# Implementasi Metode Lucas-Kanade dengan *Weighted Least Squares* dalam *Optical Flow*

Aria Judhistira - 13523112[1,2]
*Program Studi Teknik Informatika*
*Sekolah Teknik Elektro dan Informatika*
*Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia*
[1]13523112@std.stei.itb.ac.id, [2]judhistiraaria@gmail.com

***Abstract*—Aliran optik adalah teknik visi komputer yang digunakan untuk memprediksi pergerakan piksel-piksel di antara dua *frame*. Salah satu metode yang kerap digunakan dalam aliran optik untuk mengestimasi gerak dan pelacakan objek adalah metode Lucas-Kanade. Metode ini mencakup berbagai ilmu dalam Aljabar Linear dan Geometri, seperti Sistem Persamaan Linear, vektor gerak, serta nilai dan vektor eigen. Metode Lucas-Kanade yang menggunakan prinsip kuadrat terkecil atau *least square* dapat dibobotkan menggunakan suatu fungsi pembobotan. Dengan fungsi jendela Gaussian, metode ini dapat digunakan untuk fokus pada pergerakan yang terjadi di pusat jendela sehingga secara efektif mengurangi dampak *noise* di area sekitar pusat tersebut.**

***Keywords*—Lucas-Kanade, Weighted, Gaussian**

## I. PENDAHULUAN

*Optical flow*, atau aliran optik, adalah teknik dalam visi komputer yang digunakan untuk memperkirakan pergerakan yang terjadi di antara dua bingkai citra berdasarkan perubahan intensitas piksel atau pola. Konsep ini dicetuskan oleh James J. Gibson pada tahun 1940an sebagai bagian dari teorinya tentang persepsi gerak oleh makhluk hidup. Dalam perkembangannya, aliran optik telah menjadi landasan bagi berbagai aplikasi teknologi, seperti visi komputer, *augmented reality*, pelacakan objek, dan stabilisasi video. Salah satu bentuk aplikasi aliran optik yang sering digunakan adalah untuk estimasi gerak atau *motion estimation*. Estimasi gerak merupakan teknik untuk menganalisis dan memprediksi pergerakan suatu benda dalam suatu rangkaian citra. Teknik ini melibatkan perhitungan suatu vektor gerak yang menggambarkan pergerakan piksel dari satu bingkai ke bingkai berikutnya. Dengan begitu, estimasi gerak memungkinkan komputer untuk melacak suatu benda, mengenali gestur tubuh, dan berbagai hal lainnya yang berhubungan dengan bidang visi komputer.

Salah satu algoritma populer untuk estimasi gerak dengan aliran optik adalah metode Lucas-Kanade. Metode ini dicetuskan oleh Bruce D. Lucas dan Takeo Kanade pada tahun 1980an. Metode ini didasarkan pada asumsi bahwa aliran optik piksel pada suatu *window* adalah konstan dan pergerakan yang terjadi kecil. Hal ini memungkinkan perhitungan vektor gerak menggunakan sistem persamaan linier. Selain itu, metode Lucas-Kanade memungkinkan suatu solusi alternatif menggunakan prinsip kuadrat terkecil atau *least squares* yang dapat dibobotkan. Pendekatan menggunakan prinsip ini dapat menemukan solusi terbaik untuk vektor gerak dan meminimalisasikan kesalahan dalam perhitungan.

Dalam makalah ini, metode Lucas-Kanade untuk estimasi gerak akan dikembangkan dengan pendekatan *weighted least squares*. Pada pendekatan *least squares* konvensional, setiap piksel dalam window pengamatan memberikan kontribusi yang sama terhadap solusi vektor gerak. Namun, dengan adanya pembobotan menggunakan suatu fungsi pembobotan, kontribusi setiap piksel dapat diatur berdasarkan relevansinya terhadap estimasi gerak.. Pendekatan ini tidak hanya memungkinkan komputer untuk memprioritaskan piksel-piksel yang signifikan, tetapi juga meningkatkan efisiensi komputasi dan mengurangi pengaruh variabel yang tidak diinginkan sehingga dapat menghasilkan estimasi gerak yang lebih akurat.

## II. LANDASAN TEORI

### A. Sistem Persamaan Linier

Sistem persamaan linier adalah kumpulan dari satu atau lebih persamaan yang melibatkan variabel-variabel dengan pangkat tidak lebih dari satu. Berikut adalah bentuk umum dari sebuah sistem persamaan linier.

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

Gambar 2.1 Bentuk umum SPL
Sumber: [1]

Sistem persamaan linier juga dapat direpresentasikan menggunakan matriks seperti sebagai berikut.

