Indonesian Journal of Electronics and Instrumentation Systems (IJEIS)
Vol.8, No.2, October 2018, pp. 131~142
ISSN (print): 2088-3714, ISSN (online): 2460-7681
DOI: 10.22146/ijeis.23031

Rancang Bangun Sistem Pendeteksi ARTag sebagai Landasan Pacu AR.Drone Menggunakan *Platform* ROS *(Robot Operating System)*

Bakhtiar Alldino Ardi Sumbodo*[1], Ariestyo Rahardian[2]
[1]Departemen Ilmu Komputer dan Elektronika, FMIPA UGM, Yogyakarta
[2]Perusahaan Listrik Negara (PLN)
e-mail: ***[1]b.alldino.as@ugm.ac.id,** [2]ariestyorahardian@gmail.com

Abstrak

Pengembangan quadrotor jenis AR.Drone untuk penelitian sedang banyak dikembangkan, salah satunya adalah sistem pendeteksian objek berbasis pengolahan citra untuk melakukan misi, seperti tracking, landing, atau mendeteksi dan melewati objek halangan. Berdasarkan hal tersebut, pada penelitian ini dirancang sebuah sistem pendeteksian ARTag yang berbasis pada pengolahan citra digital menggunakan pustaka openCV yang diimplementasikan pada platform ROS yang digunakan untuk menghubungkan drone dengan PC yang kemudian akan dilakukan misi untuk pendaratan. Metode yang digunakan adalah Thresholding, Contour Detection, dan Image moments.

Hasil dari penelitian ini berupa sebuah sistem yang mampu mendeteksi objek ARTag beserta ID nya. Uji coba sistem dilakukan pengujian waktu komputasi program pada keadaan statis dan dinamis, pengujian performa pendeteksian pada keadaan statis dan dinamis, pengujian performa pendeteksian terhadap sudut antara AR.Drone dengan ARTag, pengujian performa pendeteksian terhadap intensitas cahaya dan pengujian misi pendaratan. Kesimpulan yang didapat yaitu ketinggian optimal pendeteksian pada keadaan statis dan dinamis adalah 150 cm, memiliki kehandalan yang baik terhadap intensitas cahaya, dan AR.Drone dapat melakukan misi pendaratan dengan tingkat keberhasilan 70%.

Kata kunci*— Quadrotor, Fiducial marker, OpenCV*

Abstract

The development of quadrotor type AR.Drone for research is being developed. One of which is an object detection system based on image processing to perform the mission, such as tracking, landing, or detect and pass the object hitch. Accordingly, in this research designed an ARTag detection system based on digital image processing using OpenCV library implemented in ROS platform used to connect the drone to PC which will then be carried out missions to landing. The method used is Thresholding, Contour Detection, and Image Moments.

The result of this research is a system that able to detect ARTag objects along with his ID. The system is tested by computing time of the program in the static and dynamic state, detection performance testing in static and dynamic state, detection performance testing of the angle between the AR.Drone and ARTag, detection performance testing of the light intensity, and testing landing mission. The conclusions are the optimal height of detection in the static and dynamic state is 150 cm, has excellent reliability to the light intensity, and the AR.Drone can perform landing mission with a success rate of 70%.

Keywords*— Quadrotor, Fiducial marker, OpenCV*

Received March 15th,2017; Revised May 30th, 2018; Accepted October 31th, 2018

1. PENDAHULUAN

UAV (*Unmanned Aerial Vehicle)* atau pesawat tanpa awak atau *drone* adalah sebuah mesin yang mampu terbang dan dikendalikan oleh pilot dari jarak jauh. Beberapa tahun belakangan, UAV mulai digemari di Indonesia terutama untuk keperluan peliputan berita seperti peliputan video bencana, kemacetan lalu lintas ataupun selebrasi acara tertentu. Industri hiburan dan sipil juga menggunakan UAV sebagai alat penangkap foto maupun video yang dirasa lebih baik hasilnya jika diambil dari udara. Parrot AR.Drone merupakan salah satu UAV jenis *quadrotor*. AR.Drone 2.0 merupakan salah satu contoh pesawat tanpa awak yang berjenis multirotor dan biasa digunakan untuk riset [1].

Sejak dirilisnya Parrot AR.Drone, banyak programmer dan *developer* yang mengembangkan sistem AR.Drone. Banyak produsen drone yang berkonsentrasi mengembangkan sistem pengolahan citra digital. Sehingga, banyak penelitian maupun lomba dengan menggunakan AR.Drone yang telah dikembangkan. Lomba tersebut salah satunya mengharuskan AR.Drone untuk terbang dan mendarat pada landasan secara *autonomous* dengan memanfaatkan pengolahan citra digital untuk mendeteksi landasan. Deteksi objek dalam pengolahan citra digital adalah suatu proses yang digunakan untuk menentukan keberadaan objek tertentu di dalam suatu citra digital [2].

ARTag adalah sebuah sistem *fiducial marker* untuk mendukung *augmented reality*. ARTag adalah sistem *marker* yang menggunakan teori *coding* digital untuk mendapatkan *false positive rate* dan *inter-marker confusion rate* yang sangat rendah dengan ukuran *marker* yang dibutuhkan kecil. ARTag *marker* adalah *bi-tonal planar patterns* (hanya hitam dan putih) yang berisi nomor ID [3].

