http://dx.doi.org/10.7840/kics.2016.41.10.1247

쿼드로터드론의 영상기반 자율비행연구를 위한 지상제어시스템 설계

안 희 준[◆°], 휭 꽁 앙[*], 도 딴 뚜안[*]

Design of a GCS System Supporting Vision Control of Quadrotor Drones

Heejune Ahn[◆°], Hoang C. Anh[*], Do T. Tuan[*]

요 약

소형드론의 상용화를 위해서는 안전성과 자율운행기능의 확보가 필수적이다. 최근 드론제작이 상당히 용이해졌으나, 여전히 안정적인 드론의 제작은 쉽지 않다. 따라서 자체드론제작 필요성은 영상이나 자율이동 등 상위 알고리즘의 연구에 큰 장애요소로 존재한다. 본 연구에서는 상용드론과 Raspberry PI, 및 오픈소스를 활용하여, 쿼드로터 드론의 자율운행기술 개발 중 영상기반 자율운행을 설계해볼 수 있는 지상원격제어시스템(GCS)을 설계하고 구현하였다. 설계한 시스템은 모듈화된 구성으로 통신, UI 및 영상처리 모듈로 구성하였고, 특히 주행선유지 알고리즘을 구현하여 기능 및 성능 실험을 하였다. 설계한 주행선유지 알고리즘은 Hough 변환에 의하여 검출된 차선을 소실점 검출과 자제적인 라인트래킹 알고리즘을 개발하여 사용하여 인식오류를 줄였으며, 주행선과 드론의 진행방향을 계산하고 방향 (전진, 정지, 좌우회전)제어하였다. 구현된 시스템은 현재 100m육상트랙의 직선과 완만한 곡선을 2-3 m/s로 주행할 수 있다.

Key Words : quadrotor, drone, GCS, computer vision, lane-keeping

ABSTRACT

The safety and autonomous flight function of micro UAV or drones is crucial to its commercial application. The requirement of own building stable drones is still a non-trivial obstacle for researchers that want to focus on the intelligence function, such vision and navigation algorithm. The paper present a GCS using commercial drone and hardware platforms, and open source software. The system follows modular architecture and now composed of the communication, UI, image processing. Especially, lane-keeping algorithm. are designed and verified through testing at a sports stadium. The designed lane-keeping algorithm estimates drone position and heading in the lane using Hough transform for line detection, RANSAC-vanishing point algorithm for selecting the desired lines, and tracking algorithm for stability of lines. The flight of drone is controlled by 'forward', 'stop', 'clock-rotate', and 'counter-clock rotate' commands. The present implemented system can fly straight and mild curve lane at 2-3 m/s.

※ 본 연구는 본 연구는 산업통상자원부 디자인혁신역량개발사업(10050063, 활동적으로 움직이는 사람의 고감도 다채널 추적 영상을 기록하는 비행형 드론디자인 개발)의 지원으로 수행되었습니다.
◆° First and Corresponding Author : SeoulTech, Dept. of Electrical & Information Eng., heejune@seoultech.ac.kr, 종신회원
* SeoulTech, Dept. of Electrical & Information Eng., anh.comdr@gmail.com, tuan.back@gmail.com
논문번호 : KICS2016-09-255, Received September 13, 2016; Revised September 24, 2016; Accepted September 27, 2016

Ⅰ. 서 론

드론은 초기에 미국 등 군사 강대국위주로 정찰용, 표적용 등 군사용 목적으로 고정익 (fixed-wing) 드론이 주로 연구되었으나, 최근에 와서는 쿼드로터와 같은 회전익 (rotating-wing) 드론을 바탕으로 하는 산업/상업용 드론에 대한 가능성과 관심이 증대되고 있다[1]. 특히, 항공촬영, 교통관제, 배송 분야에 우선적인 적용이 예상되고 있으며, 구체적으로 구글(Google), DHL, 아마존(Amazon), 도미노 피자 등이 수년 내에 드론을 배달서비스에 상용화하려고 노력 중이다. 드론의 용도와 규모가 다양하여 국내에서도 생산기술연구원, ETRI, 삼성전자에서부터 중소 장난감업체까지 다양한 업체에서 관심을 갖고 개발 중이다.

최근 IT 분야에서 쿼드로터 드론에 관심을 두는 원인은, 수직 이착륙 및 호버링이 가능하다는 장점과 함께, 상대적으로 전문적인 항공역학적인 지식이 요구되지 않으며, 저가의 센서를 구하기 쉬워졌고, 관련 오픈 HW와 SW등이 공개되어, 제작과 개발이 용이해진점을 들 수 있다[2]. 이런 추세에서 전자 IT 공학적 관점에서 드론을 일종의 날아다니는 로봇으로도 생각할 수도 있게 되었다. 일반적으로 로봇은 다양한 연구분야가 융합된 응용시스템으로, 소프트웨어 및 자율 및 인공지능기술이 중요한 연구대상으로, 드론 상용화에도 자율운행 기술이 중요한 위치를 차지하게 되었다. 그러나 전자공학 및 컴퓨터 공학을 전공한 저자들이 수차례 제작을 해본 바로는 안정적인 제어를 하는 드론을 만든다는 것은 여전히 어려운 작업이다. 따라서 통신, 영상처리, 인공지능 등의 전문연구 분야를 갖는 연구자들이 본인의 아이디어를 실현해볼 수 있는 개발환경을 구축하는 것이 필요하다. 본 연구에서는 현재 상용화된 드론과 보드를 활용하고, 널리 사용되는 오픈소스 프로그램 환경을 사용하여 통신 및 자율주행 연구를 할 수 있는 GCS (Ground Control System)시스템 설계를 수행하였다. 특히, 개발된 시스템의 동작을 확인하기 위하여 실제 자율운행기술을 개발하였다.

