BAB II
TINJAUAN PUSTAKA DAN DASAR TEORI

2.1 Tinjauan Pustaka

Seiring dengan berjalannya waktu, perkembangan teknologi *image processing* semakin pesat. Telah banyak penelitian yang dilakukan untuk menggali potensi-potensi teknologi *image processing*, sehingga dapat dimanfaatkan dalam berbagai bidang, seperti pada perindustrian, keamanan dan lain sebagainya. Salah satu penelitian yang berjudul "*Identifikasi dan Tracking Objek Berbasis Image Processing Secara Real Time"*, karya Hendy Mulyawan, et.al (2014), membuat suatu aplikasi yang mampu mengidentifikasi suatu objek serta dapat melakukan *tracking* objek secara *real time.* Pengambilan citra dilakukan dengan menggunakan kamera webcam, sehingga didapatkan gambar objek. Proses pengidentifikasian dan *tracking* objek dilakukan dengan menggunakan metode *template matching*, yaitu dengan membandingkan antara citra objek yang didapat dengan data citra di dalam database. Apabila cocok, maka output yang didapat adalah berupa suara yang sesuai dengan nama objek tersebut. Hasil pengujian penelitan tersebut menunjukkan bahwa tingkat keberhasilan pada siang hari di dalam ruangan adalah 54,4% dengan jarak terbaik 90 hingga 160 cm dan di luar ruangan 34,4% dengan jarak terbaik antara 90 sampai 130 cm. Sedangkan di malam hari di dalam ruangan adalah 59,94% dengan jarak terbaik antara 30 hingga 140 cm dan di luar ruangan 52,16% dengan jarak terbaik 30 sampai 130 cm.

Penelitian lain yang berkaitan dengan *image processing* berjudul *"Segmentasi Citra untuk Deteksi Objek Warna pada Aplikasi Pengambilan Bentuk Citra Rectangle"*, karya Asep Nana H, et.al (2015), memanfaatkan metode segmentasi citra dan *center of grafity* untuk mendeteksi objek warna pada aplikasi pengambilan bentuk citra *rectangle*. Dalam penelitiannya, segmentasi citra dilakukan dengan beberapa tahapan, yaitu pertama memfilter warna dengan metode *euclidean color filtering*, kemudian merubahnya dalam bentuk *greyscale* dan terakhir deteksi blob. Hasil pengujian penelitian tersebut menunjukkan bahwa

objek warna dapat terdeteksi dan dijejaki dengan baik dalam jarak terbaik antara 40 sampai 80 cm dengan tingkat intensitas cahaya terbaik 22 hingga 242 lux.

Selain itu, penelitian karya Ad'han Yulyandri, et.al (2015), yang berjudul "*Sistem Penjejakan Bola menggunakan WebCam berbasis Processor ARM 11*", berhasil membuat suatu sistem penjejak bola yang dibangun pada modul phyCORE-i.MX3, dimana memiliki arsitektur *processor* MCIMX31 dengan *core processor* ARM11, dan menggunakan sensor kamera berupa webcam. Metode *tracking* bola dilakukan dengan menggabungkan antara seleksi berdasarkan warna dan bentuk, sehingga hasilnya akan lebih akurat. Metode seleksi berdasarkan warna menggunakan *Euclidean distance,* sedangkan yang digunakan untuk menyeleksi berdasarkan bentuk adalah dengan menggunakan salah satu fungsi pada *library* OpenCV yaitu cv*HoughCircles.* Fungsi tersebut mampu mendeteksi lingkaran berdasarkan bentuk tepian objek yang didapat dari hasil *thresholding.* Fungsi tersebut juga memungkinkan untuk dapat mendeteksi titik tengah dari lingkaran beserta dengan radiusnya. Nilai titik tengah dan radius tersebut kemudian dijadikan nilai patokan untuk menggambar lingkaran yang melingkari objek (bola) yang terdeteksi dengan menggunakan fungsi cv*Circles*. Hasil penelitian tersebut menunjukkan bahwa sistem dapat mendeteksi bola dengan tingkat keberhasilan 83,3% untuk kondisi cahaya yang stabil. Sedangkan pada saat sistem pencahayaan tidak stabil maka tingkat keberhasilan menjadi 32,5% pada malam hari dan 35% pada siang hari.

Arif Rahman, et.al (2013), dalam penelitiannya yang berjudul "*Colored Ball Position Tracking Method for Goalkeeper Humanoid Robot Soccer*", melakukan penelitian tentang pelacakan posisi bola berwarna untuk robot humanoid kiper. Dalam penelitian tersebut, setiap gambar pada *frame* yang didapat dari kamera webcam kemudian difilter warnanya dengan metode *HSL color filtering.* Selanjutnya dilakukan pendeteksian blob untuk mendeteksi objek bola. Sedangkan, untuk mengetahui posisi dari bola, peneliti menggunakan metode 9 *cells coordinate.* Berdasarkan hasil penelitan tersebut menunjukkan bahwa metode 9 *cells coordinate* dapat memberikan posisi bola secara akurat, sehingga robot humanoid kiper dapat bergerak mengikuti pergerakan bola dan memblokirnya.

Halimatus Sa'diyah, et.al (2016), dalam penelitiannya yang berjudul "*Aplikasi Transformasi Hough untuk Deteksi Garis Lurus"*, melakukan pengujian menggunakan metode transformasi Hough untuk menemukan garis lurus pada suatu gambar. Pengujian tersebut dilakukan pada berbagai bentuk objek. Penelitian dilakukan dengan langkah awal merubah citra menjadi *grayscale,* kemudian mendeteksi tepi citra, selanjutnya mentransformasikan citra tepi dari koordinat kartesian kedalam koordinat polar dan terakhir merekonstruksi gambar garis pada objek tersebut. Hasil penelitian tersebut menyebutkan bahwa tingkat tertinggi keberhasilan deteksi garis lurus dalam suatu objek dengan transformasi Hough adalah 100% dan yang terendah adalah 20%. Hal tersebut dikarenakan garis lurus yang tidak terdeteksi memiliki besar nilai R (nilai hasil pentransformasian ke dalam koordinat polar) di bawah nilai R maksimum yang telah dikalikan dengan nilai ambang, sehingga garis lurus tersebut tidak terdeteksi. Hasil rata-rata persentase keberhasilan total adalah sebesar 90%.

Penelitian lain yang berjudul, "*Deteksi Fitur dan Penentuan Lokasi Robot Pemain Sepak Bola Berbasis Penanda yang Tidak Unik*", karya Ach Hadi Dahlan, et.al (2014), melakukan penelitian pendeteksian bola, gawang dan garis lapangan untuk kemudian dijadikan bahan untuk mengetahui posisi robot di lapangan. Deteksi gawang dilakukan dengan menerapkan transformasi Hough untuk mendapatkan tiang dan mistar gawang. Metode *fitting curva polynomial* digunakan untuk mendapatkan jarak antara robot dengan gawang. Setelah itu, dilakukan metode *landmark* dan *triangulation* untuk mendapatkan koordinat robot (x, y). Sedangkan untuk membedakan gawang musuh dengan gawang sendiri adalah dengan memasang sensor orientasi, sehingga didapatkan sudut arah robot (θ). Hasil dari penelitian tersebut menyebutkan bahwa tingkat *error* koordinat robot di lapangan adalah sebesar 0,1154 meter.

