

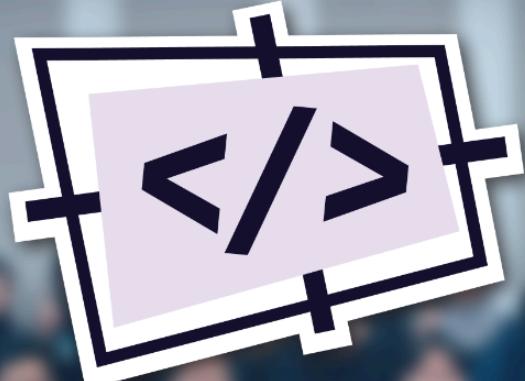


IRIS

INTERNSHIP  
MODULE

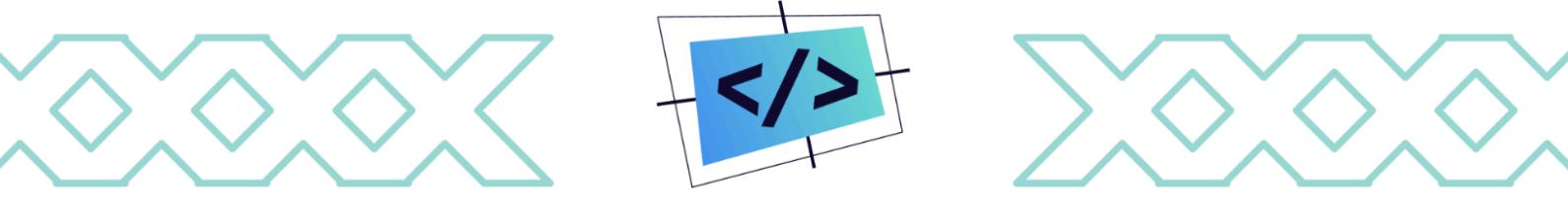
2<sup>nd</sup>  
WEEK

PROGRAMMING



# Computer Vision and OpenCV

<b>Pengantar Gambar Digital dan OpenCV.....</b>	<b>1</b>
1. OpenCV.....	1
2. Komponen Dalam Gambar.....	1
3. Video.....	3
<b>Pengolahan Citra.....</b>	<b>4</b>
1. Akses Gambar, Video, dan Kamera dengan OpenCV.....	4
2. Range of Interest (ROI).....	5
3. Image Segmentation.....	6
4. Bird Eye View.....	9



## Pengantar Gambar Digital dan OpenCV

### 1. OpenCV

OpenCV (Open Source Computer Vision Library) adalah sebuah library pemrograman Open Source yang berisi kumpulan fungsi dan algoritma untuk pengolahan citra (*image processing*) dan visi komputer (*computer vision*) secara *real-time*. Sederhananya OpenCV adalah library yang memudahkan kita untuk memanipulasi citra video maupun gambar. OpenCV juga tersedia dalam banyak bahasa pemrograman contohnya C++, dan Python. Dokumentasi untuk openCV dapat dilihat pada laman resmi mereka: <https://docs.opencv.org/4.12.0/index.html>

### 2. Komponen Dalam Gambar

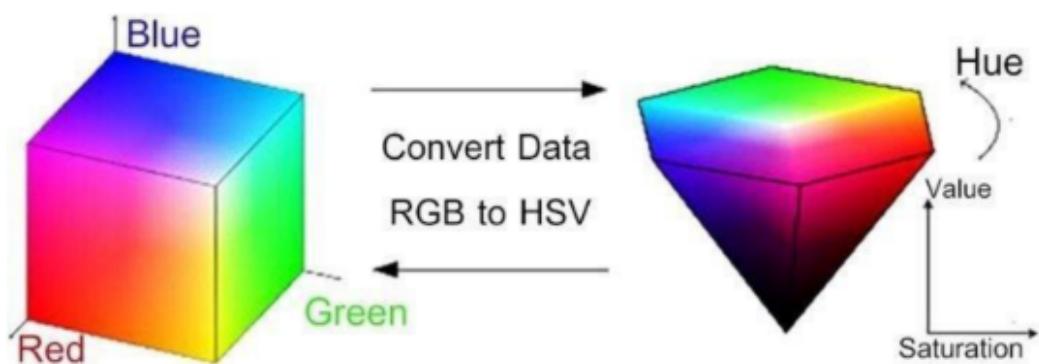
Dalam dunia komputer, gambar digital disimpan sebagai matriks dua dimensi (2D) yang terdiri dari elemen-elemen yang disebut pixel. Setiap pixel ini mewakili titik kecil pada gambar yang memiliki nilai intensitas warna atau tingkat keabuan (untuk gambar grayscale). Gambar berwarna, yang umum kita lihat, terdiri dari beberapa channel warna. Di OpenCV, gambar disimpan dalam objek yang disebut **cv::Mat**. Mat ini adalah sebuah struktur data yang berfungsi untuk menyimpan nilai-nilai pixel dalam format matriks.