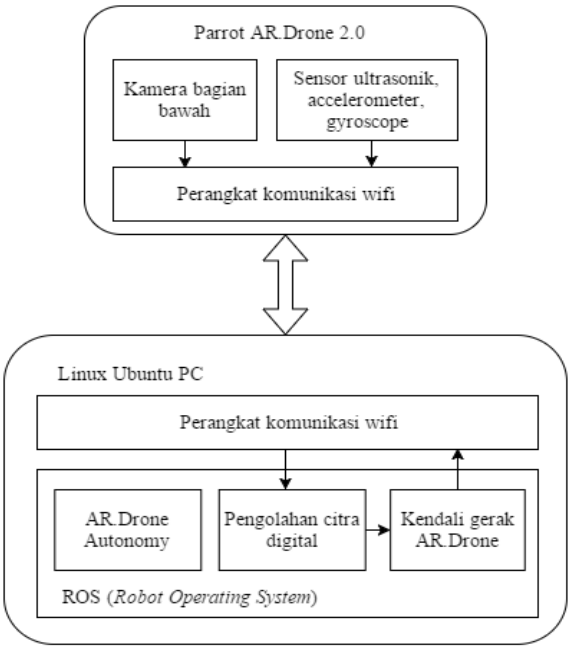
Robot Operating System (ROS) adalah *framework* yang berisi *library*, *driver*, maupun peralatan (*tools*) untuk memudahkan pembuatan program yang kompleks pada berbagai *platform* robot. ROS digunakan agar setiap *developer* robot dapat mengembangkan program dengan *style* yang sama, sehingga terjadi kesamaan *programming style* yang bertujuan agar memudahkan komunitas sesama *developer* robot untuk *sharing* dan mengembangkan robot dengan berbagai jenis [4]. Parrot AR.Drone merupakan salah satu jenis robot yang memiliki kompatibilitas dengan sistem ROS.

Dengan begitu, pengolahan citra digital dapat diterapkan pada AR.Drone dengan menggunakan *platform* ROS. Pengolahan citra digital lebih mangacu pada pengolahan gambar digital melalui komputer [5]. Penelitian yang telah dilakukan ini yaitu mengimplementasikan pengolahan citra digital pada AR.Drone agar mampu mendeteksi ARTag dan melakukan pendaratan secara *autonomous* dengan memanfaatkan pengolahan citra digital atau *computer vision* dan *platform* ROS. *Computer vision* adalah ilmu pemrograman komputer untuk memproses dan akhirnya memahami gambar dan video, atau membuat komputer dapat melihat [6].

2. METODE PENELITIAN

2.1 Analisis Sistem

Sistem yang dirancang pada penelitian ini merupakan suatu sistem pendeteksian landasan *quadrotor* AR.Drone 2.0 berbasis pengolahan citra. Perangkat lunak pengolahan citra dibuat dengan menggunakan pustaka OpenCV. Paket OpenCV diintegrasikan dengan paket AR.Drone untuk mengendalikan gerak AR.Drone. Integrasi paket dilakukan di atas *platform* ROS, sedangkan AR.Drone dihubungkan dengan PC melalui jaringan nirkabel atau *wifi* untuk dilakukan transmisi data, baik untuk penerimaan citra dari AR.Drone ke PC maupun pengiriman perintah dari PC ke AR.Drone. Rancangan sistem secara keseluruhan ditunjukkan pada diagram blok rancangan sistem pada Gambar 1.



Gambar 1 Diagram blok sistem keseluruhan

Mengacu Gambar 1, sistem ini memiliki dua bagian penting, yaitu Parrot AR.Drone 2.0 dan PC. Pada penelitian ini AR.Drone memiliki tiga komponen yang digunakan, yaitu jaringan komunikasi *wifi*, kamera bagian bawah, dan sensor *ultrasonic*, *accelerometer*, dan *gyroscope*. Sedangkan PC memiliki dua komponen, yaitu jaringan komunikasi *wifi*, dan *platform* ROS. Di atas *platform* ROS diintegrasikan tiga komponen penting, yaitu AR.Drone *autonomy*, pengolahan citra digital, dan kendali gerak *drone*. Dimana AR.Drone *autonomy* adalah sebuah *driver* ROS untuk Parrot AR.Drone versi 1 dan 2. Kedua bagian utama tersebut saling berhubungan dan dirancang untuk dapat mendeteksi ARTag dan melakukan pendaratan AR.Drone di atasnya secara *autonomous*.