Rama Okta Wiyagi, et.al (2014), dalam penelitiannya yang berjudul "*Identifikasi Titik Api Lilin Berbasis Nilai HSV, Threshold dan Momen Citra untuk Aplikasi Robot Pemadam Api*", menggunakan *single board* komputer berupa Raspberry Pi dan sebuah webcam sebagai pengganti sensor *phototransistor,* TPA81 *thermophile* dan UVtron untuk mendeteksi titik api lilin. Metode yang digunakan

adalah dengan cara mengkonversi citra RGB ke dalam ruang warna HSV, kemudian melakukan *thresholding* warna berdasarkan nilai *range* warna HSV titik api lilin. Selanjutnya, untuk mendapatkan posisi titik api lilin maka digunakan metode analisis momen. Hasil penelitian tersebut menyebutkan bahwa sistem mampu mendeteksi titik api lilin hingga pada jarak maksimal 225 cm, dengan sudut pembacaan horizontal sebesar 60˚ dan vertikal sebesar 40˚. Penelitian tersebut menggunakan tingkat resolusi citra 320 x 240 piksel dan mendapatkan nilai FPS sebesar 8.129.

2.2 Dasar Teori

2.2.1 Bahasa Pemrograman Python

Python adalah bahasa pemrograman interpretatif multiguna. Tidak seperti bahasa lainnya yang sulit untuk dibaca dan dipahami, Python lebih menekankan pada keterbacaan kode agar lebih mudah untuk memahami sintaks. Hal tersebut membuat Python sangat mudah dipelajari baik untuk pemula maupun untuk yang sudah menguasai bahasa pemrograman lain.

Python muncul untuk pertama kali pada tahun 1991, dirancang oleh Guido Van Rossum. Bahasa Python mendukung hampir semua sistem operasi, bahkan untuk sistem operasi Linux, hampir semua distronya sudah menyertakan Python di dalamnya. Hingga saat ini bahasa pemrograman Python masih terus dikembangkan oleh Python Software Foundation.

Dibandingkan dengan bahasa pemrograman lainnnya, seperti C/C++, Python lebih lambat. Meskipun demikian, Python memiliki fitur penting, yaitu dapat dengan mudah melakukan *extend* dengan bahasa C/C++. Jadi dengan demikian dapat menuliskan kode program yang membutuhkan komputasi tinggi dalam bahasa C/C++ dan kemudian dibuat Python *wrapper*, sehingga kode C/C+ tersebut dapat digunakan sebagai modul dalam bahasa Python. Fitur tersebut menjadikan Python memiliki 2 keuntungan, yaitu pertama menjadikan Python secepat bahasa C/C++ dalam melakukan komputasi dan kedua adalah menjadikan sangat mudah dan simple dalam melakukan *coding* dengan Python.

Selain itu, di dalam pemrograman bahasa Python juga terdapat *library* Numpy yang akan membuat pekerjaan numerik menjadi lebih mudah. Numpy merupakan sebuah *library* yang telah dioptimalkan secara maksimal untuk keperluan operasi numerik dan menggunakan *syntax* layaknya pada MATLAB. Tidak hanya Numpy, Python juga mensupport banyak *library* lain seperti SciPy, Matplotlib dan lain sebagainya. Jadi dengan demikian, bahasa pemrograman Python sangat cocok untuk digunakan dalam bidang *image processing.*

2.2.2 OpenCV

OpenCV (*Open Computer Vision*) merupakan sebuah API (*Application Programming Interface*) *library open source* yang sangat cocok digunakan untuk *image processing.* OpenCV pertama kali dibuat oleh Intel pada tahun 1999 oleh Gary Bradsky dan mulai dirilis keluar pada tahun 2000. Pada tahun 2005, OpenCV berhasil memenangkan DARPA *Grand Challenge*. Saat ini, OpenCV telah mendukung banyak algoritma yang terkait dengan *Computer Vision* dan *Machine Learning*. Selain itu, saat ini OpenCV juga dapat digunakan dalam berbagai macam bahasa pemrograman, seperti C++, Python, Java, dan lain sebagainya. Tidak hanya itu, OpenCV juga tersedia dalam berbagai *platform*, seperti Windows, Linux, OSX, Android, IOS, dan lain sebagainya. OpenCV juga dapat melakukan operasi *high-speed GPU* dengan menggunakan antarmuka-antarmuka berbasis CUDA dan OpenCL. Kombinasi terbaik untuk dapat melakukan operasi berkecepatan tinggi tersebut adalah dengan perpaduan antara OpenCV, C++ API dan bahasa pemrograman Python.

Pada dasarnya OpenCV terdiri atas 5 library, yaitu:

1. CV : algoritma *image processing* dan *Vision.*
2. ML : *machine learning library*.
3. Highgui : GUI, image, dan video I/O.
4. CXCORE : struktur data, *support* XML dan fungsi-fungsi grafis.
5. CvAux : modul *open source computer vision* dari Intel *Corporation.*

OpenCV memiliki banyak sekali fitur yang dapat dimanfaatkan, berikut adalah beberapa fitur utama dari OpenCV:

1. *Image* dan *Video* I/O

   Fitur ini memungkinkan untuk membaca data gambar dari file, atau dari umpan *video* langsung. Begitu pula sebaliknya dapat menciptakan file gambar maupun *video*.

2. *Computer Vision* dan *Image Processing* (API tingkat *low* dan *mid*)

   Interface ini dapat melakukan eksperimen uji coba dengan berbagai standar algoritma *computer vision,* seperti *line detection*, *edges detection*, proyeksi elisps , *image pyramid*, *template matching,* berbagai macam transformasi (Fourier, cosine diskrit, *distance transform*) dan lain sebagainya.

3. *Computer Vision High Level*

   OpenCV juga memungkinkan penggunanya untuk melakukan operasi tingkat tinggi, seperti mendeteksi wajah, pengenalan wajah, *optical flow,* dan lain sebagainya.

4. AI (*Artificial Intelegent*) dan ML(*Machine Learning*)

   OpenCV juga menyediakan paket *library* ML (Machine Learning), tentunya hal ini akan sangat mendukung komputer dalam mendeteksi, mengambil keputusan dan melakukan aksi terhadap suatu objek.

5. Sampling Gambar dan Substraksi

   Fitur ini meliputi substraksi *subregion* dari gambar, *random* sampling, *rotating*, dan lain sebagainya.

6. Gambar Biner

   OpenCV juga memiliki fitur untuk dapat membuat dan menganalisis gambar biner.

