2. TEORI PENUNJANG

Pada bab 2 dibahas tentang metode dan teori yang digunakan dalam pengerjaan tugas akhir ini. Teori penunjang ini tidak hanya berisi hal-hal yang berkaitan langsung dengan judul tugas akhir yaitu di bidang teknologi informatika, tetapi juga yang berkaitan dengan penyajian penulisan laporan akhir tugas akhir ini, seperti misalnya diagram alir informasi untuk proses di dalam *software*. Sedangkan teori yang langsung berkaitan dengan teknologi informasi adalah: *computer vision, computer graphics, OpenCV library, IPL library, image processing,* dan sistem warna, serta *bird view application.*

2.1. Flowchart (Diagram Alir)

Digunakan sebagai teknik untuk mendokumentasikan sebuah sistem ke dalam bentuk gambar atau simbol sehingga lebih mudah dimengerti dibandingkan teknik dokumentasi dengan kata-kata, dengan asumsi pembaca sudah memahami arti dari simbol-simbol yang digunakan dalam *flowchart* tersebut. *Flowchart* menggambarkan sebuah sistem dengan lebih nyata karena menggambarkan keadaan fisik dari data yang dipakai dalam sistem.

Flowchart dapat dibedakan menjadi dua, yaitu

- *Document Flowchart*

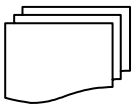
  Menggambarkan aliran dokumen dan informasi antara divisi-divisi dalam organisasi. *Document flowchart* juga menunjukkan di mana dokumen-dokumen dibuat sampai disimpan bahkan digunakan lagi dalam sistem.

- *Computer System Flowhart*

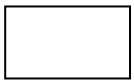
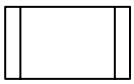
  Menggambarkan hal yang sama seperti *document flowchart*, tetapi pada *computer system flowchart* menggunakan media komputer. Jadi input data dilakukan ke dalam komputer, diproses melalui komputer atau juga secara manual, output yang dihasilkan ditampilkan melalui monitor, dan data disimpan pada media penyimpanan yang disediakan oleh komputer seperti *harddisk* atau disket.

Simbol-simbol *flowchart* terbagi menjadi beberapa kelompok:

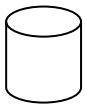
- Simbol Input / output

	*Multiple document*	menunjukkan sekumpulan dokumen yang sama (dicetak beberapa kali)
	Input / output	menunjukkan data yang dimasukkan ke proses atau data yang dihasilkan

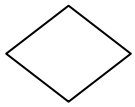
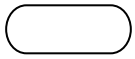
- Simbol Proses

	Proses komputer	proses yang dilakukan oleh / dengan komputer
	*Procedure*	proses komputer yang berbentuk sekumpulan perintah/ kode

- Simbol *Storage* (penyimpanan)

	*Magnetic disk*	data disimpan secara permanen pada magnetic disk

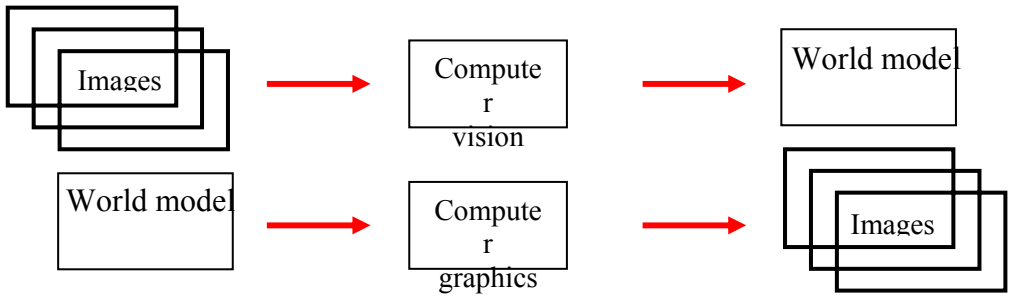
- Simbol *Flow* (aliran) dan yang lainnya

	Processing flow	arah aliran proses
	Data flow	arah aliran data/ informasi
	*Decision*	menunjukkan pengambilan keputus-an (*if*)
	*Off-page connector*	menghubungkan proses apabila berganti halaman, baik masuk /keluar
	Terminal	menunjukkan awal (start), akhir (end), atau interupsi dalam proses/ program

2.2. Computer Vision

Computer vision merupakan salah satu cabang dari *artificial intelligence* (kecerdasan buatan) yang difokuskan pada pengembangan algoritma untuk

menganalisa informasi dari suatu *image* ke dalam bentuk informasi yang sebenarnya di dunia nyata. *Computer vision* mempunyai tujuan utama untuk membuat keputusan yang berguna tentang obyek fisik nyata dan mengorganisasi pemandangan (*scenes*) berdasarkan *image* yang didapat dari sensor. Peran dari *computer vision* adalah sebagai salah satu penyedia data input bagi komputer untuk dapat mengerti keadaan di sekelilingnya. Kemudian dari data input yang telah didapatkan, akan diolah sedemikian rupa sehingga komputer dapat memberikan respon sesuai yang dinginkan untuk menentukan cara penyajian hasil data input tersebut.



