RANCANG BANGUN SISTEM PARKING SPACE COUNTER BERBASIS KAMERA WEBCAM MENGGUNAKAN OPENCV PYTHON

1st Haviz Rizka Pratama
Fakultas Teknik Elektro
Universitas Telkom Bandung,
Indonesia
havizrpratama@students.telkomuniversity.ac.id

2nd Dhoni Putra Setiawan, S.T., M.T.,Ph.D.
Fakultas Teknik Elektro
Universitas Telkom Bandung,
Indonesia
setiawandhoni@telkomuniversity.ac.id

3rd Yudiansyah, S.T., M.T.
Fakultas Teknik Elektro
Universitas Telkom Bandung,
Indonesia
yudiansyah@telkomuniversity.ac.id

Abstrak — Masalah Parkir merupakan salah satu tantangan yang sering terjadi di daerah perkotaan. Pertumbuhan kendaraan semakin padat dengan waktu yang tidak sebanding ketersediaan lahan parkir, sehingga muncul kemacetan, polusi, dan konsumsi bahan bakar berlebihan. Penelitian ini bertujuan untuk merancang dan membangun sistem *Parking Space Counter* berbasis kamera webcam dengan memanfaatkan OpenCV Python. Sistem ini diharapkan mampu mendeteksi ketersediaan tempat parkir secara otomatis dan memberikan informasi *real-time* kepada pengendara tentang slot parkir yang tersedia.Hasilnya menunjukkan presisi sebesar 67%, yang mengindikasikan bahwa sistem mengidentifikasi ruang parkir yang terisi dengan benar dalam 67% prediksi. Recall mencapai 100%, menunjukkan kemampuan model untuk mendeteksi semua ruang parkir yang terisi. Nilai F1 sebesar 80% mencerminkan kinerja yang seimbang antara presisi dan *recall*, yang menunjukkan keefektifan sistem. ini menunjukkan bahwa meskipun model ini dapat mengidentifikasi ruang parkir yang terisi dengan baik, masih ada ruang untuk perbaikan dalam mengurangi kesalahan positif untuk meningkatkan presisi lebih lanjut.

Kata kunci **- Parkir, Webcam, OpenCV, F1 Score.**

Abstract **-** Parking is one of the most common challenges in urban areas. The growth of vehicles is increasing at a rate that is not proportional to the availability of parking space, resulting in congestion, pollution, and excessive fuel consumption. This research aims to design and build a webcam camera-based Parking Space Counter system by utilizing OpenCV Python. This system is expected to be able to detect the availability of parking spaces automatically and provide real-time information to motorists about available parking slots. The results showed a precision of 67%, indicating that the system correctly identified occupied parking spaces in 67% of the predictions. Recall reached 100%, indicating the model's ability to detect all occupied parking spaces. The F1 value of 80% reflects a balanced performance between precision and recall, indicating the effectiveness of the system. This suggests that although the model can identify occupied parking spaces well, there is still room for improvement in reducing false positives to further increase precision.

Keywords **- Parking, Webcam, OpenCV, F1 Score.**

I. PENDAHULUAN

Masalah Parkir merupakan salah satu tantangan yang sering terjadi di daerah perkotaan. Pertumbuhan kendaraan semakin padat dengan waktu yang tidak sebanding ketersediaan lahan parkir, sehingga muncul kemacetan, polusi, dan konsumsi bahan bakar berlebihan. Sistem konvensional seringkali tidak efisien karena pengendara harus mencari lahan lagi secara manual. Hal ini berakibat penumpukan kendaraan yang membuat waktu yang terbuang dan menurunkan produktivitas.

Berdasarkan Penelitian dari Politeknik Negeri Banyuwangi sebelumnya mengenai Sistem Tracking dan Counting Kendaraan Berbasis YOLO. Tujuan dari penelitian ini adalah untuk mengidentifikasi dan menghitung jumlah mobil yang diparkir di area yang dipantau dengan menggabungkan identifikasi objek dan teknik pelacakan. Sistem ini dapat memetakan ketersediaan slot parkir dan memberikan data waktu nyata kepada pengelola parkir dan pengemudi dengan menggunakan kamera CCTV atau kamera lain yang ditempatkan di area parkir. Manfaat utama dari sistem ini adalah tingkat presisi yang tinggi dalam mendeteksi kendaraan, terlepas dari pergeseran pencahayaan atau penempatan kendaraan. Teknologi YOLO membuat sistem ini lebih terjangkau dan mudah diimplementasikan dengan memungkinkan pemantauan terus menerus tanpa memerlukan perangkat keras sensor tambahan. Dibandingkan dengan teknik alternatif seperti RFID atau sensor ultrasonik, teknologi ini juga memberikan solusi yang lebih mudah beradaptasi dan terukur [1].

Berdasarkan permasalahan tersebut, penelitian ini bertujuan untuk merancang dan membangun sistem *Parking Space Counter* berbasis kamera webcam dengan memanfaatkan OpenCV Python. Sistem ini diharapkan mampu mendeteksi ketersediaan tempat parkir secara otomatis dan memberikan informasi real-time kepada pengendara tentang slot parkir yang tersedia. Dengan demikian, sistem ini diharapkan dapat mengatasi permasalahan yang ada serta meningkatkan efisiensi pengelolaan parkir di berbagai fasilitas umum.

II. TINJAUAN PUSTAKA

A. Kamera *Webcam*

Webcam (*web-kam*), adalah istilah untuk jenis kamera yang beroperasi secara *real time* dan dapat mengakses gambar melalui *World Wide Web*, program pesan instan, dan aplikasi panggilan video. Webcam juga dapat didefinisikan sebagai kamera yang beroperasi pada laptop atau komputer desktop melalui port USB atau dengan jaringan komputer. [2]. Selain itu, webcam memiliki mikrofon dan kapasitas untuk merekam video, sehingga sangat membantu untuk pembelajaran jarak jauh dan konferensi video [3].

B. Tripod

Tripod adalah alat yang membantu menjaga bodi kamera tetap lurus dan stabil. apabila fotografer ingin menggunakan tripod, tentu saja mereka harus menyesuaikan kaki-kaki tripod supaya tripod rata dan sejajar. Namun yang pasti, apabila menjumpai permukaan yang tidak rata di mana tripod ditempatkan, fotografer akan mengalami lebih banyak kesulitan untuk menyesuaikan keseimbangan tripod [4].