7. Pemodelan 3D

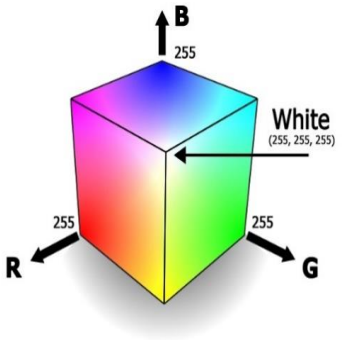
   Fitur ini sangat bermanfaat untuk *mapping* dan *localization*, baik menggunakan *stereo* kamera maupun *single* kamera dengan berbagai sudut pandang.

2.2.3 Ruang Warna (*Color Space*)

Pengertian warna itu sendiri merupakan sebuah hasil persepsi dari cahaya dalam spektrum wilayah yang terlihat oleh retina mata dan memiliki panjang gelombang antara 400nm sampai 700 nm. Sebagai contoh, suatu objek memiliki warna biru karena objek tersebut memantulkan cahaya dengan panjang gelombang antara 450nm hingga 490nm, sehingga retina akan mengasumsikan bahwa objek tersebut berwarna biru. Sedangkan yang dimaksud dengan ruang warna merupakan model matematis abstrak yang menggambarkan cara agar suatu warna dapat direpresentasikan sebagai baris angka dan umumnya dengan menggunakan nilai-nilai dari tiga atau empat buah komponen warna. Beberapa ruang warna telah ditawarkan dan setiap permodelan tersebut memiliki sistem koordinat warna yang spesifik, dan setiap titik pada domain ruang warna tersebut hanya merepresentasikan satu warna yang spesifik. Setiap jenis ruang warna memiliki kelebihan dan kekurangannya masing-masing. Jadi dengan kata lain, setiap jenis ruang warna hanya akan efektif pada suatu spesifik kasus tertentu. Terdapat beberapa jenis ruang warna, yaitu antara lain RGB, LUV, CMYK, HSL, HSI, HSV dan lain sebagainya.

1. RGB

   Permodelan atau ruang warna RGB merupakan yang paling populer dan sering digunakan. Penggunaannya antara lain pada monitor dan kamera digital. Sesuai dengan namanya RGB (Red, Green, Blue), ruang warna jenis ini menggunakan tiga buah warna dasar, yaitu merah, hijau dan biru sebagai warna pembentuk berbagai macam warna-warna lainnya. Tiga warna tersebut dipilih berdasarkan penelitian sebelumnya, bahwa warna-warna yang dapat ditangkap oleh retina mata manusia adalah hasil perpaduan antara cahaya dengan panjang gelombang yang berbeda-beda. Dimana kombinasi atau perpaduan warna yang mampu menghasilkan rentang warna paling luas adalah perpaduan warna antara merah, hijau dan biru.



Gambar 2.1 Pemodelan ruang warna RGB
(https://pemrogramanmatlab.files.wordpress.com/2016/06/ruang-warna-citra-rgb.jpg)

2. HSV

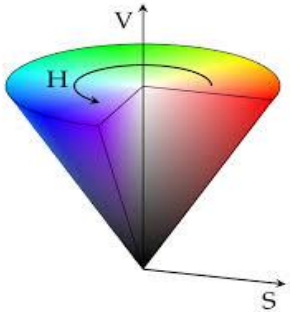
Ruang warna HSV disebut juga dengan HSI, yaitu dimana merupakan suatu ruang warna yang diformulasikan menurut apa yang diterima oleh mata manusia. HSV atau HSI merupakan singkatan dari *hue* (H), *saturation* (S), dan *value* (V) atau *intensity* (I). Sesuai dengan kepanjangannya, HSV terdiri atas 3 unsur pokok yaitu, *hue*, *saturation* dan *value*.

Hue merupakan suatu atribut atau sifat dari cahaya atau bisa juga dikatakan sifat dari permukaan yang memantulkan cahaya. Jadi dengan kata lain *hue* menyatakan warna yang sebenarnya, misalkan sebuah objek memiliki warna biru, itu artinya objek tersebut memantulkan *hue* biru. Selain itu, *hue* juga menggambarkan persepsi penglihatan manusia terhadap warna, seperti misalkan kehijauan (*greeness*), kemerahan (*redness*) dan lain sebagainya.

Berdasarkan kekuatannya *hue* dapat dikelompokkan menjadi *hue* lemah dan *hue* kuat. Tingkat kekuatan *hue* tersebut kemudian dideskripsikan dengan *saturation.* Sebagai contoh suatu warna dari sumber cahaya monokromatik menghasilkan warna dari satu *hue* saja, maka tingkat kekuatan *hue*nya akan sangat dipengaruhi oleh nilai saturasinya. Jadi dengan kata lain, saturasi merupakan komponen untuk mendeskripsikan tingkat kekuatan dari suatu warna. *Saturation* akan menentukan apakah

warna tersebut tergolong kuat (tua) atau lemah (muda), dimana semakin lemah tingkat kekuatannya menandakan warna tersebut semakin pudar dan mendekati warna putih.

Sedangkan *value* atau *intensity* digunakan untuk menentukan tingkat kecerahan atau *brightness* dari suatu warna. Atribut ini merepresentasikan banyak atau sedikitnya jumlah cahaya yang dipantulkan oleh suatu objek. Nilai atribut ini sangat mempengaruhi retina mata manusia dalam melihat warna, karena apabila dalam keadaan gelap, maka objek berwarna sekalipun tidak akan dapat terlihat. *Value* memiliki rentang nilai antara 0 sampai 100%, yaitu dimana apabila bernilai 0 maka warna tersebut akan menjadi hitam. Sebaliknya, jika semakin meningkat nilainya maka akan semakin cerah dan memunculkan variasi-variasi baru dari warna tersebut.

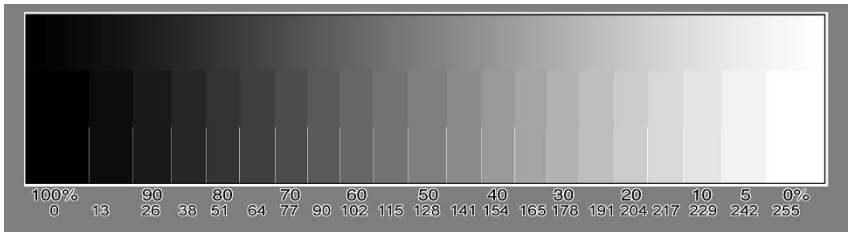


Gambar 2.2 Pemodelan ruang warna HSV
(https://pemrogramanmatlab.files.wordpress.com/2016/06/ruang-warna-citra-hsv.jpg)

3. *Grayscale*

Seperti namanya, citra *grayscale* hanya memiliki warna dengan tingkat keabuan. Penggunaan citra *grayscale* hanya membutuhkan lebih sedikit informasi dibandingkan dengan citra berwarna. Warna abu-abu pada citra *grayscale* merupakan warna RGB yang memiliki tingkat intensitas yang sama, sehingga hanya membutuhkan nilai intensitas atau atribut tunggal. Hal ini berbeda dengan citra berwarna yang setidaknya membutuhkan tiga intensitas atau atribut untuk setiap pikselnya. Nilai intensitas dari citra *grayscale* disimpan dalam 8 bit *integer*, atau dengan kata lain memiliki nilai mulai dari 0 sampai 255. Nilai 0 merepresentasikan

warna hitam, sedangkan 255 menggambarkan warna putih. Nilai intensitas diantara 0 sampai 255 merupakan derajat atau tingkat keabuan.



