論文 J. of The Korean Society for Aeronautical and Space Sciences 46(11), 911-920(2018)
DOI:https://doi.org/10.5139/JKSAS.2018.46.11.911
ISSN 1225-1348(print), 2287-6871(online)

Unity3D를 이용한 스트랩 다운 영상 추적기의 동역학 및 유도 법칙 알고리즘의 상호-시뮬레이션 방법에 관한 연구

마린미카엘*, 김태호**, 방효충*, 조한진***, 조영기***, 최용훈***

Study on Co-Simulation Method of Dynamics and Guidance Algorithms for Strap-Down Image Tracker Using Unity3D

Mikaél Marin*, Taeho Kim**, Hyochoong Bang*, Hanjin Cho***,
Youngki Cho*** and Yonghoon Choi***
Korea Advanced Institute of Science and Technology*,**, LIG Nex1***

ABSTRACT

In this study, we performed a study to track the angle between the guided weapon and the target by using the strap-down image seeker, and constructed a test bed that can simulate it visually. This paper describes a method to maintain high-performance feature distribution in the implementation of sparse feature tracking algorithm such as Lucas Kanade's optical flow algorithm for target tracking using image information. We have extended the feature tracking problem to the concept of feature management. To realize this, we constructed visual environment using Unity3D engine and developed image processing simulation using OpenCV. For the co-simulation, dynamic system modeling was performed with Matlab Simulink, the visual environment using Unity3D was constructed, and computer vision work using OpenCV was performed.

초 록

본 연구에서는 스트랩 다운 영상 탐색기를 활용한 유도무기와 목표물 사이의 관측각을 효과적으로 추적할 수 있는 연구를 수행하였고 이를 시각적으로 시뮬레이션 가능한 테스트 베드를 구축하였다. 영상 정보를 이용하여 목표물 추적을 위한 Lucas Kanade의 Optical flow 알고리즘과 같은 희박 특징점 추적 알고리즘 구현 시 고성능의 특징점 분포를 유지시키는 법을 기술하였으며, 특징점 추적 문제를 특징점 관리의 개념으로 확장하여 연구하였다. 이를 구현하기 위해 Unity3D 엔진을 이용하여 시각 환경을 구성하고 OpenCV를 이용하여 영상 처리 시뮬레이션을 개발하였다. 상호-시뮬레이션을 위해 매틀랩(Matlab) 시뮬링크(Simulink)로 동적 시스템 모델링을 하였고, Unity3D를 이용한 시각 환경을 구성, OpenCV를 이용한 컴퓨터 비전 작업을 수행하였다.

Key Words : Feature Management(특징점 관리), Optical Flow(옵티컬 플로우), Outlier Rejection(Outlier 제거), Strap-Down Image Tracker(스트랩 다운 영상 추적기), Unity3D(유니티3D)

† Received : July 4, 2018 Revised : October 19, 2018 Accepted : October 19, 2018
** Corresponding author, E-mail : thkim@ascl.kaist.ac.kr

Ⅰ. 서 론

본 연구에서 대상으로 하는 초소형 스마트탄의 경우 탑재공간의 제약이 크기 때문에 스트랩 다운 영상탐색기를 단독으로 탑재하거나 스트랩 다운 영상탐색기에 1축 Roll Rate Gyro만을 기반으로 하는 최소한의 센서 조합만을 탑재할 수 있다. 이러한 센서 조합을 가정한다면 중기 유도(Mid-Course Guidance)를 수행할 수 없고 무유도 비행을 해야 한다. 무유도 비행을 하게 되면 중기 비행 시 바람이나 표적의 이동으로 목표물이 시야각(FOV)에서 벗어날 확률이 크다. 특히 스트랩 다운 탐색기의 경우 중기 유도를 하지 않는다면 동체축에 고정된 장착각을 가지고 있어서 김벌형 탐색기에 비해 시야각이 작아, 목표물을 포착하지 못할 확률이 커진다. 따라서 무유도 중기비행 시 탐색기 시야각 내에 표적이 탐지될 확률을 높이도록 하는 알고리즘에 대한 연구가 필요하다. 이미 필터 등을 통해 관측각으로부터 시선각속도를 추정하는 방법들이 연구된 바 있으며 이들 논문들의 공통적인 특징은 관측각 변화율에 자세각속도를 보상하여 시선각속도(Line-Of-Sight Rate)를 산출하는 방식으로 요약될 수 있다[1,2,3]. 그러나 관측각 변화율에 내재된 자세 각속도와 보정을 위해 측정된 자세각속도의 다이나믹스가 서로 상이하여 실제시스템에 적용할 경우 높은 주파수의 진동현상이 발생함을 관찰할 수 있으며 유도조종루프에 불안정성을 야기한다. 위에서 언급된 문제점을 해결하기 위한 방안으로 영상센서와 목표물과의 사이각인 관측각(Look-Angle)을 피드백 변수로 사용할 수 있다. 관측각을 추정하기 위해서 스트랩 다운 방식의 영상센서를 활용하도록 한다. 영상센서에 감지되는 목표물의 Optical Flow 검출 기법을 적용함으로써 효율적으로 목표물을 추적할 수 있게 된다.

고전적인 추적 알고리즘에서는 목표물을 추적하기 위해 핵심 지점을 사용하였지만, 움직이는 동안 강체로 가정된 목표물은 배경과 구별되어 특정한 속도 벡터를 가지게 되어 Optical Flow 방정식에서 픽셀 그룹 변위의 속도와 픽셀 위치에서의 밝기 변화의 관계를 알 수 있다. TV-L1 (Total Variation Regularized L1) 기법[5]이나 Farneback이 제시한 기법[6]과 같은 고전적인 방식을 통해 Optical Flow 방정식에 대한 접근이 용이해졌지만, Bouguet[8,9]가 구현한 루카스 카나데(Lukas Kanade)[7] 기법에 비해 50-1000배 더 느리다는 단점이 존재하였다. 따라서 본 연구에서는 Optical Flow 알고리즘을 연산하기 위해 Bouguet가 구현한 방식을 적용하여 루카스 카나데 기법을 이용하였다.