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

$$A \qquad\qquad x \qquad b$$

Gambar 2.2 Representasi matriks SPL
Sumber: [1]

Dalam representasi matriks tersebut, **A** adalah matriks koefisin berukuran *m x n*, **b** adalah vektor kolom konstanta, dan **x** adalah vektor kolom variabel yang ingin dicari sebagai solusi. Representasi matriks ini dapat selebihnya disederhanakan dalam bentuk *augmented* menjadi satu matriks sebagai berikut.

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

Gambar 2.3 Operasi Baris Elementer
Sumber: [1]

Untuk menemukan solusi dari sebuah sistem persamaan linier, salah satu teknik yang dapat digunakan untuk sistem dengan representasi matriks adalah Operasi Baris Elementer (OBE). OBE dapat menyederhanakan nilai-nilai elemen baris pada suatu matriks tanpa mengubah solusi dari sistem persamaan linier yang diwakilkannya. Terdapat tiga operasi utama terhadap suatu matriks *augmented*, yaitu:

- Pertukaran baris
- Perkalian baris dengan konstanta tidak nol
- Penjumlahan antarbaris

Dengan menggunakan OBE, salah satu metode yang dapat digunakan untuk menemukan solusi dari sebuah sistem persamaan linier adalah metode Eliminasi Gauss. Metode ini melibatkan penggunaan OBE pada suatu matriks *augmented* untuk mengubah matriks **A** ke dalam bentuk baris eselon, atau dalam kata lain, mengubah setiap elemen di bawah *pivot* menjadi 0. Setelah itu, sistem dapat diselesaikan dengan melakukan substitusi mundur untuk mendapatkan nilai variabel **x** yang diinginkan.

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

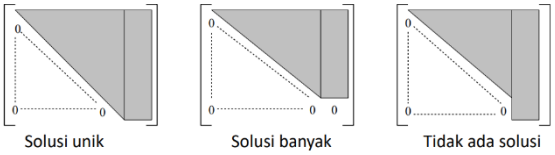
Gambar 2.4 Eliminasi Gauss
Sumber: [1]

Selain metode Eliminasi Gauss, terdapat metode Eliminasi Gauss-Jordan yang menyederhanakan matriks *augmented* lebih jauh. Metode ini melibatkan penggunaan OBE juga, hanya kali ini untuk mengubah matriks **A** ke dalam bentuk baris eselon tereduksi, atau dengan setiap elemen pada kolom *pivot* adalah nol. Setelah didapatkan bentuk tersebut, solusi untuk setiap variabel didapatkan dengan melakukan substitusi mundur

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

Gambar 2.5 Eliminasi Gauss-Jordan
Sumber: [3]

Pada penyelesaian sistem persamaan linier, terdapat tiga kemungkinan solusi yang dapat terjadi.




Gambar 2.6 Tiga kemungkinan solusi SPL
Sumber: [2]

Pertama adalah dihasilkannya solusi unik atau solusi tunggal untuk setiap variabel **x** pada sistem. Kedua, terdapat kasus yang menimbulkan tak terhingga banyaknya solusi yang memenuhi sistem, yakni ketika jumlah persamaan independen pada sistem lebih sedikit daripada persamaan dependen. Terakhir, terdapat kemungkinan bahwa tidak ada solusi atau persamaan yang memenuhi. Sistem persamaan linier yang bersangkutan dapat dinyatakan konsisten jika memiliki setidaknya satu solusi yang memenuhi. Di sisi lainnya, sistem persamaan linier dapat dinyatakan sebagai inkonsisten jika tidak terdapat solusi sama sekali yang berarti persamaan-persamaan dalam sistem tersebut saling bertentangan.

Manfaat dari metode Eliminasi Gauss-Jordan tidak hanya untuk menyelesaikan suatu sistem persamaan linier, tetapi dapat digunakan juga untuk mencari balikan suatu matriks. Balikan dari suatu matriks **A** adalah $\mathbf{A^{-1}}$ sedemikian sehingga nilai dari $\mathbf{AA^{-1}}$ adalah **I** atau berupa matriks identitas. Dengan begitu, untuk setiap matriks berukuran n x n dan dengan determinan bukan nol dapat dicari balikannya dengan mengikuti:

$$[\boldsymbol{A}|\boldsymbol{I}]\sim[\boldsymbol{I}|\boldsymbol{A^{-1}}] \quad \textbf{(1)}$$

Lebih dari itu, sistem persamaan linier dapat diselesaikan juga dengan matriks balikan. Jika terdapat suatu sistem persamaan linier $\boldsymbol{Ax = b}$, maka solusi dari sistem tersebut dapat dinyatakan sebagai $\boldsymbol{x = A^{-1}b}$.