Gambar 2.3 Tingkat Derajat *Grayscale* 8 bit
(https://www.inksupply.com/images/html/21stepwide8bit.jpg)

2.2.4 *Thresholding*

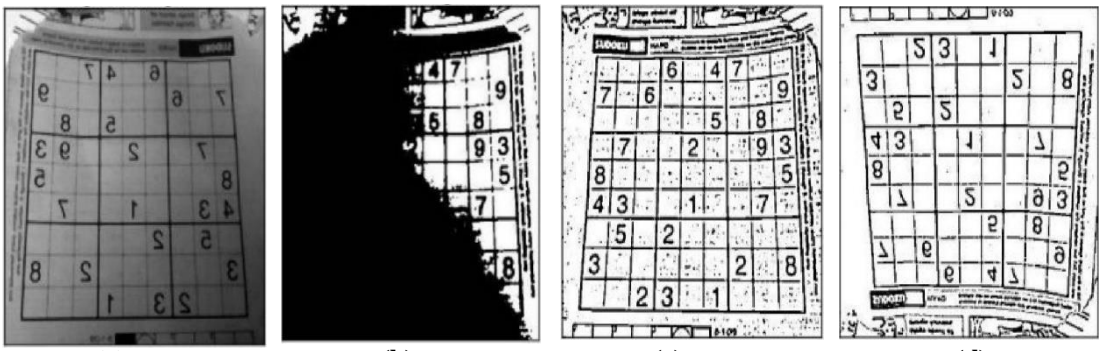
Salah satu metode dalam dunia pengolahan citra yang seringkali digunakan adalah teknik *thresholding.* Dimana *thresholding* merupakan salah satu teknik sederhana untuk memisahkan antara suatu objek yang menjadi target (objek *foreground*) dengan yang menjadi latar belakangnya (*background*). Pada umumnya metode ini menggunakan dua buah warna yaitu hitam dan putih (0 dan 1). Warna hitam akan merepresentasikan latar belakang, sedangkan warna putih digunakan untuk menggambarkan target atau objek *foreground.*

Secara sederhana *thresholding* dapat diaplikasikan dengan cara memberikan suatu nilai yang kemudian menjadi batasan nilai piksel pada suatu gambar *greyscale.* Sebagai contoh diberikan nilai 100, jadi apabila terdapat piksel yang memiliki nilai di bawah 100, maka nilai piksel tersebut akan langsung berubah menjadi 0 atau dengan kata lain berubah menjadi berwarna hitam. Sebaliknya, jika terdapat piksel dengan nilai lebih dari batas (lebih dari 100) maka nilai pikselnya akan berubah menjadi 1 atau 255, dengan kata lain berubah menjadi berwarna putih. Maka dengan demikian, warna citra secara keseluruhan akan berubah menjadi hitam dan putih. Dimana warna putih merepresentasikan objek target dan warna hitam menggambarkan *background* atau latar belakang dari objek target. Teknik semacam ini dinamakan dengan *fixed thresholding.*

Selain *fixed thresholding*, terdapat jenis lainnya, yaitu *adaptive thresholding.* Metode ini mampu mengubah suatu citra menjadi citra *threshold* secara dinamis bergantung pada tingkat pencahayaan dari tempat diambilnya citra.

Metode *adaptive thresholding* sangat efektif dalam menangani perubahan kecerahan pada citra akibat perubahan cahaya yang menyinari citra tersebut. Jadi dengan kata lain sangat cocok digunakan pada tempat-tampat yang memiliki tingkat pencahayaan yang tidak stabil.

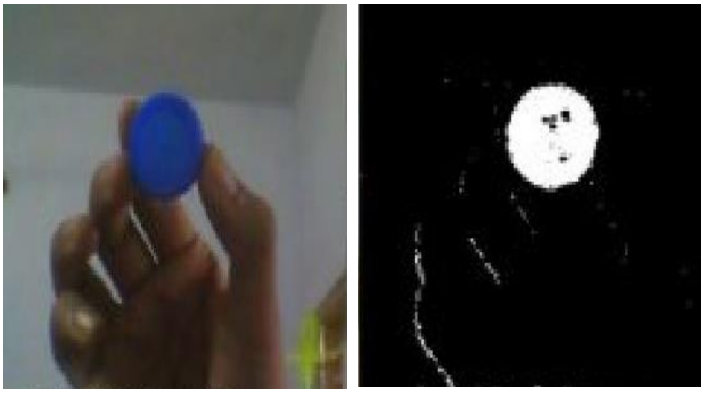
Secara khusus *adaptive thresholding* terbagi menjadi dua tipe, yaitu *Adaptive Mean Thresholding* dan *Adaptive Gaussian Thresholding.* Perbedaan diantara keduanya adalah pada penentuan nilai piksel gambar sebagai batasan nilai *threshold*nya. Pada *Adaptive Gaussian Thresholding* nilai batasan yang digunakan untuk *threshold* adalah jumlah nilai dari area target objek yang dimasukkan kedalam fungsi Gaussian. Sedangkan *Adaptive Mean Thresholding*, nilai batasan untuk *threshold* adalah nilai rata-rata dari nilai pixel suatu area di target objek.



(a) **(b)** **(c)** **(d)**

Gambar 2.4 (**a**) Gambar Asli, (**b**) Hasil Global T*hresholding* (v = 127), (**c**) Hasil *Adaptive Mean Thresholding*, dan (**d**) Hasil A*daptive* Gaussian Thresholding
(Mordvintsev, Alexander & Abid K, 2014)

Teknik *thresholding* tidak hanya dapat diterapkan pada citra *grayscale* saja, tetapi juga dapat digunakan pada citra dengan ruang warna HSV. Penerapan *thresholding* pada ruang warna HSV pada umumnya digunakan untuk menyeleksi warna tertentu yang diinginkan dan membuang warna lain selain warna yang ditargetkan tersebut. Teknik tersebut sering kali diaplikasikan dalam pendeteksian objek dengan berbasis warna. Caranya adalah dengan memberikan nilai batas bawah dan batas atas dari *range* warna yang ditargetkan. Di dalam *library* OpenCV perintah yang digunakan adalah “cv2.inRange”.



