Sistem Penghitung Manusia dan Pengenalan Wajah Menggunakan *Drone* Berbasis *Web Flask*

1st Muhammad Difa Dhiyaul Haq
Fakultas Teknik Elektro
Universitas Telkom
Bandung, Indonesia
muhammaddifadhiyaulh@student.telkomuniversity.ac.id

2nd Casi Setianingsih
Fakultas Teknik Elektro
Universitas Telkom
Bandung, Indonesia
setiacasie@telkomuniversity.ac.id

3rd Anggunmeka Luhur Prasasti
Fakultas Teknik Elektro
Universitas Telkom
Bandung, Indonesia
anggunmeka@telkomuniversity.ac.id

Abstrak* — Penelitian tentang penggunaan drone untuk menghitung manusia dan pengenalan wajah didorong oleh sejumlah tantangan di berbagai bidang. Tantangan-tantangan ini mencakup efisiensi terkait dengan penghitungan dan pengenalan secara manual di area ramai atau luas, perlunya pengelolaan kerumunan yang efektif selama suatu acara dan di ruang publik, serta urgensi untuk dengan cepat menentukan jumlah korban selamat dalam skenario bencana. Program yang menggunakan drone untuk menghitung manusia dan pengenalan wajah dapat secara efektif mengatasi berbagai tantangan. Program tersebut dapat membantu penyelenggara acara dalam mengelola kerumunan selama pertemuan besar, membantu misi pencarian dan penyelamatan dengan menghitung secara akurat korban selamat di daerah bencana, meningkatkan keamanan dan pengawasan melalui pelacakan individu secara real-time di zona terbatas, membantu pengelolaan kerumunan di tempat umum, mempercepat proses kontrol perbatasan dan memastikan kepatuhan kesehatan dan keselamatan di lingkungan industri. Solusi serbaguna ini menunjukkan potensi untuk meningkatkan efisiensi, keamanan, dan pengambilan keputusan di berbagai sektor. Hasil yang didapat pada pengujian menggunakan sistem yang dibuat oleh para penulis menuai hasil yang cukup memumpuni. Dataset yang digunakan adalah hasil train terbaik didapatkan dengan konfigurasi dataset adalah rasio 70 train : 15 valid : 15 test, learning rate = 0.0001, batch 6 dan epoch 250. Program mampu mengenali dan menghitung manusia dengan rata-rata akurasi penghitungan dari berbagai ketinggian dengan kondisi drone bergerak adalah 68,75% dan dengan drone dalam kondisi diam dari berbagai ketinggian adalah 69,70%. Hasil dari rata-rata akurasi program face recognition dengan drone keadaan diam adalah 40,74% dan 14,81% untuk akurasi dengan drone dengan kondisi bergerak

***Kata kunci* — Pengenalan wajah, Penghitung manusia, Program, Dataset, real-time**

I. PENDAHULUAN

Dewasa ini perkembangan teknologi dengan tujuan pengawasan dan keamanan sudah semakin marak dan semakin diperlukan fitur-fitur penting untuk produk-produk pengawasan tersebut. Salah satu perkembangan teknologi yang banyak diminati yaitu pengenalan wajah. Pengenalan wajah merupakan suatu kunci penting dalam teknologi pengawasan dan keamanan. fakta ini terbukti dari melimpahnya ponsel pintar di pasaran yang menggunakan pendeteksian wajah demi mengakses telepon genggam tersebut. Kami bertujuan untuk mengimplementasikan pengenalan wajah dan juga perhitungan banyaknya orang dalam suatu kumpulan orang kepada suatu *drone*. Pada masa ini, pengawasan masih banyak dilakukan oleh pihak keamanan yang mengecek lingkungan secara manual dengan menghampiri orang-orang dan mengecek sekumpulan orang-orang tersebut.