B. Vektor di Ruang Euclidean

Vektor didefinisikan sebagai salah satu representasi kuantitas fisika. Bersama dengan representasi skalar yang hanya menyatakan nilai, vektor yang menyatakan nilai dan arah diaplikasikan pada berbagai bidang ilmu. Ruang vektor, atau disebut juga Ruang Euclidean, adalah sebuah ruang dengan dimensi **n** yang menyatakan jumlah sumbu koordinat dan memiliki titik-titik yang dinyatakan sebagai vektor. Jumlah komponen vektor mengikuti besar dimensi **n** pada ruang vektor $\mathbb{R}^n$.

Vektor memiliki operasi aritmatika yang berbeda dibandingkan nilai-nilai skalar. Penjumlahan vektor melibatkan penjumlahan antarkomponen yang sesuai, sedangkan perkalian vektor dengan nilai skalar melibatkan perkalian setiap komponen vektor dengan nilai skalar tersebut. Adapun panjang atau norma vektor di ruang $\mathbb{R}^n$ yang dihitung sebagai berikut.

$$||\boldsymbol{v}|| = \sqrt{\boldsymbol{v_1^2 + v_2^2 + v_3^2 + \cdots + v_n^2}} \quad \textbf{(2)}$$

Pada suatu ruang Euclidean, jarak suatu vektor dengan suatu vektor lainnya dinamakan sebagai jarak Euclidean. Misal terdapat suatu vektor **u** dan vektor **v** dalam ruang $\mathbb{R}^n$, maka jarak Euclidean antara dua vektor tersebut mengikuti:

$$jarak = ||\boldsymbol{u - v}|| \quad \textbf{(3)}$$

Suatu vektor dapat dinyatakan sebagai hasil penjumlahan dari beberapa vektor lain yang dikalikan dengan skalar. Hal ini disebut sebagai kombinasi linier dari vektor-vektor pada ruang vektor. Syarat suatu vektor dapat dibentuk dari kombinasi linier vektor lainnya adalah vektor tersebut harus berada dalam *span* dari vektor-vektor bersangkutan. *Span* sendiri adalah suatu himpunan semua vektor yang dapat dibentuk sebagai kombinasi linier dari padanya. Selain itu, jumlah komponen vektor harus sesuai dengan dimensi ruang vektor dan vektor-vektor yang akan dikombinasikan tidak bergantung secara linier.

Operasi vektor dalam ruang Euclidean yang paling sering digunakan adalah operasi perkalian antarvektor. Operasi perkalian pertama adalah perkalian titik atau *dot product*. Hasil dari perkalian titik adalah suatu nilai skalar yang disebut sebagai *Euclidean inner product*. Perkalian titik melibatkan penjumlahan dari hasil perkalian antara masing-masing komponen vektor dengan komponen vektor yang bersesuaian. Perkalian titik juga dapat dirumuskan sebagai berikut.

$$\boldsymbol{u} \cdot \boldsymbol{v} = ||\boldsymbol{u}|| ||\boldsymbol{v}|| \cos(\theta) \quad \textbf{(4)}$$

Melalui rumus tersebut, dapat dicari juga nilai sudut $\theta$ antara kedua vektor. Hal ini dapat bermanfaat untuk mencari nilai similaritas kosinus (*cosine similarity*) untuk membandingkan vektor-vektor.

Operasi perkalian vektor kedua adalah perkalian silang (*cross product*). Perkalian silang antara dua vektor menghasilkan suatu vektor baru yang tegak lurus terhadap kedua vektor pembentuknya. Dengan kata lain, perkalian silang secara efektif mampu membentuk vektor normal suatu bidang jika kedua vektor tersebut berada pada bidang yang sama. Berdasarkan Persamaan Lagrange, perkalian silang antara dua vektor dapat dirumuskan sebagai berikut.

$$||\boldsymbol{u} \times \boldsymbol{v}|| = ||\boldsymbol{u}|| ||\boldsymbol{v}|| \sin(\theta) \quad \textbf{(5)}$$

Dengan $\theta$ sebagai sudut di antara kedua vektor. Secara geometris, perkalian silang dapat diaplikasikan untuk mencari volume suatu *parallelpiped*, yakni suatu bangunan ruang tiga dimensi yang dibentuk oleh tiga vektor di ruang $\mathbb{R}^3$.