(a) **(b)**

Gambar 2.5 (**a**) Gambar Asli, (**b**) Hasil *Thresholding* atau *Masking* Warna (Mordvintsev, Alexander & Abid K, 2014)

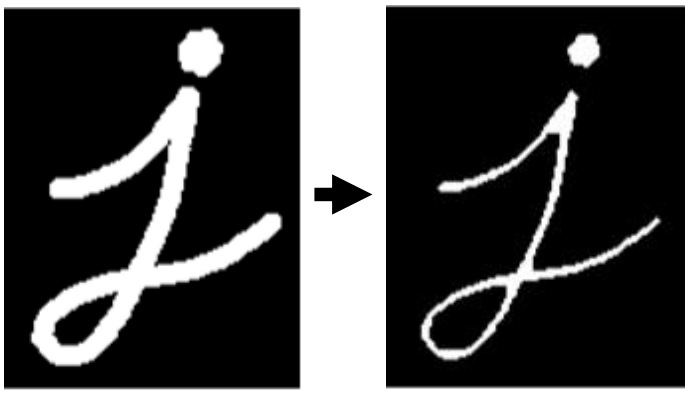
2.2.5 Transformasi Morfologi (*Morphological Transformations*)

Transformasi Morfologi merupakan salah satu jenis teknik sederhana dalam pengolahan citra yang mana dilakukan berdasarkan bentuk segmen citra. Tujuan dari teknik ini adalah untuk memperbaiki hasil segmentasi pada citra. Transformasi Morfologi biasanya dilakukan pada jenis citra biner, dan pada beberapa kasus teknik ini juga dapat digunakan pada citra *grayscale.* Gambar biner itu sendiri merupakan sebuah citra yang hanya terdiri dari dua nilai piksel, yaitu 0 dan 1 atau dengan kata lain sebuah citra hitam-putih. Dalam praktiknya, teknik erosi membutuhkan dua input, yaitu yang pertama adalah citra biner dan yang kedua adalah elemen penataan (*structuring element*) atau kernel, yang mana akan menentukan hasil dari operasi ini. Bentuk-bentuk operasi dalam Transformasi Morfologi, antara lain erosi, dilasi, *opening*, *closing*, dan lain sebagainya.

1. Erosi (Erosion)

   Ide dasar dari Erosi adalah seperti layaknya erosi atau pengikisan pada tanah. Operasi erosi akan mengikis batas-batas dari objek *foreground*. nilai piksel pada objek hanya akan dianggap 1 apabila semua piksel dibawah kernel bernilai 1, jika tidak maka akan tererosi (berubah menjadi 0). Jadi dengan kata lain, semua piksel yang berada pada tepi objek *foreground* akan dibuang, dimana jumlah piksel yang terserosi atau terbuang tergantung pada besarnya kernel yang digunakan. Erosi sangat berguna untuk menghilangkan *noise* putih disekitar objek *foreground*, teknik ini juga dapat

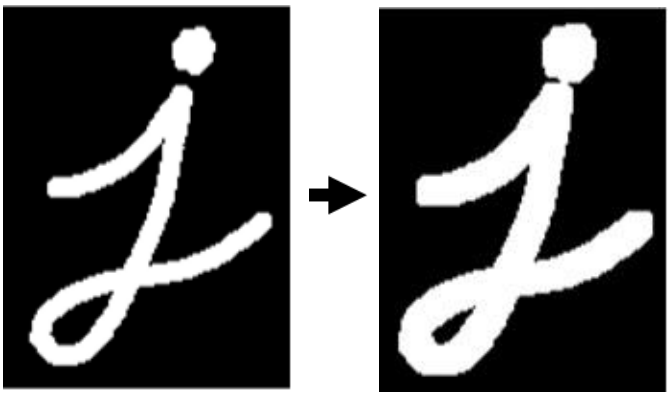
digunakan untuk memisahkan dua buah objek yang saling menempel, serta berbagai fungsi lainnya.



Gambar 2.6 Hasil Operasi Erosi (*Erosion*)
(Mordvintsev, Alexander & Abid K, 2014)

2. Dilasi (*Dilation*)

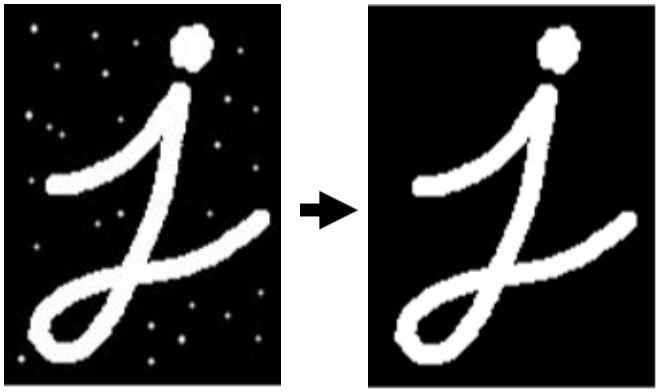
Dilasi merupakan kebalikan dari transformasi morfologi erosi, yaitu dimana ketika ada salah satu saja piksel bernilai 1 yang berada dibawah kernel, maka nilai piksel tepi objek *foreground* tersebut akan berubah menjadi bernilai 1. Jadi dengan demikian, transformasi dilasi akan meningkatkan wilayah warna putih pada citra, atau dengan kata lain ukuran objek *foreground* akan bertambah besar. Jadi dengan demikian, teknik transformasi dilasi ini sangat efektif untuk memperbesar ukuran suatu objek *foreground* yang ditargetkan.



Gambar 2.7 Hasil Operasi Dilasi (*Dilation*)
(Mordvintsev, Alexander & Abid K, 2014)

3. *Opening*

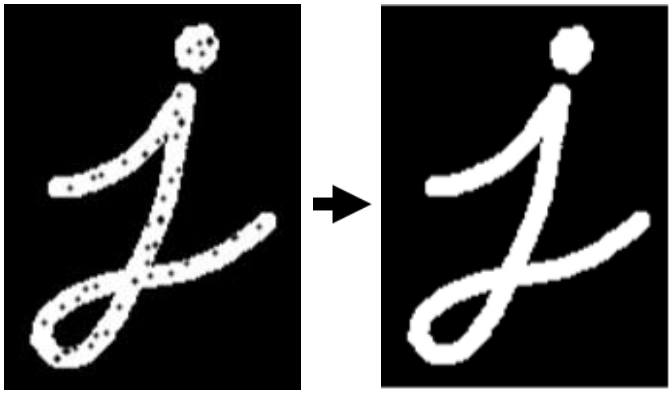
Opening adalah nama lain dari teknik transformasi erosi yang kemudian diikuti oleh dilasi. Teknik ini sangat berguna untuk menghilangkan *noise* yang berada disekitar objek. Erosi akan menghilangkan *noise* disekitar objek *foreground*, akan tetapi juga akan mengecilkan ukurannya, kemudian dengan diikuti oleh dilasi maka akan mempertebal ukurannya kembali. Sehingga dengan demikian, tingkat *noise* dapat ditekan tanpa harus mengorbankan luas wilayah warna atau ukuran dari objek *foreground.*



Gambar 2.8 Hasil Operasi *Opening*
(Mordvintsev, Alexander & Abid K, 2014)

