

## 달 탐사 궤도에서 행성과 달을 이용한 천체항법

2014-11238 김재우

### I. 연구목적

약 50년간 진척되지 못했던 유인 달 탐사 프로그램이 NASA와 SpaceX의 달 탐사 프로젝트(Artemis Program)를 필두로 다시 시작되었다. 또한 기반기술의 진보, 새로운 시장의 발견을 통해 우주 관련 기업들이 BM을 구축하게 되면서, 앞으로 우주탐사 관련 시장은 점점 더 커질 것으로 기대된다.

이에 따라 효율적인 항법 시스템의 필요성은 그 어느 때보다 높아지고 있다. 우주 발사체의 특성상 무게가 발사체 및 임무 설계에 지대한 영향을 미치기 때문에, 단순 시스템 구축과 운용 비용뿐만 아니라 시스템 자체가 차지하는 부피와 무게 역시 효율성 측면에서 제고되어야 한다.

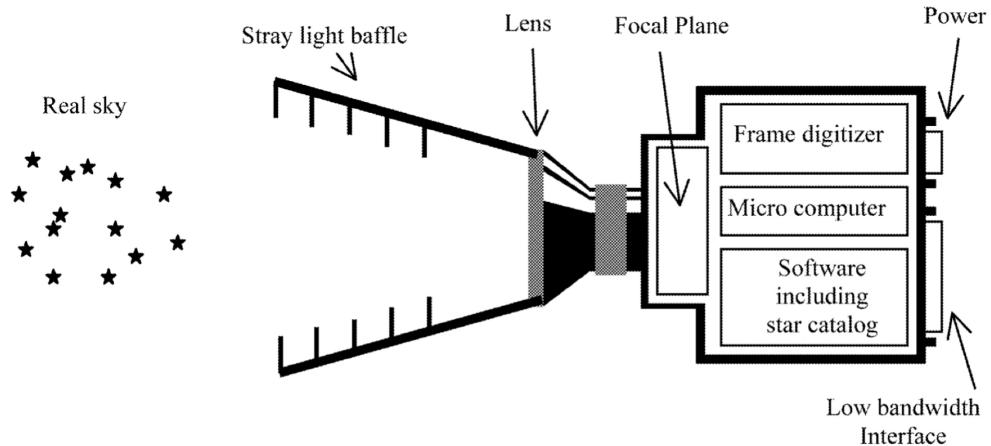
별 추적기는 거의 모든 우주비행체에 탑재되는 자세결정센서로서, 외부의 시각정보를 받아들이고 처리하여 유용한 정보를 사용자에게 제공하는 장치이다. 따라서, 이것을 활용하여 항법을 할 수 있다면 추가적인 비용 없이 항법시스템을 구축할 수 있을 것이다. 또한, 측정치가 시간에 대해 독립적이기 때문에 일반적으로 널리 사용되며 오차가 시간에 대해 누적되는 관성항법 시스템(INS, Inertial Navigation System)을 보조할 수 있는 항법수단으로도 사용 가능할 것이다. 이 연구는 위의 동기를 바탕으로 달 탐사궤도 상에서 수집 가능한 시각정보를 바탕으로 항법을 할 수 있는 방법을 연구한다.

### II. 지금까지 연구수행 내용 및 결과

#### A. 배경지식

##### 1. 별 추적기(Star Tracker)의 원리[1][2][3]

별추적기는 카메라와 마이크로컴퓨터로 이루어진 장비이다. 카메라는 컴퓨터에서 처리될 시각정보를 수집하는 역할을 한다. 여기서 시각정보란 사용자가 바라본 천체들의 상대적인 방향정보를 의미한다. 마이크로컴퓨터는 성표(Star Catalogue)데이터를 저장하고, 카메라를 통해 수집된 시각정보와 성표를 비교분석 및 처리하여 필요한 정보를 사용자에게 제공하는 역할을 수행한다. 관련 기술의 발전으로, 현재 별 추적기를 통해 자세정보와 각 별의 방향정보를  $arcsec$  단위의 정확도로 추정할 수 있는 것으로 알려져 있다.



**Fig. 1** 별 추적기 [3]

## 2. 행성 궤도력(*Planets Ephemeris*) 및 좌표계[4][5]

행성 궤도력은 시간에 대한 행성들의 위치정보를 미리 계산해놓은 것이다. 이 연구에서는 NASA JPL(Jet Propulsion Laboratory)에서 제공하는 DE405를 이용할 것이다. DE405는 1997년 5월에 제작되어 널리 사용되고 있는 궤도력 중하나이며, 1599년 12월 9일 ~ 2201년 2월 20일까지의 천체의 위치정보를 얻을 수 있다. 정확도는 내행성의 경우  $0.001 \text{ arcsec}$ , 외행성의 경우  $0.1 \text{ arcsec}$  이하이다.

DE405는 ICRF1(International Celestial Reference Frame 1)을 기준좌표계로하여 작성되었다. ICRF는 국제적으로 천체를 다룰 때 사용되는 좌표계의 표준이다. 원점은 태양계의 질량중심이며, 매우 먼 거리에 있는 별들이 발산하는 전자기파를 이용하여 축을 아주 정확하게 설정한다. 이 때, 각 축의 방향은 지구적 좌표계와 거의 동일하다.

## B. 문제상황 정의

이 연구에서는 문제의 단순화를 위해 2D 평면 위에서 생각을 전개할 것이다. 별 추적기를 통해  $n$ 개의 행성이 관측된 상황을 가정하며, 이 때 각 행성들은 정확히 식별 가능하다고 가정한다. 또한, 자세정보는 별 추적기를 통해 정확히 얻을 수 있다고 가정하여 고려대상에서 제외한다. Fig. 2은 이 연구에서 다룰 문제상황을 간단하게 나타낸 것이다. 정리하면,

$$\text{별 추적기를 통해 측정된 } n\text{개의 관측자-행성 방향벡터} : \{\hat{e}_1, \hat{e}_2, \dots, \hat{e}_n\}$$

$$\text{행성 궤도력을 통해 얻은 } n\text{개의 행성의 기준좌표계 위에서의 위치벡터} : \{\vec{r}_1, \vec{r}_2, \dots, \vec{r}_n\}$$

을 이용하여 관측자의 위치벡터  $\vec{p}_{true}$ 를 추정하는 것이 이 연구에서 다룰 문제이다.

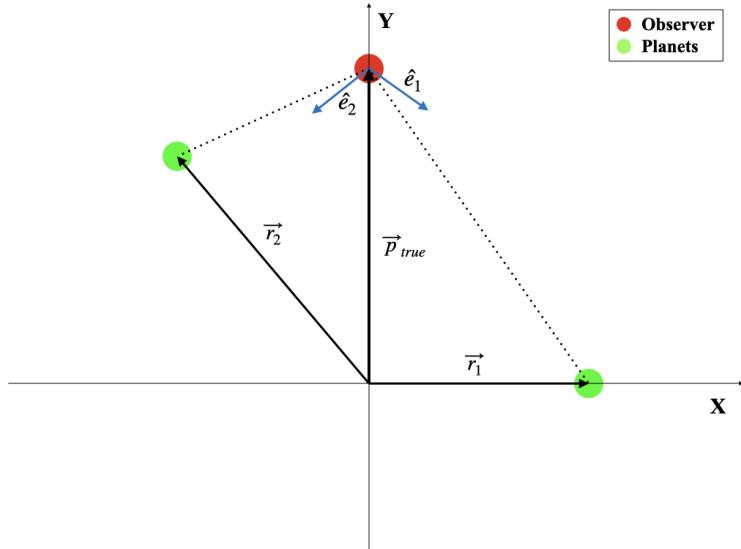


Fig. 2 문제상황

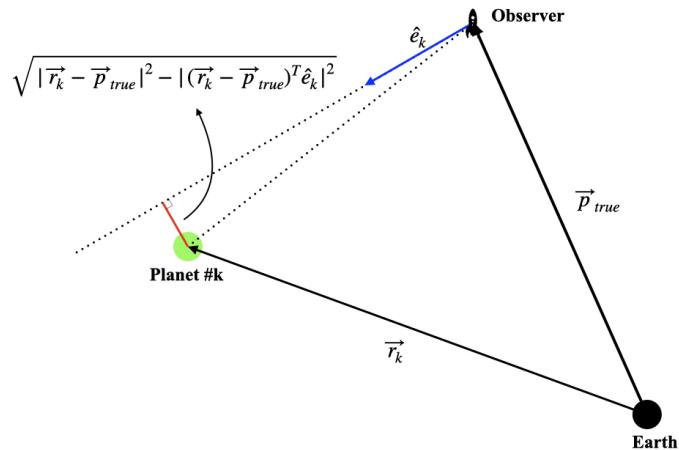


Fig. 3 k번째 관측된 행성과 측정치 손실

### C. 위치추정 알고리즘[6][7]

#### 1. 위치추정 알고리즘 - 최소제곱법(Least Squares)

별 추적기를 통해 어떤 행성을 관측하였을 때, 관측된 관측자-행성 방향벡터  $\hat{e}$ 는 각오차  $\epsilon$ 을 포함한 값이다. 이 때,  $\hat{e}$ 방향에 내린 관측자-행성을 잇는 벡터의 수선의 발의 크기(Fig. 3에서 빨간색 선분의 길이)는 각오차가 커짐에 따라 함께 커질 것이다. 이에 착안하여 관측된 각 행성들의 관측치에서 발생한 손실을  $\sqrt{|\vec{r}_k - \vec{p}_{true}|^2 - |(\vec{r}_k - \vec{p}_{true})^T \hat{e}_k|^2}$