Dalam suatu ruang vektor, dua vektor dapat dinyatakan sebagai ortogonal jika kedua vektor tersebut saling tegak lurus. Perkalian titik antara kedua vektor tersebut dapat diaplikasikan untuk mencari tahu apakah kedua vektor ortogonal, dengan hasil perkalian titik adalah 0. Vektor-vektor yang saling ortogonal dapat membentuk sebuah himpunan ortogonal. Di sisi lainnya, dua vektor satuan yang saling tegak lurus dapat dinyatakan sebagai ortonormal. Vektor satuan adalah vektor yang memiliki panjang atau norma sebesar 1. Vektor satuan **u** dari suatu vektor **v** dapat dirumuskan sebagai:

$$\boldsymbol{u} = \frac{\boldsymbol{v}}{||\boldsymbol{v}||} \quad \textbf{(6)}$$

Vektor-vektor yang saling ortonormal pun dapat membentuk suatu himpunan vektor ortonormal.

### C. Nilai Eigen dan Vektor Eigen

Vektor eigen dapat didefinisikan sebagai suatu vektor **x** yang tidak nol pada suatu ruang vektor yang arah transformasinya tidak berubah setelah dikenakan transformasi linier yang direpresentasikan oleh suatu matriks **A**. Setiap vektor eigen memenuhi ungkapan sebagai berikut.

$$\boldsymbol{Ax} = \lambda \boldsymbol{x} \quad \textbf{(7)}$$

Dengan **A** adalah matriks persegi yang merepresentasikan transformasi linier dan $\lambda$ adalah nilai eigen yang merepresentasikan faktor skala vektor eigen **x**. Berdasarkan etimologinya, nilai eigen dapat didefinisikan sebagai nilai karakteristik atau nilai asli dari pada matriks **A**.

Vektor eigen yang berkorespondensi dengan suatu matriks **A** dapat dicari melalui sistem persamaan linier sebagai berikut.

$$(\lambda \boldsymbol{I} - \boldsymbol{A})\boldsymbol{x} = 0 \quad \textbf{(8)}$$

Dengan **I** adalah matriks identitas. Berdasarkan sistem di atas sendiri, $\boldsymbol{x} = 0$ merupakan solusi trivial. Agar sistem persamaan linier di atas memiliki suatu solusi tak-nol, maka determinan dari matriks yang berkorespondensi dengan vektor eigen **x** haruslah nol. Dalam kata lain, dibentuk persamaan karakteristik sebagai berikut.

$$det(\lambda \boldsymbol{I} - \boldsymbol{A}) = 0 \quad \textbf{(9)}$$

Persamaan karakteristik ini digunakan untuk mencari nilai eigen. Setelah didapatkan nilai-nilai eigen, dilakukan Eliminasi Gauss pada persamaan awal (n) untuk masing-masing nilai eigen sehingga didapatkan vektor eigen dalam bentuk solusi parametrik.

Vektor eigen yang didapatkan dapat membentuk suatu subruang yang disebut ruang eigen (*eigenspace*) untuk setiap nilai eigen yang didapatkan.

$$E_\lambda = \{\boldsymbol{v} \in \mathbb{R}^n : \boldsymbol{Av} = \lambda \boldsymbol{v}\} \quad \textbf{(10)}$$

Vektor-vektor eigen akan membentuk basis ruang eigen, yakni himpunan vektor eigen yang linier bebas dan dapat merepresentasikan semua vektor pada ruang eigen tersebut.

Perhitungan nilai dan vektor eigen dimanfaatkan dalam diagonalisasi. Diagonalisasi adalah proses mengubah sebuah matriks persegi menjadi matriks diagonal yang ekuivalen. Matriks diagonal sendiri adalah matriks yang elemennya memiliki nilai 0 pada elemen-elemen di luar diagonal utamanya. Suatu matriks **A** dapat didiagonalisasikan jika terdapat juga suatu matriks **P** sehingga terpenuhi ekspresi berikut.

$$\boldsymbol{A} = \boldsymbol{PDP^{-1}} \quad \textbf{(11)}$$

$$\boldsymbol{D} = \boldsymbol{P^{-1}AP} \quad \textbf{(12)}$$

Dengan **D** adalah matriks diagonal. Matriks **P** adalah matriks invertibel yang kolom-kolomnya adalah basis ruang eigen dari matriks **A**. Keuntungan dari proses diagonalisasi adalah memudahkan perhitungan matriks pangkat.

$$\boldsymbol{A^n} = \boldsymbol{PD^nP^{-1}} \quad \textbf{(13)}$$

Dalam hal ini, $\boldsymbol{D^n}$ mudah dihitung juga karena merupakan matriks diagonal.

