BAB II
TINJAUAN PUSTAKA

2.1. STUDI LITERATUR

Berdasarkan dari kajian pustaka yang terdiri dari beberapa penelitian mengenai proses perkembangannya analisis sistem deteksi kantuk menggunakan pengolahan citra atau teknologi *artificial inteligence* macam analisa dan metode yang digunakan, maka referensi dari penelitian terdahulu sangat penting untuk di lakukan agar terhindar dari penjiplakan atau duplikasi. Hal ini bertujuan sebagai bahan untuk kontribusi pnelitian bagi penulis agar penelitian tentang tema ini terus berkembang. Berikut berapa ulasan tentang penelitian terdahulu yang pernah di lakukan.

Pengembangannya menggunakan *drowsiness detection* atau *microsleep* banyak dijumpai dan menjadi topik belakangan ini di bidang kesehatan. Misalnya, Anda dapat melihat kejadian kecelakaan beruntun akibat terjadinya kodisi tubuh yang Lelah [1]. Dalam bidang kesehatan, dokter, tenaga medis, maupun pemerintah selalu mengingatkan pentingnya deteksi dini kesehatan agar masyarakat senantiasa sehat dan optimal dalam menjalankan aktifitas sehari-hari sebagai langkah pencegahan dari penyakit dan musibah. Sistem deteksi kantuk digunakan untuk menunjukkan pergerakan karakter. Senatural mungkin, dan dalam bidang medis, kami mengidentifikasi kelainan motorik yang terjadi pada manusia untuk memberikan penanganan lebih lanjut kepada pasien. Khususnya di Indonesia, laboratorium dan alat untuk memonitoring deteksi kantuk saat ini belum dikembangkan, sehingga tenaga kesehatan hanya bisa mengandalkan pengamatan visual langsung untuk. Hal ini mempengaruhi hasil akhir analisis yang disebabkan oleh pengamatan subjektif pasien oleh staf medis. Oleh karena itu, diperlukan penelitian pengolahan citra untuk melakukan gait analysis yang difokuskan pada pendeteksian marker [2].

Dalam hubungannya dengan bidang kedokteran fisik dan rehabilitasi medik di Indonesia, analisis gerak manusia ini menjadi sangat penting untuk membangun basis pengetahuan tentang pola gerak normal orang Indonesia. Hal ini diperlukan karena banyak peralatan dirancang berdasarkan ukuran tubuh orang Eropa,

Amerika, atau Jepang yang berbeda dengan ukuran tubuh rata-rata orang Indonesia. Pengetahuan itu kemudian dapat digunakan untuk membantu dokter dalam mengobati pasien yang mengalami hal tersebut.

Pada penelitian Gupta melakukan penelitian tentang mendeteksi kantuk dan kelelahan pengemudi menggunakan *Histogram Oriented Gradient* (HOG) dan *Support Vector Machine* (SVM) untuk proses pendeteksian dan pengenalan titik-titik wajah. Kemudian, menggunakan *Local Binary Patterns Histograms* (LBPH) untuk menentukan apakah sistem mendeteksi pengemudi yang sama atau berbeda. Dalam pendeteksian kantuk dan menguap, digunakan penggambaran 68 *landmarks* pada daerah wajah, lalu menghitung *Eye Aspect Ratio* (EAR) dan *Mouth Aspect Ratio* (MAR). Nilai EAR dan MAR tersebut akan dibandingkan dengan nilai threshold yang telah disesuaikan berdasarkan jarak dari kamera ke pengemudi, sehingga kondisi mengantuk dan menguap akan diketahui. Sistem akan lebih sensitif dalam mendeteksi kelelahan jika waktu pengemudi dalam menyetir semakin bertambah [3].

Pada tahun 2018 penelitian dilakukan oleh Melissa Yauri, Brian Meneses, Natalia dan Avid Roman yang berjudul "*Design of a Vehicle Driver Drowsiness Detection System through Image Processing using Matlab*" Hasil dari penelitian pada tahapan sistem, dapat disimpulkan Desain sistem terutama didasarkan pada pada analisis pola seseorang saat mengemudi, kapan pengemudi menunjukkan variasi ini di wajahnya, yaitu terdeteksi bahwasanya pengetahuan hierarkis tentang bagaimana sistem akan bekerja, merupakan suatu informasi itu penting karena harus dideklarasikan secara hierarkis dalam perangkat lunak Matlab. Juga untuk setiap tahap karakteristik yang diperlukan diperoleh untuk melakukan penyesuaian saat memproses gambar.

