BAB II
TINJAUAN PUSTAKA

2.1. *Webcam*

Gambar 2.1 *Webcam*
(Sumber : https://assets.logitech.com)

Kamera web (singkatan dari web dan camera) adalah sebutan bagi kamera waktu-nyata yang gambarnya bisa dilihat melalui *Waring Wera Wanua*, program pengolah pesan cepat, atau aplikasi pemanggilan video. Istilah kamera web cam merujuk pada teknologi secara umumnya, sehingga kata web cam kadang-kadang diganti dengan kata lain yang memberikan pemandangan yang ditampilkan di kamera. Kamera web adalah sebuah kamera video digital kecil yang dihubungkan ke komputer melalui colokan USB atau pun colokan COM.

2.1.1. Fungsi *Webcam*

Fungsi dari *webcam* telah kita ketahui yaitu untuk memudahkan kita dalam mengolah pesan cepat seperti chat melaui video atau bertatap muka melalui video secara langsung. *Webcam* juga berfungsi sebagai alat untuk mentransfer sebuah media secara langsung, namun perlu di sadari kebanyakn pengguna menggunakan piranti ini hanya untuk chat video.

2.1.2. Cara Kerja *Webcam*

Sebuah web camera yang sederhana terdiri dari sebuah lensa standar, dipasang di sebuah papan sirkuit untuk menangkap sinyal gambar, casing (*cover*), termasuk casing depan dan casing samping untuk menutupi lensa standar dan memiliki sebuah lubang lensa di casing depan yang berguna untuk

memasukkan gambar. Kabel support, yang dibuat dari bahan yang fleksibel, salah satu ujungnya dihubungkan dengan papan sirkuit dan ujung satu lagi memiliki *connector*, kabel ini dikontrol untuk menyesuaikan ketinggian, arah dan sudut pandang *web camera*. Sebuah *web camera* biasanya dilengkapi dengan software, software ini mengambil gambar-gambar dari kamera digital secara terus menerus ataupun dalam interval waktu tertentu dan menyiarkannya melalui koneksi internet. Ada beberapa metode penyiaran, metode yang paling umum adalah *hardware* mengubah gambar ke dalam bentuk file JPG dan menguploadnya ke *web server* menggunakan *File Transfer Protocol* (FTP).

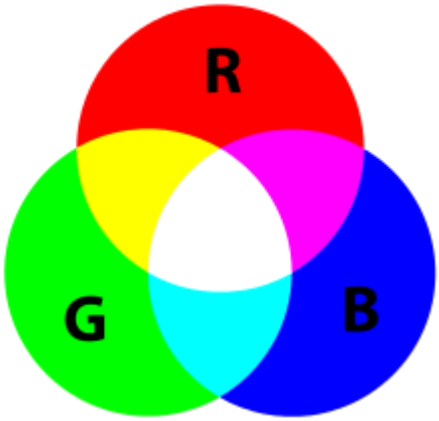
Frame rate mengindikasikan jumlah gambar sebuah software dapat ambil dan transfer dalam satu detik. Untuk *streaming* video, dibutuhkan minimal 15 *frame per second* (fps) atau idealnya 30 fps. Untuk mendapatkan *frame rate* yang tinggi, dibutuhkan koneksi internet yang tinggi kecepatannya. Sebuah web camera tidak harus selalu terhubung dengan komputer, ada web camera yang memiliki software *webcam* dan *web server bulit-in*, sehingga yang diperlukan hanyalah koneksi internet. *Web camera* seperti ini dinamakan "*network camera*". Kita juga bisa menghindari penggunaan kabel dengan menggunakan hubungan radio, koneksi Ethernet ataupun WiFi. [35]

2.2. Pengolahan Citra

Pengolahan citra atau *image processing* adalah suatu proses yang dilakukan oleh suatu sistem pada masukan (*input*) berupa citra (*image*) yang mengasilkan (*output*) sehingga menjadi citra (*image*) yang kualitasnya lebih baik. Pengolahan citra bertujuan untuk memperbaiki kualitas citra agar mudah diinterpretasikan oleh manusia atau mesin (dalam hal ini komputer). Namun dengan berkembangnya dunia komputasi yang ditandai dengan semakin meningkatnya kapasitas dan kecepatan proses komputer, serta munculnya ilmu-ilmu komputer yang memungkinkan manusia dapat mengambil informasi dari suatu citra maka *image processing* tidak dapat dilepaskan dengan bidang *computer vision*.[7]

2.3. Ruang Warna RGB

RGB adalah suatu model warna yang terdiri atas 3 buah warna: merah (*Red*), hijau (*Green*), dan biru (*Blue*), yang ditambahkan dengan berbagai cara untuk menghasilkan bermacam-macam warna. Kegunaan utama model warna RGB adalah untuk menampilkan citra/gambar dalam perangkat elektronik, seperti televisi dan komputer, walaupun juga telah digunakan dalam fotografi biasa. Sebelum era elektronik, model warna RGB telah memiliki landasan yang kuat berdasarkan pemahaman manusia terhadap teori trikromatik.



Gambar 2.2 Warna RGB
(Sumber : https://en.wikipedia.org)

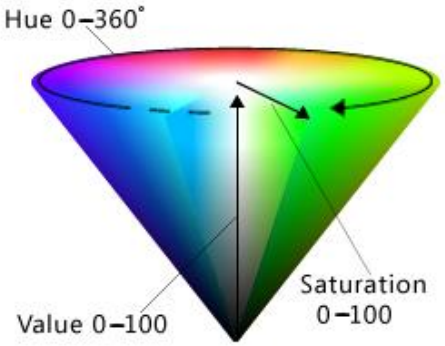
RGB merupakan model warna yang bergantung kepada peranti yang berbeda akan mengenali atau menghasilkan nilai RGB yang berbeda, karena elemen warna (seperti fosfor atau pewarna) bervariasi dari satu pabrik ke pabrik, bahkan pada satu peranti setelah waktu yang lama.