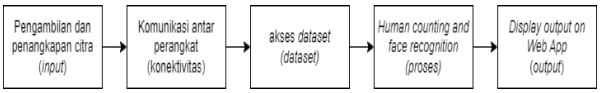
Teknologi PUTA atau *UAV* atau biasa disebut *drone* adalah pesawat yang dapat oleh seseorang. *Drone* dapat dikendalikan seyojana dengan menggunakan remot yang sudah terkoneksi dengan *drone*. *Drone* yang dapat dikendalikan dari jarak jauh dapat digunakan untuk mengakses wilayah yang sulit diakses oleh manusia [1]. Pendeteksian wajah tersebut dapat digunakan untuk mendeteksi seseorang yang sudah memasuki suatu area atau wilayah terlarang. Hal ini dapat digunakan untuk memberikan kepastian akan keamanan tanpa harus ada seseorang yang mendeteksi wajah tersebut secara dekat. Untuk fungsi kedua *drone* tersebut dapat menghitung banyaknya jumlah orang-orang yang berada pada suatu *crowd* atau disebut juga *crowd counting*. Hal ini beralasan lantaran untuk dapat menghitung suatu kumpulan orang dapat memakan waktu yang lambat jika dilakukan secara manual dan dapat memungkinkan *error* terjadi seperti terhitungnya orang yang sama secara lebih dari satu kali.

Mengaplikasikan sebuah sistem pengenalan wajah, pengenalan bentuk manusia, penghitungan jumlah manusia menggunakan *drone* merupakan tahap awal. Masalahnya adalah pembuatan sistem-sistem tersebut agar menjadi sistem yang efektif dari kamera *drone* yang berada di atas orang-orang. Pendeteksian wajah akan lebih sulit dikarenakan *angle* kamera dan juga orang-orang yang selalu bergerak secara konstan [2]. Pendeteksian wajah dan juga bentuk manusia juga mendapat sebuah halangan dikarenakan *drone* selalu berada di udara dan juga *drone* akan secara konstan mendeteksi bentuk orang-orang yang bergerak.

II. KAJIAN TEORI

Berikut sekumpulan konsep sistem yang digunakan dalam pengembangan *system Human Counting and Face Recognition Using Drone*. Pustaka-pustaka ini digunakan sebagai pedoman pada penelitian ini untuk mengembangkan sebuah sistem.

A. Sistematika *Human Counting* and *Face Recognition using Drone*

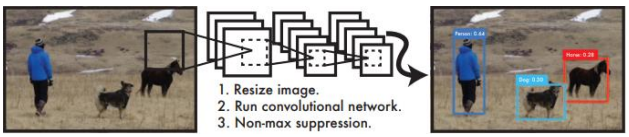


GAMBAR 1.
Diagram Human Counting and Face Recognition using Drone

Gambar 1 di atas mempengaruhi satu sama lain Drone digunakan sebagai penangkap citra yang kemudian dikirim kepada sebuah *PC* atau laptop. Citra tersebut akan menjadi input dengan didapatkan dari citra yang *di-broadcast* oleh *drone* dan *di-capture* oleh sistem yang akan diproses selanjutnya menggunakan algoritma *YOLO*. Sistem kemudian mengakses *dataset-dataset* yang ada pada *database* untuk melakukan proses selanjutnya. Proses selanjutnya merupakan proses utama untuk penelitian dengan judul ini yaitu proses pengenalan wajah dan perhitungan manusia. Setelah mengakses *database* sebelumnya, sistem dapat mengenali bentuk manusia dan wajah dengan algoritma *object detection* dengan membandingkan bentuk yang ada pada citra yang ditangkap dengan data-data yang ada pada dataset. Pengenalan wajah juga dilakukan dengan *library OpenCV* yang mana membandingkan piksel dan nilai biner yang didapat oleh sistem berdasarkan citra yang ditangkap dengan citra yang ada pada dataset. Terakhir, sistem akan menampilkan berapa banyak manusia terhitung dan juga identitas wajah yang dikenali oleh sistem.

