# BAB 3
# PELAKSANAAN KERJA MAGANG

## 3.1 Kedudukan dan Koordinasi

Pelaksanaan kegiatan magang di PT Akebono Brake Astra Indonesia berada pada divisi IT, dengan posisi sebagai *AI Engineer intern*. Berdasarkan struktur organisasi yang sudah dijelaskan, IT merupakan divisi yang berada di bawah *Administration Department* dan bekerja untuk memastikan berjalannya semua sistem dan pengembangan baru yang berhubungan dengan teknologi.

Selama periode magang, divisi IT bekerja dalam beberapa posisi yaitu *website developer, application developer* dan juga *AI Engineer*. Sebagai salah satu *developer* yang bekerja dibidang *AI Engineer*, penulis ditugaskan untuk membuat sistem yang berkaitan dengan otomasi yang mengandalkan AI. Pelaksanaan kegiatan magang difokuskan untuk membantu perusahaan untuk menyelesaikan projek yang ada.

Koordinasi selama kegiatan berlangsung dilaksanakan atas arahan dari *supervisor* dari penulis. Pada awal magang diberikan *basic* dari pekerjaan yang akan dilakukan oleh *AI Engineer*. Kemudian diberikan projek untuk penulis kerjakan dengan bantuan dan arahan dari *supervisor*. Selain itu, ada juga projek yang diberikan untuk membantu *supervisor* agar projek lebih cepat selesai dan dapat diimplementasikan.

Seluruh kegiatan yang dilakukan selama masa magang dilakukan secara *WFO* dikarenakan perlunya menghadiri lapangan untuk pengujian serta kebijakan dari perusahaan. Proses rapat dilakukan setiap minggu 1 hingga 2 kali untuk memonitor *progress* dan menjelaskan kendala yang ditemukan dengan cara melakukan presentasi dari *Microsoft PowerPoint* yang sudah dibuat. Hari rapat fleksibel berdasarkan kebutuhan ketika menemukan masalah atau ingin memberikan hasil *progress* baru. Dengan pola rapat yang ada, komunikasi serta koordinasi yang dilakukan dapat berjalan dengan efektif.

## 3.2 Tugas yang Dilakukan

Selama menjadi *AI Engineer intern*, berbagai kegiatan magang dilaksanakan yang berkaitan dengan pengembangan AI di perusahaan. Seluruh proses dilakukan secara terorganisir dengan mengikuti prosedur perusahaan dan *supervisor*. Adapun

beberapa tugas yang dilakukan selama proses magang:

- Melakukan pemrograman menggunakan *python* dengan *library* seperti *mediapipe, YOLO*, *threads, deepsort* dan lain-lain.
- Membuat program untuk deteksi *inventory* pada ruangan CST
- Memasang dan mengatur kamera pada ruangan CST
- Melakukan *training model* YOLOv11 yang akan digunakan untuk melakukan deteksi dan juga *training model* untuk mendeteksi NG dan OK
- Membuat program untuk *human productivity* yang dilakukan per *cycle*
- Membuat program yang bertujuan untuk melakukan *auto restart* pada program *human productivity* untuk meningkatkan efektivitas
- Membuat program untuk *quality inspection* agar mengetahui produk OK dan NG

Dengan melaksanakan seluruh tugas yang telah disebutkan di atas, kontribusi yang diberikan dapat secara nyata meningkatkan efektivitas perusahaan yang berbasis kecerdasan buatan (AI).

Selama penulis berada pada divisi IT, tugas yang diberikan bervariasi, pada awalnya penulis diberikan tugas individual mengenai *inventory management*. Setelah 2 bulan berlalu penulis mulai diberikan tugas secara tim dikarena projek sebelumnya sedang *on hold* karena membutuhkan pertimbangan lebih. Pada projek baru penulis membantu dari program yang sudah diberikan dengan mengoptimasi dan mengembangkan lagi kode program yang diberikan sesuai dengan arahan dari tim.

Tugas pengembangan program yang diberikan berupa penjelasan secara umum terkait dengan *output* program yang diharapkan oleh pengguna. Sedangkan penggunaan *library* serta ketentuan pengolahan *output* diberikan fleksibilitas kepada penulis, sehingga memungkinkan penulis melakukan pengembangan sesuai kebutuhan sistem. Perangkat yang digunakan selama pembuatan program memiliki spesifikasi Intel Core i7-12700KF generasi ke-12, RAM 32 GB, GPU NVIDIA RTX 4070 12 GB, serta SSD sebesar 2TB. *Progress* yang dilakukan akan dipresentasikan setiap minggunya secara langsung kepada *supervisor* dari tim untuk menginformasikan kendala dan *progress* yang sudah dilakukan.

## 3.3 Uraian Pelaksanaan Magang

Selama kegiatan magang dilaksanakan, penulis melakukan berbagai projek yang berhubungan dengan pengembangan AI. Pada beberapa minggu awal magang, yaitu sekitar 9 minggu, penulis diberikan projek mandiri yang berhubungan dengan *inventory management* pada ruangan CST. Dengan tujuan dan konsep yang diberikan secara garis besar, penulis ditugaskan untuk membuat program pada ruangan CST. Selain itu, penulis juga diberikan beberapa projek lain yang berkaitan dengan pengembangan dan penerapan teknologi AI. Berikut merupakan rincian kegiatan magang yang dapat dilihat pada Tabel 3.1 :

Tabel 3.1. Deskripsi kegiatan yang dilakukan selama kegiatan magang

| **Minggu Ke -** | **Deskripsi Kegiatan** |
|---|---|
| 1 | Perkenalan dan memberi tahu projek yang akan dilakukan serta belajar menggunakan *openpose* dan *labelling* menggunakan *Roboflow* untuk projek *inventory management* |
| 2 | Melanjutkan *labelling* dan membuat konsep baru untuk penyesuaian dari konsep sebelumnya dan memulai membuat program untuk *inventory management* |
| 3 | Membuat program yang berfokus dengan 2 kamera *angle* atas dan samping untuk mendeteksi pengurangan barang yang terjadi serta mencari solusi agar barang bisa berkurang lebih dari 1 apabila pengambilan dilakukan langsung banyak |
| 4 | Mengambil data untuk *YOLO detection* agar pengurangan barang lebih akurat dan melakukan *labelling* pada data yang sudah diambil |
| 5 | Mencoba implementasi *YOLO* pada program dalam melakukan deteksi |
| 6 | Melakukan simulasi dan *set up* langsung di ruangan CST serta melakukan *adjustment* pada kode yang sudah ada |
| 7 | Melakukan *adjustment* pada kode yang sudah dibuat sehingga segala aturan untuk pengurangan barang dan kamera berjalan lebih lancar |
| | Lanjut pada halaman berikutnya |

