Analisa Robot dengan Kamera

1. Camera dengan color recognition

Color recognition adalah teknik mendeteksi warna tertentu dalam citra, sering kali dengan mendefinisikan rentang nilai warna (misalnya, dalam ruang warna HSV atau RGB). Dalam kasus ini, warna yang dideteksi adalah biru, hijau, dan merah.

Dibawah ini adalah Analisa dari beberapa baris kode yang penting.

a. left_motor = wb_robot_get_device("left wheel motor");
   right_motor = wb_robot_get_device("right wheel motor");
   wb_motor_set_position(left_motor, INFINITY);
   wb_motor_set_position(right_motor, INFINITY);
   wb_motor_set_velocity(left_motor, 0.0);
   wb_motor_set_velocity(right_motor, 0.0);

Kode diatas berguna untuk:

- Mendapatkan perangkat motor robot.
- Menetapkan posisi target ke INFINITY, artinya motor beroperasi dalam mode kecepatan.
- Awalnya mengatur kecepatan motor ke 0.

b. while (wb_robot_step(time_step) != -1) {
   ...}

   Berguna untuk menjalankan loop utama dengan interval time_step. Selama simulasi berjalan, loop ini terus dijalankan.

c. const unsigned char *image = wb_camera_get_image(camera);

   Kode diatas untuk mengambil data gambar terbaru dari kamera.

d.
```
for (i = width / 3; i < 2 * width / 3; i++) {
 for (j = height / 2; j < 3 * height / 4; j++) {
  red += wb_camera_image_get_red(image, width, i, j);
  blue += wb_camera_image_get_blue(image, width, i, j);
  green += wb_camera_image_get_green(image, width, i, j);
 }
```

```
}
```

Kode diatas untuk mengakses nilai warna merah, hijau, dan biru dari piksel di area tertentu gambar dan juga fokus pada area tengah (dari lebar dan tinggi) untuk mendeteksi blob.

e.

```
if ((red > 3 * green) && (red > 3 * blue))
  current_blob = RED;
else if ((green > 3 * red) && (green > 3 * blue))
  current_blob = GREEN;
else if ((blue > 3 * red) && (blue > 3 * green))
  current_blob = BLUE;
else
  current_blob = NONE;
```

Kode diatas untuk menentukan warna dominan berdasarkan perbandingan intensitas komponen warna. Cara kerjanya adalah membandingkan setiap nilai warna dengan nilai lainnya menggunakan rasio (contoh: merah tiga kali lebih tinggi dari hijau dan biru) dan jika tidak ada warna dominan, dianggap NONE.

f.

```
else {
  left_speed = 0;
  right_speed = 0;
  printf("Looks like I found a %s%s%s blob.\n", ansi_colors[current_blob], color_names[current_blob], ANSI_COLOR_RESET);
```

Kode diatas berguna untuk mengentikan robot jika ditemukan warna yang signifikan.

2. Focus Camera

Camera focus adalah kemampuan kamera untuk secara otomatis menyesuaikan fokus lensa agar objek di depan terlihat jelas. Pada eksperimen ini, fokus diarahkan pada objek seperti bola sepak, kaleng, buku, buah pir, atau barrel gas.

Beberapa bagian dari kode memiliki fungsi yang sama dengan kode sebelumnya. Dibawah ini adalah penjelasan dari baris kode yang penting.

- ```
  distance_sensor = wb_robot_get_device("distance sensor");
  wb_distance_sensor_enable(distance_sensor, TIME_STEP);
  ```

Kode diatas berguna untuk mengakses perangkat sensor jarak pada robot dan mengaktifkannya untuk membaca data setiap TIME_STEP.

- ```
  const double object_distance = wb_distance_sensor_get_value(distance_sensor) / 1000;
  ```

Kode diatas berguna untuk Membaca nilai jarak dari sensor jarak dan mengonversinya ke meter.

- ```
  wb_camera_set_focal_distance(camera, object_distance);
  ```

Kode diatas berguna untuk menyetel jarak fokus kamera berdasarkan nilai jarak objek yang dideteksi.

3. Object Recognition

Dibawah ini adalah penjelasan beberapa baris kode yang penting.

a.

```
int number_of_objects = wb_camera_recognition_get_number_of_objects(camera);

printf("\nRecognized %d objects.\n", number_of_objects);
```
```

Kode diatas berguna untuk mendapatkan jumlah objek yang dikenali kamera dalam frame saat ini dan menampilkan jumlah objek tersebut di terminal.

b.

```
printf("Model of object %d: %s\n", i, objects[i].model);

printf("Id of object %d: %d\n", i, objects[i].id);
```

Kode diatas berguna untuk menampilkan nama model 3D objek (misalnya, "Can", "Ball") dan juga  Identifikasi unik untuk objek tersebut.

c.

```
printf("Relative position of object %d: %lf %lf %lf\n", i, objects[i].position[0], objects[i].position[1],
    objects[i].position[2]);
```

Kode diatas berguna untuk menampilkan posisi relatif objek dalam koordinat dunia (X, Y, Z) relatif terhadap kamera.

d.

```
printf("Relative orientation of object %d: %lf %lf %lf %lf\n", i, objects[i].orientation[0], objects[i].orientation[1],
    objects[i].orientation[2], objects[i].orientation[3]);
```

Kode diatas berguna untuk menampilkan orientasi objek dalam quaternion relatif terhadap kamera.

e.

