DETEKSI JARAK PERPINDAHAN MANUSIA MENGGUNAKAN DETEKSI *OPTICAL FLOW* BERBASIS VIDEO *OFFLINE*

SKRIPSI

untuk memenuhi salah satu persyaratan

mencapai derajat Sarjana S1

Disusun oleh:

M.Abidafi

16524114

Jurusan Teknik Elektro

Fakultas Teknologi Industri

Universitas Islam Indonesia

Yogyakarta

2020

LEMBAR PENGESAHAN

DETEKSI JARAK PERPINDAHAN MANUSIA MENGGUNAKAN DETEKSI *OPTICAL FLOW* BERBASIS VIDEO *OFFLINE*

TUGAS AKHIR

Diajukan sebagai Salah Satu Syarat untuk Memperoleh
Gelar Sarjana Teknik
pada Program Studi Teknik Elektro
Fakultas Teknologi Industri
Universitas Islam Indonesia

Disusun oleh:

M.Abidafi
16524114

Yogyakarta, 25 September 2020

Menyetujui,

Pembimbing

Elvira Sukma Wahyuni, S.Pd., M.Eng.
155231301

LEMBAR PENGESAHAN

SKRIPSI

DETEKSI JARAK PERPINDAHAN MANUSIA MENGGUNAKAN DETEKSI *OPTICAL FLOW* BERBASIS VIDEO *OFFLINE*

Dipersiapkan dan disusun oleh:

M.Abidafi

16524114

Telah dipertahankan di depan dewan penguji

Pada tanggal: 6 November 2020

Susunan dewan penguji

Ketua Penguji : Elvira Sukma Wahyuni, S.Pd., M.Eng. _____

Anggota Penguji 1 : Dwi Ana Ratna Wati, S.T., M.Eng. _____

Anggota Penguji 2 : Almira Budiyanto, S.Si, M.Eng. _____

Skripsi ini telah diterima sebagai salah satu persyaratan untuk memperoleh gelar Sarjana

Tanggal: 1 Desember 2020

Ketua Program Studi Teknik Elektro

Yusuf Aziz Amrulloh, S. T., M. Eng., Ph. D.

045240101

PERNYATAAN

Dengan ini Saya menyatakan bahwa:

1. Skripsi ini tidak mengandung karya yang diajukan untuk memperoleh gelar kesarjanaan di suatu Perguruan Tinggi, dan sepanjang pengetahuan Saya juga tidak mengandung karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis diacu dalam naskah ini dan disebutkan dalam daftar pustaka.
2. Informasi dan materi Skripsi yang terkait hak milik, hak intelektual, dan paten merupakan milik bersama antara tiga pihak yaitu penulis, dosen pembimbing, dan Universitas Islam Indonesia. Dalam hal penggunaan informasi dan materi Skripsi terkait paten maka akan diskusikan lebih lanjut untuk mendapatkan persetujuan dari ketiga pihak tersebut diatas.

Yogyakarta, 25 September 2020

M.Abidafi

KATA PENGANTAR

Bismillahirrahmanirrahim

Assalamu 'alaikum Warahmatullahi Wabarakatuh

Puji syukur penulis panjatkan kepada Tuhan Yang Maha Esa yang telah memberikan rahmat dan karunianya, sehingga penulis dapat menyelesaikan Laporan Tugas Akhir yang berdjudul DETEKSI JARAK PERPINDAHAN MANUSIA MENGGUNAKAN DETEKSI *OPTICAL FLOW* BERBASIS *VIDEO OFFLINE*. Tidak lupa juga shalawat serta salam kita panjatkan pada Nabi Muhammad SAW yang telah membimbing umatnya menjadi pribadi yang lebih baik.

Atas petunjuk dan ridho-Nya jugalah Laporan Tugas Akhir ini dapat diselesaikan dengan baik dan lancar. Kelancaran dalam mempersiapkan dan menyelesaikan Laporan Tugas Akhir ini tidak terlepas dari bantuan berbagai pihak. Oleh karena itu dengan rasa hormat dan terima kasih yang sebesar-besarnya penulis haturkan kepada:

1. Orangtua dan adik-adik yang selalu mendukung dan mendoakan serta memberi nasihat selama proses penelitian dan pembuatan laporan tugas akhir ini hingga selesai.
2. Bapak Yusuf Aziz Amrullah,S.T., M.Sc., Ph.D selaku Ketua Jurusan Teknik Elektro, Universitas Islam Indonesia
3. Ibu Elvira Sukma Wahyuni, S.Pd., M.Eng. selaku Dosen Pembimbing, yang telah memberi bantuan dan pengarahan hingga terselesaikan Laporan Tugas Akhir ini.
4. Dosen dan karyawan program studi Teknik Elektro Universitas Islam Indonesia.
5. Orang-orang terdekat sekaligus sahabat seperjuanganku sejak masa awal perkuliahan Zul, Septian, Doge, Tri, Hasbi, Herdian, Nizam, Agus, anak – anak Pohon Mangga Club. Dan juga Cici.
6. Teman-teman mahasiswa Jurusan Teknik Elektro Universitas Islam Indonesia angkatan 2016.
7. Semua pihak lain yang tidak dapat disebutkan satu persatu yang telah memberikan masukan, dorongan dan semangat dalam menyelesaikan Laporan Tugas Akhir ini.

Wassalamu 'alaikum Warahmatullahi Wabarakatuh

Yogyakarta, 25 September 2020

M.Abidafi

ARTI LAMBANG DAN SINGKATAN

CCTV	= *Closed Circuit* Television
$I(x, y, t)$	= Intensitas cahaya *Optical Flow*
$I(x + dx, y + dy, t + dt)$	= Intensitas cahaya *Optical Flow* bergerak sejauh dx, dy dan dt
I_x	= Gradien intensitas pixel pada sumbu x
I_y	= Gradien intensitas pixel pada sumbu y
I_t	= Gradien intensitas pixel pada ranah waktu
V_X	= Kecepatan pada pixel sumbu x
V_y	= Kecepatan pada pixel sumbu y
V	= Kecepatan pergerakan pada piksel
JS	= Jarak Sebenarnya
JO	= Jarak Menggunakan *Optical Flow*

ABSTRAK

Semakin tingginya jumlah penduduk lansia dalam rumah tangga di Indonesia mengharuskan para keluarga memberikan pengawasan yang sangat ekstra terhadap seseorang yang sudah lansia yang memiliki kekuatan fisik dan ingatan yang semakin melemah. Oleh sebab itu, pengawasan terhadap pergerakan para lansia agar tidak bergerak ketempat yang tidak wajar dan juga berbahaya perlu dilakukan. Sehingga dibutuhkanlah sebuah sistem yang berfungsi untuk memberikan informasi terhadap jarak pergerakan para lansia. Pada penelitian ini dilakukan deteksi jarak perpindahan manusia menggunakan pengolahan citra dengan menggunakan input video *offline* yang diambil dari kamera yang menghasilkan 13 video dengan variasi jarak yang berbeda-beda. Kemudian pada video akan dilakukan konversi dari citra RGB kebentuk *greyscale* untuk mempermudah perhitungan dan visualisasi. Kemudian *Optical Flow* akan mendeteksi pergerakan aliran intensiitas cahaya di tiap *frame* nya. Lalu GMM dan *Kalman Filter* akan mendeteksi pergerakan objek menggunakan nilai aliran yang sudah di olah oleh *Optical Flow*. Kemudian, *Optical Flow* akan menghitung kecepatan pada objek yang sudah ditentukan oleh GMM. *Kalman Filter* akan melakukan *tracking* pada objek di tiap *frame* nya berdasarkan GMM. Berdasarkan nilai kecepatan yang sudah didapatkan, nilai jarak dapat ditentukan dengan melakukan perkalian antara kecepatan dan waktu. Apabila nilai jarak sudah didapatkan di tiap *frame* nya, hitung nilai estimasi jarak dari objek mulai bergerak hingga berhenti. Hasil dari perhitungan jarak perpindahan objek dari seluruh video dengan jarak sebenarnya memperoleh nilai rata-rata *error* sebesar 0.3%.

Kata Kunci : Pengolahan citra, Kamera, *Optical Flow*.

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB 1
PENDAHULUAN

1.1 Latar Belakang Masalah

Berdasarkan data BPS (Badan Pusat Statistik), lansia (lanjut usia) menurut UU Nomor 13 Tahun 1998 adalah seseorang yang telah mencapai usia 60 (enam puluh) tahun keatas. Peningkatan jumlah penduduk lansia memberikan konsekuensi yang tidak sederhana. Persentase jumlah rumah tangga yang dihuni oleh lansia pada tahun 2019 sebesar 27,88 persen [1]. Faktor yang membuat semakin menurunya kekuatan fisik serta ingatan para lansia yang mengharuskan keluarga melakukan pengawasan yang ekstra termasuk mengawasi pergerakan lansia agar bergerak tidak menjauhi batas jarak perpindahan dari tempat yang aman.

Upaya yang dilakukan untuk mengawasi pergerakan para lansia adalah dengan memasang kamera CCTV (*Closed Circuit Television*) pada tempat-tempat yang sering ditempati dan dilalui oleh para lansia. Namun kamera CCTV hanya dapat merekam suatu kejadian saja dan tidak dapat memberikan informasi apapun selain berupa video kejadian. Maka dari itu diperlukan sistem yang dapat memberikan informasi jarak perpindahan manusia melalui kamera CCTV secara otomatis. Hal ini berguna untuk memberikan informasi kepada pengawas terkait jarak pergerakan para lansia agar tidak bergerak melebihi batas jarak dari tempat yang aman.

Berkembangnya teknologi dalam penerapan sensor visual dan disiplin ilmu *image processing* (pengolahan citra) telah menginspirasi berbagai pihak dalam penerapan disipilin ilmu *image processing*. Sebagai contoh peneliti terdahulu yang menggunakan teknik segmentasi objek dengan model HSV (*Hue, Saturation, Value*). Tujuan dari penelitian ini adalah membuat dasar konsep pengenenalan objek serta melakukan estimasi jarak yang akan digunakan untuk aplikasi robot. Apabila objek tersebut telah dideteksi maka bagian selanjutnya adalah melakukan estimasi jarak. Akan tetapi eror yang dihasilkan dari model ini masih termasuk tinggi pada saat pengukuran jarak yang jauh [2].

Pengukuran jarak perpindahan manusia berbasis pengolahan citra digital menggunakan metode yang umum seperti *Backgound Substraction* sebagai metode untuk pendeteksian objek nya masih menghasilkan eror yang tinggi. Oleh karena itu, pada penelitian ini akan dibahas sistem deteksi jarak perpindahan manusia dengan memanfaatkan sistem deteksi *Optical Flow*. Nilai *Optical Flow* adalah kecepatan (piksel/s) dari pergerakan objek yang berada di dalam citra.

1.2 Rumusan Masalah

1. Bagaimana cara mendeteksi jarak perpindahan manusia berbasis *video processing* ?
2. Bagaimana unjuk kerja deteksi jarak perpindahan manusia berbasis *video processing* ?

1.3 Batasan Masalah

1. Penelitian ini menggunakan 1 buah kamera *Hand Phone* untuk merekam objek yang dilakukan dari 1 sisi.
2. Objek yang berada didalam video berjumlah 1 orang.
3. Video yang digunakan ialah secara *offline* (rekaman video).

1.4 Tujuan Penelitian

1. Untuk mengetahui jarak perpindahan manusia berbasis *video processing*.
2. Untuk mengetahui unjuk kerja metode deteksi jarak perpindahan manusia berbasis *video processing.*

1.5 Manfaat Penelitian

Manfaat dari penelitian ini adalah sistem yang dianalisis dapat membantu dalam hal pengawasan melalui kamera pengawas terkait informasi jarak pergerakan lansia agar bergerak tidak menjauhi batas jarak perpindahan dari tempat yang aman. Sehingga hal-hal yang tidak diinginkan dapat terhindari.