Gambar 2.1. Perbandingan *Computer Vision* dengan *Computer Graphics*
(Sumber: Fleet, David. & Szeliski, Richard. Introduction to Computer Vision)

Tampak pada Gambar 2.1. bahwa fungsi *computer vision* berlawanan dengan *computer graphics,* yaitu suatu bidang ilmu untuk menyajikan informasi dunia nyata ke dalam informasi *image*. Berikut adalah beberapa permasalahan dalam *computer vision* yang merupakan fokus utama :

- *Sensing*

  Bagaimana sensor memperoleh *image* dari dunia luar (*World View*) termasuk properti dari dunia seperti material, bentuk, dan iluminasi. Bahkan pada bentuk 3D, termasuk pula geometri, tekstur, *motion*, dan identitas dari obyek di dalamnya disimpan sehingga dapat digunakan oleh komputer.

- *Decoded Information*

  Bagaimana caranya untuk membuka dan mengambil setiap informasi yang ada di dalam *image* itu sehingga komputer dapat memperoleh semua informasi selengkap–lengkapnya untuk kemudian menentukan respon sesuai dengan yang dikehendaki.

- *Using the information*

Memilih informasi apa saja yang benar–benar dibutuhkan dan harus diprioritaskan lebih dari pada yang lainnya. Juga harus dipilih informasi apa yang ada dalam *image* itu yang justru harus dibuang karena dapat mengganggu jalannya sistem.

- *Algorithms*

  Metode atau algoritma apa saja yang dibutuhkan untuk memproses informasi dari *image* dan bagaimana memanfaatkannya.

Beberapa subjek ilmu yang memanfaatkan *computer vision* antara lain:

- *Face recognition* (pengenalan wajah)
- *3D reconstruction* (rekonstruksi struktur 3 dimensi)
- *Motion tracking* (pelacakan gerakan)

2.3. Computer Graphics

Salah satu fungsi utama ilmu *computer graphics* adalah bagaimana mengelola informasi *real world* di dalam komputer sehingga dapat dimengerti dan dikendalikan oleh *user* sebagai pengguna komputer. Bidang ilmu ini berusaha menyederhanakan berjuta-juta informasi yang diperoleh komputer agar lebih mudah dipahami. Dengan kata lain *computer graphics* merupakan sebuah teknik untuk meningkatkan komunikasi manusia dengan komputer. Sebagai contoh adalah matriks rotasi dan translasi yang menggambarkan hubungan suatu objek dalam *image* dengan bidang *world*-nya, yaitu bagaimana bila objek tersebut diputar atau digeser selayaknya bila objek tersebut ada di dunia nyata.

Hal yang telah disebutkan sebelumnya adalah contoh komunikasi pasif dengan komputer. *User* dapat memberikan informasi grafik kepada komputer menggunakan berbagai macam perangkat, seperti *keyboard* dan *mouse.* Informasi input tersebut kemudian diproses dan menjadi output yang juga dapat dimengerti oleh *user.* Hal ini disebut *interactive computer graphics* karena *user* berinteraksi dengan komputer.

Computer graphics mendukung komunikasi dalam bentuk gambar, grafik, dan diagram. Beberapa aplikasi yang memakai *computer graphics*:[1)]

- *Spaceship pilot simulation*

[1)] Harrington, Steven. Computer graphics: A programming approach, page 2. (2nd eds). New York: McGraw-Hill Book Company.

- *Building arsitecture designing*
- *Video games*

2.4. OpenCV - Open Source Computer Vision Library

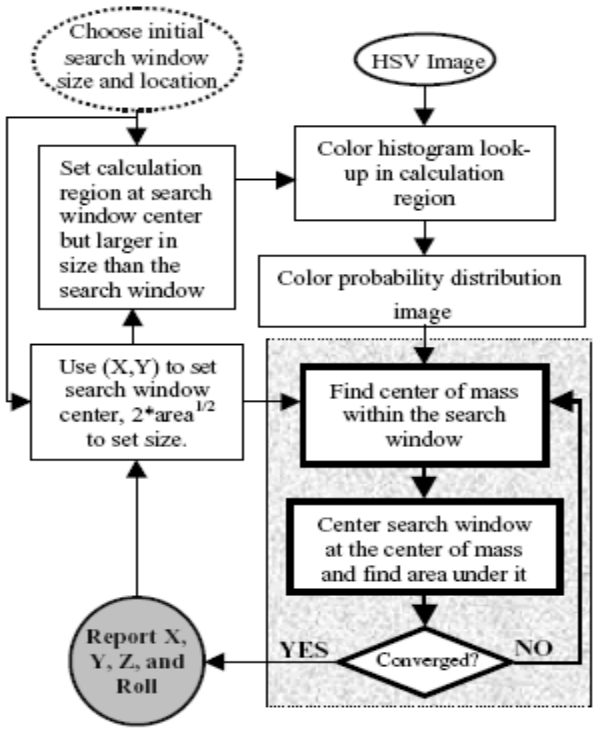
OpenSource Computer Vision Library (OpenCV) adalah komponen pemrograman yang diciptakan oleh Intel Corporation, yang berupa bahasa pemrograman C++/C. OpenCV menyediakan fungsi-fungsi pendukung bagi *computer vision*. Pada dasarnya operasi di dalam OpenCV berupa kumpulan *function* seperti yang lazim digunakan dalam bahasa pemrograman.

