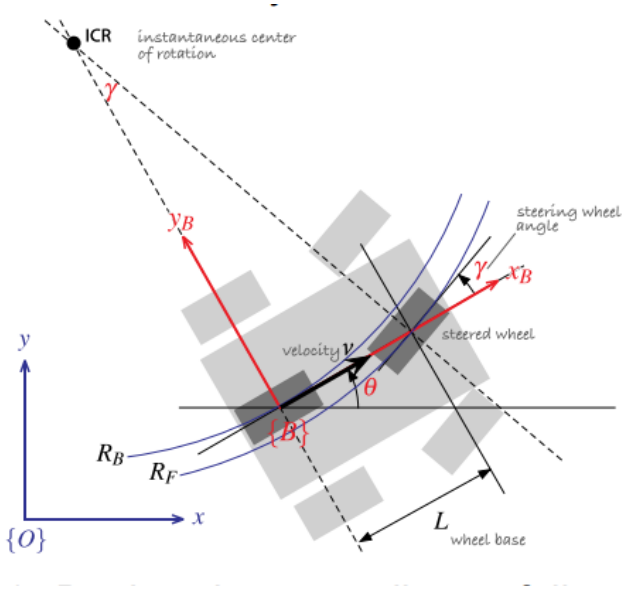
KNU Department of Electronics Engineering

Introduction of Autonomous Vehicles

 # Homework 1

전자전기공학부 석사 과정 윤시원 (2023000853)

1. Consider the bicycle model described in the Lecture note. (Wheel-base: 1m)



$$\dot{x} = v\cos\theta$$

$$\dot{y} = v\sin\theta$$

$$\dot{\theta} = \frac{v}{L}\tan\gamma$$

1.1. Design the controller to follow the line and show the simulation results. Simulation results should include trajectory in 2D plane and the time histories of x, y, theta, v, and gamma. (line equation: 1x-2y+4=0, and initial position: 5,1, heading angle: 0 degree)

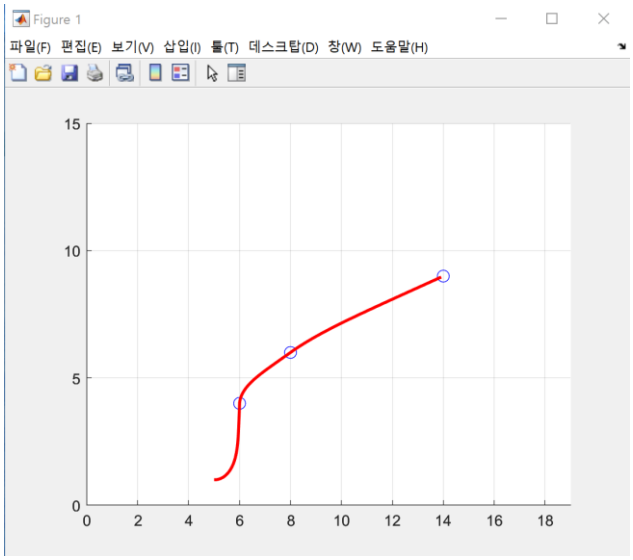


그림 1 : trajectory in 2d plane (also include time histories of x, y)

