Dokumentasi AR Drone

Kelompok IV ROBOTIKA : (SandalTerbang)

Fikrul Arif Nadra

Futuhal Arifin Annasri

M. Febrian Rachmadi

Pulung Ragil Lanang

Fakultas Ilmu Komputer

Universitas Indonesia

I. Cara instalasi node

Dalam tugas kali ini, kami membuat sebuah node yang bernama **eagle_strike** untuk menjalankan fungsi utama AR Drone (mendeteksi objek dan bergerak untuk mengikutinya). Berikut langkah-langkah dalam melakukan instalasi node **eagle_strike**.

1. Buatlah package dalam path ROS anda, dengan perintah yang telah disediakan oleh ROS, yaitu

   roscreate-pkg [nama package] [dependensi1] [dependensi2] [...]

   Penggunaan perintah ini akan mengurangi terjadinya kegagalan kompilasi karena kekurangan file atau hal-hal yang dilakukan sendiri oleh user. Oleh karena itu ROS membuatkan package untuk user.

2. Pada tugas ini, nama *package* yang digunakan adalah **eagle_strike** dengan dependensi **cmvision** dan **ardrone_brown.**

   Kami melakukan instalasi dependensi package **eagle_strike**, dengan perintah:

   ```
   rosdep install eagle_strike
   ```

   Setelah menginstal dependensi package **eagle_strike**, kami melanjutkan dengan build package:

   ```
   rosmake eagle_strike
   ```

3. Maka package **eagle_strike** telah terinstall dalam ROS. Node dapat mulai dibuat dalam path: [PATH_ROS]/eagle_strike/src. Bahasa yang didukung ada dua, yaitu C++ dan Python.

   Tutorial pembuatan nodes lebih lanjut: (kami menggunakan C++)
   ROS Tutorials – Writings Publisher Subscriber (C++)

http://www.ros.org/wiki/ROS/Tutorials/WritingPublisherSubscriber%28c%2B%2B%29

Node yang kami buat:

1. **eagle_listener**

Node ini berfungsi untuk mengikuti benda target sesuai spesifikasi warna cmvision yang tercatat pada *file* cmvision/color_spec.txt.

2. **distance_tracker**

Node ini berfungsi untuk mencatat jarak pergerakan AR Drone, sehingga dapat diketahui posisinya.

II. Cara menjalankan program

Berikut langkah-langkah dalam menjalankan program dan AR Drone.

1. Jalankan perintah `roscore` untuk menjalankan ROS.
2. Jalankan perintah `rosrun ardrone_brown ardrone_driver` untuk menyalakan driver ARDrone pada ROS. (pastikan komputer telah terhubung dengan ARDrone)
3. Jalankan perintah `roslaunch cmvision cmvision.launch` untuk menjalankan pendeteksi warna.
4. Ubah posisi kamera aktif dari kamera depan menjadi kamera bawah dengan perintah `rosservice call /ardrone/togglecam`.
5. Jalankan perintah `rosrun drone_teleop drone_teleop.py` untuk mendapatkan kendali manual atas AR Drone. (jika terjadi sesuatu yang tidak diinginkan)

Untuk menjalankan perintah mengikuti target dengan warna yang diinginkan:

1. Jalankan perintah `rosrun eagle_strike eagle_listener`
2. Tunggu beberapa saat hingga AR Drone melakukan *take off*.
3. AR Drone akan mengikuti target sesuai dengan warna yang telah tersimpan di dalam *file* cmvision/color_spec.txt.
4. Perhatian: ARDrone harus melakukan *landing* dari `drone_teleop` yang telah diaktifkan sebelumnya.

Untuk menjalankan perintah mencatat pergerakan AR Drone:

1. Jalankan perintah `rosrun eagle_strike distance_tracker`
2. Untuk melihat grafik perubahan posisi AR Drone, jalankan perintah `rxplot -M "3d" /drone_pos/position/x /drone_pos/position/y /drone_pos/position/z`
3. Jalankan AR Drone dengan perintah `drone_teleop` atau dengan `rosrun eagle_strike eagle_listener`. Perubahan posisi AR Drone akan tercatat pada grafik.

III. Penjelasan cara kerja dan teknik yang digunakan

A. Penjelasan PID

PID yang memiliki kepanjangan Proportional Integral Derivative, merupakan metode yang paling sederhana untuk melakukan kendali atau kontrol suatu robot. Namun, metode ini tidak selalu yang terbaik pada semua kondisi.