C. Citra Digital

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

Gambar 1. Representasi bentuk citra digital

Citra digital adalah representasi visual dari dunia nyata dalam bentuk digital yang dapat dimengerti dan diproses oleh komputer. Citra terdiri dari elemen titik yang disebut piksel, yang disusun dalam baris dan kolom. Setiap piksel memiliki nilai numerik yang menggambarkan tingkat kecerahan atau warna pada lokasi tertentu dalam gambar [5]. Gambar 1 di atas dimana Matriks yang dinyatakan untuk citra digital adalah dengan matriks berukuran N (baris/tinggi) x M (kolom/lebar).

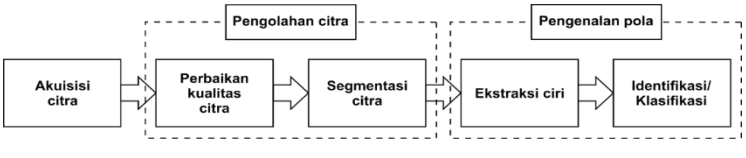
N = jumlah baris 0 = y = N – 1

M = jumlah kolom 0 = x = M – 1

L = maksimal warna intensitas 0 = f(x,y) = L – 1 (1)

(gray level/ derajat keabuan)

D. Pengolahan Citra Digital

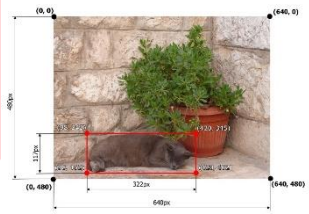


Gambar 2. Proses Pengolahan Citra Digital

Citra digital adalah sebuah piksel memiliki nilai organisasi (x,y) dan nilai kekuatan I(x,y). Gambar yang terkomputerisasi memiliki berbagai data yang berasal dari kekuatan cahaya yang diperoleh kisi-kisi sensor cahaya pada kamera. Gambar digital diambil oleh kamera dalam ruang warna RGB (Merah, Hijau, Biru), dengan 18 komponen intensitas untuk setiap piksel. Setiap kekuatan diatur dalam matriks biasa atau cluster 2 lapis [6].

Menurut Silvia Ratna (2020), Pengolahan Citra Digital (*Digital Image Processing*) adalah ilmu yang mempelajari teknik dalam mengolah citra, citra yang dimaksud adalah gambar diam (foto) atau *image* yang bergerak sedangkan makna arti digital adalah pengolahan citra/*image* dilakukan menggunakan komputer secara digital [7]. Masukan dari citra adalah citra dan keluarannya juga citra tapi dengan kualitas lebih baik dari pada citra input, misal citra warnanya kurang tajam, kabur (*blurring*), mengandung *noise* (misal bintik-bintik putih) [8].

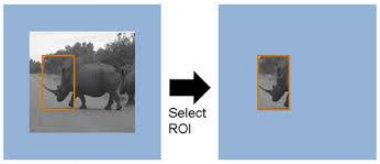
E. *BoundingBox*



Gambar 3. Konsep dasar BoundingBox

BoundingBox adalah salah satu konfigurasi dalam analisis Blob yang digunakan untuk menentukan koordinat x dan y dalam bentuk persegi panjang. Persegi panjang ini akan menutupi area terkecil yang mengelilingi objek Blob yang sedang dianalisis. Pengaturan ini berguna untuk menentukan koordinat yang dapat membatasi atau mengelilingi area target secara efisien [9].

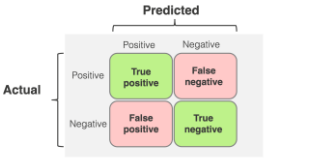
F. ROI (*Region Of Interest*)



Gambar 4. Teknik Region of Interest (ROI) pada citra

ROI (*Region of Interest*) yaitu menggambarkan bagian gambar yang telah dipilih untuk pemeriksaan tambahan. ROI sering digunakan dalam aplikasi pemrosesan gambar untuk mengisolasi bagian gambar yang signifikan, sehingga memungkinkan penyelidikan yang lebih menyeluruh pada area tersebut, sekaligus mengabaikan bagian lain yang tidak penting. Optimalisasi pemrosesan gambar dan konservasi sumber daya komputer bergantung pada pemanfaatan ROI [10].

G. *Confusion Matrix*



Gambar 5. Tabel Confusion Matrix

Confusion matrix digunakan untuk memvisualisasikan hasil evaluasi kinerja pengklasifikasi. Dalam berbagai tugas klasifikasi gambar, *confusion matrix* menunjukkan hasil dari setiap kelas dan membandingkannya dengan nilai sebenarnya. Proses ini tidak hanya menganalisis keakuratan klasifikasi tetapi juga menghitung metrik lainnya seperti *precision*, *recall*, dan *F1 score* [11].

Precision adalah rasio prediksi positif aktual terhadap keseluruhan prediksi positif. *Recall* adalah rasio positif yang benar di seluruh data positif. *F1 Score* adalah rasio presisi terhadap perolehan rata-rata tertimbang. *Confusion matrix* memberikan perbandingan antara hasil klasifikasi yang dicapai oleh model dan hasil klasifikasi aktual. *Accuracy* adalah rasio prediksi yang benar (positif dan negatif) di seluruh data [12].

Tabel 1. *Confusion Matrix*

Label	Definisi
True Positive (TP)	Jumlah positif yang dianggap positif
True Negative (TN)	Jumlah negatif yang dianggap negatif
False Positive (FP)	Jumlah positif yang dianggap negatif
False Negative (FN)	Jumlah positif yang dianggap positif

Rumus $Precision = \frac{TP}{TP+FP}$ (2)

Rumus $Recall = \frac{TP}{TP+FN}$ (3)

Rumus $F1\ Score = 2\ x\ \frac{Precision\ x\ Recall}{Precision+Recall}$ (4)

Rumus $Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$ x 100% (5)

H. OpenCV (*Open Source Computer Vision Library*)

Gambar 6. Logo OpenCV

Pustaka OpenCV (*Open Source Library for Computer Vision*) adalah Pustaka yang bersifat *open-source* sebagian besar digunakan untuk kegiatan yang berkaitan dengan visi komputer dan pembelajaran mesin. Pada awalnya, library ini dibuat dalam bahasa C++ dan mendukung sejumlah bahasa pemrograman, seperti Python, Java, dan MATLAB. Perpustakaan ini digunakan secara luas di berbagai bidang, termasuk augmented reality, pengenalan wajah, *image stitching*, dan deteksi objek secara *real-time*, serta menyediakan lebih dari 2.500 algoritma yang efisien [13].

