK-96 terdapat beberapa bagian yang diperhatikan. Bagian pertama yang harus diamati adalah lebar konveyor, lebar konveyor milik PT. Sinko Prima Alloy adalah 90 cm dan didapatkan nilai ketinggian maksimum kamera dengan permukaan belt adalah sebesar 50 cm dengan settingan kamera 1920P. 1080p (1920 × 1080 px juga dikenal sebagai Full HD atau FHD dan BT.709) adalah satu set mode video HDTV definisi tinggi yang ditandai dengan 1080 garis horizontal resolusi vertikal

Kamera dipasangkan dengan tiang berbahan aluminium profile dan disambungkan dengan sambungan khusus aluminium profile yang sedikit di modifikasi. Modifikasi yang dilakukan bertujuan untuk mengubah ukuran baut menjadi M5 yang menghubungkan kamera dengan sambungan serta memotong alas dari sambungan aluminium profile tersebut.

(a) (b)

Gambar 4.2 (a) Kamera Ketika belum dimodifikasi (b) Tampak kamera setelah dimodifikasi

Terlihat pada **Gambar 4.3** (a) pada dasar sambungan terdapat grigi yang berfungsi mengunci sambungan agar tidak lepas dari rel. Namun pada **Gambar 4.3** (b) gerigi tersebut sudah hilang dikarenakan proses pemotongan menggunakan grinda konvensional.



(a) (b)

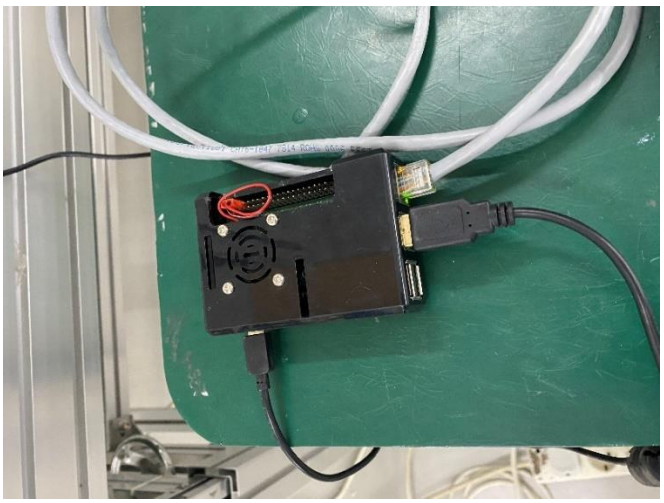
Gambar 4.3 (a) sambungan ketika belum dimodifikasi (b) Tampak sambungan setelah dimodifikasi

Proses dilanjutkan dengaan melakukan instalasi pemasangan kamera pada konveyor belt dengan kondisi mati. Pemasangan pertama yang dilakukan adalah pemasangan tiang penyangga dan dilanjutkan memasang kamera di tengah-tenga konveyor (50 cm dari bibir konveyor). Hasil pemasangan konveyor dapat dilihat pada **Gambar 4.4**

(a) (b)

Gambar 4.4 (a) Tampak depan instalasi kamera (b) Tampak belakang instalasi kamera

Proses selanjutnya adalah instalasi raspberry pada meja khusus di samping konveyor belt. Instalasi ini meliputi Pemasangan casing dan fan pada raspberry, Pemasangan kabel LAN yang akan dijadikan serial komunikasi ke PC agar dapat di monitoring, Pemasangan kable USB sebagai penghubung ke kamera yang telah terpasang pada konveyor belt, Pemasangan kabel power sebagai sumber tegangan raspberry. Gambar 4.5 adalah tampak raspberry setelah dilakukan proses instalasi.



Gambar 4.5 Hasil instalasi *Rapberry Pi 3 B*

Dilakukan pengukuran terhadap kecepatan konveyor dalam membawa barang untuk melihan fitur autofocus kamera yang berfungsi atau tidak. Berikut adalah hasil nilai kecepatan konveyor dengan menggunakan digital Tachometer merk AZ Taiwan. Pada

kondisi kecepatan minimum didapatkan nilai RPM konveyor belt sebesar 17,782 RPM. Pada kondisi kecepatan sedang didapatkan nilai RPM konveyor belt sebesar 52,780 RPM. Pada kondisi kecepatan penuh didapatkan nilai RPM konveyor belt sebesar 76,570 RPM. **Gambar 4.6** merupakan hasil dari setiap perhitungan.

Gambar 4.6 Hasil pembacaan Tachometer

Dari data tiga hasil pengukuran dengan kecepatan berbeda tersebut dapat dibuat tambel kecepatan konveyor seperti **Tabel 4.1**

Tabel 4.1 Nilai kecepatan konveyor dengan tiga kondisi

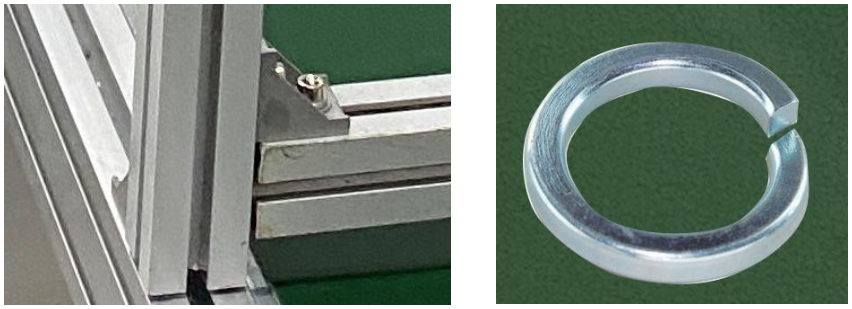
No	Jenis Kecepatan	Nilai Potensio Kecepatan	Nilai Pembacaan Alat ukur (RPM)
1	Minimum	10	17,782
2	Sedang	50	52,780
3	Maximum	100	76,570

4.2 Hasil instalasi penghalang barang tertumpuk pada konveyor

Pembuatan skat pembatas pada konveyor belt menggunakan bahan yang sama dengan pembuatan stand kamera yaitu *aluminium profile*. Terdapat beberapa kendala yang didapatkan dalam proses pembuatan skat pembatas tersebut seperti akurasi dari potongan tiang. Jarak sebenarnya dari tiang satu ke tiang lainnya adalah 90 cm namun hasil pemotongan aluminium profile adalah 80,05 cm. Kesalahan ini disebabkan oleh proses pemotongan secara manual menggunakan gerinda konvensional tidak dibantu oleh stand pengunci material. Kondisi tersebut mengakibatkan benda dapat bergeser saat proses pemotongan dan terjadi kesalahan dalam hasil pemotongan.

Untuk pemasangan hasil potongan aluminium profile dengan kondisi yang tidak sesuai dengan ukuran tetap dilakukan karena jumlah material yang sangat minim dan harga dari

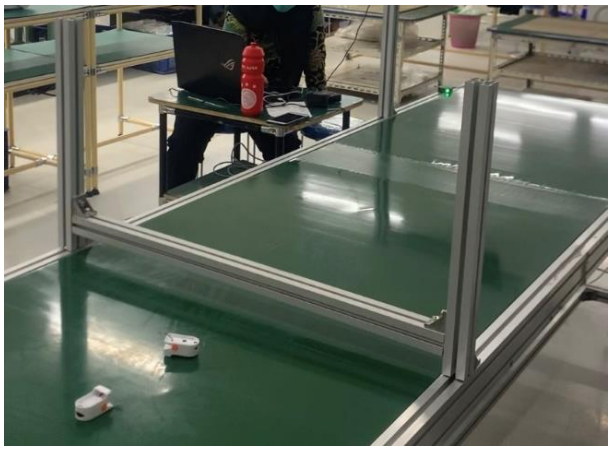
aluminium profile yang cukup mahal. Maka dilakukan proses penyambungan material dengan menggunakan sambungan alumium profile yang dilonggarjan dan baut dikombinasikan dengan washer spring agar daya cengkram sambungan kuat. Perhatikan **Gambar 4.7**



(a) (b)

Gambar 4.7 (a) hasil dari proses penyambungan (b) *Washer spring* yang digunakan

Didapatkan hasil perakitan dari aluminium profile untuk dijadikan skat penghalang seperti pada **Gambar 4.8**



Gambar 4.8 Hasil perakitan skat penghalang barang tertumpuk

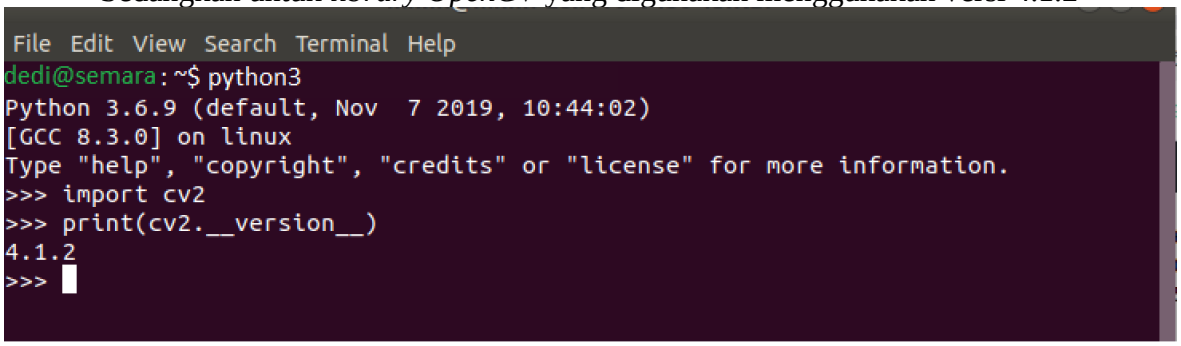
4.3 Hasil instalasi *PIP* dan *library OpenCV* pada *Raspberry Pi 3 B*

Instalasi dilakukan pada operation system ubuntu pada *raspberry pi 3 b.* Versi *PIP* yang digunakan adalah 9.0.1 dengan penyimpanan pada *folder python*.

```
Setting up python3-setuptools (39.0.1-2) ...
Setting up python3.6-dev (3.6.7-1~18.04) ...
Setting up dh-python (3.20180325ubuntu2) ...
Setting up libpython3-dev:amd64 (3.6.7-1~18.04) ...
Setting up build-essential (12.4ubuntu1) ...
Setting up python3-dev (3.6.7-1~18.04) ...
Processing triggers for libc-bin (2.27-3ubuntu1) ...
dedi@semara:~$ pip3 --version
pip 9.0.1 from /usr/lib/python3/dist-packages (python 3.6)
```

Gambar 4.9 Hasil instalasi *PIP3 Python*

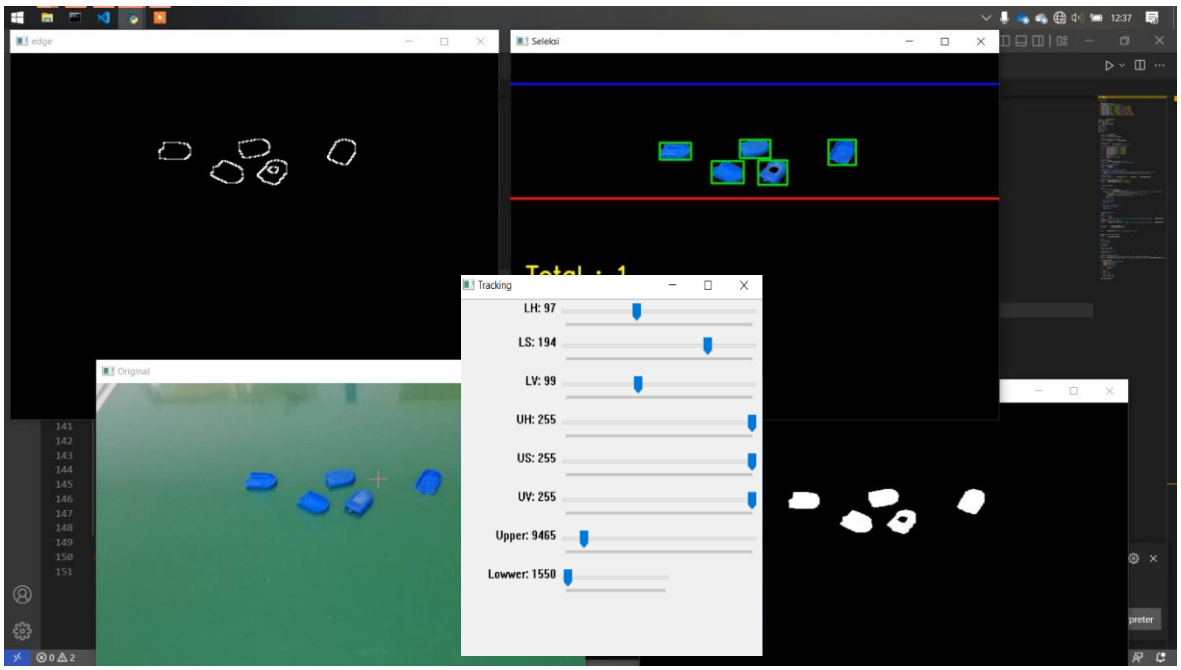
Sedangkan untuk *library OpenCV* yang digunakan menggunakan versi 4.1.2



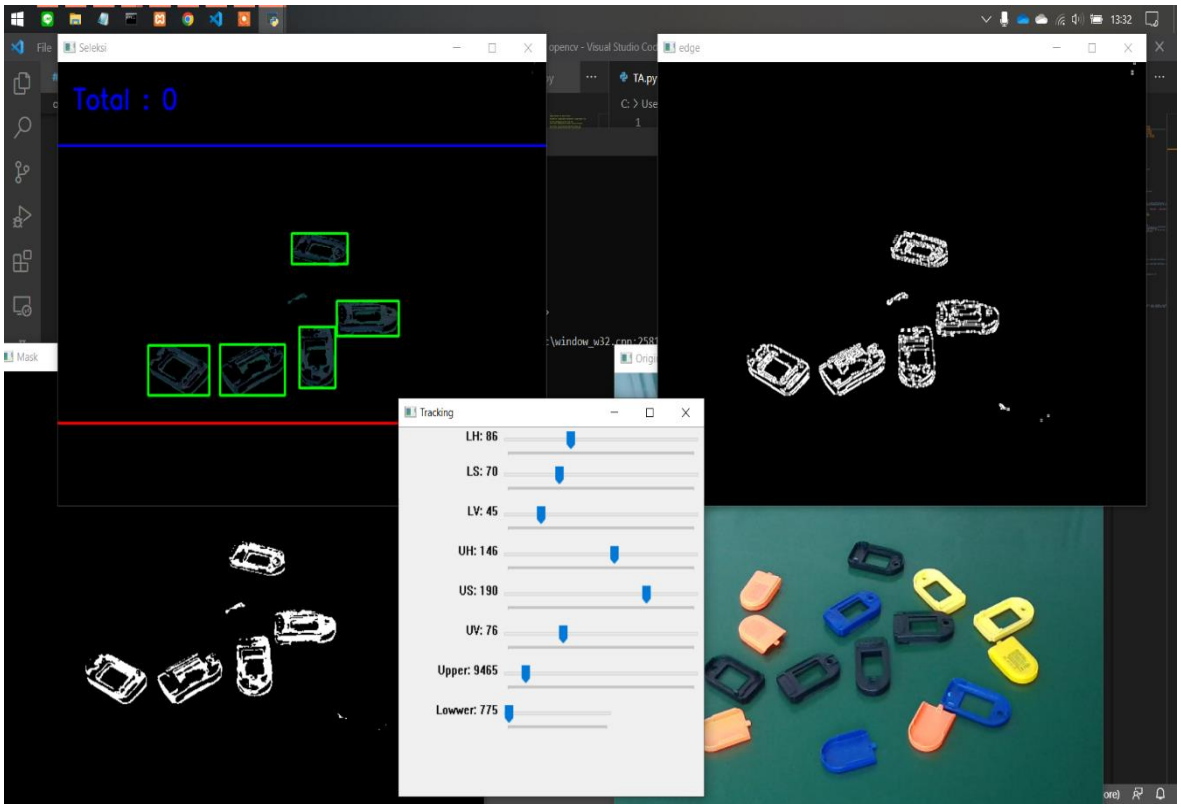
Gambar 4.10 Hasil instalasi *Library OpenCV*