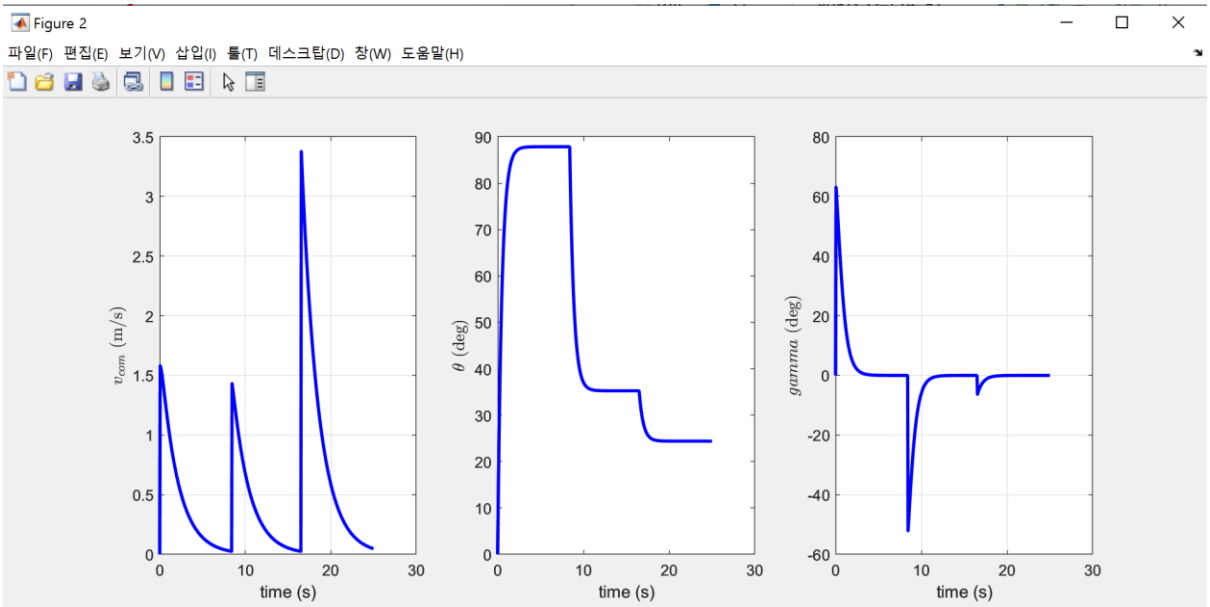


그림 2 : v, theta, gamma

Code

```
%{
자율주행시스템개론 #Homework1 (1.1)
Design the controller to follow the line and show the simulation results.
Simulation
results should include trajectory in 2D plane and the time histories of x, y,
theta, v, and gamma.
(line equation: 1x-2y+4=0, and initial position: 5,1, heading angle: 0 degree)
%}
close all; clear ; clc

%% Waypoint interplation
numWaypoint=4;      % waypoint 개수

% start : 출발 지점, x: 시간별 [x 좌표 y 좌표 heading_angle] list, time : 시간,
v_com : 속도 (순간),
% theta_com : heading angle (순간)
start=[5 1 0]; x(1,:)=start; time(1)=0; v_com(1)=0; theta_com(1)=0;
gamma_com(1)=0;
goal{1}=[6 4];      % 첫번째 목표 지점 좌표
goal{2}=[8 6];      % 두번째 목표 지점 좌표
goal{3}=[14 9];     % 세번째 목표 지점 좌표
numway=3;           % 경로 개수

dt=0.05;            % 시간 변화량 (시간 간격)

flag=true;          % boolean --> true 일 때 실행됨.
index=1; way=1;     % 첫번째 경로
kv=0.5; kh=2.5; L=1;
while(flag)
    v_com(index+1)=kv*sqrt((goal{way}(1)-x(index,1))^2+(goal{way}(2)-
x(index,2))^2); % 필요한 순간 속도 계산
```

```
    theta_com(index+1)=atan2(goal{way}(2)-x(index,2),goal{way}(1)-
x(index,1));       % 필요한 순간 heading angle 계산

    dx(index,:)=v_com(index+1)*[cos(x(index,3)),sin(x(index,3))]; % 좌표의 변화량
계산
    dtheta(index)=kh*(theta_com(index+1)-x(index,3)); % heading angle 변화량 계산

    g_com(index+1)=atan(dtheta(index)*L/v_com(index+1));     % 필요한 순간 steering
angle 계산


    x(index+1,3)=x(index,3)+dt*dtheta(index);
    x(index+1,1:2)= x(index,1:2)+dt*dx(index,:);
    time(index+1)=time(index)+dt;
    index=index+1;

    if norm(x(index,1:2)-goal{way})<0.05
        way=way+1;
        if way==numway && norm(x(index,1:2)-goal{numway})<0.05
            break;
        end
    end
    if time(index)>25
        break;
    end
end
figure(1)
xlim([0 15]); ylim([0 15]); hold on;
for i=1:numway
     plot(goal{i}(1),goal{i}(2),'ob','MarkerSize',8);
end

plot(x(:,1),x(:,2),'-r','LineWidth',2); hold off; axis equal; grid on;

figure(2)
subplot(1,3,1); plot(time,v_com,'-b','LineWidth',2);
                xlabel('time (s)');
ylabel('$v_{com}$ (m/s)','Interpreter','latex');
                grid on;
subplot(1,3,2); plot(time,x(:,3)*180/pi,'-b','LineWidth',2);
                xlabel('time (s)'); ylabel('$\theta$ (deg)','Interpreter','latex');
subplot(1,3,3); plot(time, g_com, '-b','LineWidth',2); grid on;
                xlabel('time (s)'); ylabel('$gamma$ (deg)','Interpreter','latex')
```