I. Python

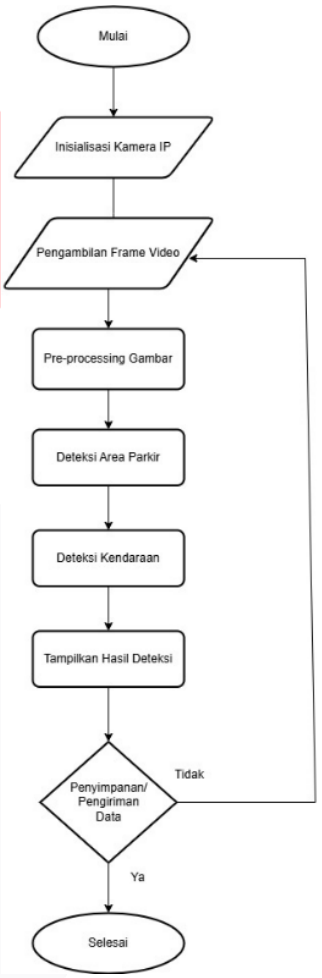
Gambar 7. Logo Bahasa Python

Python adalah bahasa pemrograman yang mudah digunakan dan cocok untuk pemrogram pemula dan berpengalaman karena penekanannya pada kode yang dapat dibaca. Python terkenal dengan sintaksnya yang mudah dan kompatibilitasnya dengan berbagai paradigma pemrograman, seperti pemrograman fungsional dan berorientasi objek. Aplikasi untuk bahasa ini sangat banyak dan mencakup penelitian data, kecerdasan buatan, dan pengembangan web [14].

III. METODOLOGI PENELITIAN

A. Flowchart Sistem

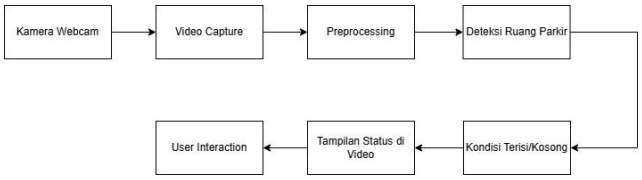
Flowchart sistem ditampilkan pada Gambar 7. Flowchart ini menggambarkan alur kerja dari aplikasi mulai dari inisialisasi kamera IP, pengambilan frame video, proses deteksi area parkir, hingga penampilan hasil dan penyimpanan data.



Gambar 7. Flowchart sistem

Pada tahap awal, dilakukan inisialisasi terhadap kamera Webcam yang akan digunakan untuk menangkap gambar atau video dari area parkir. Setelah kamera diinisialisasi, aplikasi akan mengambil frame video secara berkala. Frame-frame ini kemudian digunakan untuk mendeteksi area parkir dan kendaraan. Pada tahap selanjutnya, frame yang diambil akan melalui proses pra-pemrosesan gambar. Langkah-langkah preprocessing ini mencakup konversi ke grayscale, pengurangan noise, serta penajaman tepi gambar untuk mempermudah proses deteksi. Sistem mendeteksi area parkir yang telah ditentukan. Algoritma untuk deteksi area parkir diterapkan menggunakan OpenCV. Kendaraan yang parkir dalam area terdeteksi akan dikenali menggunakan metode deteksi objek. OpenCV dan teknik machine learning digunakan untuk mendeteksi kendaraan yang terparkir di dalam area yang telah diidentifikasi. Setelah kendaraan terdeteksi, hasil deteksi ditampilkan kepada pengguna melalui GUI yang menampilkan gambar beserta informasi area yang terisi atau kosong. Jika deteksi telah dilakukan, sistem akan menyimpan data hasil deteksi ke dalam database atau mengirimkannya ke server untuk analisis lebih lanjut.

B. Desain Sistem

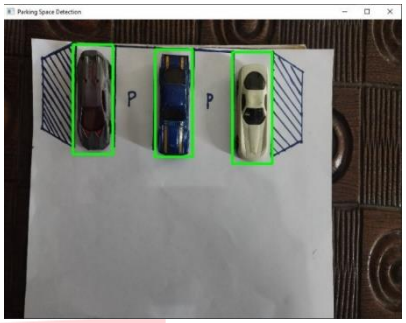


Gambar 8. Blok bagan desain sistem

Berikut adalah elemen-elemen yang dapat digambarkan dalam blok bagan desain sistem:

1. Input Kamera webcam
   - Menggunakan kamera untuk menangkap video feed dari area parkir.
   - USB webcam digunakan sebagai sumber aliran video.
2. *Video Capture*
   - Aliran video diterima oleh sistem melalui cv2.VideoCapture() dari OpenCV.
   - Proses pembacaan frame dari aliran video secara real-time.
3. *Preprocessing*
   - Setiap frame diubah ukurannya dengan cv2.resize() agar sesuai dengan dimensi yang diinginkan.
   - Proses konversi ke grayscale dan thresholding dilakukan untuk mendeteksi objek (mobil) di area parkir.
4. *Parking Space Detection*
   - Koordinat ruang parkir didefinisikan dalam kode sebagai area yang akan diperiksa.
   - Setiap area parkir diproses dan dicek apakah terisi berdasarkan jumlah piksel putih yang menunjukkan adanya objek (mobil).
5. *Occupancy Status (Logic)*
   - Berdasarkan jumlah piksel putih, sistem memutuskan apakah suatu ruang parkir kosong atau terisi.
   - Setiap area parkir akan diberikan indikator warna: hijau (kosong) atau merah (terisi).
6. *Output (Display & Status)*
   - Sistem menampilkan video dengan overlay persegi panjang yang menunjukkan ruang parkir dan statusnya.
   - Status "Kosong" atau "Terisi" dari setiap ruang parkir akan ditampilkan pada layar.
7. *User Interaction*
   - Sistem memungkinkan pengguna untuk menutup tampilan dengan menekan tombol tertentu (misalnya 'k').
8. *End Process*
   - Setelah proses selesai atau pengguna menghentikannya, aliran video akan dilepaskan dan jendela ditutup.