Model warna ini merupakan model warna yang paling sering dipakai. Contoh alat yang memakai mode warna ini yaitu TV, kamera, pemindai, komputer, dan kamera digital. Kelebihan model warna ini adalah gambar mudah disalin / dipindah ke alat lain tanpa harus di-*convert* ke mode warna lain, karena cukup banyak peralatan yang memakai mode warna ini. Kelemahannya adalah tidak bisa dicetak sempurna dengan printer, karena printer menggunakan mode warna CMYK, sehingga harus diubah terlebih dahulu. RGB merupakan model

warna *aditif*, yaitu ketiga berkas cahaya yang ditambahkan bersama-sama, dengan menambahkan panjang gelombang, untuk membuat spektrum warna akhir. [24]

2.4. Ruang Warna HSV

HSV atau *Hue, Saturation, Value*. *Hue* merupakan kombinasi dari warna-warna dasar. *Saturation* menunjukan intensitas dari hue dengan warna dasar yang terang dengan saturasi tinggi, sementara warna-warna pastel saturasinya rendah dan bias juga menjadi monokrom seluruhnya yang tidak memiliki saturasi karena tidak punya intensitas warna di dalamnya. *Value* berhubungan dengan tajam atau tidaknya sebuah warna atau tingkat hitam atau putih pada skala warna. Warna dengan nilai rendah merupakan warna yang hampir hitam dan warna dengan nilai tinggi adalah warna yang murni, yakni yang sepenuhnya pekat.



Gambar 2.3 Warna HSV
(Sumber : https://en.wikipedia.org)

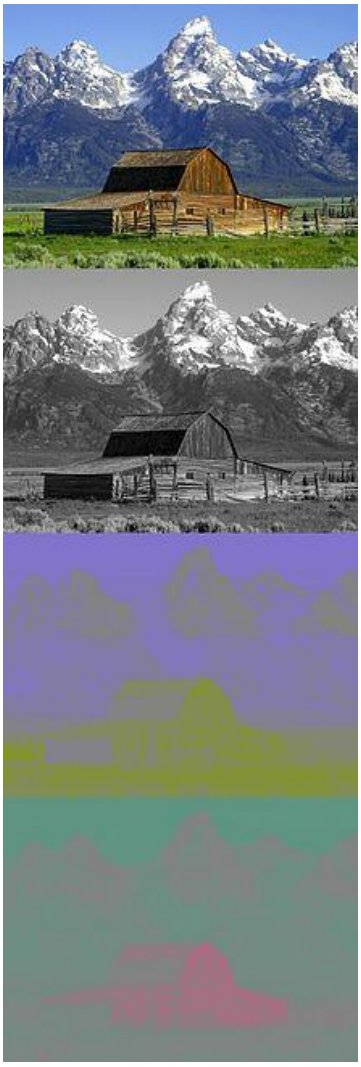
Dalam masing-masing silinder, sudut sekitar sumbu vertikal pusat sesuai dengan "*hue*", jarak dari sumbu sesuai dengan "*saturation*", dan jarak sepanjang sumbu sesuai dengan "*lightness*", "*value*" atau "*brightness*". Karena HSV transformasi sederhana dari perangkat-tergantung model RGB, warna fisik mereka menentukan tergantung pada warna primary merah, hijau, dan biru dari perangkat atau dari ruang RGB tertentu, dan di koreksi gamma digunakan untuk mewakili jumlah mereka primary. Nilai numerik HSV menggambarkan nilai-nilai warna yang berbeda untuk setiap ruang dasar RGB.

Kedua representasi digunakan secara luas dalam komputer grafis, dan satu atau yang lain dari mereka sering lebih mudah daripada RGB, tetapi HSV pun

dikritik karena tidak cukup memisahkan warna membuat atribut, atau kurangnya keseragaman persepsi. [29]

2.5. Ruang Warna YCbCr

YCbCr adalah suatu keluarga ruangan warna. Y merupakan komponen luma sedangkan Cb dan Cr merupakan komponen kroma perbedaan antara biru dan merah. Pada monitor monokrom nilai lumi digunakan untuk merepresentasikan warna RGB yang mewakili intensitas sebuah warna RGB yang diterima. Kroma merepresentasikan corak warna dan saturasi.



Gambar 2.4 Contoh Warna YCbCr
(Sumber : https://en.wikipedia.org)

Y'CbCr bukan merupakan ruang warna mutlak, tetapi merupakan cara pengkodean informasi RGB. Warna yang sebenarnya ditampilkan tergantung pada primary RGB yang sebenarnya digunakan untuk menampilkan sinyal. Oleh karena itu nilai dinyatakan sebagai Y'CbCr diprediksi hanya jika standar RGB kromatisitas warna primer yang digunakan. [36]

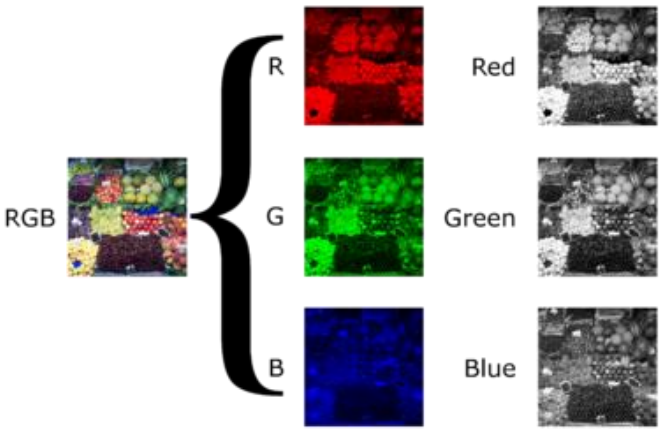
2.6. *Grayscale*

Grayscale atau *greyscale* gambar digital adalah gambar di mana nilai setiap pixel adalah sampel tunggal, yaitu, hanya membawa informasi intensitas. Gambar semacam ini, juga dikenal sebagai hitam-putih, terdiri eksklusif dari warna abu-abu, bervariasi dari hitam pada intensitas terlemah putih di terkuat. gambar *grayscale* yang berbeda dari satu-bit bi-tonal gambar hitam-putih, yang dalam konteks pencitraan komputer gambar dengan hanya dua warna, hitam dan putih (juga disebut bilevel atau gambar biner).