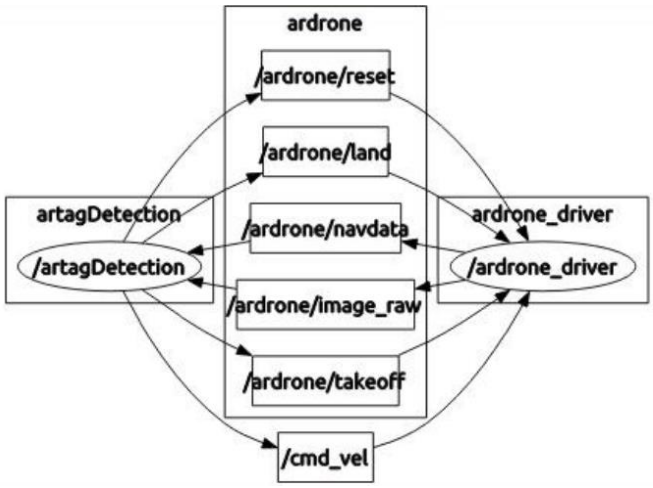
Proses kerja sistemnya ialah ARTag yang memiliki ID tetap dikategorikan sebagai objek yang dikenali sebagai landasan pacu oleh kamera bagian bawah AR.Drone. ARTag yang digunakan berukuran 14.7 x 14.7 cm. Pertama, kamera bagian bawah *drone* menangkap citra video terhadap ARTag yang berada di bawahnya pada saat *drone* melakukan penerbangan. Kemudian data citra video tersebut dikirim ke PC melalui koneksi nirkabel secara terus-menerus. Selanjutnya citra video yang diterima PC digunakan untuk mengendalikan AR.Drone melalui pengolahan citra digital. ARTag yang berhasil dideteksi oleh PC selanjutnya dibandingkan dengan data ID untuk pengendalian drone. Apabila ID ARTag sama dengan ID pendaratan, maka PC akan mengirimkan perintah ke drone untuk mendarat di atas landasan pacu tersebut. Pengolahan citra digital pada penelitian ini menggunakan pustaka dari OpenCV dan dijalankan pada *platform* ROS. Pengendalian *drone* menggunakan *platform* ROS yang telah ter*install* pada PC. *Platform* ROS pada penelitian ini mempunyai peran sebagai media pengolahan citra digital dengan menggunakan pustaka OpenCV, memproses komunikasi antar *drone* dengan PC, serta transfer data citra dari *drone* maupun memberi perintah kendali terhadap *drone*.

2.2 Rancangan Pengoperasian Robot Operating System (ROS)

Penelitian ini menggunakan pustaka/*library Robot Operating System* (ROS) sebagai *platform* pengendali Parrot AR.Drone 2.0 yang ditanam pada sistem operasi Ubuntu versi 14.04. Peran *platform* ROS pada penelitian ini adalah sebagai penghubung transfer data citra maupun sinyal kendali, pengendali gerak *drone*, mengampilkan data navigasi selama *drone* melakukan misi, serta sebagai media pengolahan citra digital berupa video. ROS yang digunakan dan

compatible dengan sistem operasi yang digunakan untuk melakukan penelitian ini adalah ROS versi ke-empat yaitu ROS Indigo. Pustaka ROS terlebih dahulu diunduh untuk kemudian di-*install* pada PC. Selanjutnya, dibuat *project workspace* yang berisi program pengolahan citra digital dan kendali gerak *drone* untuk melakukan misi pendaratan.

Dalam ROS ada yang disebut dengan *Node*, *Topic*, *Publisher*, dan *Subscriber*. *Node* pada ROS adalah seperti sebuah aplikasi yang berjalan di atas sebuah sistem operasi. Sebuah paket di dalam *workspace* dapat berisi lebih dari satu buah *node*. Sebuah *node* dapat melakukan komunikasi dengan *node* lainnya. *Topic* adalah sebuah tempat dimana node dapat mem-*publish* atau *men*-subscribe *message*/data. Tiap *topic* akan tetap aktif selama node yang melakukan *publish* data masih aktif, walaupun data dari *topic* tersebut tidak di-*subscribe* oleh *node* lain. *Publisher* adalah istilah yang diberikan kepada node yang melakukan pengiriman atau penerbitan (*publish*) data. Sedangkan s*ubscriber* adalah istilah yang diberikan kepada node yang melakukan penerimaan atau berlangganan (*subscribe*) data. Rancangan node ROS pada penelitian ini ditunjukkan oleh Gambar 2.

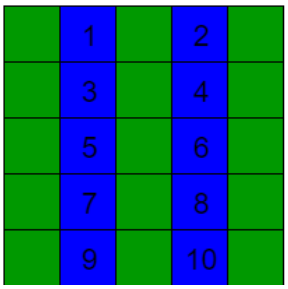


Gambar 2 Rancangan node ROS

Dalam implementasinya, dibutuhkan `NodeHandle` untuk menangani proses *publish* dan *subscribe* data pada aplikasi program. `NodeHandle` yang digunakan bernama `node`. Selanjutnya `NodeHandle node` digunakan untuk men-*subscribe* fungsi `ImageTransport` dengan target ROStopic `/ardrone/image_raw` untuk melakukan proses transfer data berupa citra video langsung dari kamera *drone*. `ImageTransport` harus selalu digunakan untuk *publish* dan *subscribe* citra gambar [7]. Pengambilan data navigasi dilakukan dengan cara men-*subscribe* ROStopic `/ardrone/navdata` dengan menggunakan `NodeHandle node`. Dalam pengoperasiannya sistem ROS perlu dijalankan terlebih dahulu agar program aplikasi ROS dapat dijalankan. Sistem ROS diaktifkan dengan menggunakan perintah `roscore` [8].

