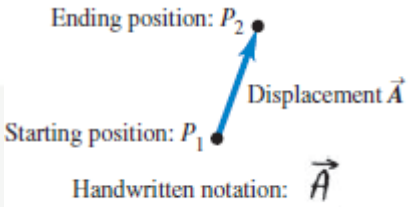
BAB II
LANDASAN TEORI

2.1 Vektor dan Kecepatan

Vektor adalah besaran fisika yang memiliki nilai dan arah. Besaran vektor dinyatakan dengan huruf tebal, miring dan dengan sebuah anak panah diatasnya seperti gambar 2.1 (Bueche dan Hecht, 1997).



Gambar 2.1 Contoh dan penulisan sebuah vektor
(Sumber : Young dan Freedman, 2011)

Berdasarkan contoh diatas, teorema *pythagoras* dapat digunakan untuk mencari nilai sebuah vektor pada bidang dua dimensi (Young dan Freedman, 2011). Nilai sebuah vektor dapat dilihat pada persamaan 2.1.

$$A = \sqrt{A_x^2 + A_y^2} \tag{2.1}$$

Keterangan :

A : Jarak perpindahan vektor.

A_x : Selisih koordinat pada sumbu x.

A_y : Selisih koordinat pada sumbu y.

Kecepatan adalah laju perpindahan suatu benda atau partikel dari satu titik ke titik lainnya. Kecepatan merupakan sebuah vektor, dimana kecepatan memiliki nilai dan arah. Kecepatan terbagi dua macam yaitu kecepatan rata-rata dan kecepatan sesaat. Kecepatan sesaat merupakan laju perpindahan suatu partikel atau benda pada suatu selang waktu tertentu (Young dan Freedman, 2011). Untuk mencari nilai kecepatan sesaat suatu benda atau partikel dapat menggunakan persamaan 2.2.

$$v = \frac{dx}{dt} \tag{2.2}$$

Keterangan :

v : Kecepatan sesaat.

dx : Jarak perpindahan partikel atau benda.

dt : Waktu tempuh partikel atau benda.

Kecepatan rata-rata merupakan laju perpidahan suatu partikel atau benda pada suatu selang waktu (Young dan Freedman, 2011). Untuk mencari nilai kecepatan rata-rata suatu benda atau partikel dapat menggunakan persamaan 2.3.

$$v = \frac{\Delta x}{\Delta t} \tag{2.3}$$

Keterangan :

v : Kecepatan rata-rata.

Δx : Total perpindahan jarak.

Δt : Total waktu tempuh.

2.2 *Background Subtraction*

Background Subtraction merupakan suatu pendekatan yang digunakan untuk mendeteksi benda bergerak didalam sebuah video. Metode dalam *background subtraction* terbagi beberapa macam yaitu *running gausian average*, *temporal mean filter*, *mixture of gaussians*, *kernel density estimation* (*KDE*), *sequential kernel density approximation*, *Cooccurence of image variation*, *eigenbackground* dan lainnya (Piccardi, 2004).

2.2.1 *Mixture of Gaussian* (*MOG*)

Diantara bermacam-macam metode *background subtraction* diatas *metode mixture of Gaussians* (*MOG*) merupakan metode yang paling umum digunakan. Metode ini diperkenalkan oleh Friedman dan Russell pada tahun 1997, metode ini menggabungkan tiga distribusi *gaussian* untuk aplikasi *traffic-surveillance*. Pada tahun 1998 dan 1999 Stauffer dan Grimson mengembangkan metode ini. Stauffer dan Grimson menjelaskan bahwa probabilitas nilai piksel terhadap selang waktu tertentu dengan *metode mixture of gaussians* digambarkan dalam persamaa 2.4 (Piccardi, 2004).

$$p(x_t) = \sum_{i=1}^{K} \omega_{i,t} n(x_t - \mu_{i,t}, \sum_{i,t}) \tag{2.4}$$

Masing-masing distrbusi gaussian (K) dianggap menggambarkan salah satu latar belakang yang diamati atau objek latar depan. Dalam beberapa kasus nilai K dapat diset antar 3 dan 5. *Gaussian* merupakan multi-variasi untuk menggambarkan nilai warna merah, biru dan hijau. jika nilai tersebut diasumsikan independen, *matriks co-variance*, diagonal.

