# Fast-Planner Dev Tests

## 

## Intro

Fast-Planner is a robust and computationally efficient planning system that enables quadrotor fast flight in complex unknown environments.

Fast Planner was originally developed by \*\*HKUST-Aerial-Robotics Group\*\*

Our goal in this project is to be able to navigate indoors around different obstacles.

In order to do so we configured the Kinodynamic path searching algorithm to work with Ardupilot and GAZEBO SITL.

After configuring the tf\_kino algorithm, we encountered two main issues:

1. Errors between Fast Planner planned path and drone's actual path (Drone control error).

2. Non-trivial path planned by the Fast-Planner (Path error)

We divide those two issues

As a result, we decided to start running tests in order to try optimizing the algo parameters and the drone PID parameters.

**This paper summarizes our tests**

## Parameters and Metrics

**The Fast-Planner parameters we are going to test are:**

* max\_vel - the planned path maximum velocity possible
* max\_acc - the planned path maximum acceleration possible
* dt\_yaw - delta time in segment of calculating yaw and yaw rate
* lambda1 - Path smoothness weight for optimization
* lambda2 - Distance from obstacles weight for optimization
* lambda3 - Path feasibility weight for optimization
* lambda4 -Endpoint weight for optimization
* lambda7 - Yaw b-spline waypoints weight for optimization

**The drones parameters we are going to test are:**

* PSC\_POSXY\_P - XY position controller P gain
* ACRO\_YAW\_P - Converts pilot yaw input into a desired rate of rotation. Higher values mean faster rate of rotation.
* PSC\_VELXY\_P - XY velocity (horizontal) P gain
* PSC\_VELXY\_I - XY velocity (horizontal) I gain
* PSC\_VELXY\_D - XY velocity (horizontal) D gain

You can see details about ardupilot parameters here: <https://ardupilot.org/copter/docs/parameters.html>

**The metrics we are going to use are:**

* Euclidean error - position euclidean error in 3 dimensions of planned path relative to drone odometry.
* Velocity euclidean error - velocity euclidean error in 3 dimensions of planned path relative to drone odometry.

## Table of Tests

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Parameters** | **Ranges**  **(start,stop,step)** | **Issue** | **Constants** | **Worlds** | **Waypoints** | **Date** | **Comments** |
| 1 | dt\_yaw  max\_vel | (0.3, 1.1, 0.1)  (1, 6, 1) | Drone control error | lambda1=10.0  lambda2=0.8  lambda3=10e-5  lambda4=0.1  lambda7=100  PSC\_POSXY\_P=1 | iris\_no\_corridor | (-7,0,1)  (0,0,1)  (0,-7,1) | 21/10/20 | max\_acc = max\_vel /3  Crashed at max\_vel=2.0  **Need to rerun** |
| 2 | max\_vel  max\_acc | (1, 8, 1)  (0.5, 4, 0.5) | Drone control error | lambda1=10.0  lambda2=0.8  lambda3=10e-5  lambda4=0.1  lambda7=100  PSC\_POSXY\_P=1  dt\_yaw=0.9 | iris\_no\_corridor | (10,10,1)  (10,-10,1)  (-10,-10,1)  (-10,10,1) | 25/10/20 | Crashed at max\_vel = 8, max\_acc = 3.5 |
| 3 | PSC\_POXY\_P  max\_vel | (1, 3, 0.25)  (1, 5, 1) | Drone control error | lambda1=10.0  lambda2=0.8  lambda3=10e-5  lambda4=0.1  lambda7=100  dt\_yaw=0.9 | iris\_no\_corridor | (10,10,1)  (10,-10,1)  (-10,-10,1)  (-10,10,1) | 27/10/20 | max\_acc = max\_vel /3 |
| 4 | lambda1  lambda2  lambda3  PSC\_POXY\_P | [1, 10, 100]  [0.1, 1, 10, 100]  [0.001, 0.1, 1]  [1, 2] | Path error | dt\_yaw = 1.0  Max\_vel = 3.0  Max\_acc = 1.0  lambda4=0.1  lambda7=0.001 | obstacles | (20, 0, 1) | 1/11/20 | Crashed at :  lambda1=100  lambda2=100  lambda3=0.1  PSC\_POXY\_P=1  Lambdas values are in logarithmic scale |
| 5 | dt\_yaw  lambda7 | (0.1, 1.1, 0.3)  [0.1, 1, 10, 100 | Drone control error  Path error | lambda1=10.0  lambda2=0.8  lambda3=10e-5  lambda4=0.1  Max\_vel = 3.0  Max\_acc = 1.0 | obstacles | (20, 0, 1) | 2/11/20 | Lambda7 values are in logarithmic scale |
| 6 | PSC\_POSXY\_P | (1, 2.75, 0.25) | Drone control error | lambda1=10.0  lambda2=0.8  lambda3=10e-5  lambda4=0.1  lambda7=10  dt\_yaw=0.5  Max\_vel = 3.0  Max\_acc = 1.0 | obstacles | (20, 0, 1) | 2/11/20 |  |
| 7 | PSC\_POSXY\_P | (2, 5, 0.25) | Drone control error | lambda1=10.0  lambda2=0.8  lambda3=10e-5  lambda4=0.1  lambda7=10  dt\_yaw=0.5  Max\_vel = 3.0  Max\_acc = 1.0 | obstacles | (20, 0, 1) | 3/11/20 | **We add previous test results to this test result** |
| 8 | ACRO\_YAW\_P | (1, 10, 1) | Drone control error | lambda1=10.0  lambda2=0.8  lambda3=10e-5  lambda4=0.1  lambda7=10  dt\_yaw=0.5  Max\_vel = 3.0  Max\_acc = 1.0  PSC\_POSXY\_P=3 | obstacles | (20, 0, 1) | 5/11/20 |  |
| 9 |  |  | Final Test | lambda1=10.0  lambda2=0.8  lambda3=10e-5  lambda4=0.1  lambda7=10  dt\_yaw=0.5  Max\_vel = 3.0  Max\_acc = 1.0  PSC\_POSXY\_P=3 | iris\_included | (32, 0, 1)  (-32, 0, 1) | 5/11/20 |  |

## Summary

In order to determine max\_vel and max\_acc parameters we began running dev tests for drone control error, using empty world (no obstacles). In test #2 we obtained that the errors between the Fast-planner’s planned path and the drone's actual path is increasing linearly with max\_vel and max\_acc parameters. In addition, we notice from examining the path manually that for max\_acc values lower than 1 the drone’s behavior is occasionally not smooth. Considering that, we will determine **max\_acc = 1 and max\_vel = 3.**

Combining both test #3 and test #7, we obtained that the drone control error is decreasing when PSC\_POSXY\_P is increasing until PSC\_POSXY\_P is reaching approximately value of 3. **Hence we determine PSC\_POSXY\_P value to be 3.**

In order to determine lambda1-3 we ran test #4 with different values using obstacles world. We didn’t obtain any correlation between lambdas parameters and drone control errors or path error. We ran manual tests and obtained ranges for lambdas parameters: **lambda1 - 1:100, lambda2- 0.1:100, lambda3: 0.001:1.**

In order to determine dt\_yaw and lambda7 parameters we ran test #5 with different values using obstacles. We found that the minimum drone control error is obtained for **lambda7=1 and dt\_yaw=0.7**. After examining these parameters manually we obtained that value for **dt\_yaw value of 0.5-0.6** the drone yaw rate is higher and the drone is more reactive, while drone control error is approximately the same.

We tried to determine ACRO\_YAW\_P parameters similarly to PSC\_POSXY\_P but we didn’t find any correlation between ACRO\_YAW\_P values to yaw errors, hence we will use the default value of ACRO\_YAW\_P.

## 

## Test #1

**Date:** 21/10/20

**Map**: iris\_no\_corridor.world

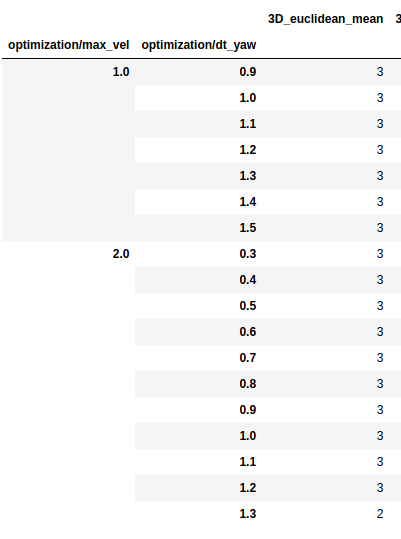
**Parameters to vary:**

* max\_vel
* max\_yaw

**Constants:**

* lambda1=10.0
* lambda2=0.8
* lambda3=10e-5
* lambda4=0.1
* lambda7=100
* PSC\_POSXY\_P=1

Parameters Table:



**This test crashed too soon. These parameters will be tested again.**

## Test #2

**Date:** 25/10/20

**Map**: iris\_no\_corridor.world

**Parameters to vary:**

* max\_vel
* max\_acc

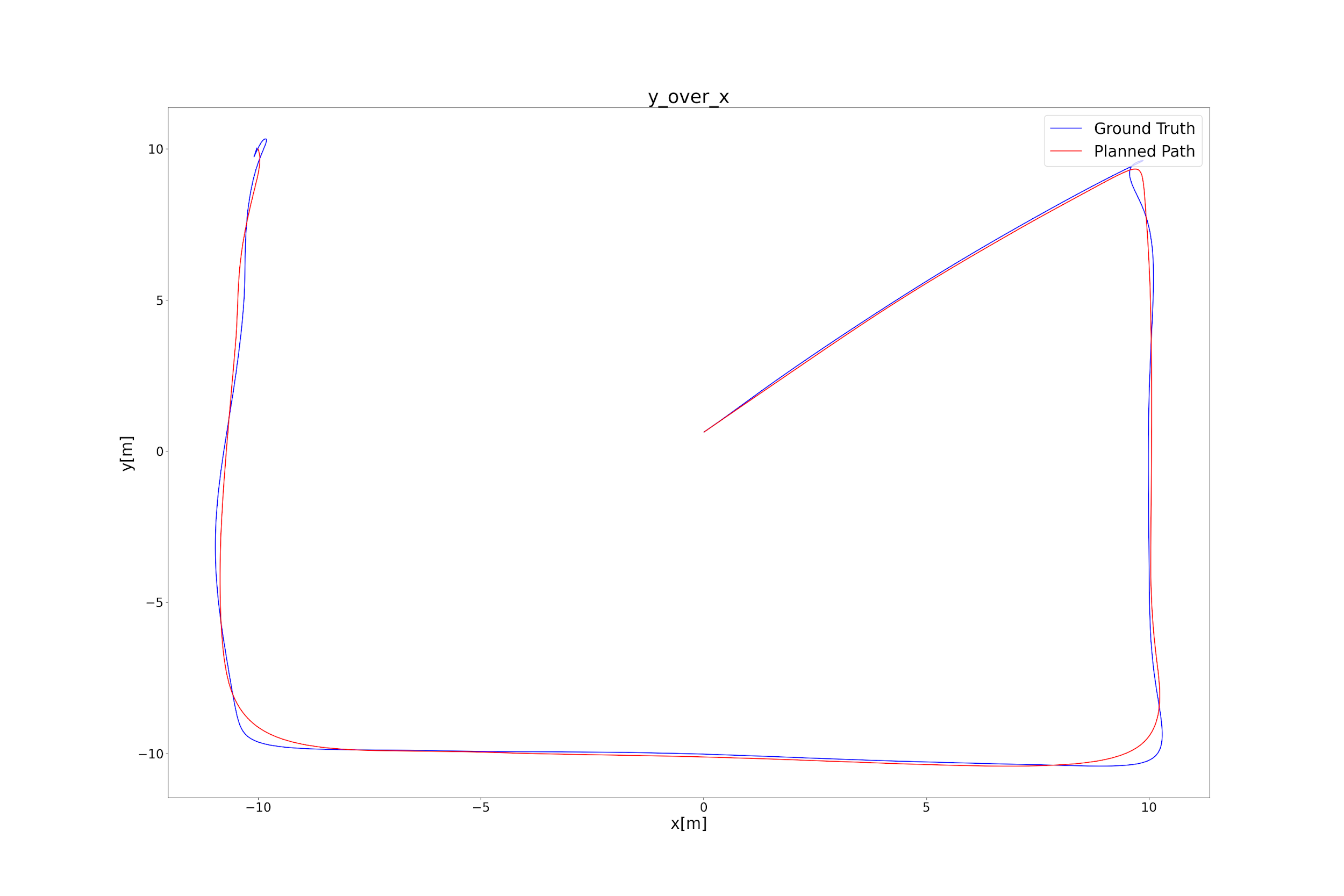
**Constants:**

* lambda1=10.0
* lambda2=0.8
* lambda3=10e-5
* lambda4=0.1
* lambda7=100
* PSC\_POSXY\_P=1
* dt\_yaw=0.9