2.3 Proses Pembuatan Objek Berupa ARTag

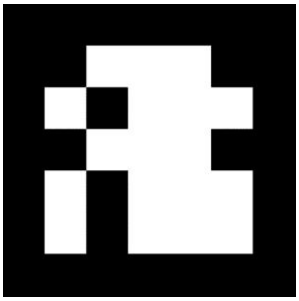
Sebelum dilakukan proses pendeteksian dan misi pendaratan maka hal yang paling utama pada penelitian ini adalah pembuatan objek berupa ARTag. Pembuatan ARTag ini menggunakan modifikasi dari prinsip kode *Hamming* [9]. Kode *Hamming* merupakan salah satu jenis *linear error correcting code* yang sederhana, yang berisi bit data dan bit paritas. ARTag yang digunakan dalam penelitian ini sebagai objek landasan pacu AR.Drone berukuran 7x7 persegi, dimana persegi-persegi terluar berwarna hitam sebagai pembatas. Sehingga kode atau ID direpresentasikan dari 5x5 persegi di dalamnya. Lima baris persegi ini direpresentasikan dari bit-bit masukan yang telah ditentukan sebagai ID.



Gambar 3 Representasi posisi bit data dan bit paritas

Mengacu pada Gambar 3, setiap baris terdiri dari lima bit, yang terbentuk dari dua bit data dan tiga bit paritas. Bit data terletak pada bit kedua dan keempat, sedangkan bit paritas terletak pada bit pertama, ketiga, dan kelima. Nilai pada bit-bit paritas ditentukan oleh bit data. Dalam setiap baris memiliki empat kemungkinan bit data, yaitu 00, 01, 10, dan 11. Untuk data 00, satu baris tersebut ditentukan dengan bilangan hexa 0x10. Bila dijadikan bilangan biner, maka satu baris tersebut berisi nilai 10000 (bit data pada bit kedua dan keempat bernilai 00 dan bit paritas pada bit pertama, ketiga, dan kelima bernilai 100). Untuk data 01, satu baris tersebut ditentukan dengan 0x17, yaitu 10111. Untuk data 10, ditentukan dengan 0x09, yaitu 01001. Dan data 11 ditentukan dengan 0x0e, yaitu 01110. Peletakan bit-bit data diatur secara urut dari input ID dalam bilangan biner dari bit pertama hingga bit ke-sepuluh. Masing-masing baris memiliki dua buah bit data sehingga totalnya terdapat 1024 kemungkinan/ID pada ARTag berukuran 5x5.

Dalam penelitian ini ARTag yang digunakan sebagai landasan pacu AR.drone memiliki ID 885. Bila dijadikan biner maka nilainya adalah 1101110101. Ketika di-*generate* maka pada baris pertama memiliki bit data 11, sehingga baris pertama bernilai 01110. Baris kedua memiliki bit data 01, sehingga baris kedua bernilai 10111. Baris ketiga memiliki bit data 11, sehingga baris ketiga bernilai 01110. Baris keempat memiliki bit data 01, sehingga baris keempat bernilai 10111. Dan baris kelima memiliki bit data 01, sehingga baris kelima bernilai 10111. Bit 1 direpresentasikan dengan warna putih, dan bit 0 dengan warna hitam. Sehingga hasil *generate* ARTag dengan ID 885 adalah seperti yang ditunjukkan pada Gambar 4.



Gambar 4 ARTag dengan ID 885

2.4 Proses Pendeteksian ARTag

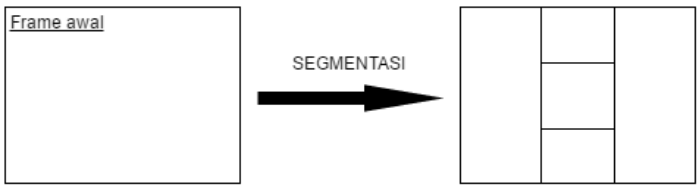
Pada proses pendeteksian ini, objek berupa ARTag ditangkap citranya oleh kamera bagian bawah AR.Drone berupa *frame*. *Frame* gambar yang ditangkap secara terus-menerus menghasilkan citra video. Citra video inilah yang digunakan untuk diproses melalui pengolahan citra digital menggunakan pustaka OpenCV. OpenCV adalah *library open-source*, yang secara luas digunakan dalam aplikasi pengolahan citra untuk mendapatkan pengakuan instan untuk aplikasi tertanam [10]. Hasilnya pengolahan citra digital mampu mengenali objek berupa ARTag yang ditangkap oleh kamera pada *drone*. Konsep ARTag hampir sama dengan deteksi landasan pacu [11].

Hasil pengenalan objek ini yang kemudian digunakan untuk melakukan kendali pada *drone.* Diagram alir pada Gambar 5 menunjukkan proses algoritma untuk mengenali objek berupa ARTag.