여기서 우선적으로 사용한 드론은 Parrot사의 제품으로 연구자들 사이에서 드론을 제어하기 위한 API가 분석되어 일부 공개된 상황이다. 문헌상에 Parrot 사의 제품들을 활용하거나 분석한 연구들을 다수 찾을 수 있다[3-5].본 연구에서는 드론에서 전송되는 영상을 사용하여 주행선 유지하는 기능을 시험적으로 설계 구현 하였다. 주행선 유지 기능을 선택한 이유는, 드론상용에 가장 우선적인 요소가 '안전성 확보'라고 볼 수 있으며, 드론에 자동차에서 적용되는 안전보조장치 (ADAS: Automatic Driving Assistant System) 기술[6]을 적용하고 가능성과 보완점을 찾는 것이 중요하다고 생각했기 때문이다.

본 논문의 구성은 다음과 같다. 제 2절에서 설계한 전체 GCS 시스템의 구조에 대하여 설명하고, 제 3절에서 설계한 주행선유지 알고리즘에 대하여 기술한다. 제4절에서 시스템의 테스트 결과와 성능 및 문제점에 대하여 기술하며, 제 5장에서 결론으로 본 시스템의 확장방안에 대하여 논한다.

Ⅱ. 드론 GCS 시스템의 설계

여기서는 설계한 드론 지상제어 시스템은 드론응용에 필요한 여러 가지 영상 처리알고리즘을 개발하고 이를 실험해 볼 수 있게 하려는 목적에서 설계하였다. 실험에서 사용한 드론(Parrot사의 Bebop)의 의존성을 최소화하기 위하여 모듈화하여 설계하였다.

2.1 소프트웨어 구조

드론 지상제어시스템의 시스템 구조는 Fig. 1과 같다. 크게 드론과 통신을 담당하는 프로토콜부, 영상을 수신하고 영상 처리를 하는 영상처리부, 영상처리 결과를 바탕으로 드론시스템을 제어하는 주제어부로 구성하였다. 각 모듈별로 실시간적 동시성을 만족하기 위하여, 각 부분 별로 쓰레드를 할당하였다. UI와 주기능을 위한 주-쓰레드, 드론과의 프로토콜통신을 담당하는 프로토콜 쓰레드, 그리고 영상처리를 담당하는 영상처리 쓰레드로 구성하였다. 영상데이터를 수신하고 디코딩하는 부분 까지는 현재 프로토콜 쓰레드에서 담당하고 있다. 영상 디코딩 하는 기능과 프로토콜 부분을 구분하는 것이 프로토콜의 응답성 향상을 위해서는 바람직하다고 고려되나, 시스템을 단순화하는 측면에서 쓰레드의 수를 최소화하기 위하여 같은 쓰레드에서 처리하도록 하였다. 실제 구성하여 테스트한 결과 (향후에 자세히 설명) 영상을 디코딩하는데 걸리는 시간은 수 (2-3) ms정도로 프로토콜 간격 25ms를 고려했을 때 실시간성에 크게 문제되지 않음을 확인하였다.

2.2 드론 제어 프로토콜 파트 설계

소형 드론제품 중 전 세계적으로 가장 시장점유율이 높은 것은 중국의 DJI사의 Phantom 계열이나, 해커나 학계에서 가장 많은 연구가 수행된 제품은 Parrot사의 제품이다[3-5]. 이러한 원인은 Parrot사 제품

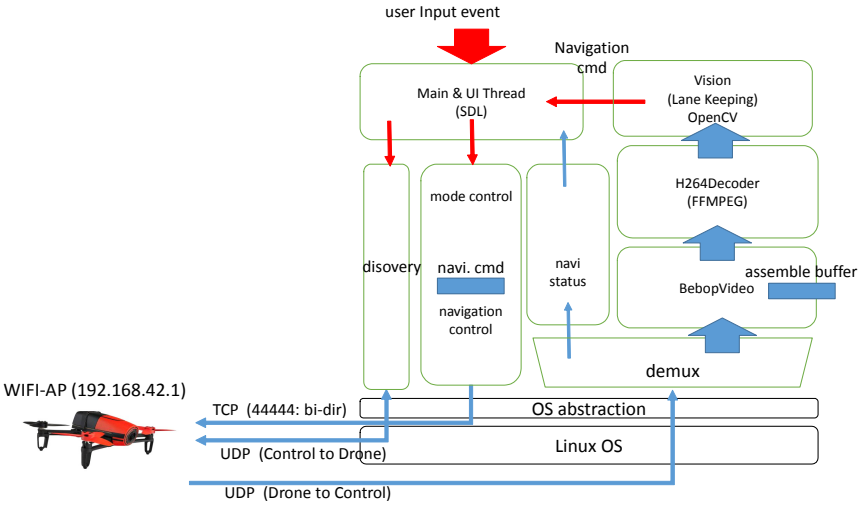


그림 1. 드론 제어 시스템의 전체 구조 및 동작 흐름
Fig. 1. Overall architecture for ground control system