의 제곱으로 정하고, 관측된  $n$ 개의 행성들의 손실의 총 합을 비용함수로 정한다.

$$L = \sum_{k=1}^n (|\vec{r}_k - \vec{p}|^2 - |(\vec{r}_k - \vec{p})^T \hat{e}_k|^2) \quad (1)$$

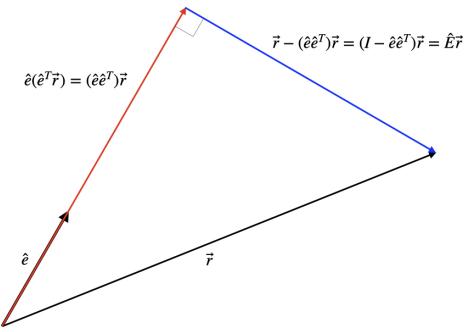
이 때, 비용함수는  $\vec{p}$ 에 대한 함수이다. 이 함수값을 최소로 만드는  $\vec{p}$ 를 최적해  $\hat{p}$ 로 정한다. 비용함수는  $p$ 에 대한 2차함수이고 항상 양수이므로( $\because |(\vec{r}_k - \vec{p})^T \hat{e}_k| \leq |\vec{r}_k - \vec{p}|$ ), 벡터미분법을 통해 최솟값이 발생하는  $\hat{p}$ 를 간단하게 구할 수 있다.

$$\frac{dL}{d\vec{p}}(\hat{p}) = -2 \sum_{k=1}^n [(\vec{r}_k - \hat{p}) - \hat{e}_k \hat{e}_k^T (\vec{r}_k - \hat{p})] = 0 \quad (2)$$

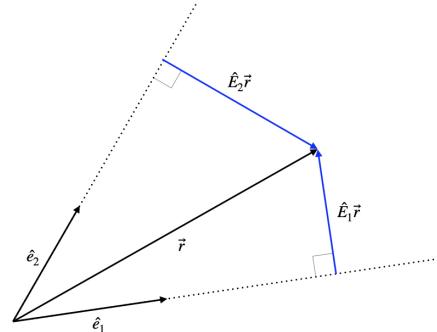
정리하면,

$$\hat{p} = \left( \sum_{k=1}^n \hat{E}_k \right)^{-1} \left( \sum_{k=1}^n \hat{E}_k \vec{r}_k \right), \text{ where } \hat{E}_k = I - \hat{e}_k \hat{e}_k^T \quad (3)$$

## 2. 최소제곱법의 기하학적 이해



**Fig. 4** 변환  $\hat{e}\hat{e}^T$  와  $\hat{E}$



**Fig. 5**  $\hat{e}_1, \hat{e}_2, \vec{r}, \hat{E}_1 \vec{r}, \hat{E}_2 \vec{r}$ 의 관계

지금까지 최소제곱법을 이용한 항법 알고리즘에 대한 아이디어를 설명하고 위치추정벡터를 구하는 과정을 간단히 소개하였다. 하지만, 이후 시뮬레이션의 방향성을 설정하고 그것의 결과를 분석하기 위해서는 항법 알고리즘에 대한 좀 더 깊은 이해가 필요하다.

모든 행렬은 선형변환이다.  $\hat{e}_k \hat{e}_k^T$ 는  $2 \times 2$ 행렬로, 임의의 벡터  $\vec{r}$ 을  $\hat{e}_k$ 위로 정사영하는 선형 변환이다. 따라서,  $\hat{E}_k = I - \hat{e}_k \hat{e}_k^T$ 는 임의의 벡터  $\vec{r}$ 을  $\hat{e}_k$ 위에 내린 수선의 발로 변환하는 선형 변환이다. Fig. 4는 이것을 그림으로 나타낸 것이다. Fig. 5는 한 예시로  $\hat{e}_1, \hat{e}_2, \vec{r}$ 와  $\hat{E}_1 \vec{r}, \hat{E}_2 \vec{r}$ 사이의 관계를 나타낸 것이다.  $\hat{p} = (\sum_{k=1}^n \hat{E}_k)^{-1} (\sum_{k=1}^n \hat{E}_k \vec{r}_k)$ 를 구하는 과정은 각 행성들의 위치벡터를 별 추적기를 통해 측정된 관측자-행성 위로의 수선의 발들의 합에  $\sum_{k=1}^n \hat{E}_k$ 의 역변환을 취하는 것이다. 또한,  $\sum_{k=1}^n \hat{E}_k \vec{p} = \sum_{k=1}^n \hat{E}_k \vec{r}_k$ 를 만족시키는  $\hat{p}$ 를 찾는 과정으로 이해할 수도 있다.

행성이 2개 관측되었을 때를 살펴보자. 위의 방법을 사용하면, 각 행성을 지나며 측정된 방향벡터에 평행한 두 직선의 교점으로 최적해가 결정된다. 이 때 손실함수값은 0이 된다. 행성이 3개 이상 관측된 경우는 두 직선의

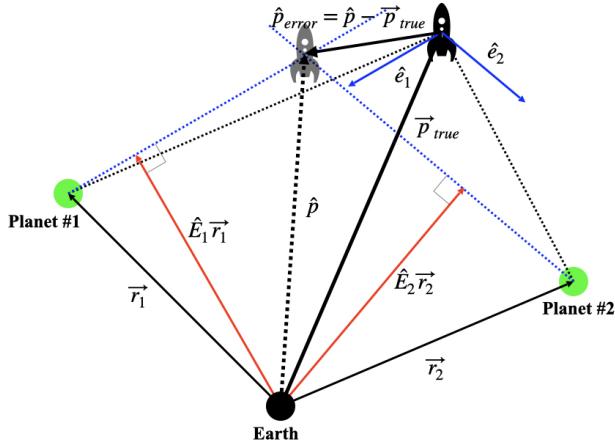


Fig. 6 2개의 행성이 관측된 상황

교점으로 최적해가 결정되지 않는다. Fig. 7과 Fig. 8의 검은 점선으로 표현된  $\hat{p}$ 가 최적해일 때, 빨간 실선 벡터들의 합은 같다. 즉, 2개의 행성을 이용할 때와 비교해서  $e_k$ 에 수직한 방향들 사이에서 조정이 일어난다. 이 조정과정이 위치추정치에 어떤 영향을 미치는지 이후 시뮬레이션에서 관찰해 볼것이다.

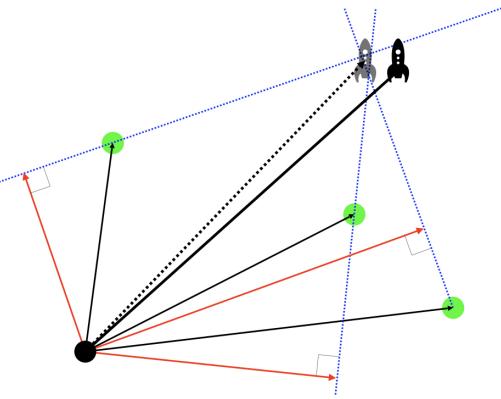


Fig. 7  $\sum \hat{E}_k \vec{r}_k$

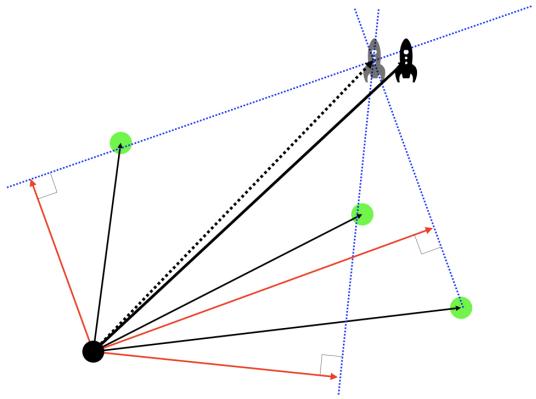


Fig. 8  $\sum \hat{E}_k \hat{p}$

### 3. 가중 최소제곱법(Weighted Least Squares)

각오차( $\epsilon$ )가 커짐에 따라, 관측자-행성 벡터의 관측된 방향벡터 위로 수선의 빌( $\sqrt{|\vec{r} - \vec{p}|^2 - |(\vec{r} - \vec{p})^T \hat{e}|^2}$ )이 함께 커지는 것에 착안하여 손실과 비용함수를 정의했다. 하지만 이 방식은 Fig. 9에서 확인할 수 있듯이, 같은 각오차가 발생하더라도 더 먼거리에 있는 행성의 각오차가 비용함수에 더 많은 영향을 끼치게 된다는 문제가 생긴다. 특히 행성들 사이의 상대거리를 고려했을 때 그 영향은 무시할 수 없다(Fig. 10). 따라서 거리차이에 의한 효과를 보정하기 위해 가중치를 설정해주어야 더 좋은 성능의 항법 알고리즘을 구현할 수 있을 것이다. 측정치를 초 단위로