Ada tiga bagian kontrol atau kendali pada metode PID ini. Ketiga bagian tersebut diantaranya ialah:

- Kendali Proposional (*Proportional Controller*)

  Pada banyak aplikasi kendali, perubahan yang mendadak antara nilai kendali motor tetap dan nol tidak mengakibatkan perilaku kendali yang baik. Hal ini dapat diatasi dengan menggunakan *term* linear atau proporsional. Rumus untuk kendali proporsional ini, yaitu:

  $R(t) = K_p . (v_{des}(t) - v_{act}(t))$

  Perbedaan antara kecepatan yang diinginkan dengan kecepatan yang sebenarnya disebut sebagai “fungsi eror”. Perubahan nilai K_p akan mempengaruhi perilaku kendali. Semakin tinggi nilai K_p yang dipilih, semakin cepat respon kendalinya. Namun, bila nilai K_p yang diberikan sangat tinggi, akan menyebabkan osilasi yang tidak diinginkan. Untuk itu, sangat penting untuk memilih nilai K_p yang tepat dan dapat menjamin respon kendali yang cepat tetapi tidak menyebabkan osilasi.

- Kendali Integral (*Integral Controller*)

  Tidak seperti kendali proposional, kendali integral sangat jarang digunakan sendirian, tetapi seringkali banyak digunakan sebagai kombinasi dengan kendali proposional dan derivatif. Kendali integral ini, dimaksudkan untuk mengurangi *steady-state error* pada kendali proposional. Dengan penambahan hubungan integral, *steady-state error* tersebut dapat dikurangi hingga nilainya nol.

- Kendali derivatif (*Derivative Controller*)

  Mirip dengan kendali integral, kendali derivatif jarang digunakan sendirian, tetapi lebih sering digunakan bersamaan dengan kendali proposional dan integral. Ide dari digunakannya kendali ini ialah untuk menambah kecepatan respon pada kendali proposional terhadap perubahan nilai input.

B. Konstanta PID Pada Dr. Frankenstein's Eagle

Terdapat dua buah kendali PID dalam Dr. Frankenstein's Eagle, yaitu kendali PID untuk kendali PID agar AR Drone terbang stabil di udara (*hovering*) dan terbang mengejar kelinci. Untuk kendali PID terbang diam dan stabil di udara, kendali PID yang digunakan adalah kendali proporsional saja. Hal ini dikarenakan kendali proporsional sudah cukup membuat AR Drone terbang stabil di udara. Konstanta kendali proporsional (K_P) yang digunakan adalah sebesar 0.006, dan konstanta-konstanta lainnya, yaitu kendali integral (K_I) dan kendali diferensial (K_D) bernilai 0.

Untuk kendali PID terbang mengejar objek, kendali PID yang digunakan adalah juga kendali proporsional saja. Hal ini dikarenakan kendali proporsional sudah cukup membuat AR Drone terbang mengikuti objek yang sudah ditentukan. Konstanta kendali proporsional (K_P) yang digunakan adalah sebesar 0.002, dan konstanta-konstanta lainnya, yaitu kendali integral (K_I) dan kendali diferensial (K_D) bernilai 0.

C. Parameter Input Fungsi Kendali PID

Kendali PID yang digunakan dalam menyelesaikan permasalahan Dr. Frankenstein's Eagle terbagi menjadi dua kelompok, yaitu kendali PID terbang stabil di udara dan PID terbang mengejar objek. Setiap kelompok, terdapat sebuah kendali PID untuk koordinat x dan koordinat y, sehingga secara total terdapat empat buah fungsi control PID.

Untuk fungsi kendali PID terbang stabil, baik koordinat x dan koordinat y, fungsi kendali PID yang digunakan adalah:

```
/** Hanya menggunakan P dan konsep PID dasar. */
float computePIDX(float v_des, float v_act) {

    float e_func = v_des - v_act;
    float r_mot = KPX * (e_func);
    r_mot = std::min(r_mot, 0.5f);
    r_mot = std::max(r_mot, -0.5f);
    ROS_INFO("RMOTX = %f", r_mot);
    return r_mot;
}
```