OpenCV mengandung sejumlah algoritma dan sampel untuk menangani banyak masalah dalam *computer vision*. Keuntungan utama dari OpenCV adalah sifatnya yang kompatibel, yaitu dapat berjalan baik pada tipe prosesor komputer yang lebih baru, sehingga untuk *running code* yang telah dibuat relatif cepat. Keuntungan lain adalah OpenCV kompatibel dengan *library* lain yaitu IPL.

2.4.1. CamShift[2)]

Salah satu subjek dalam OpenCV yang dimanfaatkan dalam tugas akhir ini adalah metode titik berat atau CamShift. Kependekan dari "Continuously Adaptive Mean-SHIFT", dimana algoritma ini mengaplikasikan algoritma Mean Shift yang mendeteksi objek pada sebuah image berdasarkan contoh warna yang diinginkan. Pada algoritma camshift, input bukan saja hanya berupa sebuah *image* tetapi dapat berupa serentetan *image* yang terus berulang, layaknya frame video.

[2)] Intel image processing library: Reference manual, page 38-41. (1997). USA: Intel Corporation.



Gambar 2.2. Proses Algoritma CamShift
(Sumber: Open source computer vision library: Reference manual)

Tampak pada Gambar 2.2. bahwa *camshift* mengakomodasi algoritma *Mean Shift* (daerah berwarna abu-abu). Adapun metode dalam penerapan *camshift* yaitu:

- Tentukan ukuran daerah untuk menghitung *colour probability distribution* terhadap seluruh bagian *image*. Warna pada daerah yang dicari dianggap sebagai warna *foreground*, sedangkan bagian *image* yang lain sebagai *background.*
- Pilih lokasi awal pencarian.
- Hitung *color probability distribution* pada daerah 2D yang dipusatkan pada titik tengah daerah pencarian dengan ukuran lebih besar dari daerah pencarian.
- Jalankan algoritma Mean Shift berikut untuk mencari titik pusat daerah pencarian dan simpan hasilnya (*area or size*) dan pusat lokasi:
  a. Hitung *mean location* pada daerah pencarian
  b. Pusatkan daerah pencarian pada mean location yang diperoleh di langkah a.
  c. Ulangi langkah a. dan b. hingga daerah pencarian memusat (atau hingga *mean location* bergeser kurang dari batas/*threshold* yang ditentukan.

- Untuk *image* berikutnya, pusatkan pencarian pada daerah (area) yang diperoleh dari langkah ke 4, dan tetapkan ukuran daerah pencarian pada hasil yang baru diperoleh. Ulangi langkah ke 3.

Perhitungan titik berat pada algoritma Mean Shift adalah sebagai berikut:

Hitung moment ke 0:

$$M_{00} = \sum_x \sum_y I(x, y) . \tag{2.1}$$

Hitung moment ke 1 untuk x dan y:

$$M_{10} = \sum_x \sum_y xI(x, y) ; M_{01} = \sum_x \sum_y yI(x, y) . \tag{2.2}$$

Lokasi *mean* daerah pencarian (*centroid*) ditemukan dengan:

$$x_c = \frac{M_{10}}{M_{00}} ; y_c = \frac{M_{01}}{M_{00}} , \tag{2.3}$$

dimana I(x,y) adalah nilai probability pixel pada posisi (*x*,*y*) pada image, dan range *x* dan *y* adalah luas daerah pencarian.

Sedangkan perhitungan camshift memakai *second moment*:

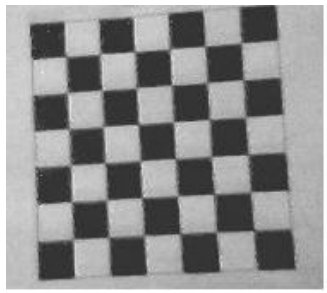
$$M_{20} = \sum_x \sum_y x^2 I(x, y) , M_{02} = \sum_x \sum_y x^2 I(x, y) . \tag{2.4}$$

dan orientasi objek, atau arah dari sumbu axis utama:

$$\theta = \frac{\arctan\left(\frac{2\left(\frac{M_{11}}{M_{00}} - x_c y_c\right)}{\left(\frac{M_{20}}{M_{00}} - x_c^2\right) - \left(\frac{M_{02}}{M_{00}} - y_c^2\right)} \right)}{2} . \tag{2.5}$$

2.4.2. Corner Detection

Fungsi ini dipergunakan sebagai metode pendukung untuk kalibrasi. *Corner* yang akan dideteksi adalah *inner corner* dari pola papan catur sebagai pola kalibrasi. Semisal papan catur berukuran 7x7, maka *innercorner*-nya adalah 6x6 (Gambar 2.3). *Corner* yang terdeteksi merupakan perwakilan posisi pixel *(image view)* dibandingkan dengan posisi 2D *(world view)* yang terlebih dulu diinputkan sesuai besaran yang diinginkan.