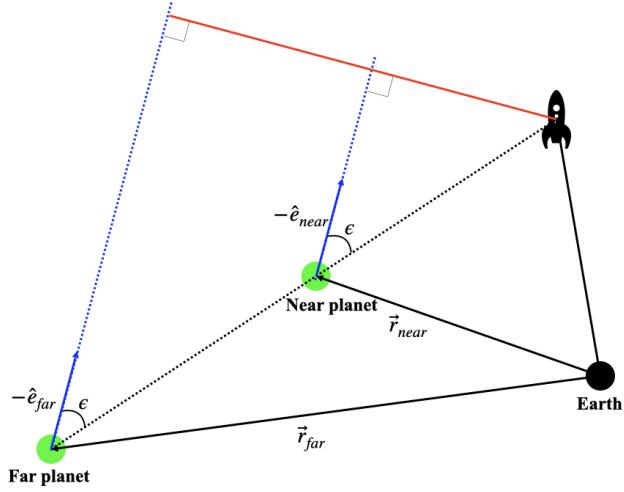


Fig. 9 면 거리와 가까운 거리에 있는 행성

생성한다고 할 때, 관측자-행성 사이의 거리의 변화량은 거리에 비해 크지 않다. 따라서, 손실의 가중치를 이전시간에 구한 최적해  $\hat{p}_{prev}$ 를 활용하여  $w_k = |\vec{r}_k - \hat{p}_{prev}|^{-2}$ 로 설정한다. 가중치의 역할은 관측자-행성간 거리에 따른 각오차의 효과를 비슷하게 조정해주는 것이기 때문에 이전시간에 구한 최적해를 이용하면 추가적인 계산 없이 목적을 달성할 수 있을 것이라 판단하였다. 이에 따라 비용함수를 다음과 같이 재정의할 수 있다.

$$L = \sum_{k=1}^n [(|\vec{r}_k - \vec{p}|^2 - |(\vec{r}_k - \vec{p})^T \hat{e}_k|^2) w_k] \quad (4)$$

이 때, 최적해는

$$\hat{p} = (\sum_{k=1}^n \hat{E}_k w_k)^{-1} (\sum_{k=1}^n \hat{E}_k \vec{r}_k w_k) \quad (5)$$

## D. 시뮬레이션

앞서 정의한 항법 알고리즘의 최적해가 어떤 특성을 보이는지 관찰하기 위한 시뮬레이션을 설계하고 진행한다. 먼저, 시뮬레이션1에서는 거리와 각도 등 관측자와 행성들의 관계에 관련된 값들이 알고리즘의 성능에 어떤 영향을 미치는지 관찰하기 위해 임의로 대상들을 배열한 시뮬레이션을 진행한다. 이후, 그 결과를 통해 시뮬레이션2에서 실제 달 탐사궤도에서 어떤 성능을 보일 것인지 예측하고, 예측과 실제 결과가 동일한지 확인하는 시뮬레이션을 진행한다.

### 1. 시뮬레이션1 - 임의의 배열

임의의 배열로 시뮬레이션을 진행할 때, 공통사항은 다음과 같다.

- 달 탐사궤도에서 안정적으로 관측할 수 있는 행성(지구, 달, 금성, 화성)의 거리수준을 고려하여 시뮬레이션을

	MERCURY	VENUS	EARTH	MOON	MARS	JUPITER	SATURN	URANUS	NEPTUNE	PLUTO
<u>Mass</u> ( $10^{24}$ kg)	0.330	4.87	5.97	0.073	0.642	1898	568	86.8	102	0.0146
<u>Diameter</u> (km)	4879	12,104	12,756	3475	6792	142,984	120,536	51,118	49,528	2370
<u>Density</u> (kg/m <sup>3</sup> )	5427	5243	5514	3340	3933	1326	687	1271	1638	2095
<u>Gravity</u> (m/s <sup>2</sup> )	3.7	8.9	9.8	1.6	3.7	23.1	9.0	8.7	11.0	0.7
<u>Escape Velocity</u> (km/s)	4.3	10.4	11.2	2.4	5.0	59.5	35.5	21.3	23.5	1.3
<u>Rotation Period</u> (hours)	1407.6	-5832.5	23.9	655.7	24.6	9.9	10.7	-17.2	16.1	-153.3
<u>Length of Day</u> (hours)	4222.6	2802.0	24.0	708.7	24.7	9.9	10.7	17.2	16.1	153.3
<u>Distance from Sun</u> ( $10^6$ km)	57.9	108.2	149.6	0.384*	227.9	778.6	1433.5	2872.5	4495.1	5906.4
<u>Perihelion</u> ( $10^6$ km)	46.0	107.5	147.1	0.363*	206.6	740.5	1352.6	2741.3	4444.5	4436.8
<u>Aphelion</u> ( $10^6$ km)	69.8	108.9	152.1	0.406*	249.2	816.6	1514.5	3003.6	4545.7	7375.9
<u>Orbital Period</u> (days)	88.0	224.7	365.2	27.3*	687.0	4331	10,747	30,589	59,800	90,560
<u>Orbital Velocity</u> (km/s)	47.4	35.0	29.8	1.0*	24.1	13.1	9.7	6.8	5.4	4.7
<u>Orbital Inclination</u> (degrees)	7.0	3.4	0.0	5.1	1.9	1.3	2.5	0.8	1.8	17.2
<u>Orbital Eccentricity</u>	0.205	0.007	0.017	0.055	0.094	0.049	0.057	0.046	0.011	0.244
<u>Oblliquity to Orbit</u> (degrees)	0.034	177.4	23.4	6.7	25.2	3.1	26.7	97.8	28.3	122.5
<u>Mean Temperature</u> (C)	167	464	15	-20	-65	-110	-140	-195	-200	-225
<u>Surface Pressure</u> (bars)	0	92	1	0	0.01	Unknown*	Unknown*	Unknown*	Unknown*	0.00001
<u>Number of Moons</u>	0	0	1	0	2	79	82	27	14	5
<u>Ring System?</u>	No	No	No	No	No	Yes	Yes	Yes	Yes	No
<u>Global Magnetic Field?</u>	Yes	No	Yes	No	No	Yes	Yes	Yes	Yes	Unknown
	MERCURY	VENUS	EARTH	MOON	MARS	JUPITER	SATURN	URANUS	NEPTUNE	PLUTO

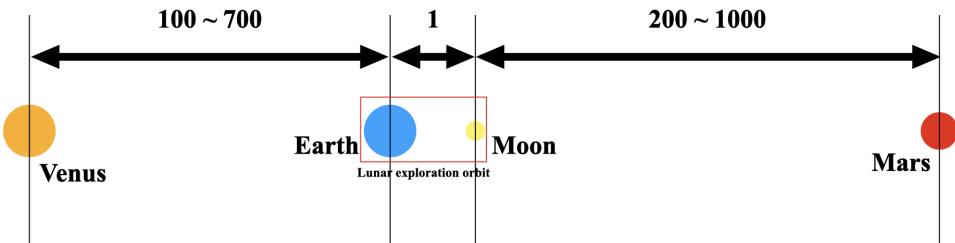
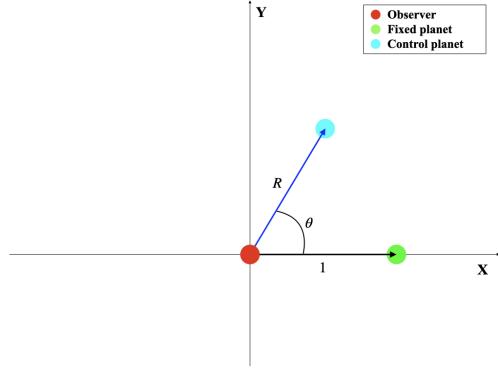


Fig. 10 달 탐사 궤도에서 안정적으로 관측 가능한 행성사이의 거리 수준

진행한다.