Pada tahun 2021 penelitian Nur Charimmah, Koredianto Usman dan Novamizanti yang berjudul "Deteksi Kantuk Melalui Citra Wajah Dengan Metode *gray Level Co-Occurence Matrix*" melakukan perancangan sistem deteksi kantuk melalui metode *Gray Level Co-occrrence Matrix* (GLCM) dan klasifikasi *Support Vector Machine* (SVM) untuk mendapatkan performansi sistem yang tinggi. Berdasarkan hasil yang telah diperoleh pula, penggunaan citra dengan intensitas berbeda, θ, D, jenis *kernel*, dan nilai σ juga sangat mempengaruhi performansi

sistem sehingga diperlukan parameter yang paling sesuai dengan sistem yang dirancang.

2.2. LANDASAN TEORI

Berdasarkan dari kajian pustaka yang terdiri atas beberapa penelitian mengenai proses klasifikasi citra dengan beberapa ekstraksi dan metode yang berbeda, berikut adalah landasan teori yang bertujuan mendukung penelitian ini.

2.2.1. Deteksi Kantuk

Teknik deteksi kantuk, sehubungan dengan jenis parameter yang digunakan dalam deteksi kantuk, dapat dibagi menjadi dua kategori: (1) Metode intrusif, (2) Metode non-intrusif. Perbedaan utama dalam kedua metode ini adalah bahwa dalam metode intrusif, instrumen dihubungkan ke driver dan kemudian nilai yang direkam diperiksa. Namun, pendekatan intrusif ini memiliki akurasi tinggi, dengan ketidaknyamanan orang, sehingga penerimaan rendah dari metode ini. Teknik deteksi kantuk umumnya diklasifikasikan dalam tiga kelompok: metode berdasarkan keadaan pengemudi, metode berdasarkan kinerja pengemudi, metode hibrida.

Metode berdasarkan status driver, sehubungan dengan jenis parameter yang digunakan untuk deteksi status, dibagi menjadi dua kategori: Teknik berdasarkan sinyal fisiologis dan teknik berdasarkan gambar. Pada metode jenis pertama, ia menggunakan gejala fisiologis dan non-visual yang dibuat dalam tubuh seseorang karena kantuk. Untuk tujuan ini, pertama-tama elektroda dihubungkan ke tubuh pengemudi dan kemudian mereka akan merekam aktivitas listrik di berbagai bagian tubuh termasuk otak, otot, jantung, dan lain-lain [4].

Menganalisis jumlah yang tercatat menentukan tingkat kantuk pengemudi. Studi Jap et al., penggunaan 30 elektroda pada sekelompok pengemudi, menunjukkan bahwa dengan meningkatnya kantuk, tingkat aktivitas theta dan alpha meningkat. Namun, metode ini memiliki akurasi yang baik tetapi tidak direkomendasikan untuk aplikasi praktis karena mengganggu dan menimbulkan ketidaknyamanan pada pengemudi. Kelelahan dan kantuk menciptakan serangkaian tanda-tanda yang terlihat di wajah seseorang bahwa tanda-tanda ini digunakan sebagai dasar metode berbasis gambar. Seringkali, langkah pertama dalam metode

berbasis gambar adalah mendeteksi wajah orang dalam gambar. Kemudian parameter yang diinginkan untuk [5].

2.2.2. Penyebab Pengemudi Mengantuk

Ada beberapa hal yang dapat menyebabkan pengemudi mengantuk pada saat mengemudikan kendaraan antara lain adalah:

1. Tidur kurang dari 8 jam

Menurut sebuah penelitian yang dilakukan di terminal bus terakhir Huancayo kepada 100 sopir bus antar provinsi, disebutkan bahwa dalam 24 jam sebelum survei, 47% pengemudi telah tidur enam jam atau kurang [6]

Contoh ini, pengemudi tidak mematuhi delapan jam yang disesuaikan dengan kualitas tidur untuk mempengaruhi kesehatan. Misalnya, orang sakit setiap saat, penurunan mood dan reaksi cepat untuk menghindari beberapa peristiwa yang tidak pantas. Selain itu, dalam dikembangkan sebuah studi tentang jumlah jam tidur pengemudi dalam bisnis formal dan informal.