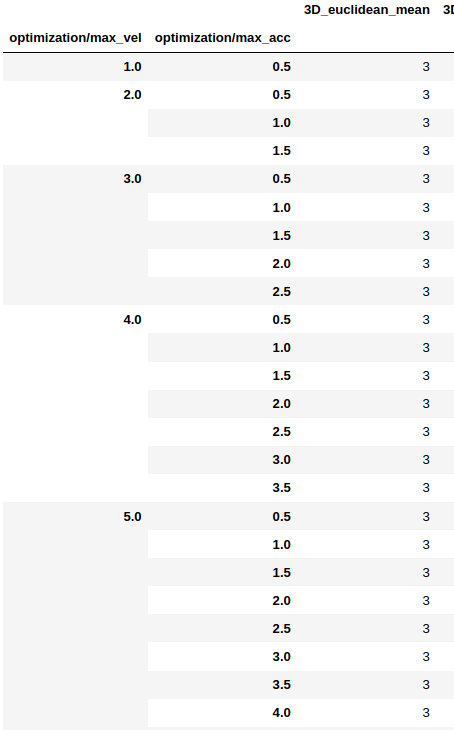
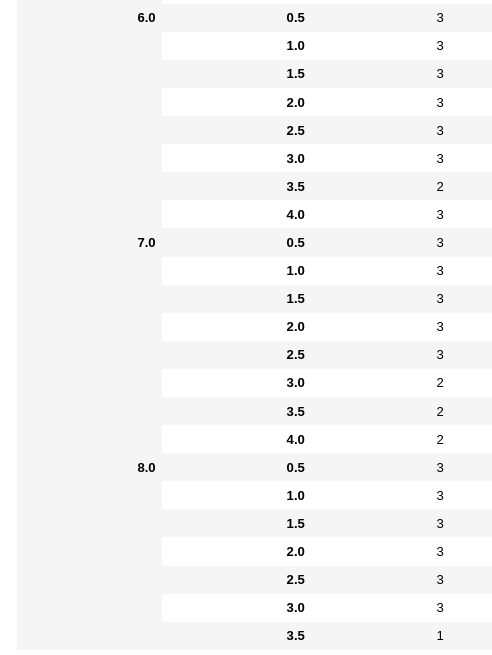
**Comments:**

* max\_acc values cannot be bigger then max\_vel values
* Simulation crashed at max\_vel=8, max\_acc=3.5

**Path Example**



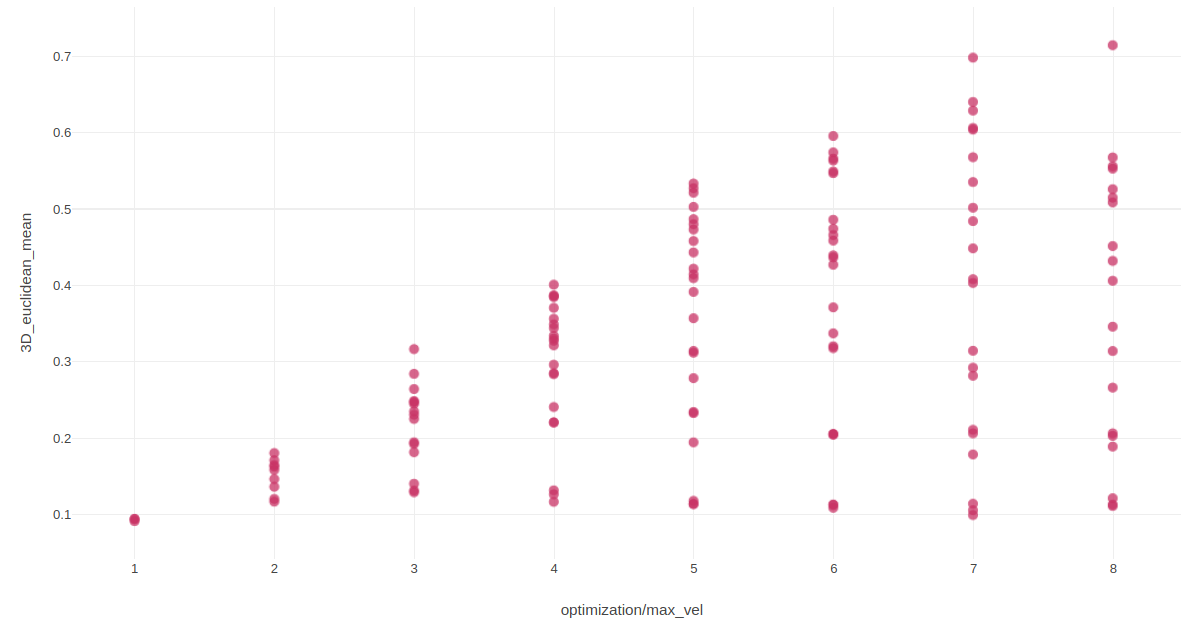
**Parameters Table:**

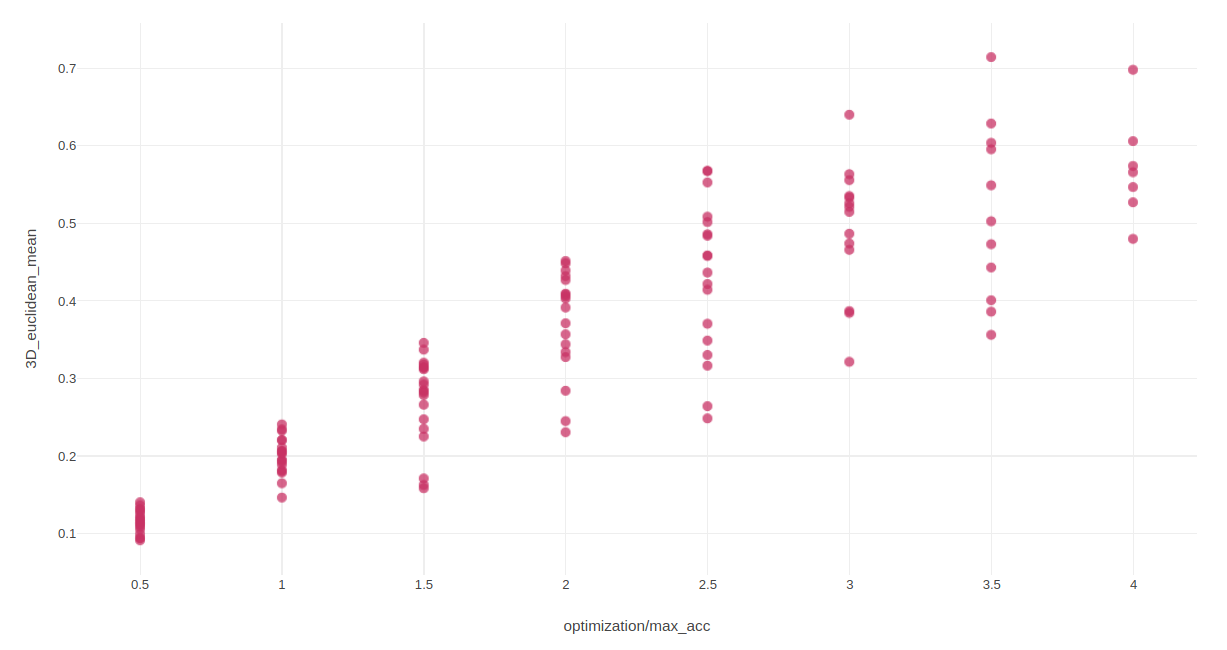


### General Results

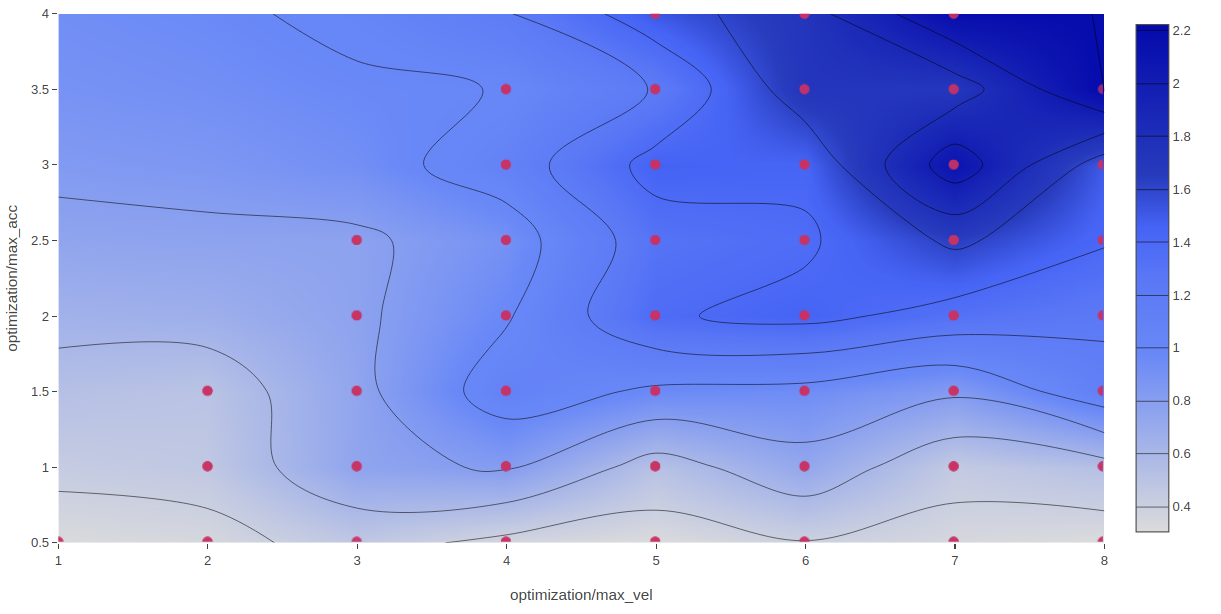
A major conclusion one can conclude from observing the parameters table is that for max\_vel 7 and 8 with max\_acc bigger than 3 we get less successful runs, this is caused because the drone crashed during the current simulation. According to that we neglect higher max\_vel values (bigger than 6 m/s) and max\_acc values (bigger than 3 m/s^2).

Let us draw the max and mean of the position euclidean error against max\_vel and max\_acc, first separately and then as contour plot:





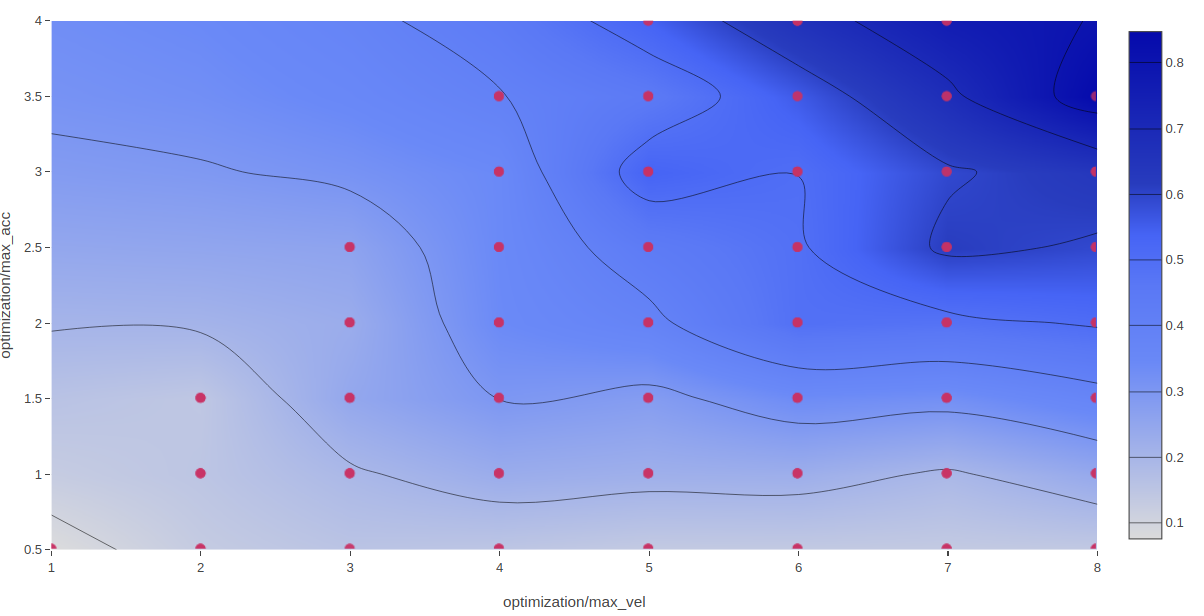




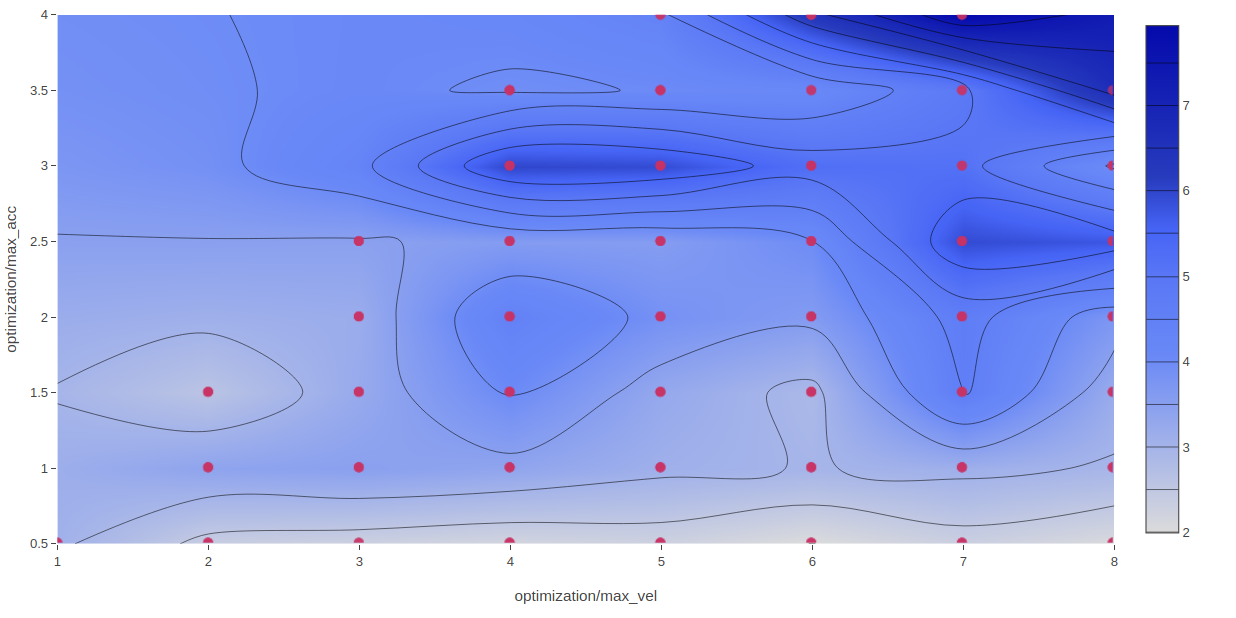
According to the graphs above, one can notice that the position euclidean error depends linearly on both max\_vel and max\_acc, more so its dependence is stronger to max\_acc value.

We observed similar results in other metrics such as euclidean max and std error, velocities errors and yaw errors:

Velocity mean error:



yaw mean error:



## Test #3

**Date:** 27/10/20

**Map**: iris\_no\_corridor.world

**Parameters to vary:**

* PSC\_POSXY\_P
* max\_vel

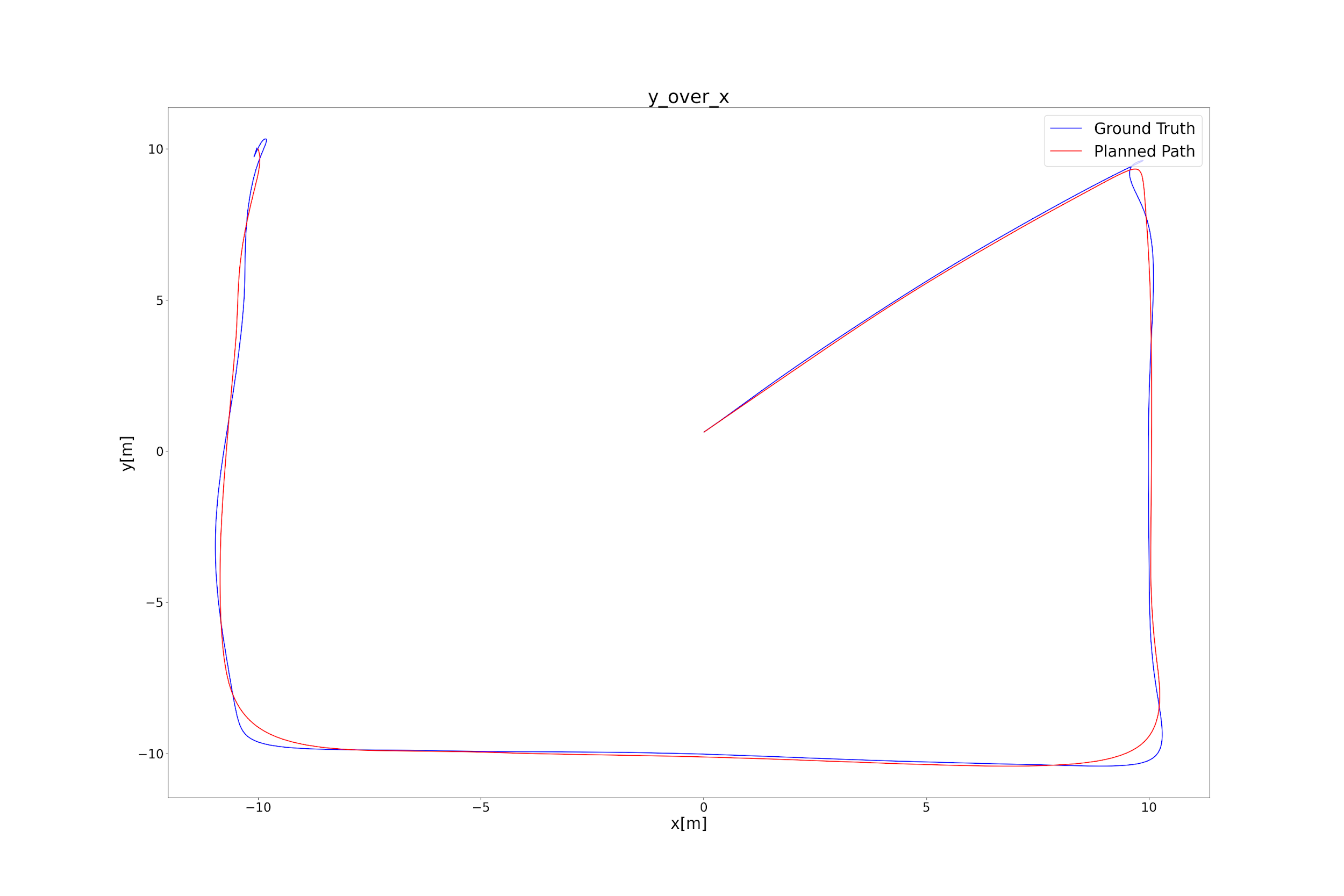
**Constants:**

* lambda1=10.0
* lambda2=0.8
* lambda3=10e-5
* lambda4=0.1
* lambda7=100
* dt\_yaw=0.9

**Comments:**

* max\_acc - max\_vel / 3

**Path Example**



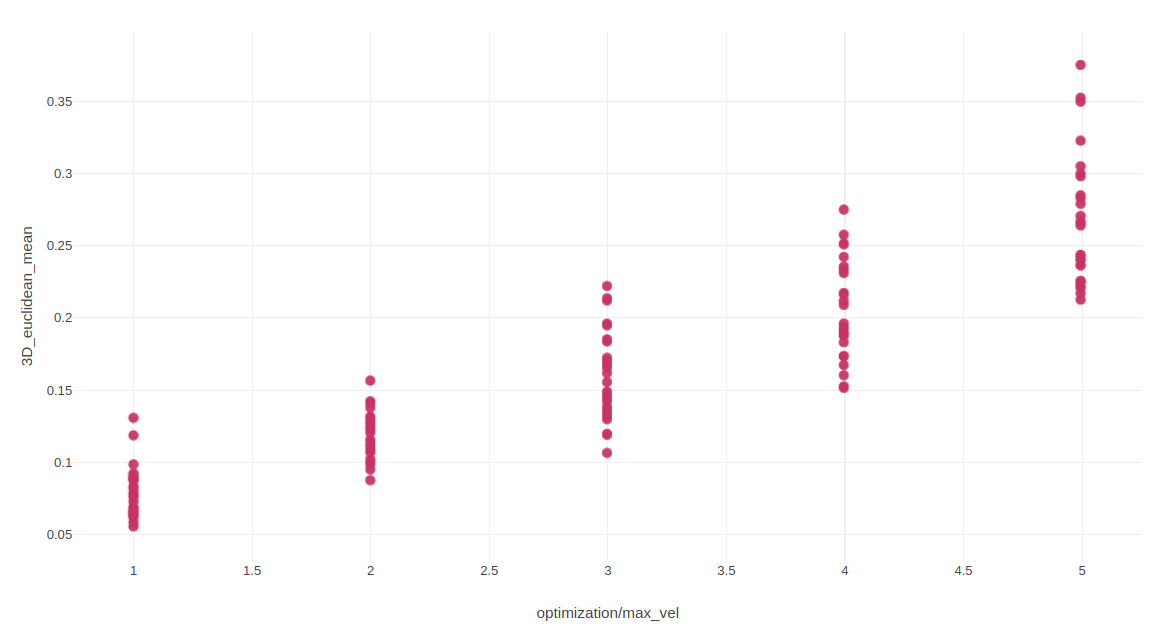
**Parameters Table**

|  |  |  |
| --- | --- | --- |
|  |  |  |

### 

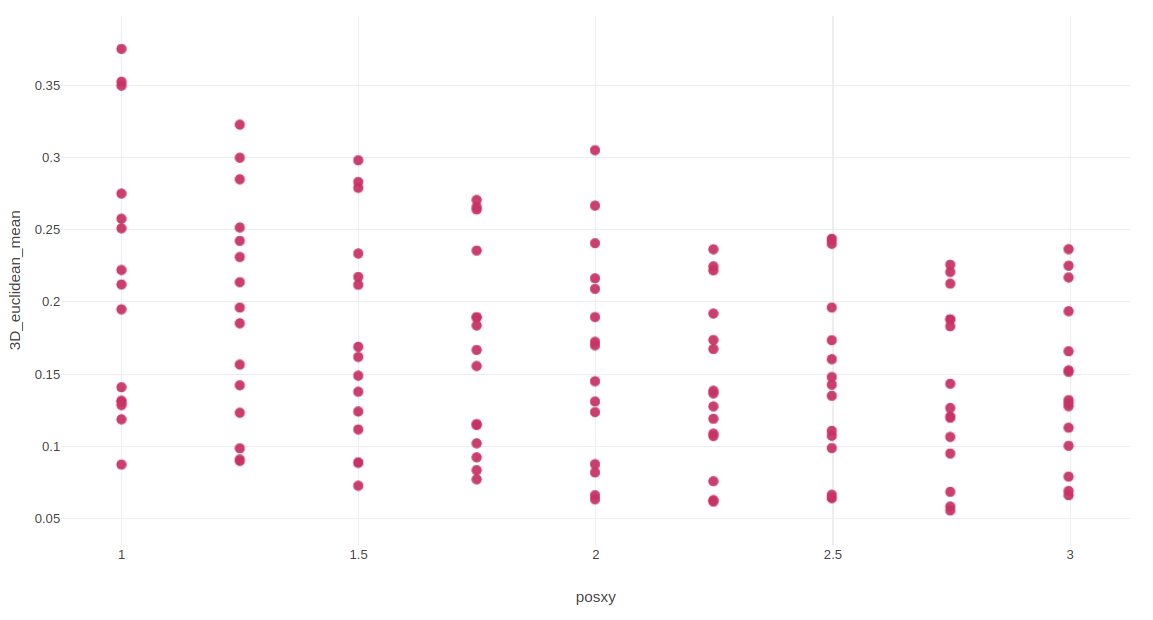
### General Results

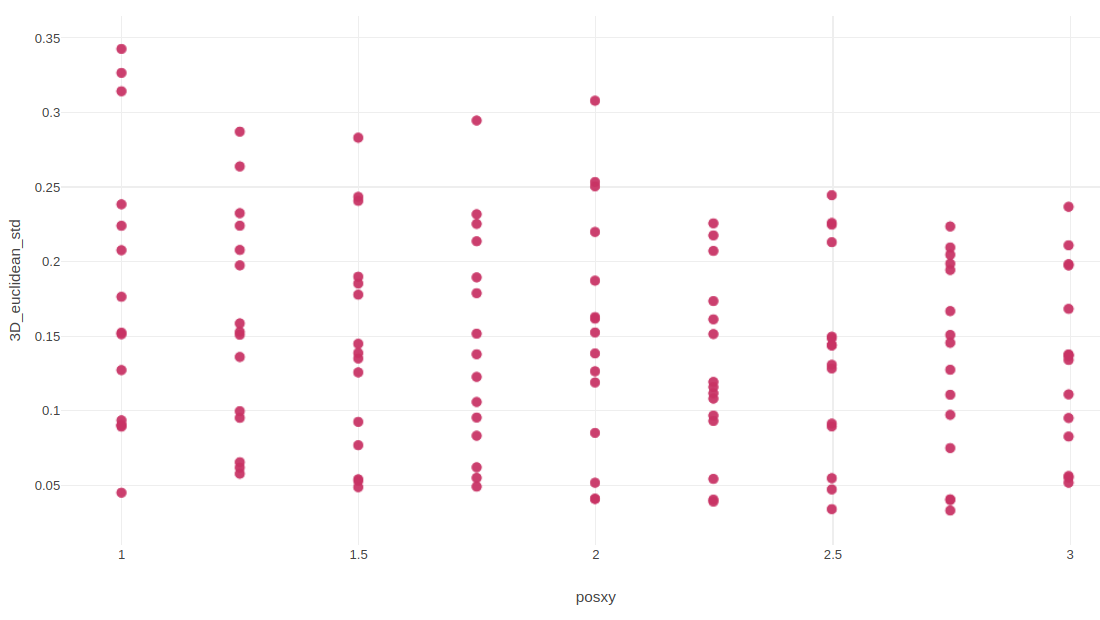
In this test we will mainly look at the position euclidean error. The PSC\_POSXY\_P need to be chosen so that the mean and max of the euclidean error will be minimum along with maintaining the standard deviation, namely the oscillations the drone is doing around the planned path, to be acceptable.

