Penerapan Optical Flow Berbasis Sistem Persamaan Linier untuk Estimasi Kecepatan Kendaraan pada Rekaman CCTV

Wafiq Hibban Robbany - 13524016
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13524016@std.stei.itb.ac.id, wafiq.robbany@gmail.com

***Abstrak* — Estimasi kecepatan kendaraan merupakan teknik krusial dalam sistem transportasi cerdas dan aplikasi pengawasan, yang memungkinkan pemantauan arus lalu lintas secara otomatis tanpa bergantung pada sensor perangkat keras khusus. Makalah ini menyajikan perspektif aljabar linier dalam estimasi kecepatan, dengan fokus khusus pada metode Optical Flow dan ketergantungannya pada kalkulus vektor. Untuk menghasilkan pelacakan gerak yang akurat, sistem menggunakan metode Lucas-Kanade untuk menyelesaikan sistem persamaan linier *overdetermined* pada titik-titik fitur yang dideteksi melalui algoritma berbasis nilai eigen (Shi-Tomasi). Perpindahan kendaraan dihitung dengan menganalisis selisih antara vektor koordinat pada *frame* yang berurutan. Magnitudo (panjang) dari vektor perpindahan ini dihitung menggunakan Norma Euclidean, yang kemudian dipetakan ke dalam satuan fisik (kilometer per jam) melalui konstanta kalibrasi linier yang diturunkan dari proyeksi perspektif permukaan jalan.**

***Keywords*— Aljabar Linier , Estimasi Kecepatan Kendaraan, Lucas-Kanade, Optical Flow, Sistem Persamaan Linier.**

I. PENDAHULUAN

Manajemen lalu lintas memegang peranan penting dalam menjaga kelancaran dan keselamatan transportasi di kawasan perkotaan. Jumlah kendaraan yang bertumbuh secara eksponensial seringkali tidak sebanding dengan penyediaan infrastruktur jalan raya, yang pada akhirnya menjadi penyebab masalah kemacetan, peningkatan risiko kecelakaan, dan pelanggaran lalu lintas. Salah satu parameter utama yang perlu dipantau dalam manajemen lalu lintas adalah kecepatan kendaraan. Data kecepatan kendaraan yang akurat tidak hanya berperan dalam sistem penegakan hukum berbasis elektronik, tetapi juga penting dalam analisis arus lalu lintas untuk mengoptimalkan sistem lampu lalu lintas serta perencanaan infrastruktur di masa mendatang.

Pada umumnya, pengukuran kecepatan kendaraan seringkali bergantung pada pengadaan perangkat keras khusus berbasis sensor fisik. Teknologi seperti *speed gun*, sensor loop induksi yang ditanam di bawah permukaan jalan, atau sensor lidar, memang memberikan tingkat akurasi yang tinggi. Namun, penerapan teknologi-teknologi ini memiliki kendala signifikan pada biaya pengadaan, pemeliharaan hingga proses pemasangan yang invasif karena seringkali memerlukan pekerjaan pada badan jalan. Di sisi lain, teknologi kamera pengawas atau *Closed-Circuit Television* (CCTV) telah menjadi infrastruktur yang umum tersedia di berbagai sudut jalan raya. Namun, sebagian besar pemanfaatan CCTV ini hanya terbatas pada pemantauan visual atau dokumentasi bukti kejadian, tanpa dilakukan pengolahan data kuantitatif secara otomatis.

Perkembangan pesat dalam bidang *Computer Vision* membuka peluang untuk meningkatkan fungsi CCTV dari sekadar alat pemantauan visual pasif menjadi sensor cerdas yang dapat melakukan pengukuran kuantitatif. Namun, tantangan utama dalam mewujudkan hal ini tidak terletak pada aspek perangkat keras, melainkan pada pemodelan matematis yang mendasari pemrosesan citranya. Permasalahan mendasar yang muncul adalah bagaimana sebuah computer mampu menafsirkan konsep “gerakan" dari sekumpulan angka-angka piksel yang tersusun dalam sebuah matriks citra digital.

Salah satu metode yang umum digunakan untuk mendeteksi urutan citra adalah *Optical Flow*. *Optical Flow* adalah teknik dalam pengolahan citra yang memodelkan pergerakan objek dengan asumsi bahwa intensitas cahaya pada suatu titik di permukaan objek akan tetap konstan meskipun objek tersebut bergerak antar frame waktu. Dengan asumsi ini, dapat dihasilkan sebuah persamaan linear dasar. Akan tetapi, persamaan tersebut menimbulkan permasalahan matematis yang disebut *aperture problem*, yaitu kondisi di mana kita memiliki satu persamaan linier dengan dua variabel yang tidak diketahui, yang dalam konteks ini adalah komponen kecepatan horizontal dan vertical. Dalam konteks aljabar linear, sistem seperti ini tidak memiliki solusi tunggal yang unik.

Untuk mengatasi permasalahan matematis tersebut, akan digunakan pendekatan Sistem Persamaan Linier (SPL) melalui metode Lucas-Kanade. Metode ini tidak menganalisis piksel secara individual, melainkan melalui pertimbangan sekumpulan piksel dalam suatu *window* dengan asumsi gerak yang seragam. Pendekatan ini akan menghasilkan sistem persamaan linear, yang kemudian diselesaikan menggunakan metode kuadrat terkecil atau

least squares untuk memperoleh vector kecepatan. Makalah ini bertujuan untuk mengkaji dan menerapkan *Optical Flow* dengan metode Lucas-Kanade dalam estimasi kecepatan kendaraan pada rekaman CCTV statis, dengan berfokus pada analisis struktur aljabar yang mendasarinya, mulai dari pembentukan SPL berbasis gradien citra hingga pemanfaatan vector di ruang Euclidean untuk mengonversi perpindahan piksel menjadi besaran kecepatan fisik.