## III. IMPLEMENTASI

Implementasi kode untuk metode Lucas-Kanade dengan *weighted least squares* dilakukan dengan menggunakan bahasa pemrograman Python 3.12 dan menggunakan perpustakaan atau *library* utama yaitu OpenCV dan NumPy. Perpustakaan OpenCV digunakan untuk melakukan pengolahan gambar, antara lain mengubah gambar menjadi *grayscale* atau dalam skala

abu-abu sehingga lebih mudah diproses oleh komputer, dan membuat gradien. Perpustakaan NumPy digunakan untuk melakukan mayoritas perhitungan rumit, seperti perhitungan matriks untuk sistem persamaan linier, dan memudahkan ekstraksi fitur serta pengolahan setiap gambar.

Pada implementasi *weighted least squares*, pembobotan dilakukan menggunakan jendela Gaussian untuk memberikan kontribusi yang lebih besar pada piksel di sekitar pusat jendela yang biasanya memiliki informasi lebih relevan dalam menghitung vektor aliran optik.

```
def weighted_least_square (window_size):
    sigma = window_size / 12.0
        x = np.linspace(-window_size/2, window_size/2, window_size)
        y = np.linspace(-window_size/2, window_size/2, window_size)
    X, Y = np.meshgrid(x, y)
    window = np.exp(-(X**2 + Y**2) / (2 * sigma**2))
    return window / window.sum()
```

Fungsi *weighted_least_square* ini menghasilkan jendela Gaussian berbasis parameter ukuran jendela dengan standar deviasi ($\sigma$) yang diatur sebagai seperduabelas dari ukuran jendela. Matriks bobot ini diterapkan pada setiap blok jendela dalam perhitungan aliran optik. Berdasarkan fungsi tersebut, pembobotan piksel akan semakin menurun semakin jauh piksel tersebut berada dari pusat jendela sehingga mengurangi dampak *noise* atau pergerakan kecil di area sekitar.

$$\begin{bmatrix} 0.0000442 & 0.0012908 & 0.0039760 & 0.0012908 & 0.0000442 \\ 0.0012908 & 0.0377236 & 0.1161968 & 0.0377236 & 0.0012908 \\ 0.0039760 & 0.1161968 & 0.3579112 & 0.1161968 & 0.0039760 \\ 0.0012908 & 0.0377236 & 0.1161968 & 0.0377236 & 0.0012908 \\ 0.0000442 & 0.0012908 & 0.0039760 & 0.0012908 & 0.0000442 \end{bmatrix}$$

Fungsi pembobotan ini akan dipanggil pada fungsi implementasi metode Lucas-Kanade utama sebagai berikut.

```
def lucas_kanade (frame1, frame2, points, window_size=15):
    …..
    weight = weighted_least_square (window_size)
    .....
    win_Ix = win_Ix * weight
    win_Iy = win_Iy * weight
    win_It = win_It * weight
    …..
```

Pada fungsi *lucas_kanade*, fungsi *weighted_least_square* dipanggil dan hasil jendelanya disimpan dalam variabel *weight*. Variabel ini kemudian dikalikan dengan gradien spasial $\mathbf{I_x}$ dan $\mathbf{I_y}$ serta gradien temporal $\mathbf{I_t}$. Gradien hasil perkalian ini akan digunakan pada perhitungan untuk membangun matriks **A** dan **b** pada sistem persamaan linier.

$$I_x(q)v_x + I_y(q)v_y = -I_t(q)$$

Gambar 3.1 SPL Metode Lucas-Kanade
Sumber: [9]

Untuk memvisualisasikan pergerakan piksel di antara *frame*, digunakan vektor gerak yang menggambarkan arah gerak piksel serta besar pergerakan yang terjadi.

```
def draw_motion_vectors(frame, points, flow, status, scale_factor=3.0):
    motion_frame = frame.copy()
    for point in points:
        x, y = point[0]
        center = (int(x), int(y))
        cv2.circle(motion_frame, center, 3, (0, 255, 0), -1)
    for i, (point, flow_vec) in enumerate(zip(points, flow)):
        if status[i]:
            x, y = point[0]
            u, v = flow_vec

            motion = np.sqrt(u*u + v*v)

            # Lower the motion threshold significantly
            if motion > 0:  # Reduced threshold
                u_scaled = u * scale_factor
                v_scaled = v * scale_factor
                color = (0, 0, 255)
                start_point = (int(x), int(y))
                    end_point = (int(x + u_scaled), int(y + v_scaled))
                cv2.arrowedLine(motion_frame,
                        start_point,
                        end_point,
                        color,
                        2,
                        tipLength=0.3)
    return motion_frame
```