Gambar 2. 1 Pengemudi Mobil mengantuk [6]

2. Tidak ada lingkungan tidur yang sesuai

Pengemudi harus melakukan perubahan dengan pengemudi lain untuk istirahat. Tetapi lingkungan tidur tidak memiliki peralatan yang diperlukan, atau kondisi ruang dan ketenangan untuk tidur. Dalam itu menggambarkan sebuah studi di stasiun bus Fiori dan Huancayo. Yang pertama memiliki 81% pengemudi tidur di bagasi mobil dan yang kedua adalah 62%. Selain itu, di terminal bus terakhir pertama, 50% pengemudi tidur saat mobil bergerak dan 42% dari terminal kedua terakhir. Oleh karena itu, persentase terbesar dari lingkungan tidur ada di bagasi mobil. Hal ini menyebabkan berbagai gangguan tidur seperti badan pegal-pegal dan tempat yang tidak sesuai karena adanya

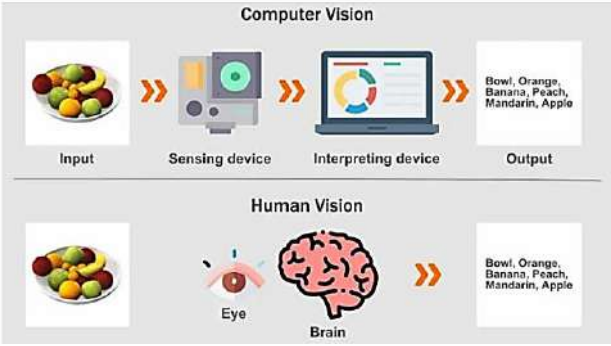
suara bising dari mobil lain. Sehingga orang tersebut menyadari bahwa kebisingan merusak kualitas tidur [7].

3. **Tidak memiliki jadwal kerja teratur**

Pengemudi menyadari hari kerja yang panjang tanpa pemrograman sistematis tidak memungkinkan istirahat yang cukup. Mereka bekerja tidak teratur pada shift siang dan shift malam dan istirahat rata-rata empat sampai lima jam per hari. Di sisi lain, pekerjaan pengemudi tidak berakhir ketika mereka tiba di tempat tujuan, karena mereka harus membersihkan mobil. Dan menurut Tabel 4 menunjukkan jumlah hari kerja pengemudi yang mengkhawatirkan. 20% di bisnis formal dan 29% di perusahaan informal, di mana pengemudi bekerja setiap hari [8].

2.2.3. *Computer Vision*

Computer vision adalah bidang interdisipliner yang membahas mengenai berbagai metode yang digunakan pada komputer untuk bisa mendapatkan pemahaman tingkat tinggi dari gambar atau video digital. Dari perspektif teknik,bidang ini berupaya mengotomasi tugas-tugas yang dapat dilakukan oleh sistem penglihatan manusia. Penglihatan komputer berkaitan dengan ekstraksi otomatis,analisis, dan pemahaman informasi yang berguna dari satu gambar atau urutan gambar. Ini melibatkan pengembangan dasar teoritis dan algoritmik untuk mencapai pemahaman visual otomatis. Sebagai disiplin ilmu, penglihatan komputer berkaitan dengan teori di balik sistem buatan untuk mengekstraksi informasi dari gambar. Data gambar dapat mengambil banyak bentuk, seperti urutan video, pandangan dari beberapa kamera, atau data multi-dimensi dari pemindai medis. *Computer vision* sangat berguna dalam pengembangan aplikasi modern. Konsep *computer vision* dapat dilihat pada Gambar 2.2.



Gambar 2. 2 Konsep *Computer Vision* [9].

Berikut adalah beberapa contoh aplikasi *computer vision* dan penggunaanya:

a. *Face Recognition*: Snapchat dan Facebook menggunakan algoritma pengenalan wajah untuk menggunakan filer dan mengenali wajah pengguna.
b. *Image Retrieval*: *Google Images* menggunakan *content-based queries* untuk mencari gambar yang menyerupai. Algoritma ini menganalisa konten gambar dan menghasilkan hasil berdasarkan best-matched content.
c. *Gaming and Controls*: *Microsoft Kinect* menggunakan stereo vision pada produknya.
d. *Surveillance*: Pendeteksi kelakuan yang mencurigakan pada kamera CCTV.
e. *Biometrics*: Fingerprint, iris dan *face matching* menggunakan metode identifikasi biometric.
f. *Smart Cars*: penggunaan kamera pada mobil untuk mendeteksi rambu lalu lintas dan garis jalan.