II. Landasan Teori

A. Sistem Persamaan Linier

Sistem Persamaan Linear (SPL) merupakan kumpulan dari m buah persamaan linear dengan n buah variabel (peubah) $x_1, x_2, \ldots, x_n$. Secara umum, sebuah sistem persamaan linear dapat dituliskan dalam bentuk:

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &= b_2 \\ \vdots \quad\quad & \quad\quad \vdots \quad\quad\quad \vdots \quad\quad \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n &= b_m \end{aligned}$$

Dimana a_{ij} merupakan koefisien dari variabel x_j pada persamaan ke-i, dan b_i adalah konstanta pada persamaan ke-i.

Atau dalam perkalian matriks $A\mathbf{x} = \mathbf{b}$:

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}$$

A $\quad$ $\mathbf{x}$ $\quad$ $\mathbf{b}$

Gambar 2.1 Representasi matriks SPL
Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2025-2026/Algeo-03-Sistem-Persamaan-Linier-2025.pdf

Di mana A adalah matriks koefisien berukuran $m \times n$, x adalah vektor kolom variabel berukuran $n \times 1$, dan b adalah vektor kolom konstanta berukuran $m \times 1$.

Untuk mempermudah penyelesaian, SPL seringkali diubah menjadi bentuk *Augmented Matrix*, yaitu notasi matriks yang menggabungkan matriks koefisien A dan vektor konstanta b menjadi satu kesatuan $[A|b]$.

$$[A \mid \mathbf{b}] = \left[\begin{array}{cccc|c} a_{11} & a_{12} & \cdots & a_{1n} & b_1 \\ a_{21} & a_{22} & \cdots & a_{2n} & b_2 \\ \vdots & \vdots & & \vdots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} & b_m \end{array}\right]$$

Gambar 2.2 Matriks Augmented SPL
Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2025-2026/Algeo-03-Sistem-Persamaan-Linier-2025.pdf

Salah satu metode fundamental untuk mencari solusi SPL adalah dengan mengubah matriks augmented menjadi bentuk yang lebih sederhana menggunakan Operasi Baris Elementer (OBE). Terdapat tiga jenis operasi yang diperbolehkan dalam OBE:

1) Menukarkan posisi dua buah baris.
2) Mengalikan sebuah baris dengan skalar tak nol.
3) Menambahkan kelipatan suatu baris ke baris lainnya.

Terdapat dua varian utama dalam metode eliminasi ini:

1) Eliminasi Gauss: Mengubah matriks augmented menjadi bentuk Eselon Baris. Dari bentuk ini, solusi dapat ditemukan dengan teknik substitusi balik (*back substitution*).

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} & b_1 \\ a_{21} & a_{22} & \cdots & a_{2n} & b_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} & b_n \end{bmatrix} \sim\text{OBE}\sim \begin{bmatrix} 1 & * & * & \cdots & * & * \\ 0 & 1 & * & \cdots & * & * \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \vdots & 1 & * \end{bmatrix}$$

Gambar 2.3 Eliminasi Gauss
Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2025-2026/Algeo-03-Sistem-Persamaan-Linier-2025.pdf

2) Eliminasi Gauss-Jordan: Melanjutkan proses eliminasi Gauss hingga matriks mencapai bentuk Eselon Baris Tereduksi. Pada bentuk ini, elemen *pivot* utama bernilai 1 dan menjadi satu-satunya angka tak nol di kolomnya, sehingga solusi variabel dapat langsung diketahui tanpa substitusi balik.

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} & b_1 \\ a_{21} & a_{22} & \cdots & a_{2n} & b_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} & b_m \end{bmatrix} \sim\text{OBE}\sim \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 & * \\ 0 & 1 & 0 & \cdots & 0 & * \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \vdots & 1 & * \end{bmatrix}$$

Gambar 2.4 Eliminasi Gauss-Jordan
Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2025-2026/Algeo-05-Sistem-Persamaan-Linier-2-2025.pdf

Berdasarkan konsistensinya, solusi dari sebuah Sistem Persamaan Linear memiliki tiga kemungkinan:

1) Solusi Tunggal (Unik): Sistem memiliki tepat satu himpunan nilai variabel yang memenuhi semua persamaan. Hal ini terjadi jika matriks eselon baris tereduksi memiliki *pivot* di setiap kolom variabelnya atau jumlah persamaan sama dengan jumlah variabel independen.
2) Solusi Banyak (Tak Hingga): Sistem memiliki tak hingga banyaknya solusi. Hal ini biasanya terjadi jika terdapat variabel bebas dalam solusi umum, yang sering ditandai dengan adanya baris nol pada matriks hasil eliminasi namun konstantanya juga nol.
3) Tidak Ada Solusi (Inkonsisten): Sistem tidak memiliki satupun solusi yang memenuhi semua persamaan secara bersamaan. Secara aljabar, hal ini ditandai dengan munculnya baris pada matriks augmented di mana semua koefisien nol tetapi konstantanya tidak

nol, seperti 0 = 1, yang merupakan pernyataan matematika yang salah.