Gambar 2.3. Pola Papan Catur 7x7 dengan Inner Corners 6x6

Dalam perhitungan ada 2 cara memperoleh *corner*. Pertama *corner* dapat didefinisikan sebagai area dimana tingkat kurva dikalikan dengan gradient pangkat 3 sebagai nilai maksimum lokal:

$$D_x^2 D_{yy} + D_y^2 D_{xx} - 2 D_x D_y D_{xy}. \tag{2.6}$$

sedangkan yang kedua memakai metode *Sobel*. Operator turunan pertama *Sobel* digunakan untuk mengambil turunan dari nilai x dan y sebuah *image*, di dalam sebuah daerah yang ditentukan untuk dicari cornernya. Sebuah matriks 2x2 dari jumlah turunan x dan y secara subsequensial dibentuk sebagai berikut:

$$\lambda = \frac{\sum D_x^2 + \sum D_y^2 \pm \sqrt{\left(\sum D_x^2 + \sum D_y^2\right)^2 - 4\left(\sum D_x^2 \sum D_y^2 - \left(\sum D_x D_y\right)^2\right)}}{2}. \tag{2.7}$$

Bila $\lambda_1, \lambda_2 > t$, dimana t adalah threshold, maka corner ditemukan di lokasi itu.

2.4.3. Histogram

Histogram adalah sebuah diagram sebagai pendekatan *discrete* dari variabel *probability distribution,* dimana variabel ini bisa berupa skalar atau vektor. Histogram mewakili deskripsi statistik sederhana dari sebuah objek, semisal sebuah *image.* Karakteristik objek diukur selama proses iterasi dalam objek tersebut, contohnya histogram warna sebuah *image* dibangun dari nilai pixel-pixel pada salah satu dimensi warna. Semua nilai yang memungkinkan dari karakteristik histogram multi dimensi lebih jauh lagi diukur pada tiap koordinat. Jika karakteristik yang telah diukur dapat menampung nilai *k*1 yang berbeda pada koordinat pertama, *k*2 pada koordinat ke 2, dan *kn* untuk yang terakhir, maka histogram hasil akan berukuran:

$$size = \prod_{i=1}^{n} k_i. \tag{2.8}$$

Histogram banyak digunakan dalam *image processing* sekaligus *computer vision.* Yang akan dipakai dalam tugas akhir ini adalah histogram 1 dimensi untuk menghitung nilai HSV *color image*. Berikut beberapa jenis histogram dan aplikasinya:

- 1 *dimension histogram* dipakai untuk:
  a. penguatan *grayscale image*,
  b. menentukan tingkat *threshold* (*binary image*) yang optimal,
  c. menentukan objek berwarna lewat *hue histograms back projection*.
- 2 *dimension histogram* dipakai untuk:
  a. analisa dan segmentasi *color images*, normalisasi terhadap cahaya (misal: *red-green* atau *hue-saturation images*),
  b. analisa dan segmentasi daerah pergerakan,
  c. analisa bentuk atau tekstur.
- *Multi dimension histogram* dipakai untuk:
  a. *content based retrieval*
  b. *bayesian-based object recognition*

2.4.4. Camera Calibration

Kita mengukur segalanya dalam ukuran riil dengan menentukan sebuah *reference frame,* yang disebut *world reference frame*. Sebuah obyek dalam *image* dinyatakan dalam koordinat pixel, yang mana ada dalam *image reference frame*. Hanya karena kita tahu jarak antar pixel dalam *image*, itu tidak membuat kita tahu jaraknya dalam dunia nyata atau dalam kondisi yang sesungguhnya. Oleh karena itu, kita perlu suatu persamaan yang bisa menghubungkan *world reference frame* dan *image reference frame* sehingga kita dapat mencari hubungan antara titik koordinat ruang 2D dan titik koordinat pixel pada *image*. Namun perlu diperhatikan bahwa kita tidak bisa langsung menghubungkan antara keduanya. Untuk menghubungkan kita perlu sebuah *frame* referensi yaitu *camera reference frame*.

Ide pokok dari kalibrasi kamera adalah mencari persamaan yang tepat untuk menghubungkan *camera reference frame* dengan *image reference frame* dan sebuah persamaan lainnya untuk menghubungkan *world reference frame* dengan *camera reference frame*. Mencari kedua persamaan tersebut sama halnya dengan mencari

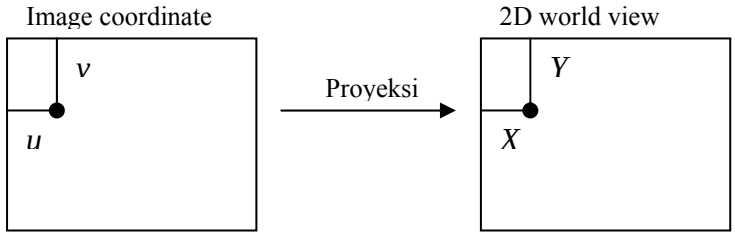
karakteristik kamera, yang mana dalam *computer vision* disebut sebagai parameter intrinsik dan parameter ekstrinsik. Proses kalibrasi ini dilakukan terhadap *webcam* sebagai peralatan yang dipakai dalam tugas akhir untuk mendapatkan data *image*, dimana hasil akhirnya adalah karakteristik optikal dan geometri internal *webcam* (parameter intrinsik) serta orientasi dan posisi 2D dari frame *webcam* yang relatif terhadap sistem koordinat dunia tertentu (parameter ekstrinsik).