위에서 소개한 추적 알고리즘은 표적을 추적하는데 용이하지만, 미사일과 같이 빠르게 이동하는 플랫폼에 탑재될 경우 목표물에 접근할수록 스케일의 변화가 커진다는 단점이 있다. 즉, 목표물에서 특징점을 추출하여 관측각이나 관측각 변화율을 계산하게 되는데, 목표물에 도달할수록 특징점의 속도가 증가하고 목표물에서 벗어나는 특징점이 존재하게 되어 추정된 관측값 및 관측각속도 성능에 영향을 끼치게 된다.

초기 목표물과의 거리만큼 떨어져 있을 때는 약한 특징점을 얻을 수 있고, 목표물과 거리가 가까워질수록 강한 특징점을 얻을 수 있다. 강한 특징점을 이용하기 위해 초기에 획득한 약한 특징점을 시간에 따라 새로 획득한 특징점으로 대체해야하기 때문에 특징점 관리 개념이 필요하다. 본 연구에서는 관심 있는 물체에 대한 균일한 특징점 매핑을 수행하기 위해 특징점 제거 알고리즘(Rejection Algorithm) 기법을 다루었다.

Optical Flow 기법을 이용하기 위해 이미지 생성이 필요하지만, 동역학적 시뮬레이션만으로는 실물과 비슷한 수준의 사진을 생성하기 매우 복잡하여 컴퓨터 비전 시뮬레이션을 이용해 목표물로부터의 영상, 스트랩 다운 카메라로부터의 영상과 같이 실재하는 제원을 이용하여 데이터를 생성할 수 있다. 그런데 컴퓨터 비전을 실행하면서 정적 데이터 세트를 이용하기에는 정적 데이터 세트의 유연성이 부족하다는 문제가 있어 동역학적 시뮬레이션과 컴퓨터 비전 시뮬레이션에서 모두 동역학적 데이터 시트를 갖도록 설정해야 하고, 카메라를 3D 엔진 화면의 중앙에 위치시킨 후, 유도법칙의 동역학적 현상을 시뮬레이션 해야 한다.

동역학적 시뮬레이션 수행 시 영상을 이용하여 매 프레임마다 화면을 캡쳐하여 추적된 특징점을 통해 관측각 및 관측각속도를 계산하여 시뮬링크로 송신하는 것이 3D엔진의 주목적이고, 미사일의 비선형 6자유도 모델[10]의 운동방정식 계산은 시뮬링크에서 수행하게 된다. 따라서 운동역학(Kinetics)을 기반으로 한 Blender나 3Dmax와 같은 엔진을 사용하는 것보다 운동학(Kinematics) 기반의 3D 게임 엔진을 사용하는 것이 더 유용하다. Unity3D나 CryEngine과 같은 3D 엔진은 자체 개발한 게임에 의해 잘 알려져 있지만, 기능성 게임, 공학 분야, 그리고 건축 분야에서 선두적인 소프트웨어로 더 유명하다. CryEngine은 거의 실물에 준하도록 사물을 묘사할 수 있고 많

은 물리적 요소들을 구현할 수 있지만 본 연구에서 고려한 목표물과 탐색기와의 거리는 초기 1.5 km 정도로 목표물에 타격하기 직전을 제외하고는 상당히 거리가 멀어 목표물의 형상이 중요하기 때문에 목표물의 아주 세밀한 모델링은 불필요하다. 또한, 실물에 준하는 그래픽을 이용할 경우 계산 속도가 현저히 저하될 우려가 있기 때문에 실물에 준하는 고사양의 그래픽은 불필요하다. 그리고 접근성 및 난이도 측면에서 Cry-Engine과 달리 OpenCV의 경우 Open Asset Store로부터 다양한 소프트웨어를 제공받을 수 있고 튜토리얼 강좌를 쉽게 접할 수 있고 엔진의 복잡성으로 인한 속도 저하 현상이 발생하지 않으므로 빠른 개발 주기를 확보할 수 있다는 장점이 있다. 따라서 본 연구에서는 Unity3D 엔진을 이용하고, OpenCV을 이용하여 개발하였다.

마지막으로 본 연구에서 동역학 시뮬레이션과 3D 엔진과 연동시키기 위해 TCP(Transmission Control Protocol) 통신을 이용하였다. 즉, 매틀랩 시뮬링크와 Unity3D와의 정보 교환을 하게 되는데, 이 때 시뮬링크에서 Unity3D로 데이터 송신을 할 때 시간 지연이 발생하여 시뮬레이션 시 목표물에 도달하지 못하고 발산하게 되는 현상이 발생했다. 따라서 매틀랩 시뮬링크에서 TCP/IP (Transmission Control Protocol/Internet Protocol) 블록을 별도로 제작하여 시뮬링크에서 데이터를 송신하는 동안 시뮬레이션을 중단하여 문제를 해결하였다.

본 논문에서는 특징점 추출에 관한 내용 및 관리 기법에 대해 서술하고, Unity3D의 시뮬레이션 환경 구성법에 대해 서술하였다. 그리고 Unity3D에서 추출한 특징점을 이용하여 관측각 및 관측각 변화율을 계산하는 법에 대해 기술하였고, 매틀랩 시뮬링크와 Unity3D와 연동하는 과정에 대해 설명하였다. 마지막으로 시뮬레이션을 통해 시뮬링크와 Unity3D간의 상호-시뮬레이션이 가능하다는 것을 확인하였다.