4. Closing

Closing merupakan kebalikan dari *opening*, yaitu teknik dilasi yang kemudian diikuti oleh erosi. Teknik ini sangat berguna untuk menutupi lubang-lubang atau titik hitam yang terdapat di dalam objek *foreground*. Dilasi akan mempertebal atau memperluas wilayah warna dari objek *foreground*, sehingga *noise* atau titik-titik hitam di dalamnya dapat tertutupi oleh warna asli objek tersebut. Setelah *noise* berhasil tertutupi, selanjutnya teknik erosi akan mengembalikan ukuran keseluruhan objek *foreground* menjadi seperti semula. Jadi dengan demikian, teknik ini sangat efektif digunakan untuk menutupi *noise* warna lain yang terdapat di dalam warna asli dari objek *foreground.*



Gambar 2.9 Hasil Operasi *Closing*
(Mordvintsev, Alexander & Abid K, 2014)

2.2.6 Filter 2D *Convolution*

Sebagai sebuah sinyal 1 dimensi (1D), sebuah citra juga dapat dikenakan berbagai macam jenis filter, baik itu *low-pass* filter (LPF), maupun *high-pass* filter (HPF). Kedua jenis filter tersebut memiliki perbedaan dalam hal fungsi atau kegunaannya, *low-pass* filter akan dapat membantu menghilangkan *noises*, membuat gambar menjadi *blur* atau tersamarkan, dan lain sebagainya. *High-pass filter* berfungsi untuk membantu menemukan bagian tepi atau *edges* pada suatu citra, mempertajam citra, dan lain sebagainya.

Pada penelitian tugas akhir ini, filter 2D *Convolution* akan digunakan sebagai *low-pass* filter yang mana akan berguna untuk mengurangi tingkat *noise* dan membuat citra menjadi tersamarkan. Pada dasarnya operasi filter 2D *Convolution* ini membutuhkan 2 buah komponen utama, yaitu sebuah citra yang akan menjadi input dan sebuah kernel yang akan dibelitkan kedalam citra input. Berikut adalah langkah-langkah dalam menjalankan operasi filter 2D *Convolution,*

1. Menentukan atau menyeleksi sebuah koordinat (*x*, *y*) dari citra inputan.
2. Meletakkan titik tengah dari kernel pada koordinat citra yang telah ditentukan pada langkah pertama.
3. Melakukan operasi perkalian antara kernel dengan nilai pada citra inputan, kemudian menjumlahkannya untuk mendapatkan hasil akhir. Berikut adalah contoh operasi perhitungan filter 2D *Convolution* pada suatu bagian citra dengan menggunakan kernel berukuran matriks 3 x 3.

$$Output = \frac{1}{9} \times \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \times \begin{bmatrix} 197 & 50 & 213 \\ 3 & 181 & 203 \\ 231 & 2 & 93 \end{bmatrix}$$

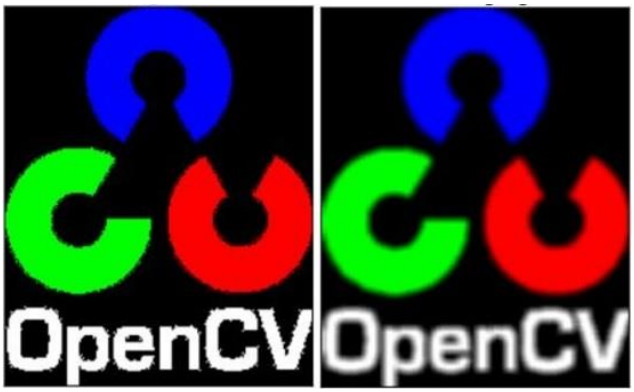
$$= \begin{bmatrix} \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \end{bmatrix} \times \begin{bmatrix} 197 & 50 & 213 \\ 3 & 181 & 203 \\ 231 & 2 & 93 \end{bmatrix}$$

$$= \Sigma \begin{bmatrix} 21 & 5 & 23 \\ 0 & 20 & 22 \\ 25 & 0 & 10 \end{bmatrix}$$

$$= 126$$

4. Selanjutnya, nilai koordinat yang telah ditentukan pada citra inputan digantikan dengan nilai hasil dari perhitungan konvolusi.

Seluruh proses tersebut akan terus menerus dilakukan pada semua koordinat citra inputan, sehingga nilai piksel pada keseluruhan citra akan berubah. Perubahan itulah yang menjadikan citra menjadi tersamarkan dan akan mengurangi tingkat *noise* pada citra tersebut. Pada *library* OpenCV, perintah filter ini dapat diakses dengan mudah menggunakan perintah, “cv2.filter2D”.

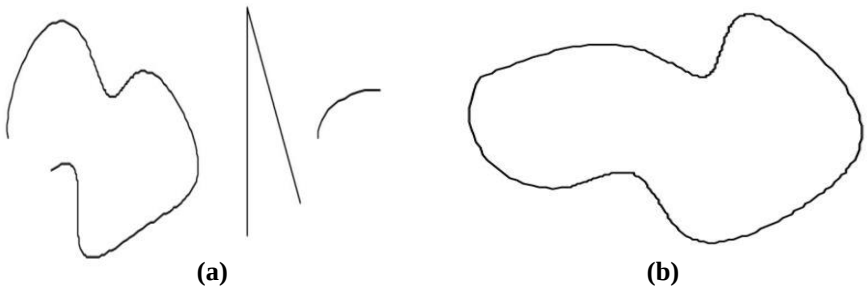


Gambar 2.10 Hasil Operasi Filter 2D *Convolution*
(Mordvintsev, Alexander & Abid K, 2014)

2.2.7 Kontur (Contours)

Kontur merupakan keadaan yang ditimbulkan oleh perubahan intensitas pada piksel-piksel yang bertetangga atau berdekatan satu sama lain. Perubahan intensitas itulah yang menyebabkan tepi (*edges*) objek pada citra dapat terdeteksi.

Jadi secara sederhana kontur dapat diartikan sebagai rangkaian piksel yang membentuk batas daerah (*region boundary*). Pada dasarnya terdapat dua jenis kontur, yaitu kontur terbuka dan kontur tertutup. Kontur tertutup berkoresponden dengan batas yang mengelilingi suatu daerah objek, atau dengan kata lain membentuk suatu sirkuit tertutup. Piksel-piksel di dalam daerah objek tersebut dapat ditemukan dengan menggunakan algoritma pengisian (*filling algorithm*). Batas-batas yang mengelilingi suatu daerah objek dapat berguna untuk mendeskripsikan bentuk dari suatu objek, sehingga dapat digunakan untuk mengenali atau mengidentifikasi suatu objek. Berbeda dengan jenis kontur tertutup yang membentuk suatu sirkuit tertutup, tipe kontur terbuka hanya berupa fragmen garis atau bagian dari batas daerah yang tidak membentuk sirkuit, dengan kata lain tidak melingkupi seluruh daerah objek.