B. *You Only Look Once (YOLO)*

YOLO(You Only Look Once) Kemampuan visual manusia begitu gesit dan tepat sehingga memungkinkan kita melaksanakan tugas-tugas kompleks seperti mengidentifikasi pengemudi yang mengantuk. Algoritma yang cekatan dan akurat dalam deteksi objek bisa membuka pintu bagi mobil otonom tanpa perlu sensor khusus. Ini akan memungkinkan perangkat bantu untuk memberikan informasi adegan secara real-time kepada pengguna manusia, dan juga menggagas potensi sistem robotik serbaguna yang tanggap. Metode deteksi yang dipakai saat ini menggunakan pengklasifikasi untuk melaksanakan deteksi tersebut. Agar bisa mengenali suatu objek, metode ini mengambil pengklasifikasi yang sesuai untuk objek tersebut, lalu mengevaluasinya di beragam posisi dan ukuran di dalam gambar uji [5].

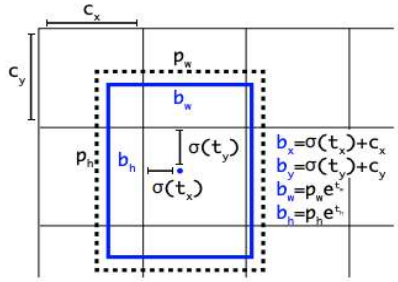


GAMBAR 2
The YOLO Detection System.

Menciptakan kerangka potensial yang melingkupi objek di dalam gambar, lalu melanjutkan dengan menjalankan pengklasifikasi pada area yang diidentifikasi tersebut. Setelah proses klasifikasi, langkah selanjutnya adalah mengolah hasilnya dengan merapihkan kotak pembatas, menghapus deteksi ganda, dan menyesuaikan posisi kotak berdasarkan objek lain yang ada. Rangkaian langkah yang rumit ini sering kali terbukti lambat dan sulit dioptimalkan, mengingat setiap komponennya harus dilatih secara terpisah. *YOLO (You Only Look Once)* mengubah pendekatan deteksi objek dengan merangkumnya sebagai permasalahan regresi tunggal. Hal ini berarti, dari setiap piksel dalam gambar, *model* secara langsung memprediksi koordinat kotak pembatas dan juga probabilitas kelas objek yang mungkin ada.

C. *Anchor Boxes*

Penggunaan *anchor boxes* adalah salah satu teknik yang umum digunakan dalam deteksi objek menggunakan metode seperti *YOLO (You Only Look Once*) dan *SSD* (*Single Shot Multibox Detector*). *Anchor boxes* adalah sejumlah kotak pembatas (*bounding boxes*) yang didefinisikan sebelumnya dengan ukuran dan rasio tertentu. Mereka diposisikan secara strategis di sepanjang gambar dan berfungsi sebagai titik referensi untuk mengidentifikasi objek. Saat algoritma melakukan prediksi, setiap *anchor box* akan mencoba untuk "menangkap" atau "mengelilingi" objek yang hadir dalam gambar [5]. Gambaran mengenai *Anchor Boxes* Hal ini terlihat dalam ilustrasi yang tertera pada Gambar 2 di bawah ini.



GAMBAR 3.
Anchor Boxes

D. Intersection over Union (IoU)

Dalam deteksi objek, *IoU* (*Intersection over Union*) merupakan parameter yang digunakan untuk mengukur sejauh mana *bounding box* yang diprediksi oleh model cocok dengan *bounding box* yang sebenarnya dari objek tersebut. *IoU* dapat dihitung dengan membandingkan area dari kedua *bounding box* yang terlibat.