Untuk X

```
/** Hanya menggunakan P dan konsep PID dasar. */
float computePIDY(float v_des, float v_act) {

    float e_func = v_des - v_act;
    float r_mot = KPY * (e_func);
    r_mot = std::min(r_mot, 0.5f);
    r_mot = std::max(r_mot, -0.5f);
    ROS_INFO("RMOTY = %f", r_mot);
    return r_mot;
}
```

Untuk X

Untuk fungsi kendali PID terbang mengejar objek, baik koordinat x dan koordinat y, fungsi kendali PID yang digunakan adalah:

```
/** Hanya menggunakan P dan konsep PID dasar. */
float computePIDblobX(float v_des, float v_act) {

    float e_func = v_des - v_act;
    float r_mot = KPblobX * (e_func);

    ROS_INFO("\nX, Blob, err = %f, %f, %f", v_des, v_act, e_func);

    r_mot = std::min(r_mot, 0.5f);
    r_mot = std::max(r_mot, -0.5f);

    ROS_INFO("RMOTX = %f", r_mot);
    return r_mot;
}
```

Untuk X

```
/** Hanya menggunakan P dan konsep PID dasar. */
float computePIDblobX(float v_des, float v_act) {

    float e_func = v_des - v_act;
    float r_mot = KPblobX * (e_func);

    ROS_INFO("\nX, Blob, err = %f, %f, %f", v_des, v_act, e_func);

    r_mot = std::min(r_mot, 0.5f);
    r_mot = std::max(r_mot, -0.5f);

    ROS_INFO("RMOTX = %f", r_mot);
    return r_mot;
}
```

Untuk Y

D. Pendeteksian Objek (deteksi blob warna)

Latar Belakang

Pada awalnya kami mencari berbagai macam benda yang memiliki warna mencolok. Namun faktor cahaya mempengaruhi berbagai macam benda, sehingga berubah dari warna aslinya. Contoh: warna biru menjadi hitam, warna kuning menjadi agak putih. Hal ini membuat kami harus mencari warna lain agar blob dapat terdeteksi dengan baik.

Faktor kamera juga berpengaruh, yaitu jika pada saat pertama kali kamera depan tidak ditutup sebelum kamera aktif diganti menjadi kamera bawah, warna dapat terdeteksi pada kamera merah dan membuat ARDrone tetap mencari titik blob tersebut walau sebenarnya blob tersebut berasal dari data lama kamera depan, bukan kamera belakang.

Pendeteksian objek, dilakukan dengan menggunakan program cmvision. Hal yang dilakukan adalah melakukan pendeteksian warna objek dengan menggunakan kamera bawah AR Drone. Pada program, akan muncul nilai RGB dan threshold objek tersebut. Selanjutnya, data-data yang didapat (berupa nilai warna objek) disimpan dalam suatu file bernama color_spec.txt yang ada pada folder cmvision.

E. Objek yang Digunakan

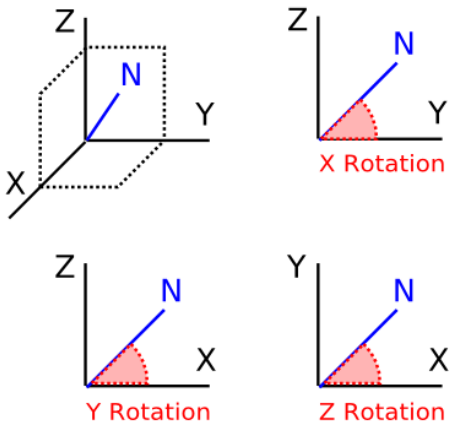
Objek yang kami gunakan adalah karpet merah milik Lab 1231. Karpet ini kami pilih karena warnanya yang merah, kontras dengan warna lantai tempat uji coba (lorong gedung A), serta warnanya yang tidak mendekati warna lantai (terang) maupun warna hitam. Permukaan benda yang lebar (saat uji coba karpetnya dilipat terlebih dahulu)

mendukung pendeteksian blob dengan peluang lebih besar. Warna karpet ini dalam RGB adalah (83, 57, 57) dengan threshold (64:74, 121:126, 134:153).