2.2.4. Pengolahan Citra

Citra dapat dibagi menjadi dua jenis yaitu citra diam (*still images*) dan citra bergerak (*moving images*). Citra diam adalah citra tunggal yang tidak ditampilkan secara beruntun, sedangkan citra bergerak adalah rangkaian citra diam yang ditampilkan secara beruntun atau berurutan dalam interval waktu yang singkat sehingga memberi kesan pada mata seolah-olah bergerak, misalnya dalam animasi. Dalam penelitian ini selanjutnya citra diam disebut sebagai citra. Menurut cara pembentukannya citra dibedakan menjadi citra kontinyu dan citra diskrit. Citra kontinu dihasilkan dari sistem optik yang menerima sinyal analog, misainya mata manusia atau kamera analog. Citra diskrit dihasilkan dari proses digitalisasi terhadap citra kontinu. Terdapat sistem yang memungkinkan fungsi digitalisasi sehingga tercipta citra diskrit, diantaranya adalah pemindai (*scanner*) atau kamera digital. Citra diskrit lebih sering disebut dengan citra digital [11].

Citra adalah sebuah fungsi intensitas cahaya 2 dimensi $f_{(x,y)}$ dimana parameter x adalah posisi baris dan parameter y adalah posisi kolom, sedangkan f adalah intensitas atau kecerahan dari citra pada koordinat (x, y) [12]. Istilah citra sering disebut juga dengan citra digital atau gambar digital. Istilah citra digital tersebut sering digunakan untuk menjelaskan bahwa citra tersebut sudah diubah menjadi citra secara digital sehingga citra tersebut dapat diproses dengan komputer.

Setiap titik dalam citra mempunyai intensitas warna yang disebut derajat keabuan. Pada umumnya citra berbentuk empat persegi panjang dan dimensi ukurannya dinyatakan sebagai tinggi x lebar [12].

Pengolahan citra (*image processing*) adalah memproses citra khususnya dengan menggunakan komputer menjadi citra yang kualitasnya lebih baik. Pada umumnya operasi pada pengolahan citra diterapkan pada citra apabila:

1. Perbaikan atau memodifikasi citra perlu dilakukan untuk meningkatkan kualitas penampakan atau untuk menonjolkan beberapa aspek informasi yang terkandung di dalam citra.
2. Elemen di dalam citra perlu dikelompokan, dicocokkan atau diukur.
3. Sebagian citra perlu digabung dengan bagian citra yang lain.

Pada prinsipnya pengolahan citra bertujuan memperbaiki kualitas citra agar mudah diinterpretasikan oleh manusia atau mesin (dalam hal ini komputer) [13]. Teknik – teknik pengolahan citra mentransformasikan citra menjadi citra lain. Jadi, masukkannya adalah citra maka keluarannya juga citra. Citra keluaran tersebut mempunyai kualitas citra yang lebih baik daripada citra masukan.

Citra yang kualitasnya tidak seperti yang diinginkan. Misalnya citra yang terlalu gelap atau terlalu terang, bahkan citra yang kabur, warna yang kurang tajam dan sebagainya. Untuk memanipulasi citra yang semacam ini maka diperlukan teknik mengolah citra tersebut. Teknik mengolah citra dinamakan pengolahan citra [14].

2.2.5. Citra Digital

Citra digital merupakan gambar dua dimensi yang dihasilkan dari analog dua dimensi yang kontinu menjadi gambar melalui proses sampling. Gambar analog dibagi menjadi N baris dan M kolom sehingga menjadi gambar diskrit. Citra digital merupakan citra yang dapat diolah computer [15] . Yang disimpan di dalam komputer hanyalah angka-angka yang menunjukkan besar intensitas pada masing-masing piksel. Karena berbentuk data numerik, maka citra digital dapat diolah dengan komputer tetapi tidak dengan citra yang bukan dari data numerik maka tidap dapat diolah dengan komputer.