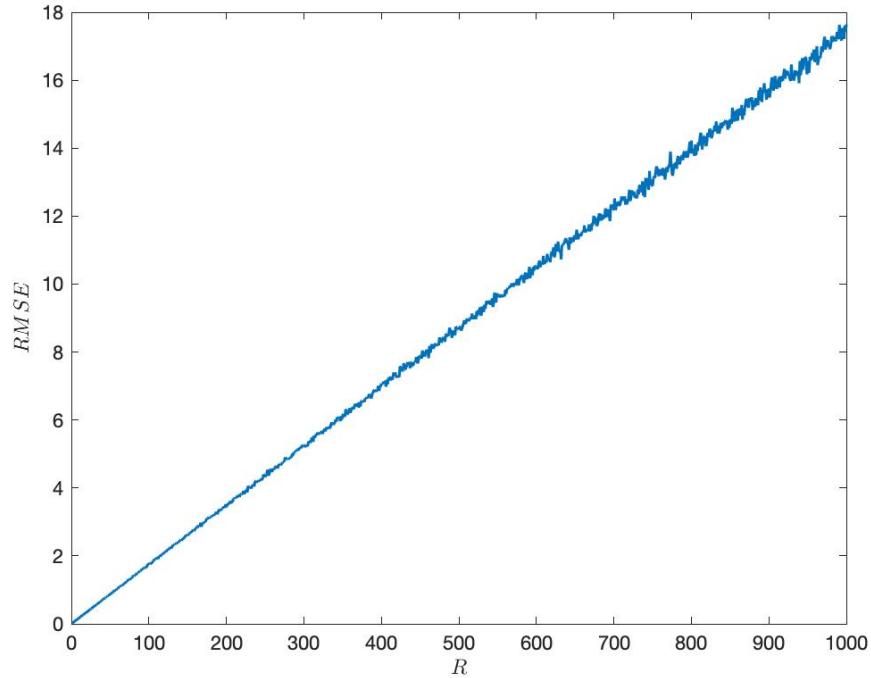
- 단순화를 위해 2D 평면 위에서 진행한다.
- 거리의 단위는 무차원으로 진행한다.
- 별 추적기를 통해 관측한 관측자-행성 방향벡터의 각오차는 표준편차가  $1^\circ$ , 평균이 0인 정규분포를 따르는 것으로 가정하여 진행한다.
- 각 상황에 대해 5000번의 몬테카를로 시뮬레이션을 진행한다.

### <Case #1 - 2개의 행성 관측>



**Fig. 11 Case #1**

Fig. 11에 표현된 것처럼 관측자를 원점, 행성 하나를  $(1, 0)$ 에 고정한 후, 나머지 행성을 거리( $R$ ), 방향( $\theta$ )을 바꾸어가며 위치추정치를 관찰한다. 2개의 행성의 경우 LS(Least Squares)와 WLS(Weighted Least Squares)의 해가 동일하므로, LS를 위치추정 알고리즘으로 채택한다.



**Fig. 12 Case #1의 결과 :  $R$  vs RMSE,  $\theta = 90^\circ$**

$R$ 이 커짐에 따라 선형적으로 오차의 크기가 증가하는 것을 확인할 수 있었으며,  $\theta = 90^\circ$ 일 때 가장 좋은 성능을 보였다. Fig. 14의 시각화를 통해 지상에서 2개의 VOR station을 이용하여 위치추정치를 생성할 때와 같은 모양을 보이는 것을 확인할 수 있다. 특히,  $\theta = 0^\circ, 180^\circ$ 부근, 즉, 관측자와 행성들이 일직선상에 놓일 때 오차가 기하급수적으로 커지는 것을 관찰할 수 있었고, 이것에 대한 설명이 필요하다고 판단하였다.

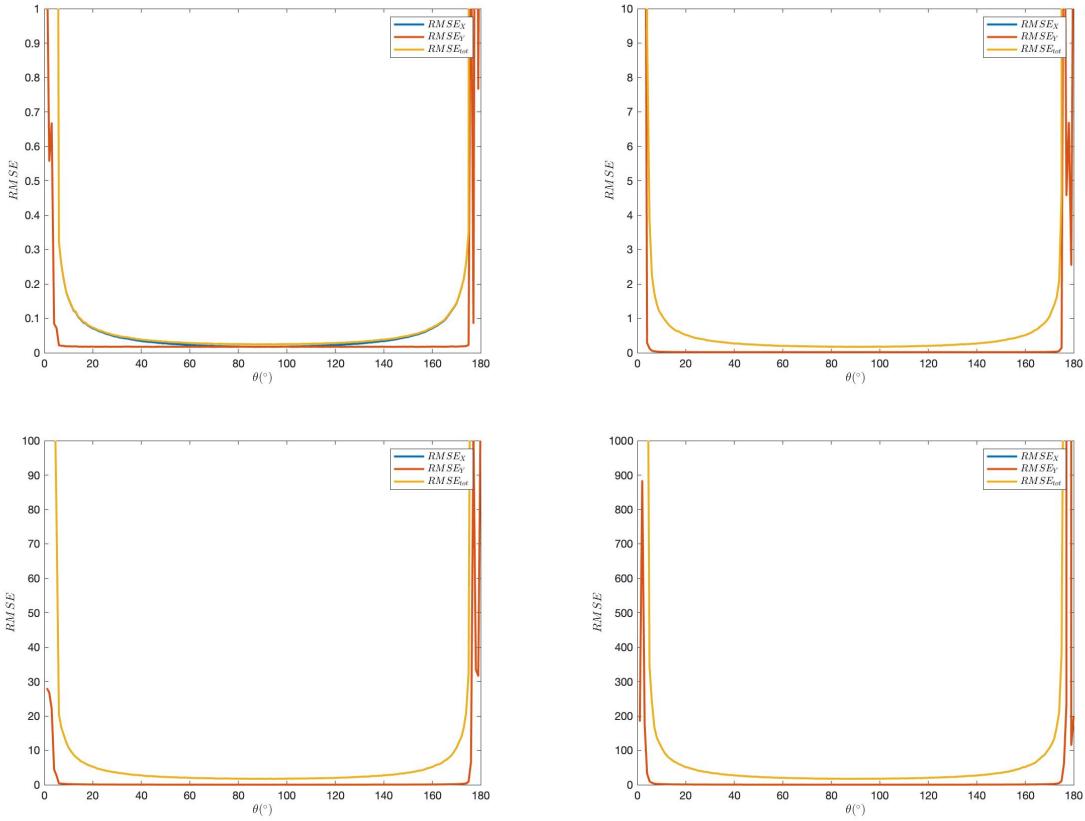


Fig. 13 Case #1의 결과 :  $\theta$  vs RMSE, 원쪽 위부터 차례대로  $R = 1, 10, 100, 1000$

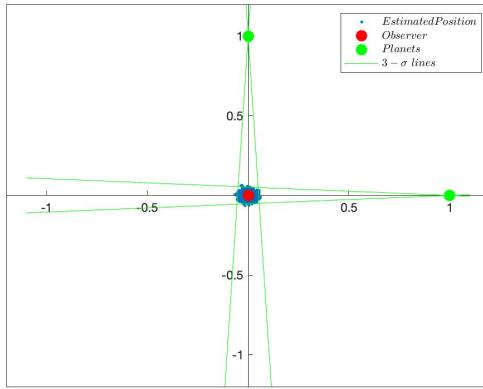


Fig. 14 Case #1의 결과 :  $R = 1, \theta = 90^\circ$

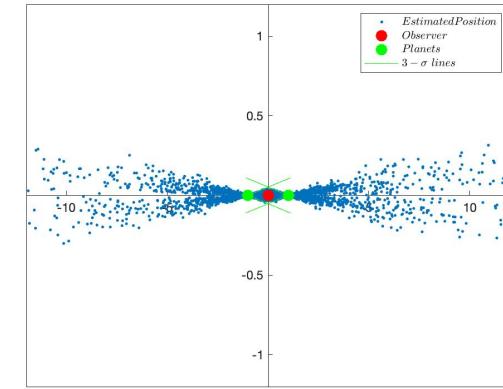


Fig. 15 Case #1의 결과 :  $R = 1, \theta = 1800^\circ$

Fig. 15를 보면,  $3 - \sigma$ 경계선을 벗어나는 지점에 생성된 위치추정치가 다수 발생한 것을 확인할 수 있다. 이것은 Fig. 16와 같이 관측자-행성 방향벡터의 각오차가 서로 엇갈려서 발생했을 때의 위치추정치에 의한 것이다. 이는 일반적인 상황에서 발생하는 오차보다 오차의 크기가 훨씬 크다. 따라서 안정적인 항법을 위해서는 이 특수한 경우의 위치추정치에 대한 관찰이 필요해보인다. 이 연구에서는 앞으로 이것을 특별히 '치명적 오차'라 정의한다.