BAB 2
TINJAUAN PUSTAKA

2.1 Studi Literatur

Penelitian yang dilakukan oleh Umam dan Sukma Negara pada tahun 2016 berjudul "Deteksi Obyek Manusia Pada Basis Data Video Menggunakan Metode *Background Substraction* dan Operasi Morfologi". Penelitian ini bertujuan untuk menciptakan sebuah sistem yang dapat mendeteksi jumlah objek pejalan kaki yang direkam dalam sebuah video dengan menggunakan metode *Background Substraction* untuk mendeteksi objek pada *background* dengan cara mengubah citra menjadi citra biner. Cara ini dapat menentukan tingkat kepekaan dari *pixel background* pada video. Selanjutnya Operasi Morfologi digunakan untuk menghilangkan *noise* yang dapat menghalangi keberhasilan dalam deteksi pada *video.* Berdasarkan dari hasil pengujian, program yang telah dirancang dalam penelitian ini dapat mendeteksi dan menghitung objek pejalan kaki dengan sangat baik pada saat kondisi pencahayaan yang baik. Pada saat kondisi pencahayaan yang sedikit redup dan didalam ruangan, program tidak dapat mendeteksi objek dengan baik [3]. Untuk itu, hal ini sangat berkaitan dengan penelitian yang penulis lakukan dalam hal pendeteksian objek yaitu menggunakan Operasi Morfologi di dalamnya.

Pada tahun 2018, Raditya Faisal Waliulu melakukan penelitian yang bertujuan untuk mendeteksi objek dan penggolongan berbagai macam kendaraan dengan menggunakan metode *Kalman Filter* dan *Gaussian Mixture Model* untuk memisahkan *Background* dan *Foregorund* pada video. *Foreground* di-*filter* dengan segmentasi *Bit Large Object* untuk mendapatkan dimensi dari kendaraan tersebut. Apabila objek yang bergerak tidak terdeteksi pada *frame*, maka segmentasi *Bit Large Object* melanjutkan perhitungan pada *frame* berikutnya. Hasil dari ekstraksi ciri dari kendaraan tersebut berdasarkan dimensi piksel kendaraan yang terdeteksi. Hasil segmentasi yang didapatkan akan digunakan oleh *Kalman Filter* untuk menghitung pelacakkan posisi objek yang bergerak. Pengambilan data yang dilakukan pada malam hari dengan pencahayaan dari lampu jalan, hanya kendaraan yang yang berwarna cerahlah yang dapat terdeteksi. Karena kendaraan dengan berwarna gelap memiliki warna yang sangat mirip dengan warna *background* nya [4]. Hal ini berkaitan dengan penelitian yang penulis lakukan karena salah satu metode untuk menandai objek yang bergerak, penulis menggunakan metode *Kalman Filter* dan *Gaussian Mixture Model* di dalamnya.

Penelitian tentang perhitungan jumlah dan kecepatan kendaraan melalui video rekaman CCTV yang dilakukan oleh Firdaus dan Imelda pada tahun 2018 adalah penelitian yang

menggunakan metode *Gaussian Blur* dan *Absolute Difference*. Proses deteksi kendaraan diawali dengan proses *Preprocessing* yang bertujuan untuk mengatur *Rest Of Interest (ROI)* sehingga dapat ditentukanya batas wilayah dalam perhitungan. Tahap selanjutnya adalah melakukan segmentasi menggunakan metode *Gaussian Blur, Absulte Diffrenece, Morfologi, Dilate* dan metode *Blob Detection.* Untuk tahap selanjutnya yaitu melakukan ektraksi ciri. Setelah dilakukanya ekstraksi ciri, dilakukan proses perhitungan jumlah kendaraan untuk mendapatkan urutan kendaraan. Tahap terakhir dari penilitian ini adalah menghitung kecepatan kendaraan. Informasi yang dihasilkan dari perhitungan objek blob yang terdeteksi adalah jarak, waktu dan kecepatan pada setiap kendaraan yang terdeteksi. Untuk akurasi dari perhitungan jumlah kendaraan yaitu mencapai 100%. Untuk perhitungan kecepatan yang terdeteksi dan sebenarnya memiliki selisih terkecil yaitu sebesar 5,11% [5]

Studi berikutnya adalah tentang deteksi kecepatan kendaraan menggunakan *Euclidean Distance* untuk perhitungan jarak yang dilakukan oleh Setiyono dkk. Studi ini memeriksa sudut kamera pada video yang diakuisisi terhadap akurasi estimasi kecepatan. Sehingga perkiraan akurasi kecepatan akan jadi lebih baik. Untuk pemisahan *background* dan *foreground* menggunakan metode *Gaussian Mixture Model.* Untuk mendapatkan nilai kecepatan, dilakukan dengan perbandingan jarak perpindahan dan jumlah *frame per second (fps).* Hasil dari penelitian ini mendapatkan akurasi sebesar 99,38% [6]

Pada penelitian berikutnya adalah pelacakan objek mobil yang kemudian menghitung jumlah mobil yang melintas di jalan. Penelitian yang dilakukan oleh Supriyatin dan Widya Ariestya ini menggunakan metode *Optical Flow.* Metode *Optical Flow* merupakan metode yang bisa mendapatkan estimasi vector gerak di setiap pergereakan *frame* dari *frame* awal hingga *frame* terakhir. Objek yang terdeteksi oleh blok program kemudian diubah menjadi citra biner. dalam penelitian ini menggunakan *tools Simulink Matlab* untuk dilakukanya analisis serta melihat hasil dari pengujianya. Objek yang digunakan dalam penelitian ini adalah kendaraan mobil dengan menggunakan kamera diam dan bergerak. Hasil dari penelitian adalah pelacakan pada jumlah mobil yang sedang melintas dengan menggunakan kamera bergerak tidak hanya menangkap objek mobil yang bergerak, akan tetapi juga menangkap objek seperti marka jalan, pohon, bukit, jalan hingga pembatas jalan. Akan tetapi pada saat menggunakan kamera yang tidak bergerak, pendeteksian yang dihasilkan lebih baik dan cukup efektif untuk pendeteksian mobil yang bergerak. Karena pada saat kamera diam, objek lain seperti marka, pohon, bukit, jalan, pembatas jalan tidak ikut terdeteksi [7]

Penelitian yang dilakukan oleh Lugianti dkk tentang "Deteksi Kecepatan Kendaraan Bergerak Berbasis Video Menggunakan Metode *Frame Difference"* merupakan upaya untuk menertibkan perilaku berbahaya pengendara sepeda motor. Untuk itu dibangun sebuah sistem

yang bertujuan untuk mengawasi para pengguna lalu lintas. Metode *Frame Difference* ini bekerja dengan cara membandingkan antar *frame* untuk memperoleh informasi ada atau tidaknya sebuah pergerakan dari objek. Dari hasil deteksi pergerakan tersebut bisa didapatkan nilai kecepatan dari suatu kendaraan. Tingkat akurasi dari pengukuran kecepatan pada penelitian itu sangat dipengaruhi oleh banyaknya *noise* atau objek asing pada *background* dan pra pengolahan video sebelum masuk ke pengolahan sistem. Maka dari itu, akurasi yang dihasilkan dari penggunaan metode ini adalah sebesar 76,67% [8]

Studi berikutnya adalah pengukuran kecepatan kendaraan berbasis deteksi gerak untuk sistem transportasi cerdas yang dilakukan oleh Ali Tourani pada tahun 2019. Metode yang digunakan untuk mendeteksi kecepatan berdasarkan dari deteksi pergerakan. Objek yang bergerak kemudian terdeteksi ketika memasuki ROI dengan menggunakan metode *Mixture of Gaussian* untuk pengurangan *background*. Selanjutnya diteruskan oleh operasi morfologi. Untuk pengukuran kecepatan pada kendaraan, kendaraan yang terdeteksi kemudian diproses menggunakan algoritma Blob dan jarak perpindahan kendaraan di perbandingan dengan *frame* sebelumnya. Metode ini memiliki akurasi sebesar 94% untuk deteksi kendaraan dan 94,8% untuk pengukuran kecepatan kendaraan [9].

Setelah memaparkan berbagai macam metode seperti di atas, maka peneliti menggunakan metode *Optical Flow* untuk mendeteksi jarak perpindahan manusia. Metode *Optical Flow* pada dasarnya dapat digunakan untuk menghitung kecepatan perpindahan objek berdasarkan *frame.* Dengan demikian, metode *Optical Flow* sangat relevan dan dapat digunakan dalam penelitian ini, sebab salah satu parameter untuk mendeteksi jarak perpindahan objek adalah kecepatan.

2.2 Tinjauan Teori

2.2.1 *Gaussian Mixture Model (GMM)*

GMM merupakan sebuah metode yang berfungsi untuk memodelkan warna – warna *background* dari masing – masing pikselnya. Setiap pikselnya dikelompokkan berdasarkan dari distribusi yang sudah dianggap paling efektif untuk model *background* nya. Apabila nilai standar deviasinya semakin besar, semakin besar juga lebar distribusi kernel *Gaussian* maka, semakin kuat hasil penghalusan yang terjadi pada citra. Algoritma GMM ini masih dapat dikatakan kurang sempurna untuk hasil segmentasinya, sehingga masih terdapat banyak *noise* pada objek. Jadi diperlukan penambahan suatu filter untuk menghilangkan *noise* tersebut yaitu operasi morfologi, sehingga hasil segmentasinya akan jauh lebih baik [4]. Operasi morfologi adalah teknik dari sebuah pengolahan citra digital yang menggunakan bentuk objek sebagai pedoman untuk pengolahanya. Operasi ini bergantung kepada urutan dari kemunculan masing – masing pikselnya. Pada operasi morfologi terdapat berbagai macam proses didalamnya, yaitu operasi *opening*, *closing* dan *filling*. Operasi *opening* berfungsi untuk menghilangkan objek – objek kecil yang bertujuan untuk menghaluskan batas dari area yang besar tanpa mengubah area objeknya secara signifikan. Operasi *closing* bertujuan untuk menutup celah pada objek. Kemudian operasi *filling* yang bertujuan untuk menghaluskan kedua proses dengan cara melakukan pengisian di area lubang untuk menghasilkan segmentasi yang lebih baik. Operasi morfologi ini sangat sesuai jika digunakan untuk dilakukanya pengolahan *binary image* dan *grayscale image* [3].

2.2.2 *Kalman Filter*

Kalman Filter merupakan sebuah sistem yang memiliki kemampuan untuk memprediksi apa yang akan terjadi dan akan melakukan analisa terhadap korelasi dari berbagai data yang mungkin tidak berhubungan. *Kalman Filter* terbagi menjadi 2 aspek, yaitu prediksi dan koreksi terhadap error pada hasil deteksi [4]. Untuk proses prediksi menggunakan persamaan berikut.

$$X_k = A \cdot X_{k-1} + B \cdot U_{k-1} \quad (2.1)$$

$$P_{\bar{k}} = A \cdot P_{k-1} \cdot A^T + Q \quad (2.2)$$

X_k merupakan sebuah vektor dari sebuah proses dari suatu keadaan dalam waktu *k*. Sedangkan X merupakan sebuah vektor 4 dimensi dari (x, y, dx, dy), dengan *x* dan *y* merupakan posisi dari pusat objek. Sedangkan dx dan dy mewakili dari kecepatan. X_{k-1} merupakan vektor dari proses sebuah kondisi dalam waktu $k-1$. A merupakan sebuah proses dari matriks transisi 4 x 4. U_k merupakan suatu vektor kendali dan B terhubung secara opsional dengan vektor kendali U_k dalam kondisi suatu ruang. $P_k$merupakan sebuah kovariansi *error* pada waktu *k*.

