Journal of Computer Science and Technology
Volume 4 ; Nomor 2 ; Mei 2026 ; Page 88-96
DOI : https://doi.org/10.59435/jocstec.v4i2.752
https://jurnal.padangtekno.com/index.php/jocstec
E-ISSN : 2985-4318
P-ISSN : 2985-5772

Sistem Penghitungan Jumlah Kendaraan Pada Area Parkir Menggunakan Background Subtraction Dan OpenCV

Nando Juliansyah[1], Wendi Saputra[2], Febri Dristyan[3]

[1,2,3]Teknologi Rekayasa Perangkat Lunak, Politeknik Jambi
[1]240658302005@politeknikjambi.ac.id, [2]240658302007@politeknikjambi.ac.id,
[3]febri.dristyan@politeknikjambi.ac.id

Abstrak

Masalah pengelolaan parkir di Politeknik Jambi sering menyebabkan ketidakefisienan akibat sistem penghitungan manual yang rentan *human error*. Penelitian ini bertujuan merancang sistem penghitungan kendaraan otomatis berbasis pengolahan citra digital. Metode yang digunakan adalah *Background Subtraction* dengan algoritma MOG2 dan *OpenCV* menggunakan video kamera *smartphone*. Sistem mengintegrasikan proses *preprocessing* citra, operasi morfologi, dan *Euclidean Distance Tracker* untuk melacak serta menghitung kendaraan secara *real-time*. Hasil penelitian menunjukkan sistem mampu membedakan kendaraan yang bergerak dengan objek statis secara akurat melalui *Virtual Counting Line*. Dengan beban komputasi yang ringan dan biaya rendah, sistem ini efektif menjadi solusi otomatisasi manajemen parkir di lingkungan kampus.

Kata Kunci : *Background Subtraction*, *OpenCV*, Penghitungan Kendaraan, Pengolahan Citra, MOG2.

Abstract

Parking management issues at Politeknik Jambi often lead to inefficiencies due to manual counting systems prone to human error. This study aims to design an automatic vehicle counting system based on digital image processing. The method utilizes Background Subtraction with the MOG2 algorithm and OpenCV using smartphone video input. The system integrates image preprocessing, morphological operations, and Euclidean Distance Tracker to track and count vehicles in real-time. The results demonstrate that the system can accurately distinguish between moving vehicles and static objects via a Virtual Counting Line. With low computational requirements and cost-effectiveness, this system offers an efficient automation solution for campus parking management.

Keyword : *Background Subtraction, OpenCV, Vehicle Counting, Image Processing, MOG2.*

PENDAHULUAN

Perkembangan teknologi pengolahan citra digital (*digital image processing*) saat ini telah mengalami transformasi yang sangat signifikan. Teknologi ini tidak lagi terbatas pada manipulasi gambar statis, melainkan telah diimplementasikan secara luas pada berbagai sektor kehidupan dinamis, mulai dari sistem keamanan hingga manajemen infrastruktur area parkir yang otomatis[1]. Salah satu masalah yang sering muncul di lingkungan kampus, seperti di Politeknik Jambi, adalah efektivitas pengelolaan area parkir. Lonjakan jumlah volume kendaraan pada jam-jam sibuk perkuliahan sering kali menimbulkan ketidakefisienan karena pengendara kesulitan mengetahui kapasitas ruang parkir secara langsung.

CC Attribution-ShareAlike 4.0 License.

Menurut Hariyanto, Sofwan, dan Hidayatno, keterbatasan informasi ketersediaan parkir sering memicu antrean dan pemborosan waktu bagi pengendara. Selama ini, penghitungan manual oleh petugas keamanan rentan terhadap *human error* akibat kelelahan dan penurunan konsentrasi saat melakukan pengawasan monoton dalam waktu lama. Oleh karena itu, diperlukan solusi berbasis komputer yang mampu bekerja otomatis dan konsisten untuk mengawasi serta menghitung kendaraan melalui integrasi perangkat penangkap gambar[1][2].