Citra digital dapat dibedakan menjadi tiga yaitu citra biner (*binary image*), citra keabuan (*grayscale image*) dan citra warna (*color image*).

1. *Color Image* atau RGB (*Red, Green, Blue*)

   Citra berwarna (color image) atau bisa disebut citra RGB adalah jenis citra yang menyajikan warna dalam bentuk komponen merah, hijau, dan biru. Setiap komponen warna menggunakan 8 bit (nilainya berkisar antara 0 sampai dengan 255). Dengan demikian, kemungkinan warna yang bisa disajikan mencapai 2553 atau 16.581.375 (16K) warna [16].

2. *Grayscale* (Keabu-abuan)

   Piksel dari citra digital *grayscale* (keabu-abuan) memilliki gradasi warna mulai dari putih sampai hitam. Rentang warna pada *grayscale* image banyak digunakan dalam dunia kedokteran (X-ray).

2.2.6. *Eye Aspect Ratio*

Eye aspect ratio merupakan rasio dari tinggi dan lebar mata. Jarak antara kelopak mata atas dan kelopak mata bawah dianggap sebagai tinggi dari mata. Jarak dari pojok kiri hingga ke pojok kana mata dianggap sebagai lebar dari mata [17]. Nilai dari *eye aspect ratio* akan meningkat jika mata sedang terbuka dan akan menurun jika mata sedang tertutup. Untuk melakukan hal ini dibutuhkan sistem penanda landmark pada wajah agar posisi dari titik yang digunakan pada persamaan *eye aspect ratio* dapat ditemukan. *Landmark eye aspect ratio* dapat dilihat pada Gambar 2.3.

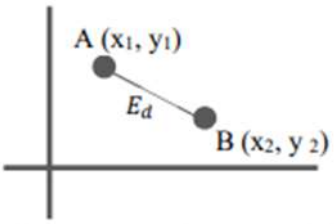
Gambar 2. 3 *Landmark* Dari Mata Pada *Eye Aspect Ratio*

Eye Aspect Ratio (EAR) seseorang dihitung untuk setiap *frame* dari video *real-time*. Dengan demikian, penurunan EAR diharapkan saat pengguna menutup matanya akan kembali ke tingkat normal saat mata terbuka kembali. Pendekatan ini

dapat digunakan untuk mendeteksi kedipan serta keterbukaan mata [18]. Pada saat mata bergerak, kelopak mata juga mengalami perubahan. Saat melihat ke atas mata akan terlihat melebar tidak tertutup kelopak. Saat melihat ke bawah mata akan terlihat tertutup kelopak.

2.2.7. *Euclidean Distance*

Euclidean distance merupakan salah satu metode untuk klasifikasi terhadap objek. *Euclidean distance* merupakan salah satu dari metode jarak yang digunakan untuk mengukur kemiripan terhadap citra yang lain apabila memiliki jarak yang dekat dengan citra tersebut. Metode ini didapatkan dari representasi citra di dimensi banyak (*hyperspace*) sehingga memungkinakan untuk mengatur jarak kemiripan dua citra yang berbeda [19]. Metode *Euclidean distance* dapat dipahami secara bertahap dengan menggambarkan vektor 2 dimensi yang merepresentasikan 2 citra yang berbeda. Kedua vektor ini direpresentasikan dalam diagram kartesian x dan y. Masing-masing dari vektor tersebut memiliki pada nilai pada bidang x dan y.



Gambar 2. 4 Penggambaran Metode *Euclidean Distance* [19]

Gambar 2.4 memperlihatkan dengan jelas bahwa jarak dari kedua vektor tersebut didapatkan dengan menarik garis diagonal yang lurus dari 1 titik ke titik yang lainnya. Penggambaran ini mirip dengan mencari resultan dari dua buah vektor. Metode ini direpresentasikan secara matematis menjadi:

$$E_d = \sqrt{\sum_1^x \sum_1^y (A_{x,y} - B_{x,y})^2} \qquad (2.1)$$

Jarak dari citra A dan citra B didapatkan dengan mengurangi nilai piksel dari citra A dan citra B pada koordinat x dan y yang sama. Metode ini digunakan karena nilai dari jarak yang didapat adalah jarak yang terdekat yang dapat ditempuh oleh dua titik. Perhitungan dengan jarak Euclidean sangat cocok untuk aplikasi pengolahan citra [20].

