PERCEPTION, LOCALIZATION AND MAPPING FOR MOBILE ROBOTS

FIRST PROJECT

- Marco Cella 10578855
- Giacomo Delcaro 10560602
- Alessandro Colombo 10573335

POLITECNICO MILANO 1863

FILES 1/3

- **cfg**
  - parameters.cfg: file for the dynamic reconfigure
- **launch**
  - scout_launch.launch: launches the velpubsub, odompubsub and residual nodes. It is also possible to change the inital position of the robot
  - scout_launch_rviz.launch: just like the other one but it also runs rviz with the right configuration
- **msg**
  - OdomInt.msg: contains the custom message to publish odometry and integration method
  - er_array.msg: custom message to publish the residuals between the ground truth and the estimated odometry

FILES 2/3

- **srv**
  - **resetOdom**.srv: empty, it does not need any input from the user since it simply resets the pose
  - **setOdom**.srv: defines the types of the requested input from the user to set the pose to x, y, theta

- **src**
  - **velpubsub**.cpp: the node that subscribes to the rpm topics and publishes a twist message containing all of the robot's seeds
  - **odompubsub**.cpp: the node that subscribes to the topic advertised by velpubsub and publishes the complete odometry and the custom message
  - **residuals**.cpp: the node publishes the pose errors between /scout_odom and our computed odometry /our_odom and also the pose errors between /gt_pose (in odom reference frame) and /our_odom

FILES 3/3

- **config**
  - plot_layout.xml: layout for plotjuggler
  - rototranslation.m: MATLAB file to compute the “odom” to “world” rototranslation parameters
  - scout_rviz.rviz: rviz configuration to visualize the computed odometry vs the ground truth

PARAMETERS & TF TREE

PARAMETERS

- **x_init**: initial x position in the odom frame
- **y_init**: initial y position in the odom frame
- **theta_init**: initial yaw angle wrt the odom frame
- **CHI**: ratio between apparent and real base_line
- **Inv_RATIO**: 1/ratio of the gearbox

TF TREE

- **odom**
  - base_link
  - world (static_transform)

CUSTOM MESSAGES

1) Odometry custom message

OdomInt.msg:
nav_msgs/Odometry odo
std_msgs/String int_method

2) Message used to compute the position errors and the error cost function

er_array.msg:
std_msgs/Float64 dx
std_msgs/Float64 dy
std_msgs/Float64 dtheta
std_msgs/Float64 cumulateError

HOW TO RUN EVERYTHING 1/2

1) `catkin_make` in the catkin workspace root
2) source catkin_ws/devel/setup.bash
3) `roslaunch project1` **`scout_launch`**`.launch`
4) `rosbag play -l bag1.bag`

NB: for the step 4 you must cd to the folder where your bag files are.

It is now possible to echo the data from the different topics.

If you want to visualize the movement on rviz:

1) Launch `roslaunch project1` **`scout_launch_rviz`**`.launch`
2) Play the bag

HOW TO RUN EVERYTHING 2/2

DYNAMIC RECONFIGURE

While the nodes are running, type in the bash:

```
rosrun dynamic_reconfigure dynparam set /odompubsub intmethod value
```

"Value" can be either 0 for euler integration or 1 for runge – kutta

SERVICES

Reset service: `rosservice call /reset_odom`
Set service: `rosservice call /set_odom x y theta`

NB: theta is in [deg] !

APPARENT BASELINE AND TRANSMISSION RATIO

In order to find the parameters to match /scout_odom, we compared the estimated velocities values with the ground truth ones. For the transmission ratio we've used the values of the longitudinal velocity vx, whereas for the apparent baseline we've used the angular velocity wz.

The best values we've found are the following:

- Transmission ratio: 1:38.7
- Apparent baseline: 1.75 * B

To have a quantitative assessment on how we were improving when changing the parameters, we used PlotJuggler and the Residuals node mentioned above.
In particular we used 4 different types of errors:

- dx, dy, dtheta: longitudinal, lateral and angular difference between our computed pose and /scout_odom or /gt_pose
- cumulateError: this is a filtered sum of the errors, which means that it is the average of the last 100 quadratic errors. Each quadratic error is the square root of the sum of the square of dx, dy, dtheta (each multiplied by a particular quadratic weight).