4.4 Hasil pencarian nilai *set point HSV* warna dari setiap *spare parts FOX-BABY*

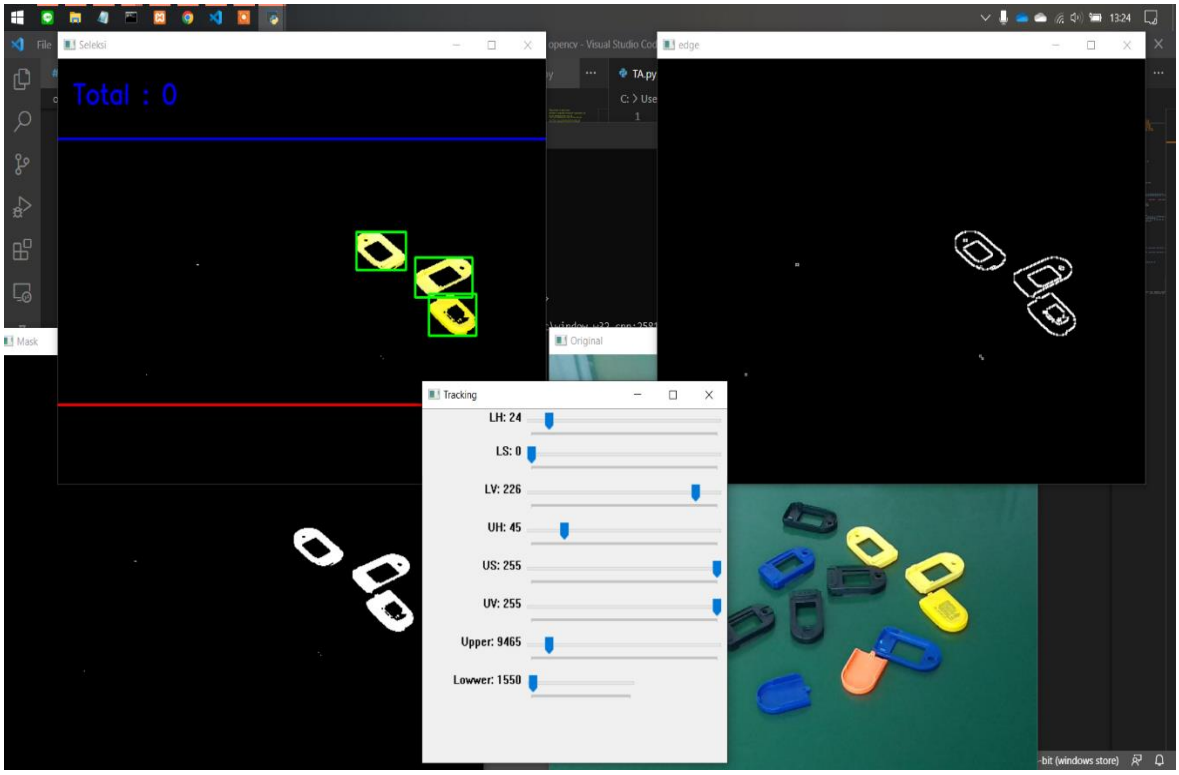
Pada proses ini dilakukan percobaan memberikan nilai maksimum dan minimum *HSV* pada gambar yang telah ditangkap oleh kamera dengan tujuan mencari *set point* setiap warna *spare parts FOX-BABY*. Pada proses ini dilakukan penangkapan gambar dengan ketentuan pencahayaan ruangan sebesar 137,3 Lux dan kamera yang digunakan adalah NEMESIS tipe A96. Untuk warna yang dihitung adalah warna biru, kuning, Orange, Putih, Hitam, dan hijau muda. Program yang digunakan dalam proses ini menggunakan tracker berbasis OpenCV dan juga tracker berbasis website dengan Bahasa pemrograman *JavaScript*. *JavaScript library* yang digunakan pada proses ini adalah *Ajax* dengan ketentuan nilai minimum dan maksimum *Hue* = 255, Saturation = 255, dan *Value* = 179.



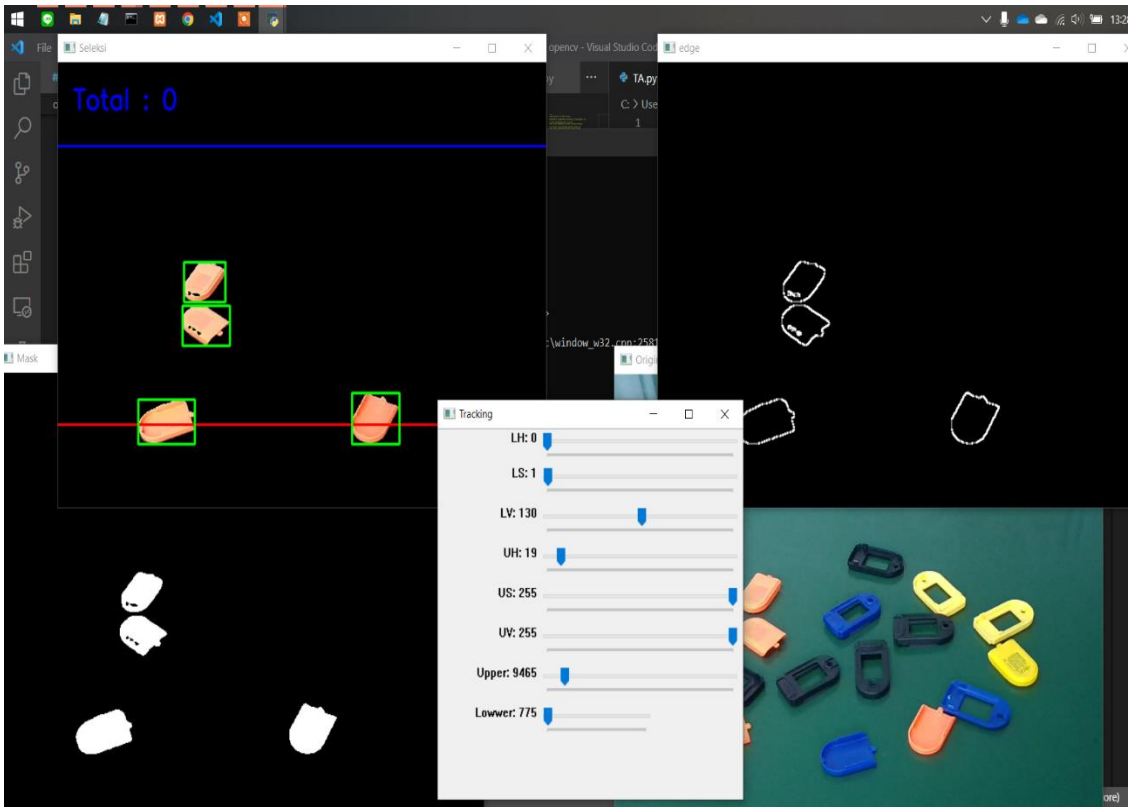
Gambar 4.11 Proses pencarian nilai *HSV* biru dengan sensor kamera



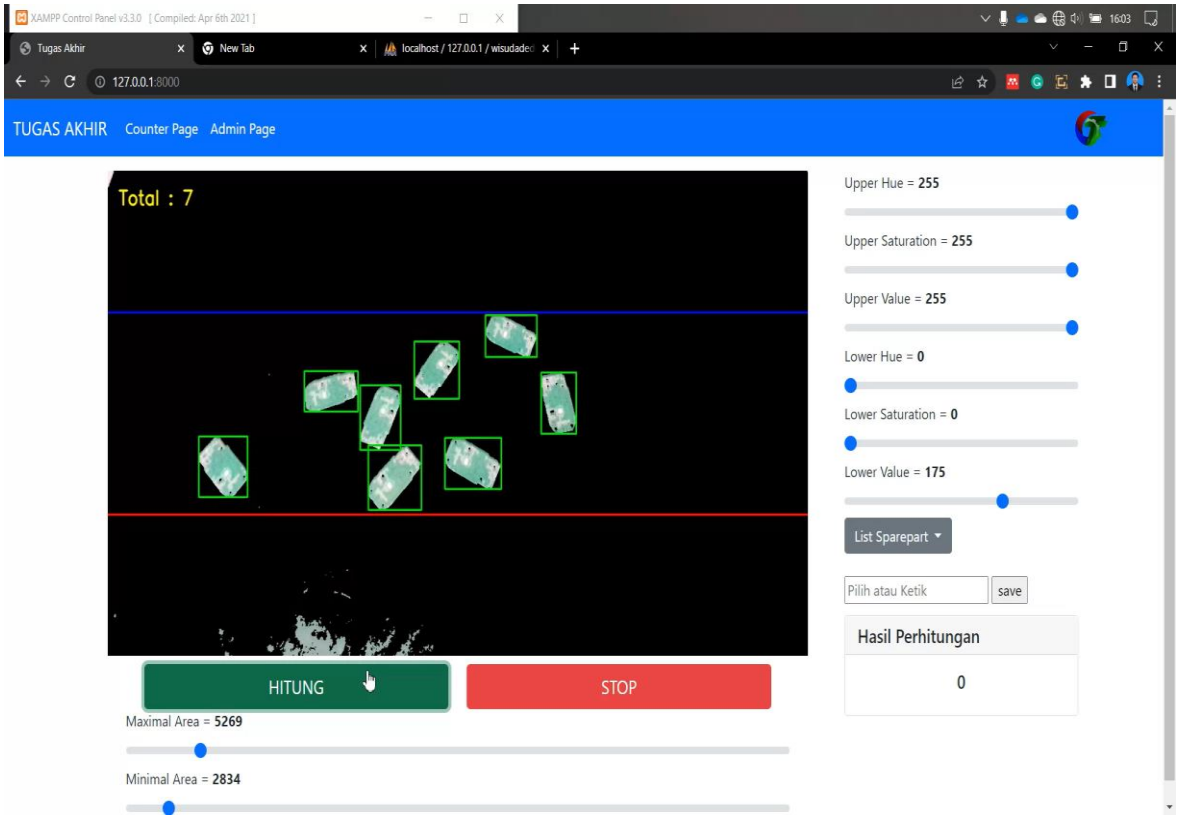
Gambar 4.12 Proses pencarian nilai *HSV* Hitam dengan sensor kamera



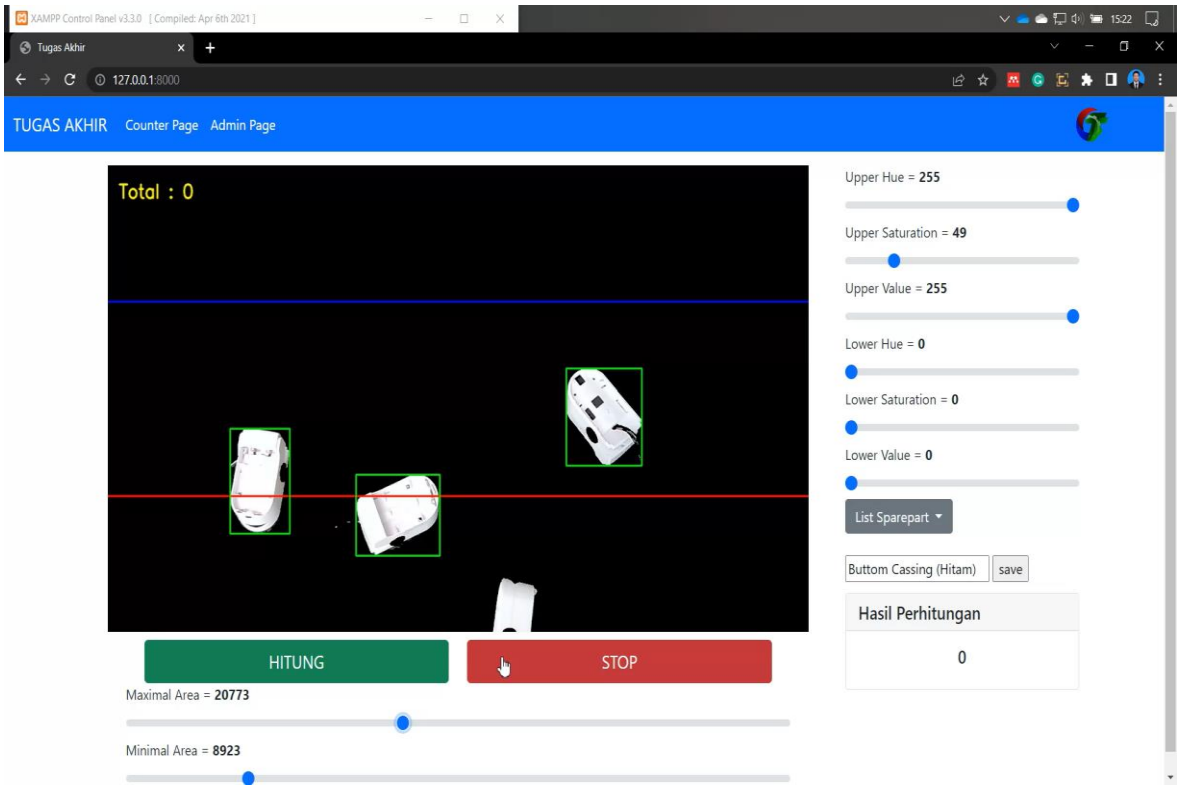
Gambar 4.13 Proses pencarian nilai *HSV* kuning dengan sensor kamera



Gambar 4.14 Proses pencarian nilai *HSV* orange dengan sensor kamera



Gambar 4.15 Proses pencarian nilai *HSV* hijau muda dengan sensor kamera



Gambar 4.16 Proses pencarian nilai *HSV* putih dengan sensor kamera

Dari **Gambar 4.11, Gambar 4.12, Gambar 4.13** sampai dengan **Gambar 4.16** didapatkan nilai minimum dan maksimum HSV dari warna biru, hitam, kuning, orange, hijau muda dan putih pada **Table 4.2**

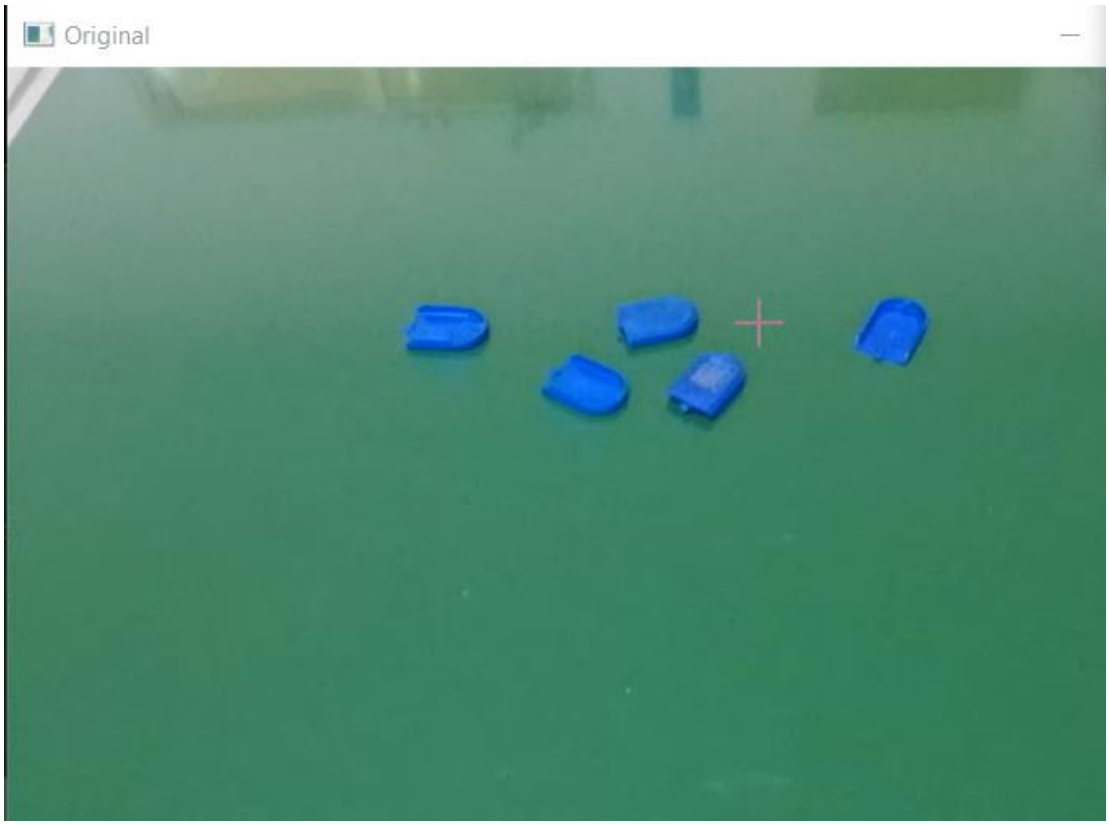
Tabel 4.2 Hasil nilai minimum dan maksimum *HSV* pada proses pendeteksian warna

No	Warna	Min Hue	Min Saturation	Min Value	Max Hue	Max Saturation	Max Value
1	Biru	97	194	99	255	255	255
2	Hitam	86	70	45	146	190	76
3	Kuning	24	0	226	45	255	255
4	Orange	0	1	130	19	255	255
5	Hijau muda	0	0	0	255	255	175
6	Putih	0	49	0	255	255	255

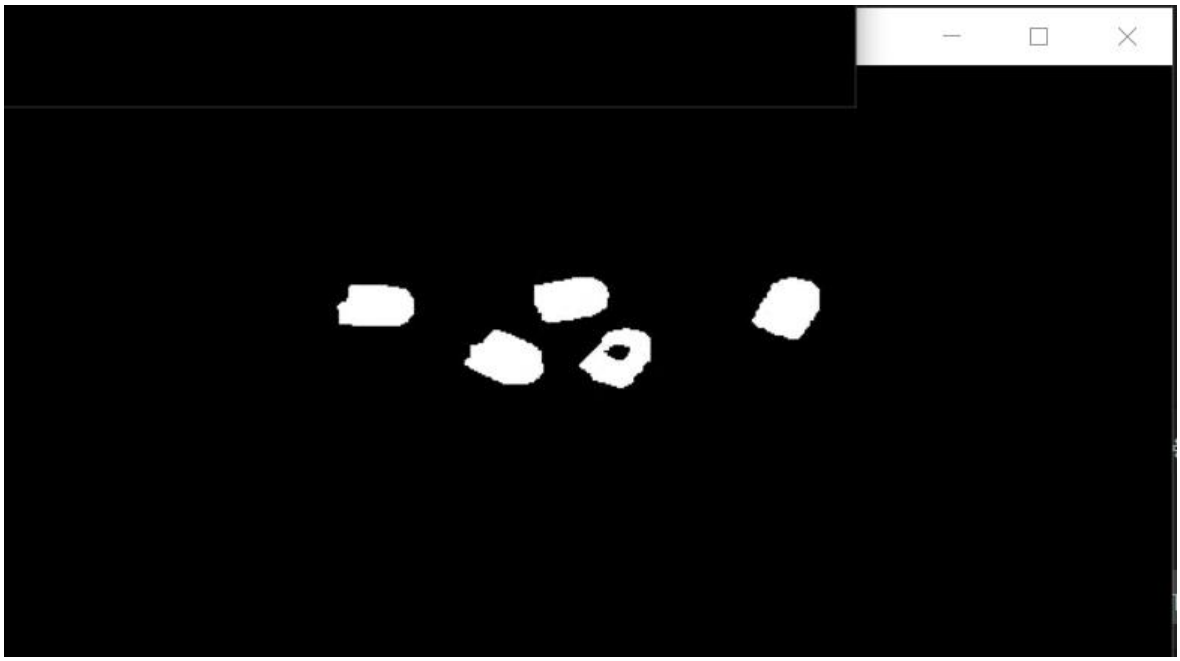
4.5 Hasil pendeteksian tepi menggunakan metode *Sobel edge detection* dan proses identifikasi luas wilayah