Sedangkan P_{k-1} adalah suatu matriks yang mewakili kovarian *error* dalam kondisi prediksi pada saat waktu $k-1$. Dan Q merupakan proses dari kovarian derau [10].

Untuk proses koreksi dapat kita lihat pada persamaan berikut.

$$K_k = P_{\bar{k}} \cdot H^T (H \cdot P_{\bar{k}} H^T + R)^{-1} \quad (2.\ 3)$$

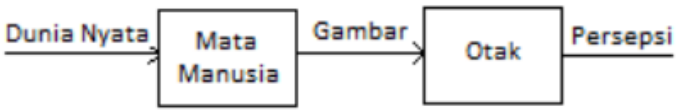
$$X_k = X_{\bar{K}} + K_k \cdot (Z_k - H \cdot X_{\bar{k}}) \quad (2.\ 4)$$

$$P_k = (1 - K_k \cdot H) \cdot P_{\bar{k}} \quad (2.\ 5)$$

K_k merupakan penguat dari *kalman. H* adalah matriks yang dapat merubah kondisi dari ruang pengukuran dan *R* merupakan kovarian pengukuran derau. X_k merupakan suatu proses dari kondisi sebenarnya. Dengan menggunakan K_k dan juga pengukuran Z_k, sehingga kondisi dari proses X_k dapat dilakukanya pembaruan. Z_k yang paling memungkinkan adanya posisi *x* dan *y* dari objek yang berada di dalam *frame*. Langkah terakhir dari metode *Kalman Filter* adalah memperbarui kovarian *error* P_k yang dapat kita gunakan menggunakan persamaan (2. 5) [10].

2.2.3 *Video Processing*

Video processing merupakan sebuah cabang ilmu yang merupakan bagian dari sebuah bidang sistem *computer vision*. *Video* pada dasarnya merupakan sekumpulan citra yang dijalankan secara sekuensial dengan *frame rate* tertentu. *Computer vision* merupakan sebuah pembelajaran untuk menganalisis gambar dan *video* untuk mendapatkan hasil sebagaimana yang bisa dilakukan manusia. *Vision* dapat diartikan sebagai sebuah proses pengamatan apa yang ada pada dunia nyata melalui panca indra penglihatan manusia. computer vision mencoba meniru cara kerja sistem visual manusia. Manusia melihat objek dengan indra penglihatan (mata), lalu citra objek diteruskan ke otak untuk diinterpretasi sehingga manusia bisa dan mengerti objek apa yang sedang tampak dalam pandangan matanya [3]. Hasil interpretasi ini bisa digunakan untuk pengambilan sebuah keputusan (misalnya menghindar kalau ada jalan yang berlubang). Ilustrasi pada *computer vision* dapat dilihat pada Gambar 2. 1.



Gambar 2. 1 Ilustrasi *computer vision* [3].

2.2.4 Citra Digital

Citra digital merupakan sebuah representasi (gambaran), kemiripan dari sebuah objek. Informasi yang diberikan adalah representasi dari komponen-komponen seperti warna, tepian dan tingkat kecerahan. Citra digital terbentuk dari sekumpulan titik yang bernama piksel (*pixel*

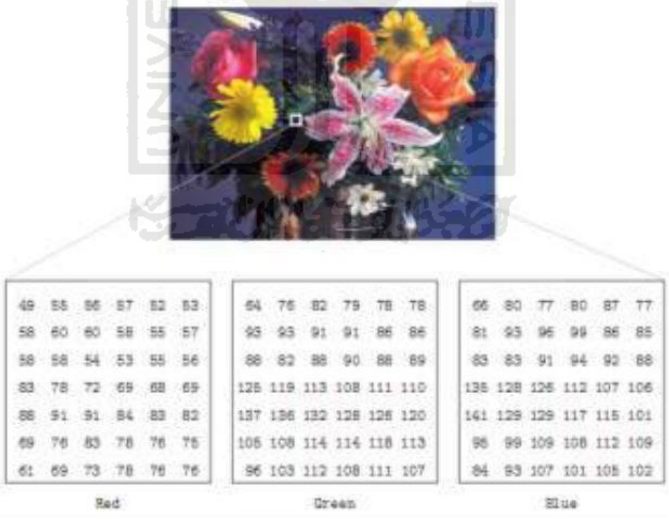
atau *picture element).* Masing-masing piksel dari sebuah citra direpresentasikan oleh data biner [11]. Citra dikelompokkan menjadi dua macam yaitu citra diam (*still image*) dan citra bergerak (*moving image*). Citra diam adalah citra yang tidak bergerak sedangkan citra bergerak merupakan rangkaian citra diam yang ditampilkan beruntun sehingga terlihat seperti gambar yang bergerak. Setiap citra didalam sebuah rangkaian biasa disebut dengan *frame* [8]. Citra dapat dinyatakan dalam bentuk fungsi *f(x,y),* dimana *x* dan *y* merupakan titik dari koordinat spasial. Dan nilai fungsi *f* merupakan nilai intensitas cahaya yang mana merupakan representasi dari warna cahaya yang ada pada citra kontinu atau bisa disebut derajat keabuan [12].

2.2.5 Jenis Citra Digital

Aplikasi dari pengolahan citra digital pada umumnya terbagi menjadi 3 jenis yaitu, citra RGB, citra *grayscale*, dan citra biner.

- Citra RGB (*Red, Green, Blue).*

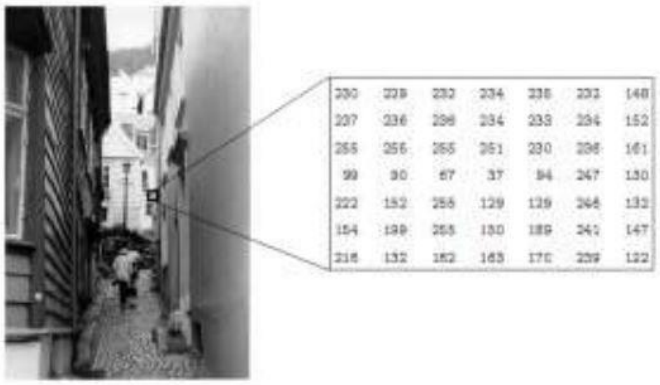
  Citra RGB terdiri dari tiga warna di masing – masing pixelnya, seperti pada Gambar 2. 2 yaitu warna merah (*Red),* hijau *(Green),* dan biru *(Blue).* Dari masing –masing warna memiliki range 0 – 255. Jadi apabila di totalkan sekitar 16 juta variasi warna yang berbeda pada gambar [2].



Gambar 2. 2 Representasi Citra RGB [2].

- Citra *Grayscale.*

  Citra digital *black and white (grayscale)* seperti pada Gambar 2. 3 memiliki gradasi yang berwarna mulai dari putih hingga hitam di setiap pikselnya. Dari masing – masing piksel dapat diwakili oleh 8 bit, atau sama dengan 1 byte. Biasanya jenis citra ini digunakan di dalam kedokteran (X-ray) [2].



Gambar 2. 3 Representasi Citra Grayscale [2].

- Citra Biner.

  Dari masing – masing pikesl dalam citra biner ini hanya terdiri dari warna hitam dan putih seperti Gambar 2. 4. Karena citra ini hanya memiliki dua warna dalam setiap pikselnya, jadi hanya perlu 1 bit per pikselnya (0 dan 1) atau apabila dalam 8 bit (0 dan 255). Sehingga citra biner ini lebih efisien dalam hal penyimpanan. Penggunaan citra ini biasanya sangat cocok dalam hal teks cetak atau tulis tangan, sidik jari dan juga gambar arsitektur [2].

Gambar 2. 4 Representasi Citra Biner [2].

2.2.6 *Optical Flow*

Optical Flow adalah aliran pergerakan dari sebuah objek yang bergerak dalam *gray mode*. Perubahan dari *gray value* dinyatakan oleh bidang *Optical Flow* bisa digunakan untuk mengamati informasi sebuah target yang bergerak di dalam citra. Informasi yang diberikan berisi vektor kecepatan sesaat titik citra pada saat bergerak dari target. *Optical Flow* untuk pertama kalinya diusulkan oleh Gibson. Pada dasarnya perhitungan *Optical Flow* pada objek yang bergerak berdasarkan dua asumsi : yang pertama, *gray value* suatu objek tidak dirubah dalam waktu yang sangat singkat atau aliran/flow atau intensitas cahaya pada pixel yang bergerak ke

frame selanjutnya adalah konstant. Yang kedua, memiliki asumsi bahwa pergerakan piksel adalah sangat kecil [13].

Apabila titik P di dalam sebuah koordinat citra (x, y) bergerak ke $(x + dx, y + dy)$ selama *dt*, maka nilai *gray value* nya adalah $I(x, y, t)$ pada (x, y) dan $I(x + dx, y + dy, t + dt)$ pada $(x + dx, y + dy)$. Karena *dt* pendek, hal ini dapat dilihat pada persamaan (2. 1)

$$I(x + dx, y + dy, t + dt) = I(x, y, t) \tag{2. 6}$$

Keterangan :

$I(x, y, t)$ = Intensitas cahaya *Optical FLow*

$I(x + dx, y + dy, t + dt)$ = Intensitas cahaya *Optical FLow* yang bergerak sejauh dx, dy dan dt

Untuk asusmsi bahwa pergerakan piksel adalah sangat kecil, maka persamaan (2. 6) dapat dikembangkan menjadi:

$$I(x + dx, y + dy, t + dt) = I(x, y, t) + \frac{\partial I}{\partial x}\frac{dx}{dt} + \frac{\partial I}{\partial y}\frac{dy}{dt} + \frac{\partial I}{\partial t}\frac{dI}{dt} \tag{2. 7}$$

Setelah dilakukanya penyederhanaan, persamaan *Optical Flow* (2. 7) disederhanakan ke persamaan (2. 8).

$$\frac{\partial I}{\partial x}\frac{dx}{dt} + \frac{\partial I}{\partial y}\frac{dy}{dt} + \frac{\partial I}{\partial t} = 0 \tag{2. 8}$$

Dengan komponen $V_x = dx/dt$ dan $V_y = dy/dt$ merupakan kecepatan gerak dari piksel pada titik piksel spasial citra (*x*, *y*) dalam arah horizontal dan vertical. (V_x, V_y) dapat disebut juga dengan bidang *Optical Flow*. Persamaan dasar dari intensitasi cahaya *Optical Flow* dapat dinyatakan dalam persamaan berikut :

$$I_x V_x + I_y V_y + I_t = 0 \tag{2. 9}$$

Apabila persamaan disederhanakan menjadi :

$$I_x V_x + I_y V_y = - I_t \tag{2. 10}$$

Keterangan :

I_x = Gradien intensitas pixel pada sumbu x

I_y = Gradien intensitas pixel pada sumbu y

I_t = Gradien intensitas pixel pada ranah waktu

V_x = Kecepatan pada pixel sumbu *x*

V_y = Kecepatan pada pixel sumbu *y*

Persamaan (2. 10) merupakan persamaan akhir dari *Optical FLow* untuk mendapatkan kecepatan piksel pada sumbu X dan sumbu Y [13] .