Parameter intrinsik dari *webcam* yaitu parameter yang diperlukan untuk menghubungkan koordinat pixel sebuah titik *image* dengan titik yang sama dalam *camera reference frame*, dimana nilainya sama untuk 1 posisi kamera:

- Jarak fokus, yaitu jarak antara lensa kamera dan bidang *image* (disimpan dalam matriks berdimensi 2x1).
- Letak pusat *image* dalam koordinat pixel (*principal point*) dan disimpan dalam matriks berdimensi 2x1.
- Efektifitas besar pixel.
- Koefisien distorsi radial dari lensa yang disimpan dalam bentuk matriks berdimensi 4x1

Parameter ekstrinsik merupakan parameter yang menegaskan hubungan antara lokasi dan orientasi dari *camera reference frame* terhadap *world reference frame,* dimana nilainya berbeda untuk tiap *image*:

- Matriks rotasi, disimpan dalam matriks 3x3.
- Matriks translasi, disimpan dalam matriks 3x1.



Gambar 2.4. Hubungan Titik Koordinat Pada *Image* dan *World Coordinate*

Berikut adalah rumus untuk menjelaskan hubungan *image reference frame* dan *world reference frame* pada Gambar 2.4:

$$pix = A\ |Rt|\ M \quad (2.9)$$

$$pix = s\ m \quad (2.10)$$

$$s\ m = A\ |Rt|\ M \quad (2.11)$$

dimana:

$$pix = \begin{bmatrix} pix1 \\ pix2 \\ pix3 \end{bmatrix}$$ = matriks suatu nilai – belum diskala dengan *s*

s = *scale factor* (baris ke 3 matriks *pix*), berupa skalar (*pix*3)

$$m = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$$ = posisi (x,y) suatu titik pada *image coordinate* (pixel) – setelah diskala dengan *s*

$$A = \begin{bmatrix} fx & (alpha\_c * fx) & cx \\ 0 & fy & cy \\ 0 & 0 & 1 \end{bmatrix}$$ = matriks intrinsik

Keterangan untuk A:

(cx,cy) = koordinat titik pokok (*principal point*) yang menjelaskan posisi pusat image (1/2 dari dimensi *image*)

(fx,fy) = jarak fokus dalam koordinat x dan y (*focal*) dalam unit pixel, dimana *fx* dan *fy* tidak memiliki selisih yang jauh.

alpha_c = sudut antara sumbu x dan y dalam pixel yang berupa skalar. Sesuai standar penggunaan kalibrasi kamera pada umumnya, yang mengasumsikan bahwa bentuk pixel adalah empat persegi panjang bukan bulat, maka metode *camera calibration* yang dipakai inipun mengasumsikan koefisien kemiringan bernilai nol (alpha_c = 0).

$$M = \begin{bmatrix} X \\ Y \\ 0 \end{bmatrix}$$ = posisi (x,y) suatu titik pada *world coordinate* (Z=0)

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$$ = matriks rotasi

$$t = \begin{bmatrix} t_1 \\ t_2 \\ t_3 \end{bmatrix}$$ = matriks translasi

Pada Rumus 2.9. tampak bahwa matriks *R* dan *t* (*Rt*) tidak dikalikan, tapi dijadikan satu matriks berdimensi 3x4. Sedangkan untuk mendapatkan titik 2D dalam pixel (*m*), bagi tiap nilai matriks *pix* dengan *s*.

2.4.5. Lens Undistortion

Setiap kamera umumnya memiliki nilai distorsi yang cukup signifikan, terutama distorsi radial. Perbedaan *image* sebelum dan sesudah dilakukan proses undistorsi ditunjukkan oleh Gambar 2.5. Distorsi dilambangkan dengan 4 koefisien: 2 koefisien radial (k1, k2) dan 2 distorsi tangensial (p1, p2). Semisal (u, v) adalah koordinat pixel yang ideal (bebas distorsi), sedang (ũ, ŭ) adalah nilai pixel yang terundistorsi. Serupa dengan itu, (x, y) adalah ideal dan (ŷ, ž) terdistorsi dari nilai koordinat fisik *image*. Maka:

$$\hat{y} = x + x[k_1 r^2 + k_2 r^4] + [2p_1 xy + p_2(r^2 + 2x^2)] \quad (2.12)$$

$$\check{z} = y + y[k_1 r^2 + k_2 r^4] + [2p_2 xy + p_2(r^2 + 2y^2)] \quad (2.13)$$