| Minggu Ke - | Deskripsi Kegiatan |
|---|---|
| 8 | Menemukan solusi untuk *frame* yang sering turun akibat dari *frame* kamera terlalu berat menggunakan *YOLO* dan MediaPipe |
| 9 | Implementasi CUDA pada *OpenCV* dan juga *YOLO* untuk membuat *frame* pada kamera lebih stabil |
| 10 | Mendapatkan projek baru mengenai *human productivity* dan melakukan optimasi pada kode dan membantu *labelling* untuk projek *quality inspection* |
| 11 | Melanjutkan *labelling* dan melakukan *trial* pada projek *quality inspection* |
| 12 | Mengerjakan projek *Human Productivity* dari *line* yang berbeda dan menghubungkannya pada *server* |
| 13 | Melakukan *labelling* pada *line production* serta membuat program yang dapat digunakan dengan model yang sudah dibuat berdasarkan data yang sudah dilabel |
| 14 | Melakukan *labelling* ulang pada *Disc Brake* (DB2) D14 dan D20 untuk menambahkan data yang kurang, serta melakukan *trial* pada model yang sudah di-*learning*. D14 dan D20 merupakan jenis rem cakram pada setiap *line* seperti DB2 |
| 15 | Menggabungkan program yang sudah dibuat untuk sampel D14 dan D20 agar lebih mudah melakukan *quality inspection* ketika mengubah sampel |
| 16 | Membuat *heatmap* pada projek *human productivity* untuk melihat pergerakan pekerja |
| 17 | Melanjutkan pembuatan pada *line* terakhir yaitu DB3 untuk projek *quality inspection* dengan melakukan label pada sampel D38L, D40L dan YHA serta membuat program untuk melakukan deteksi |
| 18 | Membuat program untuk DB3 agar bisa digunakan pada *line* dengan ketentuan dapat mengubah semua model pada program dengan *user input* |
| 19 | Menghubungkan API dengan semua program pada DB2 dan DB3 sehingga dapat di implementasikan pada *line* |

Pada minggu pertama, penulis melakukan pengenalan dan dijelaskan mengenai projek-projek yang ada pada bidang IT. Semua projek yang dijelaskan berhubungan dengan AI, maka dari itu awal masuk magang disarankan untuk mempelajari hal dasar dari projek-projek yang diberikan. Pada akhir minggu ke-1, penulis sudah mulai diberikan tugas mengenai projek yang berkaitan dengan AI.

Memasuki minggu kedua dan ketiga, penulis mulai berfokus untuk mengerjakan projek yang diberikan mengenai *inventory management* yang berfokus pada ruangan CST untuk melakukan pengurangan barang secara otomatis. Selama 2 minggu penulis berfokus pada pembuatan logika untuk pengurangan serta memikirkan konsep yang sesuai untuk projek yang diberikan sehingga masalah yang muncul dapat diselesaikan.

Pada minggu keempat dan kelima, setelah melakukan konsultasi *progress* disarankan untuk menggunakan *YOLO object detection* sehingga pengurangan lebih akurat. Selama minggu ini, penulis mengambil data berupa barang yang ada di ruangan CST yang kemudian akan dilakukan *labelling* dan *training* sehingga dapat diimplementasikan di program yang sudah dibuat.

Memasuki minggu keenam hingga kesembilan, pembuatan program mulai berfokus untuk simulasi langsung di ruangan CST. Untuk memastikan semua program berjalan dengan lancar, kode dari program yang sudah dibuat butuh dilakukan penyesuaian atau optimisasi untuk menyesuaikan tempat melakukan simulasi. Penyesuaian yang dilakukan selama minggu ini adalah membuat *mediapipe* menjadi tidak terlihat untuk membuat kamera tidak *blur* karena *frame* yang terlalu berat, penyesuaian mengenai pergerakan laci untuk mengetahui apabila laci sudah terbuka. Hal tersebut dapat membantu menghindari deduksi yang salah apabila laci belum terbuka.

Selain dari itu penyesuaian pada *YOLO* juga dilakukan agar resolusi dari *YOLO detection* tidak melebihi resolusi dari kamera yang membuat *frame* kamera turun secara signifikan. Untuk mengatasi *frame* pada kamera yang sering turun, penulis menggunakan GPU CUDA yang membuat kamera dapat memproses *frame* lebih cepat dan mencegah terjadinya *frame drop* ketika *YOLO* maupun *mediapipe* aktif.

Memasuki minggu kesepuluh dan kesebelas, setelah projek *inventory management* ditunda akibat banyak kendala dari *angle* kamera dan kebutuhan dari perusahaan maka penulis diberikan projek baru yang berhubungan dengan *human productivity*. Projek *human productivity* berfokus pada perhitungan seberapa efektif karyawan bekerja dengan menghitung *cycle* dari setiap pekerjaan yang dilakukan. Pada projek ini penulis ditugaskan untuk melakukan optimisasi pada kode yang

sudah ada sehingga kamera dan program yang dijalankan dapat berjalan dengan lancar tanpa adanya kendala. Penulis juga ditugaskan membuat program agar ketika program rusak/*crash* akan melakukan *restart* secara otomatis.

Pada minggu ini, penulis juga ditugaskan untuk membantu melakukan *labelling* pada projek lain untuk melakukan deteksi pada barang yang NG atau sudah bagus. Setelah *labelling* penulis ditugaskan untuk membuat program yang dapat bekerja sesuai dengan arahan yang sudah diberikan.

Memasuki minggu kedua belas, penulis diberikan tugas untuk menghubungkan projek *human productivity* pada server untuk mengetahui *event* apa saja yang dilakukan. Penulis membuat program sebagai *listener* untuk melihat *event* yang terjadi pada *server*.

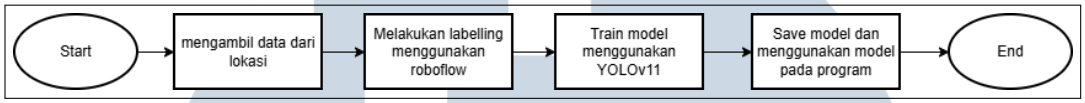
Pada minggu ketiga belas dan keempat belas, penulis ditugaskan untuk melanjutkan *labeling* dari setiap *production line* yang ada dan juga mengambil ulang sampel DB2 untuk memperbanyak data. *Labelling* dilakukan untuk mendeteksi bagian yang NG pada barang yang akan dideteksi untuk mempercepat dan mempermudah pekerjaan. Proses pelabelan itu sendiri menggunakan *segmentation* agar bagaian dari barang lebih mudah terlihat setelah dilakukan *train* dengan model *YOLO*. Selain itu, penulis juga diberikan tugas lanjutan untuk membuat program dari masing-masing *line* sehingga hasil model *YOLO* dapat digunakan untuk mendeteksi barang NG.

Memasuki minggu kelima belas, penulis ditugaskan untuk membuat program yang menggabungkan model D14 dan D20 untuk mempermudah melakukan *quality inspection* pada sampel. D14 dan D20 sendiri merupakan jenis rem cakram berbeda yang berada pada *line* DB2. Pergantian model dapat diganti menggunakan tombol *keyboard*. Model yang di *load* tidak bersamaan untuk mengurangi penggunaan *Graphics Processing Unit (GPU)*.

Pada minggu keenam belas dan ketujuh belas, penulis ditugaskan untuk menambahkan fitur *heatmap* pada projek *human productivity* untuk mengetahui pergerakan dari pekerja di jam kerja. Selain itu, pada 2 minggu ini juga melanjutkan pengerjaan program dari DB3 yaitu pengambilan sampel, pelabelan dan juga pembuatan program. Projek DB3 baru dilakukan pengambilan sampel dikarenakan *production line* yang selalu jalan sehingga belum sempat untuk mengambil data.