2.2.7 *Lucas – Kanade Optical FLow*

Untuk bisa memperoleh asumsi bahwa bidang dari *Optical Flow* yang disebabkan dari objek yang bergerak. Metode ini menggunakan asumsi vector pada sebuah piksel sama dengan piksel yang berdekatan [14]. *Lucas – Kanade* berfungsi untuk mengukur posisi, arah, dan kecepatan gerak pada masing – masing piksel disetiap pergerakan *frame*.

$$\begin{aligned} I_x(q_1)V_x + I_y(q_1)V_y &= -I_t(q_1) \\ I_x(q_2)V_x + I_y(q_2)V_y &= -I_t(q_2) \\ &\vdots \\ I_x(q_n)V_x + I_y(q_n)V_y &= -I_t(q_n) \end{aligned} \tag{2. 11}$$

q_n merupakan piksel yang ada di dalam sebuah *frame.* Sedangkan $I_x(q_n), I_y(q_n), I_t(q_n)$ adalah hasil dari turunan parsial dari sebuah gambar yang berhubungan dengan posisi *x*, *y* dan *t*. Persamaan tersebut dapat ditulis ke dalam bentuk matriks

$$Av = (-b)$$

$$A = \begin{bmatrix} I_x(q_1) & I_y(q_1) \\ \vdots & \vdots \\ I_x(q_n) & I_y(q_n) \end{bmatrix}, v = \begin{bmatrix} V_x \\ V_y \end{bmatrix}, \quad b = \begin{bmatrix} -I_t(q_1) \\ -I_t(q_2) \\ \vdots \\ -I_t(q_n) \end{bmatrix} \tag{2. 12}$$

Pada penelitian ini parameter yang dibutuhkan adalah nilai *velocity* dari sebuah objek yang terdeteksi, maka persamaan yang dibutuhkan adalah nilai V_x dan V_y sehingga persamaan menjadi (2. 13).

$$v = (A^T A)^{-1} A^T (-b) \tag{2. 13}$$

Persamaan apabila dalam bentuk matriks menjadi :

$$\begin{bmatrix} V_x \\ V_y \end{bmatrix} = \begin{bmatrix} \sum_i I_x(q_i)^2 & \sum_i I_x(q_i) I_y(q_i) \\ \sum_i I_x(q_i) I_y(q_i) & \sum_i I_x(q_i)^2 \end{bmatrix}^{-1} \begin{bmatrix} -\sum_i I_x(q_i) I_t(q_i) \\ -\sum_i I_y(q_i) I_t(q_i) \end{bmatrix} \quad (2.\ 14)$$

Dengan menggunakan persamaan (2. 14) maka nilai V_x dan V_y dapat diketahui, sehingga dapat dimasukkan ke dalam persamaan berikut (2. 15) [15]. Untuk mencari nilai jarak dari pergerakan objek dapat menggunakan persamaan (2. 16).

$$V = \sqrt{{V_x}^2 + {V_y}^2} \quad (2.\ 15)$$

$$S = V * t \quad (2.\ 16)$$

Keterangan :

V = Kecepatan pergerakan pada piksel

V_x = Kecepatan pada piksel sumbu *x*

V_y = Kecepatan pada piksel sumbu *y*

S = Jarak pergerakan piksel

t = waktu

2.2.8 Metode Evaluasi

Evaluasi pada penelitian adalah tercapainya proses hasil deteksi perhitungan jarak perpindahan manusia dengan menggunakan sistem deteksi dari *Optical Flow* pada video masukan yang memiliki variasi jarak dan arah dengan membandingkan hasil perhitungan *Optical Flow* dan pengukuran jarak secara manual dengan menggunakan meteran untuk memperoleh persentasi nilai erornya.

- Rumus perhitungan eror :

$$\frac{(JS - JO)}{JS} * 100\% \quad (2.\ 17)$$

Keterangan :

JS = Jarak Sebenarnya

JO = Jarak Menggunakan *Optical Flow*

- Rata – rata eror :

$$\frac{(Total\ eror)}{Jumlah\ data} \quad (2.\ 18)$$

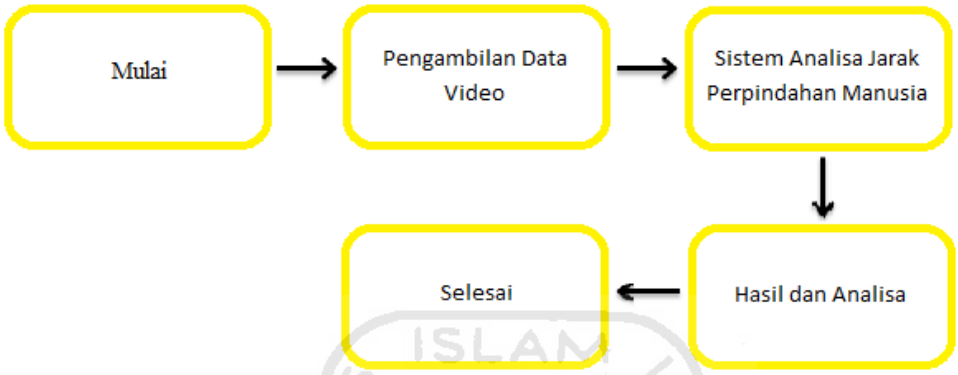
- Rata – rata akurasi :

$$100\% - (Rata - rata\ eror) \quad (2.\ 19)$$

BAB 3
METODOLOGI

3.1 Alur Penelitian

Pada penelitian yang dilakukan, proses untuk mendapatkan jarak perpindahan manusia dapat dilihat pada diagram blok alur penelitian pada Gambar 3. 1.



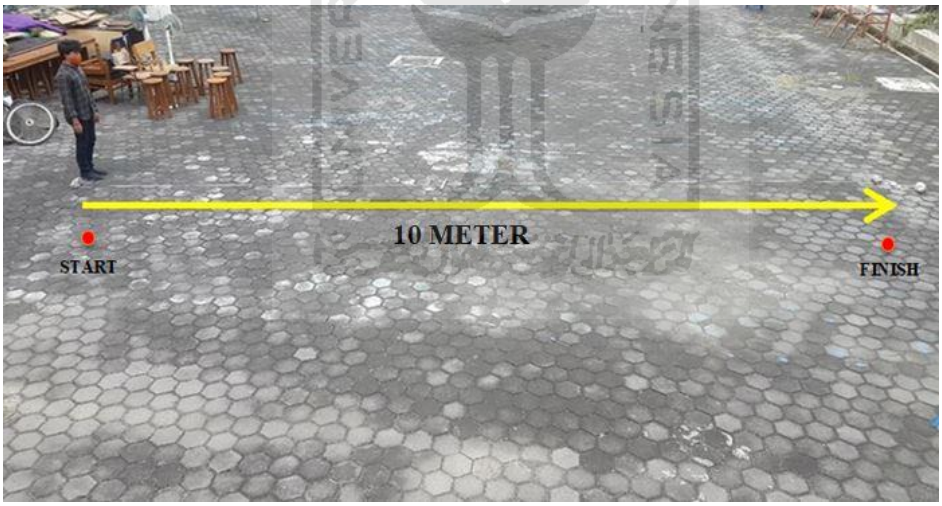
Gambar 3. 1 Diagram blok alur penelitian

3.1.1 Pengambilan Data Video

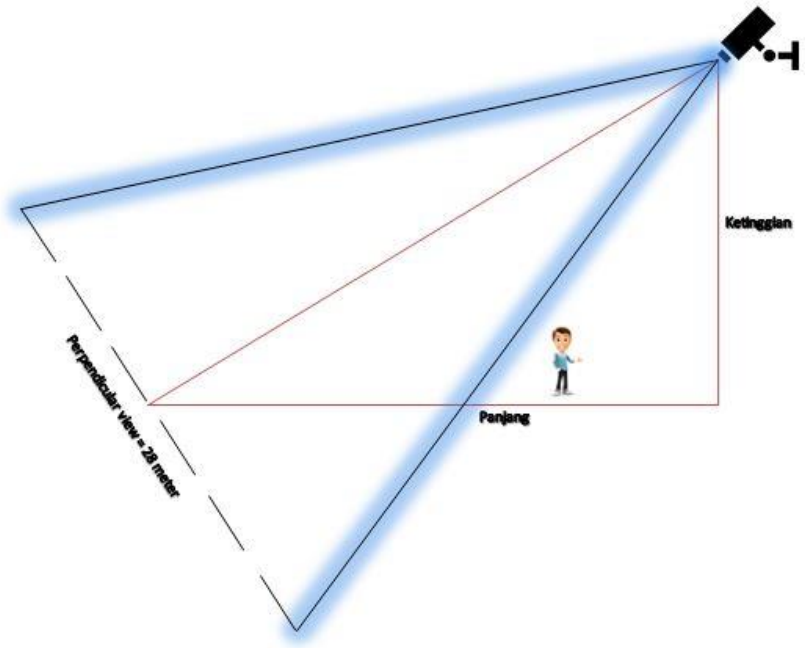
Tahap pertama dari proses penelitian ini adalah dilakukanya pengambilan data video yang berfungsi sebagai data masukan. Di karenakan tempat pengambilan rekaman video tidak memiliki CCTV, sehingga penulis menggunakan kamera HP sebagai alat perekam. Data masukan pada penelitian ini berupa video *offline*. Proses perekaman video dilakukan di Lapangan Parkir Fakultas Ilmu dan Agama Islam Universitas Islam Indonesia. Format ekstensi rekaman video *offline* yang digunakan adalah .*mp4. Ukuran *output* untuk pengolahan video pada penelitian ini adalah 360x640 piksel. Untuk penelitian ini, jumlah data video yang digunakan berjumlah 13 data video. Untuk mengetahui jarak perpindahan manusia dari masing – masing data, yaitu dengan variasi jarak yang berbeda – beda, serta dapat dibandingkan antara jarak perpindahan aktual dengan jarak yang terukur pada data video. Variasi jarak dan arah dari data yang diambil adalah 3 meter dengan arah ke kanan, 4 meter dengan arah kanan, 4 meter dengan arah kiri, 5 meter dengan arah ke kanan, 5 meter dengan arah ke kiri, 6 meter dengan arah kanan, 6 meter dengan arah ke kiri, 7 meter dengan arah ke kanan, 7 meter dengan arah ke kiri, 8 meter dengan arah ke kanan, 8 meter dengan arah ke kiri, 10 meter dengan arah ke kanan, dan yang terakhir adalah 10 meter dengan arah ke kiri. Untuk jumlah data dan terkait variasi video yang dihasilkan dapat dilihat padaTabel 3. 1. Hasil visualisasi proses perekaman video seperti pada Gambar 3. 2 dan *layout* proses perekaman video pada seperti pada Gambar 3. 3.