APPARENT BASELINE AND TRANSMISSION RATIO

For the errors with respect to /scout_odom we chose the weights so as to give the same importance to a longitudinal or lateral error of 1 cm and an angular error of 60 degrees.
For the errors with respect to /gt_pose, 1 cm and 120 degrees (since the orientation measurement was not smooth and had an offset problem mentioned below).

To find the parameters to match /gt_pose we also had to find the rototranslation from world to odom and find an orientation offset of /gt_pose with respect to its trajectory (see below).
Since we did not have any velocities, we used the following heuristic function: we started from the previous values and decreased the transmission ratio to match /gt_pose movement sequence, and then we decreased the apparent baseline so that the curves had a smaller radius (greater curvature).

We obtained the following results:

- Transmission ratio: 1:40.0
- Apparent baseline: 1.64 * B

“ODOM” TO “WORLD” ROTOTRANSLATION

To compute this rototranslation we copied 2 messages with rostopic echo from /scout_odom and /gt_pose.
/gt_pose is not accurate in the first seconds, so we waited for some seconds (about 16s) and as soon as we saw a coherent /gt_pose we copied its pose and the corresponding /scout_odom pose.
We assumed that at the beginning /scout_odom is correct, so we can use its pose as reference to find the rototranslation from "odom" (frame_id of /scout_odom) to "world" (frame id of /gt_pose). This has been computed in MATLAB (in the script rototranslation.m).
Doing so we noticed that there was an inconsistency between the /gt_pose theta orientation and the direction of motion of /gt_pose, as if the orientation vector were not indicating the motion direction (so it was not tangent to the trajectory). As a matter of fact, the orientation was very accurate but the trajectory seemed to have a different rotation.
We made the hypothesis that the orientation field in /gt_pose could have an offset wrt the motion of the robot.

“ODOM” TO “WORLD” ROTOTRANSLATION

Therefore, we tuned CHI, Inv_RATIO, the rotation of the trajectory (so the rotation from "world" to "odom") and the constant offset of the orientation of /gt_pose.

We noticed that this offset was very consistent (in the first 60 seconds of simulation it is almost constant) and was about 13.5 degrees.

Therefore, we fixed the reference system accordingly (and we removed this offset in the residuals node when computing the difference between /gt_pose orientation and our computed orientation).