Memasuki minggu akhir, yaitu minggu ketujuh belas dan kedelapan belas, penulis diminta untuk membuat program pada DB3 dengan ketentuan yang sama dengan DB2. Ditambah lagi penulis diminta untuk memasukkan API yang sudah disiapkan kedalam program agar dapat di implementasikan pada *line*

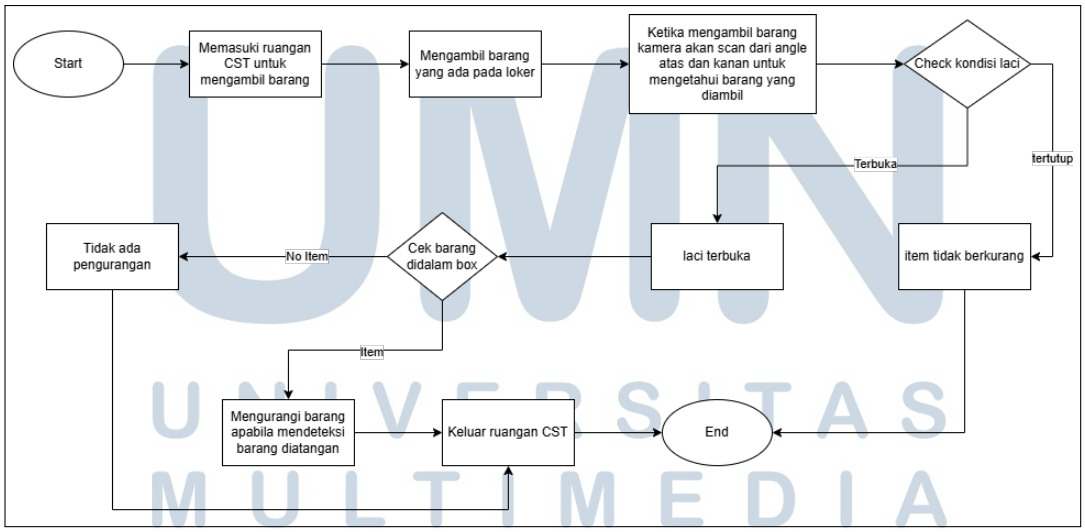
Selain dari rincian pekerjaan yang dilakukan setiap minggu, ada juga proses yang dilakukan untuk pembuatan model *YOLO* yang dapat dilihat pada Gambar 3.1. Langkah-langkah yang dilakukan pada *flowchart* dilakukan secara terus menerus setiap membutuhkan model baru pada program yang berbeda.




Gambar 3.1. *Flowchart* Pembuatan model YOLO

### 3.3.1 Inventory Management pada Ruangan CST

Tugas pertama yang diberikan selama kegiatan magang merupakan pembuatan kamera pada ruangan CST sehingga pengambilan barang dapat dilakukan secara otomatis. Projek pertama ini merupakan projek individu yang diberikan dari *supervisor*. Hal pertama yang dilakukan selama berlangsungnya projek ini adalah membuat konsep yang dapat menjalankan tugasnya sesuai dengan yang sudah diminta oleh perusahaan. Pembuatan konsep disarankan untuk menggunakan *flowchart* agar alur dari proses dapat dilihat secara jelas. Pada gambar 3.2 dapat dilihat alur untuk proses terjadinya otomasi untuk *inventory management*.




Gambar 3.2. Alur konsep dari *inventory management*

Pada dasarnya semua proses akan dilakukan menggunakan kamera untuk mempermudah deteksi, pada proses awal dibutuhkan peletakan kamera dari sudut atas dan samping sehingga dapat melihat laci dari 2 koordinat agar dapat

membedakan setiap barang dari laci [4]. Sistem pengurangan dilakukan apabila laci terbuka, tangan berada pada laci yang dideteksi menggunakan *landmark* pada *mediapipe* dan mendeteksi barang dengan menggunakan *YOLO* [5]. Kode penggunaan *mediapipe* dapat dilihat pada Kode 3.1. Pada program terdapat *Region of Interest (ROI)* pada masing masing baris dan kolom. Pada awalnya, ROI menampilkan kondisi laci dalam keadaan tertutup. Laci akan terdeteksi terbuka apabila terdapat pergerakan di dalam ROI. Setelah terbuka, laci akan tetap berada dalam kondisi terbuka selama beberapa waktu untuk mencegah pendeteksian berulang yang tidak diperlukan. Kode 3.2 menunjukkan logika yang digunakan ketika laci akan terbuka dan durasi laci terbuka. Setelah laci terbuka maka kamera akan melakukan deteksi pada tangan yang menggunakan *mediapipe* untuk mengetahui kerangka tangan dan juga *YOLO* untuk deteksi barang yang diambil [6].

```
use_cached = False
now_t = time.time()
if self._hands_last is not None and (now_t - self._hands_last_t)
   <= self.hands_ttl_sec:
    use_cached = True

if (self._frame_idx % self.hands_every_n) == 0 or not use_cached:
    rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
    results = self.hands.process(rgb)
    self._hands_last = results
    self._hands_last_t = now_t
else:
    results = self._hands_last
```

Kode 3.1: Kode menunjukkan penggunaan mediapipe

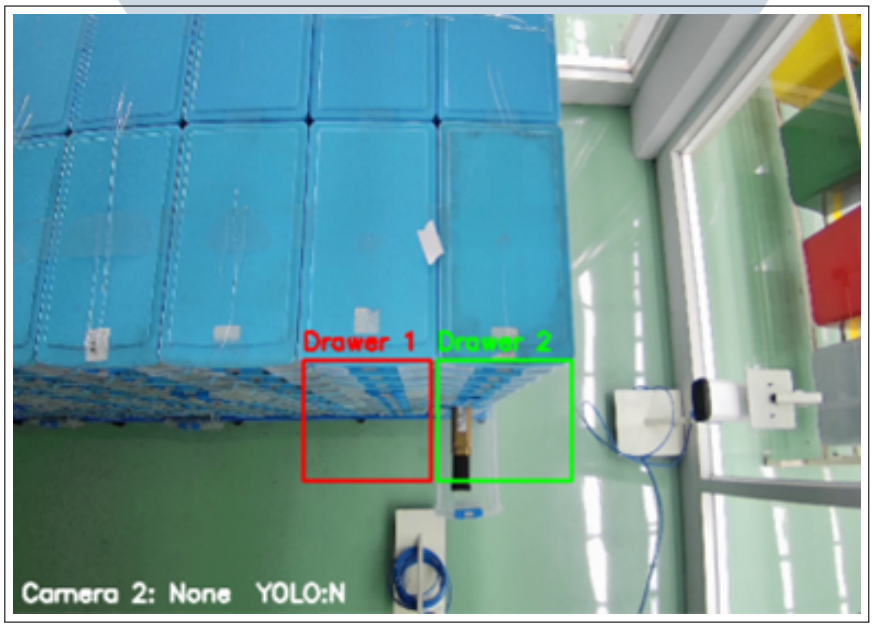
```
if not drawer_open and motion_detected and not
   hand_inside_any_drawer[drawer_name]:
    self.drawer_state[drawer_name] = True
    self.drawer_open_until[drawer_name] = time.time() + self.
   KEEP_OPEN_TIME

if self.drawer_state[drawer_name]:
    if motion_detected:
        self.drawer_open_until[drawer_name] = time.time() + self.
   KEEP_OPEN_TIME
    elif time.time() > self.drawer_open_until[drawer_name]:
        self.drawer_state[drawer_name] = False
        self.grab_latched[drawer_name] = False
        self.backgrounds[drawer_name] = None
```