Apabila tiga *channel* diasumsikan sama, maka diagonal dapat disederhanakan dalam persamaan (Piccardi, 2004).

$$\sum_{i,t} = \sigma_{i,t}^2 I \tag{2.5}$$

Pada persamaan 2.4 untuk menjadi model latarbelakang, beberapa kriteria diperlukan untuk membedakan antara latar belakang danlatar depan. Untuk mendapatkan latarbelakang nilai distribusi harus lebih tinggi dan lebih padat terhadap semua distribusi pada rasio antara puncak amplitudo dan standar deviasi, sehingga dapat dirumuskan dalam persamaan 2.5 (Piccardi, 2004).

$$\sum_{i=1}^{B} \omega_i > T \tag{2.6}$$

Selanjutnya setiap pergantian *frame* terdapat dua masalah yang harus dipecahkan, masalah tersebut yaitu menempatkan nilai x_t pada obervasi *frame* terbaru dan estimasi pembaharuan parameter $\omega_{i,t}, \mu_{i,t}, \sigma_{i,t}$. Hal ini dapat dipecah dengan menggunakan algoritma *expectation maximization* (*EM*). Penyelesaian masalah tersebut dengan menggunakan algoritma *EM* dapat ditulis dalam persamaan 2.6 (Piccardi, 2004).

$$(x_t - \mu_{i,t})/\sigma_{i,t} > 2.5 \tag{2.7}$$

Model *background MOG* terbukti berkinerja sangat baik dalam situasi *indoor* maupun *outdoor* dengan variasi yang terbatas. Pendekatan *MOG* merupakan metode yang banyak digunakan untuk *background subtraction* karena sederhana dan efisien (Piccardi, 2004).

2.2.2 Mixture Of Gaussian 2 (MOG2)

Metode mixture of gaussian 2 merupakan peningkatan dari metode *gaussian mixture model*. Dimana pada *metode mixture of gaussian*, setiap warna piksel memiliki nilai distribusi berdasarkan nilai k. Sedangkan pada *MOG2*, nilai warna setiap piksel memiliki nilai distribusi *gaussian* (Zivkovic, 2004). Persamaan yang digunakan dalam untuk mendapatkan latardepan dan latarbelakang pada metode *MOG2* adalah sebagai berikut :

$$\hat{p}(\vec{x}|\mathcal{X}_T, BG+FG) = \sum_{m=1}^{M} \hat{\pi}_m \mathcal{N}(\vec{x}; \hat{\vec{\mu}}_m, \hat{\sigma}_m^2 I) \tag{2.8}$$

Dimana :

M : nilai distribusi gaussian setiap piksel.

$\hat{\pi}_m$: bobot pencampuran.

$\hat{\vec{\mu}}_m$: perkiraan means.

$\hat{\sigma}_m^2$: perkiraan *Gaussian* komponen.

$\vec{x}$: nilai piksel dalam selang waktu.

I : nilai matriks setiap piksel.

2.2.3 *K-Nearest Neighbors* (*KNN*)

Algoritma *k-Nearest Neighbor* merupakan metode non-parametik yang digunakan untuk klasifikasi dan regresi. *Output* yang dihasilkan tergantung pada klasifikasi atau regresi. Dalam klasifikasi, *output* yang dihasilkan adalah anggota kelas. Objek yang diklasifikasi berdasarkan mayoritas dari tetangga k. Dalam regresi, *output* yang dihasilkan adalah nilai properti untuk objek. Nilai ini adalah rata-rata nilai dari tetangga terdekat K (Bishop, 1995). Persamaan yang digunakan adalah sebagai berikut.

$$P(C_k|x) = \frac{K_k}{K} \quad (2.9)$$

Keterangan :

$P(C_k|x)$: Probabilitasi kesalahan vektor x dalam kelas k.

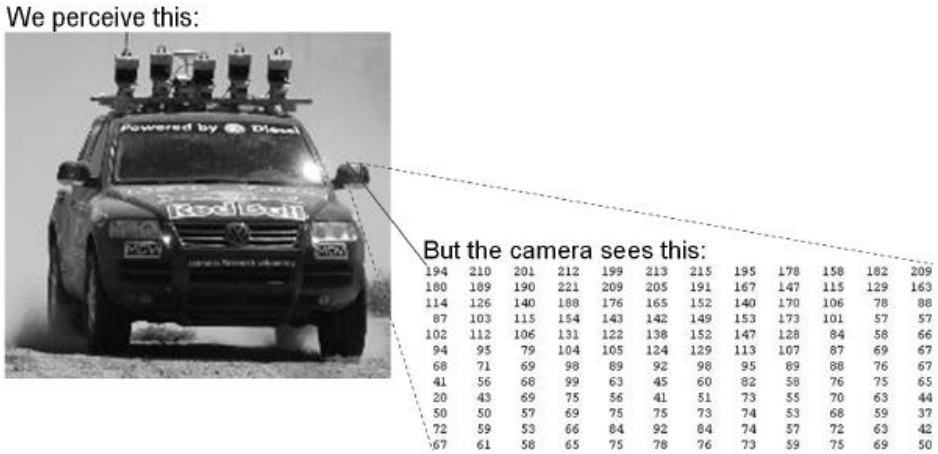
K_k : Nilai mayoritas dalam kelas k.