Selain untuk mencari nilai variabel secara langsung, metode Gauss-Jordan juga memiliki aplikasi krusial dalam menentukan Invers Matriks. Invers atau balikan dari sebuah matriks bujursangkar A (A^{-1}) adalah matriks yang jika dikalikan dengan matriks A akan menghasilkan matriks identitas I ($AA^{-1} = \mathrm{I}$). Untuk mencari invers suatu matriks A, metode Gauss-Jordan digunakan dengan cara menyandingkan matriks A dengan matriks identitas I membentuk matriks augmented $[A|I]$. Kemudian, dilakukan serangkaian OBE untuk mengubah sisi kiri (A) menjadi matriks identitas (I). Jika berhasil, maka sisi kanan akan secara otomatis berubah menjadi inversnya (A^{-1}).

$$[\,A \mid I\,] \sim [\,I \mid A^{-1}\,]$$

Dengan demikian, metode Gauss–Jordan menyediakan cara yang efisien dan terstruktur untuk menentukan invers matriks bujursangkar.

B. Vektor di Ruang Euclidean

Dalam aljabar linier, vektor didefinisikan sebagai besaran yang memiliki arah dan nilai atau *magnitude.* Vektor di dalam ruang Euclidean-n, yang dinotasikan dengan R^n, dapat direpresentasikan sebagai *n-tupel* bilangan riil terurut. Contohnya, Jika n = 2, vektor berada di ruang dimensi dua, dan jika n = 3, vektor berada di ruang dimensi tiga.

Secara notasi, sebuah vektor v di R^n umumnya dituliskan dalam bentuk matriks kolom sebagai berikut:

$$v = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_\mathrm{n} \end{bmatrix}$$

Di mana $v_1, v_2, \ldots, v_n$ disebut sebagai komponen-komponen dari vektor v.

Sifat-sifat dasar dan operasi aljabar yang berlaku pada vektor di ruang Euclidean meliputi:

1) Penjumlahan Vektor: Dua buah vektor u dan v di ruang yang sama (R^n) dapat dijumlahkan dengan cara menjumlahkan komponen-komponen yang bersesuaian. Jika $\mathbf{u} = [u_1, u_2, \ldots, u_n]^T$ dan $\mathbf{v} = [v_1, v_2, \ldots, v_n]^T$, maka:

$$\mathbf{u} + \mathbf{v} = \begin{bmatrix} u_1 + v_1 \\ u_2 + v_2 \\ \vdots \\ u_\mathrm{n} + v_\mathrm{n} \end{bmatrix}$$

   Operasi penjumlahan vector bersifat komutatif (u + v = v + u) dan asosiatif ((u + v) + w = u + (v + w))

2) Perkalian Skalar: Sebuah vektor dapat dikalikan dengan suatu skalar k (bilangan riil). Operasi ini mengalikan setiap komponen vektor dengan skalar tersebut.

$$\mathbf{kv} = \begin{bmatrix} \mathrm{k}v_1 \\ \mathrm{k}v_2 \\ \vdots \\ \mathrm{k}v_\mathrm{n} \end{bmatrix}$$

   Perkalian skalar mengubah panjang vektor (penskalaan) dan dapat membalik arah vektor jika $k < 0$. Sifat distributif berlaku pada operasi ini, yaitu $k(\mathbf{u} + \mathbf{v}) = k\mathbf{u} + k\mathbf{v}$.

3) Norma Euclidean (Magnitude): Norma atau panjang dari sebuah vektor v di R^n, yang dinotasikan dengan $||\mathbf{v}||$, didefinisikan sebagai akar kuadrat dari jumlah kuadrat komponen-komponennya. Definisi ini merupakan generalisasi dari Teorema Pythagoras.

$$||\mathbf{v}|| = \sqrt{{v_1}^2 + {v_2}^2 + \ldots + {v_\mathrm{n}}^2}$$

   Nilai norma selalu non-negatif ($||\mathbf{v}|| > 0$) dan $||\mathbf{v}|| = 0$ jika dan hanya jika **v** adalah vektor nol.

4) Jarak Euclidean (Euclidean Distance): Jarak antara dua titik P dan Q dalam ruang Euclidean dapat dihitung sebagai norma dari vektor selisih antara kedua titik tersebut. Jika vektor posisi titik P adalah p dan titik Q adalah q, maka jarak Euclidean $d(\mathbf{p}, \mathbf{q})$ didefinisikan sebagai:

$$d(\mathbf{p}, \mathbf{q}) = ||\mathbf{p} - \mathbf{q}|| = \sqrt{(\mathrm{p}_1 - \mathrm{q}_1)^2 + \ldots + (\mathrm{p}_\mathrm{n} - \mathrm{q}_\mathrm{n})^2}$$

C. Eigenvalue dan Eigenvector

Dalam aljabar linier, konsep nilai eigen (*Eigenvalue*) dan vektor eigen (*Eigenvector*) merupakan aspek fundamental yang berkaitan dengan transformasi linear pada ruang vektor. Misalkan A adalah sebuah matriks bujursangkar berukuran $n \times n$. Sebuah vektor tak nol x di dalam R^n disebut sebagai Vektor Eigen dari matriks A jika Ax adalah kelipatan skalar dari x.

Hubungan tersebut dinyatakan dalam persamaan matematis berikut:

$$A\mathbf{x} = \lambda\mathbf{x}$$

Di mana λ adalah suatu skalar yang disebut sebagai Nilai Eigen dari matriks A yang bersesuaian dengan vektor eigen x.