Grayscale atau derajat keabuan merupakan proses awal dalam *image processing* yang merupakan suatu metode dalam sistem yang dibangun untuk dilakukan konversi citra dari berbagai format warna seperti RGB, BGR, RGBA, YCrCb, HSV. Tujuannya adalah untuk melakukan pencarian batas atau *edge* atau tepi citra lalu menggambarkan tepi-tepi tersebut sehingga komputer akan dapat dengan mudah melihat bentuk atau kontur-kontur yang terdapat dengan citra. Setelah komputer dapat melihat bentuk tersebut maka komputer dapat dengan mudah melakukan berbagai proses komputasi terutama yang berhubungan dengan *recognizing, tracking,* dan *motion* pada suatu gambar bergerak baik itu dari *disk* atau pun *real-time* dengan sumber *input* menggunakan kamera atau *webcam*. [18]

Grayscale sebagai saluran tunggal gambar berwarna *multichannel*, gambar berwarna sering dibangun dari beberapa saluran warna ditumpuk, masing-masing mewakili tingkat nilai saluran yang diberikan. Misalnya, gambar RGB terdiri dari tiga saluran independen untuk merah, hijau dan biru komponen warna dasar. CMYK gambar memiliki empat saluran untuk cyan, magenta, kuning dan piring tinta hitam, dll Berikut adalah contoh dari saluran warna membelah dari gambar warna RGB penuh. Kolom di sebelah kiri menunjukkan saluran warna terisolasi dalam warna natural, sementara di sebelah kanan ada ekuivalensi grayscale mereka,

sebaliknya juga untuk membangun citra penuh warna dari saluran grayscale mereka terpisah. Dengan mangling saluran, menggunakan offset, berputar dan manipulasi lainnya, efek artistik dapat dicapai bukannya akurat mereproduksi gambar asli. [5]



Gambar 2.5 Komposisi RGB dari 3 gambar Grayscale
(Sumber : https://en.wikipedia.org)

2.7. Segmentasi Citra

Dalam *Computer Vision*, segmentasi citra adalah proses partisi gambar digital menjadi beberapa segmen (set piksel, juga dikenal sebagai *super-pixel*). Tujuan dari segmentasi adalah untuk menyederhanakan dan / atau mengubah representasi dari sebuah gambar menjadi sesuatu yang lebih bermakna dan lebih mudah untuk menganalisis. Segmentasi citra biasanya digunakan untuk mencari benda-benda dan batas-batas (garis, kurva, dll) dalam gambar. Lebih tepatnya, segmentasi citra adalah proses untuk menempatkan label untuk setiap pixel dalam sebuah gambar sehingga piksel dengan label berbagi karakteristik tertentu yang sama. Hasil segmentasi citra adalah seperangkat segmen yang secara kolektif mencakup seluruh gambar, atau satu set kontur diekstraksi dari gambar (lihat deteksi tepi). Setiap piksel di suatu daerah yang sama sehubungan dengan beberapa properti karakteristik atau dihitung, seperti warna, intensitas, atau tekstur. daerah yang berdekatan secara signifikan berbeda sehubungan dengan karakteristik yang sama (s). [28]

2.8. *Threshold*

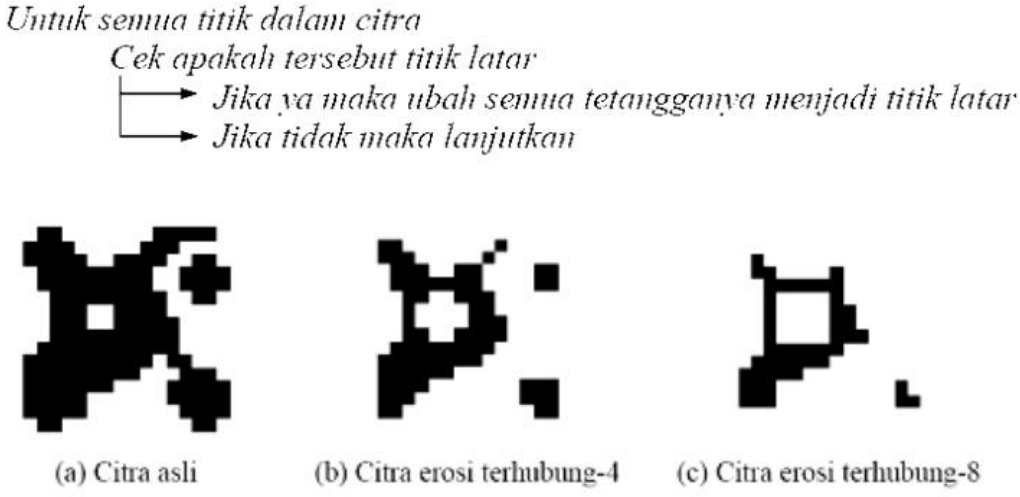
Metode paling sederhana dari segmentasi citra disebut metode *thresholding*. Metode ini didasarkan pada klip-tingkat (atau nilai ambang batas) untuk mengubah citra skala abu-abu menjadi citra biner. Ada juga *histogram thresholding* yang seimbang. Kunci dari metode ini adalah untuk memilih nilai ambang batas (atau nilai ketika beberapa-tingkat yang dipilih). [31]