이 제어기와 드론사이에 표준 통신방식 (Wifi)을 사용하고, 스마트폰 등에서 응용을 개발할 수 있는 SDK를 제공하여, 설계구조 등의 추측이 용이하고 역컴파일이나 패킷 갈무리를 통해 프로토콜을 알아낼 수 있기 때문이다. 예로서 , 기존에 연구자들과 프로그래머들이 Parrot사 드론의 프로토콜을 역공학하여 Github등에 공개한 'node-bebop'이 있으나, 본 연구에서는 이러한 정보를 바탕으로, 새롭게 C++ 기반 프로토콜 엔진 설계를 하였다. 우리가 새롭게 구현한 주요한 이유는, 기존의 소프트웨어들이 다양한 소프트웨어들을 조합하여 일관성이 떨어지며, 영상 처리하기에는 무거운 Javascript 언어기반의 node.js환경 등을 사용하고 있기 때문이다.

Fig. 2와 제어기가 드론에 통신 연결을 하는 절차를 보여준다. Bebop 드론은 제어시스템과의 통신을 위하여, 자체적으로 Wifi AP로 동작한다. 각 시스템의 고유번호를 SSID로 사용하며, 설정 디스커버리 기능을 위하여 TCP 포트 44444를 지정 번호로 사용하며, 설정내용은 json 포맷[12]으로 주고받는다. 조정을 원하는 장치는 패스워드 없이 Wifi에 접근할 수 있으며, 프로토콜도 패스워드 등 인증 절차를 요구하지 않는다. 현재 버전의 Bebop은 IPv4 "192.168.42.1"을 AP의 인터넷 주소로 사용하고, TCP:44444 번을 통신에 필요한 정보를 확보하기 위한 디스커버리 채널로 사용한다.

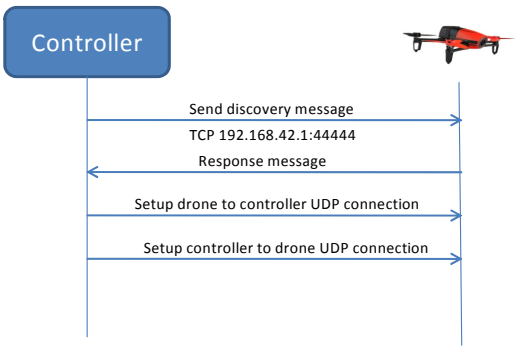


그림 2. 드론 통신 연결 및 디스커버리 절차
Fig. 2. Connection setup and discovery

Fig. 3은 드론과 GCS간의 두 가지 제어용 프로토콜 동작을 보인다. 드론의 상태정보와 이동동작명령는 실시간성 정보로 비신뢰적으로 연속적인 방법으로 전송한다. 반면 이착륙, 긴급모드 등 일회성이면서 신뢰성이 필요한 명령은 Stop-N-Wait 형태로 송수신한다. 실제 구현을 통하여 송수신한 데이터는 제 4장에서 제시한다. Fig. 4는 이때 사용하는 프레임의 포맷을 설명한다. 총 7바이트의 기본헤더를 통하여 다양한 프레임을 다중화(type, id)하며 세그먼테이션

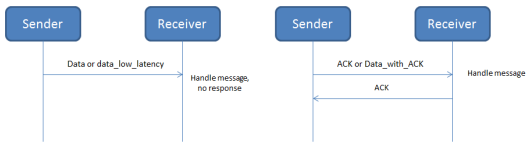


그림 3. Parrot Bebop 드론의 프로토콜 (좌측: 저 지연, 비신뢰성, 우측: 신뢰성)
Fig. 3. Parrot Bebop Drone Protocol (Left: low delay & unreliable, Right: reliable)

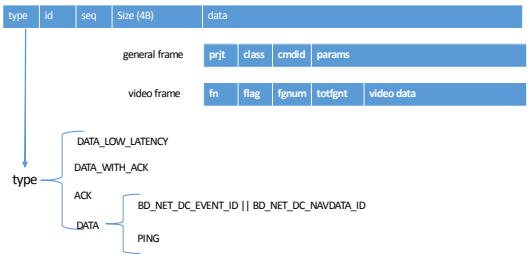


그림 4. Parrot Bebop 드론의 프로토콜 (좌측: 저 지연, 비신뢰성, 우측: 신뢰성)
Fig. 4. Parrot Bebop Drone Protocol (Left: low delay & unreliable, Right: reliable)

(length)을 지원한다.

2.3영상 스트리밍

영상 데이터를 제외한 모든 프레임은 컨트롤 모듈에서 직접 처리하며, 비디오의 경우는 프레임 포맷과 에러제어 방식을 위하여 별도의 구현이 수행된다. 영상 스트림은 실시간성을 위하여 Bebop은 H.264포맷으로 30fps로 전송되며 영상 속도는 약 300kbps로 30화면마다 갱신용 I 프레임을 포함한다. 영상데이터를 처리하는 과정은 앞서 전체 소프트웨어 구성도와 같이 크게 3개의 단계를 거친다.

2.3.1 영상 프레임 전송

하나의 영상 프레임은 일반적으로 UDP 세그먼트보다 커서, 여러 개로 나누어져서 전송되며, 프레임 번호와 fragment 번호를 사용하여 SR (Selective Repeat) 방식의 오류제어를 수행한다. 재전송이 반복되면 되면, 실시간성을 위하여 해당 프레임의 전송의 포기하는데, 이때는 수신측에서는 I 프레임(플래그가 표시)을 사용하여 디코딩을 재동기화에 사용할 수 있다.