Gambar 1 Karpet Merah Lab 1231

F. Variasi Tugas (Menunjukkan perkiraan posisi ARDrone)

Variasi tugas A mengharuskan kami untuk menunjukkan posisi AR.Drone dalam koordinat absolut dan melakukan *publish* topik bernama "/drone_pos" yang berisi posisi absolut (**geometry_msgs/Pose**) AR.Drone. Koordinat absolut didapatkan dengan melakukan integrasi terhadap data vx, vy, dan vz yang ada di dalam navdata. Sebelum diintegrasikan, sebelumnya vx, vy, dan vz harus ditransformasikan dengan menggunakan matriks rotasi terhadap sumbu x, sumbu y, dan sumbu z secara bersamaan sehingga didapatkan vx', vy', dan vz' dalam dimensi linear tiga dimensi yang bersesuaian dengan sumbu-sumbu x, y, dan z.



Gambar 2 Rotasi 3D terhadap sumbu x, sumbu y, dan sumbu z

Transformasi gerak AR.Drone dapat ditransformasikan dengan menggunakan matriks transformasi. Secara umum, matriks rotasi untuk masing-masing sumbu adalah sebagai berikut.

a. Rotasi terhadap sumbu x

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix}$$

b. Rotasi terhadap sumbu y

$$R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}$$

c. Rotasi terhadap sumbu z

$$R_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Perhatikan bahwa θ adalah besar sudut yang terjadi antara arah gerak objek (dalam hal ini AR.Drone) terhadap masing-masing sumbu. Data ini dapat diperoleh dari navdata AR.Drone yang bernama rotX, rotY, dan rotZ. Karena pergerakan AR.Drone adalah pergerakan dalam bidang 3D, maka transformasi rotasi juga harus dalam bentuk matriks rotasi bidang 3D. Untuk mendapatkan matriks rotasi 3D, maka dilakukanlah perkalian matriks terhadap ketiga matriks rotasi Rx, Ry, dan Rz. Hasil perkalian ketiga matriks rotasi Rx, Ry, dan Rz akan menghasilkan matriks rotasi 3D di bawah ini,

$$\begin{bmatrix} \cos\theta\cos\psi & -\cos\phi\sin\psi + \sin\phi\sin\theta\cos\psi & \sin\phi\sin\psi + \cos\phi\sin\theta\cos\psi \\ \cos\theta\sin\psi & \cos\phi\cos\psi + \sin\phi\sin\theta\sin\psi & -\sin\phi\cos\psi + \cos\phi\sin\theta\sin\psi \\ -\sin\theta & \sin\phi\cos\theta & \cos\phi\cos\theta \end{bmatrix}$$

dimana:
θ = Besar sudut antara AR.Drone dengan sumbu y
$\emptyset$ = Besar sudut antara AR.Drone dengan sumbu x
φ = Besar sudut anatara AR.Drone dengan sumbu z

Misalkan hasil perkalian ketiga matriks rotasi tersebut adalah R, maka vx', vy', dan vz' dalam dimensi linear tiga dimensi yang bersesuaian dengan sumbu-sumbu x, y, dan z adalah

$$R.\begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} = \begin{bmatrix} v'_x \\ v'_y \\ v'_z \end{bmatrix}$$

dimana:

v_x = Kecepatan AR.Drone terhadap sumbu x (pergerakan ke depan bernilai positif)

v_y = Kecepatan AR.Drone terhadap sumbu y (pergerakan ke kiri bernilai positif)

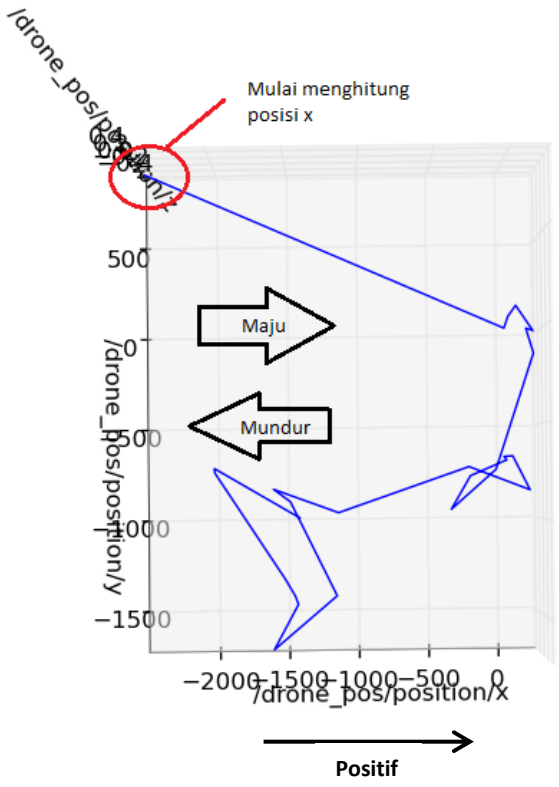
v_y = Kecepatan AR.Drone terhadap sumbu z (pergerakan ke atas bernilai positif)

v'_x = Kecepatan AR.Drone pada sumbu x (pergerakan ke depan bernilai positif)