Tiap pixel dalam gambar terdiri dari 1 atau lebih nilai tergantung pada jenis gambar:

- Grayscale: Hanya satu nilai yang menyatakan intensitas (0 untuk hitam, 255 untuk putih).
- Berwarna (RGB/BGR): Setiap pixel memiliki tiga nilai yang masing-masing mewakili warna Red, Green, dan Blue atau Blue, Green, dan Red dalam konteks OpenCV. Selain RGB juga terdapat banyak channel warna lain diantaranya ada RGB, HSV, HSL, YUV Contoh sederhananya, sebuah pixel dalam gambar berwarna. BGR mungkin terlihat seperti ini:
  - (255, 0, 0) untuk pixel berwarna biru.
  - (0, 255, 0) untuk pixel berwarna hijau.
  - (0, 0, 255) untuk pixel berwarna merah.

# IRIS TEAM

## ITS ROBOTICS



# IRIS TEAM

## ITS ROBOTICS

### 3. Video

Video adalah rangkaian gambar yang ditampilkan secara berurutan pada kecepatan tertentu. Setiap gambar dalam video disebut frame. Biasanya, video ditampilkan pada kecepatan 24 hingga 60 frame per detik (fps), yang membuat mata manusia melihatnya sebagai gerakan yang mulus. Di OpenCV, kita dapat mengakses video frame-by-frame dan memperlakukan setiap frame sebagai gambar individual (matriks 2D). Ini berarti apa yang berlaku pada gambar statis juga berlaku pada setiap frame dalam video.

Contoh Teori: Misalnya, video dengan resolusi 1920x1080 yang berjalan pada 30 fps berisi 30 gambar berukuran 1920x1080 yang ditampilkan setiap detiknya. Kita bisa memanipulasi setiap frame ini sama seperti kita memanipulasi gambar statis.

## Pengolahan Citra

### 1. Akses Gambar, Video, dan Kamera dengan OpenCV

Langkah awal jika kita ingin mengolah citra adalah bagaimana cara kita untuk dapat mengakses citra tersebut menggunakan bahasa pemrograman. Dalam OpenCV hal itu dapat kita lakukan seperti berikut:

- Gambar

```
cv::Mat image = cv::imread("{path to image}");
cv::imshow("Gambar", image);
cv::waitKey(0);
```

- Video

```
cv::VideoCapture cap("{path to video}");
cv::Mat frame;

while (cap.isOpened()){
    cap >> frame;

    if (frame.empty()){
        break
    }

    cv::imshow("video", frame)

    if (cv::waitKey(25) == 'q'){
        break
    }
}
```

- kamera

```
cv::VideoCapture cap(0);
cv::Mat frame;
```

```
while (cap.isOpened()){

    cap >> frame;

    if (frame.empty()){

        break

    }

    cv::imshow("video", frame)

    if (cv::waitKey(25) == 'q'){

        break

    }

}
```

## **2. Range of Interest (ROI)**

Terkadang untuk memproses semua frame membutuhkan komputasi yang lebih berat dengan ROI kita dapat memproses hanya bagian frame yang kita inginkan. Misalnya kita hanya ingin mendekripsi frame bagian tengah saja. Maka kita dapat membuat ROI berbentuk persegi pada bagian tengah frame.

```
cv::Mat image = cv::imread("{path to image}");

cv::Rect ROI(100, 150, 250, 200);
cv::Mat roi_image = image(ROI)

cv::imshow("Gambar", image);
cv::imshow("Region of Interest", roi_image);

cv::waitKey(0);
```

# IRIS TEAM

## ITS ROBOTICS



### 3. Image Segmentation

Konsep segmentasi gambar ini cukup simple yaitu memisahkan atau menyeleksi objek yang diinginkan. Saat ini banyak cara segmentasi gambar baik dengan machine learning ataupun seleksi warna. Pada modul ini segmentasi berdasarkan warna.