C. Proses Pengolahan Citra



Gambar 9. Hasil Bounding Box

Pengolahan citra dalam penelitian ini menggunakan beberapa teknik dasar dalam OpenCV, yaitu:

- *Grayscale Conversion*: Mengubah citra warna menjadi citra grayscale untuk memudahkan pengolahan lebih lanjut.
- *Thresholding*: Membantu dalam memisahkan objek dari latar belakang dengan menggunakan metode seperti *Otsu thresholding*.
- *Edge Detection*: Menggunakan algoritma seperti *Canny* untuk mendeteksi tepi dari kendaraan yang terparkir.
- *Bounding Box*: Untuk menandai kendaraan yang terdeteksi, bounding box ditampilkan di sekitar objek.

D. Pendefinisian Area Parkir

```
# Koordinat ruang parkir
parking_spaces = [
    [(123, 50), (235, 330)],  # Ruang Parkir Kiri
    [(267, 50), (385, 330)],  # Ruang Parkir Tengah
    [(415, 50), (540, 330)]   # Ruang Parkir Kanan
]
```

Gambar 10. Kode Python untuk mendefinisikan koordinat tiga ruang parkir

Kode di atas menunjukkan tiga ruang parkir yang telah diatur dalam koordinat citra video. Koordinat ini digunakan sebagai acuan untuk mendeteksi apakah ada kendaraan yang terparkir di area tersebut. Sistem mendeteksi area parkir berdasarkan koordinat yang telah ditetapkan sebelumnya. Setiap area parkir diwakili oleh dua titik koordinat (pojok kiri atas dan pojok kanan bawah) yang mendefinisikan batas area parkir.

IV. IMPLEMENTASI DAN PENGUJIAN

A. Tahapan Implementasi

Tahap 1 : Inisialisasi Kamera

```
import cv2

# Ganti dengan indeks kamera USB (biasanya 0 untuk webcam bawaan, 1 atau lebih untuk webcam eksternal)
usb_webcam_index = 0

# Memuatkan aliran video dari webcam USB
video = cv2.VideoCapture(usb_webcam_index)

# Lepaskan video dan tutup jendela
video.release()
cv2.destroyAllWindows()
```

Gambar 11. Tahapan Insialisasi Kamera

Aplikasi ini dimulai dengan menginisialisasi webcam yang akan digunakan untuk menangkap gambar area parkir secara real-time. OpenCV digunakan untuk membaca dan memproses setiap frame yang ditangkap kamera.

Tahap 2 : Menentukan Area Ruang Parkir

```
# Koordinat ruang parkir
parking_spaces = [
    [(140, 50), (220, 270)],  # Ruang Parkir Kiri
    [(305, 60), (380, 275)],  # Ruang Parkir Tengah
    [(460, 65), (540, 289)]   # Ruang Parkir Kanan
]
```

Gambar 12. Tahapan Area Ruang Parkir

Area yang merepresentasikan ruang parkir ditentukan dengan koordinat piksel. Tiga ruang parkir didefinisikan sebagai sebuah list dengan dua titik koordinat yang membentuk persegi panjang.

Tahap 3 : Deteksi Ruang Parkir Kosong atau Terisi

```
# Berfungsi untuk memeriksa apakah ruang parkir sudah terisi
def is_occupied(space, frame, threshold=130):
    x1, y1 = space[0]
    x2, y2 = space[1]

    # Memangkas area ruang parkir
    parking_area = frame[y1:y2, x1:x2]

    # Mengonversi area yang dipotong menjadi skala grayscale
    gray_parking_area = cv2.cvtColor(parking_area, cv2.COLOR_BGR2GRAY)

    # Menerapkan ambang batas biner untuk menyorot mobil
    _, binary_image = cv2.threshold(gray_parking_area, threshold, maxval: 255, cv2.THRESH_BINARY_INV)

    # Menghitung piksel putih (yaitu, mobil)
    white_pixel_count = np.sum(binary_image == 255)

    # Jika jumlah piksel putih melebihi jumlah tertentu, piksel tersebut terisi
    return white_pixel_count > 5000  # Adjust this value based on your image
```

Gambar 13. Tahapan Deteksi Ruang Parkir

Setiap area ruang parkir dipotong dari frame video, diubah ke gambar grayscale, lalu diterapkan metode *thresholding* untuk mendeteksi keberadaan objek (mobil). Jumlah piksel putih pada gambar hasil thresholding dihitung untuk menentukan apakah ruang parkir tersebut terisi atau kosong.

Tahap 4 : Menghitung Ruang Kosong dan Terisi

```
    # Menampilkan status pada frame
    status_text = f"Terisi: {occupied_count} | Kosong: {empty_count}"
    cv2.putText(frame, status_text, org: (10, 30), cv2.FONT_HERSHEY_SIMPLEX, fontScale: 1, color: (255, 255, 255), thickness: 2)

    # Menampilkan frame dengan persegi panjang dan status
    cv2.imshow( winname: 'Parking Space Detection', frame)

    # Keluar dari perulangan jika tombol 'k' ditekan
    if cv2.waitKey(12) & 0xFF == ord('k'):
        break
```

Gambar 14. Tahapan Menghitung Ruang

Aplikasi akan menghitung jumlah ruang parkir yang terisi dan kosong. Setiap ruang parkir ditampilkan dengan kotak berwarna hijau (terisi) atau merah (kosong). Informasi status juga ditampilkan di layar video.

Tahap 5 : Menampilkan Hasil Deteksi

```
    # Keliling setiap ruang parkir dan periksa apakah sudah terisi atau belum.
    for space in parking_spaces:
        if is_occupied(space, frame):
            color = (0, 0, 255)  # Merah untuk terisi
            occupied_count += 1
        else:
            color = (0, 255, 0)  # hijau untuk kosong
            empty_count += 1
        cv2.rectangle(frame, space[0], space[1], color, thickness: 3)
```

Gambar 15. Tahapan hasil deteksi

Video hasil deteksi ditampilkan menggunakan OpenCV dengan informasi jumlah ruang parkir kosong dan terisi pada setiap frame. Jika pengguna menekan tombol 'k', aplikasi akan berhenti.