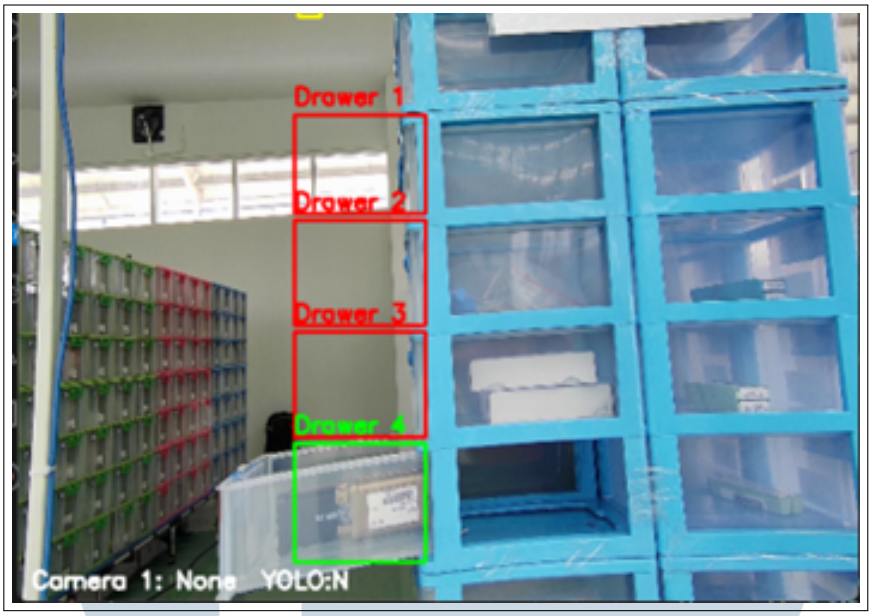
```
12          self.prev_gray_roi[drawer_name] = None
```

Kode 3.2: Kode menunjukkan logika untuk pembukaan laci

Pada gambar 3.3 dan 3.4 merupakan bentuk dari kamera yang sudah dijalankan menggunakan Python. Kotak berwarna merah dan hijau merupakan ROI yang memiliki peran terpenting karena segala proses pengurangan terjadi di dalam ROI tersebut. ROI berwarna merah menunjukkan bahwa laci masih tertutup dan ROI hijau menunjukkan laci yang terbuka. Pada kamera samping akan melakukan deteksi *YOLO* sehingga jika tangan dan barang berada di dalam ROI maka proses deduksi akan terjadi. Untuk mengetahui barang apa yang dideduksi sudah di *set* pada program, apabila barang di dalam *drawer 1* pada kamera samping dan *drawer 1* pada kamera atas maka merupakan barang satu dan seterusnya. Kode 3.3 merupakan logika yang digunakan sehingga pengurangan barang dapat dilakukan secara akurat.




Gambar 3.3. Kamera tampak atas pada ruang CST




Gambar 3.4. Kamera tampak samping pada ruang CST

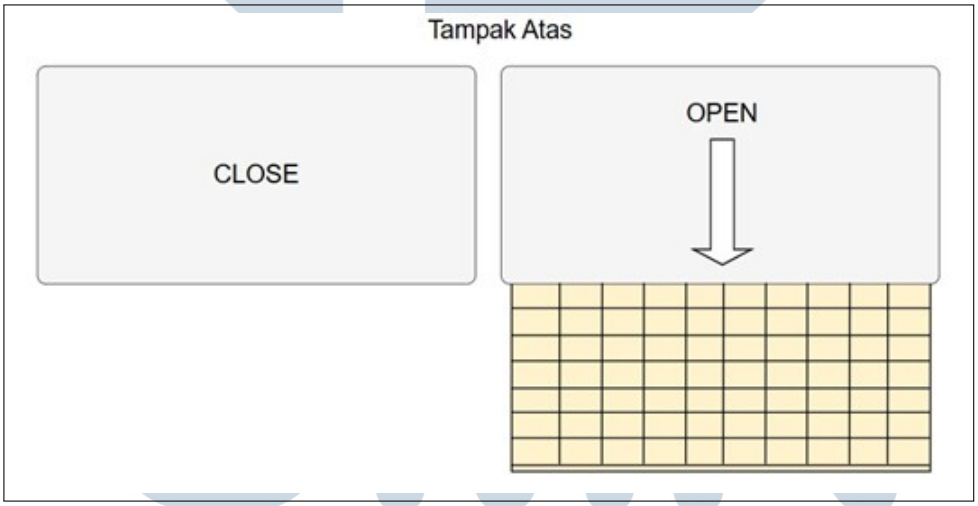
```
DEDUCTION_RULES = {
    ("Drawer 1", "Drawer 1"): "Item A",
    ("Drawer 1", "Drawer 2"): "Item B",
    ("Drawer 2", "Drawer 1"): "Item D",
    ("Drawer 2", "Drawer 2"): "Item E",
    ("Drawer 3", "Drawer 1"): "Item G",
    ("Drawer 3", "Drawer 2"): "Item H",
    ("Drawer 4", "Drawer 1"): "Item C",
    ("Drawer 4", "Drawer 2"): "Item F",
}

now = time.time()
if od1 and od2 and (od1, od2) in DEDUCTION_RULES:
    hand_ok = (hd1 == od1) or (hd2 == od2)
    yolo_ok = cam1.yolo_item_in_drawer.get(od1, False) if cam1.
    enable_yolo else True

    if hand_ok and yolo_ok and (now - last_deduction_time) >
    DEBOUNCE_SEC:
        item = DEDUCTION_RULES[(od1, od2)]
        deduct_stock(item)
        last_deduction_time = now
```