II. 피라미드 구현법을 적용한 루카스 카나데 기법

Lucas와 Kanade는 패치를 이용해서 Optical Flow를 계산할 수 있는 방법을 제안했다[7]. 패치를 이용한 Optical Flow 계산을 수행하기 위하여 인접한 픽셀끼리 함께 움직인다고 가정하였는데, 이렇게 가정하면 문제의 정의가 수학적으로 모호하게 된다. 하지만 소수의 특정 픽셀을 위한 Optical Flow만을 알아야 하는 경우, 추적 알고리즘 연산 시 Optical Flow의 계산 속도를 향상시킬 수 있다. 이러한 방법으로 특정픽셀의 위치를 예측할 수 있고, 코너(Corner)의 형상을 매칭시킬 수 있다. 또한 픽셀 그룹의 특정 형상을 매칭시켜 추적 성능을 향상시킬 수 있다. 하지만, 이 방법은 너무 작은 변위 값을 가정한다는 단점이 존재한다. 이러한 단점을 해결하기 위해 Bouguet는 루카스 카나데 기법에 피라미드 구현법(Pyramidal Implementation)[9]을 적용하여 원하는 지점을 추적하고 다중 스케일 피라미드로 이미지를 정의하여 계산 속도가 50~800배 증가하였고 트래킹 정확도가 향상되어[9] 이 문제를 해결하였다. 피라미드 구현법에 관한 상세한 알고리즘은 참고문헌 [9]에 기술되어있다.

본 연구에서는 피라미드 구현법을 적용한 루카스 카나데 기법을 이용하여 Optical Flow를 계산하였다. 다음 장에서는 Optical Flow 계산 중 트래킹이 더 강건해지고, 정확성이 향상될 수 있도록 특징점 관리 차원에서 발생할 수 있는 문제를 정리하고 해결 방안에 대해서 기술하였다.

III. 특징점 관리 기법

본 장에서는 시스템이 목표물에 다가갈 때 성능이 좋고 강건성이 뛰어난 특징점을 획득하기 위하여 일관성이 없는 특징점을 제거하는 기법에 관하여 서술하고자 한다.

외란은 다양한 원인에 의하여 발생하게 되는데 가장 일반적으로 목표물이 장애물에 의해 가려지는 경우(Occultation)나 배경에 특징점이 발생하는 경우, 목표물 추적에 실패할 때 발생하게 된다. Fig. 1(a), (b)에 이를 나타내었다. 외란 발생의 또 다른 원인은 목표물 주변에 특징점이 불균일하게 분포하기 때문이다. 두 가지의 기준을 정의하여 성능이 좋은 특징점의 분포를 유지할 수 있다. 첫째, 다른 특징점과는 다르게 가장 비정상적인 속도를 가지는 특징점을 제거하는 방법이다. 둘째, 특징점이 가장 밀집되어 있는 영역의 특징점을 제거하는 방법이다. 첫 번째 방법과 두 번째 방법을 적절히 이용한 하이브리드 기법을 통해 목표물 추적 시 오류를 줄일 수 있고, 배경에 발생하는 특징점을 제거할 수 있다. 또한, 목표물의 구석부근을 균일하게 추적할 수 있는 장점이 있다. 이러한 특징점 제거 기법들은 목표물에 느린 속도로 다가갈 경우 충분히 성능을 나타내지만, 속도가 빠른 영상 추적기의 경우 Fig. 1(b)에 나타낸 것과 같이 추적 말미에 주밍 모션(Zooming Motion)을 동반하게 된다. 즉, 목표물

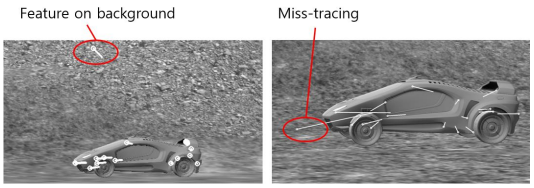

Fig. 1. Sparse optical flow tracking: (a) Feature on the background(left), (b) Zooming and miss-tracing(right)

에 가까이 다가갈수록 목표물이 화면에 확대되는 속도가 급격히 증가하기 때문에, 목표물 추적에 실패한 특징점이 존재하여 특징점이 목표물 밖으로 벗어나는 현상이 발생하게 된다.

이러한 문제는 필수행렬(Essential Matrix)을 이용한 에피폴라 기하학(Epipolar Geometry) 기반의 특징점 제거 기법을 통하여 해결할 수 있다[11]. 단일 목표물을 고려했을 때 목표물에 분포한 특징점의 움직임은 카메라 위치의 변화로 생각할 수 있다. 임의의 영상에서 바로 다음 영상 에피폴라 라인까지의 필수행렬 매핑 지점은 아웃라이어(Outliers)를 제거하는데 이용할 수 있다. 이러한 작업들은 보통 필수 행렬을 강건하게 추정하거나 아웃라이어를 제거하기 위해 RANSAC (Random Sample Consensus)이나 PROSAC (Progressive Sample Consensus)과 같은 알고리즘을 이용하여 구현할 수 있다.

이와 같은 제거기법을 구성할 때 특징점이 충분히 분포할 수 있도록 새로운 특징점을 생성하였다. 관심 있는 구석 부분을 스캐닝 하는 과정에서 많은 계산량이 요구되기 때문에 해당 제거 기법 알고리즘을 추적 시뮬레이션과 병렬로 구동시키거나, 고정된 주파수 조건 내에서 구동시켜야 한다. 또한 이미 알고 있는 특징점 주위에 적절한 크기의 영상을 클립핑(Clipping)하고 적절한 수의 특징점이 생성되도록 생성될 특징점 수를 정해야 하는데, 클립핑을 이용하여 성능 문제를 해결할 수 있지만, 실제 특징점을 포함하는 공간의 마진 크기에 관하여 정의해야 한다. 본 논문에서는 Fig. 2와 같이 추적기가 점점 가까워짐에 따라 물체가 커지는 것을 보상하기 위하여 공간의 마진 크기를 표적 크기의 두 배만큼 정하였다.