2.3.2 영상 디코딩

이렇게 디코딩된 데이터는 FFMPEG 라이브러리를 사용하여 복원한다. FFMPEG 라이브러리는 자체적으로 다중 코어를 사용하여 시스템 자원을 효과적으로 사용하며, 따라서 실험에 사용한 Raspberry PI 3나 랩톱에서 실시간 처리 (해당 영상을 2-3ms에 디코딩)가 가능하다.

2.3.3 영상 처리

영상 처리과정은 일반적으로 계산량이 많이 필요한 과정이다. 본 연구에서는 개발의 편이성을 위하여 공개 소프트웨어인 OpenCV (버전 3.1)을 사용하였다. 알고리즘에 따라 다르겠으나 일반적으로 30fps 정도의 속도로 내비게이션 알고리즘을 동작시키는 것은 임베디드 환경 뿐 아니라 랩톱과 같은 환경에서도 쉽지 않을 일이다. 따라서 복호화된 모든 영상을 처리하는 대신 영상 처리가 실시간이 될 수 있도록 흐름제어를 하는 것이 필요하다. 이처럼 비동기적으로 동작하는 경우를, 생산자-소비자 쓰레드 모델로 쓰레드 간에 버퍼를 사용하고 세마포를 이용하여 동기화 하는 것이다. 그러나 본 응용의 경우에는 생산 빈도가 소비 빈도에 비하여 2-3배 이상 높으며 실시간 성 요구 때문에 버퍼에 저장하는 것이 크게 효율이 없기 때문에 버퍼는 하나만을 사용하여 구성을 단순화 하였다.

2.4 UI 설계

현재 UI 인터페이스 (Fig. 5)는 최소한의 기능만을 구형하였다. SDL (Simple DirectMedia Layer) 라이브러리를 사용하였는데. 그 이유는 SDL이 윈도우, 리눅스 환경에서 호환적이며, 기능이 적은 대신 프로그램이 간단하여 쉽게 배울 수 있기 때문이다. 드론의 모드는 모드를 구분하지 않고, 기본적으로 자율동작 모드로 움직이며, 매뉴얼 명령은 언제든지 수행할 수 있다. 언제든 비상시에는 사용자가 개입하여 시스템을 정지시키기나 비상정지 시킬 수 있도록 하였다. 단 현재는 운행 속도 등 파라미터는 사전에 세팅하고, UI에서 설정을 포함하지는 않았다.

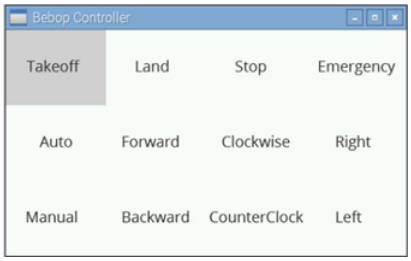


그림 5. GCS UI 및 메뉴구성
Fig. 5. GCS UI and menus

Ⅲ. 주행선 유지 알고리즘

본 연구는 영상제어가 가능한 GCS 시스템을 구성하는 것이 목적이나, 설계가 충분히 의미가 있음을 확인하기 위하여 구체적인 응용인 주행선 유지 알고리즘을 구현하였다. LDWS (Lane departure warning System)은 자동차에서 운전자 안전 보조기능으로 활발히 연구되고 있는 알고리즘이다 [6]. 따라서 기존의 차량용 알고리즘을 기초로 드론에 적용하고 차이를 살펴보았다. 차량용 LDWS 알고리즘과 근본적인 차

이는 없다고 볼 수 없으나, 주행선 이탈을 경고하는데 그치지 않고, 조향을 제어하여 직접 자율 비행 한다는 점과, 상대적으로 높은 고도에서 주행선을 내려다보고 동작함으로써 주변경계물들이 영상에 많이 포함되어 이를 처리해야한다는 점등이 차이점이다.

주행선 유지 알고리즘은 영상처리와 드론제어로 구분한다. Fig. 6은 주행선 유지 알고리즘의 전반적인 절차를 보여 주고 있으며, 기본적으로 OpenCV 3.1 API와 직접구현을 통하여 구현하였다. 각 단계별 설계는 다음과 같다.

1) ROI 영상 추출: 복호화된 영상 (해상도: 수평 640, 수직 368픽셀)은 이미지단위로 처리 한다. 우선 컬러정보는 사용하지 않으므로, RGB 신호(Fig. 7 (a))을 밝기신호로 변환한다. 이때 원 영상에 Hough변환을 직접 적용하면, 건물의 직선부나 노이즈에 의한 선분 등 주행 라인 외의 선분들이 다수 포함 된다 [7]. 불필요한 라인들을 제거하기 위하여, 주행영역에 해당하는 부분 (수직방향으로 아랫부분)에 ROI를 정의하고 ROI 영역만을 처리하게 된다. 본 연구에서는 실험적으로 얻어진 하위 1/2영역을 사용하였다 (Fig. 7 (b)).