(a) **(b)**

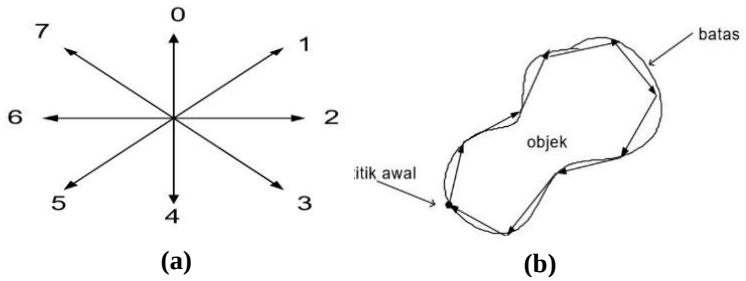
Gambar 2.11 (**a**) Kontur Terbuka, (**b**) Kontur Tertutup
(Munir, Reynaldi, 2004)

Dalam praktiknya, representasi kontur dapat berupa daftar notasi tepi (*edge list*) atau berupa kurva.

1. Daftar notasi tepi atau *edge list*

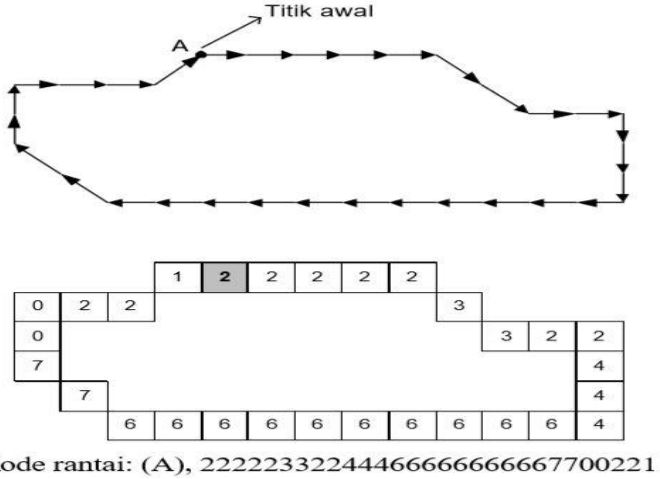
   Edge list merupakan himpunan berurutan piksel-piksel tepi yang menjadi batas daerah suatu objek. Teknik yang sering kali digunakan untuk merepresentasikan kontur dalam bentuk *edge list* adalah dengan menggunakan kode rantai.

Kode rantai atau *chain code* merupakan notasi yang digunakan untuk mengkodekan *edge list* yang membentuk batas daerah suatu objek. Kode rantai menspesifikasikan arah setiap piksel-piksel tepi di dalam *edge list.* Arah yang digunakan adalah 8 arah mata angin.



Gambar 2.12 (**a**) Arah Kode Rantai, (**b**) Representasi Batas Objek dalam Kode Rantai (Munir, Reynaldi, 2004)

Pengkodean dimulai dari sebuah piksel tepi dan berjalan searah jarum jam. Arah setiap piksel tepi yang membentuk batas objek dikodekan dengan salah satu dari delapan kode rantai. Kode rantai merepresentasikan batas objek dengan koordinat piksel tepi pertama lalu diikuti dengan daftar kode rantai berikutnya. Karena terdapat 8 arah, maka cukup 3 bit untuk mengkodekan setiap arah. Gambar berikut adalah contoh pengkodean batas objek dengan menggunakan kode rantai.



Gambar 2.13 Contoh Hasil Pengkodean Kode Rantai pada Tepi Objek (Munir, Reynaldi, 2004)

2. Kurva

Representasi kontur dalam bentuk kurva dianggap sebagai bentuk penggambaran yang paling efektif dan efisien untuk menganalisis sebuah citra. Misalnya, rangkaian piksel tepi yang membentuk garis dapat direpresentasikan hanya dengan sebuah persamaan garis lurus. Representasi semacam ini menyederhanakan perhitungan selanjutnya, yaitu seperti arah dan panjang garis. Metode yang sering digunakan untuk merepresentasikan kontur dalam bentuk kurva adalah dengan teknik pencocokan kurva atau *curve fiting.*

Terdapat dua macam teknik pencocokan kurva, yaitu interpolasi dan penghampiran (approximation). Interpolasi kurva adalah metode dimana mencari kurva yang melalui semua piksel tepi, sedangkan penghampiran kurva merupakan metode dimana mencari kurva yang paling dekat melalui piksel-piksel tepi, tetapi tidak perlu melalui semua piksel-piksel tersebut.

Dalam *library* OpenCV terdapat berbagai macam fitur-fitur yang berkaitan dengan kontur, yaitu antara lain,

1. Moments

Dengan mengetahui momentum dari suatu citra maka akan banyak sekali memberikan keuntungan, yaitu diantaranya adalah memungkinkan untuk dapat mengetahui pusat massa dari objek target, luas daerah objek, dan lain sebagainya. Di dalam *library* OpenCV perintah yang digunakan adalah “cv2.moments()”.

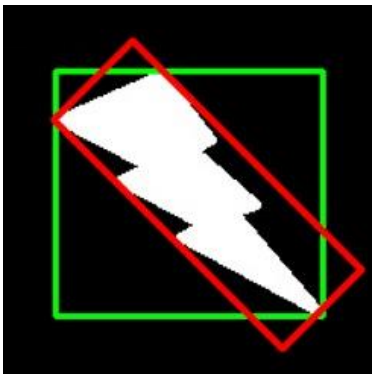
2. *Contour Area*

Selain dengan memanfaatkan fitur momentum, luas wilayah target objek juga bisa didapatkan dengan perintah “cv2.contourArea()”

3. Bounding Rectangle

Fitur *Bounding Rectangle* merupakan fitur yang memungkinkan untuk melingkupi area kontur objek didalam sebuah persegi. Terdapat dua jenis *Bounding Rectangle*, yaitu *Straight Bounding Rectangle* dan *Rotated Bounding Rectangle.* Di dalam *library* OpenCV perintah *Straight Bounding*

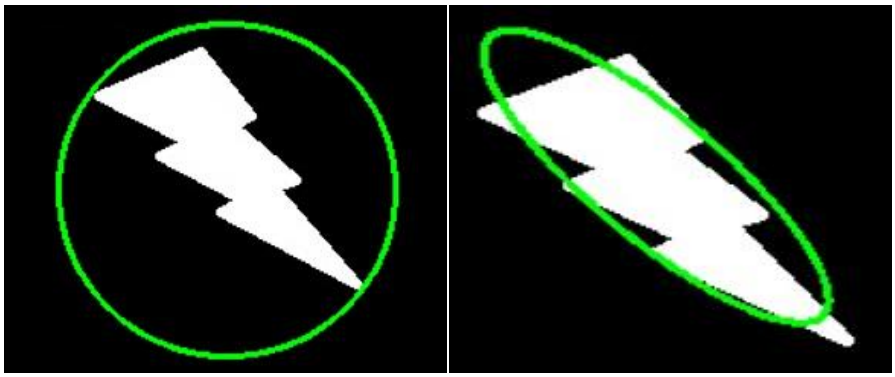
Rectangle yang digunakan adalah "cv2.boundingRect()". Sedangkan perintah untuk *Rotated Bounding Rectangle* adalah "cv2.minAreaRect()" kemudian diikuti dengan "cv2.boxPoints()".