Secara geometris, persamaan ini mengimplikasikan bahwa perkalian matriks A dengan vektor eigen x tidak mengubah arah dari vektor tersebut. Transformasi linear yang direpresentasikan oleh matriks A hanya mengakibatkan vektor x mengalami *scaling*, yaitu diperpanjang atau diperpendek sebesar faktor λ. Jika $\lambda > 0$, arah vektor tetap sama; jika $\lambda < 0$, arah vektor berbalik; dan jika $\lambda = 0$, vektor tersebut dipetakan ke vektor nol, yang berarti matriks A adalah singular.

Untuk menemukan nilai eigen dari matriks A, persamaan $Ax = \lambda x$ dapat ditulis ulang sebagai:

$$A\mathbf{x} - \lambda I\mathbf{x} = \mathbf{0}$$
$$(A - \lambda I)\mathbf{x} = \mathbf{0}$$

Di mana I adalah matriks identitas berukuran $n \times n$. Agar persamaan linear homogen ini memiliki solusi non-trivial (solusi di mana $x \neq 0$), determinan dari matriks koefisien $(A - \lambda I)$ harus bernilai nol. Persamaan ini dikenal sebagai Persamaan Karakteristik:

$$\det(A - \lambda I) = 0$$

Penyelesaian dari determinan tersebut akan menghasilkan sebuah polinomial derajat n dalam variabel λ, yang disebut polinomial karakteristik. Akar-akar dari polinomial ini adalah nilai-nilai eigen dari matriks A. Setelah nilai eigen λ diperoleh, vektor eigen yang bersesuaian dapat ditentukan dengan mencari basis ruang

nol (*null space*) dari matriks $(A - \lambda I)$ melalui operasi baris elementer.

Sifat-sifat penting yang berkaitan dengan nilai eigen antara lain:

1) Trace dan Determinan: Jumlah dari semua nilai eigen matriks A sama dengan jumlah elemen diagonal utama (*trace*) matriks A, sedangkan hasil kali dari semua nilai eigen sama dengan determinan matriks A.
2) Matriks Simetris: Jika A adalah matriks simetris ($A = A^T$), maka semua nilai eigennya adalah bilangan riil dan vektor-vektor eigen yang bersesuaian dengan nilai eigen yang berbeda akan saling ortogonal.

D. Optical Flow: Metode Lucas-Kanade

Optical Flow didefinisikan sebagai distribusi kecepatan gerak semu dari pola kecerahan dalam sebuah citra visual. Metode ini dibangun di atas landasan utama yang dikenal sebagai Asumsi Kekekalan Kecerahan (*Brightness Constancy Assumption*). Asumsi ini menyatakan bahwa intensitas piksel suatu titik objek, yang dinotasikan sebagai $I(x, y, t)$, akan bernilai konstan walaupun titik tersebut bergerak sedikit sejauh (dx, dy) dalam selang waktu dt. Secara matematis, kondisi ini dapat dituliskan sebagai persamaan:

$$I(x, y, t) = I(x + dx, y + dy, t + dt)$$

Untuk mendapatkan hubungan linear, ruas kanan persamaan tersebut diekspansi menggunakan Deret Taylor orde pertama. Dengan mengabaikan suku-suku berorde tinggi (Higher Order Terms) yang dianggap tidak signifikan, persamaan menjadi:

$$I(x, y, t) \approx I(x, y, t) + \frac{\partial I}{\partial x}dx + \frac{\partial I}{\partial y}dy + \frac{\partial I}{\partial t}dt$$

Dengan membagi persamaan hasil penyederhanaan tersebut dengan dt, diperoleh Persamaan Kendala Optical Flow (*Optical Flow Constraint Equation*):

$$I_x u + I_y v + I_t = 0$$

Di mana I_x, I_y adalah gradien spasial citra, I_t adalah gradien temporal, serta $u = \frac{dx}{dt}$ dan $v = \frac{dy}{dt}$ adalah komponen vektor kecepatan aliran optik yang dicari.

Namun, persamaan diferensial parsial di atas menghadirkan masalah ketidaktunggalan solusi yang dikenal sebagai *Aperture Problem*. Hal ini terjadi karena terdapat dua variabel tak diketahui (u dan v) dalam satu persamaan linear tunggal. Untuk mengatasi hal ini, Metode Lucas-Kanade menerapkan pendekatan penyelesaian lokal dengan asumsi bahwa aliran optik (u, v) bersifat konstan dalam sebuah jendela ketetanggaan kecil (Ω) berukuran $n \times n$ di sekitar titik yang ditinjau. Asumsi ini mengubah masalah yang semula kekurangan kendala (*under-constrained*) menjadi sistem persamaan linear yang memiliki lebih banyak persamaan daripada variabel (*overdetermined system*).

Jika sebuah jendela berukuran 3×3 diambil di sekitar piksel pusat, maka akan terbentuk 9 persamaan linear untuk satu pasang (u, v) yang sama. Sistem ini dapat direpresentasikan dalam bentuk matriks $A\mathbf{v} = \mathbf{b}$ sebagai berikut:

$$\underbrace{\begin{bmatrix} I_{x_1} & I_{y_1} \\ I_{x_2} & I_{y_2} \\ \vdots & \vdots \\ I_{y_n} & I_{y_n} \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} u \\ v \end{bmatrix}}_{\mathbf{v}} = \underbrace{\begin{bmatrix} -I_{t_1} \\ -I_{t_2} \\ \vdots \\ -I_{t_n} \end{bmatrix}}_{\mathbf{b}}$$

Di mana A adalah matriks gradien spasial berukuran $n \times 2$, v adalah vektor kecepatan yang dicari, dan b adalah vektor gradien temporal. Karena adanya *noise* pada citra digital, sistem ini umumnya tidak memiliki solusi eksak ($A\mathbf{v} \neq \mathbf{b}$).