위는 LMS에 교수님께서 올려 주신 example code 를 참고하여 작성한 매트랩 코드이다. 감마의 경우 주어진 식에 따라 theta의 순간적인 변화량(dtheta)에 대한 식을 감마에 대한 식으로 다시 작성한 후 코드로 작성해 주었다. 다음이 해당 부분이다.

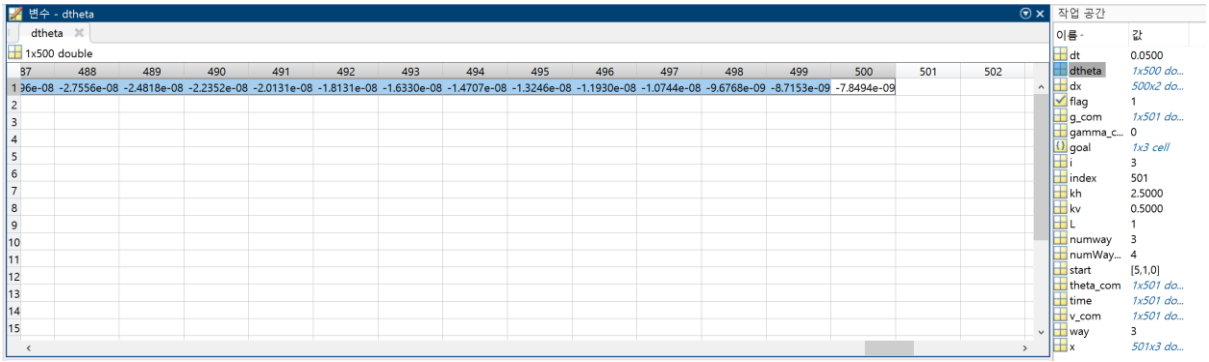
```
g_com(index+1)=atan(dtheta(index)*L/v_com(index+1));                    % 필요한
순간 steering angle 계산
```

해당 코드는 단계별 목표하는 지점을 설정한 후 각 파라미터별 순간적으로 필요한 양을 계산해 주면서 매시간 조금씩 갱신해 나가는 방식의 자전거 모델을 구현하고 있다. 마지막 goal 에 도달

한 이후에는 주어진 line equation 에 대해 계속해서 따라갈 수 있어야 하는데, 이를 위해서는 충족해야 하는 조건이 있다. 첫째로, 마지막 goal 도달 이후 theta(heading angle) 과 gamma(steering angle) 의 차이를 동일하게 유지해야 한다. 왜냐하면 선형적으로 계속 동일한 기울기의 선 위를 나아가기 때문이다. 둘째로, 두 값 모두 변화량이 없어야 한다. 첫번째 조건을 만족한다고 가정하였을 때, 두 번째 조건은 둘 중 한 가지 값만 만족해도 된다.

첫번째 조건은 그래프 값만 봐도 이미 만족한 것을 쉽게 확인 가능하다. 미세 조정을 위해 goal 자체 설정을 아예 처음부터 line equation 에 가깝게 했다. Goal1 은 line equation 위의 지점은 아니지만, line equation 에 가까운 점으로 지정했다. 또한 goal2 에서 이미 line equation 위에 올라가도록 했고, goal3 에서 한 번 더 line equation 위의 점을 지나게 했다. 따라서 goal2 와 goal 3 사이의 값들을 통해 이후의 변화도 예측 가능한데, 그래프를 보면 두 값 모두 해당 지점에서 변화가 미세한 것을 확인 가능하다.