Kode 3.3: Kode untuk *rule* dari pengurangan barang

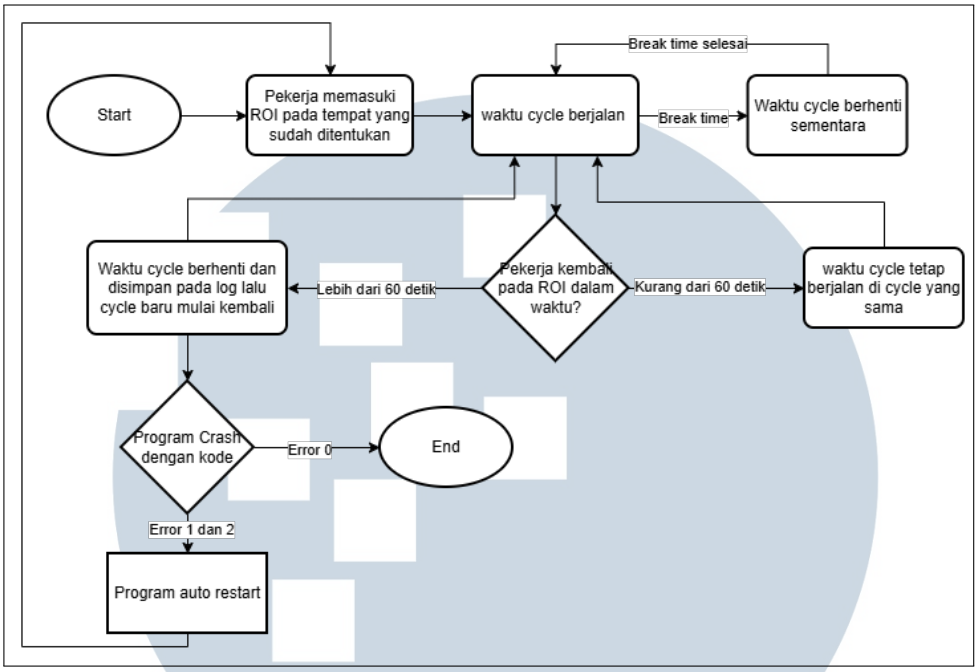
Selama berjalannya projek ini ada beberapa masalah setelah semua berjalan dan proses deduksi dapat dilakukan. Pada kamera apabila jarak terlalu jauh maka ROI akan berada di luar dari laci yang dipakai sehingga deteksi barang akan menjadi tidak akurat selain itu, semakin jauh jarak kamera maka deteksi barang menggunakan *YOLO* semakin tidak akurat atau tidak terdeteksi, maka dari itu ada konsep baru yang ditawarkan penulis agar deteksi dilakukan lebih mudah dan akurat. Pada Gambar 3.5 menunjukkan konsep baru pada *inventory management* dengan menggunakan kamera tampak atas dan juga sensor. Laci yang digunakan merupakan laci besar yang memiliki banyak baris dan setiap baris dapat diisi oleh banyak barang. ROI dari kamera ataspun juga akan dibuat dinamik sehingga setiap sensor mendeteksi baris terbuka maka ROI akan muncul berdasarkan baris laci yang dibuka. Proses deduksi yang terjadi masih sama dengan konsep yang sebelumnya. Setelah menyelesaikan tahap ini, penulis diberikan projek lain karena projek sebelumnya berfungsi sebagai *proof of concept* untuk penerapan sistem *inventory management*




Gambar 3.5. Konsep baru untuk *inventory management*

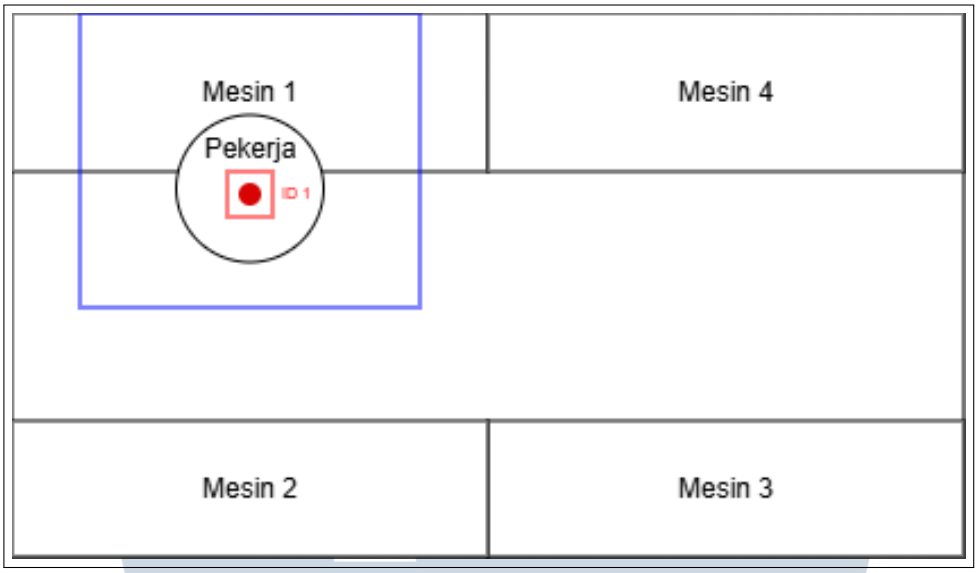
### 3.3.2 Projek Human Productivity

Setelah melakukan projek *inventory management*, penulis ditugaskan untuk membantu projek *human productivity*. Pada projek ini penulis membantu *supervisor* untuk membuat *cycle* pada setiap mesin untuk mengetahui seberapa efektif pekerjaan dalam setiap mesin. Alur berjalannya projek ini dapat dilihat pada Gambar 3.6.




Gambar 3.6. *Flowchart* alur berjalannya projek *human productivity*

Program yang dibuat bertujuan untuk menghitung waktu yang dibutuhkan untuk setiap pekerja menyelesaikan tugasnya dalam 1 *cycle*. Pada program ini terdapat waktu *cycle* dan juga jumlah *cycle* yang dilakukan. *Cycle* yang didapat akan disimpan pada file csv dan juga gambar berupa *graph* untuk mempermudah melakukan perbandingan waktu dan melihat rata-rata waktu setiap *cycle*. Gambar 3.7 merupakan hasil ilustrasi dari program yang dibuat. lingkaran merupakan pekerja sedangkan lingkaran merah merupakan warna helm yang digunakan untuk melakukan deteksi pergerakkan pekerja yang memiliki memiliki ID.




Gambar 3.7. Hasil illustrasi program untuk projek *human productivity*

Pada Gambar 3.7 dapat dilihat terdapat ID pada helm yang menunjukkan pekerja pada kamera. ID pada helm didapat menggunakan *library deepsort* pada Python. Dengan melakukan kombinasi *library* antara *deepsort* dan *YOLO* maka bisa mendeteksi warna merah pada helm serta memberikan ID. ROI berwarna biru tua merupakan tempat *cycle* dimulai dan juga berganti. *Trigger cycle* akan terjadi apabila ID pada helm memasuki ROI berwarna biru tua. *Cycle* akan memulai *cycle* baru apabila ID pada helm berada di luar ROI selama lebih dari 60 detik untuk menghindari pergantian *cycle* yang terlalu cepat apabila kembali pada ROI kurang dari 60 detik. Kode 3.4 menunjukkan potongan kode yang mengenai logika *cycle* yang digunakan.

```
if inside and st['cycle_start'] is None:
    st['cycle_start'] = now
    st['last_exit_time'] = None
    print(f"[CYCLE] ID {track_id} - Cycle started")

if (not inside) and st['inside']:
    st['last_exit_time'] = now
    print(f"[CYCLE] ID {track_id} - Left polygon (starting 60s
    grace)")
```

Kode 3.4: Kode menunjukkan logika *cycle*

Selain itu, ada sistem *break* yang membuat waktu *cycle* akan berhenti apabila sedang jam istirahat sehingga waktu tidak terus berjalan meskipun sedang tidak bekerja. Logika pada kode untuk jam istirahat dapat dilihat pada Kode 3.5. Pada

projek ini terdapat juga program sh yang berfungsi melakukan *auto restart* dalam 3 detik apabila program *crash* sehingga tidak perlu melakukan *run* ulang secara manual. Kode 3.6 menunjukkan *rule* apabila program *error* dengan kode 1,2 dan juga 134. Di akhir dari membuat program untuk projek *human productivity*, penulis diminta untuk menambahkan *heatmap* untuk melihat pergerakan dari pekerja ketika melakukan pekerjaan. Kode *heatmap* dapat dilihat pada Kode 3.7. Hasil dari *heatmap* dapat dilihat pada Gambar 3.8.

```
def is_break_time():
    now = datetime.datetime.now().time()

    breaks = [
        (datetime.time(10, 0), datetime.time(10, 10)),
        (datetime.time(12, 0), datetime.time(12, 45)),
        (datetime.time(15, 30), datetime.time(15, 40)),
    ]

    for start, end in breaks:
        if start <= now <= end:
            return True
    return False
```