2) 초벌 라인 검출: Canny 에지알고리즘을 통하여 라인의 경계 픽셀을 검출한 후, 이들로 구성된 Hough 변환을 사용한다. 알고리즘에 사용한 변수들은 모두 사전에 획득한 영상을 바탕으로 실험적으로 결정하였다. Cany Edge에 사용된 경계치 값은(theta = CV_PI/180, threshold = 70, minLineLen = 10, maxLineGap = 10)을 사용하고, Hough 변환에 사용한 경계치 값은 (threshold1 = 180, threshold2 = 120, apertureSize = 3) 을 사용하였다 (Fig. 7. (c))

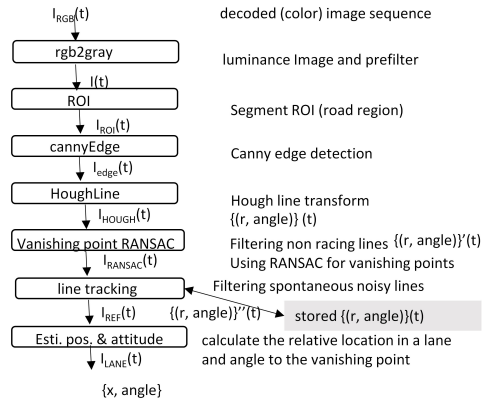
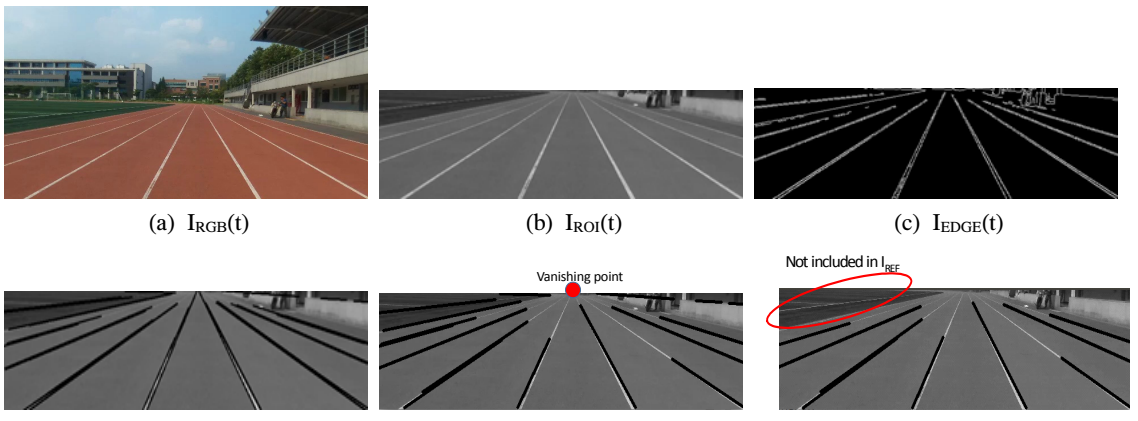


그림 6. 차선유지를 위한 영상처리 절차
Fig. 6. Image processing for lane-keeping

3) 라인 정제: ROI가 적절한 경우에도, 초벌 라인들은 주행선이외에 다양한 노이즈가 섞여있다. 이를 해결하기 위하여 두 가지 알고리즘을 적용하였다. 우선, 관심 라인들이 직선이라는 가정에서 (곡선라인 제외) 모두 소실점에 연결된다. 따라서 공통적인 소실점을 갖지 않는 선분들은 주행라인일 가능성이 떨어진다. RANSAC 알고리즘[8]을 적용하여 공통소실점을 갖는 선분들만을 추출하였다. 또한 레이싱 라인이 굵어서 두 라인이 찾아지는 경우가 발생하는데, 이 중 복된 Hough 라인을 제거하기 위하여 간격이 일정 픽셀 이하인 경우 강한 선분만 남기고 제거하였다 (Fig 7 (e)).

4) 라인 추적: 하나의 이미지가 아니가, 영상 (이미지 시퀀스)을 처리하려면, 각 이미지처리에서 나온 결과를 종합적으로 고려하여 대상을 추적하는 방법이 필요하다. 이에 대하여 Particle 필터나, Kalman필터



(a) $I_{RGB}(t)$ (b) $I_{ROI}(t)$ (c) $I_{EDGE}(t)$

(d) $I_{Hough}(t)$ (e) $I_{RANSAC}(t)$ (f) $I_{REF}(t)$

그림 7. 영상처리 결과 단계별 출력
Fig. 7. Image processing result example

방식은 시스템의 모델링이 필요하고 계산량도 증가하는 점 등이 있다. 본 연구에서는 추정된 선분결과를 기록하고, 새롭게 추정된 선분과 비교하여 일정한 거리 안에 있는 선분을 매칭(Fig. 8)하는 방식을 사용하였다 (Fig. 7 (f)).

5) 드론 위치 및 방향 추정: 추출된 라인들은 비행체의 시점이 영상의 중심이라는 가정에서, 순차적으로 해당하는 라인을 부여할 수 있다. 추적된 주행라인들을 바탕으로 중앙에 근접한 좌측, 우측 라인을 선정하여 현재 드론의 주행선을 결정하고, 이를 바탕으로 드론이 현재 주행선에서 상대적인 위치를 계산한다. 가장 아래쪽 픽셀라인의 절단점인 (x_left, x, right)에서 x_center 를 뺀 (|x_left- x_center|, |x, right -x_center|)가 좌측 라인과 우측라인까지의 거리가 된다. 소실점과의 각도를 고려하여 Fig. 9과 같이 계산할 수 있다.