Hal pertama yang kita lakukan untuk segmentasi adalah mengubah channel warna gambar kita ke channel warna lain contohnya dari BGR ke HLS agar kita lebih mudah untuk melakukan segmentasi. Dengan OpenCV kita dapat melakukannya dengan perintah berikut

```
cv::Mat hlsImage;
cv::cvtColor(image, hlsImage, cv::COLOR_BGR2GRAY);
cv::imshow("HLS Image", hlsImage);
```

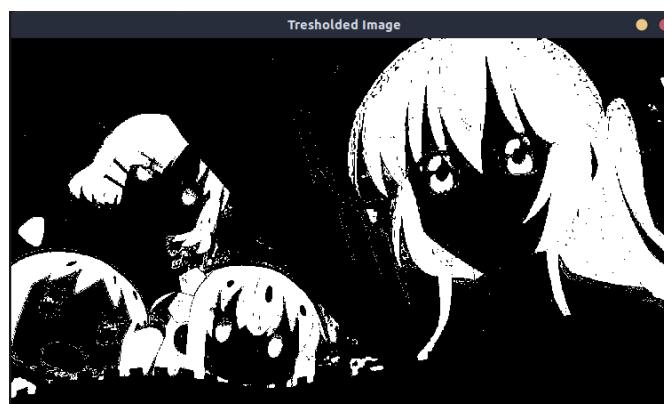


# IRIS TEAM

## ITS ROBOTICS

Setelah itu kita dapat melakukan segmentasi dengan metode thresholding. Seperti yang kita pelajari sebelumnya bahwa pixel dalam gambar memiliki nilai yang mewakili warna mereka. Dalam kasus image kita yaitu HLS, ada 3 nilai yang merepresentasikan Hue (0-255), Lightness (0-255), dan Saturation (0-255). Metode Thresholding akan mengubah nilai warna tersebut sesuai dengan batas yang kita atur. Misal kita ingin nilai minimumnya adalah 100 dan maksimumnya adalah 160, maka nilai-nilai yang termasuk kedalam batas tersebut akan diubah menjadi 255 (putih) dan yang di luar akan diubah menjadi 0 (hitam).

```
cv::Mat thresholdedImage;
cv::inRange(hlsImage, cv::Scalar(11, 42, 61), cv::Scalar(255, 211,
255), thresholdedImage);
cv::imshow("Tresholded Image", thresholdedImage);
```



setelah thresholding kita akan lakukan edge detection dengan mendeteksi kontur. Kontur sendiri sederhananya adalah kelompok piksel-piksel putih yang saling berdekatan.

```
std::vector<std::vector<cv::Point>> contours;
cv::findContours(thresholdedImage, contours, cv::RETR_EXTERNAL,
```