두 번째 조건은 매트랩에서 출력해주는 값을 통해 쉽게 확인할 수 있는데, 아래 캡쳐 자료를 참고하면, 변화량이 10의 -8승 정도의 단위로 매우 미세하다. 따라서 두 조건 모두 충족함을 확인하였다.



1.2. Design the controller to move a specific pose and show the simulation results. Simulation results should include trajectory in 2D plane and the time histories of x, y, theta, v, and gamma. (initial position: 5,5, heading angle: 90 degrees, final pose: 5,9 heading angle: 0 degree)

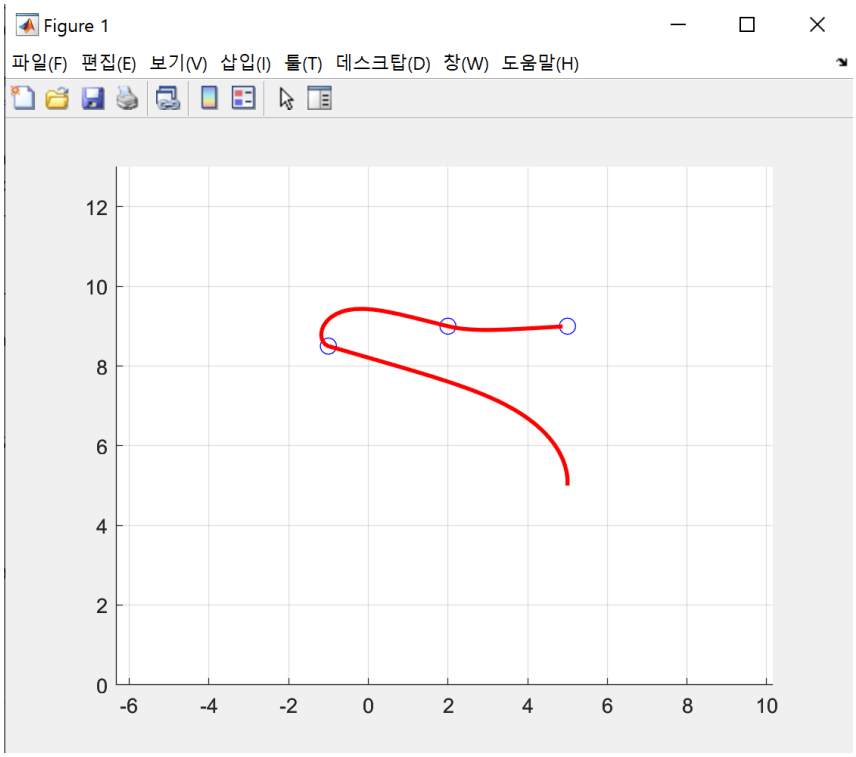


그림 3 : trajectory in 2d plane (also include time histories of x, y)