2.2.8. Algoritma *Haar Cascade Classifiers*

Algoritma Haar Cascade Classifier adalah salah satu algoritma yang digunakan untuk mendeteksi sebuah wajah. Algoritma tersebut mampu mendeteksi dengan cepat dan realtime sebuah benda termasuk wajah manusia. *Algoritma Haar Cascade Classifier* memiliki kelebihan yaitu perihal komputasi yang cepat karena tersebut hanya bergantung pada jumlah piksel dalam persegi dari sebuah image. *Cascade classifiers* digunakan untuk menghasilkan performansi deteksi yang meningkat, namun mengurangi waktu komputasi [21]. *Cascade classifier* akan melakukan klasifikasi secara bertingkat. Hasil *strong classifier* digunakan untuk memasuki pengklasifikasian secara bertingkat pada setiap *sub window* (masukan pada tiap tingkatan) dengan *cascade classifier*. Klasifikasi pada setiap tingkatnya akan menyatakan apakah *sub window* yang diberikan mengandung wajah atau tidak. Jika tidak, maka *sub window* tersebut akan dibuang sehingga tidak akan memasuki tingkatan selanjutnya [22].

2.2.9. *Python*

Python merupakan bahasa pemrograman dengan tujuan umum yang dikembangkan secara khusus untuk membuat *source code* mudah dibaca. Python juga memiliki *library* yang lengkap sehingga memungkinkan programmer untuk membuat aplikasi yang mutakhir dengan menggunakan *source code* yang tampak sederhana [24].

2.2.10. *Library Python*

Library python sendiri memiliki arti sebagai kumpulan modul yang berisi kode yang dapat digunakan dalam program. *Library* pada *Python* sangat banyak dan dapat digunakan untuk membuat program menjadi lebih sederhana. Dengan begitu, *library python* sangat pennting dalam pembelajaran mesin, *data science*, visualisasi dan manipulasi data serta beberapa bidang lainnya. Terdapat lebih dari 137.000 *library python* yang digunakan untuk eliminasi kebutuhan saat menulis kode dari awal. Beberapa *library python* yang sering digunakan adalah sebagai berikut.

Gambar 2. 5 Produk *Library Python*

2.2.10.1. *Numerical Python (Numpy)*

Numpy menjadi salah satu *library* dari *Python* yang banyak digunakan untuk melakukan proses komputasi numerik. *Library* ini mampu untuk membuat objek N-dimensi *array*. Maksud *array* disini adalah sekumpulan variabel yang tipe datanya sama. Penggunaan *Numpy* ini memiliki memiliki kelebihan untuk memudahkan operasi komputasi pada data dan cocok digunakan untuk melakukan akses secara random. *Numpy* ini menyediakan berbagai macam rutinitas untuk operasi cepat pada *array*, termasuk matematika, logika, manipulasi bentuk, *sorting*, *selecting, I/O, discrete Fourier transforms*, *basic linear algebra*, *basic statistical operations*, *random simulation* dan banyak lainnya [42].

2.2.10.2. *Tensorflow*

Tensorflow merupakan *library open source* yang digunakan oleh komputasi numerik dan *project machine learning* berskala besar. Pada *Tensorflow* ini menggabungkan banyak model dan *algoritma machine learning* yakni *deep learning (neural network)*. API tingkat tinggi *TensorFlow* didasarkan pada standar Keras API untuk mendefinisikan dan melatih jaringan saraf. Sehingga mempermudah proses *epoch* dan mendapatkan model yang optimal [25].

2.2.10.3. Keras

Keras merupakan *Application Programming Interface* (API) tingkat tinggi dari *Tensor Flow 2.0* dimana berupa antarmuka yang dapat di dekati dan sangat produktif untuk memecahkan masalah *Machine Learning*, yang berfokus pada *Deep Learning Modern*. Keras ini memiliki 3 karakteristik yaitu *Simple, Flexible* dan *Powerfull*.

2.2.10.4. *Matplotlib*

Pada *library* ini digunakan untuk membuat visualisasi dari angka-angka dan memplotkannya dalam bentuk seperti diagram lingkaran, *histogram, scatterplot*, grafik, dan lain-lain, sehingga banyak digunakan dalam analisa data.