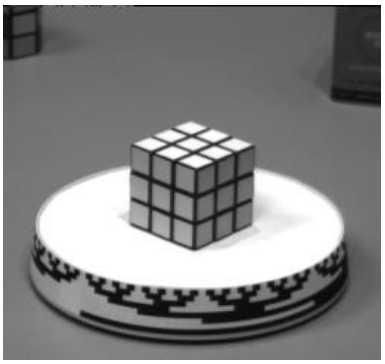
Fungsi *draw_motion_vectors* akan menggambar vektor gerak pada sebuah *frame* berdasarkan hasil perhitungan aliran optik menggunakan metode Lucas-Kanade. Fungsi ini menerima input berupa *frame*, titik-titik awal (*features*), vektor gerak (*flow*), status validitas *flow*, dan skala untuk memperbesar vektor. Setiap titik awal digambar sebagai lingkaran hijau, lalu vektor gerak valid digambar sebagai panah biru yang keluar dari lingkaran yang menunjukkan arah dan panjang perpindahan. Panjang vektor dihitung berdasarkan magnitudo komponen horizontal dan vertikalnya, kemudian diskalakan dengan *scale_factor* agar lebih terlihat. Panah digambar menggunakan fungsi *cv2.arrowedLine*, dengan titik awal pada koordinat awal dan titik akhir sesuai perpindahan vektor. Hasilnya adalah *frame* dengan visualisasi gerak objek berupa gerak vektor yang keluar dari titik-titik fitur.

## IV. Hasil dan Analisis

Hasil implementasi kode kemudian digunakan untuk mengolah sebuah citra GIF (*Graphics Interchange Format*). Citra GIF ini menjadi dasar perbandingan utama performa antara metode Lucas-Kanade saja dan metode Lucas-Kanade dengan pembobotan berupa *weighted least square*. Citra GIF tersebut melibatkan suatu kubus rubik yang berputar di atas suatu piringan bermotif. Citra ini berada dalam skala abu-abu untuk mempercepat komputasi dan pengolahan. Untuk memudahkan perbandingan antara

kedua metode, diambil dua *frame* dari citra GIF yang mengalami pergerakan dan perbedaan yang cukup signifikan. Berikut adalah dua *frame* pada citra GIF yang dimaksud sebagai referensi.

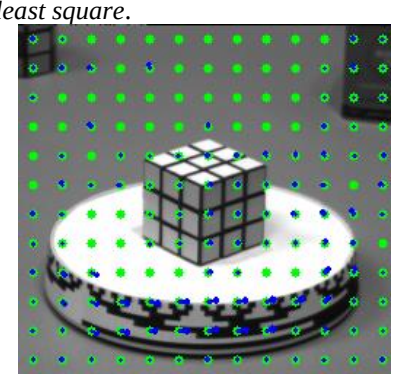
Gambar 4.1 *Frame* pertama GIF
Sumber: [13]



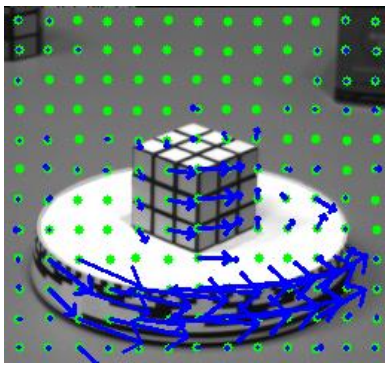
Gambar 4.2 *Frame* kedua GIF
Sumber: [13]

Berdasarkan kedua *frame* ini, program implementasi akan jalan dan menggambarkan vektor gerak yang muncul akibat pergeseran piksel dari *frame* pertama hingga *frame* kedua.

Berikut adalah hasil pengolahan aliran optik menggunakan metode Lucas-Kanade tanpa implementasi *weighted least square*.