Tabel 3. 1 Data video dan variasinya

Nomor	Data	Arah	Jarak Sebenarnya	Durasi Video
1	Video 1	Kanan	3 meter	2 detik
2	Video 2	Kanan	4 meter	3 detik
3	Video 3	Kiri	4 meter	3 detik
4	Video 4	Kanan	5 meter	4 detik
5	Video 5	Kiri	5 meter	4 detik
6	Video 6	Kanan	6 meter	5 detik
7	Video 7	Kiri	6 meter	5 detik
8	Video 8	Kanan	7 meter	5 detik
9	Video 9	Kiri	7 meter	5 detik
10	Video 10	Kanan	8 meter	6 detik
11	Video 11	Kiri	8 meter	6 detik
12	Video 12	Kanan	10 meter	8 detik
13	Video 13	Kiri	10 meter	8 detik



Gambar 3. 2 Hasil visualisasi proses perekaman video pada jarak 10 meter

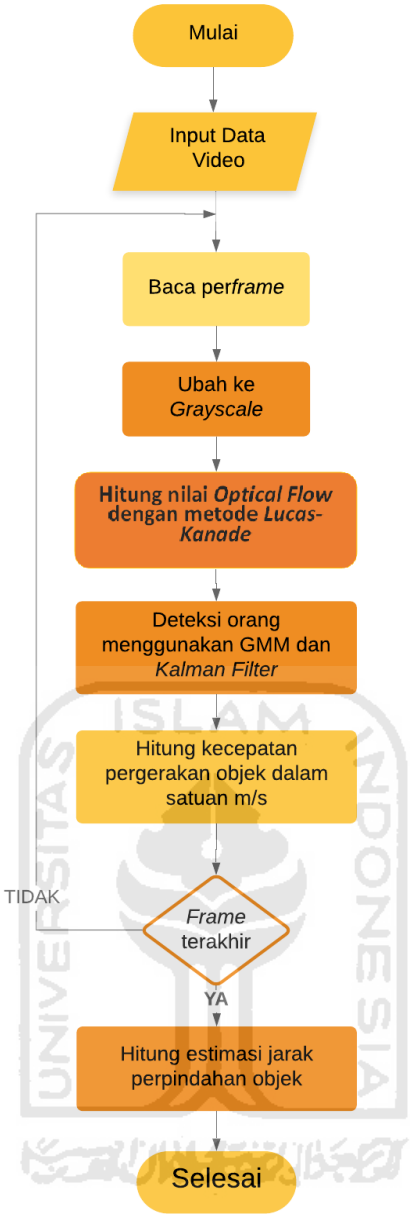


Gambar 3. 3 *Layout* proses perekaman video

Gambar 3. 3 menunjukkan proses untuk dilakukanya perekaman video yang berlokasi di Lapangan Parkir Fakultas Ilmu dan Agama Islam Universitas Islam Indonesia. Yang mana, *perpendicular view* merupakan bidang tegak lurus pandang dari kamera [12]. Nilai *perpendicular view* sangat diperlukan untuk dilakukanya perhitungan skala dari piksel ke meter. Pengukuran nilai *perpendicular view* dilakukan secara manual yaitu dengan menggunakan meteran kain.

3.1.2 Sistem Untuk Proses Pendeteksian Jarak Perpindahan Manusia.

Pada tahap ini akan dilakukan pendeteksian jarak perpindahan manusia menggunakan video yang sudah diambil pada tahap sebelumnya. Sistem yang digunakan untuk dilakukanya pendeteksian jarak dengan menggunakan algortima *Lucas – Kanade Optical Flow*. Dalam penelitian ini menggunakan metode *Lucas – Kanade Optical Flow* karena memiliki sensitifitas yang lebih rendah dari pada metode *Horn – Schunck* dan juga perhitungan yang lebih sederhana yang membuatnya lebih cepat mendeteksi vektor pergerakan [16]. Untuk proses pengolahan algoritma *Lucas – Kanade Optical Flow* menggunakan *software* Matlab *(Matrix Laboratory)* 2017. Untuk proses mencapai tujuan deteksi jarak perpindahan manusia pada penelitian ini dapat dilihat pada Gambar 3. 4 berikut ini :



Gambar 3. 4 Diagram alir deteksi jarak perpindahan manusia

- Input data video

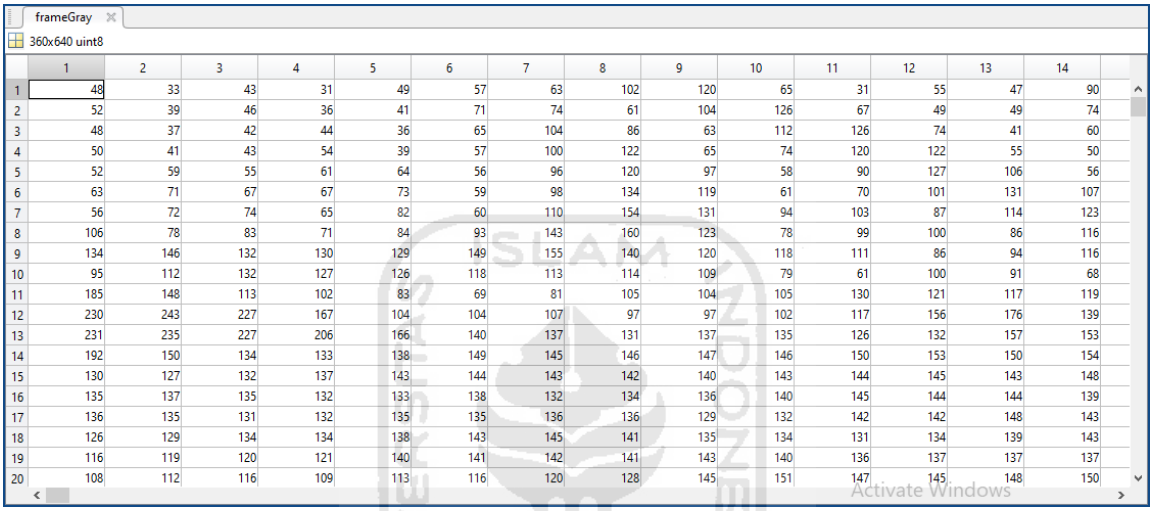
  Data masukan yang digunakan pada penelitian deteksi jarak perpindahan manusia ini berupa rekaman video *offline* pergerakan manusia dengan berbagai macam variasi jarak dan arah pergerakan manusia. Sudut pandang yang diambil ketika merekam video adalah dari atas dengan nilai *perpendicular view*-nya adalah 28 meter. Format ekstensi dari rekaman video *offline* berupa : .*mp4.

- Baca perframe

  Pada tahapan ini tiap *frame*-nya akan dibaca ketika data video *offline* yang sudah direkam di input ke dalam program matlab. Karena secara teori video merupakan sekumpulan *frame* yang bergerak.

- Ubah ke *grayscale*

Rekaman video yang merupakan data sebagai masukan yang pada awalnya berbentuk citra RGB dikonversikan menjadi citra *grayscale*. Hal ini dikarenakan *grayscale* bekerja berdasarkan intensitas cahaya, sehingga hal ini sangat sesuai dengan prinsip kerja *Optical Flow* yang memanfaatkan aliran intensitas cahaya untuk menentukan vektor pergerakan. Nilai *grayscale* pada dasarnya memiliki rentan antara 0 – 255 pada derajat keabuan. Nilai *grayscale* yang dihasilkan dari konversi di representasikan dalam bentuk angka. Rentan tersebut berasal dari *grayscale* 8 bit yang dapat dinyatakan seperti Gambar 3. 5.



	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	48	33	43	31	49	57	63	102	120	65	31	55	47	90
2	52	39	46	36	41	71	74	61	104	126	67	49	49	74
3	48	37	42	44	36	65	104	86	63	112	126	74	41	60
4	50	41	43	54	39	57	100	122	65	74	120	122	55	50
5	52	59	55	61	64	56	96	120	97	58	90	127	106	56
6	63	71	67	67	73	59	98	134	119	61	70	101	131	107
7	56	72	74	65	82	60	110	154	131	94	103	87	114	123
8	106	78	83	71	84	93	143	160	123	78	99	100	86	116
9	134	146	132	130	129	149	155	140	120	118	111	86	94	116
10	95	112	132	127	126	118	113	114	109	79	61	100	91	68
11	185	148	113	102	83	69	81	105	104	105	130	121	117	119
12	230	243	227	167	104	104	107	97	97	102	117	156	176	139
13	231	235	227	206	166	140	137	131	137	135	126	132	157	153
14	192	150	134	133	138	149	145	146	147	146	150	153	150	154
15	130	127	132	137	143	144	143	142	140	143	144	145	143	148
16	135	137	135	132	133	138	132	134	136	140	145	144	144	139
17	136	135	131	132	135	135	136	136	129	132	142	142	148	143
18	126	129	134	134	138	143	145	141	135	134	131	134	139	143
19	116	119	120	121	140	141	142	141	143	140	136	137	137	137
20	108	112	116	109	113	116	120	128	145	151	147	145	148	150

Gambar 3. 5 Contoh hasil konversi citra RGB ke *grayscale* pada *frame*

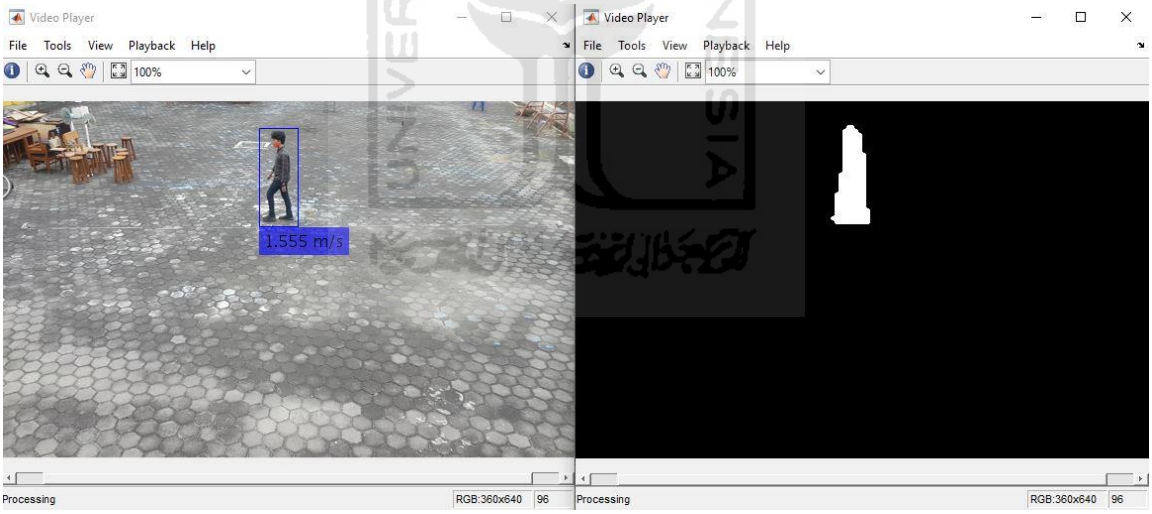
- Hitung nilai *Optical Flow* dengan menggunakan metode *Lucas – Kanade*

Setelah citra RGB melalui tahap konversi menjadi citra *grayscale* maka nilai *Optical Flow* akan diperoleh. Perhitungan nilai *Optical Flow* yang dihasilkan adalah nilai pergerakan aliran intensitas cahaya dari satu *frame* penuh tidak hanya pada objek. Algoritma *Lucas – Kanade Optical Flow* secara otomatis akan mengasumsikan semua gambar atau video dengan *Optical Flow*. Hasil perhitungan menggunakan *Optical Flow* menghasilkan nilai kecepatan dari pergerakan vektor . Satuan nilai pergerakan *Optical Flow* yang dihasilkan adalah piksel per detik.

- Deteksi orang dengan GMM (*Gaussian Mixture model*) dan *Kalman Filter*

Proses pendeteksian orang untuk ditandai dengan *bounding box* di setiap pergerakanya menggunakan algoritma GMM dan *Kalman Filter* memiliki fungsinya masing – masing. GMM berfungsi sebagai pendeteksian *foreground* (objek). Fungsi GMM dan *Kalman Filter* pada penelitian ini adalah untuk memudahkan dalam proses

memvisualisai pergerakan manusia agar dan juga memperecepat dalam menganalisis pergerakan manusia. Deteksi orang dengan *Gaussian Mixture model* atau GMM yang berfungsi untuk mengsegmentasi pergerakan objek atau mendeteksi *foreground* (latar depan/objek) dengan cara memisahkan nilai pixel yang tidak sesuai dengan distribusi *Gaussian* pada *background*. Untuk memperhalus hasil dari deteksi GMM, maka dibutuhkanlah operasi morfologi sebagai proses penghalusanya. Pada operasi morfologi terdapat berbagai macam proses didalamnya, yaitu operasi opening, closing dan filling. Operasi opening berfungsi untuk menghilangkan objek – objek kecil disekitar objek utama yang bertujuan untuk menghaluskan batas dari area yang besar tanpa mengubah area objeknya secara signifikan. Operasi *closing* bertujuan untuk mengisi lubang kecil yang ada pada objek dengan cara menggabungkan objek yang berdekatan. Kemudian operasi *filling* yang bertujuan untuk menghaluskan kedua proses dengan cara melakukan pengisian di area lubang untuk menghasilkan segmentasi yang lebih baik [3]. *Kalman filter* berfungsi untuk melakukan *tracking* suatu objek yang telah didefinisikan sebelumnya oleh GMM. Pada saat objek bergerak atau *frame* berjalan maka *kalman filter* akan meprediksi pergerakan objek setiap *frame*-nya. Hasil deteksi pergerakan manusia menggunakan GMM dan *kalman filter* dapat dilihat pada Gambar 3. 6.