Dalam disiplin *computer vision*, terdapat berbagai pendekatan untuk deteksi kendaraan secara dinamis. Penelitian oleh Alfian dan Lusiana menunjukkan bahwa metode *Haar Cascade* mampu mendeteksi kendaraan dengan akurasi 75%. Namun, metode ini membutuhkan proses klasifikasi dan pelatihan data yang sangat intensif agar performanya tetap stabil dalam menghadapi variasi bentuk kendaraan di lapangan[3].

Meskipun memerlukan pelatihan intensif, algoritma *Haar Cascade* telah banyak diimplementasikan dalam berbagai penelitian transportasi dan keamanan, seperti deteksi kecepatan kendaraan, pengawasan protokol kesehatan, serta estimasi jarak aman antar-kendaraan. Algoritma ini juga kerap dikombinasikan dengan *Optical Character Recognition* (OCR) untuk identifikasi plat nomor otomatis, bahkan hingga pengenalan wajah dalam manajemen hubungan pelanggan. Evaluasi dari berbagai studi menunjukkan bahwa *Haar Cascade* sangat efektif untuk deteksi fitur spesifik, namun beban komputasi penyiapan *dataset*-nya dinilai kurang efisien jika diterapkan pada purwarupa tingkat tugas kuliah dengan spesifikasi perangkat keras standar[4].

Metode *background subtraction* yang diimplementasikan menggunakan OpenCV mampu mendeteksi kendaraan dan menghitung kapasitas parkir secara **real-time** dengan tingkat akurasi yang tinggi, penggunaan metode *Background Subtraction* terbukti sangat efektif untuk pemantauan arus kendaraan dinamis dan mampu mencatatkan tingkat akurasi harian yang sangat tinggi hingga mencapai 90,57%. Selain itu, terdapat pula pendekatan berbasis *deep learning* seperti algoritma YOLO (*You Only Look Once*) yang diteliti oleh Rachmawati dan Widhyaestoeti untuk mendeteksi jumlah kendaraan di jalur padat. Meskipun YOLO memberikan performa segmentasi objek berbasis pemrosesan video digital yang sangat membantu menghasilkan penghitungan akurat, implementasinya memerlukan sumber daya komputasi kartu grafis yang besar. Oleh karena itu, beban komputasi *Background Subtraction* yang rendah namun tetap menghasilkan akurasi yang presisi menjadikannya opsi terbaik untuk tingkat tugas proyek ini[2].

Selain faktor penentuan metode algoritma, komponen penangkap video juga memegang peranan penting dalam keberhasilan sistem. Berbeda dengan penelitian konvensional yang umumnya mengandalkan kamera pengawas komersial seperti CCTV dengan instalasi kabel yang rumit dan permanen, proyek tugas ini memanfaatkan rekaman video dinamis yang diambil secara mandiri menggunakan Kamera HP melalui metode pengumpulan data luring (*offline data acquisition*). Perangkat kamera HP modern saat ini sudah dibekali dengan sensor optik yang sangat baik dan mampu merekam video beresolusi tinggi, sehingga sangat fleksibel, portabel, dan hemat biaya untuk digunakan sebagai media akuisisi data skala purwarupa laboratorium. File video format .mp4 hasil rekaman tersebut nantinya dimasukkan ke dalam sistem pemrograman menggunakan fungsi pembaca video OpenCV (cv2.VideoCapture)[5].

Pustaka OpenCV dengan bahasa pemrograman Python[6] memudahkan integrasi berbagai tahapan pengolahan citra untuk menyempurnakan hasil akhir. Melalui penerapan operasi morfologi seperti *opening* dan *closing*, gangguan berupa *noise* digital dari sensor kamera HP maupun pergeseran bayangan kendaraan dapat diminimalisir secara optimal. Optimasi deteksi area ruang

CC Attribution-ShareAlike 4.0 License.

parkir biner ini juga didukung oleh pengembangan sistem berbasis OpenCV sebagaimana yang diteliti oleh Afifudin dan Ardi untuk mendeteksi ketersediaan ruang parkir secara visual, serta purwarupa deteksi ketersediaan slot parkir berbasis pengolahan citra oleh Sani dan Ayyasy[7].