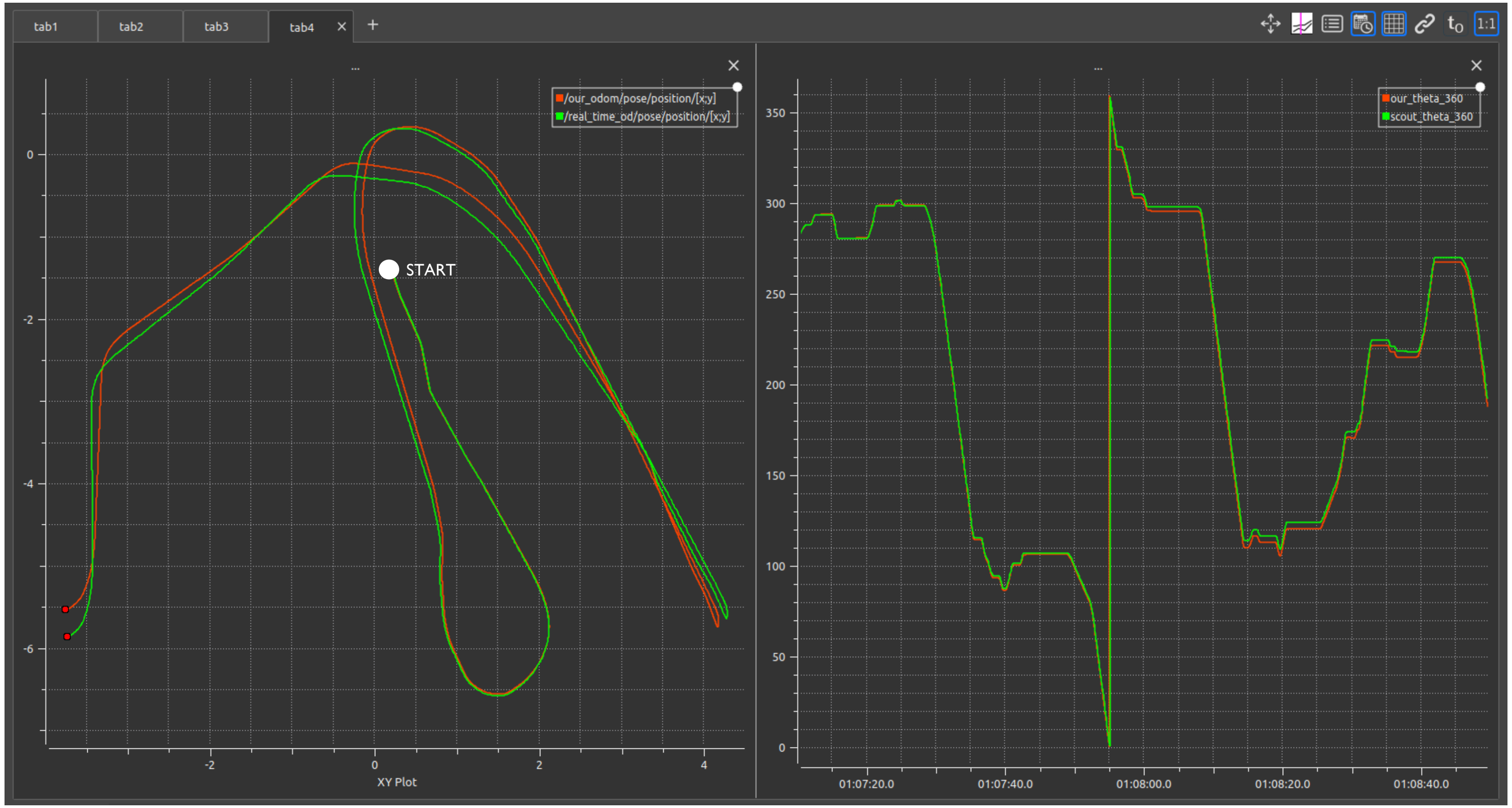
In the end the “world” reference frame turned out to have a yaw rotation of about 67 deg with respect to “odom” frame, and /gt_pose has an orientation offset of about 13.5 degrees with respect to its trajectory.

EXPERIMENT 1 - /scout_odom

POLITECNICO MILANO 1863

We will now show some pictures of the results we got from PlotJuggler.

The first picture shows on the left the XY trajectory (in meters) of the computed odometry versus the odometry taken from /scout_odom with NON optimized parameters. On the right the orientation of the robot is shown (in degrees).