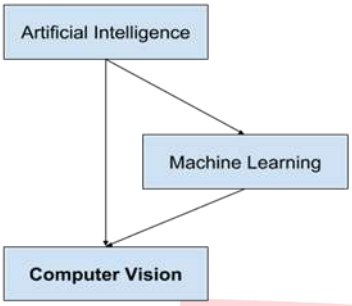
$$IoU = \frac{Predictions}{ground\ truth}$$

Untuk menghitung *IoU*, kita perlu melihat bagian yang beririsan antara kotak pembatas yang diprediksi dan kotak pembatas yang sebenarnya. Kemudian, kita membagi area irisan ini dengan gabungan dari kedua *bounding box* tersebut. Semakin tinggi nilai *IoU*, semakin baik deteksi objeknya, karena *IoU* mendekati nilai 1 menunjukkan keselarasan yang sempurna antara *bounding box* prediksi dan sebenarnya [6].

E. Open CV

OpenCV (OpenSource Computer Vision Library) adalah sebuah perangkat lunak yang dirancang untuk melakukan pengolahan citra digital secara real-time. *OpenCV* merupakan bagian dari cabang Kecerdasan Buatan *(Artificial*

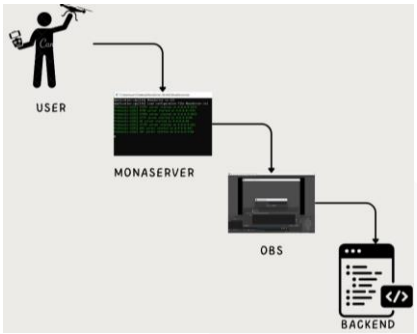
Intelligence) yang digunakan untuk mengembangkan dan menganalisis isi dari gambar. Cara kerja *OpenCV* mencerminkan prinsip kerja sistem visual manusia, di mana objek dilihat melalui "mata" atau penglihatan, dan citra objek tersebut kemudian diproses oleh otak.



GAMBAR 4.
Tinjauan Kecerdasan Buatan dan Visi Komputer

III. RANCANGAN SISTEM

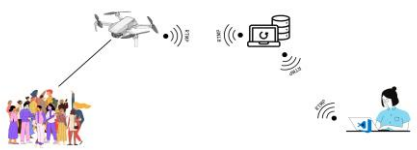
A. Koneksi Drone dengan Perangkat Keras



GAMBAR 5.
Ilustrasi Koneksi Drone dengan Perangkat Keras

Gambar 5 di atas merupakan ilustrasi koneksi *drone* dengan perangkat keras dengan urutan yaitu User-*MonaServer-OBS-Backend*. *MonaServer* adalah *server* multimedia fleksibel dengan dukungan protokol seperti *RTMFP, RTMP, RTMPE, WebSocket*, dan *HTTP*. Digunakan sebagai pusat koneksi *RTMP* untuk menerima *stream* dari *OBS (Open Broadcaster Software),* perangkat lunak sumber terbuka untuk merekam dan *streaming* video. *MonaServer* berperan sebagai server *RTMP* untuk menerima *stream* dari *drone* melalui *OBS* dan menyiarkannya ke aplikasi *web*. Konfigurasi *MonaServer* perlu disesuaikan dengan alamat *IP* dan *port* yang sesuai. *OBS* digunakan untuk menangkap, memproses, dan mengirim *video* dari *drone* ke aplikasi *web*. Semua elemen dari *OBS*, termasuk *overlay* dan grafis, akan ditampilkan dalam aplikasi *web*.

B. Arsitektur Utama Sistem



GAMBAR 6.
Ilustrasi Fungsi Sistem

Sistem ini seperti namanya merupakan sebuah sistem yang mengintegrasikan *drone* pada sistem yang dapat mendeteksi manusia, menghitung manusia, mendeteksi wajah, dan mengenali wajah. Sistem ini dirancang dengan menggunakan dua buah hardware, yaitu *PC* atau laptop dan juga *drone*. Sistem ini dirancang dengan tujuan utamanya adalah mempermudah *surveillance*. Sistem ini sudah dipertimbangkan sebagai salah satu cara pengawasan area yang lebih efisien, dengan drone yang selalu mengudara dan dengan konstan mengawasi, menghitung, dan mendeteksi wajah dari udara [2].