이러한 영상처리 결과를 통한 위치 추정을 바탕으로 주행선의 일정 범위 안에 위치하는 경우는 '전진' 명명을, 범위를 넘어서게 되면 '정지'- '회전'명령을 송신한다.

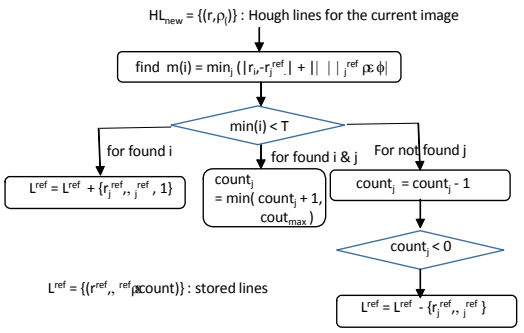


그림 8. 라인 추적 알고리즘 (참고 라인과 라인매칭사용)
Fig. 8. Line tracking algorithm using line history and line matching

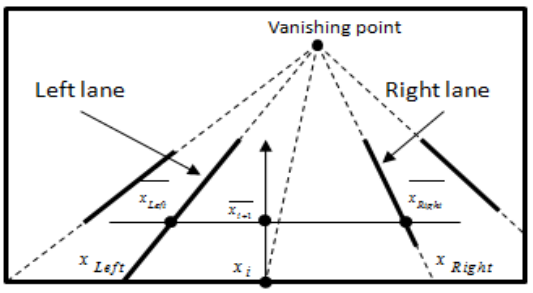


그림 9. 드론 위치 및 방향 예측
Fig. 9. Drone position and heading estimation

Ⅳ. 실 험

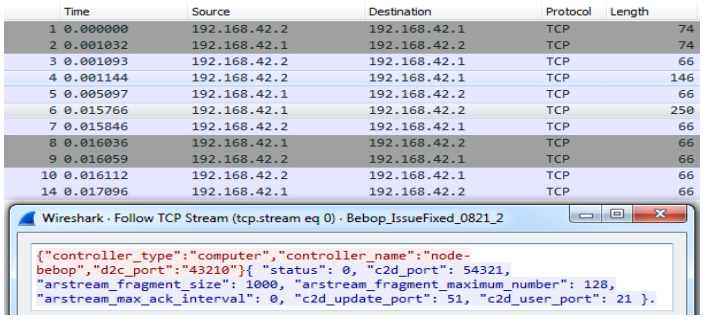
앞서 설계한 내용은 Raspberry PI3에 구현하여 적색 우레탄에 흰색라인이 그려진 육상 경기장의 트랙에서 대상으로 실험하였다. 육상 트랙 레인의 폭은 125cm로 차량용인 3-3.5m에 비하여 1/3정도로 좁아 상대적으로 어려우나, 선분이 실선인 점은 영상처리가 용이한 점이라고 볼 수 있다.

4.1 통신부 시험

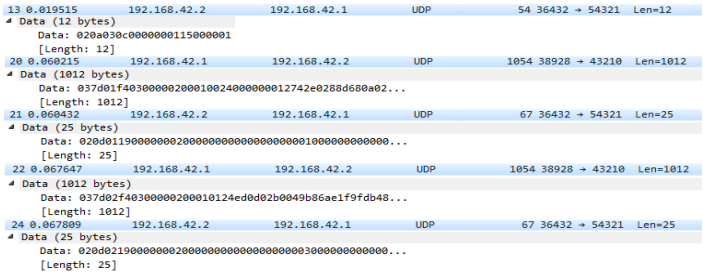
자율비행은 실시간성이 요구되므로 통신부의 동작은 프로토콜 정합성뿐 아니라 타이밍준수가 매우중요하다. 이를 확인하기 위하여 자체로그와 Wireshark를 통한 방법을 병행하였으나, 논문에서는 Wireshark 로그 결과로 설명한다. 이동명령은 해당 드론의 SDK 매뉴얼에 있는 규정에 20-500ms로 되어 있으며, 드론은 내부적으로 소프트 상태 머신을 사용하는 것으로 추정할 수 있다. 즉, 500ms 이 지나서 새로운 명령이 오지 않으면 현재 상태를 유지한다. 이 20ms 단위로 전송하기 때문에 제어속도는 20ms 라고 볼 수 있다. Fig. 10과 Table 1은 Wireshark사용하여 주고받은 메시지를 수집한 결과와 이를 메시지별로 분류한 설명이다. 첫 번째 로그는 연동을 위하여 WiFi 연동 후 디스커버리 통신을 하는 내용으로, JSON 포맷[12]으로 GCS는 43210을 드론은 54321을 수신 UDP 포트