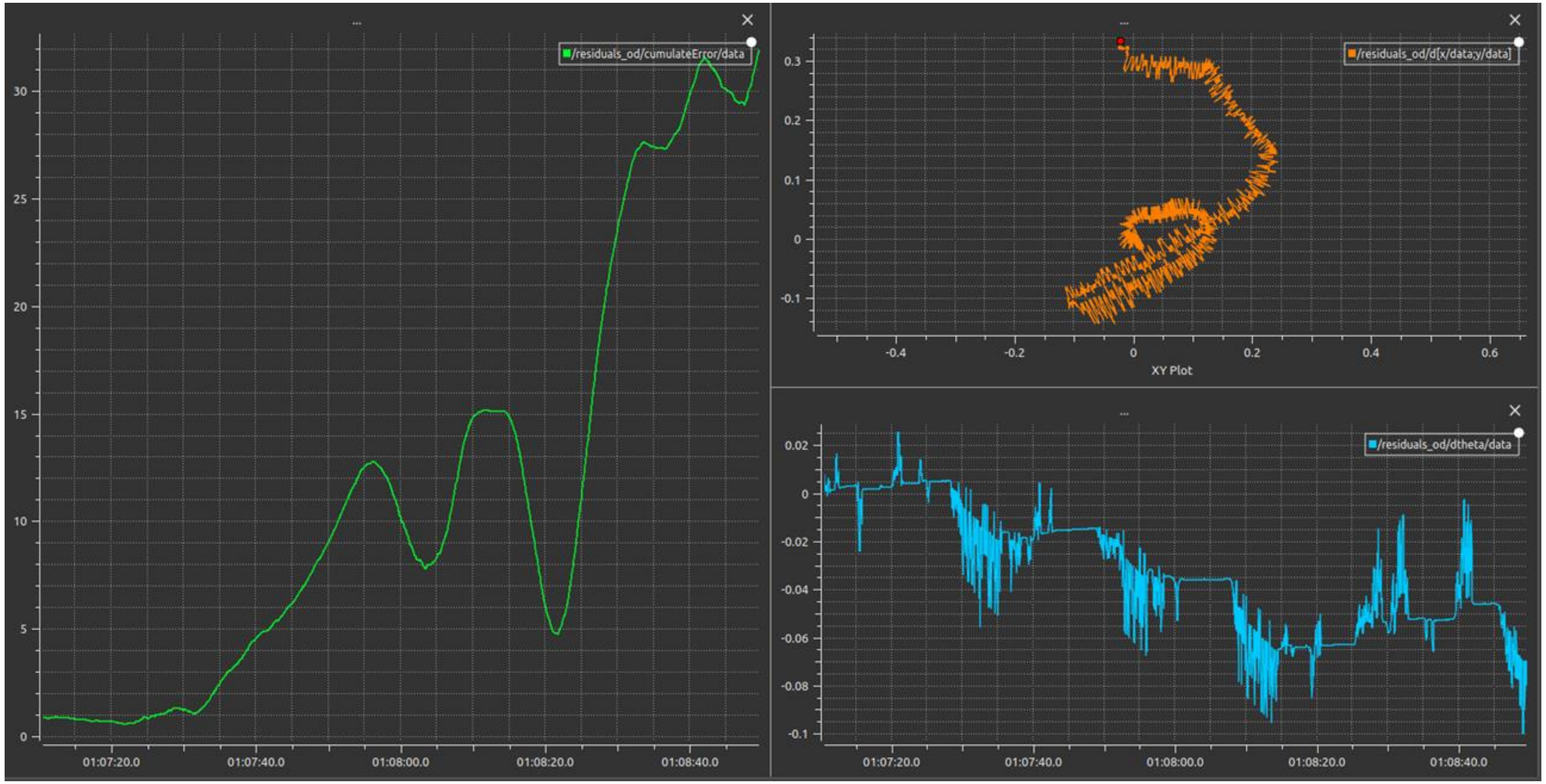


CHI: 1.76, RATIO: 1:38.3, Simulation time: 1’30”

EXPERIMENT 1 - /scout_odom

Now we can analyze the errors with respect to /scout_odom.

In particular on the left the cumulateError is shown, while on the top right there is the XY error plot [m] of dx and dy. On the bottom right dtheta [rad] over time is shown.