Gambar 5 Diagram alir algoritma pendeteksian ARTag

2.5 Proses Pendaratan AR.Drone di Atas ARTag

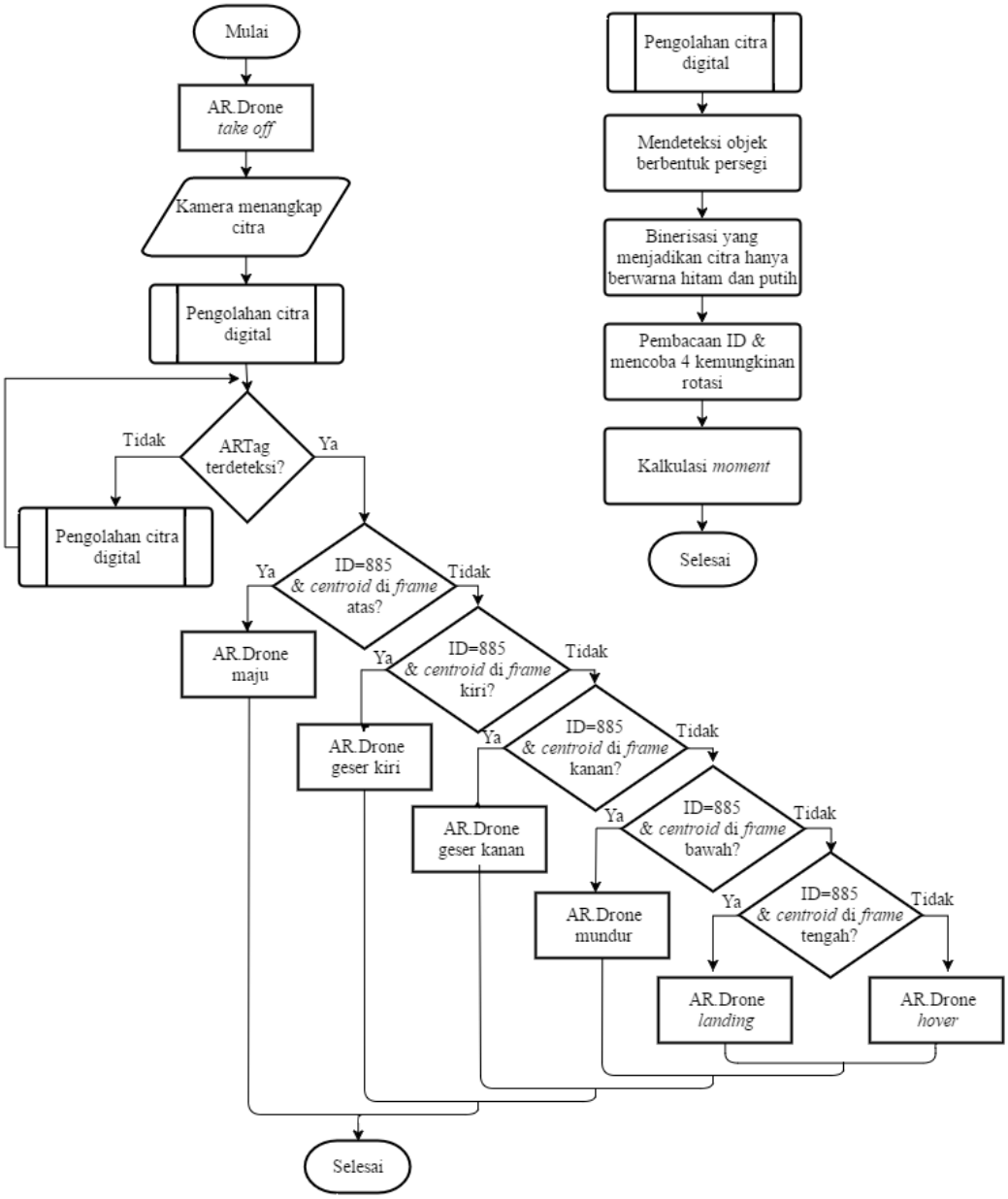
Setelah sistem berhasil mengenali objek berupa ARTag yang ditangkap pada citra video dari kamera AR.Drone, maka dilanjutkan dengan proses pendaratan oleh AR.Drone. Proses pendaratan AR.Drone dimulai dengan pembagian *frame* utuh video menjadi lima area, yaitu *frame* atas, *frame* tengah, *frame* bawah, *frame* kiri, dan *frame* kanan. Seperti yang ditunjukkan pada Gambar 6.



Gambar 6 Pembagian *frame* video

ARTag yang telah terdeteksi ditentukan *centroid*-nya menggunakan metode *image moments*. *Centroid* inilah yang menjadi acuan pergerakan *drone*. Agar *drone* menyesuaikan dirinya tepat di atas ARTag maka *centroid* harus berada pada *frame* tengah. Jika *centroid* berada pada *frame* bagian atas maka perintah kendali gerak yang

diberikan pada *drone* adalah maju. Begitu pula jika *centroid* berada pada *frame* bagian kiri, kanan, dan bawah, perintah yang diberikan adalah geser kiri, geser kanan, dan mundur. Jika *centroid* sudah berada pada *frame* bagian tengah dan ARTag memiliki ID 885 maka perintah yang diberikan adalah *landing*. Diagram alir proses pendaratan ditunjukkan oleh Gambar 7.



Gambar 7 Diagram alir misi pendaratan

3. HASIL DAN PEMBAHASAN

Pada bab ini dijelaskan mengenai pengujian sistem dan pembahasan terhadap hasil masing-masing pengujian yang telah dilakukan. Ada lima jenis pengujian pada penelitian ini. Pengujian tersebut antara lain: pengujian waktu komputasi program pendeteksian ARTag pada kondisi statis dimana drone digerakkan secara manual oleh operator tanpa harus menghidupkan propeller drone, dan dinamis dimana drone dikendalikan secara otomatis oleh sistem ROS, pengujian pendeteksian ARTag terhadap ketinggian AR.Drone pada kondisi statis dan dinamis, pengujian pendeteksian terhadap derajat kemiringan antara AR.Drone dengan ARTag, pengujian pendeteksian terhadap intensitas cahaya, dan pengujian penyesuaian pola gerak

AR.Drone dalam misi pendaratan. Kelima pengujian tersebut diharapkan telah mencakup seluruh hasil penelitian ini.