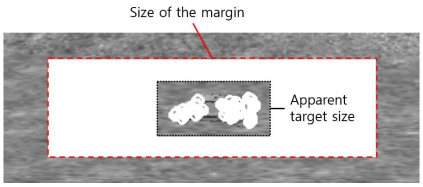

Fig. 2. Feature respawning area

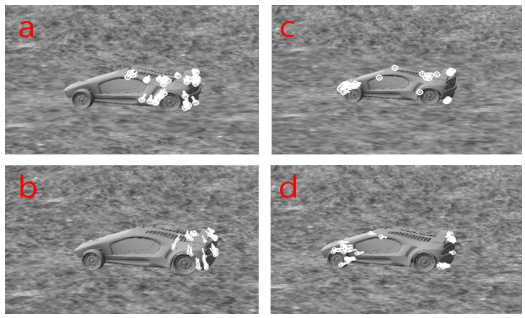

Fig. 3. Features with different rejection criteria: (a) No feature rejection, (b) Aberrant speed rejection, (c) Dense area rejection, (d) Blend: 4/5 density

매 10 프레임마다 비어있는 특징점 슬롯의 1/3 만큼의 슬롯을 생성하여 안정적인 결과를 얻을 수 있었다.

해당 특징점 관리 알고리즘은 Fig. 3에 나타낸 것과 같이 구현할 수 있다.

Figure 3(a)에서 표적의 형상이 기하학적으로 복잡하기 때문에 표적의 뒷부분에서 특징점이 축적되는 것을 확인할 수 있다. Fig. 3(b)에서 확인할 수 있듯이 알맞지 않은 속도로 특징점을 제거하면 성능이 더 악화된다. 특징점의 밀도 기준은 표적의 전방과 후방에 걸쳐 특징점의 분포가 균형을 이루도록 설정된다. 밀도 기준에 맞춰 4/5 비율로 설정했을 때 가장 좋은 성능을 나타내었다. 결과적으로, 특징점을 균형 있게 분포시키고 불규칙한 특징점을 제거함으로써 더 부드럽고 바이어스가 제거된 결과를 얻을 수 있다.

Ⅳ. 상호 시뮬레이션 환경 구성

4.1 Unity3D의 시뮬레이션 환경 구성

Unity 3D의 시뮬레이션 환경은 다음 세 가지의 주요 요소로 이루어진다.

- 맵(Map)(현재 500x500m)
- 웨이포인트 경로(Waypoint Circuit)를 추종하는 자동차
- 주 카메라가 부착된 탐색기-동역학 모션을 제공하는 요소

탐색기가 목표물에서 관성 좌표계(Fig. 4의 검은색으로 나타낸 그래프) 기준으로 X축 방향으로 1500 m, 높이(Y축) 50 m, Z축 방향으로 목표물과 같은 -350 m인 지점에서 맵을 향해 가리키는 모습을 Fig. 4에 나타내었고, 카메라에서 찍은 화면은 Fig. 5에 나타내었다.

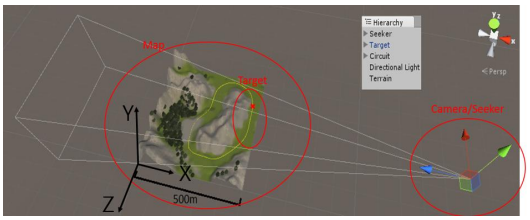

Fig. 4. Simulation environment

Fig. 5. Scene from camera/seeker

4.2 Omniscient 시뮬레이션 구성

시뮬링크에서 설계된 동적 모델링을 이용하여 시뮬레이션하기 전에, Unity 자체 내에서 간단한 요격(Intercept) 유도 시스템을 구현하고자 한다. 이때 유도 명령을 생성하기 위하여 시선벡터(Line Of Sight)를 이용한다. 본 장에서는 Omniscient 시뮬레이션이기 때문에 시선벡터를 구할 때 특징점을 이용하지 않았고, 목표물의 중심부에서 탐색기까지의 거리를 이용했다. 유도 법칙은 식 (1)의 PN(Proportional Navigation) 유도 법칙[4]을 이용하여 목표물에 접근하는 Omniscient 탐색기를 구현하였다.

$$\gamma_m(t) = -N\frac{V_m}{r(t)}\theta_m(t-\tau)$$
$$\dot{\gamma}_m(t) = \frac{A_m(t)}{V_m(t)} = \frac{A_{mc}(t)}{V_m(t)} \quad (1)$$

여기서 A_m은 미사일의 법선방향 가속도, A_{mc}는 미사일의 법선 방향 가속도 명령, γ_m은 미사일의 비행경로 각, N은 유도 게인 상수, V_m은 미사일의 속도, r은 미사일에서 목표물까지의 거리, θ_m은 미사일 속도 벡터와 시선벡터와의 각도이다.

탐색기의 동역학을 구현하기 위해 별도의 Unity Script를 작성해야 한다. Unity3D에서 제공하는 라이브러리 함수를 이용하였다. 우선 Private 선언을 하여 목표물을 변수로 지정하고,

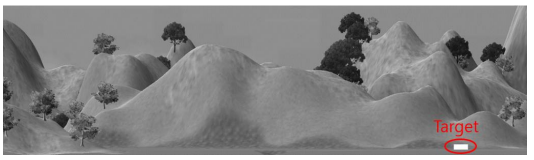

Fig. 6. Initial position of the target

목표물의 속도 및 유도 게인 상수를 설정한 후 [SerializeFeild] 속성을 지정해준다. Start() 함수를 이용하여 탐색기의 초기 속도를 설정한다. 그리고 나서 FixedUpdate() 함수를 이용하여 PN 유도법칙 알고리즘을 계산하고 업데이트한다. FixedUpdate() 함수에 PN 유도법칙 알고리즘을 코딩하면 된다.