dimana $r^2 = x^2 + y^2$. Nilai tengah dari distorsi radial sama dengan *principal point.* Karena ũ = cx + fx u dan ŭ = cy + fy v, dimana cx, cy, fx, dan fy adalah komponen matriks intrinsik, maka sistem resultan untuk melakukan proses undistorsi pada *image* dapat ditulis:

$$\tilde{u} = u + (u - c_x)\left[k_1 r^2 + k_2 r^4 + 2p_1 y + p_2\left(\frac{r^2}{x} + 2x\right)\right] \quad (2.14)$$

$$\breve{u} = v + (v - c_y)\left[k_1 r^2 + k_2 r^4 + 2p_2 x + p_1\left(\frac{r^2}{y} + 2y\right)\right] \quad (2.15)$$

Gambar 2.5. Perbedaan *Image* Sebelum (kiri) dan Sesudah (kanan) Proses Undistorsi
(Sumber: Open source computer vision library: Reference manual)

2.5. IPL - Image Processing Library

Image Processing Library (IPL) juga merupakan salah satu komponen pemrograman buatan Intel yang berupa seperangkat library dalam bahasa pemrograman C++/C dan dikhususkan untuk bidang *image processing*. IPL memakai informasi *header* dari *image*, yaitu bagian yang menyimpan seluruh karakteristik

image, untuk mengetahui format dan struktur *image* agar dapat dilewatkan ke fungsi-fungsi *image processing*, seperti: sistem koordinat dan warna. Adapun format warna *image* yang didukung IPL adalah:

- *Monochrome* ataupun *grayscale image (one color channel)*
- *Color image (3* atau *4 color channels)*
- *Multi-spectral image (any number of channels).*

IPL menyediakan berbagai fungsi operasi penting, seperti:

- Akses dan pembuatan *image*;
- Operasi logika dan aritmatika;
- *Filtering*;
- Konversi ruang dimensi warna;
- Proses geometri;

2.6. Image Processing

Untuk men-*decode* seluruh informasi yang ada pada *image*, yaitu mengubah informasi ke bentuk yang lebih umum, juga untuk memilih informasi mana yang sangat diperlukan atau pun sama sekali tidak diperlukan dan memang harus dibuang dipakailah suatu metode operasi yang disebut sebagai *image processing*. Pengertian sederhana dari *image processing* adalah manipulasi dan analisis suatu informasi *image* oleh komputer. Yang dimaksud dengan informasi *image* disini adalah citra visual pada layar monitor.

Image Processing merupakan teknologi yang paling banyak digunakan bersama *computer vision,* mengingat fungsi utamanya yaitu mengelola informasi *image*. Informasi *image* akan diolah terlebih dahulu dengan *image processing* untuk kemudian diambil informasinya yang berhubungan dengan kenyataan *image* tersebut dengan dunia nyata (bila memang ada) menggunakan *computer vision.*

2.7. Sistem Warna

Warna dari suatu obyek merupakan suatu fungsi dari panjang gelombang yang tak dapat diserap dari cahaya yang dipantulkan dari obyek tersebut. Teori tentang cahaya ini dapat diperoleh dari teori tentang optik milik Isaac Newton, yang menyatakan bahwa sensasi dari warna dapat terpancar karena perbedaan panjang

gelombang sehingga menghasilkan perbedaan efek visual yang dimunculkan oleh suatu obyek diakibatkan oleh panjang gelombang selektif yang dipantulkan dari transmisi cahaya tampak. Newton menyatakan bahwa terdapat 7 warna utama (merah, oranye, kuning, hijau, biru, dan violet) dan dipakai untuk menghitung warna dari spektrum panjang gelombang yang lainnya. Ada beberapa format penyimpanan informasi warna pada *image*, semisal: RGB, HSV, dan lain-lain.

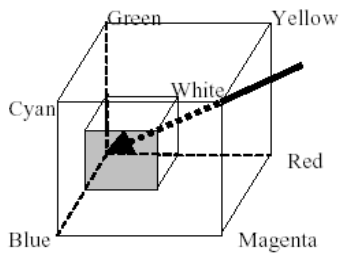
2.7.1. RGB

Model RGB, salah satu model *Additive Color*, digunakan pada monitor komputer. Model ini memiliki 3 warna utama: merah, hijau, dan biru, yang tercipta dengan memancarkan cahaya. Ketiga warna ini dikombinasikan dalam berbagai proporsi tertentu akan menghasilkan variasi warna pada layar. Dikatakan aditif karena bila dicampur akan menghasilkan warna putih.

Warna-warna utama tersebut diukur dari nilai 0-255 atau bisa juga diskala dari 0 (paling gelap) hingga 1 (paling terang). Warna yang dihasilkan dari kombinasi ke 3 warna tersebut adalah hasil dari nilai relatif untuk tiap warna utama. Nilai 0 untuk ke 3 warna menghasilkan warna hitam, sedangkan nilai penuh (255) ketiganya menghasilkan warna putih.