Sistem ini menetapkan area deteksi spesifik menggunakan *Region of Interest* (ROI) dan garis batas virtual (*counting line*). Saat kontur kendaraan melintasi garis tersebut, variabel penghitung akan bertambah secara otomatis. Penelitian Rachmawati dan Widhyaestoeti menunjukkan bahwa segmentasi objek berbasis pemrosesan video digital sangat membantu meningkatkan akurasi penghitungan. Selain itu, menurut Faruq dan Al Hakim (2025), penggunaan *bounding box* pada deteksi objek memungkinkan identifikasi lokasi kendaraan yang lebih presisi, sehingga mendukung analisis objek pada citra digital secara keseluruhan.

METODOLOGI PENELITIAN

Alur Sistem Pengolahan Citra[8][9]

Sistem ini menerapkan pendekatan pemrosesan *frame-by-frame* secara sekuensial dengan tahapan sistematis sebagai berikut:

1. **Input Video:** Membaca file video .mp4 menggunakan fungsi cv2.VideoCapture().
2. **Pra-pengolahan (*Preprocessing*):** Mengonversi citra ke skala keabuan (*grayscale*) untuk efisiensi memori dan menerapkan *Gaussian Blur* (5×5 piksel) untuk mereduksi *noise*.
3. **Segmentasi (*Background Subtraction*):** Menggunakan algoritma MOG2 untuk memisahkan objek bergerak (*foreground*) dari latar belakang statis.
4. **Operasi Morfologi:** Menggunakan elemen struktur elips 11×11 piksel melalui tiga tahapan: *Opening* (1 iterasi) untuk membersihkan *noise* kecil, *Closing* (3 iterasi) untuk menutup lubang pada siluet objek, dan dilasi tambahan (2 iterasi) untuk menyatukan kontur kendaraan yang terpecah.
5. **Deteksi Kontur & *Bounding Box*:** Mengidentifikasi tepian objek melalui cv2.findContours(), menyaring objek berdasarkan luas piksel minimum, serta menerapkan *bounding box* untuk menentukan titik koordinat *centroid* (pusat objek).
6. **Pelacakan Objek (*Object Tracking*):** Menjaga konsistensi identitas kendaraan antar-*frame* menggunakan rumus jarak Euclidean. Jika jarak *centroid* antar-*frame* < 100 piksel, sistem menganggapnya sebagai objek yang sama.
7. **Logika Garis Batas (*Counting Line Logic*):** Mengimplementasikan garis batas virtual pada koordinat tertentu. Sistem melakukan inkrementasi variabel hitung secara otomatis hanya jika *centroid* objek melintasi garis tersebut, guna memastikan kendaraan yang parkir (*stay*) tidak terhitung.

Tahapan Pemrosesan Video (*Video Processing*) Secara Detail

Pengambilan Data Video (*Input Video*)

Tahap awal yang paling penting adalah proses pengumpulan video. Proyek ini tidak menggunakan kamera pengawas komersial (CCTV), melainkan menggunakan kamera HP personal yang dipasang pada tripod secara manual.

Penggunaan tripod sangat wajib dalam metode *Background Subtraction* karena algoritma ini menuntut kamera berada dalam kondisi diam mutlak (statis). Jika kamera bergoyang sedikit saja terkena angin, komputer akan menganggap seluruh latar belakang (seperti pohon atau gedung) ikut bergerak, sehingga sistem akan salah hitung. Video direkam dalam format digital .mp4 dengan resolusi High Definition (1080p) pada kecepatan 30 bingkai per detik (*frame rate* 30 fps) untuk

CC Attribution-ShareAlike 4.0 License.

menjamin pergerakan kendaraan tertangkap dengan mulus tanpa putus-putus.