```
printf("Size of object %d: %lf %lf\n", i, objects[i].size[0], objects[i].size[1]);
```

Kode diatas berguna untuk menampilkan ukuran objek dalam dimensi X dan Y.

f.

```
printf("Position of the object %d on the camera image: %d %d\n", i,
objects[i].position_on_image[0],
    objects[i].position_on_image[1]);
printf("Size of the object %d on the camera image: %d %d\n", i,
objects[i].size_on_image[0], objects[i].size_on_image[1]);
```

Kode diatas berguna untuk menampilkan Posisi objek pada citra kamera (koordinat piksel) dan juga ukuran objek dalam citra kamera (piksel X dan Y).

g.

```
for (j = 0; j < objects[i].number_of_colors; ++j)
 printf("- Color %d/%d: %lf %lf %lf\n", j + 1, objects[i].number_of_colors,
objects[i].colors[3 * j],
      objects[i].colors[3 * j + 1], objects[i].colors[3 * j + 2]);
```

Kode diatas berguna untuk menampilkan warna objek yang diwakili dalam format RGB, dengan number_of_colors menunjukkan jumlah warna yang terdeteksi untuk objek tersebut.

4. Segmented Image

Penjelasan dari baris kode yang penting ada di bawah ini.

a. `if (wb_camera_recognition_is_segmentation_enabled(camera) && wb_camera_recognition_get_sampling_period(camera) > 0) {`

Kode diatas berfungsi untuk memastikan fitur segmentasi kamera aktif dan memiliki interval sampling yang valid.

b. const unsigned char *data =
wb_camera_recognition_get_segmentation_image(camera);

Kode diatas berfungsi untuk  mengambil segmented image dari kamera. Gambar ini berupa data pixel dalam format BGRA (Blue, Green, Red, Alpha).

c.

segmented_image = wb_display_image_new(display, width, height, data, WB_IMAGE_BGRA);

wb_display_image_paste(display, segmented_image, 0, 0, false);

wb_display_image_delete(display, segmented_image);

Kode diatas berfungsi untuk

- Membuat objek gambar dari data segmentasi.
- Menampilkan gambar tersebut pada perangkat display dengan posisi (0, 0).
- Menghapus gambar setelah ditampilkan untuk mengelola memori.

5. Noise Masking

Untuk kodenya sama dengan kode camera biasa diatas. Perbedaannya adalah untuk noise masking dapat terlihat bahwa yang di run adalah file .exe yang tidak bisa dibuka. Oleh karena itu, mungkin saya akan jelaskan saja apa yang saya pelajari dari simulasi noise masking ini.

Noise masking adalah teknik untuk menghilangkan atau mengurangi gangguan visual (noise) dalam citra yang dapat mengaburkan atau mempengaruhi proses deteksi objek. Dalam simulasi ini, terlihat ada garis garis yang menghalangi kamera. Dalam konteks deteksi blob warna (merah, hijau, biru), noise masking memastikan bahwa hanya warna-warna yang diinginkan (blob warna) yang terdeteksi dengan tepat, sementara elemen-elemen yang tidak diinginkan diabaikan. Cara kerjanya adaalh sebagai berikut :

a. Seleksi Area: Fokus hanya pada area tertentu dalam gambar untuk menghindari noise yang muncul di tepi atau area yang tidak relevan.

b. Thresholding: Menggunakan batasan atau ambang tertentu (threshold) untuk membedakan objek yang relevan berdasarkan intensitas warnanya, seperti membandingkan intensitas merah, hijau, dan biru, untuk memastikan bahwa hanya warna tertentu yang terdeteksi.

c. Pengabaian Gangguan: Dengan menggunakan teknik ini, gangguan dari cahaya, objek lain, atau variasi warna kecil dapat diabaikan, sehingga meningkatkan presisi dalam mendeteksi objek yang diinginkan.

6. Motion blur

Sama dengan nomor 5, saya tidak bisa menemukan kode untuk motion blurnya, hanya ada kode camera dan file .exe. Oleh karena itu, akan dijelaskan apa yang saya pelajari dari simulasi motion blur ini.

Motion blur dapat menjadi sebuah tantangan dalam deteksi warna. Pengaruhnya adalah sebagai berikut:

1. Pengaburan Warna: Motion blur dapat menyebabkan warna objek (seperti merah, hijau, atau biru) tercampur atau terdistorsi, sehingga mengurangi akurasi dalam mendeteksi blob warna. Misalnya, sebuah objek merah yang bergerak cepat mungkin terlihat lebih pucat atau bahkan hijau karena blur yang terjadi.
2. Kesulitan dalam Memetakan Posisi: Ketika sebuah objek bergerak cepat, gambar yang dihasilkan akan menunjukkan banyak piksel yang memiliki warna campuran dari objek dan latar belakang, sehingga mengurangi kemampuan sistem untuk memetakan posisi dan mendeteksi blob dengan tepat.
3. Penyaringan Warna yang Lebih Sulit: Jika gambar buram, sulit untuk menetapkan batas yang jelas antara warna yang relevan dan gangguan lainnya. Noise masking bisa membantu, tetapi jika blur terlalu parah, proses thresholding atau pemilihan warna berdasarkan intensitas bisa gagal. Dalam simulasi, camera terlihat lebih lama dalam mendeteksi warna.