Pemfilteran citra (*image filtering*) adalah salah satu operasi pengolahan citra yang digunakan untuk menekan frekuensi tinggi pada citra seperti memperhalus citra (*smoothing*),

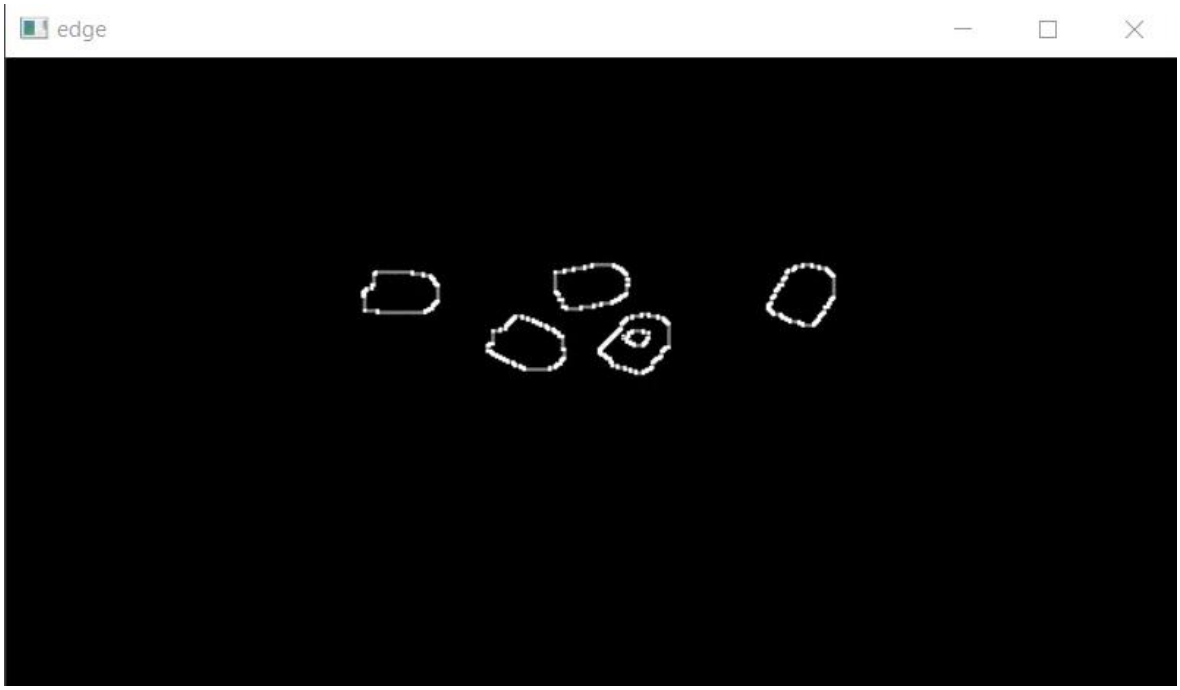
atau menekan frekuensi rendah seperti memperjelas atau mendeteksi tepi pada citra. Tujuan utama adanya proses pemfilteran citra adalah membuat citra menjadi tampak lebih baik, atau tampak lebih jelas untuk analisis. Oleh karenanya, pemfilteran citra disebut juga penyaringan citra. Sama seperti proses filtering pada umumnya dimana semua proses saling berkaitan. *image filtering* bekerja dengan cara menjalankan sebuah 'kernel' (ingat biji jagung: berukuran kecil) atau matriks persegi dengan ukuran tertentu pada seluruh citra secara bertahap. Inilah mengapa operasi kernel pada seringkali disebut sebagai 'moving window'. Proses pendeteksian tepi meliputi beberapa proses yang cukup banyak diantaranya proses pemilihan warna sesuai dengan standar dan penyimpanan data gambar. Setelah melakukan pemilahan warna didapatkan gambar yang sudah dipilah dengan cara menumpukkan **Gambar 4.17** dengan **Gambar 4.18** Sehingga gambar original dapat dipotong hingga menghasilkan **Gambar 4.20.** Setelah **Gambar 4.18** didapatkan dilanjutkan untuk mencari nilai tepi dengan cara membuat titik-titik di sekitaran luas wilayah yang telah diseleksi dengan menggunakan metode *sobel edge detection*. Titik-titik tersebut akan menghasilkan garis yang akan menjadi luas area dari objek yang akan dihitung. Maka dapat dihasilkan **Gambar 4.19** yang merupakan bentuk benda sesuai dengan gambar original pada **Gambar 4.17**



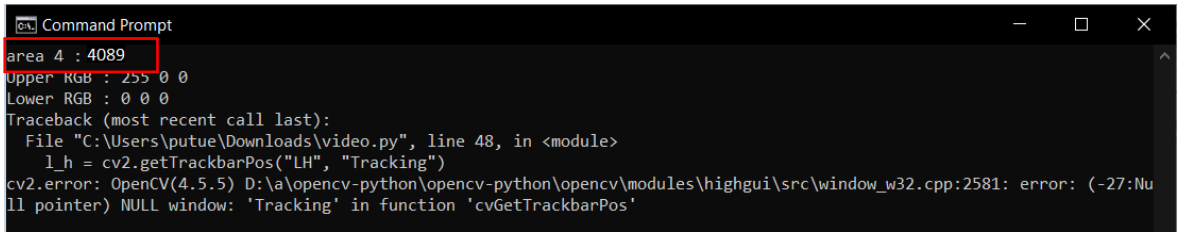
Gambar 4.17 Gambar Original tanpa *filter*



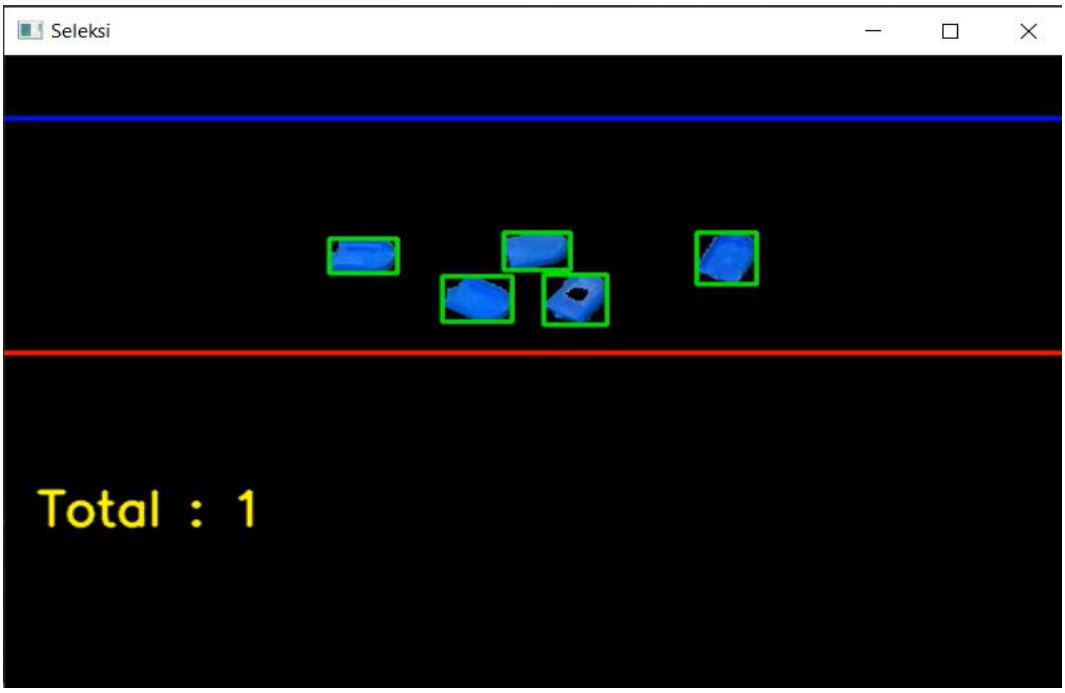
Gambar 4.18 Hasil *Masking* yang didapat dari proses seleksi warna



Gambar 4.19 Hasil deteksi tepi



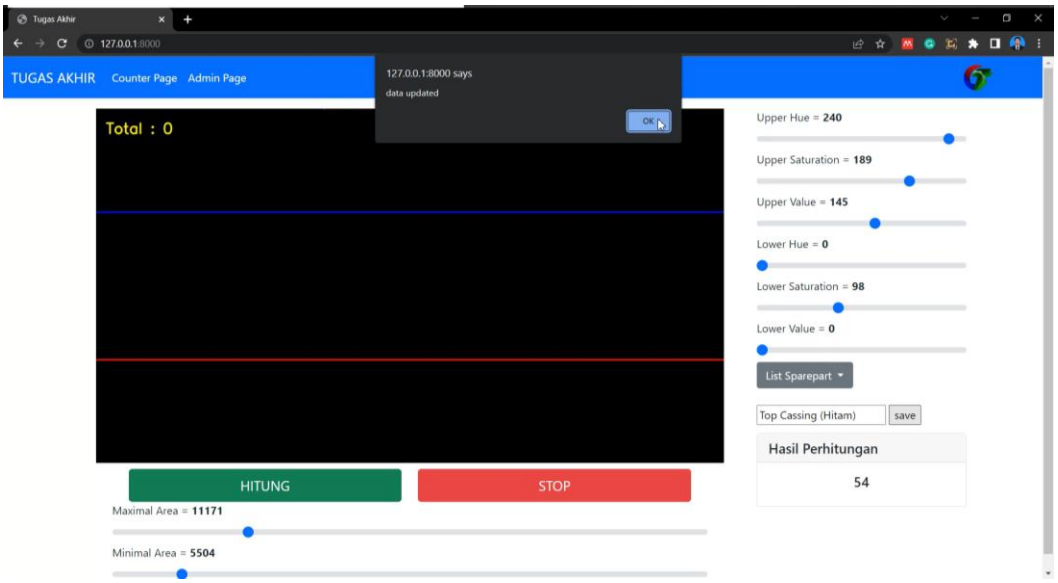
Gambar 4.20 Hasil pengukuran luas area objek dengan metode *sobel edge detection*



Gambar 4.21 Hasil seleksi objek

4.6 Hasil perhitungan *spare parts* FOX-BABY dengan metode *object color tracking*

Proses perhitungan sparepart dilakukan menggunakan website yang telah dibuat. **Gambar 4.22** adalah tampilan website sesuai dengan syarat.



Gambar 4.22 Tampilan interface website

Pada proses pengujian konveyor penghitung sparepart dengan mengunakan metode object color tracking digunakan 4 jenis barang dengan dan terdapat 6 warna berbeda. Jumlah maksimal perhitungan pada percobaan ini adalah 20 pcs. Dengan tujuan akhir adalah mencari nilai akurasi dari setiap warna yang diuji.

Berikut adalah rumus mencari nilai akurasi perhitungan (John P Bentley, 2005).

$$Akurasi = \left(1 - \left(\frac{Hasil\ perhitungan\ manual - Hasil\ perhitungan\ Otomatis}{Hasil\ Perhitungan\ manual}\right)\right) \times 100\%$$

Percobaan akan dilakukan dengan menghitung unit secara langsung (tanpa distraksi) dan memberikan distraksi lebih dari atau sama dengan 50 persen (Distraksi ≥ 50%) dari total unit percobaan. Sedangkan untuk penentuan setting kecepatan untuk mendapatkan akurasi maksimal. Maka dilakukan percobaan dengan menghitung 20 *battery cover* dengan tiga kecepatan berbeda. Didapatkan hasil percobaan sesuai dengan tabel **Tabel 4.3**

Tabel 4.3 Hasil perhitungan 20 pcs *battery cover* berdasarkan kecepatan konveyor

No	Kecepatan	Hasil perhitungan manual	Hasil perhitungan otomatis	Nilai akurasi (%)
1	Rendah	20	20	100
2	Sedang	20	20	100
3	Tinggi	20	19	95

Dari **Tabel 4.3** dapat dilihat bahwa nilai akurasi perhitungan pada kecepatan tinggi kurang dari 98% sehingga ada dua opsi dalam melakukan perhitungan untuk mencari nilai akurasi perhitungan yaitu kecepatan rendah dan sedang. Untuk mengefisiensi waktu pengujian maka dipilih kecepatan sedang untuk pelaksanaan pengujian secara menyeluruh.

a. Hasil percobaan perhitungan Top Cassing

Top Cassing unit FOX-BABY memiliki empat jenis warna yaitu warna biru, hitam, kuning dan orange. Dari berikut adalah beberapa tabel hasil perhitungan manual serta hasil perhitungan otomatis setiap warna.

Tabel 4.4 Perhitungan *Top Cassing* biru tanpa distraksi

No	Pengulangan	Jumlah perhitungan manual	Jumlah perhitungan otomatis	Nilai akurasi (%)
1	1	1	1	100
2	2	5	5	100
3	3	10	10	100
4	4	20	20	100

Tabel 4.5 Perhitungan *top cassing* biru dengan distraksi *top cassing* kuning (Distraksi ≥ 50%)

No	Jumlah distraksi	Jumlah perhitungan manual	Jumlah perhitungan otomatis	Nilai akurasi (%)
1	1	1	1	100
2	3	5	5	100
3	5	10	10	100
4	10	20	20	100

Tabel 4.6 Perhitungan *top cassing* hitam tanpa distraksi

No	Jumlah perhitungan manual	Jumlah perhitungan otomatis	Nilai akurasi (%)
1	1	1	100
2	5	5	100
3	10	10	100
4	20	22	91

Tabel 4.7 Perhitungan *top cassing* hitam dengan distraksi *top cassing* kuning (Distraksi ≥ 50%)

No	Jumlah distraksi	Jumlah perhitungan manual	Jumlah perhitungan otomatis	Nilai akurasi (%)
1	1	1	3	33
2	3	5	5	100
3	5	10	11	91
4	10	20	18	90

Tabel 4.8 Perhitungan *top cassing* kuning tanpa distraksi

No	Jumlah perhitungan manual	Jumlah perhitungan otomatis	Nilai akurasi (%)
1	1	1	100
2	5	5	100
3	10	10	100
4	20	20	100

Tabel 4.9 Perhitungan *top cassing* kuning dengan distraksi *top cassing* biru (Distraksi ≥ 50%)

No	Jumlah distraksi	Jumlah perhitungan manual	Jumlah perhitungan otomatis	Nilai akurasi (%)
1	1	1	1	100
2	3	5	5	100
3	5	10	10	100
4	10	20	20	100

Tabel 4.10 Perhitungan *top cassing* orange tanpa distraksi

No	Jumlah perhitungan manual	Jumlah perhitungan otomatis	Nilai akurasi (%)
1	1	1	100
2	5	5	100
3	10	10	100
4	20	20	100

Tabel 4.11 Perhitungan *top cassing* orange dengan distraksi *top cassing* biru (Distraksi ≥ 50%)

No	Jumlah distraksi	Jumlah perhitungan manual	Jumlah perhitungan otomatis	Nilai akurasi (%)
1	1	1	1	100
2	3	5	5	100
3	5	10	10	100
4	10	20	20	100

b. Hasil percobaan perhitungan *battery cover*

Top Cassing unit FOX-BABY memiliki empat jenis warna yaitu warna biru, hitam, kuning dan orange. Dari berikut adalah beberapa tabel hasil perhitungan manual serta hasil perhitungan otomatis setiap warna.