4.3 프레임 캡쳐 및 트래킹 초기화

이번 절부터는 특징점을 이용하여 시뮬레이션을 구성하고자 한다. 화면의 프레임 캡쳐는 단순히 Texture2D.ReadPixels 함수를 이용하면 된다. 프레임 캡쳐가 됐으면 코너의 특징점을 추출해야 한다. 초기 프레임의 경우 목표물의 초기 위치를 알고 있다고 가정하기 때문에 고정된 상태이다. 초기 프레임에서 물체의 경계선을 얻고 카메라의 투영 행렬을 이용하여 OpenCV 좌표에 변환시킨다. 그 후 Imgproc.goodFeaturesToTrack 함수를 적용시킨 서브 이미지에서 뚜렷한 코너점을 추출할 수 있다. Fig. 6은 트래킹 초기화가 이루어졌을 때 목표물의 초기 위치를 나타낸다. 특징점이 목표물 경계선에 형성되어 목표물이 완전히 가려진 것을 확인할 수 있다. 이는 초기에 목표물이 탐색기와 멀리 떨어져 있기 때문에 영상에서 작게 나타나기 때문이다.

4.4 특징점을 이용한 관측각 및 관측각의 변화율 추정

관측각은 아래 식 (2)을 이용하여 구할 수 있다.

$$(\lambda_\psi, \lambda_\theta) = \frac{FOV}{Height} \times (\Delta x, \Delta y) \quad (2)$$

여기서 λ_θ와 λ_ψ는 관측각, FOV는 탐색기의 시야각(Field of View), $Height$는 탐색기의 고도, Δx와 Δy는 현재 프레임에서 목표물의 중심 위치와 카메라 중심과의 차이로 다음과 같은 과정으로 구할 수 있다. 카메라의 중심은 카메라 스크린의 가로 및 세로 길이의 절반으로 정의할 수 있고, 목표물의 중심 위치는 Unity3D의 함수 중 findCentroid 함수를 이용하여 추출된 특징점들의 위치를 대입함으로써 구할 수 있다.

관측각속도의 경우 추출된 특징점들의 위치를 Unity3D의 함수인 findMeanDisplacement를 이용하여 평균 변위를 구한 후 FOV를 곱하고 고도와 시간 증분을 나누어 구한다. 이를 아래 식 (3)에 나타내었다.

$$(\dot{\lambda}_{\psi},\dot{\lambda}_{\theta}) = \frac{FOV}{Height} \times \frac{(\overline{\triangle x},\overline{\triangle y})}{dT} \tag{3}$$

여기서 $\overline{\triangle x}$와 $\overline{\triangle y}$는 특징점들의 평균 변위, dT는 시간 증분이다. 시간 증분은 시뮬레이션 주기에 따라 달라진다.

4.5 탐색기 동역학 구현

탐색기 동역학을 구현하기 위해 참고문헌 [10]을 참조하였다. 탐색기의 피치 방향 시선각과 요 방향 시선각은 아래와 같이 정의할 수 있다[10].

$$\begin{aligned} \sigma_{\theta} &= \lambda_{\theta} + \theta \\ \sigma_{\psi} &= \lambda_{\psi} + \psi \end{aligned} \tag{4}$$

여기서 θ와 ψ는 기준좌표계에 대한 동체의 피치, 요 자세 각이다. 피치축 가속도 명령(a_{θ})과 요축 가속도 명령(a_{ψ})은 다음과 같이 나타낼 수 있다[10].

$$\begin{aligned} a_{\theta} &= NV_m\dot{\sigma_{\theta}} + g \\ a_{\psi} &= NV_m\dot{\sigma_{\psi}} \end{aligned} \tag{5}$$

여기서 g는 중력가속도를 나타낸다. 피치 방향 시선각속도와 요 방향 시선각속도는 아래와 같이 동체각속도와 관측각속도의 합으로 정의할 수 있다[10].

$$\begin{aligned} \dot{\sigma_{\theta}} &= q + \dot{\lambda}_{\theta} \\ \dot{\sigma_{\psi}} &= r + \dot{\lambda}_{\psi} \end{aligned} \tag{6}$$

본 연구에서는 유도탄에 탐색기와 3축 각속도계를 이용하므로 유도각속도 명령을 동체각속도 명령으로 바꿔야 하고 아래와 같이 나타낼 수 있다[10].

$$\begin{aligned} q_{command} &= N\dot{\sigma_{\theta}} + \frac{g}{V_m} \\ r_{command} &= N\dot{\sigma_{\psi}} \end{aligned} \tag{7}$$

여기서 $q_{command}$는 피치 각속도 명령, $r_{command}$는 요 각속도 명령을 나타낸다.

4.6 시뮬링크와 Unity3D의 통신

지금까지 Unity3D를 이용하여 기본적인 동역학의 시뮬레이션을 3D 환경에서 구현하는 과정에 대해 기술하였다. 본 장에서는 시뮬링크에서 구성한 동역학 모델을 Unity3D와 연동하여 3D 환경에서 구현하고자 한다. 즉, 상태 변수의 업데이트는 시뮬링크에서 이루어지고 Unity3D에서는 이미지를 이용하여 관측각 및 관측각속도, 목표물의 위치를 시뮬링크로 제공하게 된다. 즉, 시뮬링크와의 수신 및 송신 과정에서 교환한 데이터는 다음과 같이 정리할 수 있다.