C. Pengujian *Device*

Pengujian dilakukan untuk mengetahui portabilitas perangkat komputasi sebagai alat untuk menjalankan algoritma deteksi hanya dari basis pembawa komputer di lokasi lalu lintas. Jelas, dari segi mobilitas, laptop lebih unggul karena laptop memiliki baterai untuk bertahan dalam kondisi *non stop* kontak/soket.

D. Pengujian *Drone*

Pengujian *drone* dilakukan dengan membandingkan spesifikasi teknis *drone*, yaitu masa pakai baterai, resolusi kamera, dan karakteristik *drone*. Ini dapat memengaruhi durasi pencarian individual dan juga memperlambat pencarian. Resolusi penting karena sifat *drone* yang terus bergerak dan karenanya memerlukan resolusi tinggi untuk *FPS* apa pun. *Drone* harus memiliki fungsi *GPS* untuk menemukan koordinat objek yang Anda cari, karena jika Anda tidak memiliki *GPS* bawaan, Anda memerlukan perangkat *GPS* eksternal yang terhubung ke *drone*.

IV. IMPLEMENTASI

Dalam proses implementasi sistem Penghitung manusia dan Pengenalan wajah menggunakan drone, ada beberapa tahapan yang memerlukan perhatian, yakni:

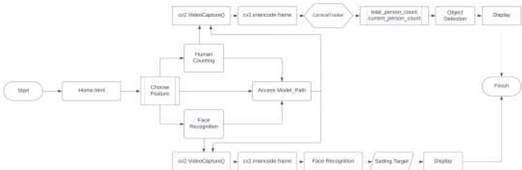
A. Konektivitas Drone ke PC

Bagian ini akan menjelaskan langkah-langkah untuk menghubungkan *drone* ke komputer *(PC)* sehingga *drone* dapat digunakan untuk mendapatkan gambar atau *video* yang akan diolah oleh *sistem* deteksi objek. Konektivitas bisa dilakukan melalui kabel atau protokol nirkabel seperti *Wi-Fi* atau *Bluetooth.*

B. Integrasi Sistem

Sistem akan diintegrasikan dengan menggabungkan komponen pendeteksian objek dengan *drone* dan memastikan semua bagian berfungsi secara sinergis. Dalam hal ini, drone akan digunakan sebagai perangkat akuisisi gambar yang akan diproses oleh *model* deteksi objek pada komputer. Keseluruhan implementasi ini akan menciptakan sistem pendeteksian objek yang dapat dijalankan pada komputer, menerima input gambar atau video dari drone, dan menghasilkan output yang menunjukkan objek yang terdeteksi dalam gambar atau video tersebut.

C. Implementasi Sistem Web



GAMBAR 7.
Flowchart Sistem WebApp

Gambar tersebut menjelaskan mulai dengan menginisialisasi *home.html* kemudian memilih pilihan *window* untuk memilih fitur *human counting* atau *face recognition*, aplikasi berjalan sesuai pilihan fitur jika memilih *human counting* sistem menginisiasi *centroid tracker* untuk *object detection* dan jika memilih *face recognition* sistem menginisiasi setting target untuk mengenali *model* kemudian program berjalan secara *real-time* menampilkan *output* dari sebuah *drone* dengan tampilan sistem pendeteksi dan rekognisi.

Penjelasan Source Code:

```
@app.route("/")
def home():
    return render_template('home.html')
```

GAMBAR 8.
Source Code Webapp (1)

Ini adalah route utama pada aplikasi *Flask.* Ketika pengguna mengakses halaman *root* ("/") dari aplikasi *web,* fungsi *home()* akan dijalankan. Fungsi ini akan merender *template "home.html"* dan mengirimkan halaman *HTML* ke browser pengguna.

```
<!-- People Counter -->
<div class="h-full relative" onclick="redirectToPeopleCounter()">
  <div
    class="absolute bg-black/50 w-full h-full hover:bg-white/50 transition duration-500 ease-in-out cursor-pointer hover:backdrop-blur-sm"
  >
    <div
      class="h-screen text-white flex justify-center items-center hover:text-black"
    >
      <h1 class="mx-auto my-auto font-bold text-4xl">PEOPLE COUNTER</h1>
    </div>
  </div>
  <img
    src="https://images.unsplash.com/photo-1513682121497-80211f36a7d3?ixlib=rb-4.0.3&ixid=...
    alt=""
    class="w-full h-full object-cover hover:filter-blur-md cursor-pointer"
```