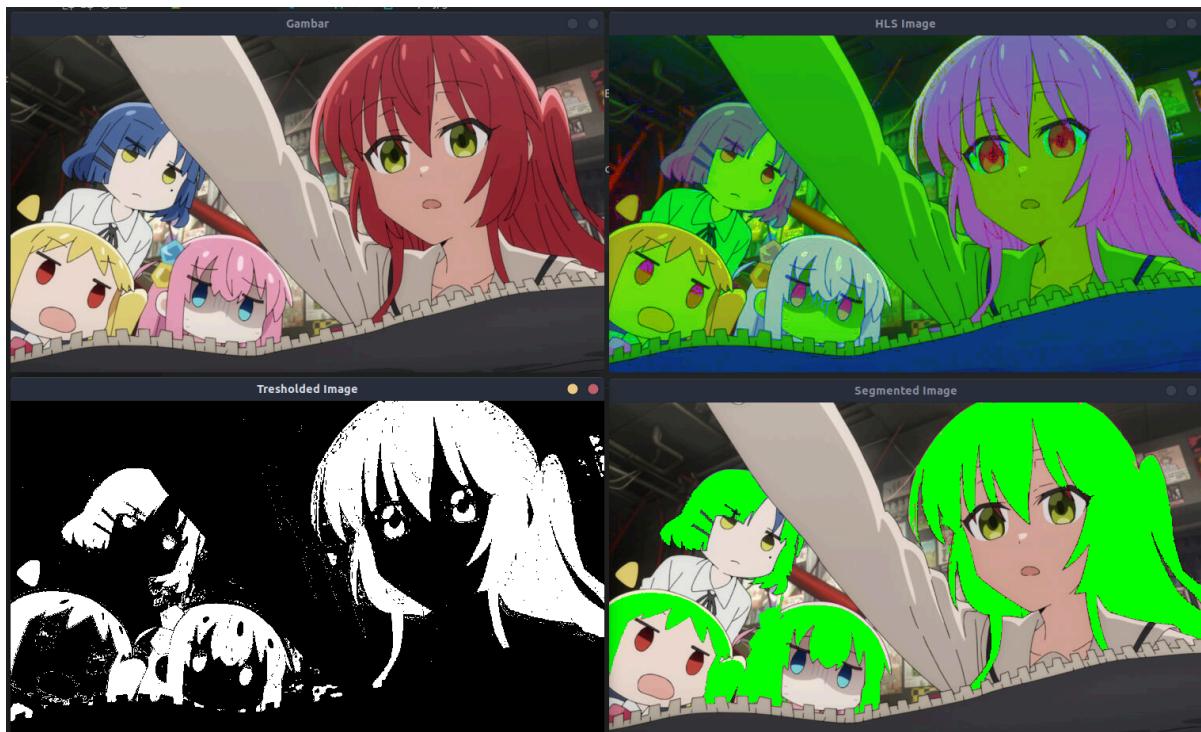
# IRIS TEAM

## ITS ROBOTICS

```
cv::CHAIN_APPROX_SIMPLE);
```

Dari data kontur tersebut kita dapat menggambarnya pada frame agar lebih jelas.

```
cv::Mat segment = image.clone();
for (size_t i = 0; i < contours.size(); i++) {
    if (cv::contourArea(contours[i]) > 1000) {
        cv::drawContours(segment, contours, (int)(i),
cv::Scalar(0, 255, 0), cv::FILLED);
    }
}
cv::imshow("Segmented Image", segment);
```



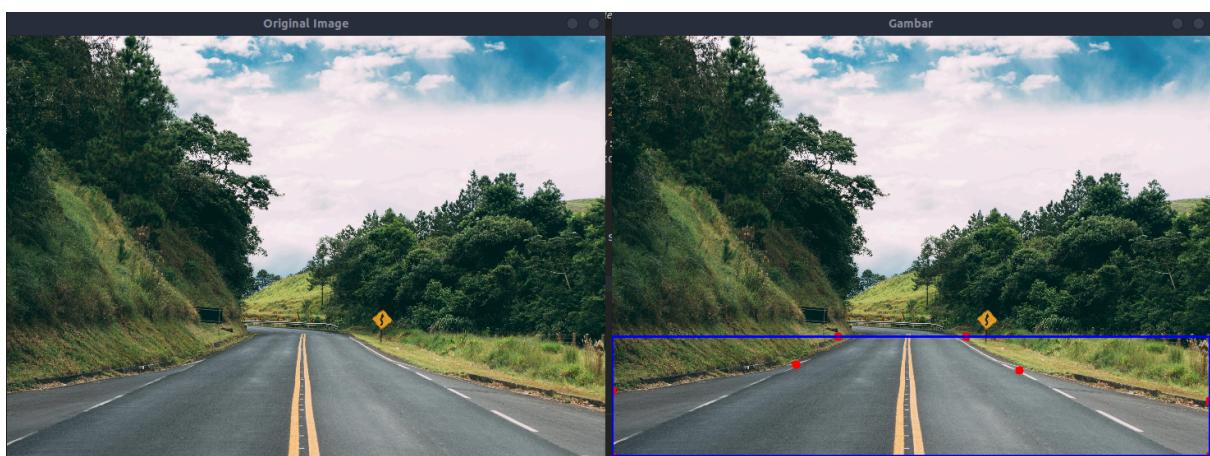
Dengan begini kita telah berhasil dalam mensegmentasi rambut karakter Bochi the Rock

### 4. Bird Eye View

Konsep *bird's eye view* dalam pengolahan citra dan visi komputer merujuk pada suatu teknik di mana sebuah gambar yang awalnya diambil dari perspektif depan atau samping diubah sedemikian rupa sehingga seolah-olah dilihat dari atas, seperti pandangan seekor burung. Transformasi ini sering disebut sebagai *perspective transformation* atau *inverse perspective mapping* (IPM).