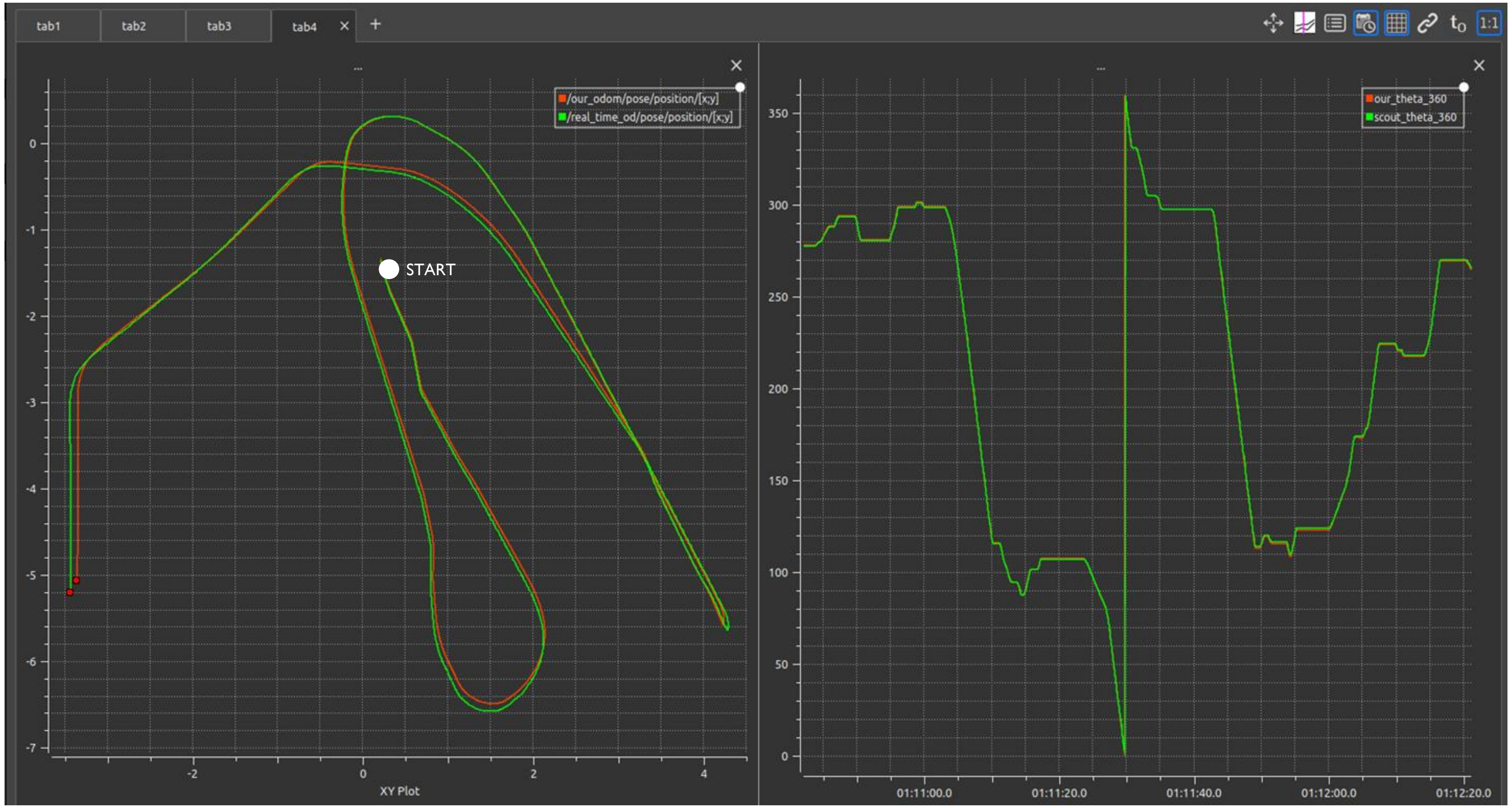


CHI: 1.76, RATIO: 1:38.3, Simulation time: 1’30”

EXPERIMENT 2 - /scout_odom

After optimizing the parameters we managed to get a better result. The tracking is very accurate.