Tabel 4.12 Perhitungan *battery cover* biru tanpa distraksi

No	Jumlah perhitungan manual	Jumlah perhitungan otomatis	Nilai akurasi (%)
1	1	1	100
2	5	5	100
3	10	10	100
4	20	20	100

Tabel 4.13 Perhitungan *battery cover* biru dengan distraksi *battery cover* kuning (Distraksi ≥ 50%)

No	Jumlah distraksi	Jumlah perhitungan manual	Jumlah perhitungan otomatis	Nilai akurasi (%)
1	1	1	1	100
2	3	5	5	100
3	5	10	10	100
4	10	20	20	100

Tabel 4.14 Perhitungan *battery cover* hitam tanpa distraksi

No	Jumlah perhitungan manual	Jumlah perhitungan otomatis	Nilai akurasi (%)
1	1	1	100
2	5	5	100
3	10	10	100
4	20	20	100

Tabel 4.15 Perhitungan *battery cover* hitam dengan distraksi *battery cover* kuning (Distraksi ≥ 50%)

No	Jumlah distraksi	Jumlah perhitungan manual	Jumlah perhitungan otomatis	Nilai akurasi (%)
1	1	1	1	100
2	3	5	5	100
3	5	10	10	100
4	10	20	22	91

Tabel 4.16 Perhitungan *battery cover* kuning tanpa distraksi

No	Jumlah perhitungan manual	Jumlah perhitungan otomatis	Nilai akurasi (%)
1	1	1	100
2	5	5	100
3	10	10	100
4	20	20	100

Tabel 4.17 Perhitungan *battery cover* kuning dengan distraksi *battery cover* biru (Distraksi ≥ 50%)

No	Jumlah distraksi	Jumlah perhitungan manual	Jumlah perhitungan otomatis	Nilai akurasi (%)
1	1	1	1	100
2	3	5	5	100
3	5	10	10	100
4	10	20	20	100

Tabel 4.18 Perhitungan *battery cover* orange tanpa distraksi

No	Jumlah perhitungan manual	Jumlah perhitungan otomatis	Nilai akurasi (%)
1	1	1	100
2	5	5	100
3	10	10	100
4	20	20	100

Tabel 4.19 Perhitungan *battery cover* orange dengan distraksi *battery cover* biru (Distraksi ≥ 50%)

No	Jumlah distraksi	Jumlah perhitungan manual	Jumlah perhitungan otomatis	Nilai akurasi (%)
1	1	1	1	100
2	3	5	5	100
3	5	10	10	100
4	10	20	20	100

c. Percobaan perhitungan *buttom cassing*

Buttom cassing unit FOX-BABY memiliki dua jenis warna yaitu warna putih dan hitam. Dari berikut adalah beberapa tabel hasil perhitungan manual serta hasil perhitungan otomatis setiap warna.

Tabel 4.20 Perhitungan *buttom cassing* hitam tanpa distraksi

No	Jumlah perhitungan manual	Jumlah perhitungan otomatis	Nilai akurasi (%)
1	1	1	100
2	5	8	62
3	10	9	90
4	20	16	80

Tabel 4.21 Perhitungan *buttom cassing* hitam dengan distraksi *battery cover* kuning (Distraksi ≥ 50%)

No	Jumlah distraksi	Jumlah perhitungan manual	Jumlah perhitungan otomatis	Nilai akurasi (%)
1	1	1	5	2
2	3	5	5	100
3	5	10	9	90
4	10	20	16	80

Tabel 4.22 Perhitungan Buttom cassing putih tanpa distraksi

No	Jumlah perhitungan manual	Jumlah perhitungan otomatis	Nilai akurasi (%)
1	1	1	100
2	5	5	100
3	10	10	100
4	20	14	70

Tabel 4.23 Perhitungan *buttom cassing* putih dengan distraksi *battery cover* biru (Distraksi ≥ 50%)

No	Jumlah distraksi	Jumlah perhitungan manual	Jumlah perhitungan otomatis	Nilai akurasi (%)
1	1	1	3	33
2	3	5	8	62
3	5	10	5	50
4	10	20	11	55

d. Percobaan perhitungan *mainboard*

Mainboard unit FOX-BABY memiliki satu jenis warna yaitu warna hijau muda. Dari berikut adalah beberapa tabel hasil perhitungan manual serta hasil perhitungan otomatis setiap warna.

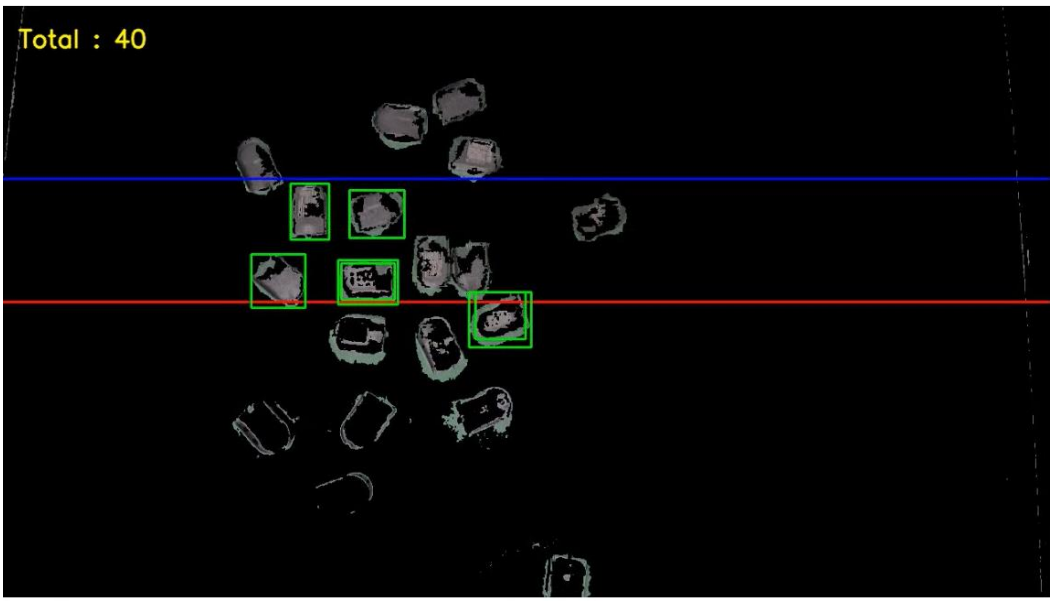
Tabel 4.24 Perhitungan *mainboard* hijau muda tanpa distraksi

No	Jumlah perhitungan manual	Jumlah perhitungan otomatis	Nilai akurasi (%)
1	1	1	100
2	5	5	100
3	10	10	100
4	20	17	85

Tabel 4.25 Perhitungan *mainboard* hijau muda dengan distraksi *battery cover* biru (Distraksi ≥ 50%)

No	Jumlah distraksi	Jumlah perhitungan manual	Jumlah perhitungan otomatis	Nilai akurasi (%)
1	1	1	1	100
2	3	5	5	100
3	5	10	10	100
4	10	20	16	80

Dari **Tabel 4.6, 4.7, 4.15, 4.20 dan 4.21** di atas dapat diketahui bahwa warna hitam memiliki akurasi perhitungan yang cukup rendah dikarenakan saat deteksi objek banyak *noise* yang muncul dan itu menyebabkan unit tidak terdeteksi dan terdeteksi lebih dari satu kali. Pada area perhitungan banyak unit yang tidak terdeteksi. Perhatikan **Gambar 4.23**



Gambar 4.23 Unit yang tidak terdeteksi pada area perhitungan

Berdasarkan jurnal "*An automatic car counting system using overfeat framework"* (Biswas et al., 2017). Dijelaskan bahwa jumlah minimum pengambilan data untuk mencari nilai rata-rata akurasi adalah 10 kali percobaan dengan nilai maksimum dari jumlah pembacaan alat ukur. Rumus untuk mencari nilai rata-rata akurasi adalah

$$Rata - Rata\ Nilai\ Akurasi = \left(\frac{Jumlah\ dari\ nilai\ Akurasi\ Total}{Banyaknya\ jumlah\ pengukuran}\right)$$

Untuk kasus FOX-BABY ini dikarenakan stok unit yang terbatas nilai maksimal yang digunakan adalah 20 pcs. Sehingga didapatkan data seperti pada berikut

Tabel 4.26 Perhitungan *top cassing* biru dengan pengulangan sebanyak 10 kali

No	Pengulangan	Jumlah perhitungan manual	Jumlah perhitungan otomatis	Nilai akurasi (%)
1	1	20	20	100
2	2	20	20	100
3	3	20	20	100
4	4	20	20	100
5	5	20	20	100
6	6	20	20	100
7	7	20	20	100
8	8	20	20	100
9	9	20	20	100
10	10	20	20	100
Rata-Rata Nilai Akurasi				100

Tabel 4.27 Perhitungan *top cassing hitam* dengan pengulangan sebanyak 10 kali

No	Pengulangan	Jumlah perhitungan manual	Jumlah perhitungan otomatis	Nilai akurasi (%)
1	1	20	16	80
2	2	20	15	75
3	3	20	18	90
4	4	20	20	100
5	5	20	17	85
6	6	20	18	90
7	7	20	18	90
8	8	20	15	75
9	9	20	14	70
10	10	20	18	90
Rata-Rata Nilai Akurasi				84.5

Tabel 4.28 Perhitungan *top cassing* kuning dengan pengulangan sebanyak 10 kali

No	Pengulangan	Jumlah perhitungan manual	Jumlah perhitungan otomatis	Nilai akurasi (%)
1	1	20	20	100
2	2	20	20	100
3	3	20	20	100
4	4	20	20	100
5	5	20	20	100
6	6	20	20	100
7	7	20	20	100
8	8	20	20	100
9	9	20	20	100
10	10	20	20	100
Rata-Rata Nilai Akurasi				100

Tabel 4.29 Perhitungan *top cassing* orange dengan pengulangan sebanyak 10 kali

No	Pengulangan	Jumlah perhitungan manual	Jumlah perhitungan otomatis	Nilai akurasi (%)
1	1	20	20	100
2	2	20	20	100
3	3	20	20	100
4	4	20	20	100
5	5	20	20	100
6	6	20	20	100
7	7	20	20	100
8	8	20	20	100
9	9	20	20	100
10	10	20	20	100
Rata-Rata Nilai Akurasi				100

Tabel 4.30 Perhitungan *battery cover* biru dengan pengulangan sebanyak 10 kali

No	Pengulangan	Jumlah perhitungan manual	Jumlah perhitungan otomatis	Nilai akurasi (%)
1	1	20	20	100
2	2	20	20	100
3	3	20	20	100
4	4	20	20	100
5	5	20	20	100
6	6	20	20	100
7	7	20	20	100
8	8	20	20	100
9	9	20	20	100
10	10	20	20	100
Rata-Rata Nilai Akurasi				100

Tabel 4.31 Perhitungan *battery cover* hitam dengan pengulangan sebanyak 10 kali

No	Pengulangan	Jumlah perhitungan manual	Jumlah perhitungan otomatis	Nilai akurasi (%)
1	1	20	14	70
2	2	20	17	85
3	3	20	18	90
4	4	20	17	85
5	5	20	19	95
6	6	20	16	80
7	7	20	18	90
8	8	20	15	75
9	9	20	16	80
10	10	20	18	90
Rata-Rata Nilai Akurasi				84

Tabel 4.32 Perhitungan *battery cover* kuning dengan pengulangan sebanyak 10 kali

No	Pengulangan	Jumlah perhitungan manual	Jumlah perhitungan otomatis	Nilai akurasi (%)
1	1	20	20	100
2	2	20	20	100
3	3	20	20	100
4	4	20	20	100
5	5	20	20	100
6	6	20	20	100
7	7	20	20	100
8	8	20	20	100
9	9	20	20	100
10	10	20	20	100
Rata-Rata Nilai Akurasi				100

Tabel 4.33 Perhitungan *battery cover* orange dengan pengulangan sebanyak 10 kali

No	Pengulangan	Jumlah perhitungan manual	Jumlah perhitungan otomatis	Nilai akurasi (%)
1	1	20	20	100
2	2	20	20	100
3	3	20	20	100
4	4	20	20	100
5	5	20	20	100
6	6	20	20	100
7	7	20	20	100
8	8	20	20	100
9	9	20	20	100
10	10	20	20	100
Rata-Rata Nilai Akurasi				100

Tabel 4.34 Perhitungan *buttom cassing* hitam dengan pengulangan sebanyak 10 kali

No	Pengulangan	Jumlah perhitungan manual	Jumlah perhitungan otomatis	Nilai akurasi (%)
1	1	20	16	80
2	2	20	15	75
3	3	20	18	90
4	4	20	18	90
5	5	20	15	75
6	6	20	14	70
7	7	20	16	80
8	8	20	19	95
9	9	20	20	100
10	10	20	15	75
Rata-Rata Nilai Akurasi				83

Tabel 4.35 Perhitungan *buttom cassing* putih dengan pengulangan sebanyak 10 kali

No	Pengulangan	Jumlah perhitungan manual	Jumlah perhitungan otomatis	Nilai akurasi (%)
1	1	20	17	85
2	2	20	14	70
3	3	20	15	75
4	4	20	13	65
5	5	20	15	75
6	6	20	13	65
7	7	20	17	85
8	8	20	14	70
9	9	20	17	85
10	10	20	15	75
Rata-Rata Nilai Akurasi				75

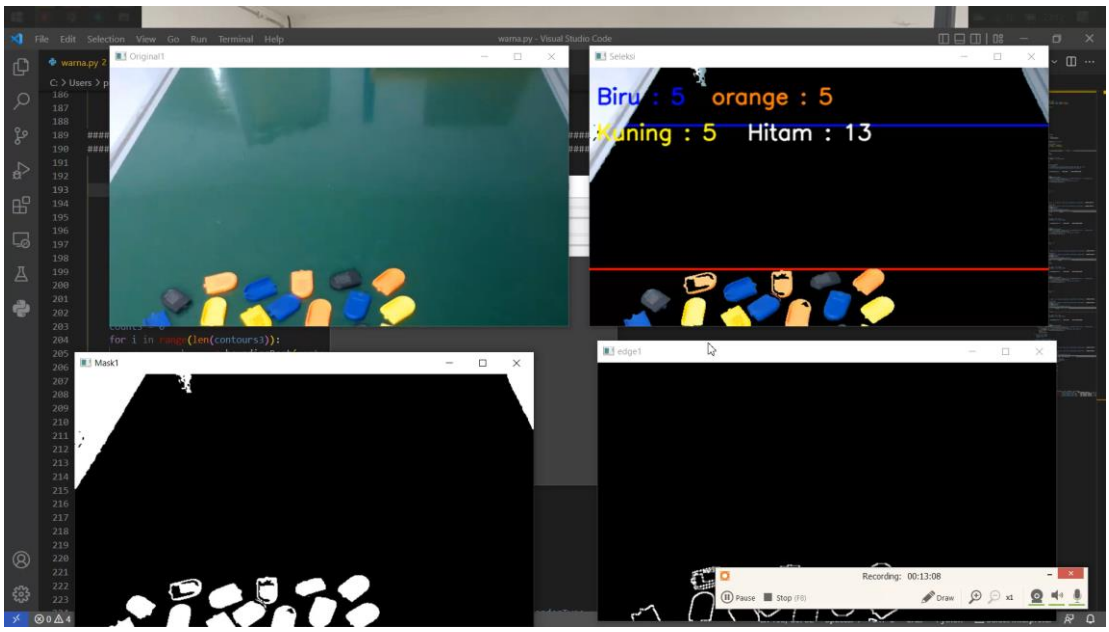
Tabel 4.36 Perhitungan *mainboard* hijau muda dengan pengulangan sebanyak 10 kali

No	Pengulangan	Jumlah perhitungan manual	Jumlah perhitungan otomatis	Nilai akurasi (%)
1	1	20	18	90
2	2	20	19	95
3	3	20	20	100
4	4	20	19	95
5	5	20	18	90
6	6	20	18	90
7	7	20	18	90
8	8	20	19	95
9	9	20	17	85
10	10	20	19	95
Rata-Rata Nilai Akurasi				92.5

Berdasarkan hasil percobaan didapatkan bahwan rata-rata nilai akurasi dari barang berwarna biru, kuning, dan orange adalah 100% maka dapat dikatakan keakurasian perhitungan sangat baik. Sedangkan untuk percobaan barang dengan warna hitam, hijau muda dan putih didapatkan nilai akurasi dibawah 95% maka dapat dikatakan keakurasian perhitungan kurang baik.

4.7 Hasil perhitungan banyak warna pada satu objek yang sama

Pada proses perhitungan banyak warna pada satu object dilakukan perhitungan dengan melewatkan 1 jenis barang (battery cover) dengan 4 warna yang berbeda serta jumlah barang per warna adalah 5 pcs.