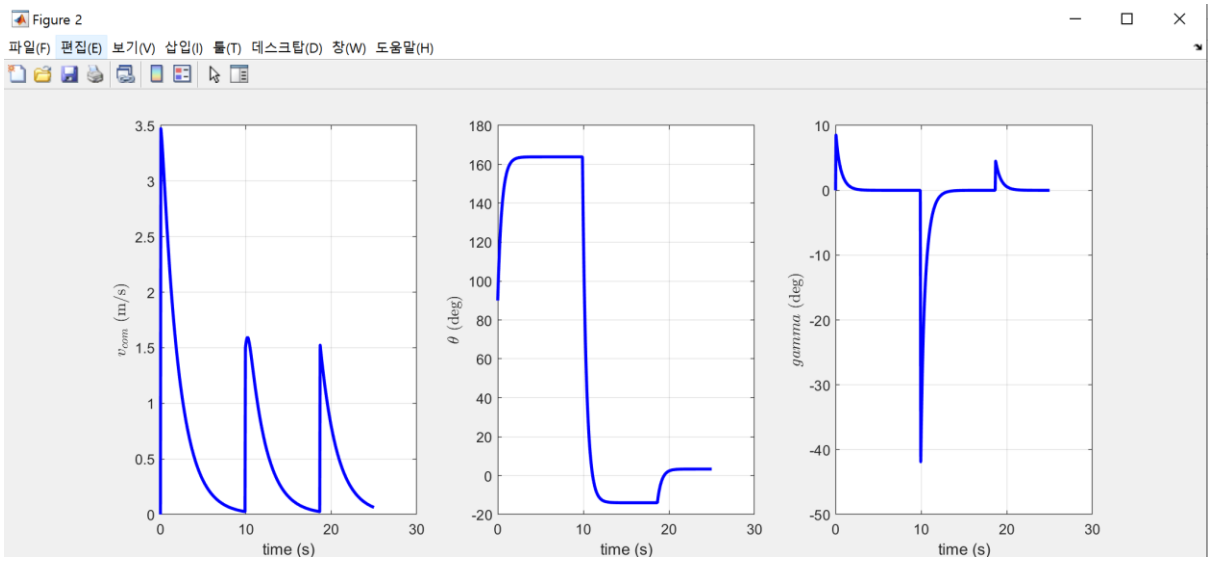


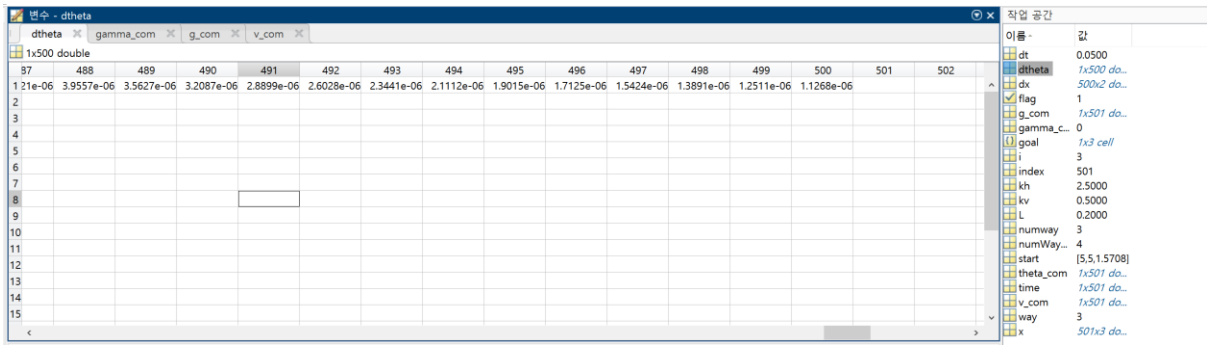
그림 4 : v, theta, gamma

코드는 대체로 1.1 과 같고, 아래 부분만 다르다.

```
% start : 출발 지점, x: 시간별 [x 좌표 y 좌표 heading_angle] list, time : 시간,
v_com : 속도 (순간),
```

```
% theta_com : heading angle (순간)
start=[5 5 deg2rad(90)]; x(1,:)=start; time(1)=0; v_com(1)=0; theta_com(1)=0;
gamma_com(1)=0;
goal{1}=[-1 8.5];      % 첫번째 목표 지점 좌표
goal{2}=[2 9];         % 두번째 목표 지점 좌표
goal{3}=[5 9];         % 세번째 목표 지점 좌표
```

원리도 동일하다. Dtheta 값만 첨부하도록 하겠다.



마찬가지로 10의 -6 승 정도로 매우 작다. 즉, 1.1 에서 언급한 두 조건 모두를 만족함으로써 문제에서 주어진 조건을 만족함을 알 수 있다.

2. Consider the camera model (focal length=0.015) and the pixels are 10 μm square and the pixel array is 1280 × 1024 pixels with its principal point at image-plane coordinate (640, 512). Given the P points in world frame below, show the oblique view of the plane with the camera pose 0.9 rad rotation around y axis and (-1,0,0.5) meter translation (See the lecture note).