2.2.11. *MediaPipe Framework*

MediaPipe merupakan *open-source framework* untuk membangun *pipelines* untuk melakukan inferensi visi komputer atas data sensorik seperti video atau audio. Menggunakan *MediaPipe*, pipa persepsi seperti itu dapat dibangun sebagai grafik komponen modular. *MediaPipe* saat ini dalam versi alfa pada v0.7, dan mungkin masih ada perubahan API yang melanggar. API yang stabil diharapkan oleh v1.0

Framework MediaPipe Python memberikan akses langsung ke komponen inti *framework MediaPipe C++* seperti *Timestamp*, *Packet*, dan *CalculatorGraph*, sedangkan solusi *Python* yang siap digunakan adalah menyembunyikan detail teknis kerangka kerja dan hanya mengembalikan hasil inferensi model yang dapat dibaca kembali kepada para penelepon.

2.2.11.1. *Packet*

Packet adalah unit aliran data dasar di *MediaPipe*. Packet terdiri dari stempel waktu numerik dan penunjuk bersama ke muatan yang tidak dapat diubah. Dengan *Python*, paket *MediaPipe* dapat dibuat dengan memanggil salah satu metode pembuat paket di modul *mp.packet_creator*. Sejalan dengan itu, *payload* paket dapat diambil dengan menggunakan salah satu metode pengambil paket dalam modul *mp.packet_getter* .

Bahwasanya muatan paket menjadi tidak berubah setelah pembuatan paket. Dengan demikian, modifikasi isi paket yang diambil tidak mempengaruhi muatan sebenarnya dalam paket. Kerangka *MediaPipe Python API* mendukung tipe data *MediaPipe* yang paling umum digunakan (misal, *ImageFrame, Matrix, Protocol Buffer*, dan tipe data primitif) dalam pengikatan inti. Tabel lengkap di bawah ini menunjukkan pemetaan tipe antara tipe data *Python* dan C++ bersama dengan pembuat paket dan metode pengambil konten untuk setiap tipe data yang didukung oleh *API framework MediaPipe Python*.

2.2.11.2. *TimeStamp*

Kerangka *mediapipe* dapat digunakan untuk memproses aliran data baik *online* maupun *offline*. Untuk pemrosesan *offline*, paket didorong ke dalam grafik setelah kalkulator siap memproses paket tersebut. Untuk pemrosesan *online,* satu paket untuk setiap *frame* didorong ke dalam grafik pada saat *frame* tersebut direkam. Kerangka kerja *MediaPipe* hanya membutuhkan paket yang berurutan untuk diberikan stempel waktu yang meningkat secara monoton.

Berdasarkan hasil konvensi, kalkulator dan grafik waktu nyata menggunakan waktu perekaman atau waktu presentasi sebagai stempel waktu untuk setiap paket, dengan setiap stempel waktu mewakili mikrodetik sejak 1 Januari/1970:00:00:00. Hal ini memungkinkan paket dari berbagai sumber untuk diproses dalam urutan yang konsisten secara global. Biasanya untuk pemrosesan *offline*, setiap paket input diproses dan pemrosesan berlanjut selama diperlukan. Untuk pemrosesan online, seringkali perlu untuk menjatuhkan paket input untuk mengimbangi kedatangan *frame* data input. Ketika input datang terlalu sering, teknik yang disarankan untuk menjatuhkan paket adalah dengan menggunakan kalkulator *MediaPipe* yang dirancang khusus untuk tujuan ini seperti *Flow Limiter Calculator* dan *Packet Cloner Calculator*. Untuk pemrosesan *online,* penting juga untuk segera menentukan kapan pemrosesan dapat dilanjutkan.

MediaPipe mendukung ini dengan menyebarkan batas cap waktu antar kalkulator. Batas stempel waktu menunjukkan interval stempel waktu yang tidak berisi paket input, dan memungkinkan kalkulator untuk segera mulai memproses stempel waktu tersebut. Kalkulator yang dirancang untuk pemrosesan waktu nyata harus menghitung batas cap waktu dengan hati-hati untuk memulai pemrosesan secepat mungkin. Misalnya, *Make Pair Calculator* menggunakan *SetOffset* API untuk menyebarkan batas stempel waktu dari aliran *input* ke aliran *output* [26].