First, let us draw the mean of the position euclidean error against max\_vel, in order to compare to the previous tests. ****

One can notice that similar to test #2 the position euclidean error is increasing linearly with max\_vel parameter.

Now let us draw the mean and the standard deviation of the euclidean error against PSC\_POSXY\_P:



****

According to the graphs above, one can notice that there is a reversed ratio between PSC\_POSX\_P value and the euclidean error. Particularly it seem that the error is decreasing until PSC\_POSX\_P is reaching a value of 2.25 then it maintaining approximately the same values.

## Test #4

**Date:** 1/11/20

**Map**: obstacles.world

**Parameters to vary:**

* lambda1
* lambda2
* lambda3
* PSC\_POSXY\_P

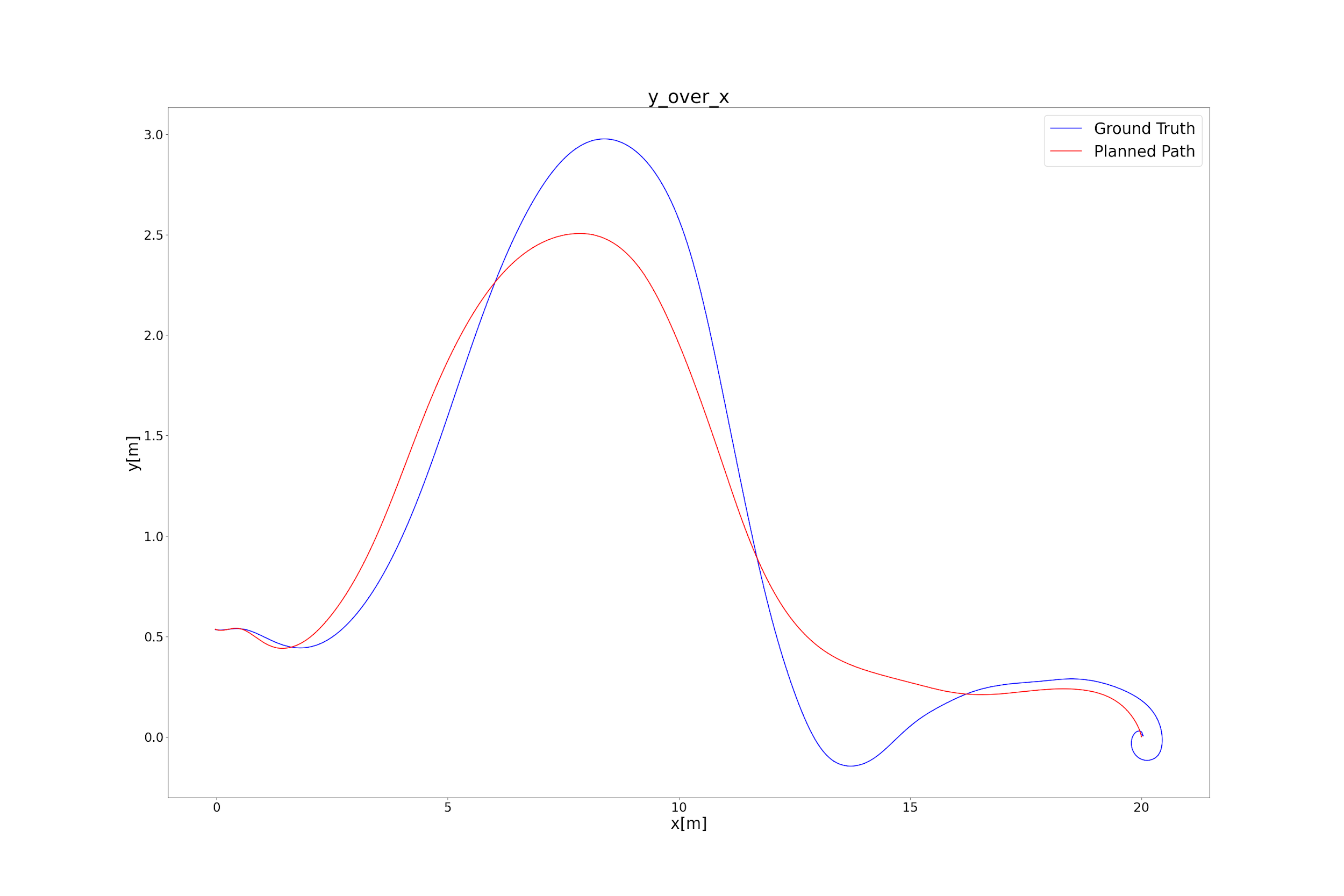
**Constants:**

* dt\_yaw = 1.0
* Max\_vel = 3.0
* Max\_acc = 1.0
* lambda4=0.1
* lambda7=0.001

**Comments:**

Lambdas values are in logarithmic scale

**Path Example**



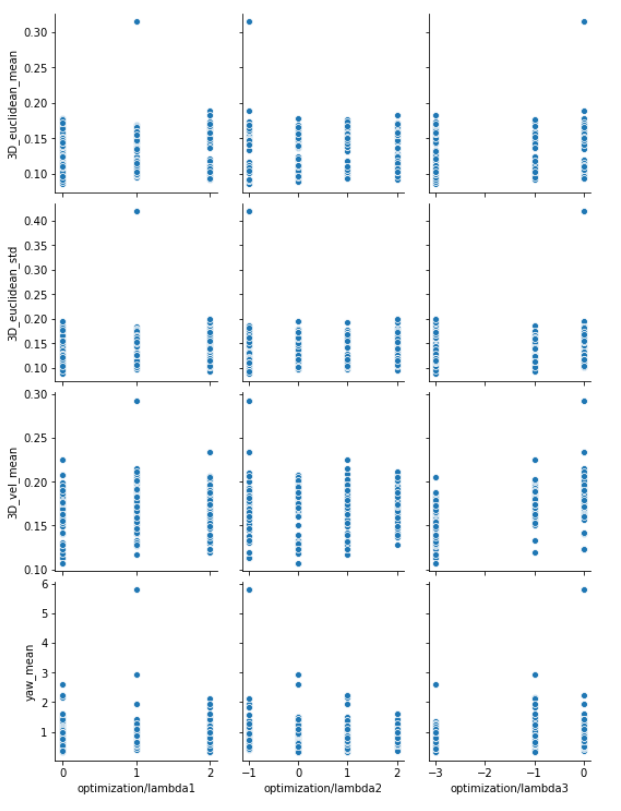
**Parameters Table**

|  |  |  |
| --- | --- | --- |
|  |  |  |

### 

### General Results

One main conclusion one may differ from the following graphs is that changing the lambdas values does not affect the errors between fast-planner planned path and drone's actual path (Drone control error).



You can notice that there is no correlation between the lambdas values (in logarithmic scale) to the drone control error.

## Test #5

**Date:** 2/11/20

**Map**: obstacles.world

**Parameters to vary:**

* dt\_yaw
* lambda7

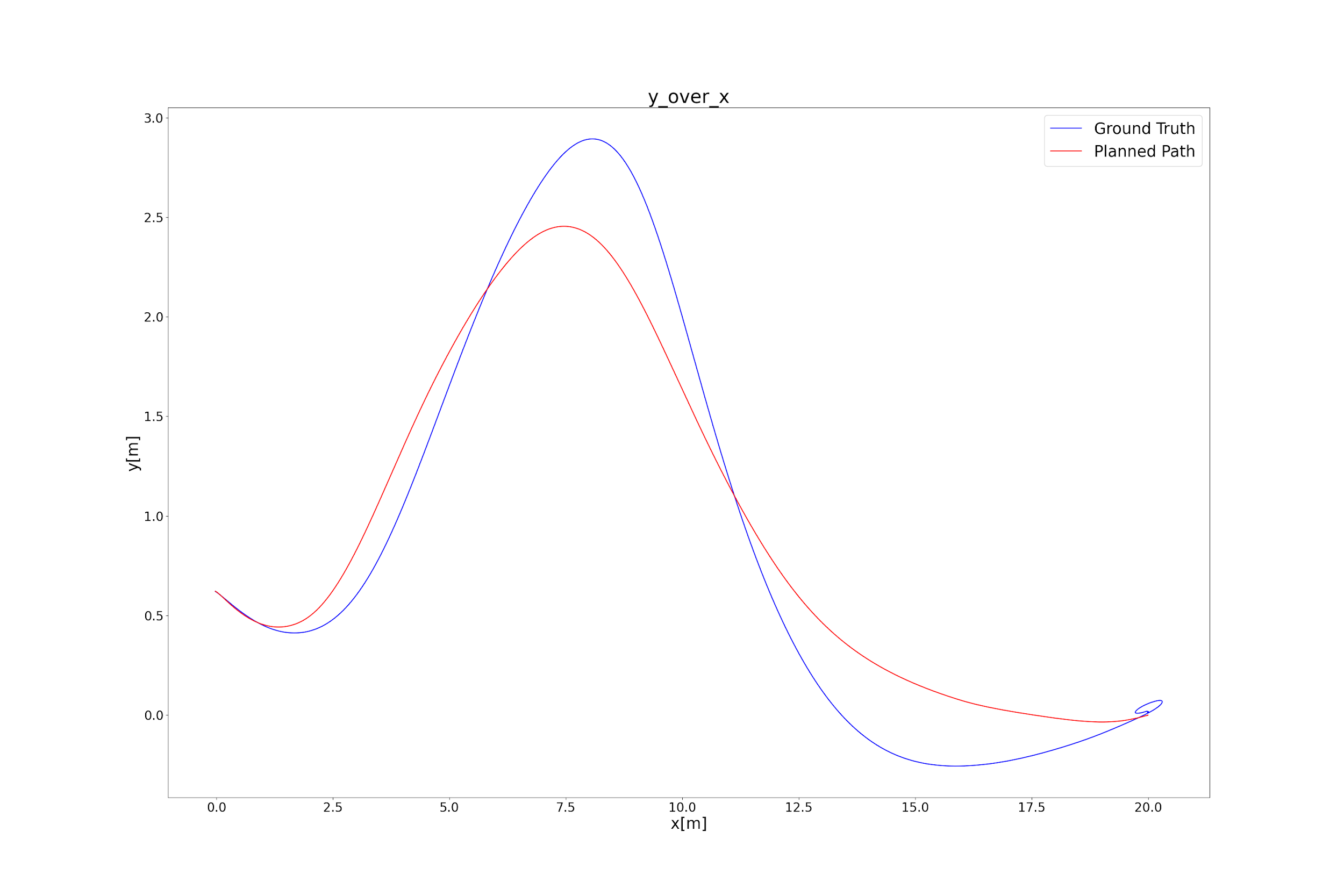
**Constants:**

* lambda1=10.0
* lambda2=0.8
* lambda3=10e-5
* lambda4=0.1
* Max\_vel = 3.0
* Max\_acc = 1.0
* PSC\_POSXY\_P = 1