3.1 Pengujian Waktu Komputasi Program Pendeteksian

Pengujian waktu komputasi program pendeteksian ARTag dilakukan pada dua kondisi, yaitu statis dan dinamis. Pengujian ini dimaksudkan untuk mengetahui respon kamera bawah AR.Drone terhadap dua kondisi tersebut. Cara yang digunakan untuk menghitung waktu komputasi pendeteksian adalah dengan menghitung selisih waktu saat awal proses pendeteksian hingga waktu saat program telah dapat mendeteksi. Satuan waktu yang digunakan adalah detik (sekon). Pengujian ini menghasilkan informasi mengenai waktu komputasi program saat mendeteksi ARTag dengan menggunakan metode yang diterapkan pada penelitian ini. Pengujian dilakukan pada perangkat PC dengan spesifikasi *processor* Intel Core i3-2350 M yang memiliki kecepatan 2.30 GHz dan RAM sebesar 4 GB.

Posisi *drone* pada pengujian waktu komputasi ini adalah 150 cm di atas permukaan datar dengan data yang diambil sebanyak 1000 data kemudian dilakukan perhitungan rata-rata untuk diketahui selisih nilai ketika *drone* berada pada kondisi statis dan dinamis. Data hasil pengujian waktu komputasi program deteksi dapat dilihat pada Tabel 1.

Tabel 1 Data waktu komputasi program deteksi

No.	Waktu komputasi program deteksi ARTag dalam kondisi statis (dalam detik)	Waktu komputasi program deteksi ARTag dalam kondisi dinamis (dalam detik)
1	0.00811696	0.00655961
2	0.00376534	0.00417972
3	0.00276136	0.00653744
.	.	.
.	.	.
999	0.004498	0.00237155
1000	0.00584817	0.00234985
Rata-rata	**0.005893626**	**0.004771953**

3.2 Pengujian Performa Pendeteksian ARTag Terhadap Ketinggian AR.Drone

Pengujian performa pendeteksian ARTag terhadap ketinggian AR.Drone dilakukan agar diketahui tinggi optimal sistem dapat mendeteksi ARTag yang nantinya akan dijadikan sebagai landasan. Pengujian ini juga dilakukan dengan dua kondisi, yaitu statis dan dinamis. Kondisi statis dimana drone digerakan secara manual dengan bantuan operator sedangkan kondisi dinamis dimana drone terbang secara otomatis, tanpa ada campur tangan operator.

Tabel 2 Data performa pendeteksian terhadap ketinggian

Ketinggian	Tingkat keberhasilan pendeteksian pada kondisi statis	Tingkat keberhasilan pendeteksian pada kondisi dinamis
90 cm	100.00%	71.69%
120 cm	100.00%	79.70%
150 cm	99.90%	100.00%
180 cm	90.90%	97.40%
210 cm	51.40%	68.50%
240 cm	39.90%	39.59%
Rata-rata	**80.35%**	**76.15%**

Pengujian ini dilakukan dengan cara menerbangkan drone kemudian dilakukan pendeteksian ARTag pada ketinggian tertentu kemudian dilakukan pendeteksian ARTag sebanyak 1.000

frame secara real time. Performa keberhasilan pendeteksian ditentukan dengan melihat jumlah frame yang berhasil mendeteksi keberadaan ARTag dari 1.000 frame tersebut. Data hasil pengujian performa sistem pendeteksian ARTag terhadap ketinggian dapat dilihat pada Tabel 2.

3.3 Pengujian Performa Pendeteksian ARTag Terhadap Sudut

Pengujian performa pendeteksian terhadap sudut kemiringan antara *drone* dengan objek berupa ARTag dilakukan untuk mengetahui seberapa efektif sistem dapat mendeteksi objek terhadap kemiringan objek. Alat ukur yang digunakan adalah berupa satu buah busur untuk memvariasikan kemiringan objek. Prosedur pengujian ini adalah *drone* diposisikan dengan ketinggian 150 cm di atas objek, kemudian objek yang terletak tepat di bawahnya divariasikan sudut kemiringannya terhadap *drone*. Kemudian dilakukan pendeteksian sebanyak 1.000 *frame* secara *real time*. Performa keberhasilan pendeteksian ditentukan dengan melihat jumlah *frame* yang berhasil mendeteksi objek dari 1.000 *frame* tersebut. Data hasil pengujian performa sistem pendeteksian ARTag terhadap sudut kemiringan dapat dilihat pada Tabel 3.

Tabel 3 Data performa pendeteksian terhadap sudut

Sudut kemiringan	Keberhasilan
15°	0.80%
30°	72.10%
45°	99.90%
60°	100%
75°	99.90%
90°	100%

3.4 Pengujian Performa Pendeteksian ARTag Terhadap Intensitas Cahaya

Pengujian performa pendeteksian terhadap intensitas cahaya dilakukan untuk mengetahui seberapa efektif sistem dapat mendeteksi ARTag terhadap variasi intensitas cahaya pada saat pendeteksian. Alat ukur intensitas cahaya yang digunakan adalah *Light Meter* Lutron LX-100. Prosedur pengujian ini adalah dengan memposisikan AR.Drone pada jarak vertikal 150 cm di atas ARTag. Kemudian dilakukan pendeteksian ARTag sebanyak 1.000 *frame* secara *real time* pada intensitas cahaya yang berbeda. Performa keberhasilan pendeteksian ditentukan dengan melihat jumlah *frame* yang berhasil mendeteksi keberadaan ARTag dari 1.000 *frame* tersebut. Data hasil pengujian performa sistem pendeteksian ARTag terhadap variasi intensitas cahaya dapat dilihat pada Tabel 4.