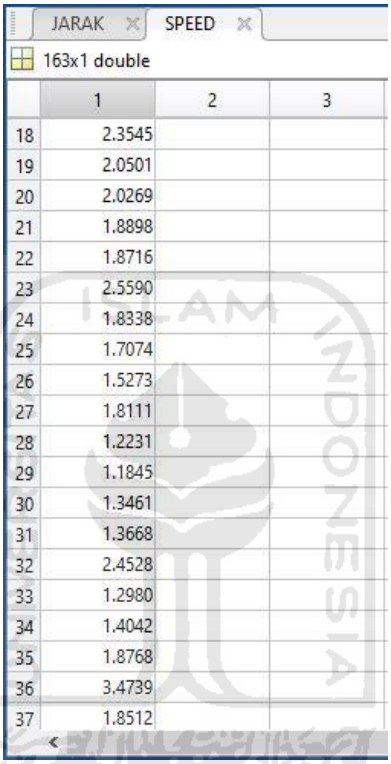


Gambar 3. 6 Hasil deteksi pergerakan manusia menggunakan GMM dan *kalman filter*

- Hitung kecepatan objek dalam satuan meter per detik

Setelah *Optical Flow* menghitung kecepatan objek yang sudah ditentukan oleh GMM, untuk mencapai proses pendeteksian jarak perpindahan objek maka dibutuhkanya parameter kecepatan objek dalam satuan meter per detik dan juga waktu dalam detik. Sehingga dalam penelitian ini dilakukanya proses pendeteksian kecepatan pergerakan dari objek di setiap *frame.* Untuk mendapatkan skala dalam meter per detik maka dibutuhkan informasi atas nilai *perpendicular view* dari kamera dalam satuan meter. Nilai

perpendicular view yang diukur secara manual adalah 28 meter. Sehingga untuk mendapatkan skala dalam satuan meter per detik adalah dengan melakukan pembagian antara nilai *perpendicular view* dari kamera dalam satuan meter dengan ketinggian citra dalam satuan piksel (meter/piksel). Setelah didapatkanya skala dalam (meter/piksel), maka nilai kecepatan objek dari tiap *frame* dalam satuan meter per detik dengan mengalikan nilai *Optical Flow* dari pergerakan objek dari metode perhitungan *Lucas – Kanade Optical Flow* dengan nilai skala yang didapat. Hasil pendeteksian kecepatan rata – rata objek di tiap *frame* nya dapat dilihat padaGambar 3. 7 Gambar 3. 7:


JARAK SPEED

163x1 double

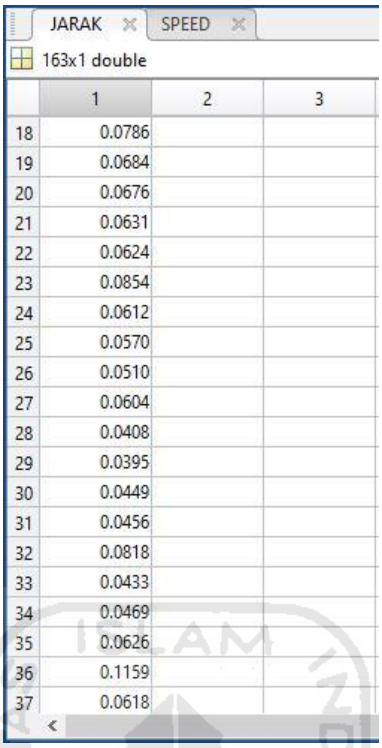
	1	2	3
18	2.3545		
19	2.0501		
20	2.0269		
21	1.8898		
22	1.8716		
23	2.5590		
24	1.8338		
25	1.7074		
26	1.5273		
27	1.8111		
28	1.2231		
29	1.1845		
30	1.3461		
31	1.3668		
32	2.4528		
33	1.2980		
34	1.4042		
35	1.8768		
36	3.4739		
37	1.8512		

Gambar 3. 7 Nilai kecepatan objek masing – masing *frame* dalam satuan meter per detik

- Hitung estimasi jarak perpindahan manusia

Pada tahap proses perhitungan jarak perpindahan manusia ini merupakan tahapan akhir dari semua proses yang bertujuan untuk mendapatkan informasi berupa jarak perpindahan manusia dalam satuan meter. Setelah didapatkanya nilai kecepatan pergerakan objek dari masing – masing *frame* dan nilai waktu yang dibutuhkan untuk perpindahan antar *frame* satu ke *frame* selanjutnya dalam satuan detik maka untuk memperoleh nilai jarak perpindahan dari objek yaitu dengan cara melakukan pengalian antara kecepatan dalam satuan meter per detik dengan waktu dalam satuan detik. Setelah itu nilai jarak perpindahan objek dalam satuan meter dapat diperoleh dari *frame* awal hingga *frame* akhir. Setelah mendapatkan nilai jarak perpindahan objek disetiap *frame*-nya, selanjutnya dilakukan proses penjumlahan jarak yang diperoleh dari *frame* awal

hingga *frame* terakhir. Sehingga estimasi jarak perpindahan objek mulai bergerak hingga objek berhenti dapat diperoleh. Sehingga dapat kita lihat pada Gambar 3. 8.



JARAK | SPEED

163x1 double

	1	2	3
18	0.0786		
19	0.0684		
20	0.0676		
21	0.0631		
22	0.0624		
23	0.0854		
24	0.0612		
25	0.0570		
26	0.0510		
27	0.0604		
28	0.0408		
29	0.0395		
30	0.0449		
31	0.0456		
32	0.0818		
33	0.0433		
34	0.0469		
35	0.0626		
36	0.1159		
37	0.0618		

Gambar 3. 8 Nilai jarak yang diperoleh dari masing – masing pergerakan objek di setiap *frame* dalam satuan meter

- Frame terakhir

  Pada penelitian ini, *frame* terakhir akan terdeteksi apabila video rekaman sudah berakhir. Pada saat penjalanan video dan proses perhitungan jarak perpindahan dari pergerakan manusia mamasuki tahap pada *frame* terakhir, maka sistem yang berjalan pada Matlab akan berhenti dan berakhir setelahnya akan menampilkan hasil perhitungan estimasi jarak pada *frame* terakhir.

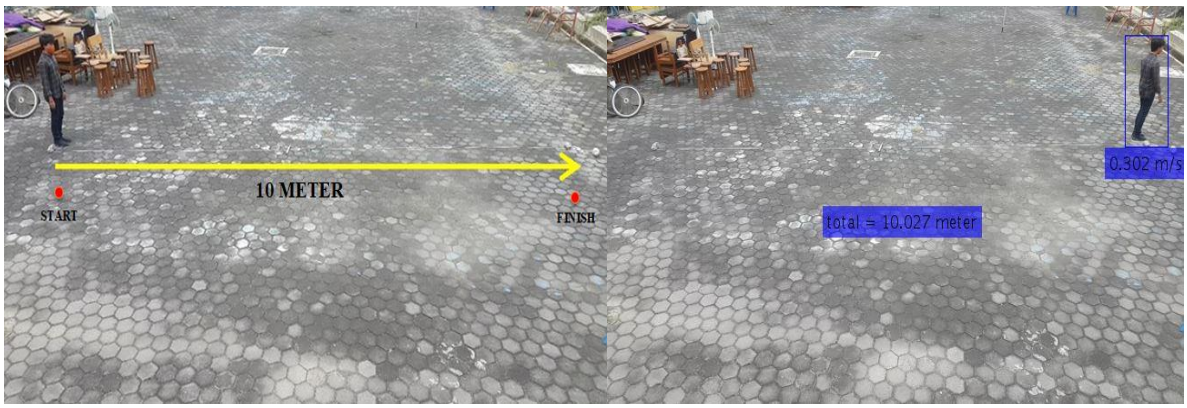
3.1.3 Hasil dan Analisis

Untuk validasi terhadap estimasi pengukuran estimasi jarak yaitu dengan membandingkan hasil pengukuran menggunakan meteran dan sistem. Perpindahan objek menghasilkan pengukuran yang cukup konsisten karena memiliki perhitungan eror yang cukup rendah. Faktor yang sangat mempengaruhi dari proses pengukuran jarak perpindahan objek ada pada saat proses deteksinya. Intensitas cahaya yang baik dan kondisi lingkungan sekitar sangat mempengaruhi dalam perhitungan estimasi jarak perpindahan objek.

BAB 4
HASIL DAN PEMBAHASAN

4.1 Hasil deteksi jarak perpindahan objek menggunakan *Optical Flow* dan pengukuran jarak secara manual

Hasil pengukuran jarak perpindahan objek menggunakan *Optical Flow* dan pengukuran jarak secara manual akan ditampilkan pada bagian ini. Proses pengambilan data video rekaman dengan berbagai macam variasi jarak dan arah perpindahan objek pada saat perekaman video. Jumlah data video rekaman yang digunakan pada penelitian ini berjumlah 13 buah video rekaman. Variasi jarak dan arah dari data yang diambil adalah 3 meter dengan arah ke kanan, 4 meter dengan arah kanan, 4 meter dengan arah kiri, 5 meter dengan arah ke kanan, 5 meter dengan arah ke kiri, 6 meter dengan arah kanan, 6 meter dengan arah ke kiri, 7 meter dengan arah ke kanan, 7 meter dengan arah ke kiri, 8 meter dengan arah ke kanan, 8 meter dengan arah ke kiri, 10 meter dengan arah ke kanan, dan yang terakhir adalah 10 meter dengan arah ke kiri. Dari masing – masing data video rekaman akan ditampilkan gambar hasil visualisasi perhitungan jarak menggunakan *Optical Flow*. Kemudian akan dilakukanya perbandingan terhadap hasil proses pendeteksian jarak perpindahan objek menggunakan *Optical Flow* dengan perhitungan jarak secara manual yaitu dengan menggunakan meteran. Hasil perbandingan antara hasil deteksi jarak perpindahan objek menggunakan *Optical Flow* dengan pengukuran jarak perpindahan objek secara manual yaitu dengan cara memberikan titik – titik yang sudah diukur menggunakan meteran yang bertujuan sebagai penanda posisi tempat awal objek mulai bergerak hingga objek berhenti. Hal ini dapat dilihat pada Gambar 4. 1.