B. Skenario Pengujian Jarak Pandang Kamera Webcam

Pengujian ini bertujuan untuk menentukan jarak optimal bagi kamera untuk mendeteksi mobil secara akurat. Pengujian dilakukan dengan mengukur jarak antara kamera dan objek (mobil) serta mencatat hasil deteksi pada setiap jarak. Dalam eksperimen ini, digunakan tiga model mobil yang diletakkan pada posisi tetap dengan variasi jarak antara kamera dan mobil. Jarak yang diuji berkisar antara 7 cm hingga 22 cm. Pada setiap jarak sistem mendeteksi apakah mobil terdeteksi atau tidak.

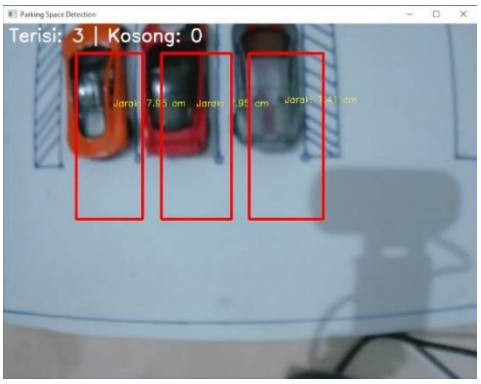
.

Pengukuran dilakukan secara bertahap dengan mencatat hasil deteksi sistem terhadap mobil-mobil tersebut. Data yang diperoleh kemudian dianalisis untuk mengetahui jarak maksimal di mana sistem masih dapat mendeteksi mobil secara konsisten. Berikut Tabel di bawah ini menunjukkan hasil deteksi mobil pada berbagai jarak:

Tabel 2. Hasil Pengujian Jarak pandang kamera ke objek

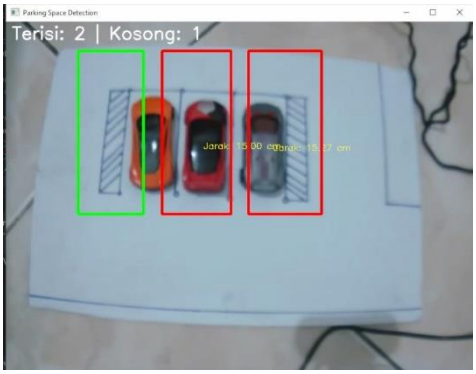
Jarak (cm)	Hasil		
	Mobil 1	Mobil 2	Mobil 3
7 cm	Terdeteksi	Terdeteksi	Terdeteksi
8 cm	Terdeteksi	Terdeteksi	Terdeteksi
9 cm	Terdeteksi	Terdeteksi	Terdeteksi
10 cm	Terdeteksi	Terdeteksi	Terdeteksi
11 cm	Terdeteksi	Terdeteksi	Terdeteksi
12 cm	Terdeteksi	Terdeteksi	Terdeteksi
13 cm	Terdeteksi	Terdeteksi	Terdeteksi
14 cm	Terdeteksi	Terdeteksi	Terdeteksi
15 cm	Tidak Terdeteksi	Terdeteksi	Terdeteksi
16 cm	Tidak Terdeteksi	Terdeteksi	Terdeteksi
17 cm	Tidak Terdeteksi	Terdeteksi	Terdeteksi
18 cm	Tidak Terdeteksi	Terdeteksi	Terdeteksi
19 cm	Tidak Terdeteksi	Terdeteksi	Terdeteksi
20 cm	Tidak Terdeteksi	Terdeteksi	Tidak Terdeteksi
21 cm	Tidak Terdeteksi	Terdeteksi	Tidak Terdeteksi
22 cm	Tidak Terdeteksi	Tidak Terdeteksi	Tidak Terdeteksi

Berdasarkan hasil pengujian Jarak 7 cm hingga 14 cm, Semua kendaraan (Mobil 1, Mobil 2, dan Mobil 3) terdeteksi oleh kamera pada jarak ini. Ini menunjukkan bahwa kamera memiliki tingkat akurasi yang baik pada jarak dekat hingga menengah.



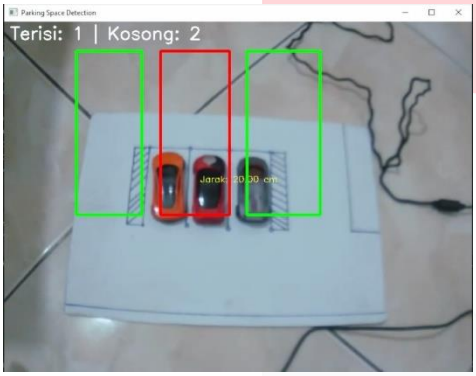
Gambar 16. Hasil pengujian jarak ketinggian 7 cm

Dari hasil jarak 15 cm hingga 19 cm, Mobil 1 mulai tidak terdeteksi, sedangkan Mobil 2 dan Mobil 3 masih terdeteksi hingga 19 cm. Hal ini menunjukkan adanya variasi dalam sensitivitas kamera terhadap ukuran atau posisi objek pada jarak menengah hingga jauh.



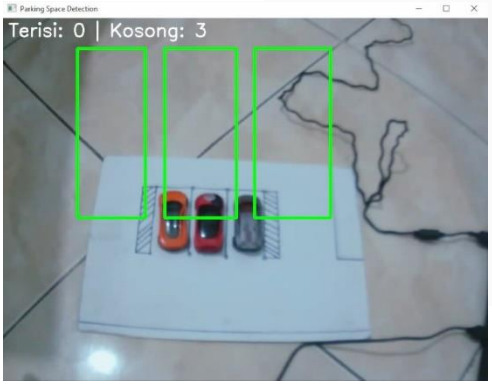
Gambar 17. Hasil pengujian jarak ketinggian 15 cm

Dari hasil jarak 20 cm hingga 21 cm, Mobil 1 dan Mobil 3 gagal, namun Mobil 2 masih terdeteksi hingga jarak 21 cm. Ini menunjukkan bahwa Mobil 2 mungkin memiliki karakteristik tertentu (misalnya ukuran atau warna) yang lebih mudah dikenali oleh kamera pada jarak lebih jauh.