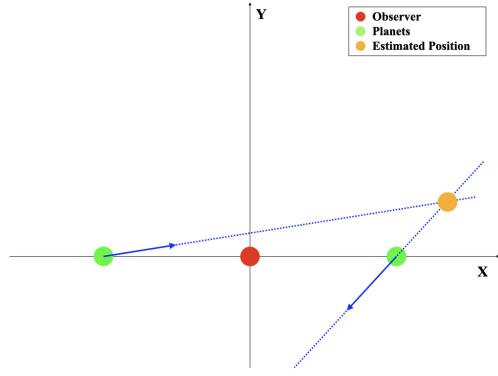


Fig. 16 치명적 오차 발생 경위

<Case #2 - 2개의 행성 관측, 치명적 오차 관찰>

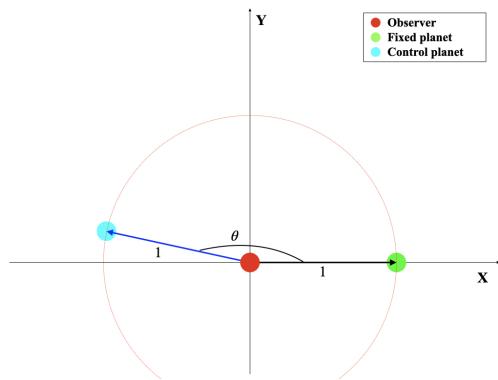


Fig. 17 Case #2

앞서 정의한 치명적 오차의 발생빈도를 확인하기 위해 Case 1에서 오차의 수준의 급격하게 증가한 구간  $0^\circ < \theta \leq 10^\circ$ ,  $170^\circ < \theta \leq 180^\circ$ 에서 치명적 오차의 빈도를 관찰하는 시뮬레이션을 진행한다. 이 때, 거리는 모두 1로 고정한다. Fig. 17에 뺄간 점선으로 표시된 반지름이 1이고, 중심이 원점인 원의 경계를 벗어난 위치추정치를 치명적 오차로 간주한다.

Table 1 Case #2 결과 : 치명적 오차 빈도

치명적 오차의 빈도(%)	$\theta$ 의 범위( $^\circ$ )
< 10	$3.60 < \theta < 176.75$
< 1	$6.61 < \theta < 174.87$
< .1	$8.65 < \theta < 173.39$
< .01	$9.86 < \theta < 170.15$

결과를 통해 치명적 오차의 빈도가 0.01% 이하가 되도록 위치추정치를 얻으려면, 즉, 99.99%로 위치추정치를 신뢰할 수 있으려면 관측자-행성-행성 사이의 각도가  $10^\circ < \theta < 170^\circ$ 인 것이 바람직하다. 이후 추가로 관측 가능한 행성이 존재할 때 치명적 오차의 빈도와 그 크기가 어떻게 변화하는지 관찰해 볼 것이다.

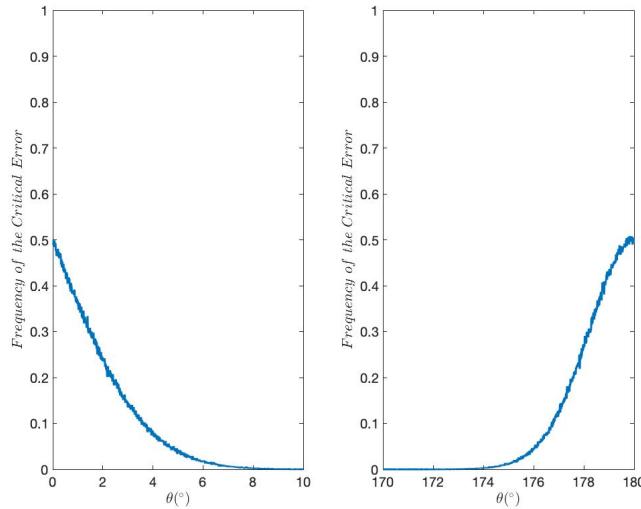


Fig. 18 Case #2의 결과 :  $\theta$  vs 치명적 오차 빈도 그래프

<Case #3 - 3개의 행성 관측>

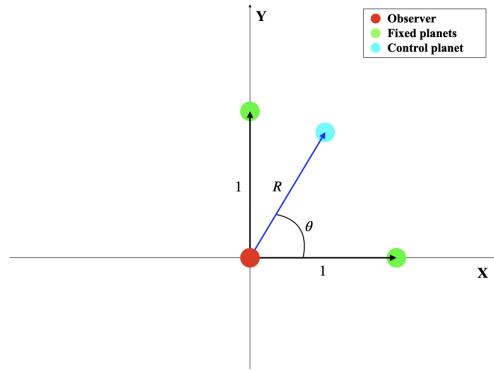


Fig. 19 Case #3

Case #1, #2 에서는 2개의 행성이 관측된 상황에서 위치추정치를 관찰하였다. Case #3, #4 에서는 3개의 행성이 관측된 상황에서 2-Planet(2개의 행성만 이용할 때)/LS/WLS 세 가지 방법을 모두 이용하여 위치추정치를 생성하고 그것들을 비교 및 관찰한다. Case #3 에서는 Fig. 19과 같이 관측자를 원점에, 2개의 행성을 Case #1에서 가장 좋은 성능을 보여줬던 위치인  $(1, 0)$ ,  $(0, 1)$ 에 각각 배치하고, 세 번째 관측된 행성의 위치를 바꾸어가면서 위치추정치를 관찰한다.

결과를 통해 세 번째 행성의 방향은 오차의 크기에 영향을 주지 않는다는 것을 알 수 있었다. 또한, 세 번째 행성이 나머지 두 행성과 비슷한 수준의 거리에 있을 때에는 LS/WLS 모두 2-Planet 보다 좋은 성능을 보였지만, 거리가 증가함에 따라 WLS는 2-Planet과 같은 수준의 오차를 보였고, LS는 오차가 선형적으로 증가해 오히려 2-Planet보다 성능이 떨어졌다. 따라서 3개 이상의 행성을 이용하여 항법을 한다면 LS는 사용하지 않는 것이 바람직하다. 또한,