Kode 3.5: Logika *break time*

```
if [ $EXIT_CODE -ne 1 ] && [ $EXIT_CODE -ne 2 ] && [ $EXIT_CODE -
    ne 134 ]; then
    echo "[INFO] Exit code $EXIT_CODE. Stopping restart loop."
    break
fi
```

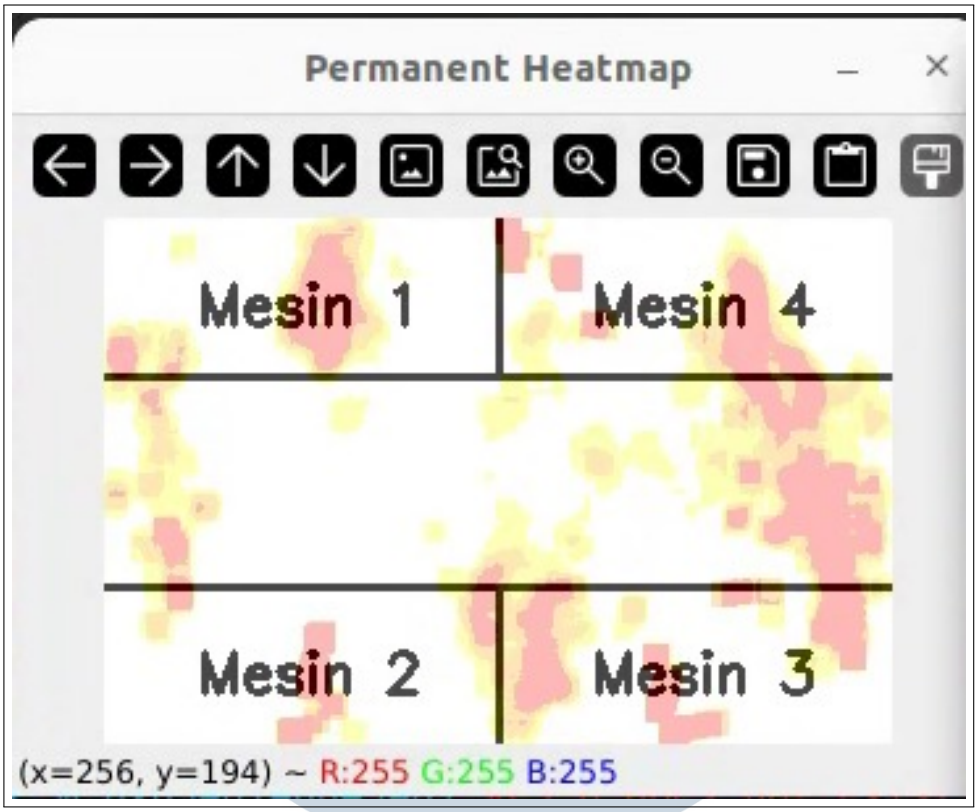
Kode 3.6: *Rule* melakukan *restart* pada program

```
def update_heatmap(cx, cy, frame_w, frame_h):
    hx = int((cx / frame_w) * HEATMAP_W)
    hy = int((cy / frame_h) * HEATMAP_H)

    if 0 <= hx < HEATMAP_W and 0 <= hy < HEATMAP_H:
        blob_size = 5
        y1 = max(0, hy - blob_size)
        y2 = min(HEATMAP_H, hy + blob_size)
        x1 = max(0, hx - blob_size)
        x2 = min(HEATMAP_W, hx + blob_size)

        heatmap_buffer[y1:y2, x1:x2] += 1
```

Kode 3.7: Kode menunjukkan pembuatan *heatmap*




Gambar 3.8. Heatmap setelah program dijalankan

Selain itu, penulis juga ditugaskan untuk membuat projek *human productivity* pada *line* yang lain. Pada *line* lain, helm memiliki warna yang berbeda sehingga tidak terjadi masalah ketika pekerja melewati *line* lain. Dengan itu, pada *line* ini penulis ditugaskan untuk membuat program yang serupa dengan mengganti fungsi *YOLO* sehingga dapat mendeteksi warna helm yang berbeda. Deteksi warna berbeda pada helm tidak dilakukan dengan menggunakan model *YOLO* yang berbeda tetapi menyesuaikan *opencv* sehingga mengubah warna biru yang ada pada *frame* menjadi warna merah sehingga tetap terdeteksi. Perubahan warna menggunakan *opencv* dapat dilihat pada kode 3.8. Untuk meminimalkan kesalahan deteksi, penyesuaian warna seperti kontras dilakukan dengan sangat presisi agar sistem dapat berjalan dengan lancar saat diuji. Semua logika *cycle* yang ada pada program ini sama dengan program dari projek *human productivity* sebelumnya.

```
frame_for_yolo = frame.copy()

hsv = cv2.cvtColor(frame_for_yolo, cv2.COLOR_BGR2HSV)
mask_blue = cv2.inRange(hsv, BLUE_LOWER_HSV, BLUE_UPPER_HSV)
mask_blue = cv2.morphologyEx(mask_blue, cv2.MORPH_OPEN,
    MASK_KERNEL)
```

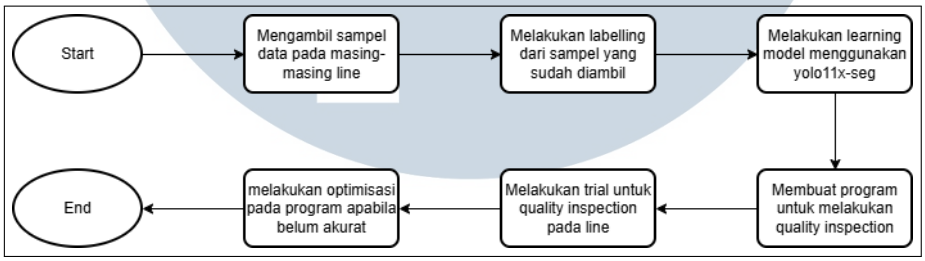
```
mask_blue = cv2.morphologyEx(mask_blue, cv2.MORPH_CLOSE,
    MASK_KERNEL)

frame_for_yolo[mask_blue > 0] = (0, 0, 255)
```