GAMBAR 9.
Source Code WebApp (2)

Kode di atas adalah bagian dari aplikas*i web* yang menggunakan *framework Flask* untuk mengimplementasikan fitur pengenalan wajah *("FACE RECOGNITION").* Fungsi ini *diakses* melalui rute */detect_class/<selected_class>*, di mana *<selected_class>* adalah variabel yang menentukan kelas orang yang akan dikenali.

```
@app.route("/webcam_feed")
def webcam_feed():
    cap = cv2.VideoCapture(0)
    cap.set(cv2.CAP_PROP_FRAME_WIDTH, 1280)  # Set the desired width
    cap.set(cv2.CAP_PROP_FRAME_HEIGHT, 720)  # Set the desired height
```

GAMBAR 10.
Source Code WebApp(3)

Ketika *endpoint /webcam_feed* diakses, fungsi *webcam_feed()* akan dijalankan. Pada awal fungsi, sebuah objek *video capture* dibuat menggunakan *library OpenCV* dengan mengatur kamera menggunakan indeks 0, yang berarti menggunakan kamera utama pada perangkat. Selanjutnya, resolusi *frame* dari *webcam* diatur menjadi *1280x720* piksel untuk menghasilkan gambar berkualitas tinggi.

```
if __name__ == "__main__":
    parser = argparse.ArgumentParser(
        description="Flask app exposing YOLOv7 models")
    parser.add_argument("--port", default=8000, type=int, help="port number")
    args = parser.parse_args()
    model = torch.hub.load('.', 'custom', model_path, source='local').eval()

    logger.info("Starting the application...")
    app.run(host="0.0.0.0", port=args.port)
```

GAMBAR 11.
Source Code Webapp(4)

Skrip mempersiapkan argumen baris perintah menggunakan *modul argparse*, yang memungkinkan pengguna untuk menentukan nomor *port* pada saat menjalankan aplikasi. Jika argumen *port* tidak diberikan, nilai *default* yang akan digunakan adalah 8000. Skrip memuat model *YOLOv7* yang telah di-*custom* sebelumnya menggunakan *PyTorch Hub* dari direktori lokal dengan menggunakan *path* yang telah ditentukan melalui *model_path*. Setelah model berhasil dimuat, model tersebut diatur dalam *mode* evaluasi dengan menggunakan fungsi *.eval()*, sehingga siap untuk digunakan dalam tahap inferensi.

Setelah itu, pesan informasi *"Starting the application..."* ditampilkan menggunakan *logger*, yang merupakan mekanisme untuk mencatat informasi dalam aplikasi. Aplikasi *Flask* dijalankan dengan menggunakan *app.run()*. Aplikasi akan berjalan pada alamat *IP* "0.0.0.0", yang berarti dapat menerima permintaan dari semua alamat *IP* yang terhubung ke perangkat, dan *port* yang telah ditentukan melalui argumen baris perintah atau nilai default *(8000).*