- Unity3D의 수신 데이터: 미사일의 위치(2 singles), 미사일의 자세(2 Singles)
- Unity3D의 송신 데이터: 관측각(2 singles), 관측각속도(2 Singles), 목표물의 위치(3 Singles)

시뮬링크를 이용하여 Unity3D를 구동할 때 Unity3D로부터 데이터를 수신 받을 때 발생하는 딜레이를 줄이기 위해 TCPIP 통신을 이용하여 시뮬링크 실행을 중단시키는 과정을 추가하였다.

Ⅴ. 시뮬레이션

시뮬레이션은 두 가지로 나뉘어 구현하였다. 첫 째로는 유도 법칙과 동역학 모두 Unity3D에서 코딩하여 상태 변수를 업데이트 하는 방식으로 유도 법칙으로는 참고문헌 [5]의 PN 유도 법칙, 동역학 법칙으로는 참고문헌 [10]에서 사용한 비선형 6자유도 시뮬레이션을 수행하였다. 참고문헌 [10]에 주어진 미사일 운동방정식을 시뮬링크로 구현하였고, Unity3D를 통해 측정된 관측각과 관측각속도 값을 시뮬링크로 송신하였다.

두 번째 시뮬레이션으로는 시뮬링크와 Unity3D를 연동하였고, 4.6절에서 서술한대로 미사일 위치 및 자세는 시뮬링크에서 계산하고, 관측각 및 관측각 변화율, 목표물 위치는 Unity3D에서 계산한다.

Table 1. Simulation conditions

Parameter	Value
Initial missile position	(1500, 20, -250)m
Initial target position	(0, 0, -250)m
Initial missile velocity	(250, 0, 0)m/sec
Inital target velocity	(0, 0, 50)km/h
Proportionality constant	150
Guidance loop frequency	50Hz
Unity3D Communication freq. (Only for second case)	5msec

두 시뮬레이션의 환경 조건 및 시나리오 모두 동일하다. 시나리오는 하나의 포탄이 웨이포인트를 움직이는 하나의 표적을 맞추는 간단한 예제이다. 환경 조건은 Table 1과 같다.

5.1 Unity3D만을 이용한 시뮬레이션 결과

Figure 7에 프레임에 따른 추적된 특징점을 나타내었다. 초기 프레임은 0이고 이후 20, 40, 60, 63, 66, 68, 69번째 프레임을 캡쳐하여 나타내었다. 이후 목표물과 충돌하게 된다.

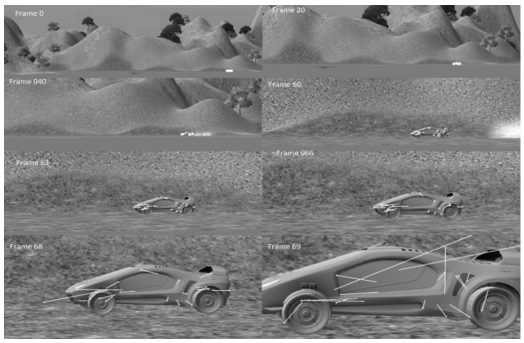

Fig. 7. Features tracked along the frame

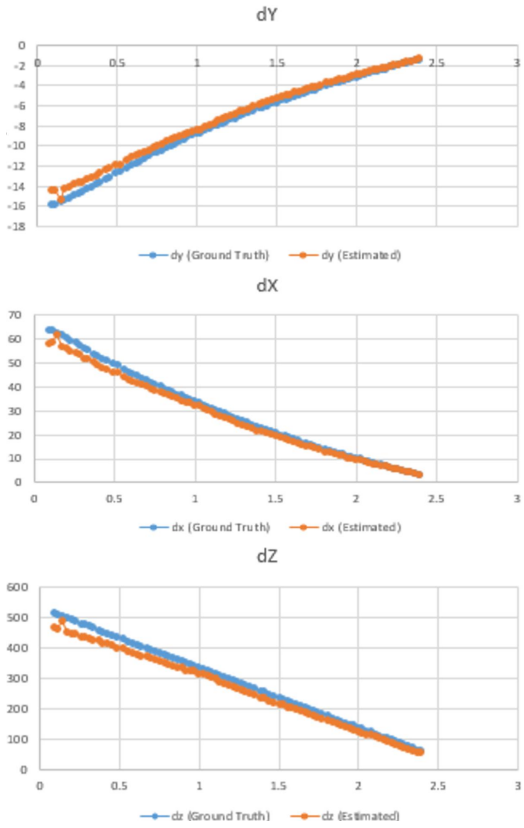

Fig. 8. Displacement(m) vs time(sec)

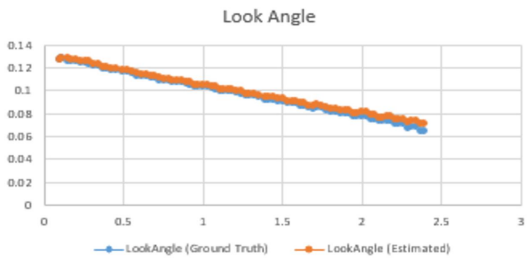

Fig. 9. Look angle(rad) vs time(sec)

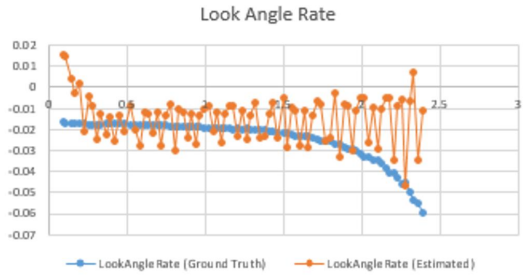

Fig. 10. Look angle rate(rad/sec) vs time(sec)

초기 프레임 0일 때 목표물과 거리가 멀기 때문에 목표물이 작게 포착되어 특징점에 가려진 모습이 나타났다. 프레임이 지날수록 점점 목표물에 가까워지면서 트래킹된 특징점이 증가하게 되고 충돌하는 시점(프레임 68, 69)에서는 주밍 현상에 의해 특징점이 목표물 영역을 벗어나는 현상이 발견된다.