Gambar 18. Hasil pengujian jarak ketinggian 20 cm

Pada jarak 22 cm, semua mobil tidak terdeteksi, yang menandakan batas maksimum jarak deteksi kamera dalam kondisi pengujian ini.



Gambar 19 Hasil pengujian jarak ketinggian 22 cm

Pengujian ini dapat menunjukkan batas efektif jarak pandang kamera webcam, dan dari data ini bisa menyimpulkan bahwa jarak maksimal untuk mendeteksi mobil secara andal adalah antara 14 hingga 21 cm, tergantung pada kondisi objek. Hasil ini penting untuk menetapkan parameter optimal penggunaan kamera dalam aplikasi yang dirancang.

C. Skenario Pengujian Sudut Rotasi Kamera Webcam

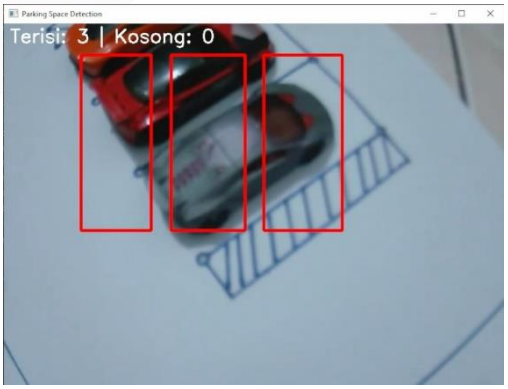
Pengujian dilakukan untuk mengevaluasi kemampuan deteksi objek (Mobil 1, Mobil 2, dan Mobil 3) pada berbagai sudut rotasi kamera menggunakan OpenCV. Tujuan pengujian ini adalah untuk mengetahui sejauh mana kamera dapat mendeteksi kendaraan dengan perubahan sudut pandang, dari 0° hingga 150°. Berikut tabel menunjukkan hasil deteksi kendaraan pada berbagai sudut rotasi kamera:

Tabel 3. Hasil Pengujian Sudut Rotasi pada kamera ke objek

Derajat (°)	Hasil		
	Mobil 1	**Mobil 2**	**Mobil 3**
0°	Terdeteksi	Terdeteksi	Terdeteksi
30°	Terdeteksi	Terdeteksi	Terdeteksi
45°	Terdeteksi	Terdeteksi	Terdeteksi
60°	Tidak Terdeteksi	Terdeteksi	Terdeteksi
90°	Tidak Terdeteksi	Tidak Terdeteksi	Terdeteksi
120°	Tidak Terdeteksi	Tidak Terdeteksi	Terdeteksi
135°	Tidak Terdeteksi	Tidak Terdeteksi	Terdeteksi
150°	Tidak Terdeteksi	Tidak Terdeteksi	Tidak Terdeteksi

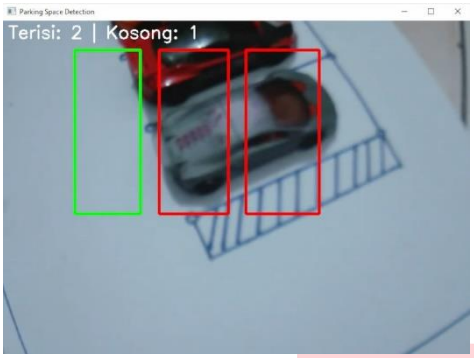
Berdasarkan hasil pengujian Sudut 0° hingga 45°, pada sudut ini kamera mampu mendeteksi semua mobil. Ini menandakan bahwa kamera bekerja dengan baik pada sudut lurus hingga miring moderat (0° hingga 45°), di mana pandangan kamera terhadap kendaraan relatif tidak terhalang.

Gambar 20. Hasil pengujian sudut rotasi 0 derajat



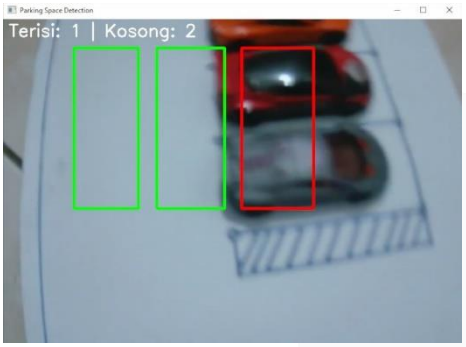
Gambar 21. Hasil pengujian sudut rotasi 45 derajat

Dari hasil pengujian Sudut 60°, Pada sudut ini deteksi Mobil 1 mulai gagal sedangkan Mobil 2 dan Mobil 3 masih terdeteksi. Ini menunjukkan bahwa rotasi 60° memengaruhi visibilitas Mobil 1, mungkin karena posisi atau orientasi objek tersebut yang berubah terhadap sudut kamera.



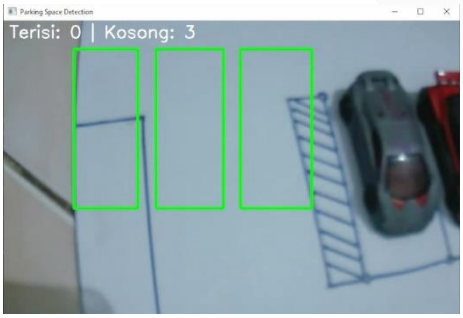
Gambar 22. Hasil pengujian sudut rotasi 60 derajat

Pada sudut Sudut 90°, hanya Mobil 3 yang terdeteksi sementara Mobil 1 dan Mobil 2 tidak terdeteksi. Artinya, saat kamera berotasi 90°, hanya objek tertentu yang dapat dideteksi dengan baik, mungkin karena perbedaan ukuran atau fitur visual kendaraan.



Gambar 23. Hasil pengujian sudut rotasi 90 derajat

Pada sudut rotasi Sudut 150°, tidak ada kendaraan yang berhasil terdeteksi. Ini menunjukkan bahwa pada sudut rotasi 150°, semua kendaraan keluar dari jangkauan efektif deteksi kamera.