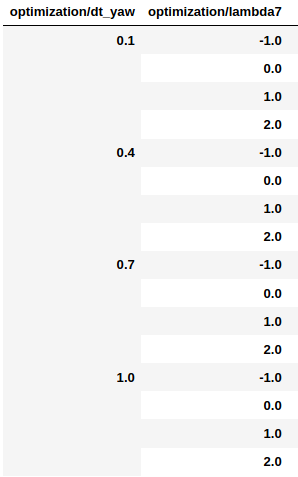
**Comments:**

Lambda7 values are in logarithmic scale

**Path Example**



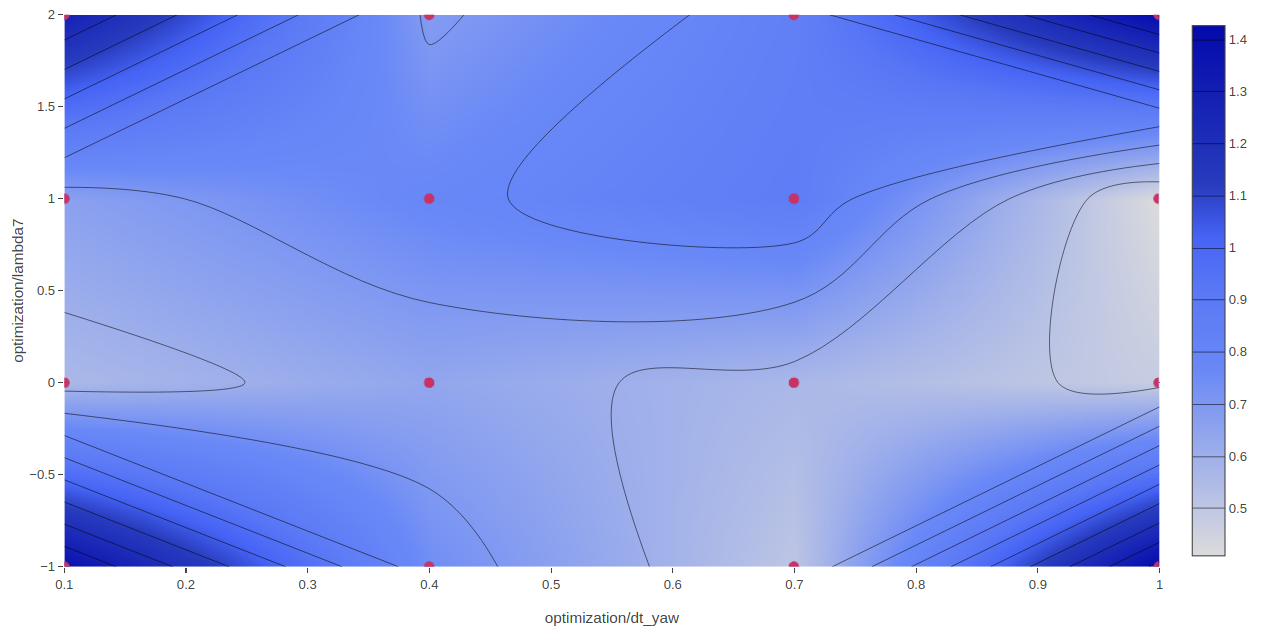
### Parameters Table

****

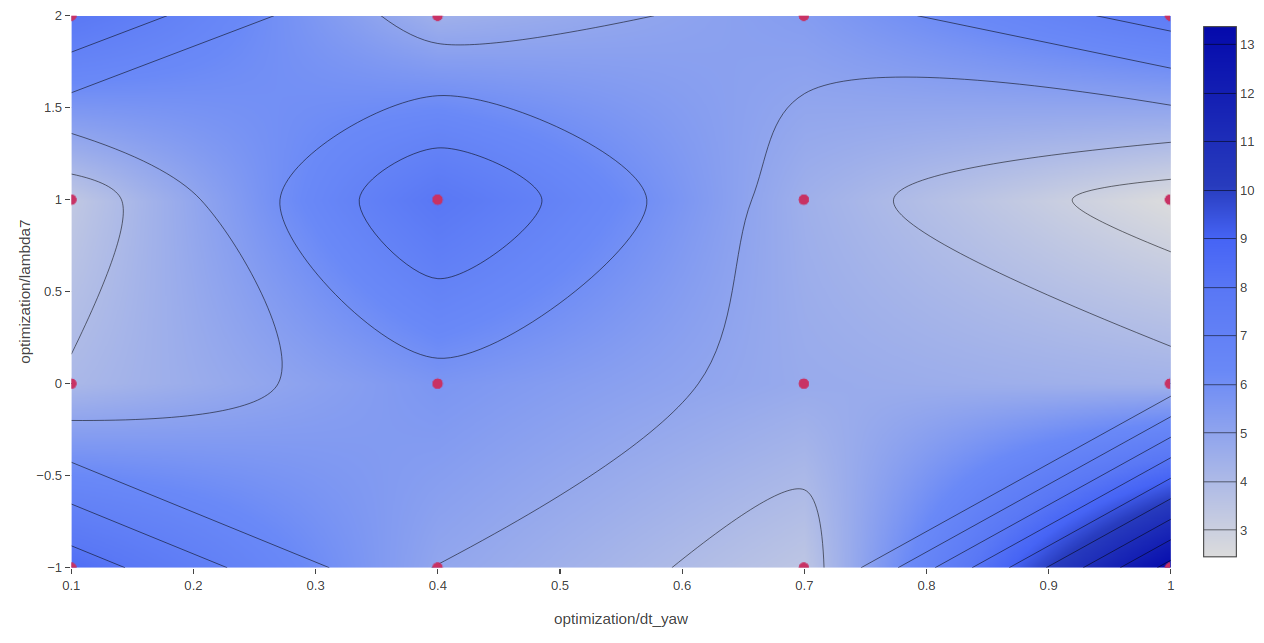
### General Results

Let us plot a contour plot of the yaw and yaw dot mean error, std and max error according to dt\_yaw and lambda7 values:

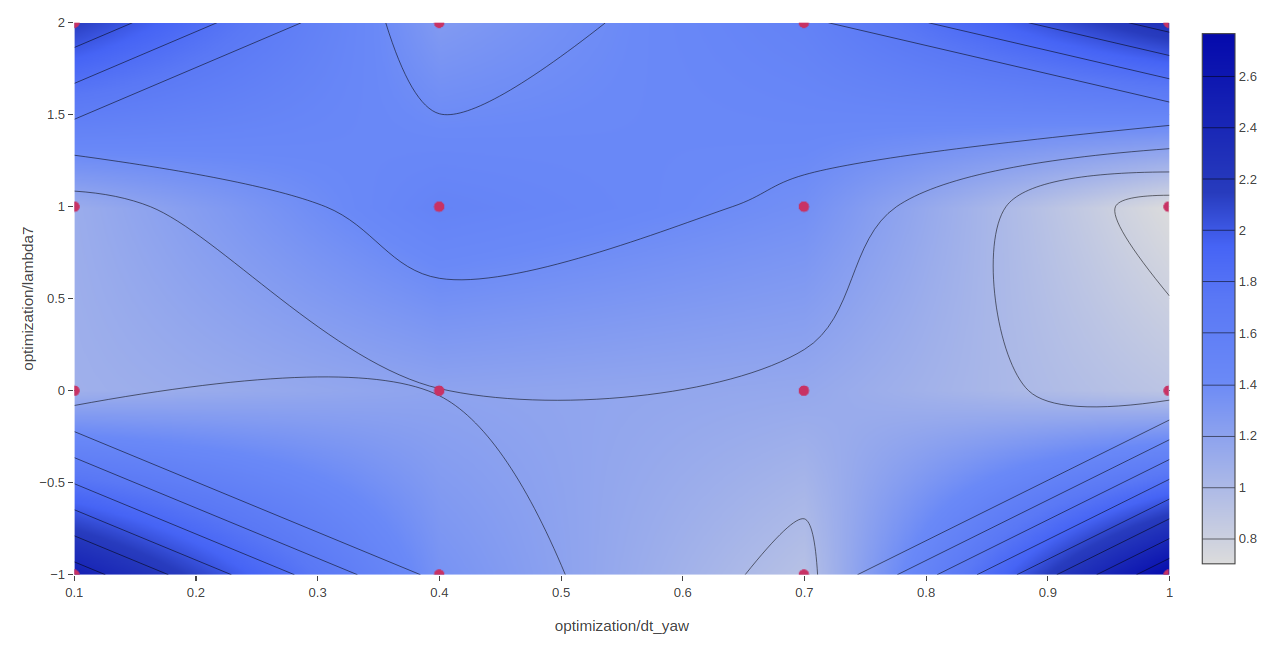
Yaw mean error:

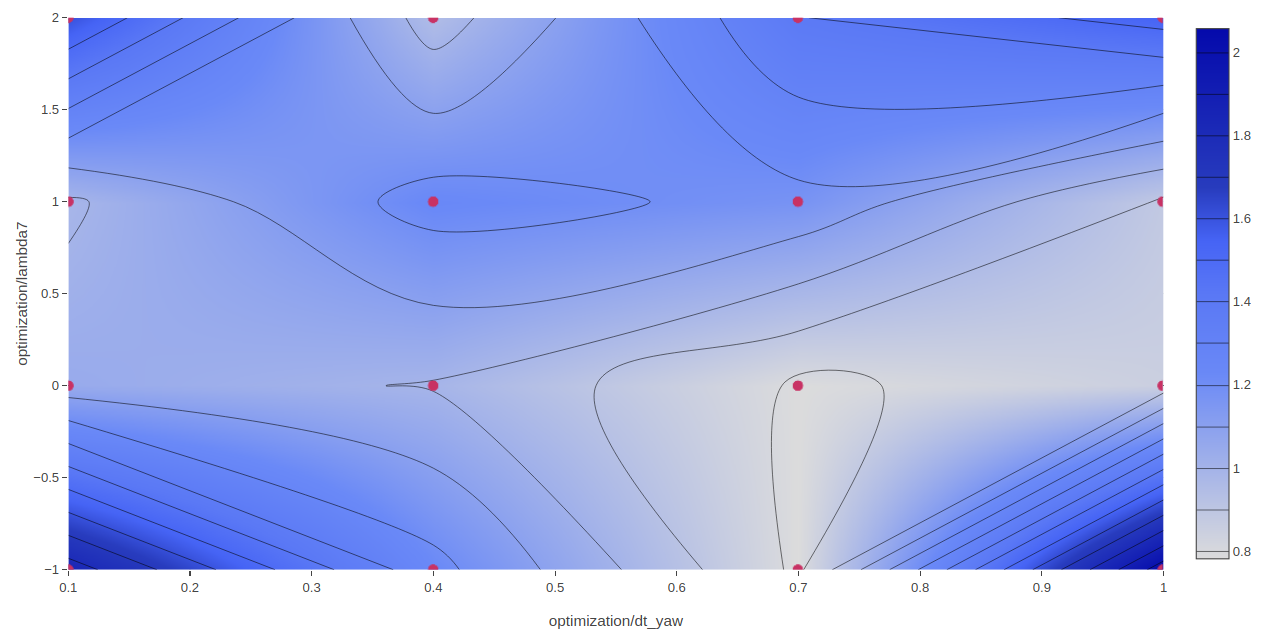


Yaw max error:



Yaw std:

yaw dot mean error:



In addition let us plot yaw and yaw dot mean errors for different lambda7 values:

According to these figures, one can notice that for lambda7 value of 0 (actual value is 1) a minimum error value is obtained.

Once we choose the value of lambda7, we will choose a value of 0.7 for dt\_yaw according to the figures above.