Langkah *Train Dataset YOLO:*

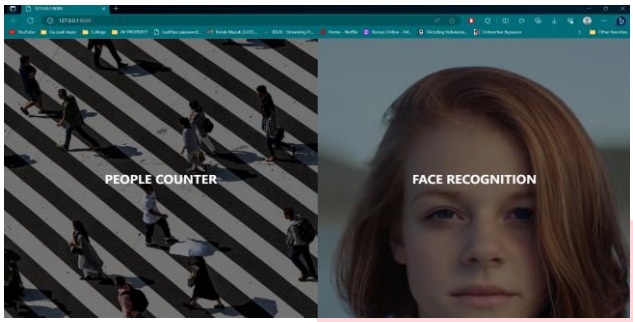
1. *Darknet*:
   Unduh kode sumber *Darknet* dari repositori resmi atau cabang yang sesuai. *Darknet* adalah kerangka kerja yang umumnya digunakan untuk pelatihan model *YOLO*.
2. Konfigurasi:
   Pindah ke direktori *Darknet* dan atur file konfigurasi yang sesuai dengan versi *YOLO* yang Anda gunakan. *File* konfigurasi ini mengandung informasi tentang arsitektur *model*, jumlah kelas, path untuk dataset, dll.
3. *Dataset*:
   Siapkan *dataset* pelatihan dan validasi Anda. *Dataset* ini harus berisi gambar-gambar dengan anotasi *bounding box* yang sesuai.
4. *File Label*:
   Buat file yang berisi informasi label kelas untuk dataset Anda. Setiap baris harus mencantumkan nama kelas dalam urutan yang sesuai.
5. Konversi *Dataset*:
   Gunakan alat yang disediakan oleh *Darknet* untuk mengkonversi *dataset* Anda menjadi format yang sesuai dengan *Darknet*. Biasanya, setiap gambar akan memiliki file teks yang berisi informasi tentang objek-objek dalam gambar.
6. Inisialisasi Pelatihan:
   Jalankan perintah di terminal untuk memulai pelatihan. Perintah ini biasanya akan mengacu pada file konfigurasi, bobot pra-pelatihan (opsional), dan *dataset* yang sudah diformat Contoh:

"Perintah './darknet detector train path/to/your/data.data path/to/your/yolov7.cfg path/to/your/pretrained.weights'"

7. Evaluasi Hasil:

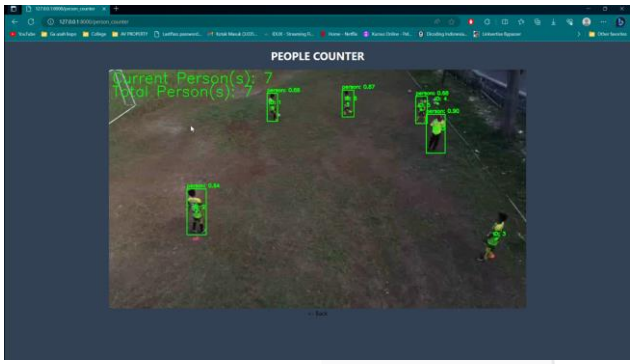
Setelah pelatihan selesai (jumlah *epoch* yang ditentukan telah tercapai), Anda dapat mengevaluasi model dengan menggunakan *dataset* validasi. Anda dapat mengukur metrik seperti *mean Average Precision (mAP)* untuk menilai performa deteksi.

V. HASIL DAN PEMBAHASAN



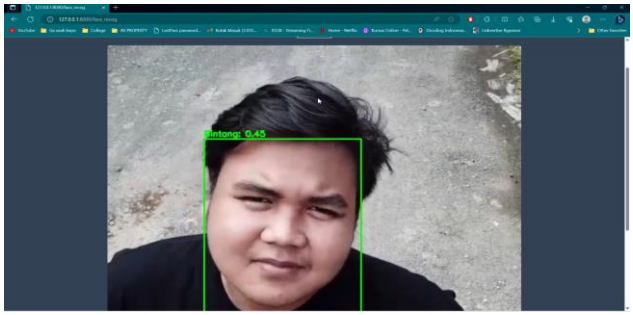
GAMBAR 12.
Tampilan *Homepage Website*

Gambar *home page* menu website berisikan 2 *route*, *route People Counter* untuk menghitung jumlah orang, *route Face Recognition* untuk mendeteksi wajah sesorang yang muncul pada kamera.