Tujuan utama dari penerapan *bird's eye view* adalah untuk menghilangkan distorsi perspektif yang melekat pada gambar standar, sehingga objek dan fitur yang ada di dalamnya dapat diukur dan dianalisis dengan lebih akurat. Misalnya, dalam sistem bantuan pengemudi tingkat lanjut (ADAS) atau mobil otonom.

dengan OpenCV kita dapat melakukan transformasi perspektif pula. Untuk melakukannya pertama-tama kita harus mempersiapkan titik-titik yang ingin kita ubah perspektifnya contohnya seperti berikut



Untuk mendapatkan titik-titik tersebut kalian bisa melalui beberapa proses seperti metode segmentasi yang telah dijelaskan tadi, menggunakan fungsi erode dan dilate untuk mereduksi noise pada threshold, atau sebagainya sesuai kebutuhan.

Setelah kalian mendapatkan titik awal, selanjutnya kalian menentukan dimana titik tersebut harus diplot. Untuk mendapatkan bird eye maka titik harus diplot membentuk persegi. Setelah itu kalian bisa melakukan transformasi perspektif pada OpenCV dengan perintah berikut

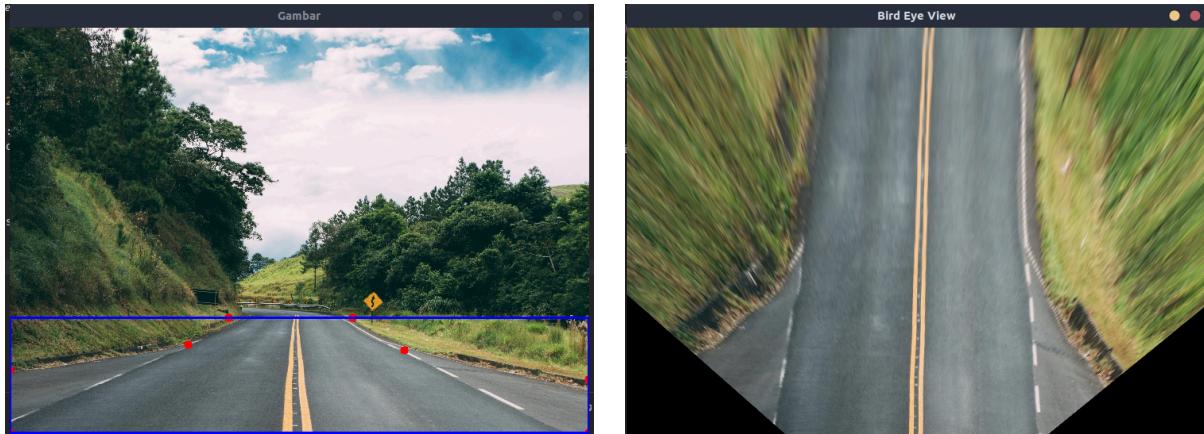
```
// Membuat matriks transformasi perspektif dan menerapkannya ke  
gambar asli  
cv::Mat BEV_plot;  
cv::Mat BEV_view;
```

# IRIS TEAM

## ITS ROBOTICS

```
BEV_plot = cv::getPerspectiveTransform(point_awal, point_tujuan);
cv::warpPerspective(image, BEV_view, BEV_plot, BEV_view.size());
```

maka gambar hasil pemrosesan akan terlihat seperti berikut



jika ingin mencoba berikut merupakan contoh kode untuk memproses gambar jalan raya tersebut.