2.7.2. HSV

Sistem warna HSV (*Hue, Saturation, Value*) adalah warna pelengkap yang dihasilkan dari pengaturan warna utama (RGB). Sistem warna HSV merupakan hasil proyeksi sistem warna standar RGB ke diagonal utama yaitu dari putih ke hitam (lihat Gambar 2.6). Semakin kecil nilai V pada HSV akan berkorespondensi dengan semakin gelapnya warna pada RGB.



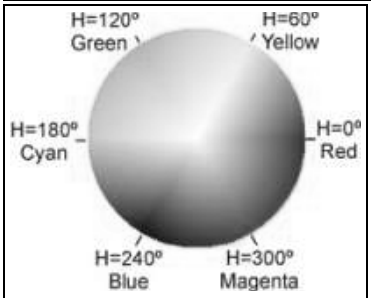
Gambar 2.6. Korespondensi Sistem Warna RGB dengan HSV
(Sumber: Wijono, Hendarto. Pemetaan Bidang 3D Ke Bidang 2D Dengan Kamera)

Hue adalah atribut sensasi visual berdasarkan kemiripan sebuah area terhadap salah satu dari warna-warna yang diketahui, merah, kuning, hijau dan biru, atau kombinasi dua warna dari itu. *Saturation* mengindikasikan kejernihan spektrum warna tersebut dalam hubungannya dengan cahaya. Saturation memiliki batas daerah mulai dari abu-abu netral hingga warna-warna serapan. Contohnya warna terang umumnya memiliki tingkat saturasi tinggi (sedikit abu-abu). *Brightness* atau *value* menyatakan jumlah cahaya yang diterima oleh mata dari warna yang bersangkutan dari batas paling redup sampai paling terang.

Tingkatan *hue* ditunjukkan dari 0 ke derajat 359 (0º = 360º = Merah); sedangkan *saturation* dan *value* ditunjukkan dalam tingkat skala abu-abu, dari 0% ke 100%. Setiap komponen pada sistem warna HSV dapat dijabarkan dalam suatu perhitungan matematis. Perhitungan *hue*, *saturation*, dan *value* ditunjukkan oleh Rumus 2.16, sedangkan perbandingan nilai RGB dengan HSV ditunjukkan oleh Tabel 2.1.

$$\begin{aligned} \text{Hue} = \text{Redh} &: \text{red} - \min(red, green, blue) \\ \text{Greenh} &: \text{green} - \min(red, green, blue) \\ \text{Blueh} &: \text{blue} - \min(red, green, blue) \\ \text{Saturation} &= \frac{\max(red, green, blue) - \min(red, green, blue)}{\max(red, green, blue)} \\ \text{Value} &= \max(red, green, blue) \end{aligned} \tag{2.16}$$

Tabel 2.1. RGB Dibandingkan dengan HSV[3)]

RGB Color space equivalence to HSV Channels							
		HSV Channels values			RGB Channels values		
	Color	**Hue Degree**	Saturation %	Value %	**Red**	**Green**	**Blue**
	Red	0º = 360º	100	100	255	0	0
	Yellow	60º	100	100	255	255	0
	Green	120º	100	100	0	255	0

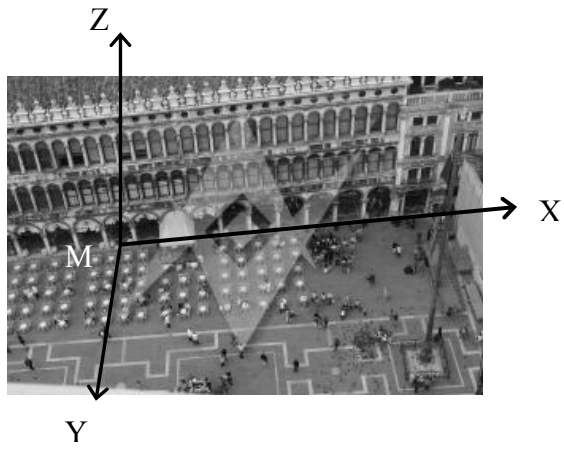
[3)] Wijono, Hendarto. (2003). Pemetaan Bidang 3D Ke Bidang 2D Dengan Kamera. Universitas Kristen Petra.

Tabel 2.1. RGB Dibandingkan dengan HSV (sambungan)

	Cyan	180º	100	100	0	255	255
	Blue	240º	100	100	0	0	255
	Magenta	300º	100	100	255	0	255

2.8. Bird View

Sebutan *bird view* adalah untuk menyebut cara melihat sesuatu dari atas layaknya sudut pandang seekor burung. Aplikasi *bird view* secara garis besar merupakan penerapan dari pengelolaan informasi dunia nyata dimana informasi tersebut diperoleh dari pengamatan lewat atas objek atau hal yang diamati. Informasi itu dalam dunia komputer merupakan sebuah kumpulan *image*, dimana tidak diketahui ketinggian dari objek yang berada dalam *image* tersebut, namun diketahui posisinya terhadap sumbu X dan Y (bidang datar pada permukaan bumi). Referensi koordinat dalam sebuah tampilan *bird view* ditunjukkan oleh contoh Gambar 2.7.