- our computed odometry
- /scout_odom

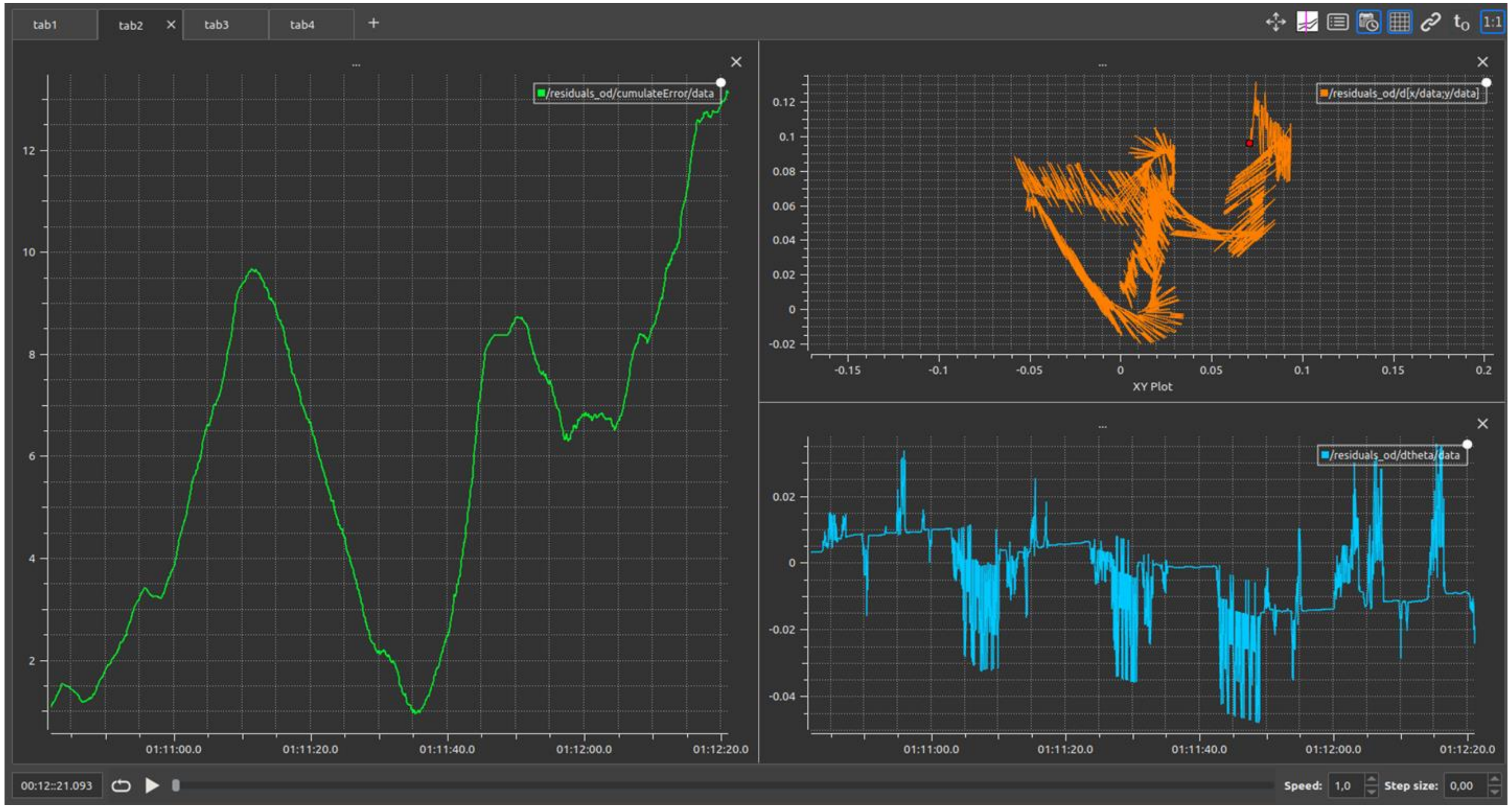


CHI: 1.75, RATIO: 1:38.7, Simulation time: 1’30”

EXPERIMENT 2 - /scout_odom

POLITECNICO MILANO 1863

With the optimized parameters the cost function decreased a lot (the maximum value decreases from about 33 to about 13). Also the XY error plot shows that we are much closer to /scout_odom.