Tabel 4 Data performa pendeteksian terhadap intensitas cahaya

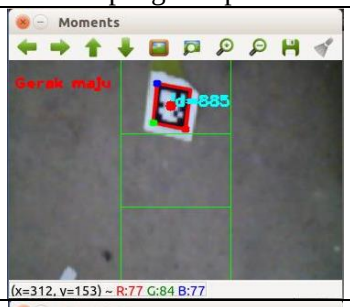
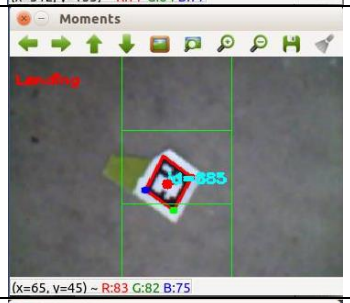
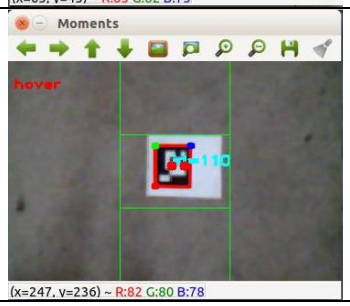
Intensitas cahaya (dalam lux)	Keberhasilan
4	51.40%
224	99.80%
595	99.90%
1555	100%
17340	99.30%

3.5 Pengujian Penyesuaian Pola Gerak AR.Drone

Pengujian ini dilakukan dengan menggabungkan metode pendeteksian dengan *image moments* untuk menentukan lokasi titik *centroid* pada objek. Kemudian dari acuan *centroid* tersebut dapat dikelompokkan gerakan-gerakan perintah pada AR.Drone untuk melakukan misi seperti gerak maju, mundur, geser kanan, geser kiri, *hover*, dan *landing* berdasarkan keberadaan lokasi titik *centroid* tersebut pada *frame* yang telah tersegmentasi. Pengujian ini dilakukan untuk

mengetahui respon sistem terhadap lokasi objek pada *frame* dan memberi perintah gerak pada *drone* dengan satu atau lebih objek. Prosedur pengujian ini adalah dengan memposisikan *drone* pada ketinggian 150 cm kemudian memindahkan objek di bawahnya pada setiap bagian *frame*. Data gerak yang dihasilkan ditunjukkan pada Tabel 5.

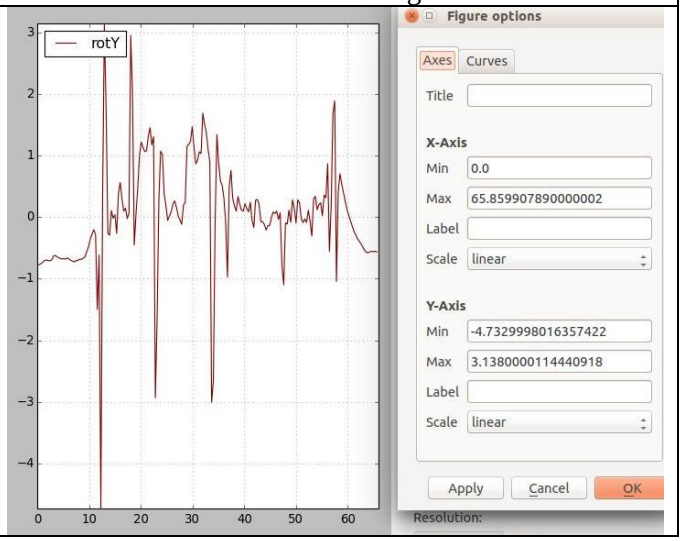
Tabel 5 Perintah gerak AR.Drone berdasarkan lokasi titik *centroid*

Gerakan	Hasil pengelompokkan
Gerak maju	
Landing	
Hover	

Mengacu pada Tabel 5, sistem akan memberi perintah gerak maju, *landing*, dan yang lainnya jika ARTag yang terdeteksi memiliki ID dengan nomor 885. Jika ID yang terdeteksi bukan 885, maka dibagian *frame* manapun ARTag terdeteksi sistem akan memberi perintah *hover*. Hal ini dimaksudkan agar AR.Drone dapat mendarat pada ARTag dengan nomor ID tertentu, bukan sembarang ID.