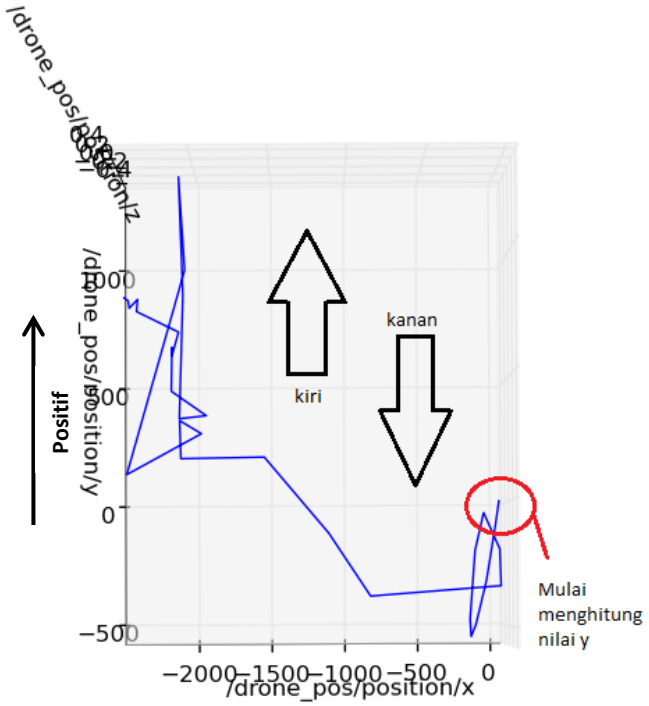
v'_y = Kecepatan AR.Drone pada sumbu y (pergerakan ke kiri bernilai positif)

v'_y = Kecepatan AR.Drone pada sumbu z (pergerakan ke atas bernilai positif)

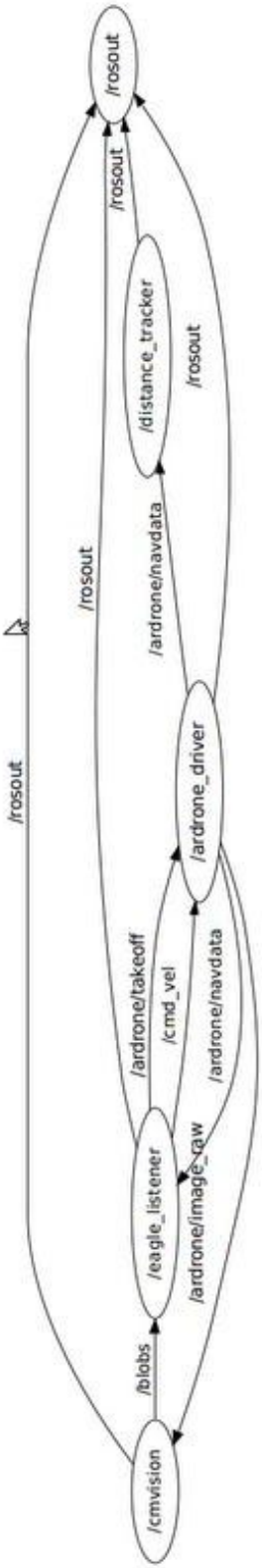
Berikut adalah contoh plot diagram pergerakan AR.Drone pada sumbu x.



Berikut adalah contoh plot diagram pergerakan AR.Drone pada sumbu y.



Berikut di bawah ini adalah tampilan rx_graph dari keseluruhan sistem.



IV. Pembagian kerja dalam kelompok

Nama	Tugas
Fikrul Arif Nadra	Membantu Coding
Futuhal Arifin Annasri	Membantu Coding
	Dokumentasi
	Membuat video
M. Febrian Rachmadi	Coding
	Membantu Dokumentasi
Pulung Ragil Lanang	Coding
	Membantu pembuatan video