Gambar 4. 1 Visualisasi hasil perhitungan jarak aktual dan menggunakan sistem

4.2 Pembahasan hasil deteksi jarak perpindahan manusia

Pengujian pada sistem ini dilakukan dengan dilakukanya *input* data video rekaman objek yang bergerak dengan berbagai macam variasi jarak dan arah pergerakan. Pengambilan data video dilakukan di Lapangan Parkir Fakultas Ilmu dan Agama Islam Universitas Islam Indonesia pada waktu sore hari. Durasi pada rekaman juga bermacam – macam di setiap video rekamanya. Untuk melakukan pengujian pada sistem ini, perlu dimasukkanya beberapa parameter, yaitu:

1. *Perpendicular view* (bidang tegak lurus dari kamera) = 28 meter
2. Tinggi citra = 360 piksel

Parameter – parameter ini diperlukan untuk menghitung skala dari piksel ke meter. Proses perhitungan ini didapatkan dari proses perhitungan *Optical Flow* dari objek yang bergerak kemudian dikalikan dengan parameter yang sudah dimasukkan pada proses sebelumnya. Sehingga didapatkanya hasil perhitungan jarak pergerakan objek mulai dari awal bergerak hingga objek berhenti dalam satuan meter. Eror yang dihasilkan dari data pertama hingga data terakhir tidak memiliki eror yang terlalu berbeda jauh. Hal ini disebabkan oleh proses pada saat pengambilan data atau perekaman video menggunakan kamera yang sama, posisi saat perekaman video yang sama dan waktu pada saat perekaman video yang berdekatan yaitu di sore hari. Berikut nilai eror yang dihasilkan oleh perhitungan menggunakan sistem yang dapat dilihat pada Tabel 4. 1.

Tabel 4. 1 Perhitungan eror pada sistem

No	Data rekaman	Jarak sebenarnya	Hasil pengukuran deteksi Optical Flow	Persentase eror $\frac{(JS-JO)}{JS} * 100\%$
1	Rekaman 1	3 meter	3,067 meter	2,23%
2	Rekaman 2	4 meter	4,021 meter	0,50%
3	Rekaman 3	4 meter	4,011 meter	0,27%
4	Rekaman 4	5 meter	4,981 meter	0,38%
5	Rekaman 5	5 meter	5,008 meter	0,16%
6	Rekaman 6	6 meter	6,026 meter	0,40%
7	Rekaman 7	6 meter	6,051 meter	0,80%
8	Rekaman 8	7 meter	7,035 meter	0,50%
9	Rekaman 9	7 meter	7,028 meter	0,40%
10	Rekaman 10	8 meter	8,001 meter	0,01%
11	Rekaman 11	8 meter	8,005 meter	0,06%
12	Rekaman 12	10 meter	10,027 meter	0,20%
13	Rekaman 13	10 meter	9,961 meter	0,40%
Rata – rata eror				0,48%

Pada Tabel 4. 1 hasil deteksi jarak perpindahan objek menggunakan deteksi *Optical Flow* pada seluruh data video rekaman berhasil terdeteksi dengan cukup baik. Hal ini dapat dilihat dari data eror yang dihasilkan dari pengukuran menggunakan sistem deteksi *Optical Flow*. Hasil pengukuran menggunakan sistem deteksi *Optical Flow* menghasilkan perhitungan eror paling kecil yang dihasilkan pada saat pengukuran jarak perpindahan objek dengan jarak 8 meter yang menghasilkan eror sebesar 0,01% dengan rata-rata jarak perpindahan objek di tiap *frame* nya sebesar 5 cm dengan durasi video selama 2 detik. Hal ini disebabkan oleh pada saat dilakukanya pendeteksian objek, *bounding box* dapat mendeteksi bagian objek yang bergerak secara berurutan dengan baik mulai dari awal objek bergerak hingga berhenti. Sedangkan eror yang paling besar adalah pada saat pengukuran jarak 3 meter yaitu sebesar 2,23% dengan rata-rata jarak perpindahan objek di tiap *frame* nya sebesar 4,9 cm dengan durasi video selama 6 detik. Hal ini disebabkan karena kegagalan dalam pendeteksian objek dengan *boundingbox,* di beberapa *frame, boundingbox* hanya dapat mendeteksi sedikit bagian dari objek sehingga menyebabkan eror pada saat melakukan estimasi jarak perpindahan objek. Berdasarkan penelitian yang sudah dilakukan, hal ini menunjukkan bahwa hasil perhitungan menggunakan deteksi *Optical Flow* memiliki akurasi yang cukup tinggi dalam hal pendeteksian jarak perpindahan objek.

Berdasarkan tingkat rata – rata eror yang dihasilkan perhitungan dengan menggunakan sistem deteksi *Optical Flow* adalah 0,48% sehingga menghasilkan nilai akurasi sebesar 99,52%. Algoritma untuk pendeteksian jarak pergerakan objek menggunakan *Optical Flow Lucas-Kanade* ini memiliki algortitma yang sangat sederhana yaitu secara langsung mendeteksi pergerakan vektor dari objek yang bergerak. Nilai eror paling rendah dihasilkan adalah 0,01% dan yang paling tinggi adalah 2,23%. Hal yang menyebabkan terjadinya eror pada sistem adalah karena proses perekaman video dilakukan pada sore hari dan di ruangan terbuka dan berawan. *Optical Flow* sangat sensitif terhadap perubahan cahaya, hal ini sangat mempengaruhi perhitungan pergerakan vektor. GMM dan *Kalman Filter* berfungsi sebagai proses visualisasi dan pembentukan *bounding box* pada objek sebagai pembatas area untuk dilakukanya perhitungan terhadap pergerakan objek.

4.3 Perbandingan dengan penelitian sebelumnya

Tabel 4. 2 Ringkasan perbandingan akurasi dengan penelitian sebelumnya

No	Peneliti	Metode	Akurasi
1	Budi Setiyono, Dwi Ratna Sulistyaningrum, Soetrisno, Danang Wahyu Wicaksono (2019)	*Gaussian Mixture Model*	99,38%
2	Indah Lugianti, Jayanti Yasmah Sari, Ika Purwanti Ningrum (2012)	Metode *Frame Difference*	76,67%
3	Alan Firdaus dan Imelda (2018)	*Gaussian Blur* dan *Absolute Difference*	94,89%
4	Penelitian pada skripsi ini (2020)	*Optical Flow*	99,52%

Berdasarkan perbandingan dengan penelitian sebelumnya seperti pada Tabel 4. 2, akurasi yang dihasilkan perhitungan menggunakan *Optical Flow* jauh lebih baik yaitu sebesar 99,52%. Hal ini disebabkan oleh keberhasilan *Optical Flow* dalam mendeteksi pergerakan objek dengan sangat baik. Sehingga perhitungan estimasi jarak memiliki akurasi yang baik.

BAB 5
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan penelitian yang telah dilakukan, maka didapatkan kesimpulan sebagai bahwa sistem deteksi jarak perpindahan manusia menggunakan sistem deteksi *Optical Flow Lucas-Kanade* pada penelitian ini dapat melakukan pendeteksian jarak perpindahan objek pada semua data video dengan sangat baik. Hal ini dapat dilihat dari hasil pengujian yang sudah dilakukan. Tingkat akurasi yang dihasilkan adalah 99,52%.

5.2 Saran

Melihat hasil yang sudah dicapai pada penelitian ini, terdapat beberapa hal yang penulis sarankan untuk pengembangan sistem ini ke depan yaitu:

1. Menggunakan video secara *real time* atau *online* agar deteksi jarak perpindahan manusia bisa diaplikasikan secara langsung untuk keperluan sistem pengawasan rumah untuk lansia, dan lain – lain.
2. Program belum dapat melakukan pendeteksian jarak perpindahan objek lebih dari 1 objek. Penelitian ke depan diharapkan dapat mengembangkan sistem untuk bisa mendeteksi jarak perpindahan objek untuk lebih dari 1 objek pada waktu yang bersamaan.

DAFTAR PUSTAKA

[1] Bada Pusat Statistik (BPS), "Katalog: 4104001," *Stat. Pendud. LANJUT USIA 2019*, vol. 2019, pp. 1–286, 2019.

[2] W. S. Pambudi *et al.*, "Deteksi dan Estimasi Jarak Obyek Menggunakan Single Camera Dengan Model Segmentasi HSV," *Semin. Nas. Teknoim*, pp. 134–137, 2011.

[3] K. Umam and B. S. Negara, "Deteksi Obyek Manusia Pada Basis Data Video Menggunakan Metode Background Subtraction Dan Operasi Morfologi," *Deteksi Obyek Mns. Pada Basis Data Video Menggunakan Metod. Backgr. Subtraction Dan Operasi Morfol. Khairul*, vol. 2, no. 2, pp. 31–40, 2016.

[4] R. F. Waliulu, "Deteksi dan Penggolongan Kendaraan dengan Kalman Filter dan M odel Gaussian di Jalan Tol," *J. Sist. Inf. Bisnis*, vol. 01, pp. 1–8, 2018.

[5] A. Firdaus and Imelda, "Penerapan Metode Gaussian Blur Dan Absolute Difference Pada Jumlah Dan Kecepatan Kendaraan," *SKANIKA*, vol. 1, no. 3, pp. 1003–1011, 2018.

[6] B. Setiyono and D. R. Sulistyaningrum, "Multi Vehicle Speed Detection Using Euclidean Distance Based On Multi Vehicle Speed Detection Using Euclidean Distance," *Int. J. Comput.*, no. January, pp. 431–442, 2019.

[7] W. Supriyatin and W. W. Ariestya, "Analisis pelacakan objek mobil dengan optical flow pada kamera diam dan bergerak," *Semin. Ris. Teknol. Inf. tahun 2016 Anal.*, no. July 2016, pp. 48–56, 2016.

[8] I. Lugianti, J. Yusmah Sari, and I. Purwanti Ningrum, "Deteksi Kecepatan Kendaraan Bergerak Berbasis Video Menggunakan Metode Frame Difference," *Semin. Nas.*, vol. 1, pp. 324–332, 2012.

[9] A. Tourani, A. Shahbahrami, A. Akoushideh, S. Khazaee, and C. Y Suen, "Motion-based Vehicle Speed Measurement for Intelligent Transportation Systems," *Int. J. Image, Graph. Signal Process.*, vol. 11, no. 4, pp. 42–54, 2019.

[10] L. Rahmawati, D. Fisika, F. Sains, and U. Diponegoro, "Rancang bangun penghitung dan pengidentifikasi kendaraan menggunakan Multiple Object Tracking," *Youngster Phys. J.*, vol. 6, no. 1, pp. 70–75, 2017.

[11] T. Urip, K. Adi, E. Widodo, D. Fisika, F. Sains, and U. Diponegoro, "Pengukuran jarak objek pejalan kaki terhadap kamera menggunakan kamera stereo terkalibrasi dengan segmentasi objek histogram of oriented gradient," *Youngster Phys. J.*, vol. 6, no. 3, pp. 249–262, 2017.

[12] D. W. Wicaksono, "Pengembangan sistem estimasi kecepatan pada kendaraan bergerak berbasis pengolahan citra digital," *TESIS – SM 142501*, 2017.

[13] A. ROMLI, "Pengukuran Kecepatan Kendaraan Menggunakan Optical Flow," *TUGAS AKHIR – SM141501 PENGUKURAN*, pp. 1–85, 2017.

[14] Y. Mi, P. K. Bipin, and R. K. Shah, "Using Lucas-Kanade Algorithms to Measure Human Movement," *Commun. Comput. Inf. Sci.*, pp. 118–130, 2018.

[15] D. A. Wahyudi and I. H. Kartowisastro, "Menghitung Kecepatan Menggunakan Computer Vision," *J. Tenik Komput.*, vol. 19, no. 2, pp. 89–101, 2011.

[16] W. Supriyatin, U. Gunadarma, O. Flow, and P. Objek, “Analisis Perbandingan Pelacakan Objek Menggunakan Algoritma Horn-Schunck Dan Lucas-Kanade,” *KOMPUTASI J. Ilm. Ilmu Komput. dan Mat.*, vol. 17, no. 2, pp. 362–371, 2020.