Kode 3.8: Kode perubahan warna menggunakan opencv

### 3.3.3 Quality Inspection

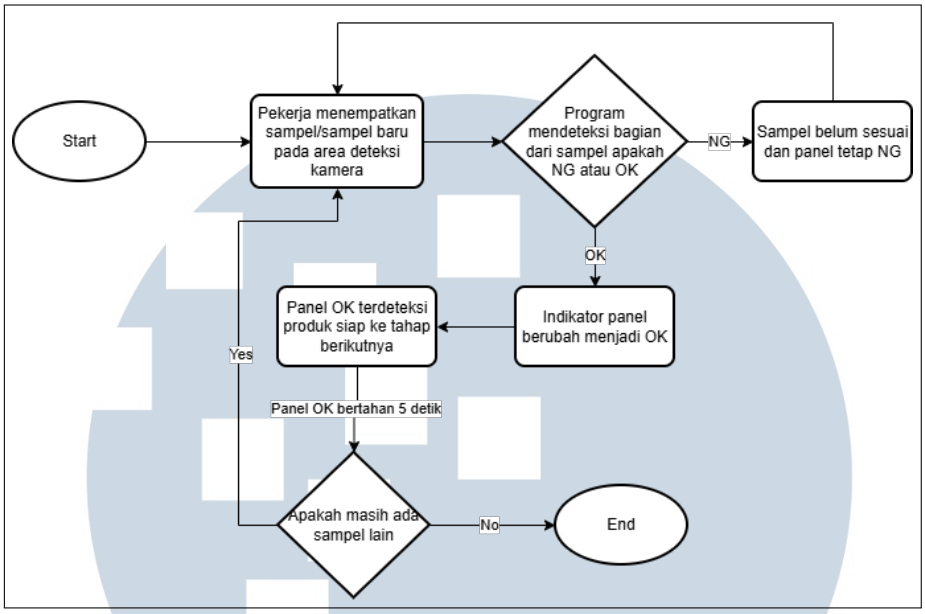
*Quality inspection* merupakan sebuah projek yang mengecek rem cakram apakah sudah lengkap atau kualitasnya sudah bagus untuk digunakan. Pada *quality inspection* terdapat banyak *line* yang memiliki berbagai macam jenis rem cakram yang digunakan seperti D20, D14 dan lain-lain. Pada projek ini *YOLO* merupakan *library* yang digunakan untuk melakukan deteksi. Gambar 3.9 merupakan alur pengerjaan projek *quality inspection*




Gambar 3.9. Alur pengerjaan projek *quality inspection*

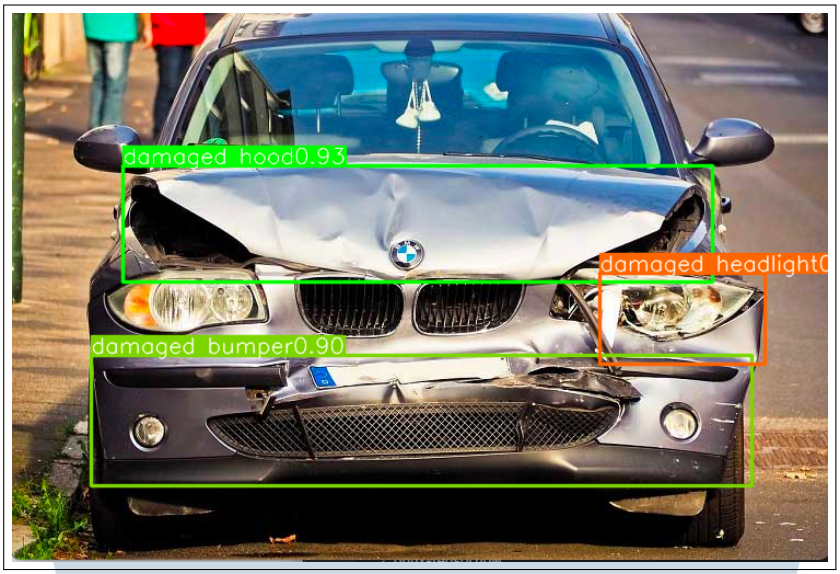
#### A Disc Brake 2

*Disc Brake 2 (DB2)* merupakan salah satu *line* yang digunakan untuk melakukan *quality inspection* pada rem cakram yang ada. Pada *line* ini terdapat 2 jenis rem cakram berbeda yaitu D14 dan D20. Setiap dari D14 dan D20 terdapat bagian kanan dan kiri. Saat melakukan *quality inspection* setiap rem cakram yang berbeda jenis membutuhkan model yang berbeda meskipun ada beberapa bagian berbentuk sama. Langkah yang dilakukan dalam projek ini cukup panjang dan juga repetitif. Gambar 3.10 merupakan alur kerja program DB2.




Gambar 3.10. Alur kerja program DB2

Langkah awal yang dilakukan adalah mengambil data sampel langsung pada *line* DB2 sehingga ketika melakukan *labelling* memiliki penempatan yang sama sehingga deteksi dapat dilakukan lebih mudah dan akurat. Label yang digunakan adalah NG dan juga OK, label NG merupakan produk yang belum siap pakai sedangkan OK merupakan produk yang sudah siap untuk digunakan. Data yang diambil sekitar 100 dari setiap sampel yang ada seperti 100 untuk kanan NG, 100 untuk kiri NG, 100 untuk kanan ok dan 100 untuk kiri ok. Setelah pengambilan data maka penulis akan melakukan *labelling* sesuai dengan kriteria yang sudah ditentukan. Gambar 3.11 merupakan ilustrasi untuk pelabelan sampel. Sampel DB yang dilabel juga serupa dengan ilustrasi yang diberikan dengan melabel beberapa *part* yang dibutuhkan.




Gambar 3.11. Hasil ilustrasi untuk penggunaan YOLO saat *labelling*

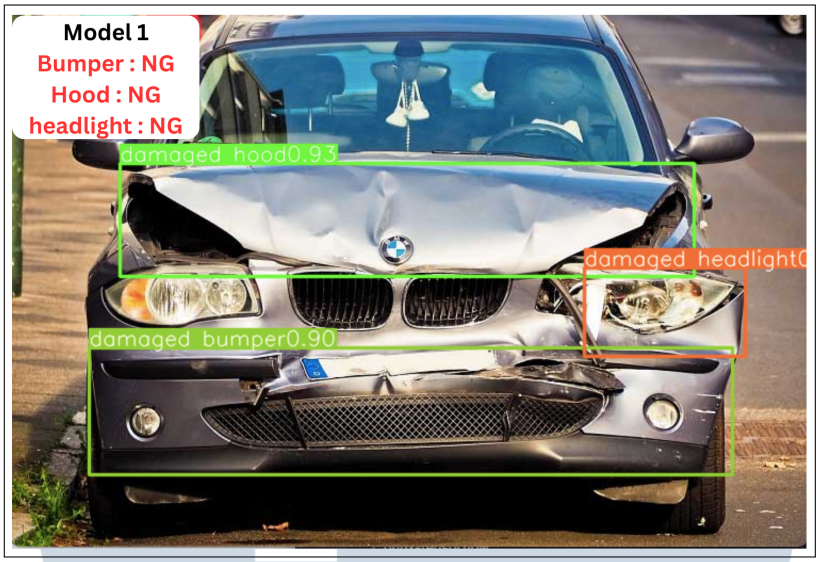
Setelah tahap *labelling* selesai dilakukan maka akan melakukan *learning* pada model menggunakan yolo11x-seg untuk mendapatkan model yang akan digunakan pada *quality inspection*. Setelah mendapatkan model, penulis membuat program untuk melakukan *inspection* sehingga model *YOLO* yang sudah dibuat dapat digunakan. Program yang dibuat memiliki panel untuk mendeteksi apakah bagian yang diberi label sudah ok atau masih NG. Selama pengecekan sampel yang masih mendapat status NG tidak akan lanjut ke proses selanjutnya dan akan dipisahkan sebagai sampel yang masih harus dilengkapi untuk beberapa *part* yang masih kurang. Pembuatan panel pada program dapat dilihat pada Kode 3.9. Bagian yang sudah ok akan berubah menjadi ok dan apabila masih NG atau tidak dapat dideteksi maka tidak akan berubah. Gambar 3.12 merupakan hasil dari ilustrasi program yang sudah dijalankan.

```
if cls_name in CLASS_MAP and conf >= 0.75:
    mapped_label = CLASS_MAP[cls_name]
    detected_now.add(mapped_label)
    last_seen[mapped_label] = current_time

    if first_seen[mapped_label] == 0:
        first_seen[mapped_label] = current_time

    elif current_time - first_seen[mapped_label] >= OK_DELAY:
        status[mapped_label] = "ok"
```