Gambar 2.14 Kotak Warna Hijau adalah *Straight Bounding Rect*, dan Kotak Warna Merah adalah *Rotated Bounding Rect*
(Mordvintsev, Alexander & Abid K, 2014)

4. Minimum Enclosing Circle & Fitting an Ellipse

Fitur *Minimum Enclosing Circle* adalah fitur yang memungkinkan untuk melingkupi area kontur suatu objek didalam sebuah lingkaran. Perintah yang digunakan untuk memanggil fitur ini adalah "cv2.minEnclosingCircle()". Sedangkan fitur *Fitting an Ellipse* seperti halnya pada fitur *Rotated Bounding Rect*, tetapi dalam bentuk *ellipse.* Perintah untuk menggunakan fitur ini adalah "cv2.ellipse()".



(a) **(b)**

Gambar 2.15 (**a**) Hasil *Minimum Enclosing Circle*, (**b**) *Fitting an Ellipse*
(Mordvintsev, Alexander & Abid K, 2014)

5. *Extrem Points*

Extrem points atau titik-titik ekstrim yang dimaksud pada kontur suatu objek terdeteksi adalah titik bagian paling atas, paling bawah, paling kanan dan paling kiri dari kontur objek tersebut. Perintah yang digunakan di dalam OpenCV untuk dapat menemukan titik-titik ekstrim tersebut adalah sebagai berikut,

b. PalingKiri = tuple(cnt[cnt[:,:,0].argmin()][0])
c. PalingKanan = tuple(cnt[cnt[:,:,0].argmax()][0])
d. PalingAtas = tuple(cnt[cnt[:,:,1].argmin()][0])
e. PalingBawah = tuple(cnt[cnt[:,:,1].argmax()][0])

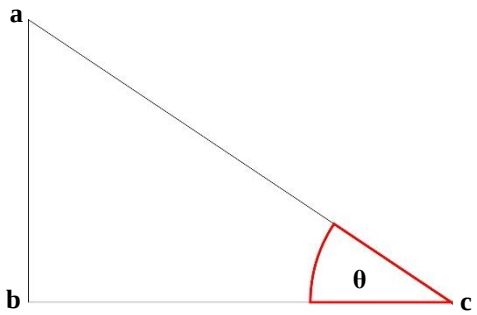
Gambar 2.16 Titik-Titik Ekstrim Kontur Objek
(Mordvintsev, Alexander & Abid K, 2014)

6. *Point Polygon Test*

Fungsi *Point Polygon Test* memungkinkan untuk dapat menemukan jarak terpendek dari sebuah titik menuju sebuah kontur suatu objek terdeteksi. Hasil dari fungsi ini akan bernilai positif ketika titik tersebut berada di dalam kontur suatu objek terdeteksi, sedangkan jika titik tersebut berada tepat pada garis kontur, maka akan bernilai nol (0). Sebaliknya, ketika titik tersebut berada di luar kontur, maka hasil dari fungsi ini akan bernilai negatif. Perintah di dalam OpenCV yang digunakan untuk memanggil fungsi ini adalah “cv2.pointpolygontest”.

2.2.8 Trigonometri Segitiga Siku-Siku

Secara sederhana, trigonometri merupakan salah satu cabang disiplin dalam ilmu matematika yang mana mempelajari tentang hubungan antara sisi dan sudut. Kata "trigonometri" berasal dari bahasa Yunani, yaitu "trigonon" yang berarti segitiga atau tiga sudut, dan "metron" yang berarti ukuran. Sedangkan segitiga siku-siku merupakan salah satu jenis segitiga dimana salah satu sisinya memiliki sudut sebesar 90°.



Gambar 2.17 Segitiga Siku-Siku

Tampak terlihat pada gambar di atas, bahwa **ab** merupakan sisi depan yang mana berhadapan langsung dengan sudut θ (theta). Sisi **ac** adalah sisi miring terhadap sudut **θ**. Sedangkan sisi **bc** merupakan sisi samping terhadap sudut **θ**. Berdasarkan konsep tersebut, maka dapat dirumuskan sebagaimana berikut,

1. $Sin\ \theta = \frac{sisi\ depan}{sisi\ miring}$ **(2.1)**
2. $Cos\ \theta = \frac{sisi\ samping}{sisi\ miring}$ **(2.2)**
3. $Tan\ \theta = \frac{sisi\ depan}{sisi\ samping}$ **(2.3)**

Hasil dari rumus diatas masih dalam bentuk satuan radian, sehingga untuk mengetahui seberapa besar sudut θ (theta) dalam satuan derajat, maka perlu dikonversikan terlebih dahulu, yaitu dengan cara,

$$derajat° = radian \times \frac{180}{\pi} \quad \textbf{(2.4)}$$

Dimana:

1. $\pi = 3{,}14$

2.2.9 Komunikasi Serial

Komunikasi serial merupakan jenis komunikasi yang melakukan pengiriman data per-bit secara bergantian dan berurutan. Proses pengiriman data tersebut hanya akan menggunakan satu jalur, sehingga akan lebih hemat daripada tipe komunikasi paralel yang menggunakan lebih dari satu jalur dalam proses pengiriman datanya. Meskipun demikian, dari segi kecepatan pengiriman komunikasi serial lebih lambat dibandingkan dengan komunikasi paralel.

Pada dasarnya terdapat dua jenis komunikasi serial, yaitu *synchronous* dan *asynchronous* serial. Pada tipe *synchronous* serial, antara pengirim dan penerima data harus menyelaraskan atau menyamakan nilai *clock* diantara keduanya. Disisi lain, tipe komunikasi *asynchronous* serial tidak membutuhkan proses penyelarasan *clock* antara pengirim dan penerima data, akan tetapi sebagai gantinya pihak pengirim akan memberikan sinyal sinkronisasi terlebih dahulu kepada penerima sebelum proses komunikasi dilaksanakan.

Pada setiap board arduino paling tidak dibekali dengan satu buah *port* serial, hal tersebut bertujuan agar arduino dapat berkomunikasi dengan komputer maupun dengan *device*s lainnya. Pada dasarnya komunikasi serial antara *board* arduino dan komputer sangatlah sederhana, yaitu dengan cara menggunakan kabel USB *type* A (komputer) to *type* B (arduino). Pada kabel tersebut, *port* TX arduino akan terhubung dengan RX pada komputer, dan sebaliknya *port* RX arduino akan terhubung dengan TX pada komputer. Komunikasi tersebut menggunakan dua jalur yang berbeda, itu artinya komunikasi antara komputer dengan arduino dapat berjalan secara *full duplex*, atau dengan kata lain dapat saling mengirimkan data secara bersamaan. Meskipun demikian, tetap saja proses antara mengirim dan menerima data pada *board* arduino tetap akan berjalan secara bergantian.

Gambar 2.18 USB *Type* A to *Type* B