**After examining these parameters manually we obtained that value for dt\_yaw value of 0.5-0.6 the drone yaw rate is higher and the drone is more reactive, while drone control error is approximately the same.**

## 

## Test #6

**Date:** 2/11/20

**Map**: obstacles.world

**Parameters to vary:**

* PSC\_POSXY\_P

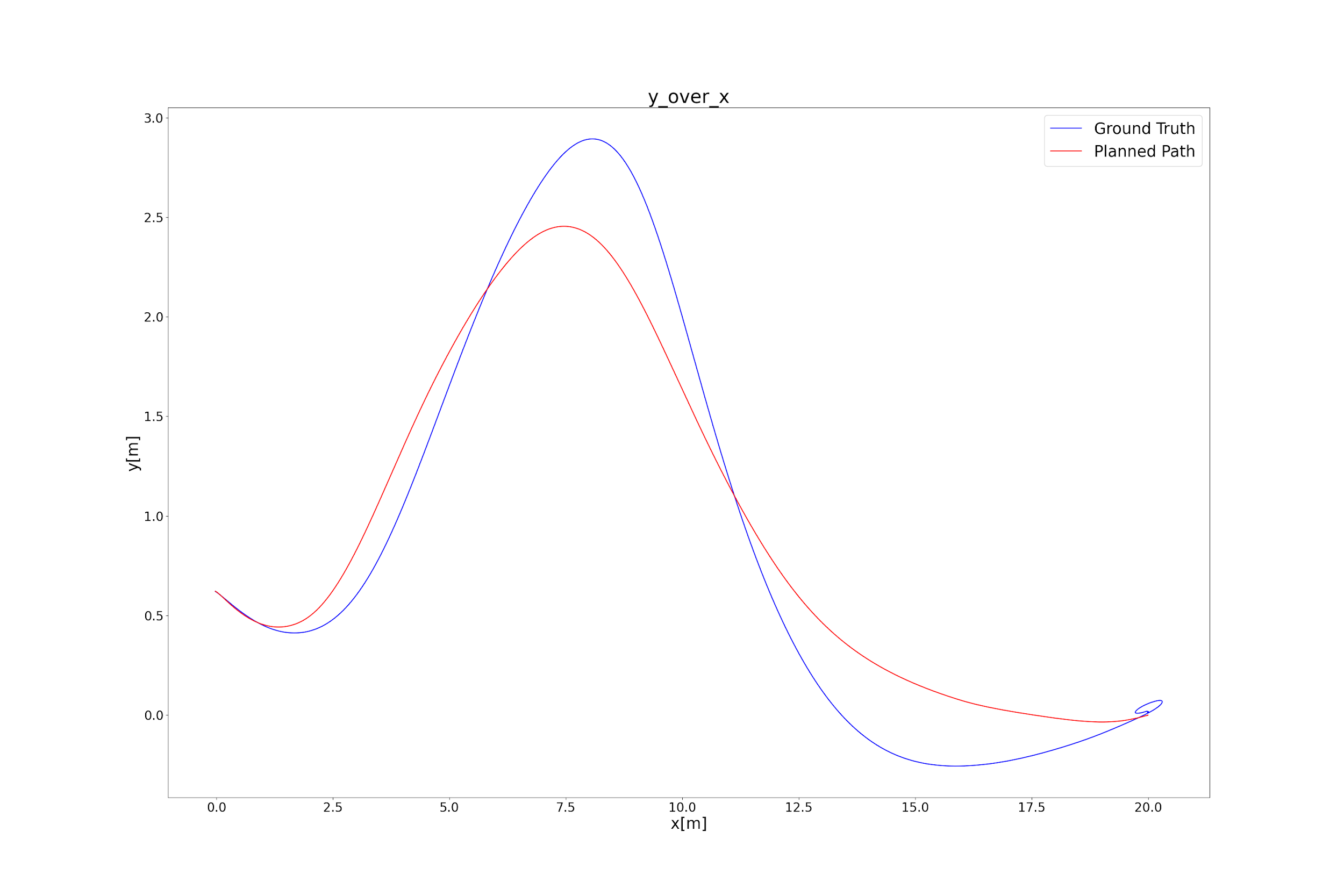
**Constants:**

* lambda1=10.0
* lambda2=0.8
* lambda3=10e-5
* lambda4=0.1
* lambda7=10
* dt\_yaw=0.5
* Max\_vel = 3.0
* Max\_acc = 1.0

**Comments:**

Crashed at PSC\_POSXY\_P=2.75 in the 4th iteration out of 5

**Path Example:**

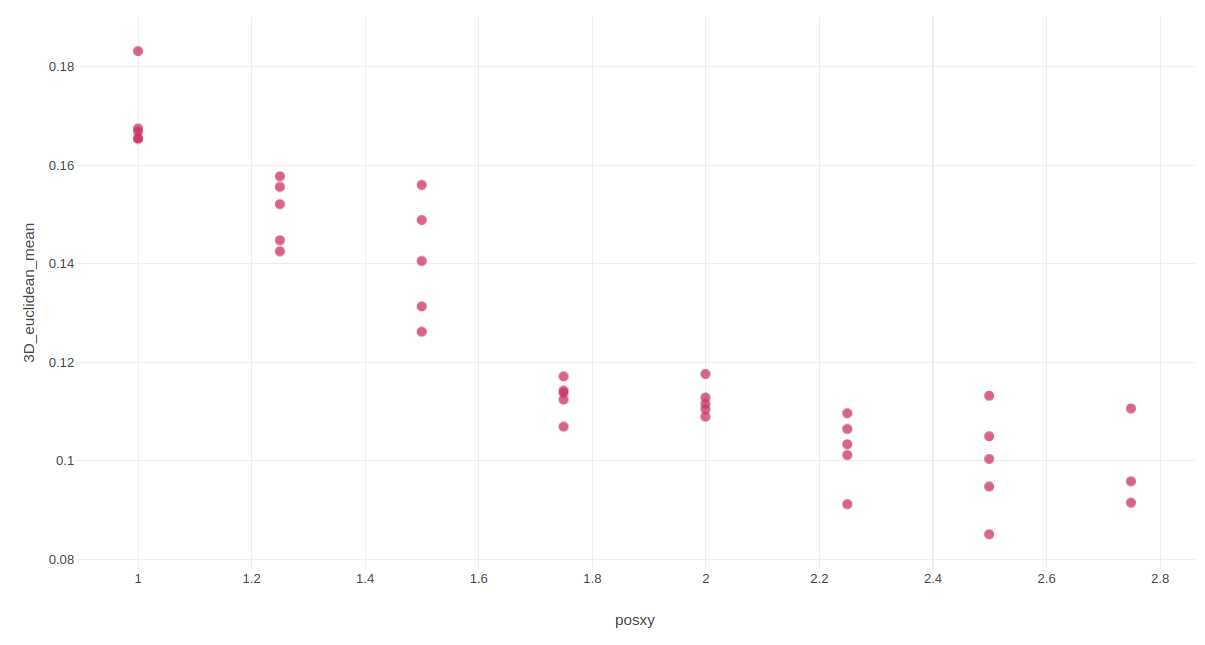


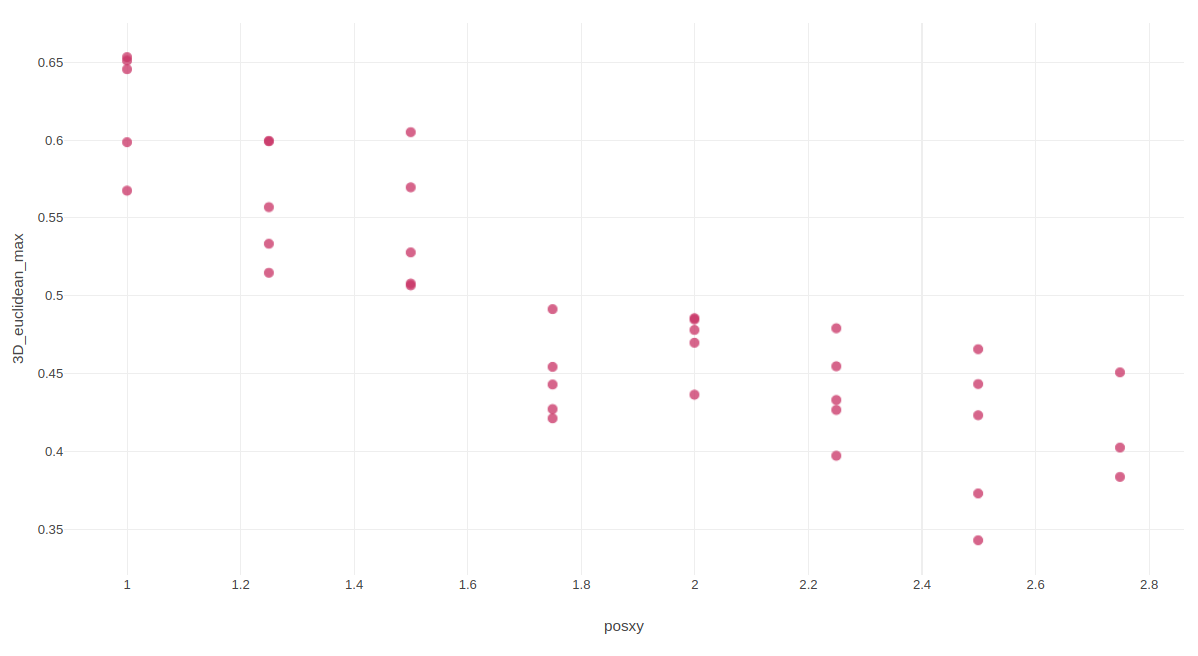
### Parameters Table

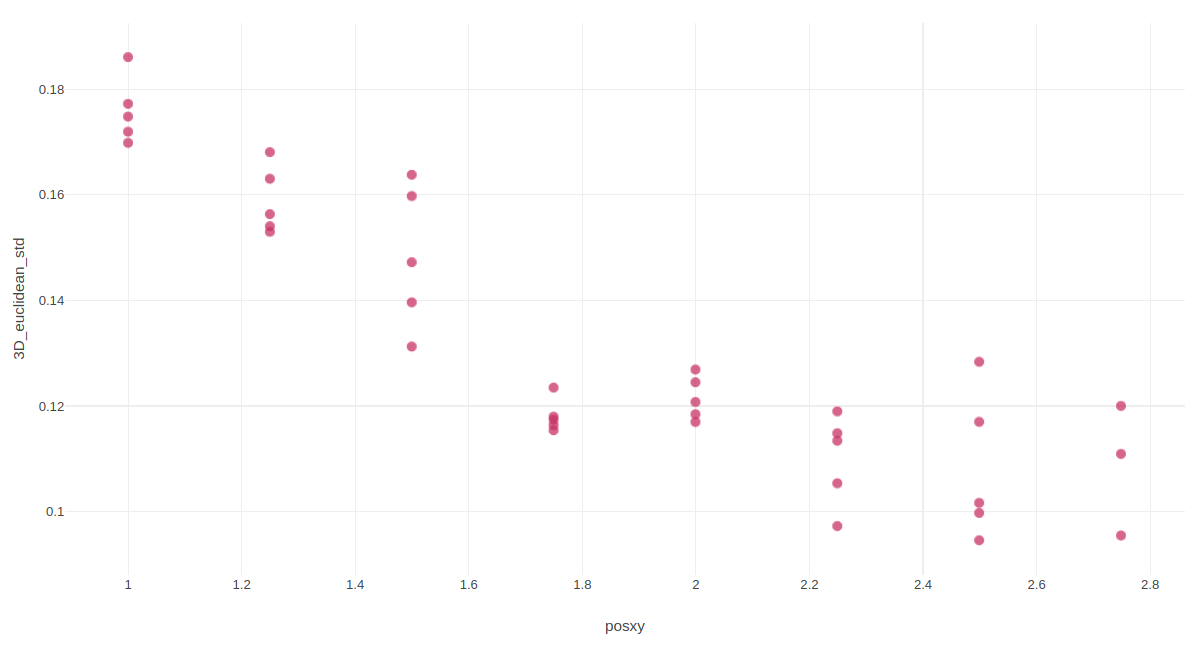
****

### General Results

Let us plot a contour plot of the position euclidean mean error, std and max error according to PSC\_POSXY\_P values:

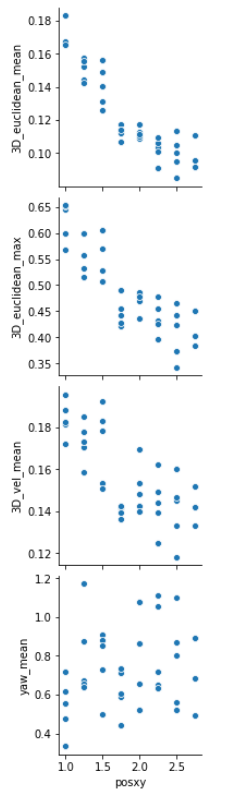






It is easy to notice that there is a strong opposite correlation between PSC\_POSXY\_P values and the position euclidean errors.

For summary, lets plot position euclidean, velocity and yaw errors against PSC\_POXY\_P values:



In addition to the euclidean errors, there is also an opposite correlation between velocity errors values and PSC\_POSXY\_P values, but there is no correlation to yaw errors values.