Pra-pengolahan Citra (*Preprocessing*)[10]

Tahap ini bertujuan untuk menyederhanakan data citra agar beban komputasi berkurang dan deteksi objek menjadi lebih optimal. Mengingat citra mentah berformat RGB memiliki dimensi data yang berat, dilakukan dua proses utama:

- **Konversi Skala Keabuan (*Grayscale Conversion*):** Citra tiga dimensi (RGB) dipangkas menjadi satu saluran warna (hitam-putih) menggunakan fungsi cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY). Langkah ini mereduksi beban komputasi hingga sepertiga dari ukuran asli tanpa menghilangkan informasi bentuk siluet kendaraan.
- **Filter Perataan (*Gaussian Blur*):** Untuk mereduksi *noise* digital akibat sensor kamera atau gangguan lingkungan, digunakan filter cv2.GaussianBlur(gray, (5, 5), 0). Filter ini meratakan nilai setiap piksel berdasarkan 25 piksel tetangganya (jendela 5×5), sehingga menghasilkan citra yang lebih halus, rapi, dan siap untuk tahap segmentasi berikutnya.

Segmentasi Objek Bergerak (*Background Subtraction*)[11]

Sistem menggunakan algoritma *Mixture of Gaussians* versi 2 (MOG2) via cv2.createBackgroundSubtractorMOG2() sebagai inti pendeteksian. Algoritma ini dikonfigurasi dengan history=500 untuk pemodelan latar belakang, varThreshold=50 sebagai sensitivitas, dan detectShadows=True untuk pengenalan bayangan otomatis. Metode ini unggul dalam beradaptasi secara dinamis terhadap perubahan pencahayaan luar ruangan dengan mempelajari objek statis sebagai *background model*, sehingga setiap perubahan piksel akibat kendaraan yang melintas diidentifikasi sebagai *foreground*.

Untuk mengatasi distorsi bayangan, sistem menerapkan penghapusan bayangan (*shadow removal*) menggunakan fungsi cv2.threshold(mask, 200, 255, cv2.THRESH_BINARY). Fungsi ini memfilter piksel pada *mask*: piksel bernilai di atas 200 (objek bergerak) dipertahankan sebagai warna putih, sedangkan piksel di bawah 200 (bayangan bernilai 127 dan latar belakang bernilai 0) diubah menjadi hitam. Melalui proses ini, objek kendaraan dapat diisolasi secara bersih dan akurat untuk tahap deteksi selanjutnya.

Operasi Morfologi (*Morphological Operations*)[10]

Meskipun proses segmentasi telah dilakukan, citra biner sering kali mengandung *noise* berupa bintik kecil atau lubang pada siluet kendaraan akibat pantulan cahaya atau tekstur bodi. Untuk menyempurnakan hasil ini, sistem menjalankan tiga operasi morfologi secara berurutan menggunakan elemen struktur elips berukuran 11×11 piksel melalui cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (11, 11)). Bentuk elips dipilih karena lebih merepresentasikan bentuk alami kendaraan dibandingkan bentuk kotak.

Tahapan operasi tersebut meliputi:

- **Opening (1 iterasi):** Kombinasi erosi yang diikuti dilasi untuk menghilangkan *noise* kecil di luar objek tanpa mengubah dimensi kendaraan secara signifikan.
- **Closing (3 iterasi):** Kombinasi dilasi yang diikuti erosi untuk menutup lubang-lubang kosong pada siluet kendaraan sehingga bentuk objek menjadi lebih padat dan utuh.
- **Dilasi Tambahan (2 iterasi):** Dilakukan untuk mempertebal tepi kontur, sehingga bagian-bagian objek yang terpisah namun berdekatan dapat menyatu menjadi satu kesatuan objek yang utuh sebelum diproses ke tahap deteksi kontur.

CC Attribution-ShareAlike 4.0 License.

Hasil akhir dari ketiga operasi morfologi ini adalah siluet kendaraan biner berwarna putih yang padat, utuh, bersih dari noise, dan siap untuk diproses ke tahap deteksi kontur berikutnya.

Deteksi Kontur dan Kotak Pembatas (*Contour Detection & Bounding Box*)[7]

Setelah mendapatkan siluet putih kendaraan yang solid, komputer harus mengenali objek tersebut sebagai koordinat matematika agar bisa dilacak posisinya. Proses ini dilakukan oleh fungsi kontur lewat perintah cv2.findContours(). Fungsi ini bekerja seperti mendeteksi garis tepi luar yang memisahkan warna putih dengan warna hitam.