표 1. 프로토콜 로그 상세 설명
Table 1. Protocol logs description

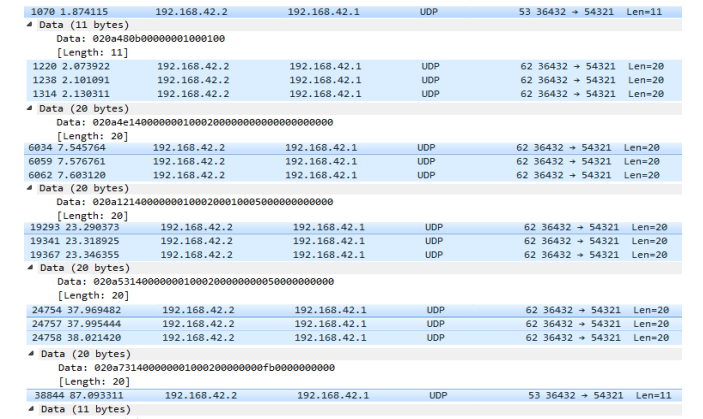
Message number	Direction	Description
1-10, 14	GCS↔ Drone	Discovery TCP packet
13	GCS→ Drone	Video streaming enable
20, 22	Drone→ GCS	Video data
21, 24	GCS→ Drone	Video data received ACK
1070	GCS→ Drone	Take off command
1020,1238,1314	GCS→ Drone	Stop commands (25ms interval)
6034, 6059, 6062	GCS→ Drone	Forward commands (25 ms interval)
19293, 19341, 19367	GCS→ Drone	Clockwise rotation commands (25ms interval)
24754, 24757, 24758	GCS→ Drone	Counter-Clockwise rotation commands (25 interval)
38844	GCS→ Drone	Land command



(a) discovery protocol log



(b) video streaming log



(c) control message log

그림 10. Parrot Bebop 드론의 프로토콜 데이터 로그 (좌측: 저 지연, 중앙: 비신뢰성, 우측: 신뢰성)
Fig. 10. Drone protocol trace (top: discovery, middle: video streaming, bottom: control and status report)

로 사용한다는 정보를 교환한다. 두 번째 는 영상 스트림과 오류제어용 ACK이 오가는 로그로, 데이터양이 많아 인터벌이 매우 짧다. 세 번째 는 25ms마다 실시간성 제어 및 상태 명령이 송수신 되는 것을 확인할 수 있다.

4.2영상 처리 및 제어 실험

일반적인 처리 결과는 앞서 설명하였으므로, 구현 후 실험에서 나온 수행결과에 대해 기술한다. Raspberry PI3 환경에서 영상처리 각 단계의 실행시간은 Table 2와 같다. 하나의 프레임의 영상처리시간은 약 250ms 정도로 1초당 4개의 처리가 가능하다. 영상의 크기를 줄이면 15 렌 정도가 가능하나 영상 처리 결과가 만족스럽지 못하여 원해상도를 사용하였다. Hough처리가 가장 큰 시간이 소요되는데, OpenCV는 현재 단일 쓰레드활용 만을 하고 있어, 향후 4 core를 활용한 알고리즘[9]이나 GPU를 활용할 필요가 있다. 영상 처리 주기에 비하여 고속으로 비행하는 경우 현

표 2. 영상처리 단계별 수행 시간 (RPI3)
Table 2. Execution time for each step for image processing in RPI3

Processing	Execution time (RPi 3, ms)
Video decoding	25.685
CannyEdge	31.167
HoughLine	170.4
Vanishing point filter (RANSAC)	1.665
Tracking	2.292
command to drone reaction	0.194

재 주행하는 라인을 추적하는 것이 불가능하다. 따라서 영상 처리 속도에 따른 드론의 비행속도를 제한하여야하며, 본실험에서는 2-3 m/s를 설정하여 동작하도록 하였다.

마지막으로, 이러한 구현을 종합적으로 주행 기능과 성능을 확인할 수 있는 그래프를 Fig. 11에 제시하였다. 연구의 최종 목적은 육상 경주 트랙을 완주하는 것으로 하였으나, 곡선 주로에 대한 영상 처리 알고리즘의 성능 안정적이지 못하여, 본연구의 결과는 직선 주로에서의 결과로 제한하였다. Snake 알고리즘[10] 등을 사용하여 곡선차선을 모델링하는 것이 추후 사항이다. 수평측은 이동 시간이고, 수직측은 이동 중의 드론의 주행선 안에서의 위치와, 소실점과의 각도를 보여준다. 목표한 바와 같이 선 안을 따라 움직이고 있음 을 확인할 수 있다. 주행 중에 두 번의 방향 수정이 일어나는 것을 알 수 있다. 이는 초기 방향각이 잘 못되어 있기 보다는 비행체의 한계와 바람 등의 외부 영향으로 볼 수 있다. 그래프에서 보는 바와 같이, 해당 문제 상황에서 정이와 방향각을 수정하여 주행선을 유지하는 목적을 만족하고 있음을 확인하였다. 정

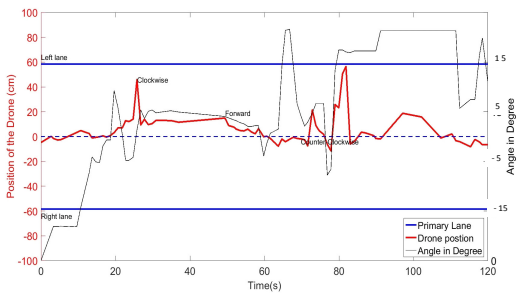


그림 11. 비행 로그 (주행 라인에서의 위치와 소실점방향과의 각도)
Fig. 11. Flight control log (position and angle from the vanishing point line)

량적인 평가와 성능 개선을 위하여, 중앙선에서 벗어나는 정도와 주행 속도를 평가 모델로 하고 알고리즘 개선작업을 하고 있다.