Pengujian penyesuaian pola gerak AR.Drone dilanjutkan dengan misi pendaratan. Pengujian ini dilakukan agar diketahui persentase keberhasilan sistem dapat mendeteksi ARTag kemudian melakukan pendaratan tepat di atasnya secara *autonomous*. Prosedur pengujian ini adalah dengan cara menerbangkan *drone* secara manual menggunakan *keyboard* pada PC dan menyesuaikan ketinggiannya pada 150 cm dan menempatkan objek pada *frame* bagian atas, bawah, kiri, dan kanan secara acak, kemudian dijalankan program otomatis agar *drone* menyesuaikan agar ARTag berada pada *frame* bagian tengah dan melakukan pendaratan secara *autonomous*. Hasilnya *drone* berhasil melakukan misi melakukan pendaratan sebanyak 7 kali dari 10 kali percobaan. Hasil yang didapatkan dari pengujian ini adalah grafik `navdata` selama proses penyesuaian posisi *drone* agar tepat di atas objek hingga mendarat yang ditampilkan dalam bentuk grafik yang ditampilkan pada rqt_bag. Salah satu `navdata` yang direkam pada proses pengujian misi pendaratan ditunjukkan pada Tabel 6.

Tabel 6 Data navigasi dalam misi pendaratan

Jenis data	Grafik data navigasi
Data kemiringan *pitch* (rotY) (detik, derajat)	

Mengacu pada Tabel 6, pada data grafik *pitch* (Y) terjadi nilai naik-turun dari *set point* 0°. Ketika garis grafik berada pada nilai positif, maka keadaan *drone* adalah melakukan gerakan maju, sedang nilai negatif adalah keadaan *drone* melakukan gerak mundur. Grafik pada Tabel 6 merepresentasikan gerak maju dan gerak mundur AR.Drone selama melakukan misi pendaratan dari tinggal landas hingga mendarat.

4. KESIMPULAN

Berdasarkan penelitian yang dilakukan ini telah berhasil dirancang bangun sebuah sistem pendeteksian ARTag sebagai landasan pacu AR.Drone menggunakan *platform Robot Operating System*, dapat diambil beberapa kesimpulan sebagai berikut:

1. Jarak atau ketinggian optimal pendeteksian dalam keadaan statis dan dinamis adalah 150 cm.
2. Memiliki kehandalan yang baik terhadap variasi intensitas cahaya. Pada tingkat intensitas cahaya 4 lux sistem masih dapat mendeteksi hingga 51.40%. Sedangkan pada tingkat intensitas cahaya 224 lux hingga 17.340 lux sistem dapat mendeteksi dengan tingkat keberhasilan 99.30% hingga 100%.
3. Tingkat keberhasilan sistem dapat mendeteksi ARTag dan AR.Drone melakukan pendaratan di atasnya adalah 70%.

5. SARAN

Berikut saran-saran yang dapat dilakukan untuk pengembangan dalam penelitian selanjutnya yang sejenis.

1. Penggunaan sistem kendali yang dapat melengkapi sistem gerak AR.Drone.
2. Penggabungan dengan metode *tracking*, agar AR.Drone dapat mendeteksi objek dari objek tidak terlihat pada jangkauan kamera bawah.
3. Maksimalkan penggunaan fasilitas yang tersedia pada *Robot Operating System*, seperti fasilitas simulator.

UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih kepada semua pihak yang telah membantu dalam mengerjakan penelitian ini.

DAFTAR PUSTAKA

[1] Ritya L., Andriana G., Indah G., 2005, Pengendalian AR Drone 2.0 dan Pengambilan Data Citra Berdasarkan Koordinat GPS, Universitas Telkom, Bandung.

[2] Nagataries, D., Hardiristanto, S., Purnomo, M.H., 2012, Deteksi Objek pada Citra Digital Menggunakan Algoritma Genetika untuk Studi Kasus Sel Sabit.

[3] Fiala, M. (2005) ARTag, a fiducial marker system using digital techniques. *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2005*. [Online] 590-596. Available from: doi:10.1109/CVPR.2005.74.

[4] Ataka, A., 2014, ROS, Satu Langkah Menuju Pemrograman Robot yang Lebih Mudah, *http://ieeesb.ft.ugm.ac.id/ros-satu-langkah-menuju-pemrograman-robot-yang-lebih-mudah/*, diakses pada 6 November 2015.

[5] Weickert, J., 2001, *Image Processing and Computer Vision*. [Online]. Available from: http://books.google.com/books?id=eSu5I9pU3rUC&pgis=1.

[6] Culjak, I., Abram, D., Pribanic, T., Dzapo, H. dan Cifrek, M., 2012, A brief introduction to OpenCV, *MIPRO, 2012 Proceedings of the 35th International Convention*, Croatia, May 21-25.

[7] Woodall W., 2015, ROS, Image transport, *http://wiki.ros.og/image_transport*, diakses pada 19 Desember 2015.

[8] Thomas D., 2013, ROS, Roscore, *http://wiki.ros.og/roscore*, diakses pada 19 Desember 2015.

[9] Salinaz, R.M., 2011, *Aruco: a minimal library for Augmented Reality applications based on OpenCV, http://www.uco.es/investiga/grupos/ava/node/26*, diakses pada 19 November 2015.

[10] Deepthi, R. dan Sankaraiah, S., 2011, Implementation of mobile platform using Qt and OpenCV for image processing applications, *Open Systems (ICOS), 2011 IEEE …*. [Online] 284–289. Available from: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6079235

[11] Ardi, M. S., Harjoko, A., dan Sumiharto, R., 2012, Purwarupa Sistem Pendeteksi Garis Landasan Pacu pada Pesawat Terbang, Indonesian Journal of Electronics and Instrumentation, Vol. 2, No. 2, Yogyakarta.