Gambar 2.7. Contoh Koordinat Titik M (putih) dalam Sebuah *Image Bird View*

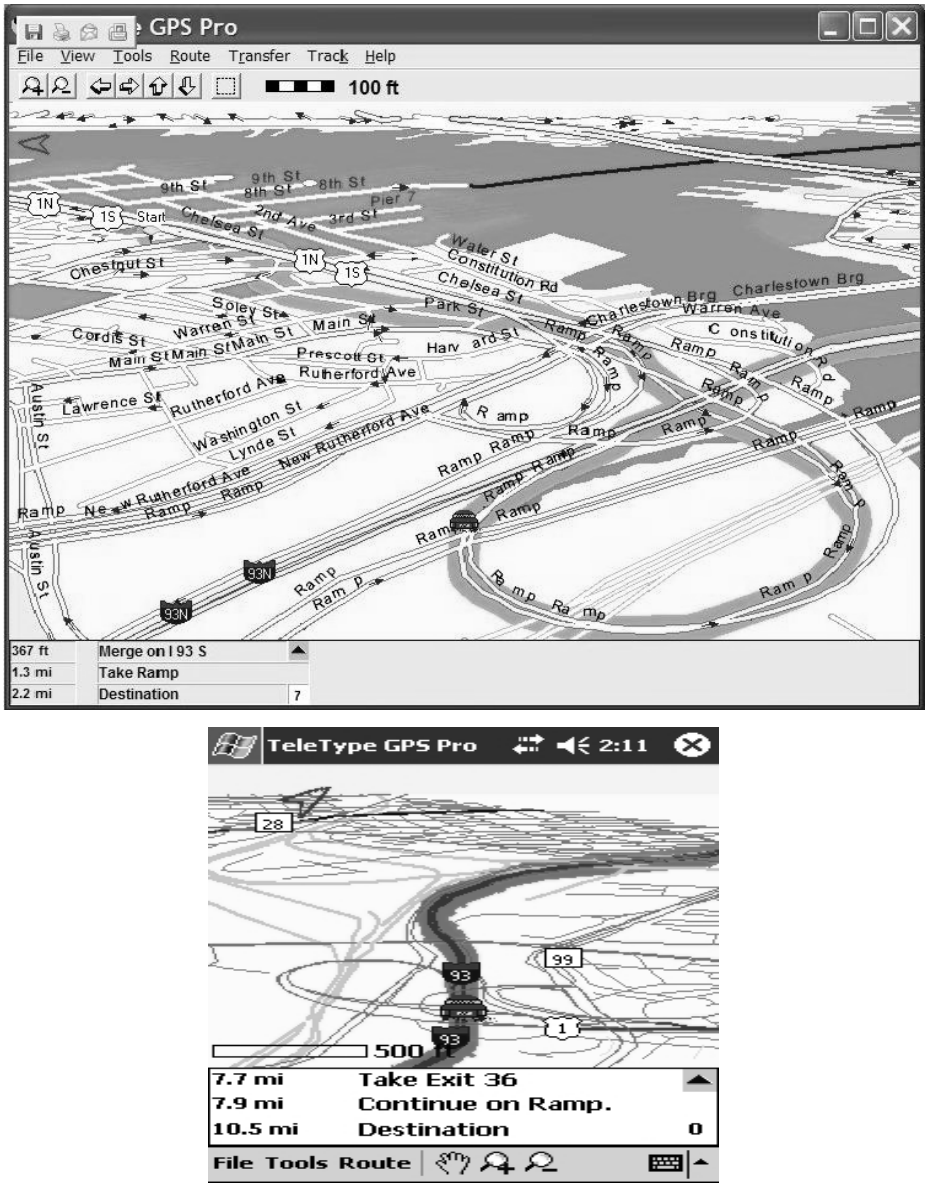
Pada metode *bird view,* hubungan antara titik *2D world view* (M) dan pixel *image* (m) digambarkan sebagai berikut. Keterangan dan penjelasan rumus seperti yang terdapat di subbab 2.4.4. karena rumus 2.17. merupakan bentuk lain dari rumus 2.11. pada subbab 2.4.4, dimana rumus 2.17. berlaku untuk 1 kamera:

$$s\,m = A\,(R\,M + t) \tag{2.17}$$

$$M = R^{-1}\,(A^{-1}\,s\,m - t) \tag{2.18}$$

Pada Gambar 2.7. berikut terlihat contoh aplikasi *bird view* dan hasilnya. Tampak penerapan metode *bird view* dalam pengamatan untuk pemetaan, semisal

untuk sistem GPS dalam suatu daerah lokal, dimana aplikasinya diperuntukkan bagi pemakai jalan untuk mencari jalur alternatif agar terhindar dari kemacetan.



Gambar 2.7. Penggunaan Metode *Bird View* dan Aplikasinya dalam Sistem GPS[4)]

[4)] Teletype GPS: Bird View Screens. http://www.teletype.com/pages/gps/streetbirdview.htm.