K : nilai *input* awal.

Untuk meminimalisir probabilitas dari kesalahan klasifikasi vektor x dalam kelas C_K, nilai K_K / K harus besar. Hal ini disebut dengan aturan klasifikasi *k-nearest neighbor* (Bishop, 1995). Metode *KKN* yang simpel (dengan K = 1) mengklasifikasikan objek yang di uji ke dalam klasifikasi yang paling dekat dengan objek tes. Ketika K > 1, anggota K yang terdekat dari *set* akan dipilih dan objek yang di uji dan di klasifikasikan kedalam mayoritas K objek (Coomans dan Massart, 1982).

2.3 *Open Computer Vision* (*OpenCV*)

OpenCV (*open source computer vision*) merupakan sebuah *library* perangkat lunak untuk proses *computer vision* dengan *interface* bahasa pemrograman *C++*, *C*, *Phyton* dan *java*. *OpenCV* juga *support* terhadap sistem operasi *Windows*, *Linux*, *Mac OS*, *iOS* dan *Android*. *OpenCV* juga mendukung pemrosesan paralel *openCL*(OpenCV Developers Team, 2011). Sedangkan *computer vision* merupakan transformasi data dari bentuk 2D/3D foto atau video yang direpresentasikan nilai desimal. Transformasi data tersebut dapat dilihat pada gambar 2.2 (Kaehler dan Bradski, 2008).



Gambar 2.2 Contoh transformasi data pada sebuah citra
(Sumber : Kaehler dan Bradski, 2008)

OpenCV memiliki beberapa fitur untuk pemrosesan suatu citra atau video. Berikut fitur yang banyak digunakan dalam *OpenCV*:

1. *Input*/*Output*

Modul *imgcodecs* dapat mengangani dalam membaca dan menulis *file* gambar. *File* gambar *input* yang diolah dapat disimpan dengan format *png* atau *jpg* dengan perintah sederhana. Untuk video, *OpenCV* memiliki modul *videoio* yang dapat membaca *file* video dengan berbagai format dan menangkap video dari kamera *webcam*. Properti pada video dapat di *setting* seperti ukuran *frame*, kecepatan *frame* dan lainnya (Joshi dkk, 2016).

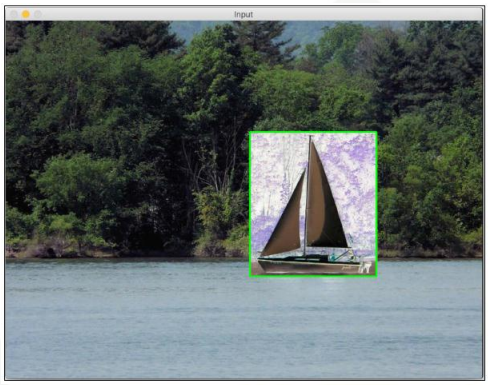
2. Operasi pengolahan citra

Operasi pengolahan citra pada *OpenCV* sebagian besar menggunakan modul *imgproc*. Modul ini mendukung *filtering* pada citra, operasi *morphologi*, transformasi *geometric*, konversi warna, menggambar pada citra, *histogram*, analisa gerak, dan lainya (Joshi dkk, 2016). Salah satu operasi pengolahan citra dapat dilihat pada Gambar 2.3.

Gambar 2.3 operasi pengolahan citra menggunakan transformasi *geometric*
(Sumber : Joshi dkk, 2016)

3. Membangun *Graphical user interface* (*GUI*)

OpenCV memiliki sebuah modul *highgui* yang menangani semua operasi *High-level user interface*. Modul *highgui* memiliki banyak fungsi yaitu membuat *windows* untuk menampilkan citra atau video, membuat *trackbar*, menggambar persegi panjang pada *input windows* kemudian wilayah persegi panjang pada citra diproses sesuai keingan *user* (Joshi dkk, 2016). Salah satu citra yang diproses menggunakan *GUI* seperti terlihat pada Gambar 2.4.