Figure 8은 각 축의 실제 변위와 특징점을 이용하여 추정한 변위를 비교한 그래프이다. 초기에는 오차가 크지만, 충돌 시점에 도달할수록 오차가 점점 줄어드는 것을 확인할 수 있다.

Figure 9는 관측각을, Fig. 10은 관측각 변화율을 나타낸다. 관측각 및 관측각 변화율에 바이어스가 생기는 것을 확인할 수 있는데 정밀하게 분석하기 위해 RMSE(Root Mean Square Error)를 구하였고 이를 Fig. 11과 Fig. 12에 나타내었다.

Figure 11은 시간에 따른 관측각 RMSE를, Fig. 12는 시간에 따른 관측각 변화율 RMSE를 나타내었다. 목표물에 도달할수록 관측각 및 관

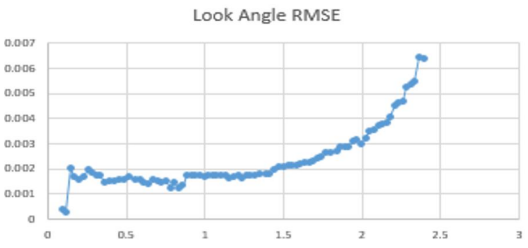

Fig. 11. Look angle RMSE(rad) vs time(sec)

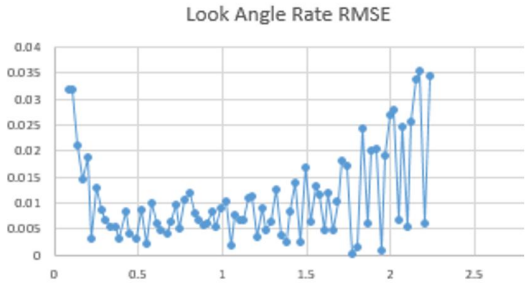

Fig. 12. Look angle rate RMSE(rad/sec) vs time(sec)

측각 변화율에 바이어스가 생기는 것을 확인할 수 있다. 이는 목표물에 가까이 갈수록 특징점들의 속도가 증가함에 따라 주밍 모션이 발생하여 특징점이 목표물의 중심에서 멀어지기 때문에, 목표물 중심이 부정확하게 추정되기 때문이다. 이 오차는 추가적인 알고리즘을 적용하여 완화할 수 있지만, 존재하더라도 현재 시뮬레이션 조건에서 목표물 타격에는 큰 영향이 없다.

5.2 Unity3D와 시뮬링크의 통신을 통한 시뮬레이션 결과

TCP 통신을 이용하여 Unity3D와 시뮬링크 연동을 할 때 시뮬링크에서 Unity3D로 데이터를 보내는 기간 동안 딜레이가 발생하게 되어 관측각 그래프에 위상 오차가 발생하게 된다. 이를 Fig. 13에 나타내었다.

Figure 13에서 확인할 수 있듯이 관측각 추정값에 4 프레임 정도의 통신 딜레이로 인한 위상 오차가 발생하는 것을 알 수 있다. 따라서 위상 오차로 인해 약 30프레임부터는 목표물이 화면에서 벗어나 더 이상 목표물로 유도할 수 없게 되고, 이는 30프레임 이후 일정한 관측각을 갖는 이유이다.

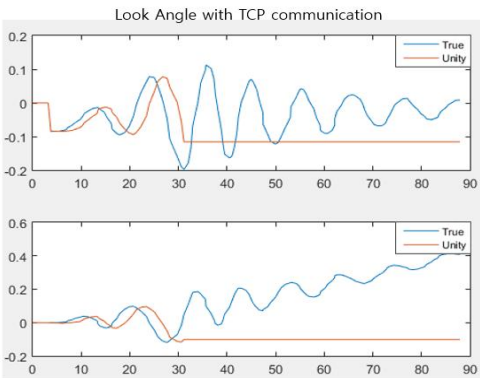

Fig. 13. Look angle (rad) vs frame

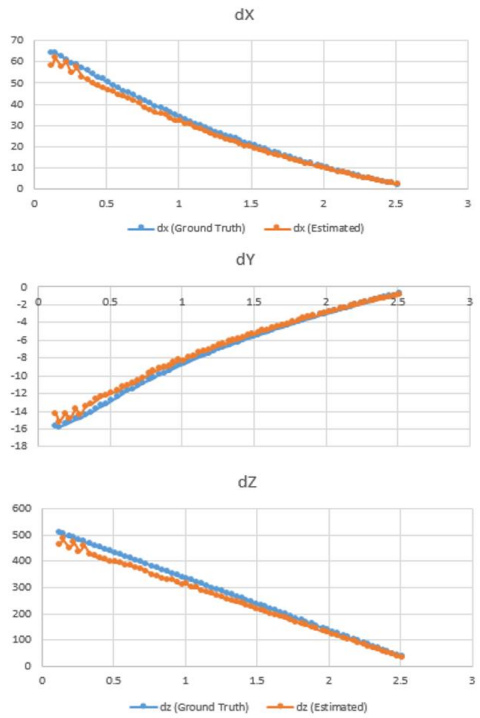

Fig. 14. Displacement(m) vs time(sec) with Simulink

TCPIP 블록을 시뮬링크의 매틀랩 함수에 추가했을 때는 Figs. 14, 15, 16과 같이 딜레이가 훨씬 줄어들어 목표물을 타격할 수 있었다. Unity3D만을 이용했을 때의 결과인 Fig. 11과 시뮬링크와 통신을 하여 얻어진 결과인 Fig. 16을 비교해보면 통신 딜레이가 완전히 없어지지는 않아 관측각 RMSE가 TCPIP 블록을 이용했을 때 충돌 시 약 두 배이지만 충돌 시를 제외한 나머지 구간에서는 비슷한 수준의 오차가 나타나기 때문에 타격 여부에 영향을 주지 않는다. Fig. 17은 시간에 따른 특징점의 개수를 나타냈는데, 초기에는 목표물이 작기 때문에 특징점의 수가 적고 시간