V. 결 론

본 논문에서 드론의 자율주행 연구를 위한 GCS 설계 제작하였다. 개발된 시스템은 성능 평가를 위한 응용 예로, 육상 트랙 경기장의 트랙을 인식하여 레인을 따라서 일주행이 가능하도록 구현하였다. 기존에 발표된 주행선 유지(lane-keeping) 알고리즘의 연구들을 바탕으로 환경에 맞게 개선하여, 직선과 완만한 곡선에서 성공적인 주행이 가능하였다. 현재 운행의 속도는 약 2m/s 정도로 저속으로 동작 시 에만 제어가 가능한 제약사항이 있다. 이는 크게 드론 장치의 제어안정성과 영상처리의 계산량에 따른 제어속도의 제약 때문인데, 현재 알고리즘 및 구현 수준의 최적화를 수행 10m/s (36kph)정도로 높이기 위한 중이다.

본 시스템을 기반으로 영상처리 인공지능 연구자들이 드론자체개발에 어려움이 없이 연구를 진행 할 수 있었으면 한다. 예를 들어 용용은 또한, 최근 국내외에서 조정방식의 드론레이싱이 스포츠 산업화[11]하는 시도가 있다. 현재는 FPV (First Person View)를 사용하여 경기자가 직접 제어하는 방식을 사용하고 있지만, 가까운 미래에 자율형 방식을 이용한 레이싱도 가능하리라고 판단되면, 여러 대의 드론을 사용한 평대주행도 가능하다고 판단된다.

이처럼 다양한 활용을 위해서는 현재 시스템을 개선을 할 필요성이 있다. 현재 개발된 시스템은 Parrot 사의 Bebop 모델과 연동하여 테스트하였고, 이로 인하여 해당 제품에 종속적인 부분들이 상당수 존재한다. 대표적으로 제어 및 영상에 사용된 프로토콜 포맷은 절대적으로 해당 제품에 따라서 구현하였다. 하지만, 시스템의 확장성을 고려하여, 각 클래스 API들은 가능한 범용 적으로 사용할 수 있도록 설계하였다. 따라서 드론시스템에 따라 제어용으로 MAVLINK 프로토콜을 사용하도록 변경이 가능하며, 현재 자체 제작한 드론을 사용하여 이를 구현 중에 있다.

References

[1] J.-H. Jin and K-.B. Lee, "The understanding and trends of UAV/Drone," *J. KICS*, vol. 33, no. 2, pp. 80-85, 2016.

[2] H, Lim, J. Park, D. Lee, and H. J. Kim,

"Build your own quadrotor: Open-source projects on unmanned aerial vehicles," *IEEE Robotics & Automation Mag.*, vol. 19, no. 3, pp. 33-45, 2012.

[3] A. Hernandez, "Identification and path following control of an AR. Drone quadrotor," in *Proc. IEEE ICSTCC*, pp. 583-588, 2013.

[4] B. Childers, "Hacking the Parrot AR drone," *Linux J.*, vol. 241, no. 1, 2014.

[5] T. Krajník, "AR-drone as a platform for robotic research and education," in *Proc. Int. Conf. Res. Edu. Robot.*, pp. 172-186, Prague, Czech Republic, Jun. 2011.

[6] H.-P. Sun, "A study on the automotive ADAS market diffusion factors," in *Proc. KICS Int. Conf. Commun.*, pp. 942-945, Jun. 2009.

[7] J. F. Liu, Y. F. Su, M. K. Ko, and P. N. Yu, "Development of a vision-based driver assistance system with lane departure warning and forward collision warning functions," in *Proc. IEEE DICTA*, pp. 480-485, Dec. 2008.

[8] J. P. Tardif, "Non-iterative approach for fast and accurate vanishing point detection," *IEEE 12th Int. Conf. Comput. Vision*, pp. 1250-1257, Sept. 2009.

[9] Y. K, Chen, W. Li, J. Li, and T. Wang, "Novel parallel hough transform on multi-core processors," in *Proc. IEEE Int. Conf. Acoust. Speech and Sign. Process.*, pp. 1457-1460. Mar. 2008.

[10] Y. Wang, e. K. Teoh, and D. Shen, "Lane detection and tracking using B-Snake," *Image and Vision Computing*, vol. 22, no. 4, pp. 269-280, 2004.

[11] D. Schneider, "Is US drone racing legal? Maaaaybe," *IEEE Spectrum*, vol. 52 no. 11, pp. 19-20, 2015.

[12] D. Crockford, *The json data interchange format*, Technical report, ECMA International, Oct. 2013.

안 희 준 (Heejune Ahn)

2000년 2월 : KAIST 전기및전자공학과 박사
1997년~2002년 : LG 전자
2002년~2003년 : Tmax soft
2004년 2월~현재 : 서울과학기술대 전기정보공학과 교수
<관심분야> 컴퓨터통신, 데이더 마이닝, 영상처리

황 꽁 앙 (Hoang C. Anh)

2013년 8월 : 베트남 하노이공대 정보통신과 졸업
2013년-2014년 : 삼성전자 하노이 모바일센터 연구원
2015년 3월~현재 : 서울과학기술대학교 전기정보공학과 석사과정
<관심분야> 드론, 영상처리, 임베디드 SW

도 딴 뚜안 (Do T. Tuan)

2013년 8월 : 베트남 하노이공대 정보통신과 졸업
2013년-2014년 : 삼성전자 하노이 모바일센터 연구원
2015년 9월~현재 : 서울과학기술대학교 전기정보공학과 석사과정
<관심분야> 드론, 영상처리, 임베디드 SW