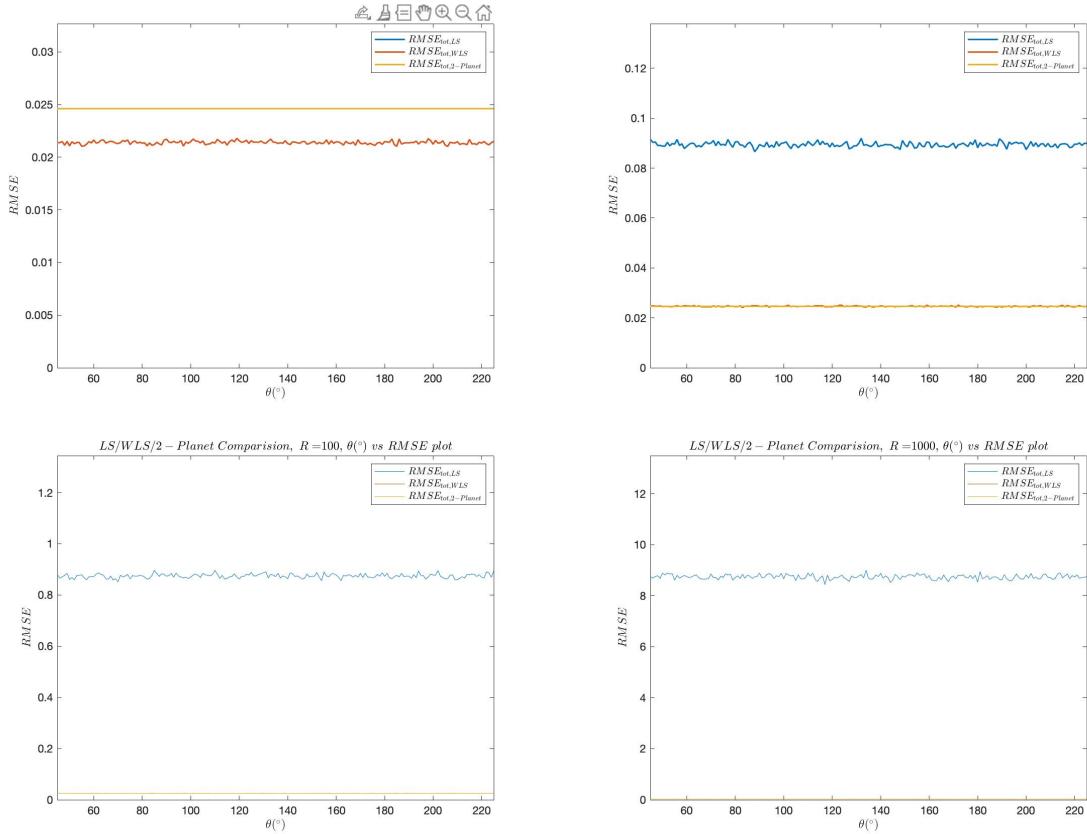


Fig. 20 Case #3의 결과 :  $\theta$  vs RMSE, 왼쪽 위부터 차례대로  $R = 1, 10, 100, 1000$

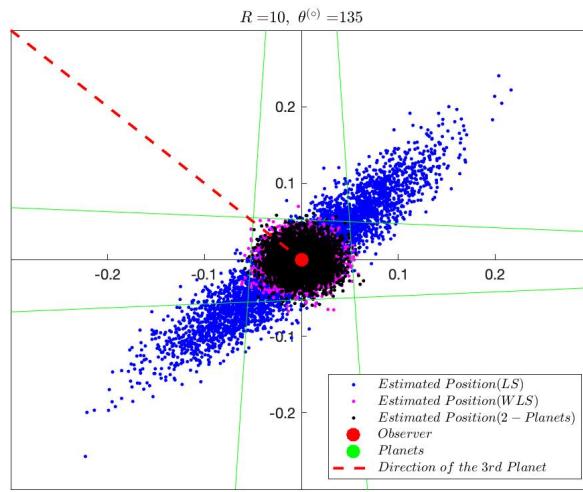


Fig. 21 Case #3의 결과 :  $R = 100, \theta = 135^\circ$  일 때, 위치추정치 시각화

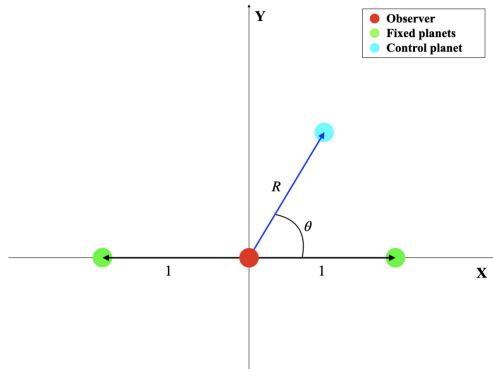
관측자-행성 사이의 거리를 생각해봤을 때 WLS의 경우 가장 가까이 있는 행성 2개로 위치추정치를 생성하는 것과 같은 결과를 얻을 수 있을 것이다. 정리하면, 일반적인 상황에서 3개 이상의 행성을 사용해도 LS/WLS 모두 2개 행

**Table 2 Case #3 결과 :  $R = 100, \theta = 135^\circ$  일 때, RMSE**

	RMSE
2-Planet	0.0246
LS	0.0890
WLS	0.0248

성을 이용하는 것보다 더 좋은 성능을 기대하기는 어려우며, 오히려 계산량만 증가시킨다. 위치추정치를 시각화한 산점도(Fig. 21)를 보면 LS의 경우 세 번째 행성이 비용함수에 미치는 영향이 WLS보다 더 커서 위치추정치가 세 번째 행성 방향에 수직한 방향으로 더 넓게 분포되어있는 것을 확인할 수 있다.

#### <Case #4 - 3개의 행성 관측, 치명적 오차 관찰>



**Fig. 22 Case #4**

Case #2에서는 두 행성의 위치변화에 대한 치명적 오차의 빈도를 관찰했다. Case #4에서는 2개의 행성과 관측자가 일직선상에 배열되어있을 때(가장 치명적 오차의 빈도가 클 때), 세 번째 행성의 거리와 방향을 바꾸어가며 치명적 오차의 빈도와 오차의 크기를 관찰한다.

**Table 3 Case #4 결과 :  $R = 500, \theta = 90^\circ$  일 때, RMSE**

	RMSE
2-Planet	132.7010
LS	8.8434
WLS	2.6857

지구-달 거리와 금성-화성 거리를 고려했을 때, LS는 오히려 치명적 오차의 빈도를 높이는 결과 보여준다. WLS는 방향과 거리에 따라 50% 이내로 치명적 오차의 빈도를 줄일 수 있다. 하지만 Fig. 24과 Table 3를 확인해보면, LS/WLS 모두 위치추정치를 2-Planet 보다 밀집하여 분포하게 한다. 그 결과 두 알고리즘 모두 2-Planet에 비해 오차수준이 유의미하게 줄어드는 것을 확인할 수 있다. 따라서 Case #3, #4를 종합해보았을 때, 3개 이상의 행성을 관측하여 WLS 알고리즘을 활용하면 위치추정치는 관측자와 가장 가까운 2개의 행성에 의해 지배받으며(2개의

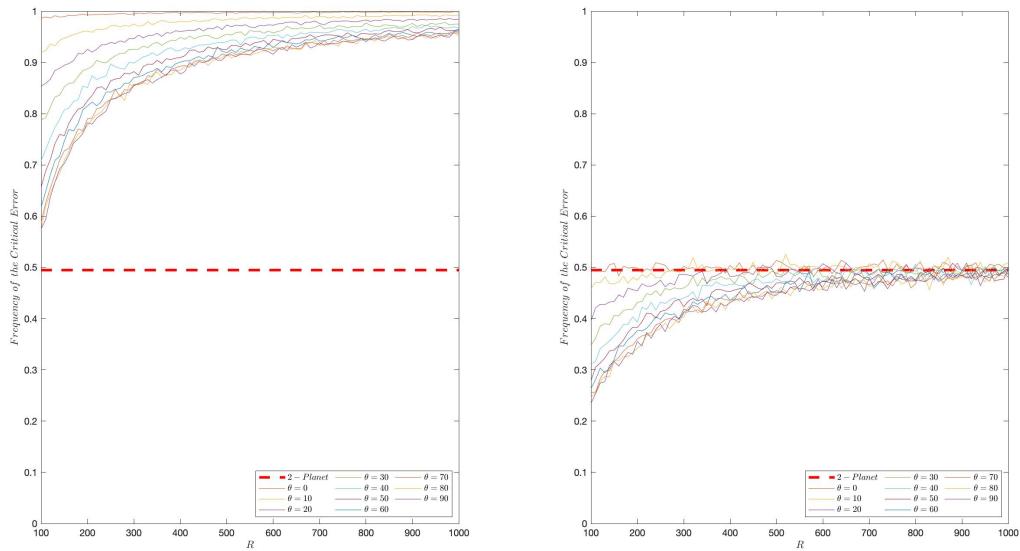


Fig. 23 Case #4의 결과 :  $R, \theta$ 에 대한 LS(왼쪽)/WLS(오른쪽)의 치명적 오차 빈도 그래프

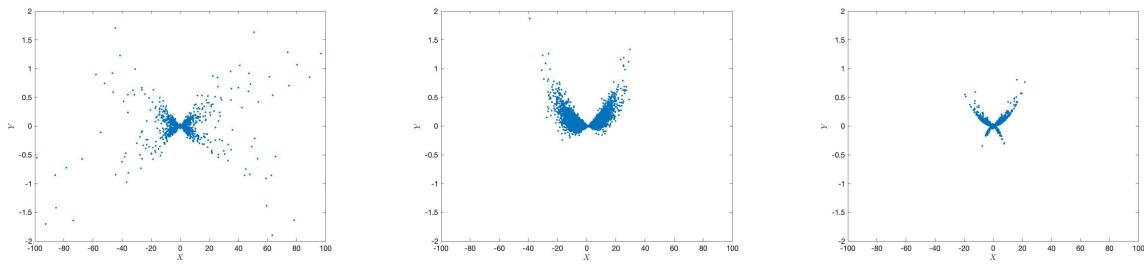


Fig. 24 Case #4의 결과 :  $R = 500, \theta = 90^\circ$  일 때, 2-Planet(왼쪽)/LS(가운데)/WLS(오른쪽) 위치추정치 시각화

행성만 사용할 때와 성능이 비슷하며), 관측자와 가장 가까운 2개의 행성이 서로 일직선상에 놓일 때에는 치명적 오차의 빈도와 수준을 줄여 2-Planet 보다 더 좋은 성능을 기대할 수 있을 것으로 예측할 수 있다.

## 2. 시뮬레이션 2 - 달 탐사궤도

2.4.1. 에서 관찰한 바를 바탕으로 달 탐사 궤도 위에서의 성능을 검증한다. 공통사항은 다음과 같다.

- 2021년 1월 1일 00시 00분에 태원궤도 진입을 시작으로 405890초 이후 2차 추진 전까지 호만전이궤도()에서 시뮬레이션을 진행한다[8].
- 달 궤도는 를 반지름으로 하는 원궤도로 가정하여 진행한다.
- 단순화를 위해 2D 평면(달 궤도면)위에서 진행하며, 금성, 화성의 좌표는 DE405를 이용해 구한 좌표를 달 궤도면에 정사영한 값으로 진행한다.
- 출발 시점에 2D 평면 위에서 지구-금성 거리는, 지구-화성 거리는 이다.
- 관측자-행성 방향벡터의 각오자는 표준편차가  $10 \text{ arcsec}$ 이고, 평균이 인 표준편차로 가정하여 진행한다.

- 2-Planet(지구, 달)/LS/WLS 모두 같은 측정치를 이용하여 위치추정치를 생성한다.