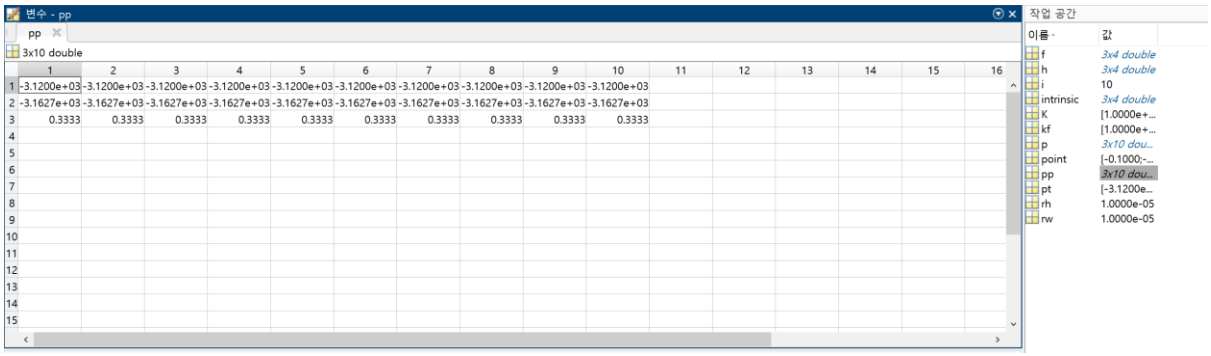
CODE

```
%{
자율주행시스템개론 #Homework1 (2)
Consider the camera model (focal length=0.015)
and the pixels are 10 μm square and the pixel array is 1280 × 1024 pixels
with its principal point at image-plane coordinate (640, 512).
Given the P points in world frame below,
show the oblique view of the plane
with the camera pose 0.9 rad rotation around y axis
and (-1,0,0.5) meter translation.

p =
-0.1000 -0.1000 -0.1000       0       0       0   0.1000   0.1000   0.1000   0.1000
-0.1000       0  0.1000  -0.1000       0  0.1000  -0.1000  -0.1000        0   0.1000
 1.0000  1.0000  1.0000  1.00000  1.0000  1.0000   1.0000   1.0000   1.0000   1.0000
%}
p = [-0.1000 -0.1000 -0.1000 0 0 0 0.1000 0.1000 0.1000 0.1000;
-0.1000 0 0.1000 -0.1000 0 0.1000 -0.1000 -0.1000 0 0.1000;
 1.0000 1.0000 1.0000 1.00000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000];
```

```
for i =1:10
    point = p(:,1);
    rw = (1e-5);
    rh = (1e-5);
    K = [1/rw 0 640; 0 1/rh 512; 0 0 1];
    f = [1 0 0 0; 0 1 0 0; 0 0 1 0];
    h = [1 0 0 0; 0 1 0 0; 0 0 1 0];
    kf = K*f;
    kf = kf(:,1:3);
    intrinsic=kf*h;
    point = [point;1];
    pt = diag([1/3, 1/3, 1/3])*intrinsic*point;
    pp(:,i)=pt
end
```

강의 자료 lecture 05 를 참고하여 작성한 코드이다.