Gambar 4.24 Pembacaan nilai objek satu jenis dengan empat warna

Dari hasil percobaan tersebut object berhasil terdeteksi dengan baik dan dapat dipilah sesuai dengan warna masing-masing. Namun saat proses perhitungan object pada conveyor belt nilai yang didapatkan tidak sesuai dengan nilai aktual. Nilai aktualnya adalah jumlah unit per warna sebanya 5 sedangkan hasil perhitungannya warna orange, kuning dan biru sama dengan nilai sebenarnya sedangkan warna hitam tidak sama dengan nilai sebenarnya dengan hasil perhitungan 13. Perhatikan percobaan perhitungan pada **Gambar 4.24**. Dari percobaan diatas, didapatkan data percobaan yang dituliskan pada **Tabel 4.37**

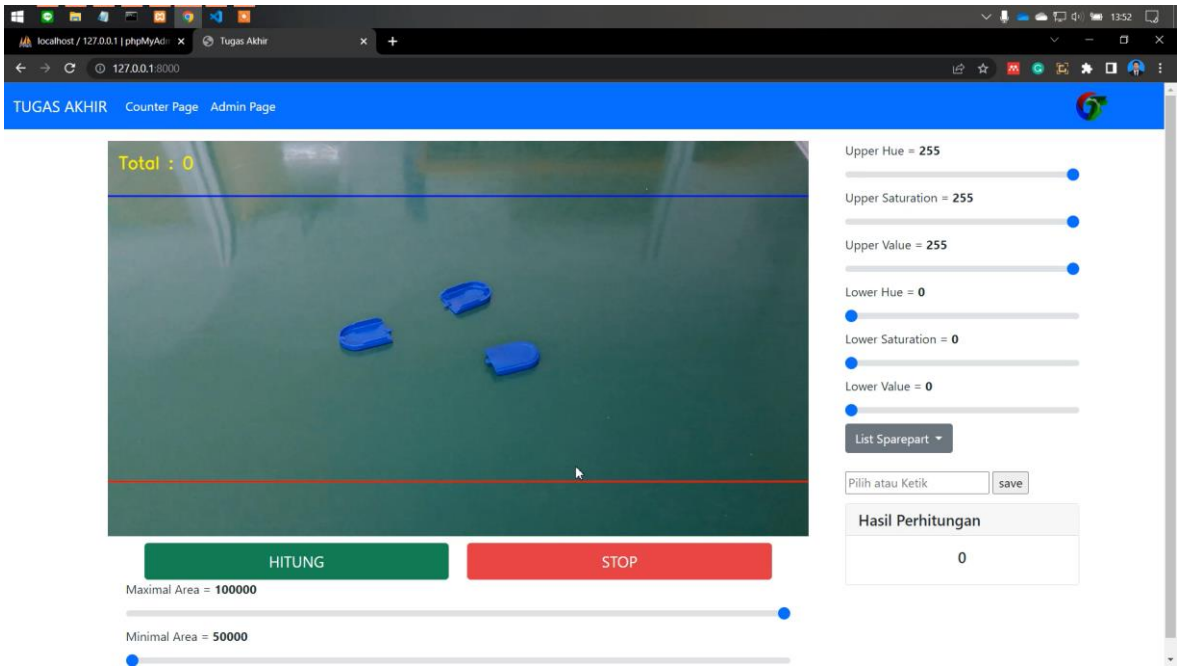
Tabel 4.37 Hasil perhitungan buttom cassing dengan beberapa warna

No	Warna	Hasil perhitungan manual	Hasil perhitungan otomatis	Nilai akurasi (%)
1	Biru	5	5	100
2	Hitam	5	13	38
3	Kuning	5	5	100
4	Orange	5	5	100

Namun untuk memperjelas hasil pengujian, maka dilakukan perhitungan dengan berbagai campuran warna dengan satu objek yang sama. Jenis objek yang digunakan dalam pengambilan data ini adalah battery casing dengan jumlah maksimal sebanyak 10 pcs dengan ketentuan pengambilan data yaitu dengan melewatkan 1 pcs, 3 pcs, 5 pcs dan terakhir 10 pcs. Warna dasar yang digunakan pada percobaan ini adalah warna Biru, Hitam, Kuning dan Orange. Terdapat beberapa campuran warna yaitu satu warna, dua warna, tiga warna dan empat warna. Hasil dari percobaan tersebut dilampirkan pada bab lampiran poin B.

4.8 Hasil pembuatan *website* dengan *framework Django*

Pembuatan *website* menggunakan *framework Django* dapat direalisasikan. Pembuatan *website* tersebut menggunakan *interface* dengan *library bootstrap* dan *CSS*. Terdapat beberapa kendala yang dialami saat proses pembuatan *website* dilakukan. Diantaranya adalah integrasi antara database *MySQL* dan *framework*. Dimana sebenarnya *framework Django* telah memiliki database yang sifatnya terbatas. *Database* tersebut bersifat terbatas dikarenakan database tersebut sebenarnya berfungsi sebagai penyimpanan *username* dan *password* serta strukturnya tidak seluas database *MySQL*. *Webserver* yang digunakan pada *website* ini bersifat *local host* dengan *port* 8080. Sedangkan untuk struktur data yang digunakan pada *database MySQL* terdiri dari *ID*, Tanggal input data, Nama *Spare parts*, dan Jumlah barang. Hasil dari tampilan website *local host* untuk sistem perhitungan sparepart secara otomatis (*machine vision*) dengan metode *Object Color Tracking* dan sistem penyimpanan data menggunakan database *MySQL* dapat dilihat pada **Gambar 4.25**



Gambar 4.25 Website untuk perhitungan Objek FOX-BABY

4.9 Hasil *upload* data menuju database MySQL menggunakan webserver

Data dapat diupload menuju database dengan ketentuan hitung terlebih dahulu barang dan masukan nama dari barang tersebut maka data akan terupload. Jika tidak maka akan muncul alert seperti **Gambar 4.26**

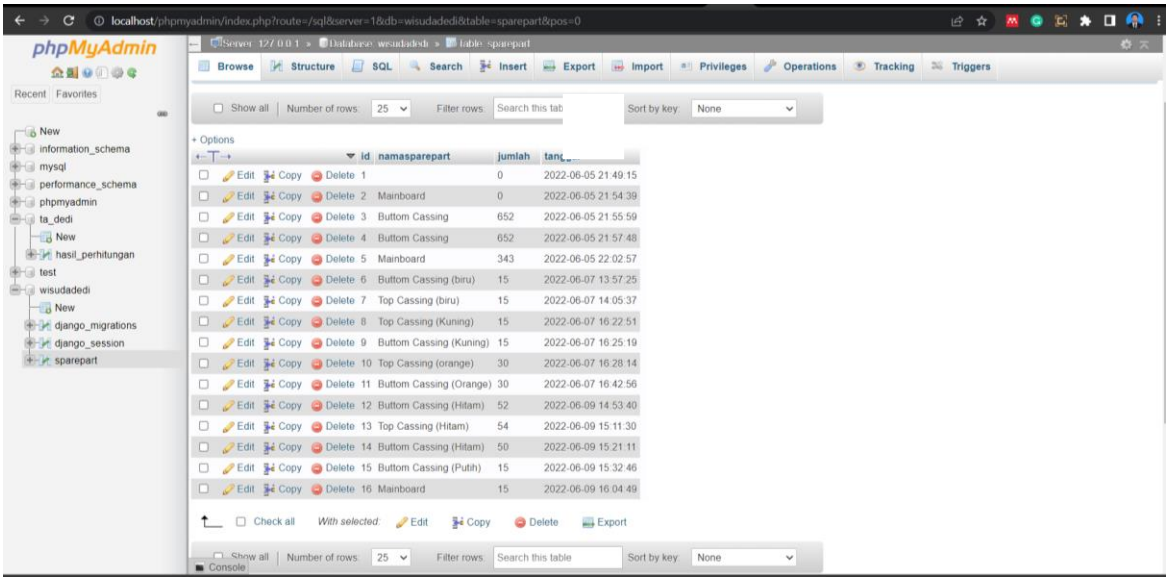
Gambar 4.26 *Alert* kesalahan upload data menuju database

Namun Ketika unit telah dipilih dan hasil perhitungan tidak sama dengan 0 seperti **Gambar 4.27** maka data akan berhasil terupload dengan alert seperti **Gambar 4.28**

Gambar 4.27 Ketentuan form agar data dapat terupload menuju *database*

Gambar 4.28 *Alert* sukses melakukan *upload* data

Gambar 4.29 adalah tampilan *database*. Kemudian data dapat diakses melalui halaman http://localhost/phpmyadmin/index.php?route=/sql&server=1&db=wisudadedi&table=sparepart&pos=0



Gambar 4.29 Tampilan *database MySQL*

4.10 Pembahasan Penelitian

Pembuatan rancang bangun penghitung *spare parts* unit FOX-BABY merek Elitech dengan metode *Object Color Tracking* menggunakan *Raspberry Pi 3 B* bertujuan untuk mengurangi proses perhitungan barang secara manual di PT. Sinko Prima Alloy. Dimana dari wawancara terhadap Supervisor Gudang banyak terjadi kesalahan perhitungan diakibatkan faktor *human error*. Sistem informasi yang belum memadai membuat pengiriman data cukup lama ke bagian *PPIC*. Akibatnya proses produksi mengalami penurunan kecepatan bahkan dapat tertunda. Pembuatan rancang bangun penghitung *spare*

parts unit FOX-BABY akan memanfaatkan konveyor *belt* yang telah tersedia di pabrik tersebut.

Konveyor *belt* yang digunakan memiliki spsesifikasi ukuran Panjang x Lebar sebesar 298 x 95 dengan sabuk (*belt*) berwarna hijau tua. Konveyor tersebut digerakkan meenggunakan motor electric dengan sumber tegangan AC 220 volt. Pada kondisi kecepatan minimum didapatkan nilai *RPM* konveyor *belt* sebesar 17,782 *rpm*. Pada kondisi kecepatan sedang didapatkan nilai *RPM* konveyor *belt* sebesar 52,780 *rpm*. Pada kondisi kecepatan penuh didapatkan nilai *RPM* konveyor *belt* sebesar 76,570 *rpm*. Konveyor tersebut memiliki kelebihan dapat diatur kemiringannya dengan memutar tuas pada bagian bawah konveyor. Konveyor milik PT. Sinko Prima Alloy berbahan *alluminium profile* yang cukup mudah untuk dilakukan proses modifikasi kerangka. Modifikasi kerangka yang digunakan adalah pemasangan kamera di atas *belt* konveyor, pemasangan skat penghalang barang tertumpuk, dan pemasangan *Raspberry Pi 3 B*. Kamera yang dipasangkan dia atas *belt* konveyor adalah kamera Nemesis dengan tipe NYK-96. Kamer ini memiliki spesifikasi resolusi 1920P 2K 18 MegaPixel dengan auto focus. Hasil yang didapatkan kamera dapat menangkap gambar dengan kualitas 2K dengan resolusi 1920 tanpa terputus-putus. Serta perhitungan kondisi lingkungan seperti sama dengan suhu 28.5^{o}C, nilai kelembaban adalah 72 % dan nilai intensitas cahaya sebesar 137,3 Lux. Nilai ini sudah dapat dikatakan sebagai tempat untuk melakukan proses perhitungan yang nyaman dan aman berdasarkan standar ISO 9001.

Pada proses pembuatan *software* penghitung barang menggunakan metode *object color tracking*, terdapat beberapa tahapan awal seperti instalasi *PIP python* untuk membuka dokumentasi python dan instalasi *library OpenCV*. Versi pip yang digunakan adalah versi 9.0.1 sedangkan versi *library OpenCV* yang digunakan adalah 4.1.2. Pada proses instalasi terdapat beberapa halangan seperti *virtual envirolment* yang belum dibuka dan *numpy* yang belum terinstall. Maka dilakukan instalasi *virtual envirolment* dan numpy pada *operation system Raspberry Pi 3 B*. Tahapan selanjutnya adalah pembuatan algoritma yang dapat memilah warna dengan menginputkan nilai minimum dan maksimum *HSV*. Nilai minimum dan maksimum dibutuhkan agar terdapat toleransi ketika barang yang lewat mendapat sedikit distraksi dari lingungan terutam faktor cahaya. Nilai minimum dan maksimum *HSV* dapat dilihat pada **Tabel 4.2**. Kemudian dilanjukan dengan pembuatan algoritma pemilahan tepi menggunakan metode *sobel edge detection*. Metode ini menghasilkan bentuk gambar sesuai dengan kontras warna putih dan warna hitam. Dimana akan dibuatkan titik-titik di

transisi warna putih dan warna hitam agar membentuk suatu objek sesuai dengan **Gambar 4.19.** Dilanjutkan dengan algoritma perhitungan yang mengacu pada diagram alir yang di jelaskan di **Gambar 3.7**.

Proses pengujian dilakukan untuk mencari nilai akurasi perhitungan dari setiap *spare parts* sesuai dengan tabel *Bill of Material (BOM)* di **Tabel 2.1.** Perhitungan dilakukan berdasarkan beberapa skema, yang pertama perhitungan objek tanpa distraksi dan tanpa distraksi warna. Distraksi yang digunakan adalah lebih dari 50% jumlah unit yang dihitung, dengan ketentuan jumlah perhitungan maksimal yaitu 20 pcs saat dilewatkan pada konveyor. Jumlah ini dipilih dikarenakan jumah *spare parts* di PT. Sinko Prima Alloy yang cukup terbatas. Selain itu minimnya jumlah sparepart yang dilewatkan dikarenakan pihak Gudang yang tidak berani mengambil resiko untuk melakukan percobaan pada unit berharga mahal. Dimana harga unit tersebut mencapai Rp. 3.267.000,00. Dari hasil percobaan didapatkan unit berwarna hitam, hijau muda dan putih memilki akurasi kurang dari 95%. Jenis perhitungan kedua yang dilakukan untuk mencari nilai akurasi dari barang ketika dilewatkan pada konveyor secara berulang-ulang sebanyak 10 kali dengan jumlah maksimum sparepart yaitu 20 pcs. Hasil yang didapatkan unit berwarna hitam, hijau muda dan putih memilki akurasi rata-rata kurang dari 95%. Percobaan jenis ketiga yang digunakan yaitu dengan melakukan perhitungan pada lima titik berbeda pada konveyor belt dengan melewatkan barang yang sama *(Battery Cover)* dengan warna yang sama (Kuning) sejumlah 1 pcs, 3 pcs, 5 pcs, 10 pcs dan 20 pcs. Letak titik dan hasil percobaan dapat dilihat pada BAB Lampiran di sub–BAB A dengan kesimpulan bahwan letak barang yang dilewatkan tidak mempengaruhi hasil dari perhitungan *spare parts*. Kemudian pengujian terakhir yang digunakan adalah melakukan pengujian terhadap *spare parts* sama namum berbeda warna dengan ketentuan, 2 warna berbeda, 3 warna berbeda dan 4 warna berbeda. Hasil dari perhitungan dapat dilihat di BAB Lampiran pada sub-BAB B. Dengan kesimpulan unit berwarna hitam dan putih memilki akurasi kurang dari 95%.

Sistem informasi yang digunakan dalam penelitian ini adalah website berbasis database MySQL. Proses integrasi sistem menggunakan framework Django Python dengan *server local (localhost).* Kemudian port yang digunakan pada alamat website adalah 8080. Interface website dengan bahasa pemrograman *HTML* digabungkan dengan *library bootstrap 4* agar mendapatkan *user interface* yang menarik dan mudah dipahami. Sedangkan untuk struktur data yang digunakan pada *database MySQL* terdiri dari *ID,* Tanggal input data, Nama *Spare parts*, dan Jumlah barang. Metode yang digunakan dalam proses penyaluran data adalah metode *MVC (Models, View, and Control)*.

Halaman ini sengaja dikosongkan

BAB V
KESIMPULAN DAN SARAN

A. Kesimpulan

Berdasarkan pembahasan dari penelitian ini, maka kesimpulan dari penelitian ini adalah:

a) Telah berhasil dilakukan rancang bangun penghitung sparepart unit FOX-BABY merek Elitech dengan metode *Object Color Tracking.* Pengujian perhitungan dilakukan sesuai jenis dan warna pada Bill of Material (BOM) dengan jumlah maksimal sparepart yaitu 20 pcs. Didapatkan hasil perhitungan sparepart warna biru, kuning dan orange memiliki nilai rata-rata akurasi 100% serta sparepart warna hitam, hijau muda dan putih memiliki nilai akurasi kurang dari 95%. Faktor yang mempengaruhi akurasi perhitungan adalah bayangan dan warna dasar dari sabuk konveyor.

b) Sistem informasi perhitungan *spare parts* yang terintegrasi berhasil dibuat menggunakan *website* berbasis *database MySQL* pada *Raspberry Pi 3 B. Framework* yang digunakan untuk mengintegrasikan *Back-end* dan *Front-end* adalah *Django Python*. Metode yang digunakan untuk menghubungkan *HTML page, Python page* dan database *MySQL* secara bersamaan adalah *MVC* (*Models, View, and Control) system.* Hasil yang didapatkan adalah data perhitungan berhasil dikirim menuju *database* dan dapat diakses dengan *sever local (localhost)*.

B. Saran

Berdasarkan pembahasan dari penelitian ini, maka kesimpulan dari penelitian ini adalah:

a) Perlu dilakukan penelitian lanjutan tentang penghalang noise warna hitam dan putih pada proses pendeteksian warna.

b) Perlu dilakukan pengembangan sistem pendeteksian warna karena saat barang yang dihitung berwarna hitam banyak noise yang didapatkan.

Halaman ini sengaja dikosongkan

DAFTAR PUSTAKA

Aosoby, R., Rusianto, T., & Waluyo, J. (2016). Perancangan Belt Conveyor sebagai Pengangkut Batubara dengan Kapasitas 2700 Ton/Jam. In *Jurnal Teknik Mesin* (Vol. 3, Issue 1).

Baygin, M., & Karakose, M. (2017). *An Image Processing based Object Counting Approach for Machine Vision Application*. https://www.researchgate.net/publication/319355836

Binti, I., Syawal, M., Bin, F., & Khairul, M. (n.d.). *Identification of Fruit Size and Maturity Through Fruit Images Using OpenCV-Python and Rasberry Pi*.

Chengrui Li, Dianguo Xu, & Gaolin Wang. (2017). High Efficiency Remanufacturing of Induction Motors With Interior Permanent-Magnet Rotors and Synchronous-Reluctance Rotors. *Transportation Electrification Conference and Expo, Asia-Pacific (ITEC Asia-Pacific)*.

Christian, F. (2017). *Modul pembelajaran raspberry pi*.

Django overview | Django. (n.d.). Retrieved June 15, 2022, from https://www.djangoproject.com/start/overview/

Elitech - PT. Sinko Prima Alloy. (n.d.). Retrieved December 28, 2021, from https://elitech.id/

Hestiningsih, & Idhawati. (2018). *Pengolahan Citra*.

John P Bentley. (2005). *Principles of Measurement Systems*. www.pearsoned.co.uk

Michalko, M., Onuška, J., Lavrín, A., Cymbalák, D., & Kainz, O. (n.d.). *Tracking the Object Features in Video Based on OpenCV*.