#### <Case #5 - 특정 위치>

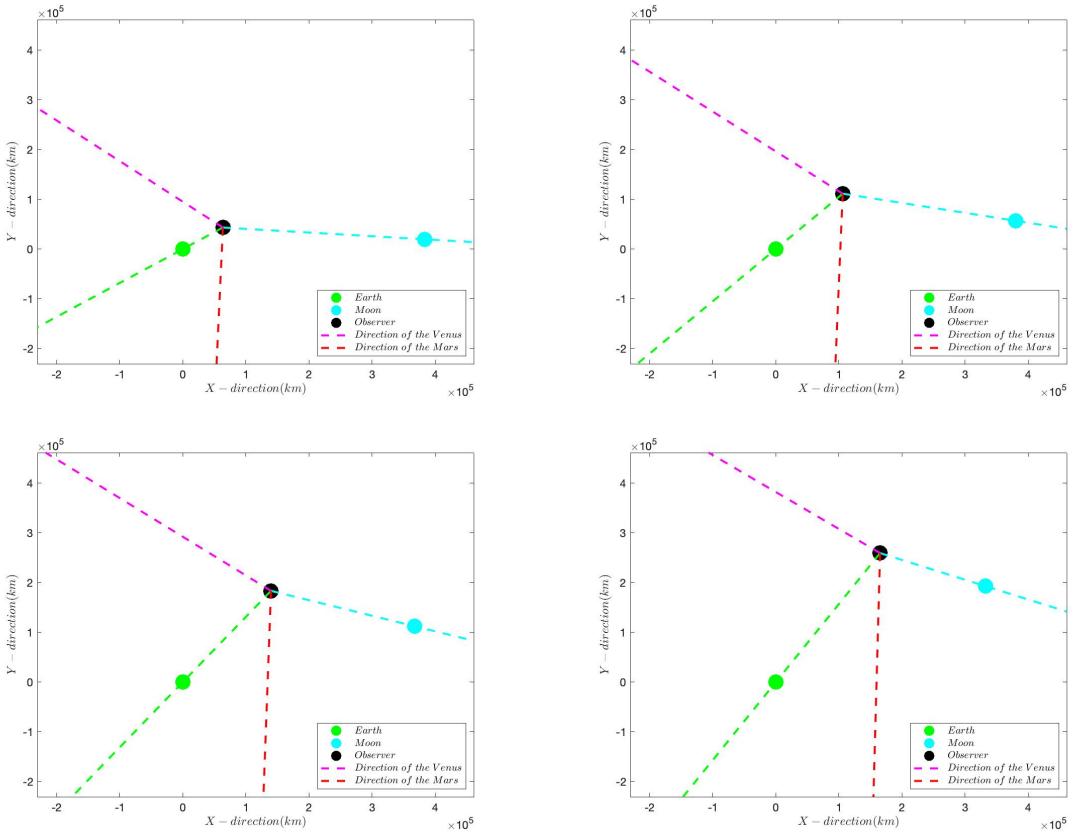
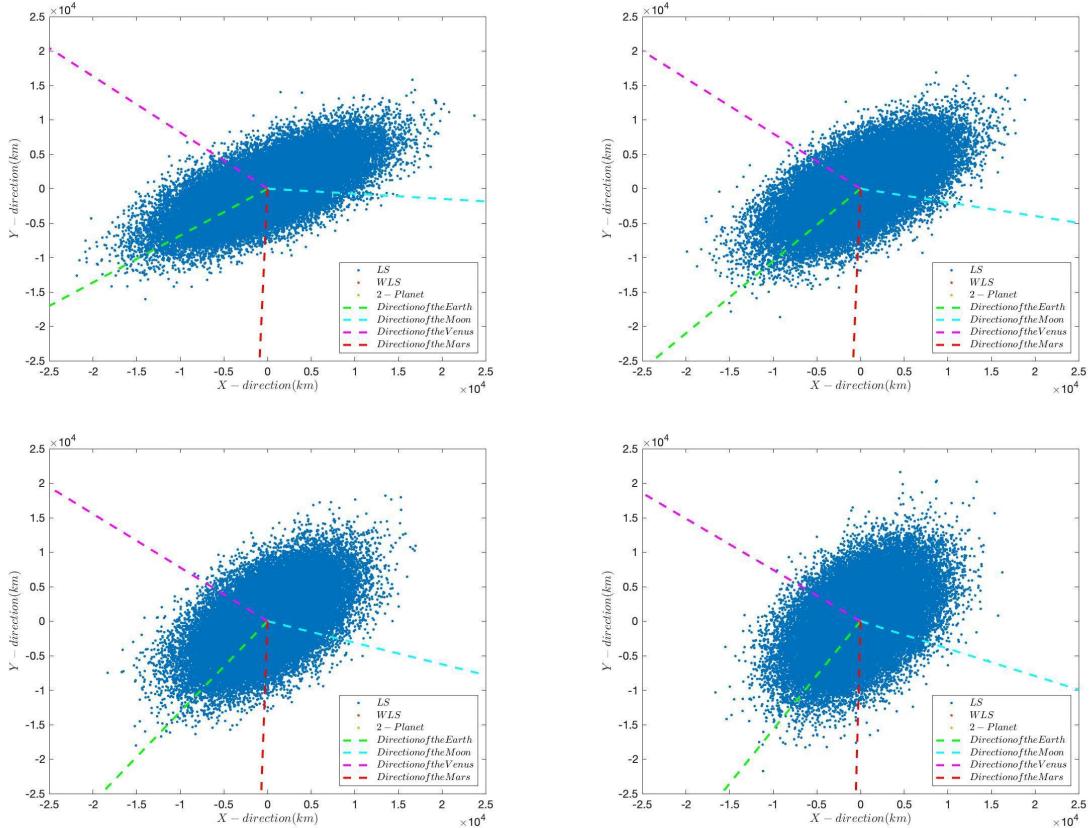


Fig. 25 Case #5 : 원쪽 위에서부터 관측자-지구 사이 거리가  $0.2R, 0.4R, 0.6R, 0.8R$ 일 때 위치관계

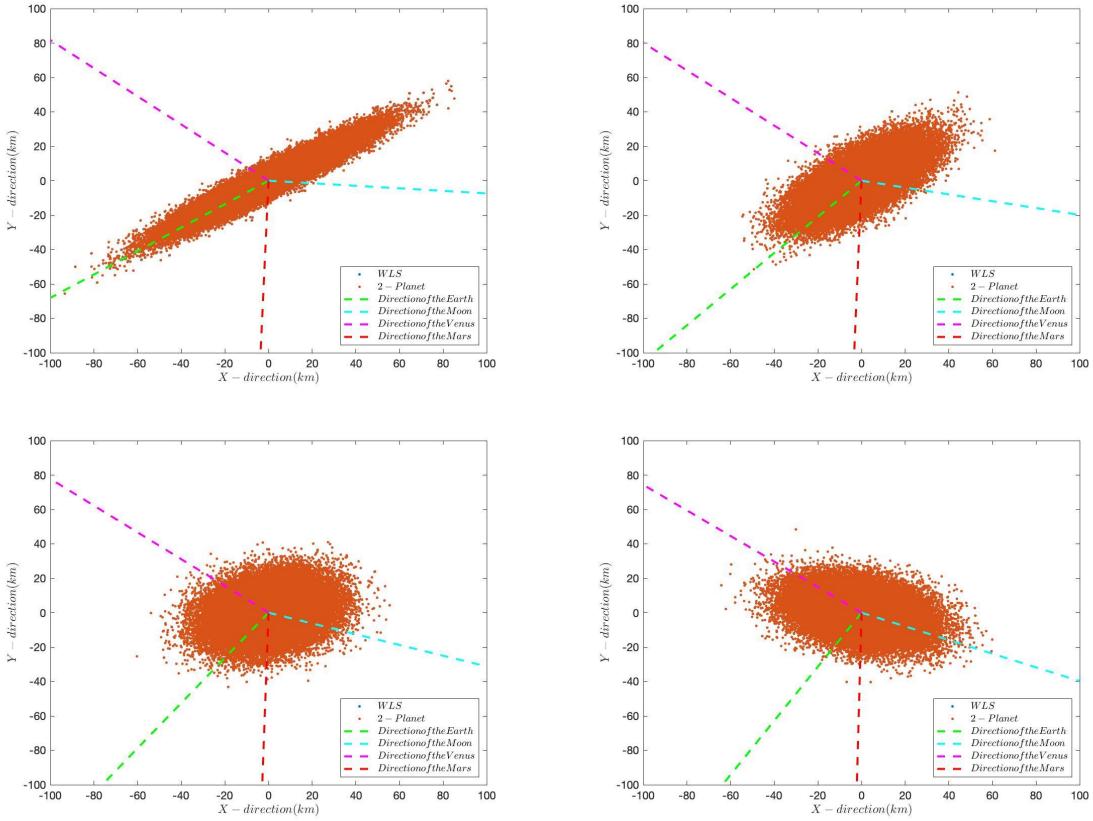
먼저 특정 위치에서 시뮬레이션을 진행한다. 지구중심-달 사이의 거리를  $R (= 384,400\text{km})$ 라고 했을 때, 호만전이 궤도 위에서 지구-관측자 사이의 거리가  $0.2R, 0.4R, 0.6R, 0.8R$ 인 위치에서 50000번의 몬테카를로 시뮬레이션을 진행한다.