Oleh karena itu, solusi optimal diperoleh dengan menggunakan metode Kuadrat Terkecil (*Least Squares*) yang meminimalkan norma kuadrat galat $\left||A\mathbf{v} - \mathbf{b}|\right|^2$. Penyelesaian dilakukan dengan mengalikan kedua ruas dengan transpose matriks A, sehingga menghasilkan persamaan normal:

$$A^T A\mathbf{v} = A^T\mathbf{b}$$

Sehingga solusi akhir untuk vektor kecepatan optik adalah:

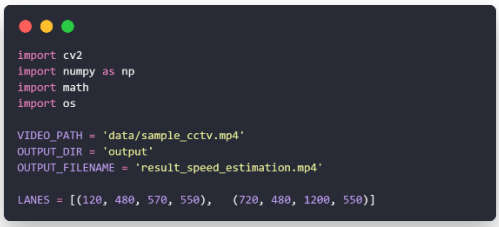
$$\mathbf{v} = (A^T A)^{-1} A^T\mathbf{b}$$

Matriks $A^T A$ dalam persamaan di atas dikenal sebagai matriks struktur tensor atau matriks gradien kedua. Keberhasilan metode ini sangat bergantung pada kondisi matriks tersebut; agar solusi v dapat dihitung, matriks $A^T A$ harus bersifat non-singular yang berarti determinannya tidak boleh bernilai nol.

III. IMPLEMENTASI

A. Bahasa Pemrograman dan Library yang digunakan

Implementasi sistem estimasi kecepatan ini dibangun menggunakan bahasa pemrograman Python. Program memanfaatkan pustaka OpenCV untuk menangani manipulasi citra digital dan menyediakan algoritma inti seperti *Lucas-Kanade*, serta pustaka NumPy untuk melakukan operasi aljabar linier pada struktur data array multidimensi yang merepresentasikan matriks citra dan vektor gerakan. Selain itu, modul standar seperti math digunakan untuk perhitungan skalar jarak Euclidean. Sedangkan modul *os* digunakan untuk pengelolaan *file path* dan interaksi dengan berkas. Seluruh pustaka tersebut diimpor dan dikonfigurasi bersama dengan parameter global sistem seperti lokasi video dan definisi jalur (*region of interest*), sebagaimana ditunjukkan pada Gambar 3.1.



```
import cv2
import numpy as np
import math
import os

VIDEO_PATH = 'data/sample_cctv.mp4'
OUTPUT_DIR = 'output'
OUTPUT_FILENAME = 'result_speed_estimation.mp4'

LANES = [(120, 480, 570, 550),   (720, 480, 1200, 550)]
```

Gambar 3.1 Library Program
Sumber: Dokumen Pribadi

B. Preprocessing dan Region of Interest

Sebelum dilakukan pelacakan fitur, setiap *frame* video yang diambil dari sumber CCTV mengalami tahap pra-pemrosesan. Citra masukan yang awalnya memiliki format warna RGB (*Red-Green-Blue*) dikonversi menjadi citra berskala abu-abu (*Grayscale*) menggunakan fungsi *cv2.cvtColor*. Langkah ini dilakukan untuk menyederhanakan ruang vektor data dari dimensi 3 (warna) menjadi dimensi 1 (intensitas), mengingat persamaan dasar *Optical Flow* hanya bergantung pada gradien intensitas cahaya (I_x, I_y, I_t) dan tidak memerlukan informasi kromatik.

Selanjutnya, untuk meningkatkan efisiensi komputasi dan mengurangi kesalahan deteksi akibat objek di luar jalan, seperti trotoar atau pepohonan, diterapkan teknik *Region of Interest* (ROI). ROI didefinisikan sebagai himpunan koordinat persegi panjang yang merepresentasikan lajur kendaraan. Sebuah *masking matrix* dibuat dengan ukuran yang sama dengan citra input, di mana piksel di dalam area jalur diberi nilai 255 (putih) dan sisanya 0 (hitam). Matrix ini kemudian digunakan sebagai parameter batasan dalam fungsi deteksi fitur, memastikan bahwa aljabar pelacakan hanya dieksekusi pada area jalan yang relevan, seperti terlihat pada Gambar 3.2.

```
while True:
        ret, frame = cap.read()
        if not ret: break

        vis_frame = frame.copy()
        frame_gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

        full_mask = np.zeros_like(frame_gray)
        for coords in LANES:
            cv2.rectangle(full_mask, (coords[0], coords[1]), (coords[2
], coords[3]), 255, -1)
```

Gambar 3.2 *Grayscaling* dan *Masking Matrix*
Sumber: Dokumen Pribadi

C. Shi-Tomasi Feature Detection

Inisialisasi pelacakan dimulai dengan mendeteksi titik-titik fitur yang memiliki karakteristik sudut yang kuat (*corners*). Implementasi menggunakan algoritma Shi-Tomasi yang tersedia pada fungsi *cv2.goodFeaturesToTrack*. Parameter deteksi didefinisikan dalam sebuah struktur data *dictionary*, mencakup *qualityLevel* yang menentukan ambang batas minimum nilai eigen yang diterima, serta minDistance untuk memastikan persebaran fitur tidak terlalu padat. Fungsi ini menerima citra *grayscale* dan masker ROI yang telah dibuat sebelumnya, sehingga sistem hanya mengekstraksi fitur visual yang berada di permukaan jalan, menghindari deteksi salah pada objek latar belakang.