Kontur yang ditemukan kemudian disaring menggunakan batas luas piksel minimal (*area thresholding*). Tujuannya agar objek yang terlalu kecil (seperti daun terbang, bayangan bendera, atau kucing lewat) langsung dibuang dan tidak ikut terhitung. Kontur yang lolos seleksi berukuran besar (pasti kendaraan) akan langsung dibungkus dengan kotak persegi panjang otomatis bernama ***Bounding Box*** menggunakan fungsi cv2.boundingRect(). Melalui kotak pembatas inilah komputer dapat menghitung titik tengah simetri objek secara akurat, yang dinamakan **Titik Centroid**.

Pelacakan Objek (Object Tracking)

Setelah bounding box dan titik centroid didapatkan, sistem harus mengenali apakah kendaraan pada frame saat ini adalah kendaraan yang sama dengan frame sebelumnya atau kendaraan baru. Proses ini dilakukan menggunakan algoritma Euclidean Distance Tracker.

Rumus jarak Euclidean: $d = \sqrt{((cx_1 - cx_2)^2 + (cy_1 - cy_2)^2)}$

Jika jarak centroid kendaraan pada frame saat ini dengan centroid terdaftar pada frame sebelumnya kurang dari ambang batas 100 piksel, maka dianggap kendaraan yang sama dan ID-nya dipertahankan. Jika tidak ada kecocokan, sistem mendaftarkan objek tersebut sebagai kendaraan baru dengan ID unik baru. Objek yang sudah tidak terdeteksi pada frame berikutnya langsung dihapus dari daftar pelacakan.

Logika Garis Batas dan Penghitungan (*Counting Logic*)

Tahap paling akhir dari seluruh rangkaian metodologi ini adalah menentukan bagaimana cara komputer menghitung kendaraan secara otomatis. Di dalam tampilan layar video, sistem akan menggambar sebuah garis horizontal virtual (garis pembatas maya) yang memotong dari sisi kiri ke sisi kanan jalan. Garis ini berfungsi sebagai gerbang sensor otomatis.

Cara kerja penghitungannya sangat sederhana dan mudah dipahami:

1. Pustaka OpenCV akan menaruh satu titik penanda warna merah tepat di tengah-tengah bodi kendaraan (disebut titik *centroid*).
2. Komputer akan terus melacak pergerakan titik merah tersebut dari atas ke bawah layar monitor atau televisi.
3. **Logika Hitungnya:** Ketika kendaraan bergerak maju dan titik merah di tengah bodinya bergerak turun menyentuh atau melewati garis pembatas maya yang sudah kita pasang, sistem akan langsung mengenali kondisi tersebut sebagai tanda bahwa ada satu kendaraan yang melintas secara sah. Ketika kendaraan melewati garis batas atau garis deteksi maka kendaraan akan di hitung, jadi dia hanya menghitung kendaraan yang bergerak, yang sudah stay di parkir tidak di hitung.

CC Attribution-ShareAlike 4.0 License.

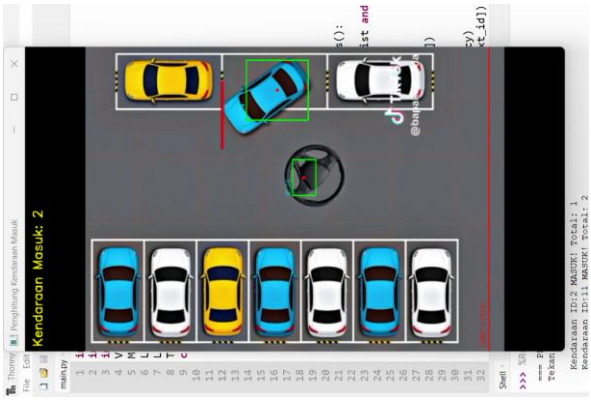
HASIL DAN PEMBAHASAN

Hasil Implementasi Sistem

Sistem penghitungan jumlah kendaraan berbasis *Background Subtraction* ini telah diimplementasikan menggunakan bahasa pemrograman Python dengan pustaka OpenCV. Pengujian dilakukan dengan memproses file video Parkir.mp4 sebagai input utama. Sistem berhasil melakukan segmentasi objek bergerak dari latar belakang statis, melacak pergerakan kendaraan melalui titik centroid, dan melakukan penghitungan otomatis saat objek melewati garis batas (*counting line*).