link gambar: <https://share.google/images/OwSu884DReTRNWjYH>

```
#include <opencv2/opencv.hpp>
#include <iostream>

int main(){
    // Membaca gambar road.png dari folder parent
    cv::Mat image = cv::imread("../road.png");

    // Mengecek apakah gambar berhasil dibuka
    if(image.empty()){
        std::cerr << "Could not open or find the image!" << std::endl;
        return -1;
    }

    // Resize gambar menjadi 1/5 ukuran aslinya
    cv::resize(image, image, cv::Size((int)(image.size().width/5), (int)(image.size().height/5)));

    // Mendefinisikan area ROI (Region of Interest) pada bagian bawah gambar
    cv::Rect ROI_area(0, (int)(image.rows/1.4), image.cols, (int)(image.rows/3.5));
    cv::Mat ROI_thresholded;

    // Konversi ROI ke format HLS dan threshold untuk mendeteksi warna tertentu (misal jalan)
    cv::cvtColor(image(ROI_area), ROI_thresholded, cv::COLOR_BGR2HLS);
    cv::inRange(ROI_thresholded, cv::Scalar(26, 0, 0), cv::Scalar(255, 166, 38), ROI_thresholded);

    // Proses morfologi untuk membersihkan hasil threshold (erode dan dilate)
    std::vector<std::vector<cv::Point>> contours;
    cv::erode(ROI_thresholded, ROI_thresholded, cv::Mat(), cv::Point(-1, -1), 2);
    cv::dilate(ROI_thresholded, ROI_thresholded, cv::Mat(), cv::Point(-1, -1), 8);
    cv::erode(ROI_thresholded, ROI_thresholded, cv::Mat(), cv::Point(-1, -1), 5);
```

# IRIS TEAM

## ITS ROBOTICS

```
cv::dilate(ROI_thresholded, ROI_thresholded, cv::Mat(), cv::Point(-1, -1), 8);
cv::erode(ROI_thresholded, ROI_thresholded, cv::Mat(), cv::Point(-1, -1), 7);

// menerapkan hasil threshold area ROI ke thresholded image yang berukuran sama dengan gambar asli
cv::Mat image_thresholded = cv::Mat::zeros(image.size(), CV_8UC1);
ROI_thresholded.copyTo(image_thresholded(ROI_area));

// Mencari kontur pada hasil threshold
cv::findContours(image_thresholded, contours, cv::RETR_EXTERNAL, cv::CHAIN_APPROX_SIMPLE);

// Mengambil titik-titik sudut dari kontur terbesar (area > 1000)
std::vector<cv::Point> titik_sudut;
for (size_t i = 0; i < contours.size(); i++) {
    if (cv::contourArea(contours[i]) > 1000) {
        cv::approxPolyDP(contours[i], titik_sudut, 0.01 * cv::arcLength(contours[i], true), true);
        break;
    }
}

// Menyalin gambar asli untuk menampilkan hasil segmentasi
cv::Mat segment = image.clone();
// Menggambar lingkaran pada setiap titik sudut yang ditemukan
for (size_t i = 0; i < titik_sudut.size(); i++) {
    cv::circle(segment, titik_sudut[i], 5, cv::Scalar(0, 0, 255), cv::FILLED);
}
// Menggambar kotak ROI pada gambar
cv::rectangle(segment, ROI_area, cv::Scalar(255, 0, 0), 2);

// Mendefinisikan 4 titik sumber (src) dan tujuan (dst) untuk transformasi perspektif (Bird Eye View)
cv::Point2f src[4] = {titik_sudut[6], titik_sudut[5], cv::Point(0, image.rows), cv::Point(image.cols, image.rows)};
cv::Point2f dst[4] = {cv::Point(0+200, 0+200), cv::Point(image.cols-200, 0+200), cv::Point(0+200, image.rows), cv::Point(image.cols-200, image.rows)};

// Membuat matriks transformasi perspektif dan menerapkannya ke gambar asli
cv::Mat BEV_plot;
cv::Mat BEV_view;
BEV_plot = cv::getPerspectiveTransform(src, dst);
cv::warpPerspective(image, BEV_view, BEV_plot, BEV_view.size());

// Menampilkan gambar asli, hasil segmentasi, dan hasil Bird Eye View
cv::imshow("Original Image", image);
cv::imshow("Thresholded Image", image_thresholded);
cv::imshow("Gambar", segment);
cv::imshow("Bird Eye View", BEV_view);

int key = cv::waitKey(0);

return 0;
}
```