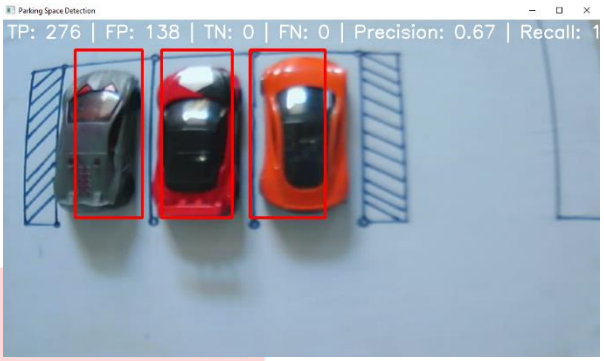


Gambar 24. Hasil pengujian sudut rotasi 150 derajat

Dari hasil ini, dapat disimpulkan bahwa sudut optimal deteksi kamera berada pada rentang 0° hingga 45°. Di luar sudut tersebut, kemampuan deteksi mulai menurun secara signifikan, dengan deteksi terhenti sepenuhnya pada 150°. Hal ini berguna untuk menentukan batas rotasi yang ideal dalam penerapan kamera webcam berbasis OpenCV untuk pengawasan atau penghitungan parkir.

D. Pengujian Akurasi dan Evaluasi Kinerja Model

Pengujian dilakukan dengan mengevaluasi metrik True Positives (TP), False Positives (FP), True Negatives (TN), False Negatives (FN), serta menghitung Precision dan Recall untuk mendapatkan gambaran kinerja model secara keseluruhan.



Gambar 25. Hasil pengujian akurasi model

Pada gambar 4.17, ditampilkan hasil deteksi ruang parkir. Model berhasil mendeteksi kendaraan yang terparkir dengan menampilkan bounding box berwarna merah di sekitar kendaraan. Pengujian ini dilakukan dalam kondisi pencahayaan stabil dan dengan tiga kendaraan yang berada di area parkir.

- *True Positives* (TP): Jumlah kendaraan yang berhasil dideteksi sebagai terisi dalam ruang parkir.
- *False Positives* (FP): Jumlah ruang parkir yang terdeteksi sebagai terisi padahal kosong.
- *True Negatives* (TN): Jumlah ruang parkir yang benar-benar kosong dan dideteksi dengan benar.
- *False Negatives* (FN): Jumlah kendaraan yang seharusnya terdeteksi terisi namun tidak terdeteksi oleh model.

Evaluasi kinerja model ini dilakukan menggunakan metrik *Accuracy*, *Precision*, *Recall*, dan *F1 Score* sebagai indikator utama dengan ketinggian 10 cm:

1. Accuracy: Accuracy adalah metrik lain yang umum digunakan yang mengukur proporsi semua prediksi yang benar (baik positif maupun negatif) dari total prediksi. Namun, dalam tugas-tugas seperti deteksi objek, terutama dengan kelas yang tidak seimbang (misalnya, lebih banyak contoh negatif daripada positif), akurasi dapat menimbulkan kesalahan. Hasil pengujian akurasi ini adalah 67%, yang berarti sama dengan presisi, karena tidak ada negatif salah atau negatif benar.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} = \frac{276+0}{276+0+138+0} = \frac{276}{414} = 0.67 \quad (6)$$

2. Precision: Precision dihitung sebagai perbandingan antara True Positives dengan jumlah keseluruhan prediksi positif yang dihasilkan oleh model (True Positives + False Positives). Nilai precision untuk pengujian ini adalah 0.67, yang berarti sekitar 67% dari semua ruang parkir yang dideteksi sebagai terisi memang benar-benar terisi. Namun, ada 33% kesalahan deteksi, di mana ruang parkir yang kosong terdeteksi sebagai terisi.

$$Precision = \frac{TP}{TP+FP} = \frac{276}{276+138} = 0.66 \text{ atau } 60\% \quad (7)$$

3. Recall: Recall mengukur seberapa baik model dalam mendeteksi ruang parkir yang benar-benar terisi, dibandingkan dengan jumlah total ruang parkir yang terisi. Recall pada pengujian ini mencapai nilai 1.0, yang berarti model mampu mendeteksi seluruh kendaraan yang terparkir dengan benar, tanpa ada kesalahan.

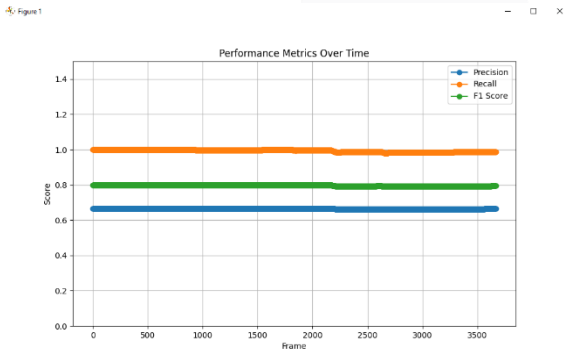
$$Recall = \frac{TP}{TP+FN} = \frac{276}{276+0} = 1.0 \quad (8)$$

4. F1 Score: F1 Score adalah metrik yang menggabungkan Precision dan Recall dalam satu nilai harmonik rata-rata, yang memberikan gambaran lebih menyeluruh tentang performa model. Pada pengujian ini, F1 Score dihitung sebagai berikut:

$$F1\ Score = 2\ x\ \frac{Precision\ x\ Recall}{Precision + Recall}$$

Dengan precision sebesar 0.67 dan recall 1.0, F1 Score untuk pengujian ini adalah:

$$F1\ Score = 2\ x\ \frac{0.67\ x\ 1.0}{0.67+1.0} = 0.8 \text{ atau } 80\% \quad (9)$$



Gambar 26. Hasil Performansi metrik dari waktu ke waktu

Berdasarkan hasil pengujian ini, terlihat bahwa model memiliki kinerja recall yang sangat baik, dengan nilai 1.0, yang menandakan bahwa semua ruang parkir yang terisi berhasil terdeteksi. Namun, nilai precision yang relatif rendah yaitu 0.67 menunjukkan adanya kesalahan deteksi, di mana ruang parkir kosong dideteksi sebagai terisi. Hal ini bisa disebabkan oleh beberapa faktor, antara lain:

- Kualitas gambar: Kondisi pencahayaan atau resolusi kamera mungkin mempengaruhi akurasi deteksi model, terutama dalam membedakan antara ruang parkir kosong dan terisi.
- Algoritma deteksi: Penggunaan algoritma yang lebih kompleks atau penambahan teknik filtering tambahan dapat meningkatkan precision.