Nummiaro, K., Koller-Meier, E., & Gool, L. van. (n.d.). *Object Tracking with an Adaptive Color-Based Particle Filter*.

OpenCV: Changing Colorspaces. (n.d.). Retrieved June 15, 2022, from https://docs.opencv.org/4.x/df/d9d/tutorial_py_colorspaces.html

OpenCV: Image Thresholding. (n.d.). Retrieved June 12, 2022, from https://docs.opencv.org/4.x/d7/d4d/tutorial_py_thresholding.html

OpenCV: Smoothing Images. (n.d.). Retrieved June 15, 2022, from https://docs.opencv.org/4.x/d4/d13/tutorial_py_filtering.html

OpenCV: Sobel Derivatives. (n.d.). Retrieved June 12, 2022, from https://docs.opencv.org/3.4/d2/d2c/tutorial_sobel_derivatives.html

Otsu's Thresholding Technique | LearnOpenCV. (n.d.). Retrieved June 13, 2022, from https://learnopencv.com/otsu-thresholding-with-opencv/

pip install OpenCV - PyImageSearch. (n.d.). Retrieved July 16, 2022, from https://pyimagesearch.com/2018/09/19/pip-install-opencv/

Prabowo, D. A., Abdullah, D., & Manik, A. (2018a). DETEKSI DAN PERHITUNGAN OBJEK BERDASARKAN WARNA MENGGUNAKAN COLOR OBJECT TRACKING. In *Jurnal Pseudocode* (Issue 2). www.ejournal.unib.ac.id/index.php/pseudocode

Prabowo, D. A., Abdullah, D., & Manik, A. (2018b). DETEKSI DAN PERHITUNGAN OBJEK BERDASARKAN WARNA MENGGUNAKAN COLOR OBJECT TRACKING. In *Jurnal Pseudocode* (Issue 2). www.ejournal.unib.ac.id/index.php/pseudocode

Principles of User Interface Design: Important Rules that Every Designer Should Follow. (2015). *Radka Nacheva*. https://doi.org/10.13140/RG.2.1.5148.8083

Raspberry Pi. (2021, December 20). https://www.raspberrypi.org/

Sachdeva, A. (2017). Development Of Industrial Automatic Multi Colour Sorting and Counting Machine Using Arduino Nano Microcontroller and TCS3200 Colour Sensor. *The International Journal of Engineering and Science*, *06*(04), 56–59. https://doi.org/10.9790/1813-0604025659

Singh, P., B.B.V.L, D., Sethi, T., & Murthy, M. D. P. (2015). Real -Time Object Detection and Tracking Using Color Feature and Motion. *International Conference on Communication and Signal Processing*.

Theodosios Pavlidis. (1982). *Algorithms for Graphics and Image Processing*. Springer-Verlag Berlin Heidelberg.

Thorat, A. (2015). Counting Conveyor For Engineering Applications. *IPASJ International Journal of Mechanical Engineering*, *3*(5), 51–53. http://www.ipasj.org/IIJME/IIJME.htm

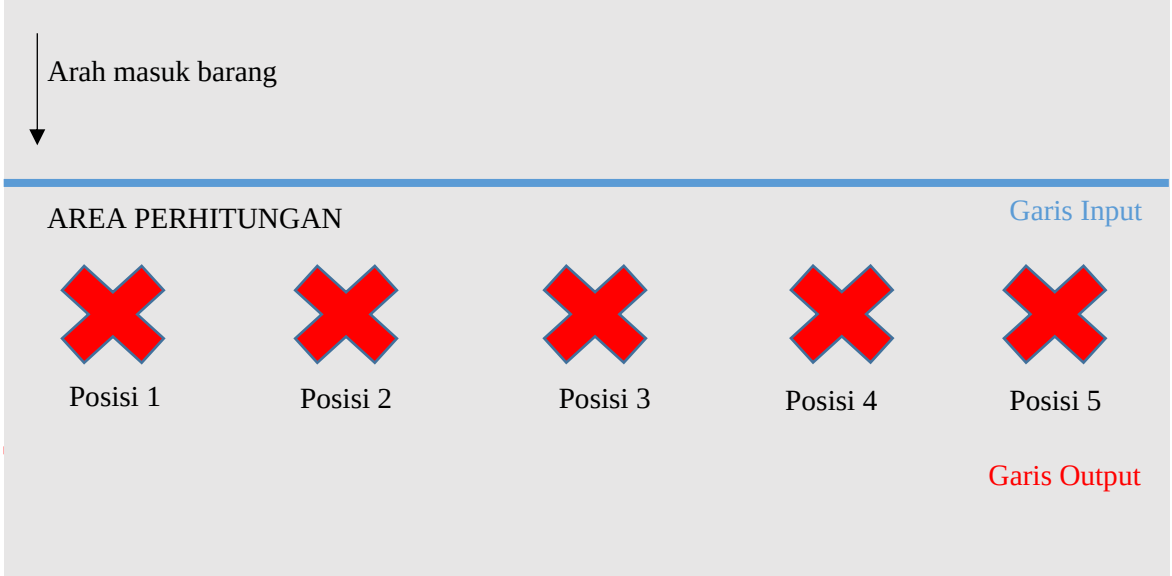
Why Choose Pro Cutter? - Pro-cutter. (n.d.). Retrieved June 15, 2022, from https://pro-cutter.com/choose-pro-cutter/

Xu, G., Qiao, Y., Wu, X., Institute of Electrical and Electronics Engineers, Institute of Electrical and Electronics Engineers. Beijing Section, & Zhongguo ke xue yuan. Shenzhen xian jin ji shu yan jiu yuan. (n.d.). *Proceedings of 2014 4th IEEE International Conference on Information Science and Technology : ICIST 2014 : April 26-28, 2014 Shenzhen, Guangdong, P.R. China*.

LAMPIRAN

A. Tabel perngujian penempatan barang secara acak

Dilakukan pengujian perhitungan pada lima titik berbeda pada konveyor *belt* dengan melewatkan barang yang sama *(Battery Cover)* dengan warna yang sama (Kuning) sejumlah 1 pcs, 3 pcs, 5 pcs, 10 pcs dan 20 pcs. perhatikan gambar ilustrasi posisi percobaan berikut berikut.



Data hasil percobaan akan ditulis pada tabel dan dicari nilai akutasinya. Berikut adalah hasil percobaan

Tabel 1 hasil pengujian perhitungan *battery cover* warna kuning dengan jumlah 1

No	Letak barang yang diuji	Jumlah perhitungan manual	Jumlah perhitungan otomatis	Nilai akurasi (%)
1	Posisi 1	1	1	100
2	Posisi 2	1	1	100
3	Posisi 3	1	1	100
4	Posisi 4	1	1	100
5	Posisi 5	1	1	100

Tabel 2 hasil pengujian perhitungan *battery cover* warna kuning dengan jumlah 3

No	Letak barang yang diuji	Jumlah perhitungan manual	Jumlah perhitungan otomatis	Nilai akurasi (%)
1	Posisi 1	3	3	100
2	Posisi 2	3	3	100
3	Posisi 3	3	3	100
4	Posisi 4	3	3	100
5	Posisi 5	3	3	100

Tabel 3 hasil pengujian perhitungan *battery cover* warna kuning dengan jumlah 5

No	Letak barang yang diuji	Jumlah perhitungan manual	Jumlah perhitungan otomatis	Nilai akurasi (%)
1	Posisi 1	5	5	100
2	Posisi 2	5	5	100
3	Posisi 3	5	5	100
4	Posisi 4	5	5	100
5	Posisi 5	5	5	100

Tabel 4 hasil pengujian perhitungan *battery cover* warna kuning dengan jumlah 10

No	Letak barang yang diuji	Jumlah perhitungan manual	Jumlah perhitungan otomatis	Nilai akurasi (%)
1	Posisi 1	10	10	100
2	Posisi 2	10	10	100
3	Posisi 3	10	10	100
4	Posisi 4	10	10	100
5	Posisi 5	10	10	100

Tabel 5 hasil pengujian perhitungan *battery cover* warna kuning dengan jumlah 1 pada posisi kelima

No	Letak barang yang diuji	Jumlah perhitungan manual	Jumlah perhitungan otomatis	Nilai akurasi (%)
1	Posisi 1	20	20	100
2	Posisi 2	20	20	100
3	Posisi 3	20	20	100
4	Posisi 4	20	20	100
5	Posisi 5	20	20	100

B. Tabel pengujian perhitungan berbeda warna.

1. Pengambilan data satu warna

   Jenis yang digunakan dalam pengambilan data ini adalah *battery casing* dengan jumlah maksimal sebanyak 10 pcs dengan ketentuan pengambilan data yaitu dengan melewatkan 1 pcs, 3 pcs, 5 pcs dan terakhir 10 pcs. Warna dasar yang digunakan pada percobaan ini adalah warna Biru, Hitam, Kuning dan Orange

   Tabel 6 *battery casing* warna biru

No.	Hasil perhitungan standar (Manual)	Hasil menggunakan mesin penghitung	Nilai Akurasi
1	1	1	100
2	3	3	100
3	5	5	100
4	10	10	100

   Tabel 7 *battery casing* warna hitam

No.	Hasil perhitungan standar (Manual)	Hasil menggunakan mesin penghitung	Nilai Akurasi
1	1	1	100
2	3	1	33.33333333
3	5	3	60
4	10	6	60

Tabel 8 *battery casing* warna Kuning

No.	Hasil perhitungan standar (Manual)	Hasil menggunakan mesin penghitung	Nilai Akurasi
1	1	1	100
2	3	3	100
3	5	5	100
4	10	10	100

Tabel 9 *battery casing* warna Orange

No.	Hasil perhitungan standar (Manual)	Hasil menggunakan mesin penghitung	Nilai Akurasi
1	1	1	100
2	3	3	100
3	5	5	100
4	10	10	100

2. Pengambilan data dua warna

Jenis yang digunakan dalam pengambilan data ini adalah *battery casing* dengan jumlah maksimal sebanyak 20 pcs dengan ketentuan pengambilan data yaitu dengan melewatkan 1 pcs, 3 pcs, 5 pcs dan terakhir 10 pcs dengan dua warna yang berbeda. Warna dasar yang digunakan pada percobaan ini adalah warna Biru, Hitam, Kuning dan Orange.

Tabel 10 *battery casing* warna biru dan warna hitam

No.	Hasil perhitungan standar (Manual)		Hasil menggunakan mesin penghitung		Nilai Akurasi	
	Biru	Hitam	Biru	Hitam	Biru	Hitam
1	1	1	1	0	100	0
2	3	3	3	2	100	66.66666667
3	5	5	5	2	100	40
4	10	10	10	7	100	70

Tabel 11 *battery casing* warna biru dan warna kuning

No.	Hasil perhitungan standar (Manual)		Hasil menggunakan mesin penghitung		Nilai Akurasi	
	Biru	Kuning	Biru	Kuning	Biru	Kuning
1	1	1	1	1	100	100
2	3	3	3	3	100	100
3	5	5	5	5	100	100
4	10	10	10	10	100	100

Tabel 12 *battery casing* warna biru dan warna orange

No.	Hasil perhitungan standar (Manual)		Hasil menggunakan mesin penghitung		Nilai Akurasi	
	Biru	Orange	Biru	Orange	Biru	Orange
1	1	1	1	1	100	100
2	3	3	3	3	100	100
3	5	5	5	5	100	100
4	10	10	10	10	100	100

Tabel 13 *battery casing* warna hitam dan warna kuning

No.	Hasil perhitungan standar (Manual)		Hasil menggunakan mesin penghitung		Nilai Akurasi	
	Hitam	Kuning	Hitam	Kuning	Hitam	Kuning
1	1	1	1	1	100	100
2	3	3	4	3	33.333	100
3	5	5	3	5	60	100
4	10	10	8	10	80	100

Tabel 14 *battery casing* warna hitam dan warna orange

No.	Hasil perhitungan standar (Manual)		Hasil menggunakan mesin penghitung		Nilai Akurasi	
	Hitam	Orange	Hitam	Orange	Hitam	Orange
1	1	1	1	1	100	100
2	3	3	1	3	33.3333	100
3	5	5	2	5	40	100
4	10	10	9	10	90	100

Tabel 15 *battery casing* warna kuning dan warna orange

No.	Hasil perhitungan standar (Manual)		Hasil menggunakan mesin penghitung		Nilai Akurasi	
	Kuning	Orange	Kuning	Orange	Kuning	Orange
1	1	1	1	1	100	100
2	3	3	3	3	100	100
3	5	5	5	5	100	100
4	10	10	10	10	100	100

3. Pengambilan data tiga warna

Jenis yang digunakan dalam pengambilan data ini adalah *battery casing* dengan jumlah maksimal sebanyak 30 pcs dengan ketentuan pengambilan data yaitu dengan melewatkan 1 pcs, 3 pcs, 5 pcs dan terakhir 10 pcs dengan tiga warna yang berbeda. Warna dasar yang digunakan pada percobaan ini adalah warna Biru, Hitam, Kuning dan Orange.

Tabel 16 *battery casing* warna biru, warna hitam dan warna kuning

No.	Hasil perhitungan standar (Manual)			Hasil menggunakan mesin penghitung			Nilai Akurasi		
	Biru	Hitam	Kuning	Biru	Hitam	Kuning	Biru	Hitam	Kuning
1	1	1	1	1	0	1	100	0	100
2	3	3	3	3	2	3	100	66.6667	100
3	5	5	5	5	3	5	100	60	100
4	10	10	10	10	4	10	100	40	100

Tabel 17 *battery casing* warna biru, warna hitam dan warna orange

No.	Hasil perhitungan standar (Manual)			Hasil menggunakan mesin penghitung			Nilai Akurasi		
	Biru	Hitam	Orange	Biru	Hitam	Orange	Biru	Hitam	Orange
1	1	1	1	1	0	1	100	0	100
2	3	3	3	3	1	3	100	33.3333	100
3	5	5	5	5	3	5	100	60	100
4	10	10	10	10	6	10	100	60	100

Tabel 18 *battery casing* warna biru, warna kuning dan warna orange

No.	Hasil perhitungan standar (Manual)			Hasil menggunakan mesin penghitung			Nilai Akurasi		
	Biru	Kuning	Orange	Biru	Kuning	Orange	Biru	Kuning	Orange
1	1	1	1	1	1	1	100	100	100
2	3	3	3	3	3	3	100	100	100
3	5	5	5	5	5	5	100	100	100
4	10	10	10	10	10	10	100	100	100

Tabel 19 *battery casing* warna hitam, warna kuning dan warna orange

No.	Hasil perhitungan standar (Manual)			Hasil menggunakan mesin penghitung			Nilai Akurasi		
	Hitam	Kuning	Orange	Hitam	Kuning	Orange	hitam	Kuning	Orange
1	1	1	1	1	1	1	100	100	100
2	3	3	3	3	3	3	100	100	100
3	5	5	5	2	5	5	40	100	100
4	10	10	10	7	10	10	70	100	100

4. Pengambilan data empat warna

Jenis yang digunakan dalam pengambilan data ini adalah *battery casing* dengan jumlah maksimal sebanyak 40 pcs dengan ketentuan pengambilan data yaitu dengan melewatkan 1 pcs, 3 pcs, 5 pcs dan terakhir 10 pcs dengan tiga warna yang berbeda. Warna dasar yang digunakan pada percobaan ini adalah warna Biru, Hitam, Kuning dan Orange.