Untuk menjaga stabilitas pelacakan sepanjang durasi video, sistem melakukan deteksi ulang secara periodik setiap 5 *frame*. Namun, penambahan fitur baru dilakukan secara selektif menggunakan operasi aljabar vektor. Program menghitung Jarak Euclidean antara kandidat fitur baru dengan himpunan fitur yang sedang dilacak (p_0) menggunakan fungsi *np.linalg.norm*. Fitur baru hanya akan ditambahkan ke dalam himpunan pelacakan jika jaraknya lebih dari 60 piksel dari fitur terdekat yang sudah ada. Mekanisme ini mencegah redundansi komputasi akibat penumpukan fitur pada satu objek kendaraan yang sama, sebagaimana diimplementasikan pada Gambar 3.3.

```
feature_params = dict(maxCorners=50, qualityLevel=0.2, minDistance=20,
blockSize=7)

p0 = cv2.goodFeaturesToTrack(old_gray, mask=full_mask, **feature_params
)

distances = np.linalg.norm(p0.reshape(-1, 2) - candidate_pt, axis=1)
                        if np.all(distances > 60):
                            filtered_features.append(candidate)
```

Gambar 3.3 *Shi-Tomasi Feature Detection*
Sumber: Dokumen Pribadi

D. Optical Flow Lucas-Kanade Method

Estimasi vektor gerak antar *frame* dilakukan menggunakan fungsi *cv2.calcOpticalFlowPyrLK*. Fungsi ini mengimplementasikan metode Lucas-Kanade secara iteratif untuk menyelesaikan sistem persamaan linear *overdetermined* pada setiap titik fitur yang dilacak. Parameter konfigurasi algoritma didefinisikan dalam *lk_params*, di mana *winSize* = (21, 21) merepresentasikan ukuran jendela ketetanggaan ($n \times n$) yang digunakan untuk membangun matriks gradien $A^T A$. Selain itu, parameter *criteria* menetapkan kondisi penghentian iterasi solver numerik, yaitu ketika perubahan epsilon sangat kecil (0.03) atau telah mencapai batas maksimum iterasi (10), menjamin konvergensi solusi yang efisien.

Fungsi ini menerima citra *grayscale* dari waktu sebelumnya (old_gray) dan waktu sekarang (frame_gray), serta himpunan titik fitur awal (p_0). Keluarannya adalah himpunan titik koordinat baru (p_1) dan status pelacakan (st). Sistem kemudian melakukan penyaringan data dengan hanya menyimpan titik-titik yang memiliki status valid (st = 1), yang berarti solusi konvergen ditemukan untuk titik tersebut. Titik yang gagal dilacak dibuang dari memori untuk menjaga integritas perhitungan kecepatan pada tahap selanjutnya, seperti ditunjukkan pada Gambar 3.4.

```
lk_params = dict(winSize=(21, 21), maxLevel=2, criteria=(cv2.
TERM_CRITERIA_EPS | cv2.TERM_CRITERIA_COUNT, 10, 0.03))

if p0 is not None and len(p0) > 0:
            p1, st, err = cv2.calcOpticalFlowPyrLK
(old_gray, frame_gray, p0, None, **lk_params)

            if p1 is not None:
                good_new = p1[st==1]
                good_old = p0[st==1]
```

Gambar 3.4 Metode Lucas-Kanade
Sumber: Dokumen Pribadi

E. Perhitungan Jarak dan Kecepatan

Tahap akhir dari sistem adalah transformasi data vektor perpindahan piksel menjadi besaran fisik kecepatan kendaraan. Proses ini dilakukan dengan mengiterasi setiap pasangan titik fitur yang valid, yaitu koordinat posisi saat ini (a, b) dan posisi sebelumnya (c, d). Magnitudo perpindahan vektor (Δs_{px}) dihitung menggunakan rumus Jarak Euclidean. Nilai ini kemudian dikonversi ke satuan meter dengan membaginya menggunakan konstanta kalibrasi *PIXELS_PER_METER*, yang telah ditentukan secara empiris bernilai 18 piksel/meter untuk menyesuaikan dengan distorsi perspektif kamera.

Kecepatan kendaraan (v) diturunkan dari hubungan jarak dan waktu, di mana waktu tempuh antar *frame* adalah kebalikan dari *frame rate* video ($1/fps$). Dengan demikian, kecepatan sesaat mula-mula diperoleh dalam satuan meter per detik (m/s) dan kemudian dikonversi ke kilometer per jam (km/jam) menggunakan faktor konversi 3,6, sehingga dirumuskan sebagai:

$$v = \Delta s_{meter} \times fps \times 3.6.$$

Hasil perhitungan kemudian divalidasi dengan logika kondisional. Informasi ini ditampilkan kembali ke layar sebagai anotasi teks dan garis lintasan vektor, memberikan representasi visual *real-time* dari analisis gerak kendaraan, sebagaimana dirangkum dalam Gambar 3.5.

```
for i, (new, old) in enumerate(zip(good_new, good_old)):
                    a, b = new.ravel()
                    c, d = old.ravel()

                    if LIMIT_Y_START < b < LIMIT_Y_END:
                        lane_id = -1
                        for idx, coords in enumerate(LANES):
                            if coords[0] < a < coords[2]:
                                lane_id = idx
                                break

                        if lane_id != -1:
                            dist_px = math.sqrt((a - c)**2 + (b - d)
**2)

                            dist_m = dist_px / PIXELS_PER_METER
                            speed_kmh = dist_m * fps * 3.6

                            if speed_kmh > 10 and speed_kmh < 180:
                                color = LANE_COLORS[lane_id]
                                mask_lines = cv2.line(mask_lines, (
int(a), int(b)), (int(c), int(d)), color, 2)
```