Kode 3.9: Kode validasi status OK menggunakan waktu deteksi




Gambar 3.12. Hasil ilustrasi program yang sudah dijalankan

Pada *line* DB2 memiliki 2 macam tipe rem cakram, maka dari itu pada program dibutuhkan sistem yang dapat mengubah model menggunakan *user input* untuk mempermudah melakukan *inspection* pada rem cakram yang berbeda. Pada program pergantian model menggunakan *keyboard* untuk sementara waktu sebelum menggunakan API untuk percobaan apakah sudah berjalan dengan lancar. Apabila model yang sudah dilakukan *training* menggunakan *YOLO* berganti, maka pada kiri atas program akan menjadi D20 dan sebaliknya. Kode 3.10 menunjukkan pergantian model apabila menekan tombol *keyboard*. Setelah semua program dijalankan maka penulis akan melakukan *trial* secara langsung pada *line* untuk melihat apakah semua program sudah berjalan dengan lancar dan memastikan apakah sudah layak untuk melakukan implementasi pada *server*.

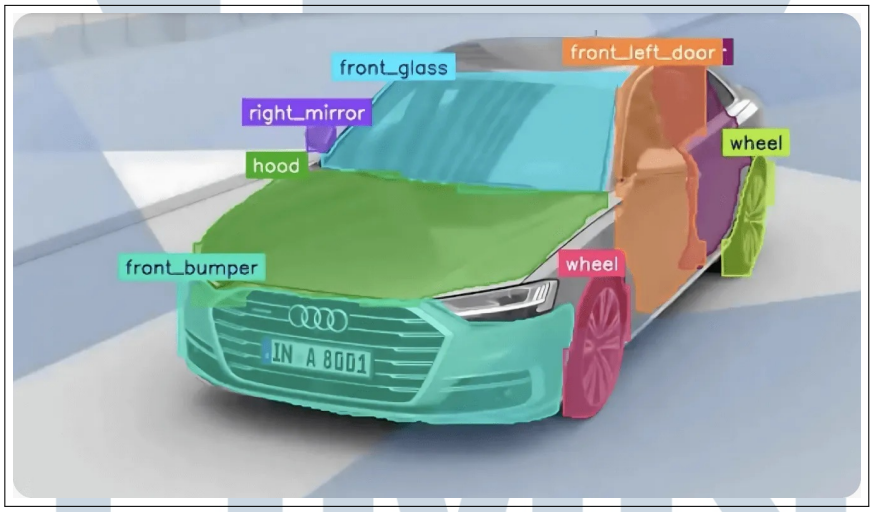
```
if key == ord('w'):
    if current_name == "Model 1":
        load_model(MODEL2_PATH, "Model 2")
        write_debug_log("D20")
    else:
        load_model(MODEL1_PATH, "Model 1")
        write_debug_log("D14")
```

Kode 3.10: Kode perubahan model menggunakan tombol *keyboard*

### B Disc Brake 3

Pada DB3 merupakan *line* yang mirip dengan DB2 yaitu *line* untuk pengecekan rem cakram dengan tipe yang berbeda. Pada DB3 terdapat 3 tipe rem cakram berbeda yaitu D38L, D40L dan YHA. Pada *line* DB3 juga memiliki proses yang sama dengan DB2 yaitu melakukan pengambilan sampel 100 untuk setiap sampel yang berbeda, lalu melakukan *labelling* menggunakan *Roboflow* berdasarkan ketentuan yang sudah dibuat.

Pelabelan yang dilakukan pada DB3 masih sama dengan DB2. Hasil label pada dapat dilihat pada ilustrasi Gambar 3.13. Pelabelan penting dilakukan seakurat mungkin dengan sudut dan penempatan sesuai dengan ketentuan yang ditentukan karena perbedaan penempatan dapat mempengaruhi akurasi saat melakukan tes pada *line.*




Gambar 3.13. Hasil ilustrasi sampel menggunakan *segmentation*

Pada *line* DB3 penulis hanya diminta untuk melakukan *labelling* pada semua sampel dan tidak diminta untuk membuat program seperti DB2. Setelah selesai melakukan *labelling* pada semua jenis sampel DB3, penulis melakukan *learning* pada setiap sampel menggunakan yolo11x-seg yang kemudian akan digunakan pada program sama seperti dengan program DB2.

## 3.4 Kendala yang Ditemukan

Selama magang untuk membuat *inventory management* terdapat beberapa kendala yang dihadapi selama pembuatan model dan sistem. Berikut merupakan

kendala yang dialami:

1. Ketika melakukan simulasi langsung di ruangan, laci yang berada pada bagian lebih dalam pada kamera tidak dapat dideteksi atau keluar dari ROI yang sudah ditetapkan.
2. Model *YOLO* yang digunakan terkadang sulit untuk melakukan deteksi karena banyaknya *noise* pada ruangan.
3. Kamera (*OpenCV*) dan *YOLO* menyebabkan *frame* pada kamera menurun sehingga sulit untuk mendeteksi *motion* pada laci yang dibuka.
4. Pada projek *human productivity*, *output* yang didapatkan belum tersimpan, tidak memiliki visualisasi, dan kamera memiliki *Frame per Second (FPS)* yang sering rusak setiap beberapa menit.
5. Kendala pada projek *Quality Inspection*: ketika melakukan *trial* program, beberapa label tidak dapat mendeteksi barang yang NG atau OK.

## 3.5 Solusi Atas Kendala yang Ditemukan

Solusi yang ditemukan untuk menghadapi kendala-kendala di atas adalah:

1. Mencoba konsep berbeda dengan sistem yang sama, seperti menggabungkan sensor dan kamera.
2. Melakukan pengambilan gambar dan pelatihan ulang (*training*) dengan resolusi yang lebih tinggi pada *YOLO* sehingga model lebih mudah mendeteksi item.
3. Menggunakan GPU pada *OpenCV* dan *YOLO* sehingga kamera memiliki FPS yang lebih stabil serta menghilangkan *grey screen* pada *OpenCV*.
4. Menghapus *info panel* untuk mengurangi beban pada *frame* sehingga kamera tetap stabil.
5. Melakukan pengambilan ulang data dengan memastikan peletakan sampel pada tempat yang tepat dan memastikan kondisi cahaya saat pengambilan sampel sama dengan saat melakukan *trial*.