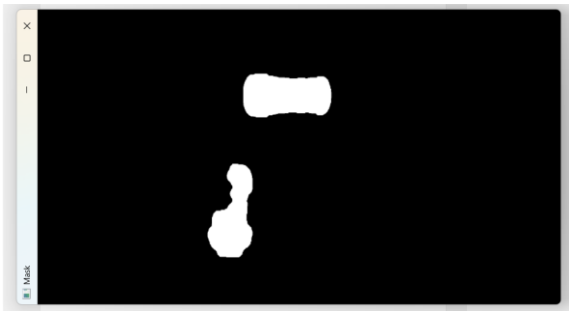
Analisis Antarmuka (Jendela Output)

Berdasarkan pengujian, sistem menampilkan dua jendela utama yang berperan penting dalam proses deteksi:

1. **Jendela "Penghitungan Kendaraan Masuk" (Output Utama):** Jendela ini merupakan representasi visual bagi pengguna. Sistem menggunakan **Bounding Box** berwarna hijau untuk mengikat objek kendaraan yang terdeteksi, serta **titik centroid** (titik merah) sebagai penanda pusat koordinat objek. Garis berwarna merah di bagian bawah layar bertindak sebagai *Virtual Counting Line*. Ketika titik centroid kendaraan beririsan dengan garis tersebut, sistem secara otomatis melakukan inkrementasi pada variabel total_masuk.
2. 

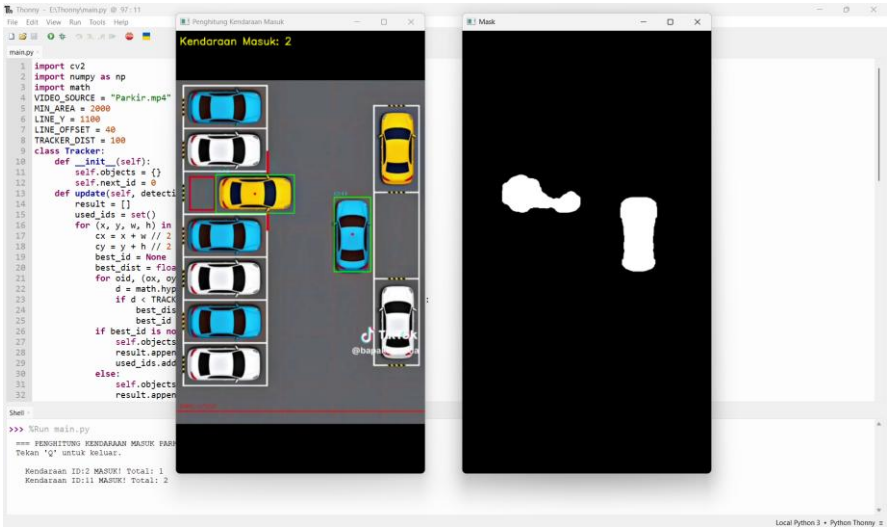
Gambar 1. Penghitungan Kendaraan Masuk

3. **Jendela "Mask" (Segmentasi Objek):** Jendela ini menampilkan hasil *Background Subtraction* (MOG2). Warna putih pada jendela ini adalah objek yang dianggap sebagai *foreground* oleh sistem, sementara warna hitam adalah *background*. Jendela ini sangat krusial untuk memastikan bahwa filter morfologi (Opening, Closing, dan Dilasi) berhasil membersihkan *noise* sehingga siluet kendaraan yang dihasilkan padat dan utuh sebelum dihitung oleh sistem.


Gambar 2. Segmentasi Objek

CC Attribution-ShareAlike 4.0 License.