GAMBAR 13.
Tampilan *Menu People Counter*

Tampilan untuk *display output* pada *people counter* memperlihatkan hasil output yang terhubung dengan *program* dan juga *drone. Drone* sebagai sumber yang terhubung dengan *OBS* akan ditampilkan *POV* nya di dalam *page* ini. *Page* ini juga memperlihatkan *bounding box, confidence score* dan *ID* pada objek-objek yang terdeteksi berapa total orang yang terdeteksi dan juga berapa jumlah orang yang sedang berada di dalam *frame*.



GAMBAR 14.
Tampilan *Menu Face Recognition*

Tampilan untuk *display output* pada *face recognition* memperlihatkan hasil output yang terhubung dengan program dan juga *drone.* Sama seperti *page people counter, drone* sebagai sumber yang terhubung dengan *OBS* akan ditampilkan *POV* nya di dalam *page* ini. *Page* ini juga memperlihatkan *bounding box, confidence score* dan nama wajah pada wajah yang terdeteksi.

VI. KESIMPULAN

Implementasi dari perancangan sistem terdiri dari *Website Local Host, MonaServer, Open Broadcast Software (OBS),* dan *model training YOLO*. Setelah semua pengujian dilakukan dan dilaksanakan, dapat disarikan bahwa terdapat faktor-faktor tertentu yang memiliki potensi untuk mempengaruhi hasil pengujian, seperti berikut ini. *Device Capability,* Kapabilitas dari laptop atau *PC* yang digunakan oleh pengguna dapat mempengaruhi hasil output sistem. *Device* dengan *GPU*, *VRAM*, *CPU* dan *RAM* yang tinggi dapat mempercepat dan memperlancar proses deteksi. Sedangkan *device* dengan *specs* yang rendah akan kesulitan dalam memproses data. Jaringan / Koneksi Internet, Koneksi internet yang digunakan untuk menyambungkan *drone* dengan laptop dapat mempengaruhi hasil output sistem. Koneksi internet yang buruk dapat membuat citra yang ditangkap menjadi *blurry* dan ber-*mosaic* sehingga hasil output akan terpengaruh. Pergerakan Objek, Pergerakan objek dapat mempengaruhi hasil dari *output* sistem. Pada saat pengujian objek yang paling umum terdeteksi adalah objek-objek yang bergerak dibandingkan dengan objek yang diam. Intensitas Cahaya, Intensitas cahaya saat dilakukan pengujian baik dari lingkungan maupun dari *ISO drone* dapat mempengaruhi hasil deteksi.

REFERENSI

[1] Prof. Dr. Widodo Budiharto, Dr. Ir. Jarot S. Suroso, M.Eng., Dr. Ir. Alexander Agung Santoso Gunawan, M.Sc., IPM., Andry Chowanda, S.Kom, M.M., Ph.D.,Desain dan Pemrograman Drone Cerdas, 2021.

[2] Sivaranjith Sivaraman. (2021, October 27). Drones With Facial Recognition: Are They The Future Of Surveillance? https://blog.mantratec.com/drones-with-facial-recognition-are-they-the-future-ofsurveillance.

[3] Menteri Perhubungan Republik Indonesia. (2020). Peraturan Menteri Perhubungan Republik Indonesia Nomor PM 37 Tahun 2020 Tentang Pengoperasian Pesawat Udara Tanpa Awak di Ruang Udara yang Dilayani Indonesia.

[4] K.G. Shanti, P. Sivalakshmi, S. Sesha Vidhya, & K. Sangeetha Lakshmi (2021). Smart drone with real time face recognition. Materials Today: Proceedings. https://doi.org/10.1016/j.matpr.2021.07.214.

[5] Jacob Solawetz, F. (2023, January 11). *Anchor Free Detection*. https://blog.roboflow.com/whats-new-in-yolov8/#:~:text=GitHub%20user%20RangeKing-,Anchor%20Free%20Detection,-YOLOv8%20is%20an

[6] Q. Song et al., "Object detection method for grasping robot based on improved YOLOv5," Micromachines (Basel), vol. 12, no. 11, p. 1273, 2021