Tabel 20 *battery casing* warna hitam, warna kuning dan warna orange

No.	Hasil perhitungan standar (Manual)				Hasil menggunakan mesin penghitung			
	Biru	Hitam	Kuning	Orange	Biru	Hitam	Kuning	Orange
1	1	1	1	1	1	1	1	1
2	3	3	3	3	3	3	3	3
3	5	5	5	5	5	2	5	5
4	10	10	10	10	10	7	10	10

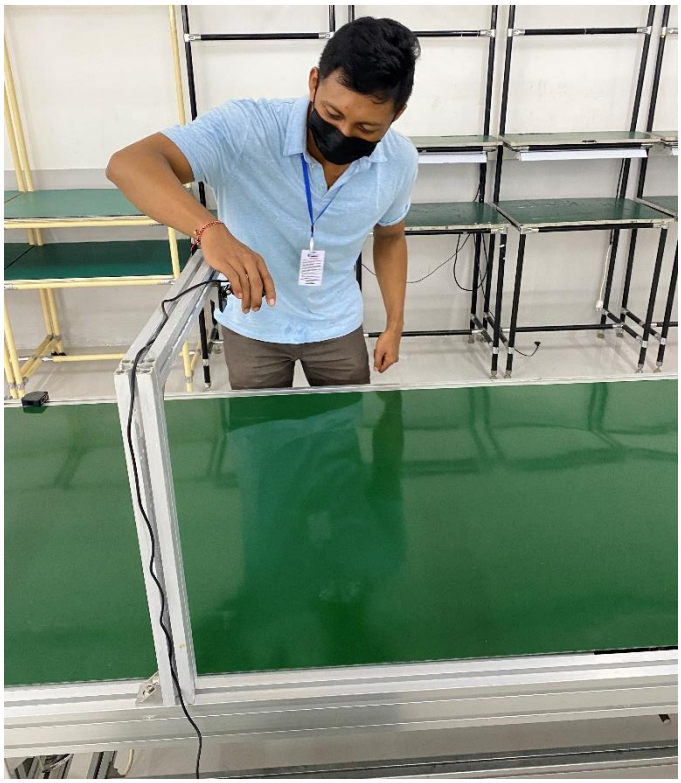
Nilai Akurasi (%)			
Biru	hitam	Kuning	Orange
100	100	100	100
100	100	100	100
100	40	100	100
100	70	100	100

C. Kontroler Penggerak Konveyor

D. Pemasangan Aluminium profile

E. Instalasi Raspberry

F. Pemasangan Kamera



G. Proses pemotongan aluminium profile

H. Kode python Django Modals untuk melakukan upload data menuju database

```
from django.db import models
# Create your models here.
class JumlahInsert(models.Model):
    class Meta:
        db_table = "sparepart"
    namasparepart=models.CharField(max_length=100)
    jumlah=models.CharField(max_length=100)
```

I. Kode python Django urls untuk kontroler page dan mengirimkan data dari page satu ke page lain

```
from django.urls import path
from . import views

urlpatterns = [
    path("", views.home, name="home"),
    path("vid", views.vid, name="vid"),
    path("hasil", views.admin, name="hasil"),
    path("uh/<int:value>", views.change_uh, name="uh"),
```

```
    path("us/<int:value>", views.change_us, name="us"),
    path("uv/<int:value>", views.change_uv, name="uv"),
    path("lh/<int:value>", views.change_lh, name="lh"),
    path("ls/<int:value>", views.change_ls, name="ls"),
    path("lv/<int:value>", views.change_lv, name="lv"),
    path("max_a/<int:value>", views.change_max_a, name="max_a"),
    path("min_a/<int:value>", views.change_min_a, name="min_a"),
    path("hitung/<int:value>", views.hitung, name="hitung"),
    path("insertdata", views.insertdata, name="insertdata"),
]
```

J. Kode python Django View sebagai algoritma aplikasi website

```
import re
from django.shortcuts import render
from django.http import HttpResponse
from .models import *
from django.core.mail import EmailMessage
from django.views.decorators import gzip
from django.http import StreamingHttpResponse
import cv2
import threading
import numpy as np
# from main.models import JumlahInsert
from django.contrib import messages

def change_uh(request, value):
    VideoCamera.uh = value
    return HttpResponse("")

def change_us(request, value):
    VideoCamera.us = value
    return HttpResponse("")

def change_uv(request, value):
    VideoCamera.uv = value
    return HttpResponse("")
```

```
def change_lh(request, value):
    VideoCamera.lh = value
    return HttpResponse("")

def change_ls(request, value):
    VideoCamera.ls = value
    return HttpResponse("")

def change_lv(request, value):
    VideoCamera.lv = value
    return HttpResponse("")

def change_max_a(request, value):
    VideoCamera.Up_CA = value
    return HttpResponse("")

def change_min_a(request, value):
    VideoCamera.low_CA = value
    return HttpResponse("")

def hitung(request, value):
    if value == 1:
        VideoCamera.start = True
        return HttpResponse(0)
    else:
        VideoCamera.start = False
        hasil = VideoCamera.count_total
        VideoCamera.count_total = 0
        return HttpResponse(hasil)
    return HttpResponse("")

def home(response):
    return render(response, "main/home.html", {})

def admin(response):
    return render(response, "main/hasil.html", {})
```

```
@gzip.gzip_page
def vid(request):
    try:
        cam = VideoCamera()
        return StreamingHttpResponse(gen(cam), content_type="multipart/x-mixed-
replace;boundary=frame")
    except:
        pass
    return render(request, 'main/home.html')

#to capture video class
class VideoCamera(object):
    lh = 0
    ls = 0
    lv = 0
    us = 255
    uv = 255
    uh = 255
    Up_CA = 100000
    low_CA = 0
    start = False
    flag = False
    count_total = 0
    max_count = 0

    def __init__(self):
        self.video = cv2.VideoCapture(0)
        self.video.set(3, 1280)
        self.video.set(4, 720)
        (self.grabbed, self.frame) = self.video.read()
        threading.Thread(target=self.update, args=()).start()

    def __del__(self):
        self.video.release()

    def get_frame(self):
```

```
        image = cv2.cvtColor(self.frame, cv2.COLOR_BGR2HSV)
        gaussian_blur = cv2.GaussianBlur(image, (3, 3), 0)

        l_b = np.array([VideoCamera.lh, VideoCamera.ls, VideoCamera.lv])
        u_b = np.array([VideoCamera.uh, VideoCamera.us, VideoCamera.uv])

        # Seleksi warna
        mask = cv2.inRange(gaussian_blur, l_b, u_b)
        warna_seleksi = cv2.bitwise_and(self.frame, self.frame, mask=mask)

        # membuat garis peerhitungan
        width = self.frame.shape[1]
        height = self.frame.shape[0]

        CoorYEntranceLine = int((height / 2)-150)
        CoorYExitLine = int((height / 2)+150)
        cv2.line(warna_seleksi, (0,CoorYEntranceLine), (width,CoorYEntranceLine), (255, 0,
0), 2)
        cv2.line(warna_seleksi, (0,CoorYExitLine), (width,CoorYExitLine), (0, 0, 255), 2)

        # contour gambar
        contours, hierarchy = cv2.findContours(mask, cv2.RETR_TREE,
cv2.CHAIN_APPROX_NONE)

         # countur rectangle

        count = 0
        for i in range(len(contours)):
            x, y, w, h = cv2.boundingRect(contours[i])
            if VideoCamera.Up_CA > cv2.contourArea(contours[i]) > VideoCamera.low_CA
and y > CoorYEntranceLine and y < CoorYExitLine:
                # print("area", i, ":", cv2.contourArea(contours[i]))
                cv2.rectangle(warna_seleksi, (x, y), (x + w, y + h), (0, 255, 0), 2)
                # print("area", cv2.contourArea(contours[i]))
                count += 1
                # print(count)
```

```
        if count > VideoCamera.max_count:
            VideoCamera.max_count = count
            VideoCamera.flag = True

        if VideoCamera.start and VideoCamera.flag and count == 0:
            VideoCamera.count_total += VideoCamera.max_count
            VideoCamera.max_count = 0
            VideoCamera.flag = False

        #Memasukan tulisan pada gambar
        # font
        font = cv2.FONT_HERSHEY_SIMPLEX

        # org
        org = (170, 50)
        org2 = (20, 50)

        # fontScale
        fontScale = 1

        # Blue color in BGR
        color = (255, 0, 0)

        # Line thickness of 2 px
        thickness = 2

        # Using cv2.putText() method
        # text = cv2.putText(img, , org, font, fontScale, color, thickness, cv2.LINE_AA)
        text = cv2.putText(warna_seleksi, "Total : " + str(VideoCamera.count_total), (20, 50),
cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 0, 0), 2, cv2.LINE_AA)

        # image = cv2.putText(image, 'OpenCV %d' % VideoCamera.uh, (50, 50),
cv2.FONT_HERSHEY_SIMPLEX,
        #           1, (255, 0, 0), 2, cv2.LINE_AA)

        _, jpeg = cv2.imencode('.jpg', warna_seleksi)
```

```
            return jpeg.tobytes()

        def update(self):
            while True:
                (self.grabbed, self.frame) = self.video.read()

    def gen(camera):
        while True:
            frame = camera.get_frame()
            yield (b'--frame\r\n'
                   b'Content-Type: image/jpeg\r\n\r\n' + frame + b'\r\n\r\n')



    def insertdata(request):
        saverecord=JumlahInsert()
        saverecord.namasparepart=request.POST.get('nama')
        saverecord.jumlah=request.POST.get('jumlah')
        saverecord.save()
        return HttpResponse("")

    # def Jumlah(request):
    #     if request=='POST':
    #         if request.POST.get('namasparepart') and request.POST.get('jumlah'):
    #             saverecord=JumlahInsert()
    #             saverecord.namasparepart=request.POST.get('namasparepart')
    #             saverecord.jumlah=request.POST.get('jumlah')
    #             saverecord.save()
    #             messages.success(request, 'Record saved Successfully...!')
    #             return render(request, 'home.html')
    #         else:
    #             return render(request, 'home.html')
```

K. Kode python Django Setting untuk melakukan pengaturan database dan target first page

```
    """
    Django settings for opencv project.

    Generated by 'django-admin startproject' using Django 4.0.4.
```

```
For more information on this file, see
https://docs.djangoproject.com/en/4.0/topics/settings/

For the full list of settings and their values, see
https://docs.djangoproject.com/en/4.0/ref/settings/
"""

from pathlib import Path

# Build paths inside the project like this: BASE_DIR / 'subdir'.
BASE_DIR = Path(__file__).resolve().parent.parent



# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/4.0/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = 'django-insecure-qu%f$3w++pf#2de!-&f(w7vt5$4m+j17&$pz-
a+bjrk5=obem3'

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

ALLOWED_HOSTS = []



# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'main.apps.MainConfig',
```

```
]

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

ROOT_URLCONF = 'opencv.urls'

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]

WSGI_APPLICATION = 'opencv.wsgi.application'


# Database
# https://docs.djangoproject.com/en/4.0/ref/settings/#databases
```

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'wisudadedi',
        'USER': 'root',
        'PASSWORD' : '',
        'HOST' : 'localhost',
        'PORT' : '3306',
    }
}


# Password validation
# https://docs.djangoproject.com/en/4.0/ref/settings/#auth-password-validators

AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]


# Internationalization
# https://docs.djangoproject.com/en/4.0/topics/i18n/

LANGUAGE_CODE = 'en-us'
```

```
TIME_ZONE = 'UTC'

USE_I18N = True

USE_TZ = True



# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/4.0/howto/static-files/

STATIC_URL = 'static/'
STATICFILES_DIRS = [
   BASE_DIR / "static",

]

# Default primary key field type
# https://docs.djangoproject.com/en/4.0/ref/settings/#default-auto-field

DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'
```

L. Kode HTML home sebagai interface website

```
<!doctype html>
<html lang="en">
  <head>
    <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">

    <!-- Bootstrap CSS -->
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css"
rel="stylesheet" integrity="sha384-
1BmE4kWBq78iYhFldvKuhfTAU6auU8tT94WrHftjDbrCEXSU1oBoqyl2QvZ6jIW3"
crossorigin="anonymous">

    <title>Tugas Akhir</title>
```

```
    {% load static %}
    <link rel="stylesheet" href="{% static 'css/home.css' %}">
  </head>
    {% comment %} <style>
    body {background-color: white;}
    h1   {color: black;}
    </style> {% endcomment %}
  <body>
    <nav class=" navbar navbar-expand-lg navbar-dark bg-primary">
      <div class="container-fluid">
        <a class="navbar-brand" href="#">TUGAS AKHIR</a>
        <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-
target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-
expanded="false" aria-label="Toggle navigation">
          <span class="navbar-toggler-icon"></span>
        </button>
        <div class="collapse navbar-collapse" id="navbarSupportedContent">
          <ul class="navbar-nav me-auto mb-2 mb-lg-0">
            <li class="nav-item">
              <a class="nav-link active" aria-current="page" href="{% url 'home' %}">Counter
Page</a>
            </li>
            <li class="nav-item">
              <a class="nav-link active" aria-current="page" href="{% url 'hasil' %}">Admin
Page</a>
            </li>
          </ul>
        </div>
          <img src="{% static 'img/tfisika7.png' %}" width="3%" alt="My image"
style="margin-right: 60px">
      </div>
    </nav>
    <!-- body -->
    <div style="margin-top: 1%;" class="container">
      <div class="row">
        <div class="col-9">
          <div class="container">
```

```
            <img src="{% url 'vid' %}" width="100%" alt="My image"><br>
            <div style="margin-top: 1%;" class="container">
              <div class="container">
                <div class="row align-items-start">
                  <div class="col-6">
                    <button type="button" class="btn-hitung btn btn-success btn-lg"
onClick="hitung(1)">HITUNG</button>
                  </div>
                  <div class="col-6">
                    <button type="button" class="btn-hitung btn btn-danger btn-lg"
onClick="hitung(0)">STOP</button>
                  </div>
                </div>
              </div>
              <form>
                <div class="form-group">
                  <label class="form-label">Maximal Area = <b
id="label_max">50000</b></label>
                  <input type="range" class="form-range" min="0" max="50000"
id="range_max" value="50000"
                    onChange="max_a()">
                </div>
                <div class="form-group">
                  <label class="form-label">Minimal Area = <b
id="label_min">50000</b></label>
                  <input type="range" class="form-range" min="0" max="50000"
id="range_min" value="0"
                    onChange="min_a()">
                </div>
              </form>
            </div>
          </div>
        </div>
        <div class="col-3">
          <form>
            {% comment %} <div class="form-group">
```

```
                <label for="formGroupExampleInput">Upper Hue</label>
                <div class="kotak">
                  <div class="der">
                    <input type="range" class="jarak" min="0" max="255">
                    <div class="nilai">100</div>
                  </div>
                </div>
              </div> {% endcomment %}
              <div class="form-group">
                <label class="form-label">Upper Hue = <b id="label_uh">255</b></label>
                <input type="range" class="form-range" min="0" max="255" id="range_uh"
value="255"
                onChange="uh()">
              </div>
              <div class="form-group">
                <label for="customRange2" class="form-label">Upper Saturation = <b
id="label_us">255</b></label>
                <input type="range" class="form-range" min="0" max="255" id="range_us"
value="255"
                onChange="us()">
              </div>
              <div class="form-group">
                <label for="customRange2" class="form-label">Upper Value = <b
id="label_uv">255</b></label>
                <input type="range" class="form-range" min="0" max="255" id="range_uv"
value="255"
                onChange="uv()">
              </div>
              <div class="form-group">
                <label for="customRange2" class="form-label">Lower Hue = <b
id="label_lh">0</b></label>
                <input type="range" class="form-range" min="0" max="255" id="range_lh"
value="0"
                onChange="lh()">
              </div>
              <div class="form-group">
```

```
                <label for="customRange2" class="form-label">Lower Saturation = <b
id="label_ls">0</b></label>
                <input type="range" class="form-range" min="0" max="255" id="range_ls"
value="0"
                onChange="ls()">
              </div>
              <div class="form-group">
                <label for="customRange2" class="form-label">Lower Value = <b
id="label_lv">0</b></label>
                <input type="range" class="form-range" min="0" max="255" id="range_lv"
value="0"
                onChange="lv()">
              </div>
            </form>
            <div class="dropdown">
              <button class="btn btn btn-secondary dropdown-toggle" type="button"
id="dropdownMenuButton1" data-bs-toggle="dropdown" aria-expanded="false">
                List Sparepart
              </button>
              <ul class="dropdown-menu" id="demolist" aria-
labelledby="dropdownMenuButton1">
                <li><a class="dropdown-item">Top Cassing</a></li>
                <li><a class="dropdown-item">Mainboard</a></li>
                <li><a class="dropdown-item">Buttom Cassing</a></li>
                <li><a class="dropdown-item">Battry Cover</a></li>
              </ul>
            </div><br>
            {% comment %} <form method='POST'>
              {% csrf_token %}
              <input type="TextBox" placeholder="Pilih atau Ketik" id="part" value=""
name="namasparepart">
              <input type="hidden" id="hasil_tot" value="0" name="jumlah">
              <input type="submit" value="Save"><br>
              {% if messages %}
              {% for messages in messages %}
              <h2 style="color:green;">{{messages}}</h2>
```

```
            {% endfor %}
            {% endif %}
          </form> {% endcomment %}
          <form>
            <input type="TextBox" placeholder="Pilih atau Ketik" id="nama_sparepart"
name="namasparepart">
            {% comment %} <input type="hidden" id="hasil_tot" value="0" name="jumlah">
{% endcomment %}
            <input type="button" onClick="insertdata(1)" value="save"><br>
          </form>

          <div style="margin-top: 4%;" class="card">
            <h5 class="card-header">Hasil Perhitungan</h5>
            <div class="card-body">
              <h5 id="hasil_total" class="card-title">0</h5>
            </div>
          </div>

      </div>
      <div style="margin-top: 1%;" class="row">
        <div class="col-9">
      {% comment %} pindah dulu {% endcomment %}
        </div>
        <div class="col-3">
            {% comment %}  {% endcomment %}
          </div>
        </div>
      </div>
    </div>

    <!-- Optional JavaScript; choose one of the two! -->

    <script src="https://code.jquery.com/jquery-3.6.0.min.js" integrity="sha256-
/xUj+3OJU5yExlq6GSYGSHk7tPXikynS7ogEvDej/m4="
crossorigin="anonymous"></script>
    <script>
      function uh(){
```

```
  let value = document.getElementById('range_uh').value;
  document.getElementById('label_uh').innerText = value;
  $.get(`http://127.0.0.1:8000/uh/${value}`)
}
function us(){
  let value = document.getElementById('range_us').value;
  document.getElementById('label_us').innerText = value;
  $.get(`http://127.0.0.1:8000/us/${value}`)
}
function uv(){
  let value = document.getElementById('range_uv').value;
  document.getElementById('label_uv').innerText = value;
  $.get(`http://127.0.0.1:8000/uv/${value}`)
}
function lh(){
  let value = document.getElementById('range_lh').value;
  document.getElementById('label_lh').innerText = value;
  $.get(`http://127.0.0.1:8000/lh/${value}`)
}
function ls(){
  let value = document.getElementById('range_ls').value;
  document.getElementById('label_ls').innerText = value;
  $.get(`http://127.0.0.1:8000/ls/${value}`)
}
function lv(){
  let value = document.getElementById('range_lv').value;
  document.getElementById('label_lv').innerText = value;
  $.get(`http://127.0.0.1:8000/lv/${value}`)
}
function max_a(){
  let value = document.getElementById('range_max').value;
  document.getElementById('label_max').innerText = value;
  $.get(`http://127.0.0.1:8000/max_a/${value}`)
}
function min_a(){
  let value = document.getElementById('range_min').value;
```