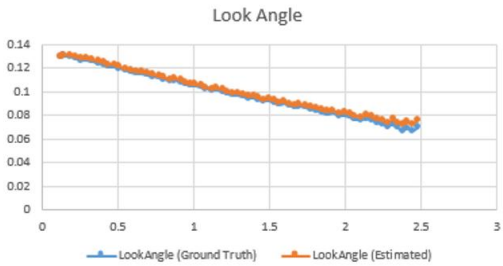

Fig. 15. Look angle(rad) vs time(sec) with Simulink

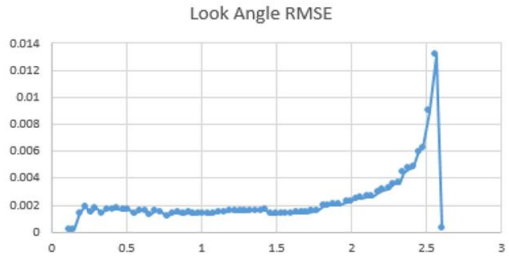

Fig. 16. Look angle RMSE(rad) vs time(sec) with Simulink

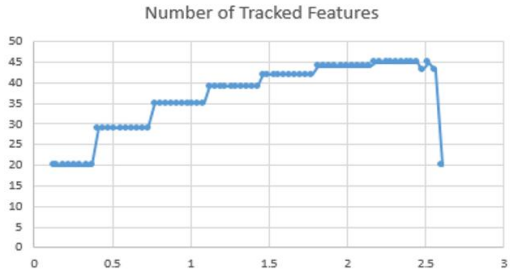

Fig. 17. Number of tracked features vs time(sec) with Simulink

이 지날수록 점점 카메라 화면의 목표물이 커지기 때문에 목표물의 경계가 넓어져 특징점의 수가 증가하게 된다. 충돌 직전 목표물의 중심에 가까워지기 때문에 화면 전체가 목표물이 잡혀 목표물의 일부만 화면에 보이기 때문에 특징점의 수가 급격히 줄어드는 것을 알 수 있다.

Ⅵ. 결 론

본 연구에서는 Optical Flow를 기반으로 스트랩 다운 영상 추적기의 항법을 구현하기 위해 특징점 관리 기법에 대해 연구하였고, 시각적 시뮬레이션 환경과 동역학적 시뮬레이션 환경이 연동된 테스트 베드를 구축하여 성능을 평가하는 데에 주안점을 두었다. 본 시뮬레이션 테스트벤치를 이용하여 여러 가지 알고리즘을 시각적으로 구현할 수 있는데, 시뮬링크를 이용하여 설계한 미사일의 동역학 및 유도법칙을 Unity3D와 통합하여 실시간으로 관측각과 관측각속도를 측정할 수 있었고, 커스텀 TCP/IP 통신을 이용하여 5msec 주기에서도 데이터를 송신할 때 딜레이 문제가 많이 개선되었다. 매개변수로부터 직관적으로 기동을 확인하기 어려운 동력학 시스템을 다루는 분야에서 본 시뮬레이션 테스트벤치의 활용도가 높을 것으로 판단된다.

후 기

본 연구는 LIG넥스원의 지원으로 이루어졌으며 이에 감사를 드립니다.

References

1) Waldmann, J., “Line-of-sight rate estimation and linearinzing control of an imaging seeker in a tactical missile guided by proportional navigation,” *IEEE Transactions on Control Systems Technology*, Vol. 10, No. 4, July, 2002, pp. 556~567.

2) Zhang, Y., Li, J., and Li, H., “Line of sight rate estimation of strapdown imaging seeker based on particle filter,” *Systems and Control in Aeronautics and Astronautics (ISSCAA), 2010 3rd International Symposium*, June, 2010, pp. 191~195.

3) Lin, Z., Yao, Y., and Ma, K. M., “The design of los reconstruction filter for strap-down imaging seeker,” *Machine Learning and Cybernetics, Proceedings of 2005 International Conference*, Vol. 4, August, 2005, pp. 2272~2277.

4) Son, D., and Seong, S., “PNG law for missile guidance under time delayed look angle measurement,” *Control Automation and Systems(ICCAS), 2010 International Conference*, 2010, pp. 1328~1331.

5) Pérez, J., Llopis, E., and Facciolo, G., “TV-L1 Optical Flow Estimation,” *Image Processing On Line*, Vol. 3, 2013, pp. 137~150.

6) Farnebäck, G., “Two-Frame Motion Estimation Based on Polynomial Expansion”, *Image Analysis: 13th Scandinavian Conference*, June, 2003, pp. 363~370.

7) Lucas, B., and Kanade, T., “An iterative image registration technique with an application to stereo vision,” *Proceedings DARPA Image Understanding Workshop*, 1981, pp. 674~679.

8) Bouguet, J., *Pyramidal implementation of the lucas kanade feature tracker*, Intel corporation, Microprocessor Research Labs, 2000.

9) Bouguet, J., *Pyramidal implementation of the affine lucas kanade feature tracker description of the algorithm*, Intel Corporation,

Microprocessor Research Labs, 2001.

10) Hong, J., and Ryoo, C., "Homing Loop Design for Missiles with Strapdown Seeker", *Journal of The Korean Society for Aeronautical and Space Sciences*, Vol. 42, No. 4, 2014, pp. 317~325.

11) Hartley, R. I., and Zisserman, A., *Multiple View Geometry in Computer Vision*, Cambridge University Press, ISBN: 0521623049, 2000.