Thresholding merupakan konversi citra hitam putih ke citra biner dilakukan dengan cara mengelompokkan nilai derajat keabuan setiap pixel kedalam 2 kelas, hitam dan putih. Pada citra hitam putih terdapat 256 level, artinya mempunyai skala "0" sampai "255" atau [0,255], dalam hal ini nilai intensitas 0 menyatakan hitam, dan nilai intensitas 255 menyatakan putih, dan nilai antara 0 sampai 255 menyatakan warna keabuan yang terletak antara hitam dan putih. [18]

2.9. Erosi

Erosi merupakan proses penghapusan titik-titik batas objek menjadi bagian dari latar, berdasarkan *structuring element* yang digunakan. Pada operasi ini, ukuran objek diperkecil dengan mengikis sekeliling objek. [11] Cara yang dapat dilakukan ada 2:

1. Dengan mengubah semua titik batas menjadi titik latar.
2. Dengan menset semua titik di sekeliling titik latar menjadi titik latar.



Gambar 2.6 Proses Erosi [11]

2.10. Dilatasi

Dilatasi merupakan proses penggabungan titik-titik latar menjadi bagian dari objek, berdasarkan *structuring element* yang digunakan. Proses ini adalah kebalikan dari erosi, yaitu merubah latar disekeliling objek menjadi bagian dari objek tersebut. [11] Terdapat 2 cara untuk melakukan operasi ini, yaitu:

1. Dengan cara mengubah semua titik latar yang bertetangga dengan titik batas menjadi titik objek, atau lebih mudahnya *set* setiap titik yang tetangganya adalah titik objek menjadi titik objek.
2. Dengan mengubah semua titik di sekeliling titik batas menjadi titik objek, atau lebih mudahnya *set* semua titik tetangga sebuah titik objek menjadi titik objek.

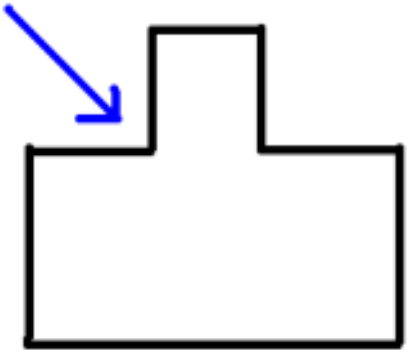
Untuk semua titik dalam citra
Cek apakah tersebut titik obyek
→ *Jika ya maka ubah semua tetangganya menjadi titik obyek*
→ *Jika tidak maka lanjutkan*

(a) Citra asli (b) Citra dilasi terhubung-4 (c) Citra dilasi terhubung-8

Gambar 2.7 Proses Dilasi [11]

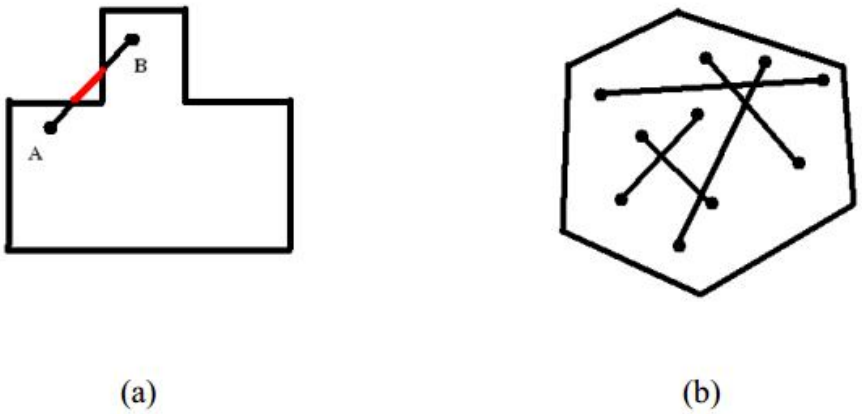
2.11. Convex Hull

Sebelum definisi *convex hull* dijelaskan maka diperlu dijelaskan terlebih dahulu definisi *convex polygon*. *Convex polygon* adalah sebuah *polygon* yang tidak memiliki bagian yang cekung (*concave*). Pada Gambar 2.8 tanda panah menunjuk pada bagian yang cekung dari sebuah *polygon*. [33]



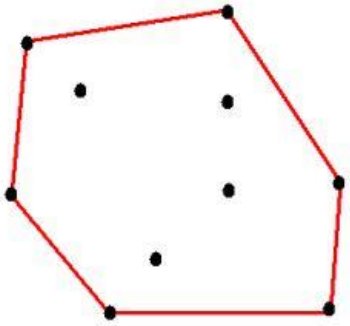
Gambar 2.8 Bukan *convex polygon [33]*

Dalam matematika, terdapat definisi yang tegas mengenai *convex polygon*. Misalkan bahwa dapat dipilih dua titik yang mana saja dari sebuah *polygon* (termasuk pada sisi-sisinya dan area yang ditutupi oleh sisi-sisinya) dan dua titik tersebut dihubungkan dengan sebuah garis lurus. Jika semua garis lurus yang dapat dibentuk dari dua titik di dalam *polygon* tidak melewati batas *polygon* maka *polygon* tersebut bisa disebut *convex polygon*. Gambar 2.9 menunjukkan bahwa terdapat dua titik (A dan B) yang dihubungkan dengan sebuah garis lurus. Dapat dilihat dengan jelas bahwa bagian dari garis (yang ditunjukkan dengan warna merah) melewati batas *polygon*, jadi dapat dikatakan bahwa *polygon* tersebut bukan *convex polygon*. [33]