Gambar 3.5 Perhitungan Jarak dan Estimasi Kecepatan
Sumber: Dokumen Pribadi

F. Visualisasi Data

Agar hasil estimasi dapat dipahami secara intuitif oleh pengguna, sistem mengimplementasikan lapisan visualisasi yang informatif di atas citra video asli. Visualisasi ini mencakup penggambaran jejak pergerakan, anotasi kecepatan numerik, dan penandaan area jalur.

Untuk merepresentasikan riwayat pergerakan kendaraan, fungsi cv2.line digunakan untuk menggambar garis penghubung antara posisi lama dan posisi baru fitur yang dilacak. Garis ini diberi warna berbeda sesuai dengan lajur tempat kendaraan berada untuk memudahkan identifikasi visual. Selain itu, informasi kecepatan ditampilkan menggunakan cv2.putText. Agar teks tetap terbaca dengan jelas di berbagai kondisi pencahayaan latar belakang, sebuah background hitam digambar terlebih dahulu di belakang teks sebagai kontras.

Sistem juga menerapkan teknik pencampuran citra untuk menvisualisasikan area deteksi tanpa menghalangi pandangan operator. Garis batas atas dan bawah ROI digambar pada salinan *frame* terpisah, yang kemudian digabungkan dengan *frame* asli menggunakan fungsi cv2.addWeighted. Fungsi ini memberikan efek transparansi sebesar 30% pada garis marka jalur, menciptakan antarmuka yang modern dan tidak mengganggu, sebagaimana ditunjukkan pada Gambar 3.6.

```
cv2.rectangle(vis_frame, (int(a)-5, int(b)-15-text_h-5), (int(a)+
text_w+5, int(b)-15+5), (0,0,0), -1)
                                    cv2.putText(vis_frame, text_str
, (int(a), int(b)-15), font, 0.6, color, 2)
                                    cv2.circle(vis_frame, (int(a),
int(b)), 6, (0, 0, 255), -1)
                                    drawn_text_positions.append((a, b
))

                old_gray = frame_gray.copy()
                p0 = good_new.reshape(-1, 1, 2)

        for idx, coords in enumerate(LANES):
            cv2.rectangle(vis_frame, (coords[0], coords[1]), (coords[
2], coords[3]), LANE_COLORS[idx], 1)

        overlay = vis_frame.copy()
        cv2.line(overlay, (0, LIMIT_Y_START), (vis_frame.shape[1],
LIMIT_Y_START), (0, 255, 255), 1)
        cv2.line(overlay, (0, LIMIT_Y_END), (vis_frame.shape[1],
LIMIT_Y_END), (0, 255, 255), 1)
        vis_frame = cv2.addWeighted(overlay, LINE_OPACITY, vis_frame
, 1 - LINE_OPACITY, 0)
```

Gambar 3.6 Visualisasi Data dan Tampilan
Sumber: Dokumen Pribadi

IV. HASIL DAN ANALISIS

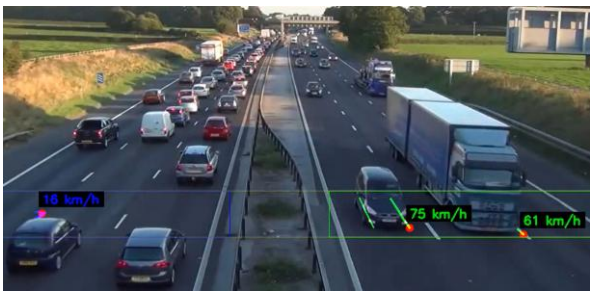
A. Hasil Implementasi

Pengujian sistem dilakukan pada data video CCTV jalan raya dengan menggunakan konstanta kalibrasi PIXELS_PER_METER = 18 yang telah ditentukan sebelumnya. Hasil eksekusi program menunjukkan bahwa integrasi antara metode deteksi fitur *Shi-Tomasi* dan pelacakan *Lucas-Kanade* berhasil menerjemahkan pergerakan piksel menjadi besaran kecepatan fisik.

Seperti terlihat pada Gambar 4.1, sistem mampu membatasi area kerja hanya pada badan jalan melalui penerapan *Region of Interest* (ROI) pada koordinat $y = 480$ hingga $y = 550$. Kendaraan yang melintas di dalam area tersebut secara otomatis ditandai dengan himpunan titik fitur berwarna merah. Stabilitas pelacakan terlihat dari terbentuknya garis jejak (*motion trails*) yang konsisten mengikuti arah laju kendaraan, membuktikan bahwa vektor pergeseran antar *frame* ($\Delta \mathbf{x}$) berhasil diakumulasikan dengan benar.