LAMPIRAN

Lampiran 1 – Rincian Biaya Skripsi

No	Rincian	Jumlah	Jumlah (Rp)
1	Ram 8 GB	1	Rp450,000,00
2	SSD 256 MB Full Set	1	Rp650,000,00
Jumlah			**Rp1,100,000.00**

Lampiran 2 – *Source Code* Penelitian

```
% function OF2020()
clc; clear; close all
%read video
videolink = [cd,'\10a_1.mp4'];
% obj = setupSystemObjects(videolink);

% Create a video file reader.
obj.reader = VideoReader(videolink);

% Buat dua pemutar video, satu untuk menampilkan video,
% dan satu untuk menampilkan topeng latar depan.
obj.maskPlayer = vision.VideoPlayer('Position', [600 350 580 350]);
obj.videoPlayer = vision.VideoPlayer('Position', [10 350 580 350]);

% Buat objek Sistem untuk deteksi latar depan dan analisis gumpalan



% Detektor latar depan digunakan untuk mensegmentasikan objek bergerak
% latar belakang. Ini menghasilkan topeng biner, di mana nilai piksel
% dari 1 berkorespondensi dengan latar depan dan nilai 0 berkorespondensi
% ke latar belakang.
obj.detector = vision.ForegroundDetector('NumGaussians', 3, ... % deteksi GMM
    'NumTrainingFrames', 40, 'MinimumBackgroundRatio', 0.8);

% Grup piksel latar depan yang terhubung cenderung sesuai dengan pemindahan
% objects.  The blob analysis System object is used to find such groups
% (called 'blobs' or 'connected components'), and compute their
% characteristics, such as area, centroid, and the bounding box.

obj.blobAnalyser = vision.BlobAnalysis('BoundingBoxOutputPort', true, ...   %
hasil deteksi dengan GMM diperbaiki dengan bloob Analyser
    'AreaOutputPort', true, 'CentroidOutputPort', true, ...
    'MinimumBlobArea', 40);

% end

% create optical flow
OpticalFlow = opticalFlowLK('NoiseThreshold',0.00001);
% shapeInserter = vision.ShapeInserter('Shape', 'lines', 'BorderColor',
'Custom', 'CustomBorderColor', 255);
nFrames = 1;

color = {'blue', 'green', 'cyan', 'red', 'magenta', 'black', 'white',};

% tracks = initializeTracks(); % Create an empty array of tracks.
```

```
% function tracks = initializeTracks()
% create an empty array of tracks
tracks = struct(...
    'id', {}, ...
    'bbox', {}, ...
    'kalmanFilter', {}, ...
    'age', {}, ...
    'totalVisibleCount', {}, ...
    'consecutiveInvisibleCount', {});
% end

nextId = 1; % ID of the next track

numFrames = 0;
v = VideoReader(videolink);
while hasFrame(v)
    readFrame(v);
    numFrames = numFrames+1;
end

while hasFrame(obj.reader)

    % read frame
    frameRGB = readFrame(obj.reader); % read the next video frame
    frameRGB = imresize(frameRGB,360/size(frameRGB,1));
    frameGray = rgb2gray(frameRGB);
    im = im2double(frameGray);

    % calculate velocity
    of = estimateFlow(OpticalFlow, im);
    Vx = of.Vx;
    Vy = of.Vy;
    velocity = sqrt((Vx.^2)+((Vy.^2)));

    % detect people
    %     [centroids, bboxes, mask] =
detectObjects(obj,imresize(frameRGB,0.25));
    %     function [centroids, bboxes, mask] = detectObjects(obj,frame)

    % Detect foreground.
    mask = obj.detector.step(imresize(frameRGB,0.25));

    % Apply morphological operations to remove noise and fill in holes.
    mask = medfilt2(mask,[2,2]);
    mask = imopen(mask, strel('rectangle', [1,1]));
    mask = imclose(mask, strel('rectangle', [15, 15]));
    mask = imfill(mask, 'holes');

    % Perform blob analysis to find connected components.
    [~, centroids, bboxes] = obj.blobAnalyser.step(mask);
    %     end
    bboxes = 4*(bboxes);
    mask = imresize(mask,4);

    %     tracks = predictNewLocationsOfTracks(tracks);
    %     function tracks = predictNewLocationsOfTracks(tracks)
    for i = 1:length(tracks)
        bbox = tracks(i).bbox;

        % Memprediksi lokasi trek saat ini.
```

```
            predictedCentroid = predict(tracks(i).kalmanFilter);

            % Geser bounding box sehingga pusatnya berada di
            % lokasi yang diprediksi.
            predictedCentroid = int32(predictedCentroid) - bbox(3:4) / 2;
            tracks(i).bbox = [predictedCentroid, bbox(3:4)];
        end
        % end

        %     [assignments, unassignedTracks, unassignedDetections] = ...
        %         detectionToTrackAssignment(tracks,centroids);

        %     function [assignments, unassignedTracks, unassignedDetections] =
...
        %     detectionToTrackAssignment(tracks,centroids)

        nTracks = length(tracks);
        nDetections = size(centroids, 1);

        % Compute the cost of assigning each detection to each track.
        cost = zeros(nTracks, nDetections);
        for i = 1:nTracks
            cost(i, :) = distance(tracks(i).kalmanFilter, centroids);
        end

        % Memecahkan masalah penugasan.
        costOfNonAssignment = 20;
        [assignments, unassignedTracks, unassignedDetections] = ...
            assignDetectionsToTracks(cost, costOfNonAssignment);
        %     end


        %     tracks = updateAssignedTracks(tracks,centroids,bboxes,assignments);
        %     function tracks =
updateAssignedTracks(tracks,centroids,bboxes,assignments)
        numAssignedTracks = size(assignments, 1);
        for i = 1:numAssignedTracks
            trackIdx = assignments(i, 1);
            detectionIdx = assignments(i, 2);
            centroid = centroids(detectionIdx, :);
            bbox = bboxes(detectionIdx, :);

            % Memperbaiki perkiraan lokasi objek
            % menggunakan deteksi baru.
            correct(tracks(trackIdx).kalmanFilter, centroid);

            % Ganti diprediksi bounding box with detected
            % bounding box.
            tracks(trackIdx).bbox = bbox;

            % Update track's age.
            tracks(trackIdx).age = tracks(trackIdx).age + 1;

            % Update visibility.
            tracks(trackIdx).totalVisibleCount = ...
                tracks(trackIdx).totalVisibleCount + 1;
            tracks(trackIdx).consecutiveInvisibleCount = 0;
        end
        % end

        %     tracks = updateUnassignedTracks(tracks,unassignedTracks);
```

```
    %     function tracks = updateUnassignedTracks(tracks,unassignedTracks)
    for i = 1:length(unassignedTracks)
        ind = unassignedTracks(i);
        tracks(ind).age = tracks(ind).age + 1;
        tracks(ind).consecutiveInvisibleCount = ...
            tracks(ind).consecutiveInvisibleCount + 1;
    end
    % end

    %     tracks = deleteLostTracks(tracks);
    %     function tracks = deleteLostTracks(tracks)
    %     if isempty(tracks)
    %         return;
    %     end

    invisibleForTooLong = 20;
    ageThreshold = 8;

    % Compute the fraction of the track's age for which it was visible.
    ages = [tracks(:).age];
    totalVisibleCounts = [tracks(:).totalVisibleCount];
    visibility = totalVisibleCounts ./ ages;

    % Find the indices of 'lost' tracks.
    lostInds = (ages < ageThreshold & visibility < 0.6) | ...
        [tracks(:).consecutiveInvisibleCount] >= invisibleForTooLong;

    % Delete lost tracks.
    tracks = tracks(~lostInds);
    % end

    %     [tracks,nextId] =
createNewTracks(tracks,centroids,bboxes,unassignedDetections,nextId);
    %     function [tracks,nextId] =
createNewTracks(tracks,centroids,bboxes,unassignedDetections,nextId)
    centroids = centroids(unassignedDetections, :);
    bboxes = bboxes(unassignedDetections, :);

    for i = 1:size(centroids, 1)

        centroid = centroids(i,:);
        bbox = bboxes(i, :);

        % Create a Kalman filter object.
        kalmanFilter = configureKalmanFilter('ConstantVelocity', ...
            centroid, [200, 50], [100, 25], 100);

        % Create a new track.
        newTrack = struct(...
            'id', nextId, ...
            'bbox', bbox, ...
            'kalmanFilter', kalmanFilter, ...
            'age', 1, ...
            'totalVisibleCount', 1, ...
            'consecutiveInvisibleCount', 0);

        % Add it to the array of tracks.
        tracks(end + 1) = newTrack;

        % Increment the next id.
        nextId = nextId + 1;
```

```
    end
    % end

    %% Menampilkan Hasil Pelacakan
    % The |displayTrackingResults| function draws a bounding box and label ID
    % for each track on the video frame and the foreground mask. It then
    % displays the frame and the mask in their respective video players.

    % Convert the frame and the mask to uint8 RGB.
    frameRGB = im2uint8(frameRGB);
    mask = uint8(repmat(mask, [1, 1, 3])) .* 255;

    minVisibleCount = 8;
    if ~isempty(tracks)

        % Noisy detections tend to result in short-lived tracks.
        % Only display tracks that have been visible for more than
        % a minimum number of frames.
        reliableTrackInds = ...
            [tracks(:).totalVisibleCount] > minVisibleCount;
        reliableTracks = tracks(reliableTrackInds);

        % Display the objects. If an object has not been detected
        % in this frame, display its predicted bounding box.
        if ~isempty(reliableTracks)
            % Get bounding boxes.
            bboxes = cat(1, reliableTracks.bbox);

            % Get ids.
            ids = int32([reliableTracks(:).id]);

            % input skala and waktu
            skala = 28/360;
            waktu = 1/obj.reader.FrameRate;

            % velocity of people
            for j = 1 : size(bboxes,1)
                bx1 = max(1,bboxes(j,1));
                bx2 = max(1,bboxes(j,2));
                bx3 = bboxes(j,3);
                bx4 = bboxes(j,4);

                bx3 = min(bx1+bx3,size(frameRGB,2))-bx1-1;
                bx4 = min(bx2+bx4,size(frameRGB,1))-bx2-1;

                vel = velocity(bx2:bx2+bx4,bx1:bx1+bx3);
                kecepatan(j) = mean(vel(:))*obj.reader.FrameRate*skala;
                jarak(j) = waktu*kecepatan(j);

                if kecepatan(j) > 0.1
                    frameRGB =
insertObjectAnnotation(frameRGB,'rectangle',bboxes(j,:),[num2str(kecepatan(j)
,'%.3f'),' m/s'],'FontSize',16,'Color',color{j});
                end
                Jarak(nFrames,j) = jarak(j);
                kecepatan(nFrames,j) = kecepatan(j);
                if nFrames == numFrames
                    JARAK = Jarak(:,j);
                    avg_jarak(j) = sum(JARAK);
```

```
                    frameRGB = insertText(frameRGB,[size(frameRGB,2)/2-
80,size(frameRGB,1)/2],['total = ',num2str(avg_jarak(j),'%.3f'),'
meter'],'BoxColor',color{j},'FontSize',16);
                end
                for j = 1 : size(kecepatan,2)
                SPEED = kecepatan(:,j);
                avg_speed(j) =
sum(SPEED(kecepatan(:,j)>0))/sum(kecepatan(:,j)>0);
                end
            end
        end
    end
     % Display the mask and the frame.
     obj.maskPlayer.step(mask);
     obj.videoPlayer.step(frameRGB);

     nFrames = nFrames + 1;

end
%release video reader and writer
release(obj.maskPlayer);
release(obj.videoPlayer);
% end
```