4. **Jendela Penghitungan Kendaraan Masuk dan Jendela Mask**



Gambar 3. Penghitungan Kendaraan Masuk dan Jendela Mask

5. **Shell:** Log yang muncul pada *shell* program saat pengujian, didapatkan data sebagai berikut:

```
Shell
>>> %Run main.py
  === PENGHITUNG KENDARAAN MASUK PARKIR ===
  Tekan 'Q' untuk keluar.

    Kendaraan ID:2 MASUK! Total: 1
    Kendaraan ID:11 MASUK! Total: 2

  === TOTAL KENDARAAN MASUK: 2 ===

>>>
```

Gambar 4. Shell Informasi pengujian

Visualisasi Proses Pengolahan Citra

Berikut adalah visualisasi hasil dari setiap tahapan pemrosesan citra yang terjadi dalam *frame* program:

- **Hasil Masking (Biner):** Setelah melewati proses MOG2, *noise filtering*, dan *morphological operations* (Opening, Closing, Dilasi), sistem menghasilkan gambar biner yang bersih. Hanya objek kendaraan yang memiliki luas lebih besar dari MIN_AREA (2000 piksel) yang dipertahankan dalam bentuk siluet putih padat.
- **Bounding Box dan Tracking:** Setiap objek yang terdeteksi diberikan kotak pembatas (*bounding box*) berwarna hijau beserta ID unik untuk melacak identitasnya dari satu *frame* ke *frame* berikutnya guna menghindari penghitungan ganda pada kendaraan yang sama.

Analisis Logika Penghitungan

Logika penghitungan didasarkan pada posisi titik tengah (*centroid*) kendaraan terhadap koordinat garis virtual yang ditetapkan pada LINE_Y = 1100.

- **Mekanisme Sensor:** Variabel counted_ids digunakan untuk menyimpan daftar ID kendaraan yang sudah melewati garis. Sistem hanya akan menambah jumlah total_masuk jika sebuah ID objek terdeteksi berada di dalam rentang area LINE_OFFSET (40 piksel di atas dan di

CC Attribution-ShareAlike 4.0 License.

bawah LINE_Y) dan ID tersebut belum pernah terdaftar sebelumnya di dalam variabel counted_ids.

- **Akurasi:** Pendekatan ini terbukti efektif untuk membedakan antara kendaraan yang sedang melintas (bergerak) dengan kendaraan yang sudah parkir (*stay*). Karena objek yang sudah diam tidak lagi menciptakan perubahan piksel yang signifikan pada *background subtractor*, sistem secara otomatis mengabaikan kendaraan yang tidak bergerak.

Uji Coba dan Evaluasi Performa

Dalam pengujian sistem, diperoleh beberapa observasi teknis terkait performa program:

Tahapan	Fungsi Utama	Peran dalam Hasil Akhir
Preprocessing	Gaussian Blur	Menghilangkan *noise* sensor kamera agar deteksi kontur lebih stabil.
Morfologi	Closing (3x)	Menutup lubang pada siluet mobil/motor agar terdeteksi sebagai satu kesatuan.
Tracking	Euclidean Dist	Menjaga konsistensi ID kendaraan meskipun terjadi fluktuasi pergerakan.
Counting	Line Logic	Memastikan kendaraan hanya terhitung satu kali saat melintasi gerbang maya.

KESIMPULAN

Penelitian ini berhasil mengimplementasikan sistem penghitungan jumlah kendaraan otomatis menggunakan metode *Background Subtraction* berbasis MOG2 dan pustaka *OpenCV* pada perangkat keras standar. Integrasi teknik pra-pengolahan citra dan operasi morfologi terbukti efektif dalam meminimalkan *noise* serta menghasilkan siluet objek yang akurat, sehingga deteksi kontur dan pelacakan *centroid* dapat berjalan dengan presisi tinggi. Melalui logika *Virtual Counting Line* dan *Euclidean Distance Tracker*, sistem mampu membedakan kendaraan yang melintas dengan kendaraan yang parkir, sekaligus meniadakan risiko penghitungan ganda. Secara keseluruhan, pemanfaatan kamera *smartphone* dengan instalasi statis pada sistem ini menawarkan solusi manajemen area parkir yang aplikatif, ekonomis, dan efisien untuk diterapkan di lingkungan kampus Politeknik Jambi.