**Table 4 Case #5의 결과 : 각 지점에서의 RMSE(km)**

	$0.2R$	$0.4R$	$0.6R$	$0.8R$
2-Planet	25.7124	18.3387	17.1643	17.5970
LS	$6.6460 \times 10^3$	$6.2697 \times 10^3$	$6.0860 \times 10^3$	$6.0938 \times 10^3$
WLS	25.7122	18.3387	17.1642	17.5969



**Fig. 26 Case #5 : 원쪽 위에서부터 관측자-지구 사이 거리가  $0.2R, 0.4R, 0.6R, 0.8R$ 일 때 2-Planet/LS/WLS 위치 추정오차 산점도**



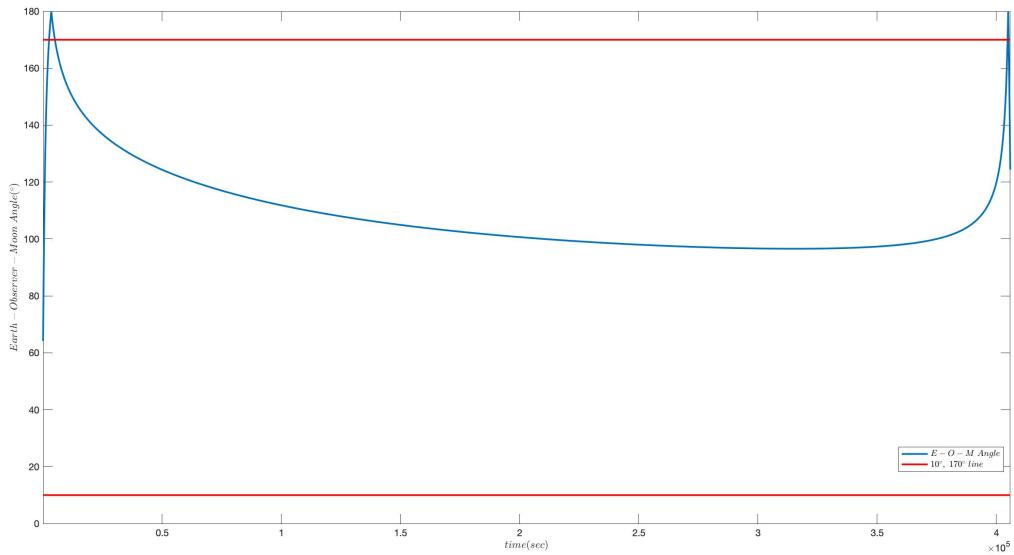
**Fig. 27 Case #5 : 원쪽 위에서부터 관측자-지구 사이 거리가  $0.2R, 0.4R, 0.6R, 0.8R$ 일 때 2-Planet/WLS 위치추정오차 산점도**

각 지점에서 지구-관측자-달이 직선에 가깝도록 배열되어있지 않기 때문에, 치명적 오차가 발생하지 않는다. 또한, 지구, 달과 관측자 사이의 거리보다 금성, 화성과 관측자 사이의 거리수준이  $10^3$ 배 가까이 크기 때문에 LS가 2-Planet/WLS 보다 오차가 크며, 그 크기가  $10^3$ 배 수준의 차이를 보인다. 오차 산점도(Fig. 26, Fig. 27)를 보면, 2-Planet/WLS의 경우 가까이 있는 두 행성(지구와 달, 푸른색 계열 점선)에 의해 산점도의 모양이 결정되며, LS의 경우 더 멀리 있는 두 행성(금성과 화성, 붉은색 계열 점선)에 의해 산점도의 모양이 결정되는 것을 확인할 수 있다. 이것은 앞서 진행한 시뮬레이션1의 결과와 일치한다.

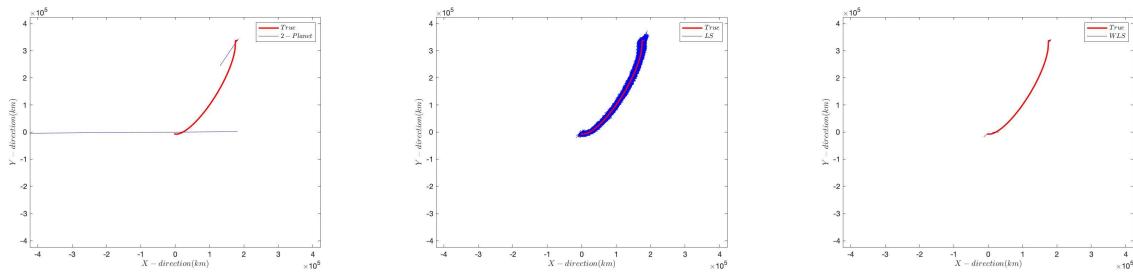
#### <Case #6 - 전체 궤도>

호만전이궤도 위에서 1초마다 위치추정치를 생성하여 시뮬레이션을 진행한다. 관측자와 가장 가깝게 위치하는 행성인 지구, 달과 관측자가 이루는 각을 계산하고, Case #2에서 얻어낸 기준인 를 벗어나는 범위에 있는 시간을 유의하여 오차를 관찰한다. Fig. 28에 그려진 빨간 실선은 치명적 오차가 유의미하게 발생하는 경계를 나타낸 것이다. 또한, WLS의 가중치를 이전시간의 위치추정치를 기반으로 정의했으므로, WLS 방법을 이용할 때 가장 처음에 생성하는 위치추정치는 LS로 구하였다.

Table 4의 WLS 팔호 안에 적혀있는 값은 가장 처음 위치추정치를 생성한 시간을 제외한 구간에서의 RMSE



**Fig. 28 Case #6 : 시간에 대한 지구-관측자-달 사잇각 변화**



**Fig. 29 Case #6의 결과 : 2-Planet(왼쪽)/LS(가운데)/WLS(오른쪽) 위치추정치 시각화**

**Table 5 Case #6의 결과 : 각 알고리즘의 구간별 RMSE(km)**

	2-Planet	LS	WLS
전체구간	$1.3313 \times 10^3$	$6.2395 \times 10^3$	84.1416
치명적 오차 범도가 높은 구간을 제외한 구간	20.1102	$6.2697 \times 10^3$	30.2331(20.1104)

값이다. LS의 경우는 구간에 관계 없이 높은 수준의 오차를 보이며, 2-Planet의 경우 치명적 오차가 유의미하게 발생하는 구간에서 발생한 오차로 인해 전체 구간에서의 오차수준이  $10^2$ 배 가량 증가하였다. 하지만 WLS는 전체 구간에서  $100\text{ km}$ 이하의 안정적인 수준의 오차를 보여준다. 이는 앞에서 진행했던 시뮬레이션들을 통해 예측한 바와 일치한다.

달 탐사선의 설계 단계에서 궤도를 적절히 설계하여 탐사선이 지구, 달 일직선상에 놓이지 않게 한다면 안정적인 성능으로 항법을 할 수 있다. 또한, 추가적으로 행성-탐사선간의 동역학적 모델을 이용하여 필터 이론을 접목하여 사용하거나, INS, GNSS 등 다른 항법 시스템과 함께 운용한다면 더 정확하고 정밀한 항법 시스템을 구축할 수 있을 것이다.

마지막으로, 이 연구에서는 광행차, 빛의 속도에 의한 효과, 자세 오차 등의 오차요인을 고려하지 않았기 때문에, 실제 달 탐사에 적용하기 위해서는 추후 이에 대한 추가적인 고려가 필요해보인다.

## References

- [1] Liebe, C. C., “Accuracy performance of star trackers-a tutorial,” *IEEE Transactions on aerospace and electronic systems*, Vol. 38, No. 2, 2002, pp. 587–599.
- [2] Liebe, C. C., “Star trackers for attitude determination,” *IEEE Aerospace and Electronic Systems Magazine*, Vol. 10, No. 6, 1995, pp. 10–16.
- [3] 주광혁, and 이상률, “별추적기의 기술개요와 개발동향,” *한국항공우주학회지*, Vol. 38, No. 3, 2010, pp. 300–308.
- [4] Williams, D. R., “JPL Planetary and Lunar Ephemerides,” , 2019. URL [https://ssd.jpl.nasa.gov/?planet\\_eph\\_export/](https://ssd.jpl.nasa.gov/?planet_eph_export/).
- [5] Petit, G., and Luzum, B., “IERS conventions (2010),” Tech. rep., Bureau International des Poids et mesures sevres (france), 2010.
- [6] Mortari, D., and Conway, D., “Single-point position estimation in interplanetary trajectories using star trackers,” *Celestial Mechanics and Dynamical Astronomy*, Vol. 128, No. 1, 2017, pp. 115–130.
- [7] Conway, D., and Mortari, D., “Single-Point and filtered relative position estimation for visual docking,” *International Conference on Computational & Experimental Engineering and Sciences (ICCES), Reno, NV*, 2015.
- [8] 강민석, “한국형 달탐사의 항법을 위한 궤도 설계,” 서울대학교 항공우주공학과 졸업논문, 2020.

지도교수 : 기창돈 교수님 (인)