**Another test with higher values of PSC\_POSXY\_P is required.**

## Test #7

**Date:** 3/11/20

**Map**: obstacles.world

**Parameters to vary:**

* PSC\_POSXY\_P

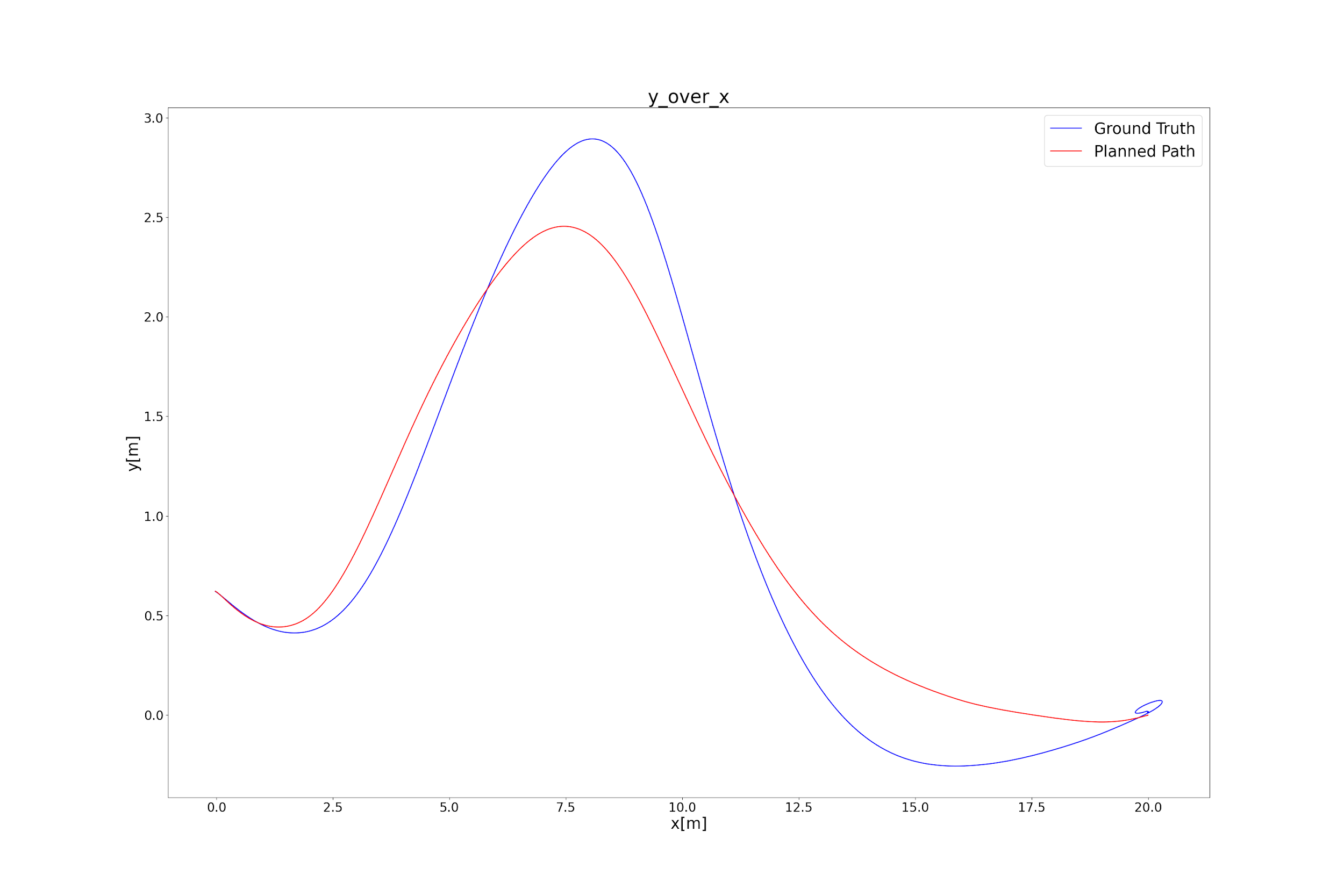
**Constants:**

* lambda1=10.0
* lambda2=0.8
* lambda3=10e-5
* lambda4=0.1
* lambda7=10
* dt\_yaw=0.5
* Max\_vel = 3.0
* Max\_acc = 1.0

**Comments:**

**These test results were united with the previous test results.**

**Path Example:**

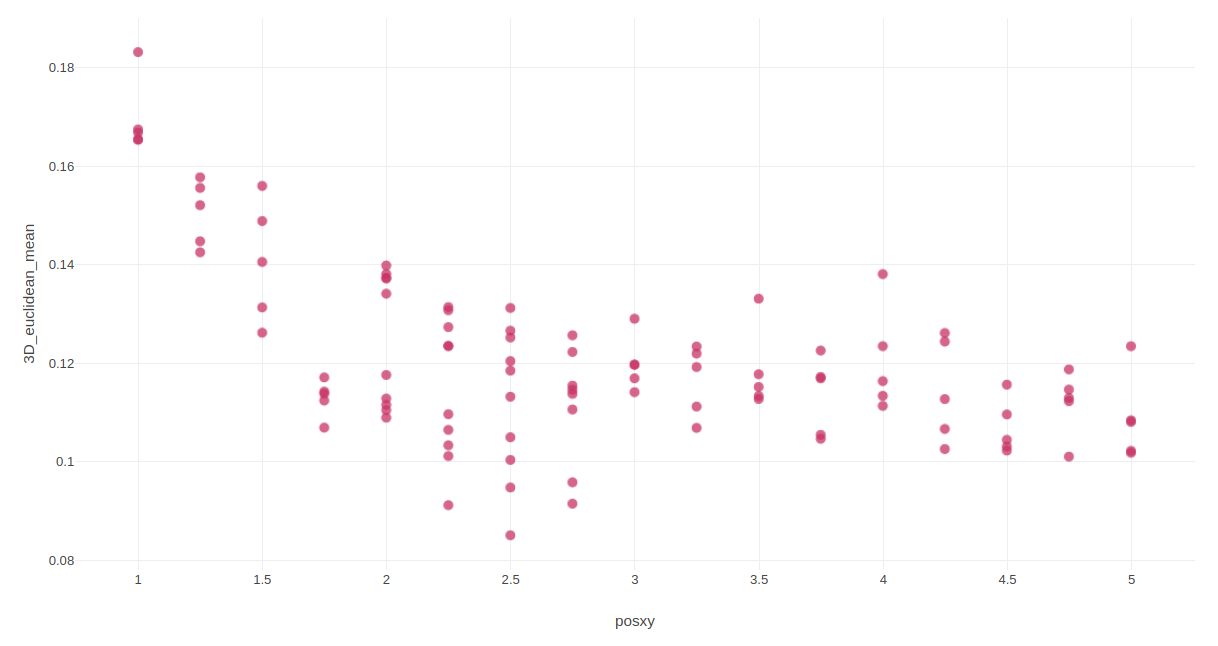


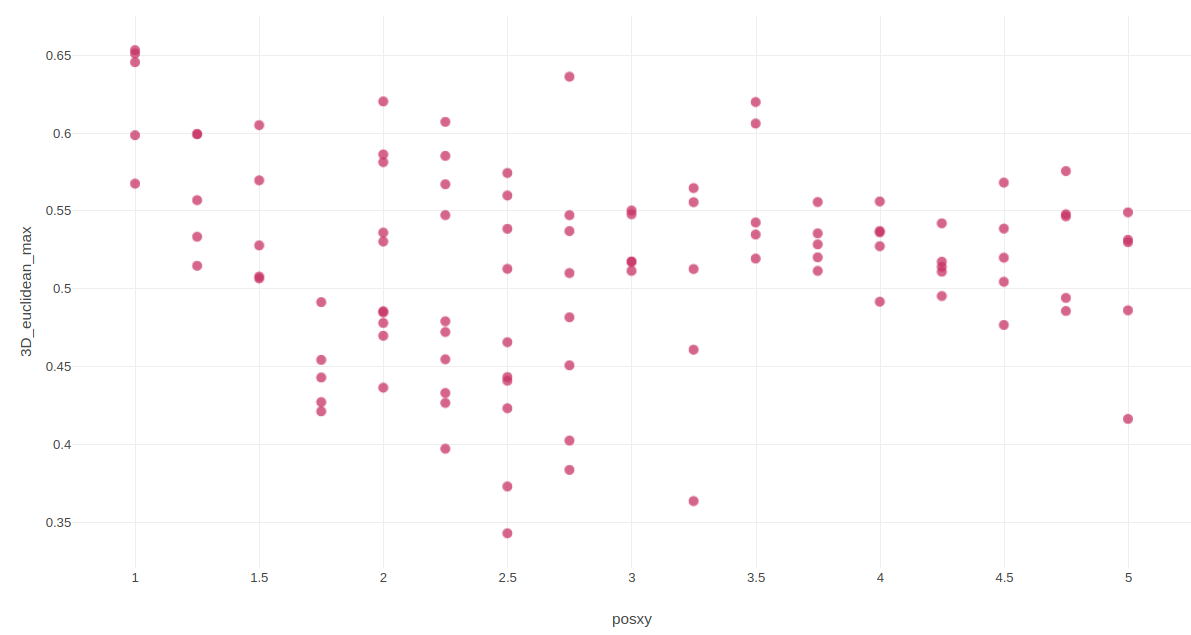
### Parameters Table

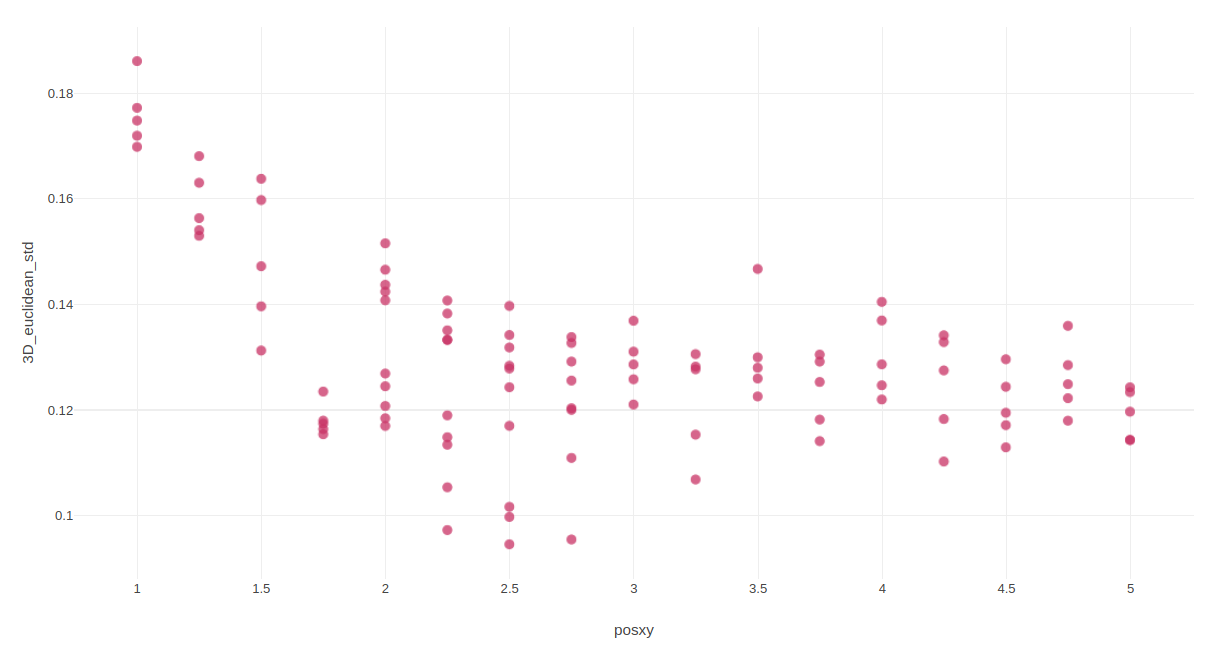


### General Results

Let us plot a contour plot of the position euclidean mean error, std and max error according to PSC\_POSXY\_P values:





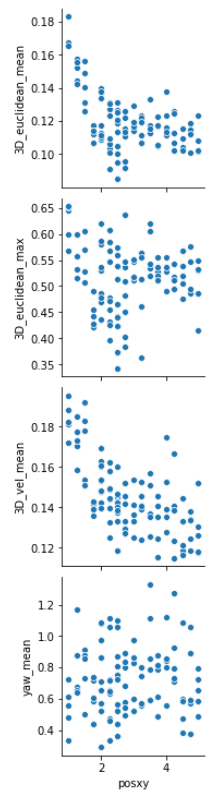


Similar to the previous test, it is easy to notice that there is a strong opposite correlation between PSC\_POSXY\_P values and the euclidean errors.

Furthermore one can notice that for PSC\_POSXY\_P values higher than 2.5 the errors value no more decreasing and additionally the euclidean max error is reaching a minimum value at PSC\_POSXY\_P=2.5.

**Hence the recommended value of PSC\_POSXY\_P is 3.**

For summary, lets plot euclidean, velocity and yaw errors against PSC\_POXY\_P values:



Similar to the previous test, in addition to the euclidean errors, there is also an opposite correlation between velocity errors values and PSC\_POSXY\_P values, but there is no correlation to yaw errors values.