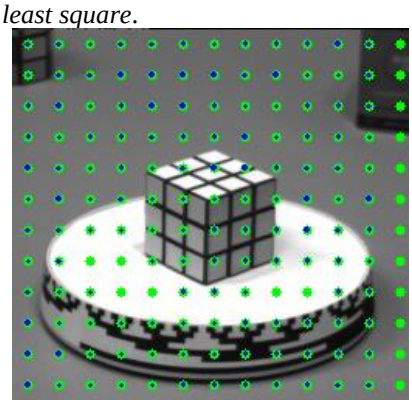


Gambar 4.3 *Frame* pertama pengolahan metode Lucas-Kanade tanpa pembobotan
(Sumber: dokumen penulis)

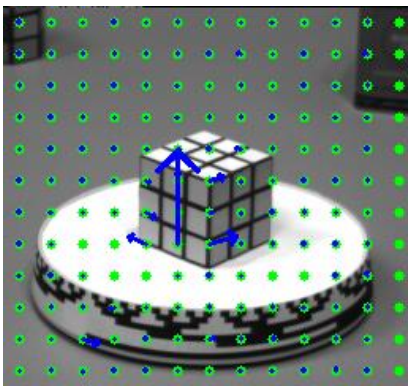


Gambar 4.4 *Frame* kedua pengolahan metode Lucas-Kanade tanpa pembobotan

Berikut adalah hasil pengolahan aliran optik menggunakan metode Lucas-Kanade dengan implementasi *weighted least square*.



Gambar 4.5 *Frame* pertama pengolahan metode Lucas-Kanade dengan pembobotan
(Sumber: dokumen penulis)



Gambar 4.6 *Frame* kedua pengolahan metode Lucas-Kanade dengan pembobotan
(Sumber: dokumen penulis)

Berdasarkan keempat citra tersebut, perbandingan antara implementasi pembobotan dengan *weighted least square* memunculkan perbedaan yang mencolok. Pada pengolahan metode Lucas-Kanade tanpa *weighted least square*, vektor gerak yang muncul akibat perubahan *frame* sangat banyak. Sebagian besar vektor gerak yang muncul berada pada bagian bawah (bagian piringan pada citra), sedangkan tidak banyak vektor gerak yang muncul pada obyek utamanya. Selain itu, vektor gerak pada bagian bawah lebih besar daripada vektor gerak pada kubus rubik sendiri, menandakan bahwa mayoritas pergerakan signifikan terjadi pada bagian piringan.

Di sisi lainnya, setelah mengimplementasi pembobotan dengan *weighted least square* pada metode Lucas-Kanade, jumlah vektor gerak besar yang muncul akibat pegerakan antara dua *frame* menurun secara signifikan. Masih terdapat beberapa vektor gerak signifikan pada bagian bawah/piringan, tetapi sebagian besar vektor gerak pada kubus rubik.

Hal ini sesuai dengan ekspektasi berdasarkan implementasi kode. Adanya pembobotan menggunakan jendela Gaussian membuat bobot piksel-piksel pada pusat jendela lebih besar daripada area di sekitar jendela. Hal ini mengakibatkan besar dan jumlah gerak vektor yang muncul di sekitar pusat jendela menurun secara signifikan. Di sisi lainnya, vektor gerak pada pusat jendela lebih difokuskan dan lebih tampak.

## V. KESIMPULAN

Metode Lucas-Kanade merupakan teknik dalam bidang visi komputer yang kerap digunakan dalam pengolahan aliran optik. Selain itu, metode ini merupakan salah satu metode paling populer untuk estimasi gerak dan pelacakan objek. Berdasarkan rumus metode, metode ini dapat ditambahkan pembobotan pada bagian *least square* oleh suatu rumus pembobotan untuk beberapa kasus berbeda yang membutuhkannya. Salah satu fungsi pembobotan yang dapat diimplementasikan pada metode Lucas-Kanade adalah jendela Gaussian. Fungsi jendela Gaussian membuat piksel-piksel pada pusat jendela lebih signifikan dibandingkan piksel-piksel di sekitarnya.

Sesuai dengan apa yang diharapkan, pembobotan menggunakan fungsi jendela Gaussian membuat piksel di pusat jendela memiliki nilai yang lebih signifikan dibandingkan piksel-piksel di sekitarnya. Hal ini tampak pada percobaan yang dilakukan, di mana tanpa pembobotan, banyak vektor-vektor gerak besar yang muncul dekat bagian bawah jendela citra. Di sisi lainnya, implementasi pembobotan jendela Gaussian membuat vektor-vektor gerak di pusat jendela lebih prominan dibandingkan area sekitarnya. Hal ini sekiranya dapat berguna untuk penelitian yang secara spesifik melacak suatu objek yang berada pada pusat jendela dan ingin mengurangi efek *noise* dari area di sekitar jendela tersebut

## VI. UCAPAN TERIMA KASIH

Puji syukur dan terima kasih saya ucapkan kepada Tuhan Yang Maha Esa karena sebab kuasa dan bimbingan-Nya, makalah ini dapat diselesaikan sesuai dengan tujuannya. Saya juga mengucapkan terima kasih sebesar-besarnya kepada teman-teman dan keluarga saya yang telah memberikan dukungan dalam penyusunan makalah ini. Ucapan terima kasih khusus saya sampaikan kepada dosen pengampuh mata kuliah IF2123 Aljabar Linear dan Geometri Dr. Ir. Rinaldi Munir, M.T yang telah mengajar dan memberi saya pengetahuan berharga dalam penyusunan makalah ini. Semoga makalah ini bisa bermanfaat dan dapat menjadi kontribusi positif bagi ilmu pengetahuan di masa mendatang.