Selain itu, logika pemisahan lajur berdasarkan koordinat sumbu-x berjalan sesuai rancangan. Kendaraan di lajur kiri dan kanan divisualisasikan dengan warna garis jejak yang berbeda, memudahkan identifikasi visual. Teks anotasi kecepatan yang muncul di atas kendaraan menunjukkan nilai yang responsif terhadap pergerakan objek; nilai

kecepatan muncul saat kendaraan masuk ROI dan menghilang saat keluar, dengan rata-rata estimasi kecepatan berkisar antara 40 km/h hingga 80 km/h, yang merepresentasikan kecepatan wajar pada kondisi lalu lintas tersebut.



Gambar 4.1 Hasil Implementasi Program
Sumber: Dokumen Pribadi

Tabel 4.1 Hasil Pengamatan Estimasi Kecepatan

Sampel Objek	Jalur Posisi	Kecepatan Terukur	Analisis Visual
Mobil Biru Tua	Kiri	16 km/h	Kecepatan kendaraan berkurang dipicu karena adanya kemacetan.
Mobil Hitam	Kanan	75 km/h	Terdeteksi di kecepatan tinggi karena di posisi menyalip truk di sebelahnya
Truk	Kanan	61 km/h	Kecepatan stabil di jalanan lancar

B. Batasan Program

Meskipun program berhasil memberikan estimasi kecepatan secara *real-time*, terdapat keterbatasan fundamental yang mempengaruhi akurasi pengukuran, yaitu masalah distorsi perspektif.

Program saat ini menggunakan pendekatan aljabar linier sederhana di mana jarak piksel dikonversi ke meter menggunakan satu konstanta skalar global ($k = 18$ px/m). Pendekatan ini mengasumsikan bahwa kamera mengambil citra secara tegak lurus. Namun, pada kondisi nyata CCTV, kamera diletakkan dengan sudut kemiringan tertentu sehingga berlaku hukum perspektif, yakni objek yang jauh terlihat lebih kecil dibandingkan objek yang dekat.

Implikasi matematisnya adalah nilai piksel per meter seharusnya tidak konstan, melainkan merupakan fungsi dari koordinat y ($f(y)$). Keterbatasan ini menunjukkan bahwa untuk meningkatkan akurasi pada penelitian selanjutnya, diperlukan penerapan matriks transformasi perspektif (Homografi) untuk memetakan citra ke sudut pandang tegak lurus (*bird's-eye view*) sebelum menghitung jarak Euclidean.

V. LAMPIRAN

Link Github: https://github.com/wafhr/optical-flow-speed-estimation

Dataset yang digunakan: https://www.youtube.com/watch?v=wqctLW0Hb_0

VI. UCAPAN TERIMA KASIH

Puji dan syukur penulis panjatkan ke hadirat Allah SWT. karena atas karunia dan rahmat-Nya penulis dapat tetap sehat dan mampu menyelesaikan makalah ini dengan baik. Selanjutnya, penulis mengucapkan terima kasih kepada Dr. Ir. Rinaldi Munir, M.T. selaku dosen pengampu mata kuliah IF2123 Aljabar Linier dan Geometri yang telah membimbing dan mendukung pematerian sehingga penelitian ini dapat terlaksana. Terakhir, penulis mengucapkan terima kasih kepada kedua orang tua penulis, yang senantiasa mendukung dan mendoakan penulis.

REFERENSI

[1] Munir, Rinaldi, "Sistem persamaan linier (SPL) – Bagian 1," 2025. [Online]. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2025-2026/Algeo-03-Sistem-Persamaan-Linier-2025.pdf [Diakses: 23 Desember, 2025].

[2] Munir, Rinaldi, "Sistem persamaan linier (SPL) – Bagian 2," 2025. [Online]. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2025-2026/Algeo-05-Sistem-Persamaan-Linier-2-2025.pdf [Diakses: 23 Desember, 2025].

[3] Munir, Rinaldi, "Tiga kemungkinan Solusi SPL," 2025. [Online]. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2025-2026/Algeo-04-Tiga-Kemungkinan-Solusi-SPL-2025.pdf [Diakses: 23 Desember, 2025].

[4] Munir, Rinaldi, "Vektor di ruang Euclidean (Bagian 1)," 2025. [Online]. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2025-2026/Algeo-11-Vektor-di-Ruang-Euclidean-Bag1-2025.pdf [Diakses: 23 Desember, 2025].

[5] Munir, Rinaldi, "Nilai Eigen dan Vektor Eigen (Bagian 1)," 2025. [Online]. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2025-2026/Algeo-19-Nilai-Eigen-dan-Vektor-Eigen-Bagian1-2025.pdf [Diakses: 23 Desember, 2025].

[6] OpenCV Team, "Optical Flow," [Online]. Available: https://docs.opencv.org/4.x/d4/dee/tutorial_optical_flow.html [Diakses: 24 Desember, 2025]

[7] GeeksforGeeks, "Corner Detection with Shi-Tomasi Corner Detection Method using OpenCV," 2025. [Online] Available: https://www.geeksforgeeks.org/python/corner-detection-with-shi-tomasi-corner-detection-method-using-opencv [Diakses: 24 Desember, 2025]

[8] Fleet, D. J., & Weiss, Y, "Optical Flow Estimation," [Online] Available: https://www.cs.toronto.edu/~fleet/research/Papers/flowChapter05.pdf [Diakses: 24 Desember, 2025]

[9] Kitani, Kris, "Lucas-Kanade Optical Flow," [Online] Available: https://www.cs.cmu.edu/~16385/s15/lectures/Lecture21.pdf [Diakses: 24 Desember, 2025]

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 24 Desember 2025

Wafiq Hibban Robbany
13524016