**PSC\_POSXY\_P chosen value is 2.5.**

**In order to deal with the oscillation at the end of the path, one can change PSC\_POSXY\_P value to 1 after reaching the target.**

## Test #8

**Date:** 3/11/20

**Map**: obstacles.world

**Parameters to vary:**

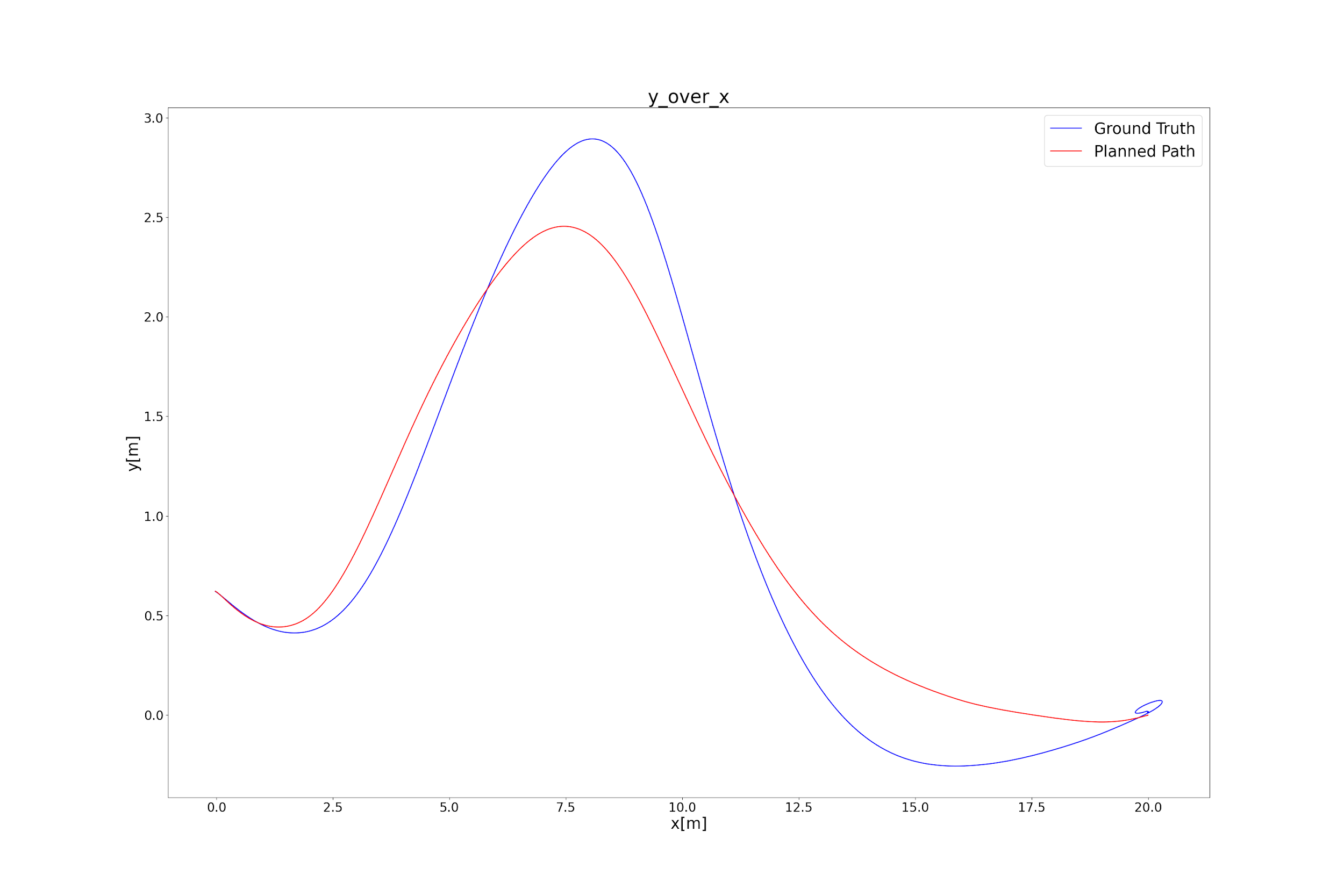
* ACRO\_YAW\_P

**Constants:**

* lambda1=10.0
* lambda2=0.8
* lambda3=10e-5
* lambda4=0.1
* lambda7=10
* dt\_yaw=0.5
* Max\_vel = 3.0
* Max\_acc = 1.0
* PSC\_POSXY\_P=1

**Comments:**

**Path Example:**

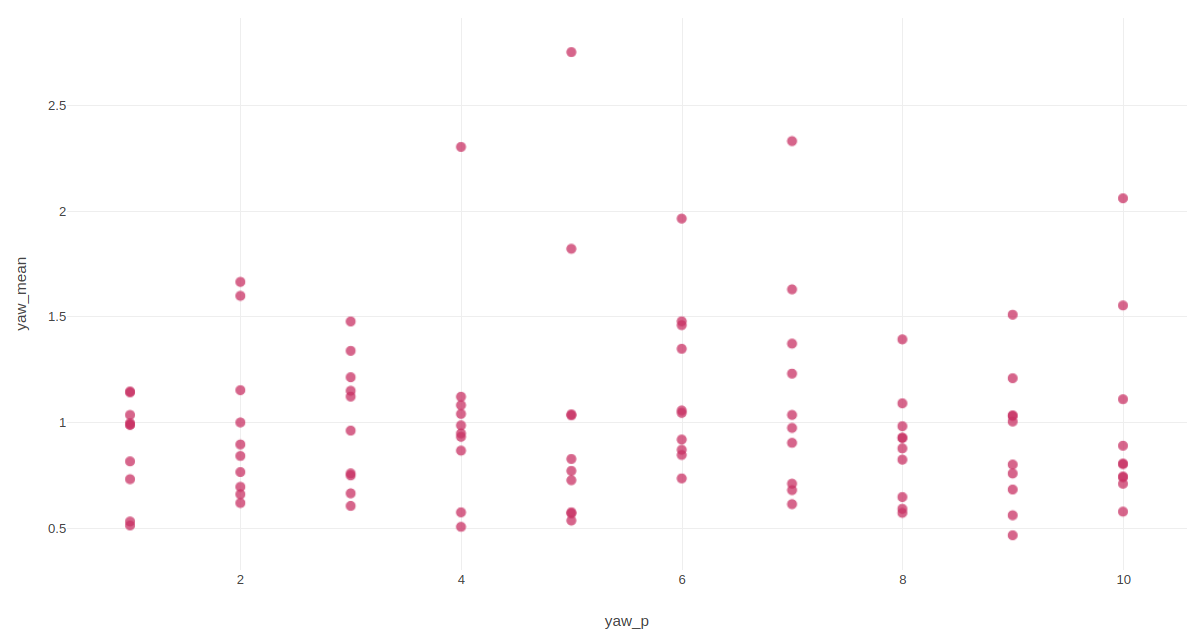


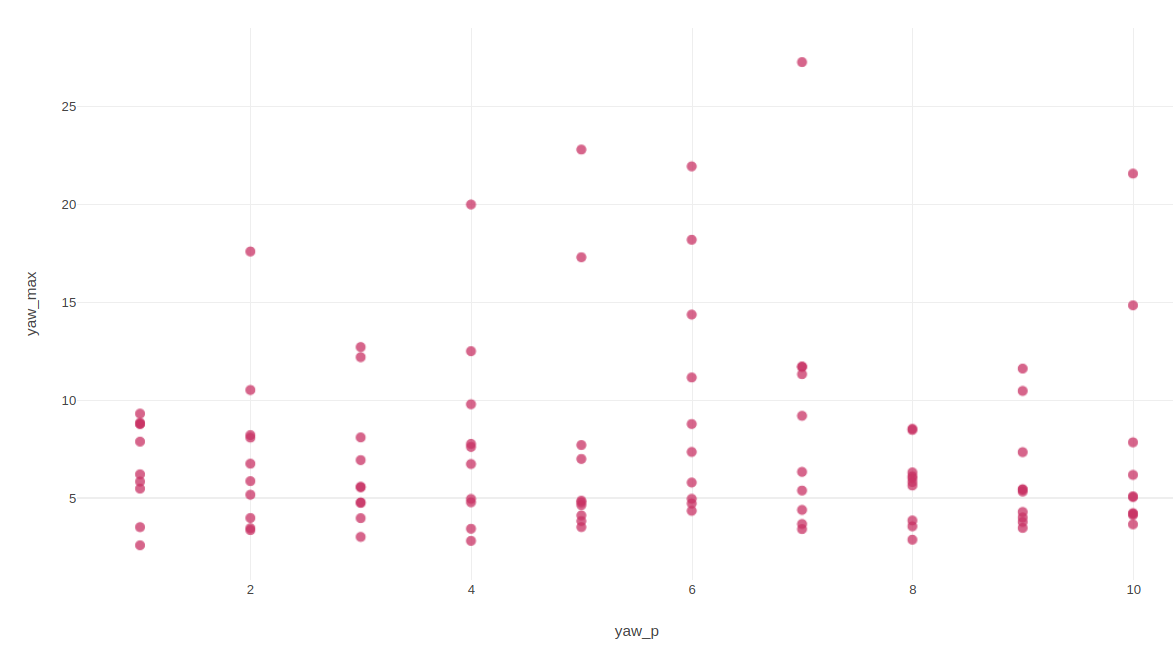
### Parameters Table

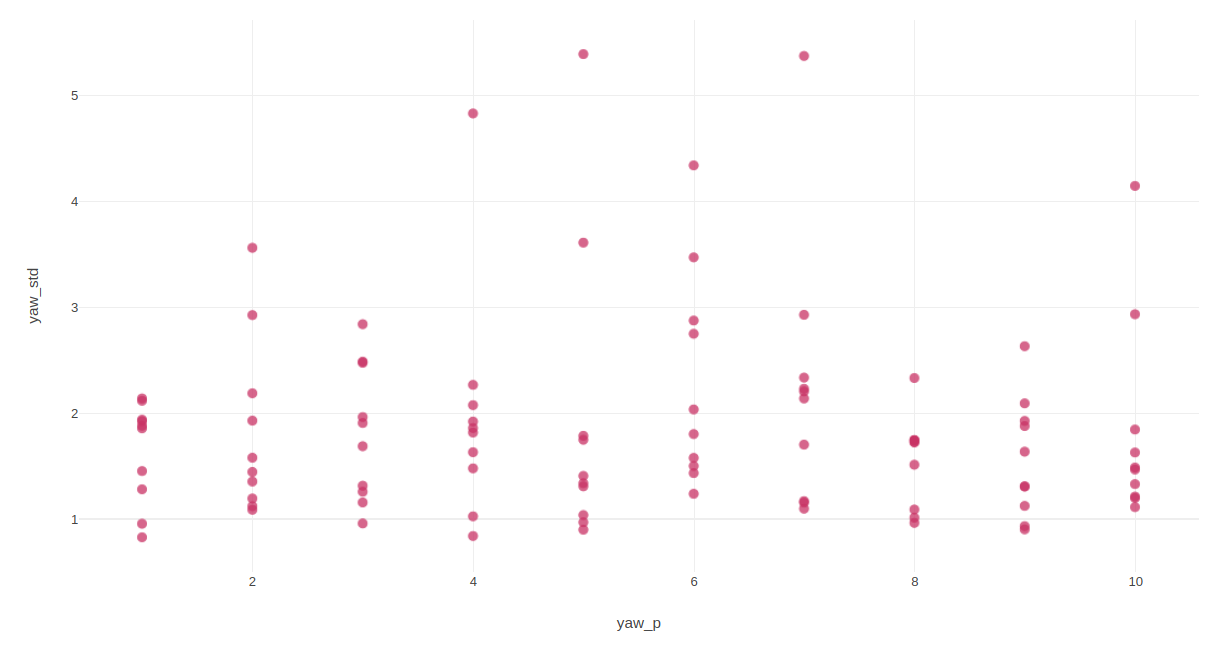


### General Results

Let us plot a contour plot of the yaw mean error, std and max error according to ACRO\_YAW\_P values:







One can notice that there is no correlation between ACRO\_YAW\_P values and yaw error values. That is also true for other path errors (manually examined).

## Test #9

**Date:** 5/11/20

**Map**: iris\_included.world

**Constants**

* lambda1=10.0
* lambda2=0.8
* lambda3=10e-5
* lambda4=0.1
* lambda7=10
* dt\_yaw=0.5
* Max\_vel = 3.0
* Max\_acc = 1.0
* PSC\_POSXY\_P=1

**Comments:**

**Path Example:**