```
  document.getElementById('label_min').innerText = value;
  $.get(`http://127.0.0.1:8000/min_a/${value}`)
}
function hitung(value){
  $.get(`http://127.0.0.1:8000/hitung/${value}`, (data)=>{
    document.getElementById('hasil_total').innerText = data
  })
}
function insertdata(value){
  let jumlah = parseInt(document.getElementById('hasil_total').innerText);
  let nama = document.getElementById('nama_sparepart').value;
  if (jumlah === 0){
    alert("Hitung terlebih dahulu!");
  } else {
    $.ajax({
      type: "POST",
      url: "/insertdata",
      data: { csrfmiddlewaretoken: "{{ csrf_token }}",   // < here
            state:"inactive",
            jumlah: jumlah,
            nama: nama,
           },
      success: function() {
         alert("data updated");
         document.getElementById('hasil_total').innerText = 0;
      }
    })
  }

  //$.post(`${window.location.href}insertdata`)
  // var hasil_total = parseInt(document.getElementById('hasil_total').innerText);
  // if (hasil_total === 0){
  //   alert("Hitung terlebih dahulu!");
  // } else {
  //   $.get(`http://127.0.0.1:8000/insertdata/${value}`);
  //   let nama_sparepart = document.getElementById('nama_sparepart').value;
  //   console.log(nama_sparepart);
```

xc

```
        // }
      }
    </script>

    <script>
      $('#demolist li').on('click', function(){
        $('#nama_sparepart').val($(this).text());
      });
    </script>


    <!-- Option 1: Bootstrap Bundle with Popper -->
    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js"
integrity="sha384-
ka7Sk0Gln4gmtz2MlQnikT1wXgYsOg+OMhuP+IlRH9sENBO0LRn5q+8nbTov4+1p"
crossorigin="anonymous"></script>

    <!-- Option 2: Separate Popper and Bootstrap JS -->
    <!--
    <script
src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.10.2/dist/umd/popper.min.js"
integrity="sha384-
7+zCNj/IqJ95wo16oMtfsKbZ9ccEh31eOz1HGyDuCQ6wgnyJNSYdrPa03rtR1zdB"
crossorigin="anonymous"></script>
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.min.js"
integrity="sha384-
QJHtvGhmr9XOIpI6YVutG+2QOK9T+ZnN4kzFN1RtK3zEFEIsxhlmWl5/YESvpZ13"
crossorigin="anonymous"></script>
    -->
  </body>
</html>
```

M. Kode CSS sebagai pengolahan interface agar lebih interaktif

```
body{
    background-color: white;
}
```

```
h1{
    color: black;
}

.tf {
    align-items: right;
}

/* slide range */
.kotak {
    background: white;
    padding: 20px 20px 20px 30px;
    border-radius: 5px;
    box-shadow: 0px 0px 5px rgba(0,0,0,0.1);
    display: flex;
    align-items: center;
}

.jarak {
    width: fit-content;
}

.card-title {
    text-align: center;
}

.btn-hitung {
    width: 100%;
}
```

N. Kode python untuk perhitungan mendapatkan 4 warna berbeda dalam satu perhitungan

```
import cv2
import numpy as np

def nothing(x):
    pass
```

```
    upper = 1000
    lower = 700
    cv2.namedWindow("Tracking")
    cv2.resizeWindow("Tracking", 400, 400)
    cv2.createTrackbar("Upper", "Tracking", upper, 20000, nothing)
    cv2.createTrackbar("Lowwer", "Tracking", lower, 20000, nothing)


    camera = cv2.VideoCapture('./1.mp4')
    start = False
    flag = False
    flag1 = False
    flag2 = False
    flag3 = False
    #########---BIRU----######
    count_total = 0
    max_count = 0
    #########----ORANGE----#######
    count_total1 = 0
    max_count1 = 0
    #########----KUNING----#######
    count_total2 = 0
    max_count2 = 0
    #########----HITAM----#######
    count_total3 = 0
    max_count3 = 0


    while True:

        _, src = camera.read()
        # img = cv2.imread('1.jpg')
        src = cv2.resize(src, (640,360))

        # mengubah warna dari RGB ke HSV
        hsv = cv2.cvtColor(src, cv2.COLOR_BGR2HSV)
```

```
    # Gausian filter
    gaussian_blur = cv2.GaussianBlur(hsv, (3, 3), 0)

    # membuat nilai tracker untuk contour
    Up_CA = cv2.getTrackbarPos("Upper", "Tracking")
    low_CA = cv2.getTrackbarPos("Lowwer", "Tracking")



##########################################################--------BIRU--------
##########################################################################################
########
    # set brightness
    l_b = np.array([97, 194, 99])
    u_b = np.array([255, 255, 255])

    # Seleksi warna
    mask = cv2.inRange(gaussian_blur, l_b, u_b)
    warna_seleksi = cv2.bitwise_and(src, src, mask=mask)

    # membuat garis peerhitungan
    width = src.shape[1]
    height = src.shape[0]
    CoorYEntranceLine = int((height / 2)-100)
    CoorYExitLine = int((height / 2)+100)
    cv2.line(warna_seleksi, (0,CoorYEntranceLine), (width,CoorYEntranceLine), (255, 0, 0), 2)
    cv2.line(warna_seleksi, (0,CoorYExitLine), (width,CoorYExitLine), (0, 0, 255), 2)

    # contour gambar
    contours, hierarchy = cv2.findContours(mask, cv2.RETR_TREE,
cv2.CHAIN_APPROX_NONE)

    # countur rectangle
    count = 0
    for i in range(len(contours)):
        x, y, w, h = cv2.boundingRect(contours[i])
```

xciv

```
        if Up_CA > cv2.contourArea(contours[i]) > low_CA and y > CoorYEntranceLine and y <
    CoorYExitLine:
            # print("area", i, ":", cv2.contourArea(contours[i]))
            cv2.rectangle(warna_seleksi, (x, y), (x + w, y + h), (255, 0, 0), 2)
            # print("area", cv2.contourArea(contours[i]))
            count += 1
            # print(count)

    if count > max_count:
        max_count = count
        flag = True

    if start and flag and count == 0:
        count_total += max_count
        max_count = 0
        flag = False



    # mencari deteksi tepi
    scale = 1
    delta = 0
    ddepth = cv2.CV_16S
    grad_x = cv2.Sobel(mask, ddepth, 1, 0, ksize=3, scale=scale, delta=delta,
    borderType=cv2.BORDER_DEFAULT)
    # Gradient-Y
    # grad_y = cv.Scharr(gray,ddepth,0,1)
    grad_y = cv2.Sobel(mask, ddepth, 0, 1, ksize=3, scale=scale, delta=delta,
    borderType=cv2.BORDER_DEFAULT)
    abs_grad_x = cv2.convertScaleAbs(grad_x)
    abs_grad_y = cv2.convertScaleAbs(grad_y)
    grad = cv2.addWeighted(abs_grad_x, 0.5, abs_grad_y, 0.5, 0)
    ###############################################################################
    ###################################################################
    ####################################----------ORANGE----------
    ###########################################################################
    # set brightness
```

```
    l_b1 = np.array([0, 0, 201])
    u_b1 = np.array([19, 255, 255])

    # Seleksi warna
    mask1 = cv2.inRange(gaussian_blur, l_b1, u_b1)
    warna_seleksi1 = cv2.bitwise_and(src, src, mask=mask1)

    # contour gambar
    contours1, hierarchy = cv2.findContours(mask1, cv2.RETR_TREE,
cv2.CHAIN_APPROX_NONE)

    # countur rectangle
    count1 = 0
    for i in range(len(contours1)):
        x, y, w, h = cv2.boundingRect(contours1[i])
        if Up_CA > cv2.contourArea(contours1[i]) > low_CA and y > CoorYEntranceLine and y
< CoorYExitLine:
            # print("area", i, ":", cv2.contourArea(contours[i]))
            cv2.rectangle(warna_seleksi1, (x, y), (x + w, y + h), (0, 127, 255), 2)
            # print("area", cv2.contourArea(contours[i]))
            count1 += 1
            # print(count)

    if count1 > max_count1:
        max_count1 = count1
        flag1 = True

    if start and flag1 and count1 == 0:
        count_total1 += max_count1
        max_count1 = 0
        flag1 = False



    # mencari deteksi tepi
    grad_x1 = cv2.Sobel(mask1, ddepth, 1, 0, ksize=3, scale=scale, delta=delta,
borderType=cv2.BORDER_DEFAULT)
    # Gradient-Y
```

xcvi

```
    # grad_y = cv.Scharr(gray,ddepth,0,1)
    grad_y1 = cv2.Sobel(mask1, ddepth, 0, 1, ksize=3, scale=scale, delta=delta,
borderType=cv2.BORDER_DEFAULT)
    abs_grad_x1 = cv2.convertScaleAbs(grad_x1)
    abs_grad_y1 = cv2.convertScaleAbs(grad_y1)
    grad1 = cv2.addWeighted(abs_grad_x1, 0.5, abs_grad_y1, 0.5, 0)

################################################################################
#####################################################################
    ##############################----------KUNING----------
################################################################################
    # set brightness
    l_b2 = np.array([24, 0, 226])
    u_b2 = np.array([45, 255, 255])

    # Seleksi warna
    mask2 = cv2.inRange(gaussian_blur, l_b2, u_b2)
    warna_seleksi2 = cv2.bitwise_and(src, src, mask=mask2)

    # contour gambar
    contours2, hierarchy = cv2.findContours(mask2, cv2.RETR_TREE,
cv2.CHAIN_APPROX_NONE)

    # countur rectangle
    count2 = 0
    for i in range(len(contours2)):
        x, y, w, h = cv2.boundingRect(contours2[i])
        if Up_CA > cv2.contourArea(contours2[i]) > low_CA and y > CoorYEntranceLine and y
< CoorYExitLine:
            # print("area", i, ":", cv2.contourArea(contours[i]))
            cv2.rectangle(warna_seleksi2, (x, y), (x + w, y + h), (0, 255, 255), 2)
            # print("area", cv2.contourArea(contours[i]))
            count2 += 1
            # print(count)

    if count2 > max_count2:
```

```
        max_count2 = count2
        flag2 = True

    if start and flag2 and count2 == 0:
        count_total2 += max_count2
        max_count2 = 0
        flag2 = False



    # mencari deteksi tepi
    grad_x2 = cv2.Sobel(mask2, ddepth, 1, 0, ksize=3, scale=scale, delta=delta,
borderType=cv2.BORDER_DEFAULT)
    # Gradient-Y
    # grad_y = cv.Scharr(gray,ddepth,0,1)
    grad_y2 = cv2.Sobel(mask2, ddepth, 0, 1, ksize=3, scale=scale, delta=delta,
borderType=cv2.BORDER_DEFAULT)
    abs_grad_x2 = cv2.convertScaleAbs(grad_x2)
    abs_grad_y2 = cv2.convertScaleAbs(grad_y2)
    grad2 = cv2.addWeighted(abs_grad_x2, 0.5, abs_grad_y2, 0.5, 0)
################################################################################
################################################################################
##
##############################----------HITAM----------
################################################################################
    # set brightness
    l_b3 = np.array([94, 0, 0])
    u_b3 = np.array([255, 125, 255])

    # Seleksi warna
    mask3 = cv2.inRange(gaussian_blur, l_b3, u_b3)
    warna_seleksi3 = cv2.bitwise_and(src, src, mask=mask3)

    # contour gambar
    contours3, hierarchy = cv2.findContours(mask3, cv2.RETR_TREE,
cv2.CHAIN_APPROX_NONE)

    # countur rectangle
```

```
    count3 = 0
    for i in range(len(contours3)):
        x, y, w, h = cv2.boundingRect(contours3[i])
        if Up_CA > cv2.contourArea(contours3[i]) > low_CA and y > CoorYEntranceLine and y
< CoorYExitLine:
            # print("area", i, ":", cv2.contourArea(contours[i]))
            cv2.rectangle(warna_seleksi3, (x, y), (x + w, y + h), (255, 255, 255), 2)
            # print("area", cv2.contourArea(contours[i]))
            count3 += 1
            # print(count)

    if count3 > max_count3:
        max_count3 = count3
        flag3 = True

    if start and flag3 and count3 == 0:
        count_total3 += max_count3
        max_count3 = 0
        flag3 = False



    # mencari deteksi tepi
    grad_x3 = cv2.Sobel(mask2, ddepth, 1, 0, ksize=3, scale=scale, delta=delta,
borderType=cv2.BORDER_DEFAULT)
    # Gradient-Y
    # grad_y = cv.Scharr(gray,ddepth,0,1)
    grad_y3 = cv2.Sobel(mask2, ddepth, 0, 1, ksize=3, scale=scale, delta=delta,
borderType=cv2.BORDER_DEFAULT)
    abs_grad_x3 = cv2.convertScaleAbs(grad_x3)
    abs_grad_y3 = cv2.convertScaleAbs(grad_y3)
    grad3 = cv2.addWeighted(abs_grad_x3, 0.5, abs_grad_y3, 0.5, 0)
##############################################################################
##############################################################################
##
##############################-------GABUNGAN WARNA, MASK DAN EDGE---------
---############################################################
```

```
    # warna
    warna_gab1 = cv2.add(warna_seleksi, warna_seleksi1)
    warna_gab2 = cv2.add(warna_seleksi2, warna_seleksi3)
    warna_hasil = cv2.add(warna_gab1, warna_gab2)
    # mask
    mask_gab1 = cv2.add(mask, mask1)
    mask_gab2 = cv2.add(mask2, mask3)
    mask_hasil = cv2.add(mask_gab1, mask_gab2)
    # Deteksi tepi
    grad_gab1 = cv2.add(grad, grad1)
    grad_gab2 = cv2.add(grad2, grad3)
    grad_hasil = cv2.add(grad_gab1, grad_gab2)

    #Memasukan tulisan pada gambar
    # font
    font = cv2.FONT_HERSHEY_SIMPLEX
    # fontScale
    fontScale = 1
    # Blue color in BGR
    color = (255, 0, 0)
    # Line thickness of 2 px
    thickness = 2
    # Using cv2.putText() method
    # text = cv2.putText(img, , org, font, fontScale, color, thickness, cv2.LINE_AA)
    text = cv2.putText(warna_hasil, "Biru : " + str(count_total), (10, 50),
cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 0, 0), 2, cv2.LINE_AA)
    text = cv2.putText(warna_hasil, "orange : " + str(count_total1), (170, 50),
cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 127, 255), 2, cv2.LINE_AA)
    text = cv2.putText(warna_hasil, "Kuning : " + str(count_total2), (10, 100),
cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 255), 2, cv2.LINE_AA)
    text = cv2.putText(warna_hasil, "Hitam : " + str(count_total3), (220, 100),
cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 255, 255), 2, cv2.LINE_AA)

    # menampilkan gambar
    cv2.imshow("Seleksi", warna_hasil)
    cv2.imshow("Mask1", mask_hasil)
    cv2.imshow("Original1", src)
```

c

```
        cv2.imshow("edge1", grad_hasil)

        key = cv2.waitKey(1)

        if key == 27:
            break
        elif key == 32:
            start = True
            count_total = 0
            count_total1 = 0
            count_total2 = 0
            count_total3 = 0
            print("menghitung")

    cv2.destroyAllWindows()
```

BIODATA PENULIS

I Putu Dedi Semara Putra. Lahir pada tanggal 02 Mei 1998 di Denpasar, Bali. Berasal dari keluarga yang berkecukupan dengan profesi sebagai petani dan pedagang keliling. Setelah menyelesaikan Pendidikan D3 Metrologi dan Instrumentasi di Institut Teknologi Sepuluh Nopember pada tahun 2020. Dedi mengisi waktu dengan bekerja sebagai *staff Quality Assurance* di salah satu perusahaan elektro medical device di Surabaya yang Bernama PT. Sinko Prima Alloy. Dedi diterima di perusahaan tersebut saat sebelum wisuda yaitu pada bulan maret 2020 dan memutuskan untuk resign dikarenakan kuliah offline pada bulan Desember 2022. Selama dua tahun menjadi staff *Quality Assurance* di perusahaan tersebut banyak hal baru yang didapatkan seperti melakukan pengujian dengan lebih dari 200 alat uji dengan kebertelusuran dari KAN Indonesia. Serta pernah melakukan uji fungsi di customer yang terletak di Papua dan Marauke,

Setelah *resign* dari perusahaan tersebut dedi melanjutkan karir baru sebagai *software Quality Assurance Intern* di Campaig.csom. Perusahaan tersebut bergerak di bidang social impact yang memiliki visi “Untuk membuat dunia menjadi tempat yang lebih baik bagi semua orang”. Selama melaksanakan magang selama 6 bulan di perusahaan tersebut didapatkan beberapa skil baru seperti penggunaan Software Jira, Slack dan Optimasi penggunaan *Google Tools (Doc, Jamboard, sheet, Slide* dan *Google calendar).* Selain bekerja dedi juga mengikuti dua kegiatan student exchange yaitu *Web design* di Asia University dan *Project Management* di Istanbul Aydin University. Dedi memiliki hobi menari, bermain music, dan berolahraga.

Jika ingin bekenalan lebih lanjut mari berkoneksi lewat linkedin.

LinkedIn: https://www.linkedin.com/in/dedisemara/