DAFTAR PUSTAKA

[1] M. S. Hariyanto, A. Sofwan, and A. Hidayatno, "Perancangan Sistem Penghitung Jumlah Kendaraan Pada Area Parkir Dengan Metode Background Subtraction Berbasis Internet of Things," *Transient*, vol. 7, no. 3, p. 775, 2019, doi: 10.14710/transient.7.3.775-781.

[2] S. A. Zai, U. N. Medan, S. F. Margaret, U. N. Medan, Y. P. Putri, and U. N. Medan, "Sistem Pendeteksi Kecepatan Kendaraan dengan Menggunakan Metode Deep Learning," vol. 1, no. 6, 2023.

[3] R. Alfian and V. Lusiana, "Penerapan OpenCV dengan Metode Haar Cascade untuk Mendeteksi Jumlah Kendaraan di Tempat Parkir," *J. JTIK (Jurnal Teknol. Inf. dan Komunikasi)*, vol. 8, no. 1, pp. 104–106, 2024, doi: 10.35870/jtik.v8i1.1360.

[4] F. Dristyan, A. B. Trisnawan, and M. S. Simanjuntak, *COMPUTER VISION : Implementasi dan Penerapan*.

[5] M. Diaz Ellyas Fenca Putra, M. Dwiky Riza Ardana, A. Ahnaf, M. Akmal, and S. Ayu Wulandari, "Implementasi Sistem Monitoring Dan Otomatisasi Perhitungan Kapasitas Parkir

CC Attribution-ShareAlike 4.0 License.

Mobil Menggunakan Metode Background Subtraction," *JATI (Jurnal Mhs. Tek. Inform.*, vol. 9, no. 2, pp. 2824–2831, 2025, doi: 10.36040/jati.v9i2.13196.

[6] A. Muis *et al.*, *KECERDASAN BUATAN DENGAN PYTHON*. PT. Faaslib Serambi Media, 2025.

[7] Clara Alyuson, Husnul Hotimah, and Febri Dristyan, "Penerapan Computer Vision Dalam Pengenalan Bendera Negara: Integrasi HSV Dan Kontur Menggunakan OpenCV," *J. Comput. Sci. Technol.*, vol. 3, no. 3, pp. 166–173, 2025, doi: 10.59435/jocstec.v3i3.548.

[8] M. Y. Efendi, R. Wulanningrum, and B. Setiawan, "Rancang Bangun Sistem Deteksi Manusia dengan YOLO pada video CCTV," *Inotek*, vol. 8, pp. 1149–1154, 2024, [Online]. Available: https://proceeding.unpkediri.ac.id/index.php/inotek/

[9] A. Alhidayah, W. J. Rohizi, F. Dristyan, T. Rekayasa, P. Lunak, and P. Jambi, "Penerapan Model Warna HSV untuk Deteksi Objek Berbasis Warna secara Real-Time pada Kondisi Pencahayaan Dinamis," *Fusion J. Res. Eng.*, vol. 2, no. 2, pp. 56–63, 2025.

[10] L. A. Kurniawan, I. P. A. Bayupati, and K. S. Wibawa, "Sistem Hitung Kendaraan Berdasarkan Jenis Menggunakan Metode Background Subtraction," *JITTER J. Ilm. Teknol. dan Komput.*, vol. 1, no. 2, pp. 265–273, 2021.

[11] M. Daffa Ramadhana, R. Putra Muhammad, F. Y. Limpraptono, T. E. S1, I. Malang, and M. Indonesia, "Sistem Low-Cost Untuk Menghitung Jumlah Pengunjung Pada Ruangan Menggunakan Metode Background Subtraction," *Magnetika*, vol. 8, no. 2, pp. 48–66, 2024.

CC Attribution-ShareAlike 4.0 License.