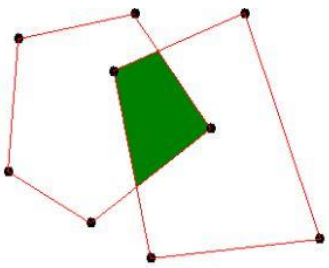
Gambar 2.9 Contoh *convex polygon*, semua garis yang terhubung dari dua titik tidak akan pernah melebihi batas *polygon*. (a) Bukan *convex polygon*
(b) *Convex Polygon [33]*

Setelah definisi *convex polygon* diketahui, maka konsep *convex hull* dapat diketahui. Untuk sekumpulan titik pada sebuah bidang, *convex hull* dari kumpulan titik tersebut adalah *convex polygon* terkecil yang mengelilingi semua titik pada kumpulan titik tersebut. Sebagai contoh, pada Gambar 2.10 terdapat 10 titik, segi enam pada gambar tersebut adalah *convex hull* dari kumpulan titik tersebut. Enam titik yang membentuk segi enam disebut "*hull points*". [33]



Gambar 2.10 Contoh *convex hull [33]*

Diketahui bahwa satu himpunan S dalam sebuah bidang atau dalam sebuah ruang adalah *convex polygon* (atau himpunan *convex*) jika dan hanya jika titik X dan Y ada di dalam S, garis XY harus berada di dalam S. Titik potong dari kumpulan manasaja dari himpunan *convex* adalah *convex* juga, sebagaimana yang ditunjukkan pada Gambar 2.11. Untuk sembarang himpunan W, *convex hull* dari W adalah titik potong dari semua himpunan *convex* yang berisi W. Batas dari *convex hull* adalah sebuah *polygon* dengan semua titik puncak dalam W. [33]



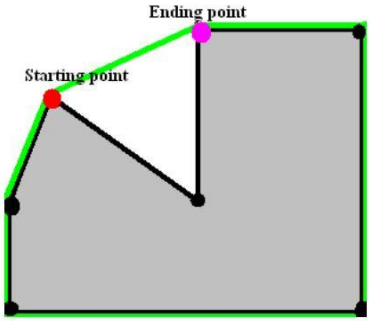
Gambar 2.11 Titik potong manasaja dari himpunan *convex* adalah juga sebuah himpunan *convex [33]*

2.12. Convexity Defect

Convex hull dari kontur lengan bawah dihitung untuk mendapatkan *convexity defect* dari kontur. *Convexity defect* menyediakan informasi yang sangat berguna untuk memahami bentuk kontur. Banyak karakteristik dari kontur yang rumit dapat digambarkan dengan *convexity defect*. Gambar 2.12 menggambarkan *convexity defect* dari bidang yang berbentuk bintang, garis hijau mewakili *convex hull*nya. Seperti yang dapat dilihat pada gambar tersebut, area yang berwarna kuning berada di dalam *convex hull* tapi tidak berada di dalam bintang. Area tersebut disebut dengan *convexity defect*. [33]

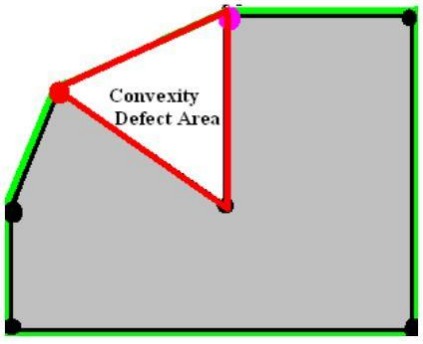
Gambar 2.12 Area yang berwarna kuning mewakili *convexity defect [33]*

Pada pembahasan sebelumnya, dijelaskan bahwa titik-titik yang membentuk *convex hull* harus merupakan bagian dari kontur. Langkah pertama dalam mencari *convexity defect* adalah menemukan titik awal (*starting point*) dari *convexity defect* pada kontur. Titik awal *convexity defect* adalah sebuah titik pada kontur yang juga termasuk dalam titik-titik *convex hull*, tapi titik selanjutnya pada kontur tidak termasuk dalam titik-titik *convex hull*. Gambar 2.13 menjelaskan titik awal dari *convexity defect*. Kontur dicari dengan jalur searah jarum jam. Titik merah adalah titik pertama yang termasuk dalam *convex hull*, tapi titik selanjutnya tidak termasuk dalam *convex hull*. [33]



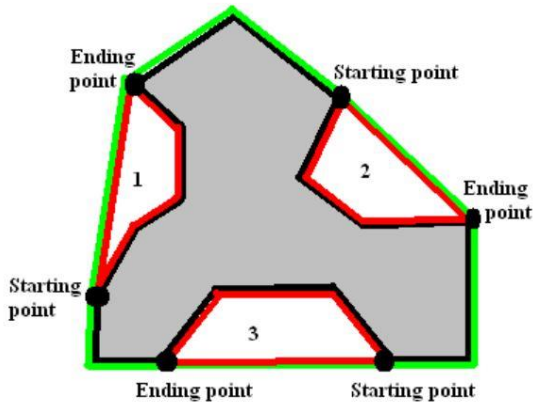
Gambar 2.13 Titik awal (*starting point*) dan titik akhir (*ending point*) *convexity defect [33]*

Setelah definisi titik awal diketahui, titik akhir pun juga demikian. Titik akhir didefinisikan sebagai titik dari kontur yang termasuk dalam titik-titik *convex hull*, tapi titik sebelumnya tidak termasuk dalam titik-titik *convex hull*. Sebagaimana yang dapat dilihat pada Gambar 2.13, titik ungu adalah titik akhir dari *convexity defect*. Dengan menghubungkan titik awal, titik akhir, dan titik diantara keduanya, area dari *convexity defect* dapat diketahui sebagaimana pada Gambar 2.14. [33]