Gambar 2.4 Citra yang diproses menggunakan *GUI*

(Sumber : Joshi dkk, 2016)

4. Analisa video

Analisa video dapat berupa menganalisa gerak antar *frame* yang berurutan dalam video, melacak objek dalam sebuah video, dan lainnya. Semual analisa video tersebut ditangani oleh modul *video*. *OpenCV* juga memiliki modul *videostab*, modul ini berfungsi untuk menstabilkan video ketika menangkap sebuah video dengan memegang kamera ditangan (Joshi dkk, 2016).

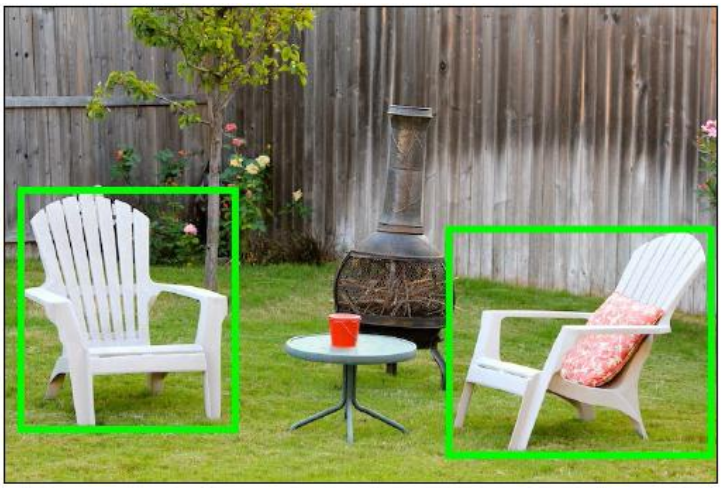
5. Rekonstruksi 3D

Citra 2D dapat dibuat rekayasa rekrontuksi 3D, rekrontuksi 3D ditangani oleh modul *calib3d*. Modul ini digunakan untuk mengkalibrasi kamera, yang dapat mengeperkirakan parameter dari kamera (Joshi dkk, 2016).

6. Deteksi objek

Deteksi objek yaitu mendeteksi lokasi objek pada citra *input*. Deteksi objek sebuah citra tidak bergantung pada jenis objek namun sesuai dengan keinginan *programer*. Apabila *programer* membuat pendeteksi kursi maka hanya lokasi kursi yang akan dideteksi. Deteksi objek ditangani oleh modul *objdetect* dan modul *xobjdetect* menyediakan *framework* untuk

deteksi objek (Joshi dkk, 2016). Salah satu citra yang menggunakan modul *objdetect* dapat dilihat pada Gambar 2.5.



Gambar 2.5 Deteksi kursi pada sebuah citra

(Sumber : Joshi dkk, 2016)

7. Algoritma *Optical Flow*

Algoritma *optical flow* digunakan dalam video untuk melacak suatu objek tertentu dalam setiap *frame*. Modul *OpenCV* yang menangani *optical flow* adalah modul *tracking*. Modul ini berisi algoritma seperti *Lukas Kanade, Farner Back* dan lainnya. Algoritma ini digunakan untuk *object tracking* (Joshi dkk, 2016). Citra yang menggunakan *optical flow* seperti terlihat pada gambar 2.6.

Gambar 2.6 Citra output menggunakan *Lucas Kanade Optical Flow*

(Sumber : Joshi dkk, 2016)

8. Pengenalan wajah dan objek

Pengenalan wajah mengacu pada mengidentifikasi wajah seseorang dalam sebuah citra *input*. Hal ini tidak sama dengan pendeteksi wajah, dimana pendeteksi wajah hanya memberikan lokasi wajah pada citra *input*. Untuk membangun sistem pengenalan wajah , hal pertama yang harus dilakukan adalah menjalankan *face detector* untuk mengidentifikasi lokasi wajah. Kemudian menjalankan *face recognizer* yang dapat mengenali wajah tersebut. Modul *OpenCV* yang menangani *face recognition* adalah modul *face* (Joshi dkk, 2016).

9. Deteksi dan pengelanan teks

Mengindentifikasi teks dalam saebuah citra merupakan hal terpenting. Beberapa aplikasi yang menggunakan deteksi dan pengenelan teks adalah *nameplate recognition*, pengenalan rambu-rambu lalu lintas untuk aplikasi *car-self driving* dan *scan* buku untuk konversi ke buku digital. Modul *OpenCV* yang menangani *detection* and *recognition text* adalah *modul text* (Joshi dkk, 2016). Citra yang menggunakan deteksi teks seperti terlihat pada Gambar 2.7.

Gambar 2.7 Citra *output* dari deteksi teks

(Sumber : Joshi dkk, 2016)