## REFERENSI

[1] R. Munir, "Sistem persamaan linier (Bagian 1: Metode eliminasi Gauss)", IF2123 Aljabar Linier dan Geometri, 2024. [Online]. Tersedia: https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2023-2024/Algeo-03-Sistem-Persamaan-Linier-2023.pdf. [Diakses: 30 Desember 2024].

[2] R. Munir, "Sistem persamaan linier (Bagian 2: Tiga kemungkinan solusi sistem persamaan linier", IF2123 Aljabar Linier dan Geometri, 2024. [Online]. Tersedia: https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2023-2024/Algeo-04-Tiga-Kemungkinan-Solusi-SPL-2023.pdf. [Diakses: 1 Januari 2025].

[3] R. Munir, "Sistem persamaan linier (Bagian 3: Metode eliminasi GaussJordan)", IF2123 Aljabar Linier dan Geometri, 2024. [Online]. Tersedia: https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2023- 2024/Algeo-05-Sistem-Persamaan-Linier-2023.pdf. [Diakses: 30 Desember 2024].

[4] R. Munir, "Vektor di ruang Euclidean (Bagian 1)", IF2123 Aljabar Linier dan Geometri, 2024. [Online]. Tersedia: https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2024-2025/Algeo-11-Vektor-di-Ruang-Euclidean-Bag1-2024.pdf. [Diakses: 31 Desember 2024].

[5] R. Munir, "Vektor di ruang Euclidean (Bagian 2)", IF2123 Aljabar Linier dan Geometri, 2024. [Online]. Tersedia: https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2024-2025/Algeo-11-Vektor-di-Ruang-Euclidean-Bag1-2024.pdf. [Diakses: 31 Desember 2024].

[6] R. Munir, "Vektor di ruang Euclidean (Bagian 3)", IF2123 Aljabar Linier dan Geometri, 2024. [Online]. Tersedia: https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2023-2024/Algeo-13-Vektor-di-Ruang-Euclidean-Bag3-2023.pdf. [Diakses: 31 Desember 2024].

[7] R. Munir, "Nilai Eigen dan Vektor Eigen (Bagian 1)", IF2123 Aljabar Linier dan Geometri, 2024. [Online]. Tersedia: https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2023-2024/Algeo-19-Nilai-Eigen-dan-Vektor-Eigen-Bagian1-2023.pdf. [Diakses: 31 Desember 2024].

[8] R. Munir, "Nilai Eigen dan Vektor Eigen (Bagian 2)", IF2123 Aljabar Linier dan Geometri, 2024. [Online]. Tersedia: https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2023-2024/Algeo-20-Nilai-Eigen-dan-Vektor-Eigen-Bagian2-2023.pdf. [Diakses: 31 Desember 2024].

[9] G. Boesch, "Optical flow – everything you need to know in 2025," viso.ai, Dec. 06, 2024. https://viso.ai/deep-learning/optical-flow/. [Diakses: 1 Januari 2025].

[10] B. D. Lucas & T. Kanade, "An Iterative Image Registration Technique with an Application to Stereo Vision", *Proceedings of Imaging Understanding Workshop*, pp. 121-130, 1981, https://cseweb.ucsd.edu/classes/sp02/cse252/lucaskanade81.pdf. [Diakses 1 Januari 2025].

[11] "Motion Estimation | CloudInary," *Cloudinary*, Oct. 04, 2023. https://cloudinary.com/glossary/motion-estimation. [Diakses 1 Januari 2025].

[12] Baeldung & M. Aibin, "Optical Flow: Lucas-Kanade Method", *Baeldung*, Mar. 18, 2024. https://www.baeldung.com/cs/optical-flow-lucas-kanade-method. [Diakses 1 Januari 2025].

[13] SandipanDey, "Implementing Lucas-Kanade Optical Flow algorithm in Python", Data Science Central, Feb. 28, 2018. https://www.datasciencecentral.com/implementing-lucas-kanade-optical-flow-algorithm-in-python/. [Diakses 1 Januari, 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, 2 Januari 2025

Aria Judhistira - 13523112