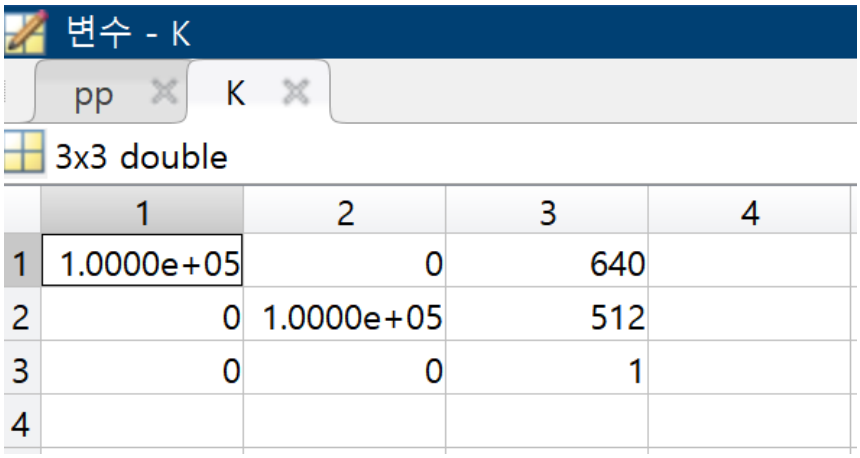
변수 - pp

pp

3x10 double

	1	2	3	4	5	6	7	8	9	10
1	-3.1200e+03	-3.1200e+03	-3.1200e+03	-3.1200e+03	-3.1200e+03	-3.1200e+03	-3.1200e+03	-3.1200e+03	-3.1200e+03	-3.1200e+03
2	-3.1627e+03	-3.1627e+03	-3.1627e+03	-3.1627e+03	-3.1627e+03	-3.1627e+03	-3.1627e+03	-3.1627e+03	-3.1627e+03	-3.1627e+03
3	0.3333	0.3333	0.3333	0.3333	0.3333	0.3333	0.3333	0.3333	0.3333	0.3333
4										

결과는 위와 같다.



변수 - K

pp K

3x3 double

	1	2	3	4
1	1.0000e+05	0	640	
2	0	1.0000e+05	512	
3	0	0	1	
4				

변수 - f

pp K f h

3x4 double

	1	2	3	4	5
1	1	0	0	0	
2	0	1	0	0	
3	0	0	1	0	
4					
5					

변수 - kf

pp K f h kf

3x3 double

	1	2	3	4
1	1.0000e+05	0	640	
2	0	1.0000e+05	512	
3	0	0	1	
4				

변수 - h

pp K f h kf

3x4 double

	1	2	3	4	5
1	1	0	0	0	
2	0	1	0	0	
3	0	0	1	0	
4					
5					

변수 - intrinsic

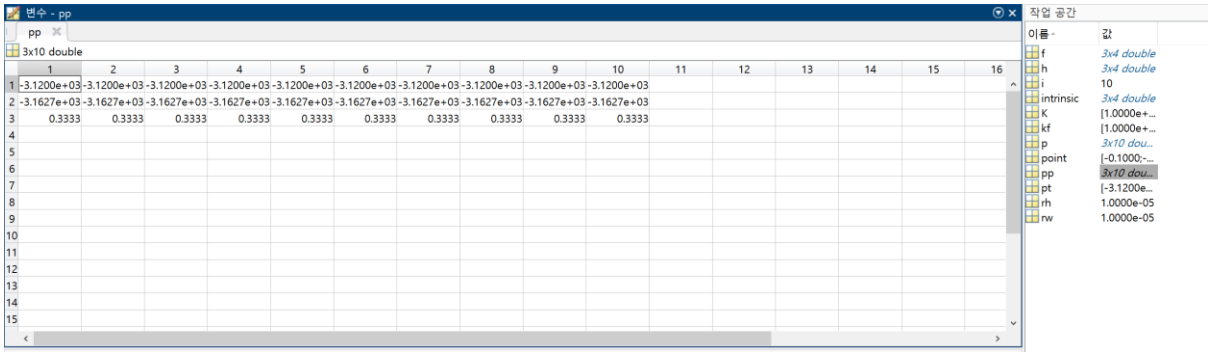
pp K f h kf intrinsic

3x4 double

	1	2	3	4	5
1	1.0000e+05	0	640	0	
2	0	1.0000e+05	512	0	
3	0	0	1	0	
4					

K 는 calibration matrix 이며, 픽셀 크기와 principal point 를 반영한다. F 는 focal length 와 관련한 매트릭스이다. K, f, h 는 결국 intrinsic parameter matrix 를 분해한 결과이고, 이 공식을 역산해 k, f, h 를 알면 intrinsinc parameter 를 알 수가 있다. 따라서 그렇게 구해 주면 위와 같고, 최종

결과는 다음과 같다.



변수 - pp

pp

3x10 double

	1	2	3	4	5	6	7	8	9	10
1	-3.1200e+03	-3.1200e+03	-3.1200e+03	-3.1200e+03	-3.1200e+03	-3.1200e+03	-3.1200e+03	-3.1200e+03	-3.1200e+03	-3.1200e+03
2	-3.1627e+03	-3.1627e+03	-3.1627e+03	-3.1627e+03	-3.1627e+03	-3.1627e+03	-3.1627e+03	-3.1627e+03	-3.1627e+03	-3.1627e+03
3	0.3333	0.3333	0.3333	0.3333	0.3333	0.3333	0.3333	0.3333	0.3333	0.3333
4										