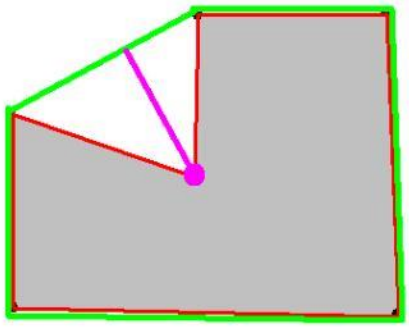
Gambar 2.14 Area *convexity defect [33]*

Setelah semua titik pada kontur telah dicari, berbagai macam *convexity defect* dapat ditemukan sebagaimana pada Gambar 2.15. Setiap *convexity defect* terdiri dari titik awal, titik akhir, dan titik di antara keduanya.



Gambar 2.15 Terdapat tiga *convexity defect* pada kontur [33]

Informasi berguna lainnya yang dapat diperoleh dari *convexity defect* selain titik awal dan titik akhir adalah *depth of defect* dan *depth point. Depth of defect* adalah jarak terjauh dari semua titik pada *defect* hingga ke tepi *convex hull* dari *defect*. Titik pada *defect* yang memiliki jarak terjauh ke tepi *convex hull* dari *defect* adalah *depth point*. [33]



Gambar 2.16 *Depth point* dari *convexity defect*

Sebagaimana yang ditunjukkan pada Gambar 2.16, hanya terdapat satu *convexity defect*. Titik ungu adalah titik yang memiliki jarak terpanjang hingga ke tepi *convex hull* dari *defect* karena panjang dari garis ungu adalah yang terpanjang maka ia termasuk *depth of defect*. Dan titik ungu adalah *depth point*. [33]

2.13. *K-Nearest Neighbors*

Algoritma *k-nearest neighbors* (k-NN atau KNN) adalah sebuah metode untuk melakukan klasifikasi terhadap objek berdasarkan data pembelajaran yang jaraknya paling dekat dengan objek tersebut.

Data pembelajaran diproyeksikan ke ruang berdimensi banyak, dimana masing-masing dimensi merepresentasikan fitur dari data. Ruang ini dibagi menjadi bagian-bagian berdasarkan klasifikasi data pembelajaran. Sebuah titik pada ruang ini ditandai kelas *c* jika kelas *c* merupakan klasifikasi yang paling banyak ditemui pada *k* buah tetangga terdekat titk tersebut. Dekat atau jauhnya tetangga biasanya dihitung berdasarkan jarak Euclidean.

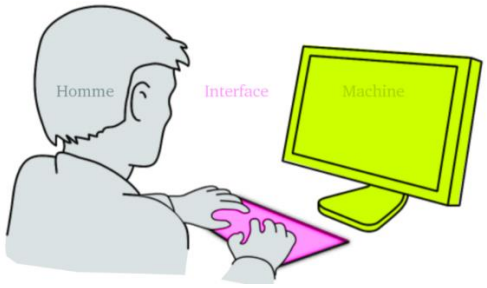
Pada fase pembelajaran, algoritma ini hanya melakukan penyimpanan vektor-vektor fitur dan klasifikasi dari data pembelajaran. Pada fase klasifikasi, fitur-fitur yang sama dihitung untuk data test (yang klasifikasinya tidak diketahui). Jarak dari vektor yang baru ini terhadap seluruh vektor data pembelajaran dihitung, dan sejumlah *k* buah yang paling dekat diambil. Titik yang baru klasifikasinya diprediksikan termasuk pada klasifikasi terbanyak dari titik-titik tersebut.

Nilai *k* yang terbaik untuk algoritma ini tergantung pada data. Secara umumnya, nilai *k* yang tinggi akan mengurangi efek *noise* pada klasifikasi, tetapi membuat batasan antara setiap klasifikasi menjadi lebih kabur. Nilai *k* yang bagus dapat dipilih dengan optimasi parameter, misalnya dengan menggunakan *cross-validation*. Kasus khusus di mana klasifikasi diprediksikan berdasarkan data pembelajaran yang paling dekat (dengan kata lain, $k = 1$) disebut algoritma *nearest neighbor*. Dalam pengenalan pola, algoritma K-Nearest Neighbors adalah metode non-parametrik digunakan untuk klasifikasi dan regresi. [12]

2.14. HCI (*Human Computer Interaction*)

Human Computer Interaction (HCI) atau Interaksi manusia dan komputer ialah disiplin ilmu yang mempelajari suatu hubungan antara manusia serta komputer yang meliputi perancangan, evaluasi, serta implementasi antarmuka pengguna komputer agar mudah digunakan oleh manusia. Sedangkan interaksi manusia dan komputer itu sendiri ialah serangkaian

proses, dialog serta kegiatan yang dilakukan oleh manusia untuk berinteraksi dengan komputer secara interaktif untuk dapat melaksanakan serta menyelesaikan tugas yang diinginkan.



Gambar 2.17 Interaksi Manusia Komputer
(Sumber : http://jeromeabel.net)

Interaksi manusia serta komputer ialah suatu ilmu yang sangat berkaitan dengan disain implementasi serta evaluasi dari sistem komputasi yang interaktif untuk dapat digunakan oleh manusia serta studi tentang ruang lingkupnya, ada interaksi antara satu ataupun lebih manusia serta satu atau lebih komputasi mesin. Agar komputer dapat diterima secara luas serta digunakan secara efektif, maka perlu dirancang secara baik.