Nilai F1 Score sebesar 0.8 menunjukkan bahwa meskipun recall model sangat baik, precision yang rendah sedikit menurunkan kinerja keseluruhan. Namun, F1 Score tetap menunjukkan performa yang cukup solid untuk sistem ini, dengan keseimbangan antara precision dan recall. Pada gambar yang ditampilkan, terlihat bahwa *bounding box* merah menunjukkan hasil deteksi oleh model. Meskipun recall sempurna, model masih mengalami kesalahan dalam mendeteksi beberapa ruang parkir kosong yang diduga terisi.

V. KESIMPULAN

Sistem yang dibangun berhasil menggunakan library OpenCV untuk mendeteksi dan menghitung jumlah tempat parkir yang tersedia dengan akurasi yang baik. Deteksi berbasis video real-time memungkinkan aplikasi untuk memantau situasi parkir secara terus-menerus dengan sumber video dari webcam yang terintegrasi. Sistem ini mampu bekerja dengan baik di lingkungan outdoor maupun indoor, selama kualitas video yang dihasilkan oleh webcam cukup jelas. Penerapan *thresholding* dan teknik deteksi kontur berhasil digunakan untuk mengidentifikasi area parkir yang kosong atau terisi. Sistem ini dapat diimplementasikan di tempat-tempat parkir publik seperti pusat perbelanjaan, kampus, atau kantor. Pengguna akan mendapatkan informasi *real-time* mengenai ketersediaan lahan parkir yang membantu manajemen parkir lebih efisien. Hasil pengujian menunjukkan bahwa model memiliki kemampuan yang baik dalam mendeteksi ruang parkir yang terisi, namun masih perlu perbaikan dalam mengurangi kesalahan deteksi terhadap ruang parkir yang kosong. Evaluasi kinerja model dengan *Accuracy*, *Precision*, *Recall*, dan *F1 Score* memberikan gambaran yang jelas mengenai aspek-aspek yang perlu ditingkatkan, terutama precision, untuk menghindari *False Positives* yang terlalu banyak. Beberapa peningkatan dapat dilakukan melalui optimasi algoritma atau penyesuaian parameter deteksi.

REFERENSI

[1] E. Ektrada, L. Hakim, and S. P. Kristanto, "Sistem Tracking dan Counting Kendaraan Berbasis YOLO untuk Pemetaan Slot Parkir Kendaraan," *SESSION (Software Development, Digital Business Intelligence, and Computer Engineering)*, vol. 01, no. 02, pp. 55-60, Maret 2023.

[2] J. Kristiadhy dan A. J. Gundo, "Perancangan Aplikasi Presensi Siswa Berbasis Website di SMK Negeri 1 Tengaran Menggunakan Webcam dan GPS Guna Mengurangi Risiko Penularan Virus COVID-19", *Jurnal Ilmiah Wahana Pendidikan*, Agustus 2022, 8 (12), 414-427.

[3] T. McClain, "The Role of Webcams in Remote Education and Work," Journal of Digital Communication, vol. 10, no. 2, pp. 15-20, Feb. 2024.

[4] A. C. Mualim, M. Yahya, dan D. A. Widhining K., "Rancang Bangun Keseimbangan Otomatis Tripod Dengan Sensor Gyroscope", *JTECS : Jurnal Sistem Telekomunikasi Elektronika Sistem Kontrol Power Sistem & Komputer*, Juli 2021, Vol.1 / No.2

[5] R. Dijaya dan H. Setiawan, *Buku Ajar Pengolahan Citra Digital*. Sidoarjo, Indonesia: UMSIDAPress, 2023, p. 1.

[6] H. Fitriyah and R. C. Wihandika, Dasar-dasar Pengolahan Citra Digital. UB Press, 2020, p. 2.

[7] Purba and Manogu Supriadi, "Perancangan Sistem Identifikasi Biometrik Iris Mata Menggunakan Metode Transformasi Hough", J. Informasi dan Teknologi Ilmiah (INTI), Vol. 7, pp.117-122, 2020

[8] Astiningrum, Mungki, Mustika Mentari, and Rezida Rismawati Nur Rachma, "Deteksi kesegaran daging sapi berdasarkan ekstraksi fitur warna dan tekstur." Seminar Informatika Aplikatif Polinema. 2019.

[9] W. A. Saputra, "Penerapan Teknik Blob Analysis dalam Pemilihan Region of Interest pada Citra Leukosit," *J. of INISTA*, vol. 2, no. 2, pp. 76-85, May 2020.

[10] Z. Yu, J. Zhang, C. Li, and Y. Zhao, "Region-level image understanding for object detection and scene graph generation," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 43, no. 12, pp. 4101-4113, Dec. 2021. Available: https://doi.org/10.1109/TPAMI.2021.3056750.

[11] A. Rahim, K. Emha, dan T. Luthfi, "Convolutional Neural Network untuk Klasifikasi Penggunaan Masker," *Jurnal Teknologi Informasi dan Komunikasi*, vol. 10, no. 2, Desember 2020, pp. 109-115.

[12] W. Hidayat, M. Ardiansyah, and A. Setyanto, "Pengaruh Algoritma ADASYN dan SMOTE terhadap Performa Support Vector Machine pada Ketidakseimbangan Dataset Airbnb," *Edumatic: Jurnal Pendidikan Informatika*, vol. 5, no. 1, pp. 11-20, Jun. 2021.

[13] J. Howse and J. Minichino, Learning OpenCV 5 Computer Vision with Python - Fourth Edition, Packt Publishing, 2022. [Online]. Available: https://www.oreilly.com/library/view/learning-opencv-5/9781803230221. [Accessed: Sep. 23, 2024].

[14] K. D. Lee, Introduction to Python Programming and Data Structures, 3rd ed. Pearson, 2023. Available: https://learning.oreilly.com/library/view/introduction-to-python/9780136747655/.