Hal tersebut tidak berarti bahwa semua sistem harus dirancang agar dapat mengakomodasi semua orang, tetapi komputer perlu dirancang agar memenuhi serta mempunyai kemampuan sesuai dengan kebutuhan pengguna secara spesifik.

Tahun 1970 mulailah dikenal istilah antarmuka pengguna (user interface), yang juga dikenal dengan istilah sebagai Man-Machine Interface (MMI), serta mulai menjadi topik perhatian bagi peneliti dan perancang sistem.

Perusahaan komputer mulai memikirkan aspek fisik dari antarmuka pengguna sebagai faktor untuk penentu keberhasilan dalam pemasaran produknya.

Istilah *human-computer interaction* (HCI) mulai muncul pada pertengahan tahun 1980-an sebagai bidang studi yang baru. Istilah HCI juga mengisyaratkan bahwa bidang studi ini mempunyai fokus yang lebih luas, tidak hanya pada perancangan antarmuka secara fisik.

HCI didefinisikan ialah sebagai disiplin ilmu yang berhubungan dengan perancangan, evaluasi, serta implementasi sistem komputer interaktif untuk dapat digunakan oleh manusia serta studi tentang fenomena di sekitarnya. HCI pada prinsipnya membuat agar sistem dapat berdialog dengan penggunanya seramah mungkin (user friendly). Tidak hanya pada perancangan layout layar monitor. [9]

2.14.1. Tujuan Interaksi Manusia Komputer

Tujuan utama interaksi manusia komputer ialah untuk :

1. Membuat sistem yang lebih dapat berguna (*usable*), fungsional, aman, produktif, efektif, efisien.
2. Meningkatkan interaksi antara manusia dengan sistem komputer. Sistem yang bermanfaat (*usable*) serta aman (*safe*), ialah sistem tersebut dapat berfungsi dengan baik. Sistem tersebut bisa juga untuk mengembangkan serta meningkatkan keamanan (*safety*), utilitas (*utility*), ketergunaan (*usability*), efektifitas (*efectiveness*) serta efisiensinya (*eficiency*). Para perancang antarmuka manusia serta komputer berharap agar sistem komputer yang dirancangnya dapat bersifat akrab serta ramah dengan penggunanya (user friendly). [9]

2.15. *Computer Vision*

Computer Vision adalah bidang interdisipliner yang berkaitan dengan bagaimana komputer dapat dibuat untuk memperoleh pemahaman tingkat tinggi dari gambar digital atau video. Dari perspektif rekayasa, berusaha untuk mengotomatisasi tugas-tugas yang sistem visual manusia dapat melakukan. Tugas *Computer Vision* mencakup metode untuk memperoleh, mengolah, menganalisis dan memahami gambar digital, dan secara umum, berurusan dengan ekstraksi data dimensi tinggi dari dunia nyata untuk menghasilkan informasi numerik atau simbolik, misalnya, dalam bentuk keputusan. Pemahaman dalam konteks ini berarti transformasi gambar visual (input dari retina) ke deskripsi dari dunia yang dapat berinteraksi dengan proses berpikir lain dan menimbulkan tindakan yang tepat. Pemahaman gambar ini dapat dilihat sebagai menguraikan informasi simbolis dari

data gambar menggunakan model dibangun dengan bantuan geometri, fisika, statistik, dan teori belajar. Sebagai suatu disiplin ilmu, visi komputer berkaitan dengan teori di balik sistem buatan bahwa ekstrak informasi dari gambar. Data gambar dapat mengambil banyak bentuk, seperti urutan video, pandangan dari beberapa kamera, atau data multi-dimensi dari *scanner* medis. Sebagai disiplin teknologi, visi komputer berusaha untuk menerapkan teori dan model untuk pembangunan sistem visi komputer. Sub-domain dari visi komputer termasuk adegan rekonstruksi, acara deteksi, pelacakan video, pengenalan obyek, obyek menimbulkan estimasi, belajar, pengindeksan, gerak estimasi, dan gambar restorasi. [3]

2.16. Microsoft Visual Studio

Gambar 2.18 Microsoft Visual Studio
(Sumber : https://upload.wikimedia.org)

Microsoft Visual Studio merupakan sebuah perangkat lunak lengkap (*suite*) yang dapat digunakan untuk melakukan pengembangan aplikasi, baik itu aplikasi bisnis, aplikasi personal, ataupun komponen aplikasinya, dalam bentuk aplikasi console, aplikasi Windows, ataupun aplikasi Web. Visual Studio mencakup kompiler, SDK, *Integrated Development Environment* (IDE), dan dokumentasi (umumnya berupa MSDN Library). Kompiler yang dimasukkan ke dalam paket Visual Studio antara lain Visual C++, Visual C#, Visual Basic, Visual Basic .NET, Visual InterDev, Visual J++, Visual J#, Visual FoxPro, dan Visual SourceSafe.

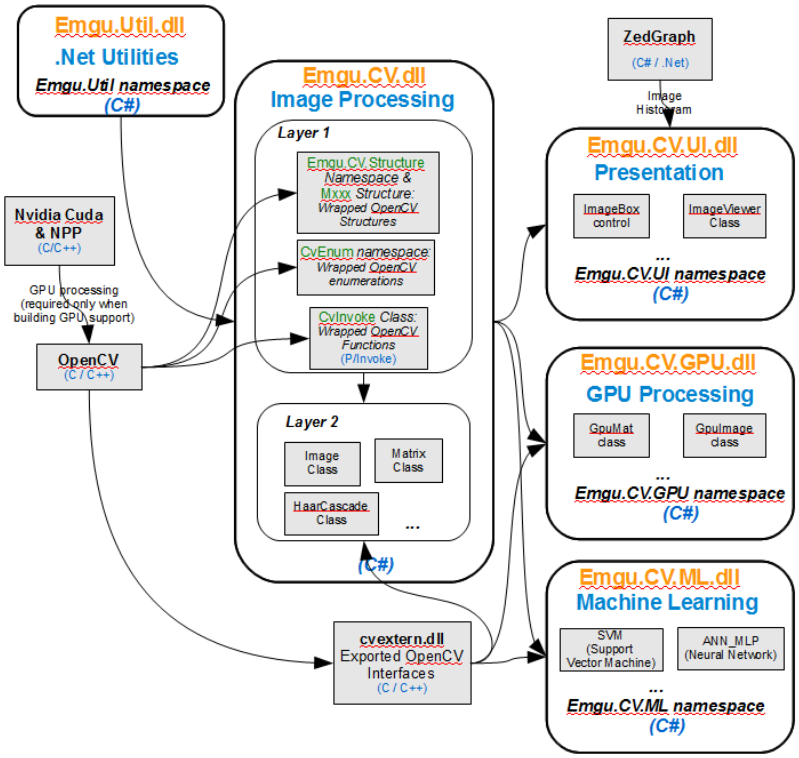
Microsoft Visual Studio dapat digunakan untuk mengembangkan aplikasi dalam *native code* (dalam bentuk bahasa mesin yang berjalan di atas Windows) ataupun *managed code* (dalam bentuk *Microsoft Intermediate Language* di atas .NET *Framework*). Selain itu, Visual Studio juga dapat digunakan untuk mengembangkan aplikasi *Silverlight*, aplikasi *Windows Mobile* (yang berjalan di atas .NET *Compact Framework*). [32]

2.17. #C atau C-Sharp

C# (C-Sharp) merupakan sebuah bahasa pemrograman yang berorientasi objek yang dikembangkan oleh Microsoft sebagai bagian dari inisiatif kerangka .NET Framework. Bahasa pemrograman ini dibuat berbasiskan bahasa C++ yang telah dipengaruhi oleh aspek-aspek ataupun fitur bahasa yang terdapat pada bahasa-bahasa pemrograman lainnya seperti Java, Delphi, Visual Basic, dan lain-lain) dengan beberapa penyederhanaan. Menurut standar *ECMA-334 C# Language Specification*, nama C# terdiri atas sebuah huruf Latin C (U+0043) yang diikuti oleh tanda pagar yang menandakan angka # (U+0023). Tanda pagar # yang digunakan memang bukan tanda kres dalam seni musik (U+266F), dan tanda pagar # (U+0023) tersebut digunakan karena karakter kres dalam seni musik tidak terdapat di dalam *keyboard* standar. [4]

2.18. OpenCV dan EmguCV

OpenCV (*Open Source Computer Vision Library*) adalah sebuah pustaka perangkat lunak yang ditujukan untuk pengolahan citradinamis secara real-time, yang dibuat oleh Intel, dan sekarang didukung oleh Willow Garage dan Itseez. Program ini bebas dan berada dalam naungan sumber terbuka dari lisensi BSD. Pustaka ini merupakan pustaka lintas platform. Program ini didedikasikan sebagaian besar untuk pengolahan citra secara real-time. Jika pustaka ini menemukan pustaka Integrated Performance Primitivesdari intel dalam sistem komputer, maka program ini akan menggunakan rutin ini untuk mempercepat proses kerja program ini secara otomatis.



Gambar 2.19 Arsitektur EmguCV
(Sumber : http://www.emgu.com)

Emgu CV adalah cross platform .Net wrapper untuk perpustakaan OpenCV pengolahan gambar. Memungkinkan fungsi OpenCV untuk dipanggil dari bahasa NET kompatibel seperti C #, VB, VC ++, IronPython dll wrapper dapat dikompilasi oleh Visual Studio, Xamarin Studio dan Unity, dapat berjalan di Windows, Linux, Mac OS X, iOS, Android dan Windows Phone. Emgu CV memiliki dua lapisan pembungkus seperti yang ditunjukkan di bawah ini Dasar lapisan (layer 1) berisi fungsi, struktur dan pencacahan pemetaan yang secara langsung mencerminkan di OpenCV. Lapisan kedua (layer 2) berisi kelas yang campuran di advantanges dari dunia NET.

Keuntungan dari Emgu CV Cross platform Emgu CV seluruhnya ditulis dalam C #. Manfaat adalah bahwa hal itu dapat disusun dalam Mono dan karena itu dapat berjalan pada platform Mono mendukung, termasuk iOS, Android, Windows Phone, Mac OS X dan Linux. Banyak upaya telah dihabiskan untuk memiliki C # implementasi murni karena header harus porting, dibandingkan dengan dikelola C

++ implementasi di mana file header hanya dapat disertakan. Tapi itu layak jika Anda melihat Emgu CV berjalan pada Fedora 10! Plus itu selalu memberikan kenyamanan mengetahui bahwa kode Anda adalah cross-platform. Lintas Bahasa dan dilengkapi dengan contoh kode Emgu CV dapat digunakan dari beberapa bahasa yang berbeda, termasuk C #, VB.NET, C ++ dan IronPython. Pada wiki ini, kami menyediakan contoh untuk semua bahasa mereka, yang tersedia dari bagian Contoh pada halaman Tutorial. Forum Diskusi kami juga tersedia jika Anda memiliki pertanyaan yang berhubungan dengan bahasa pemrograman favorit Anda. Keuntungan lainnya kelas gambar dengan Generik Warna dan Depth pengumpulan sampah otomatis XML Serializable Gambar Dokumentasi XML dan dukungan intellisense pilihan untuk baik menggunakan gambar kelas atau Invoke langsung fungsi dari OpenCV operasi generik pada piksel citra. [2]