CHI: 1.75, RATIO: 1:38.7, Simulation time: 1’30”

EXPERIMENT 3 - /gt_pose

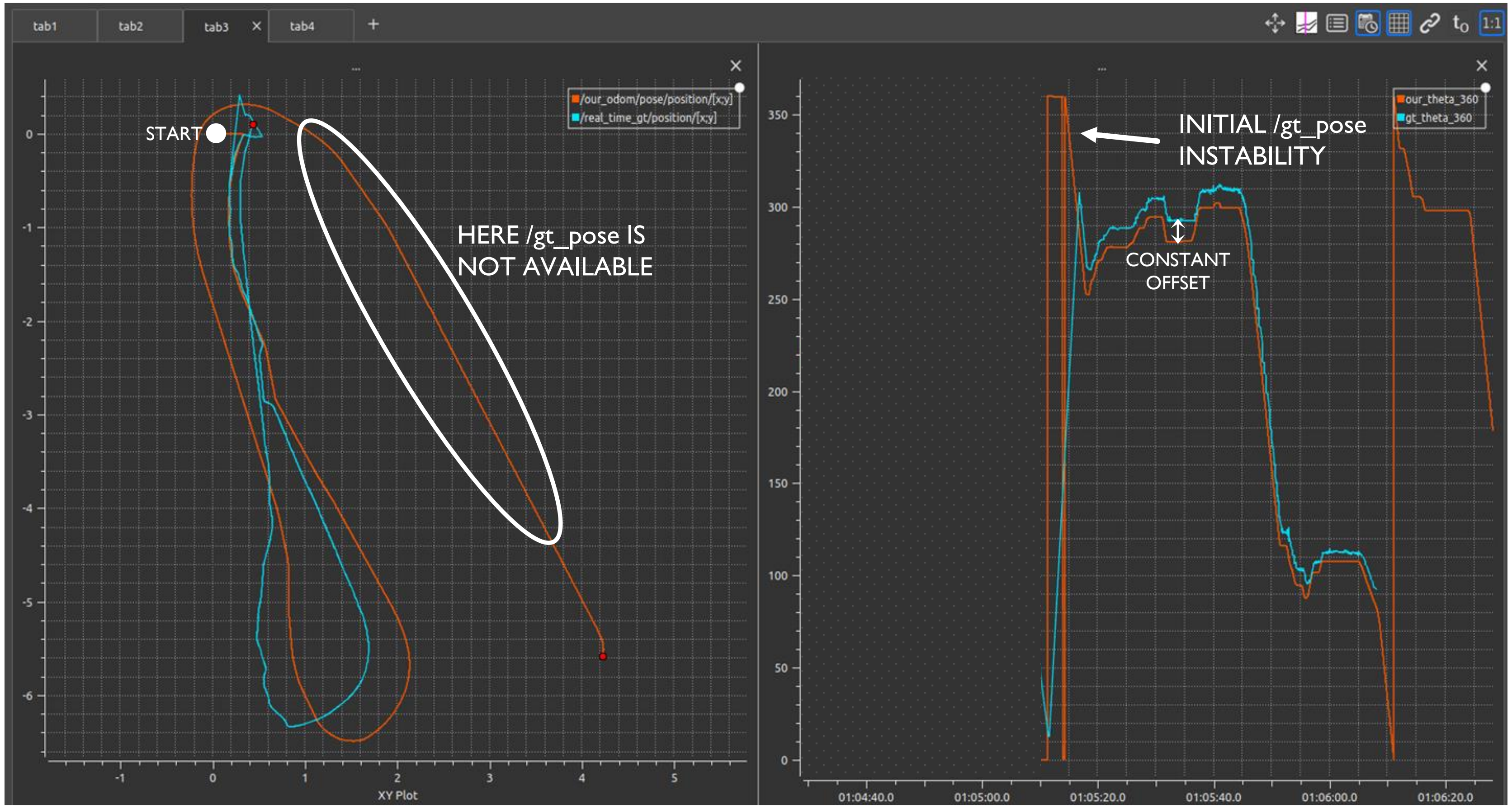
POLITECNICO MILANO 1863

We tried to compare our computed odometry with the optimized parameters to /gt_pose (in "odom" reference frame).

The results were not good.

The simulation could only last 40 seconds due to /gt_pose unavailability.

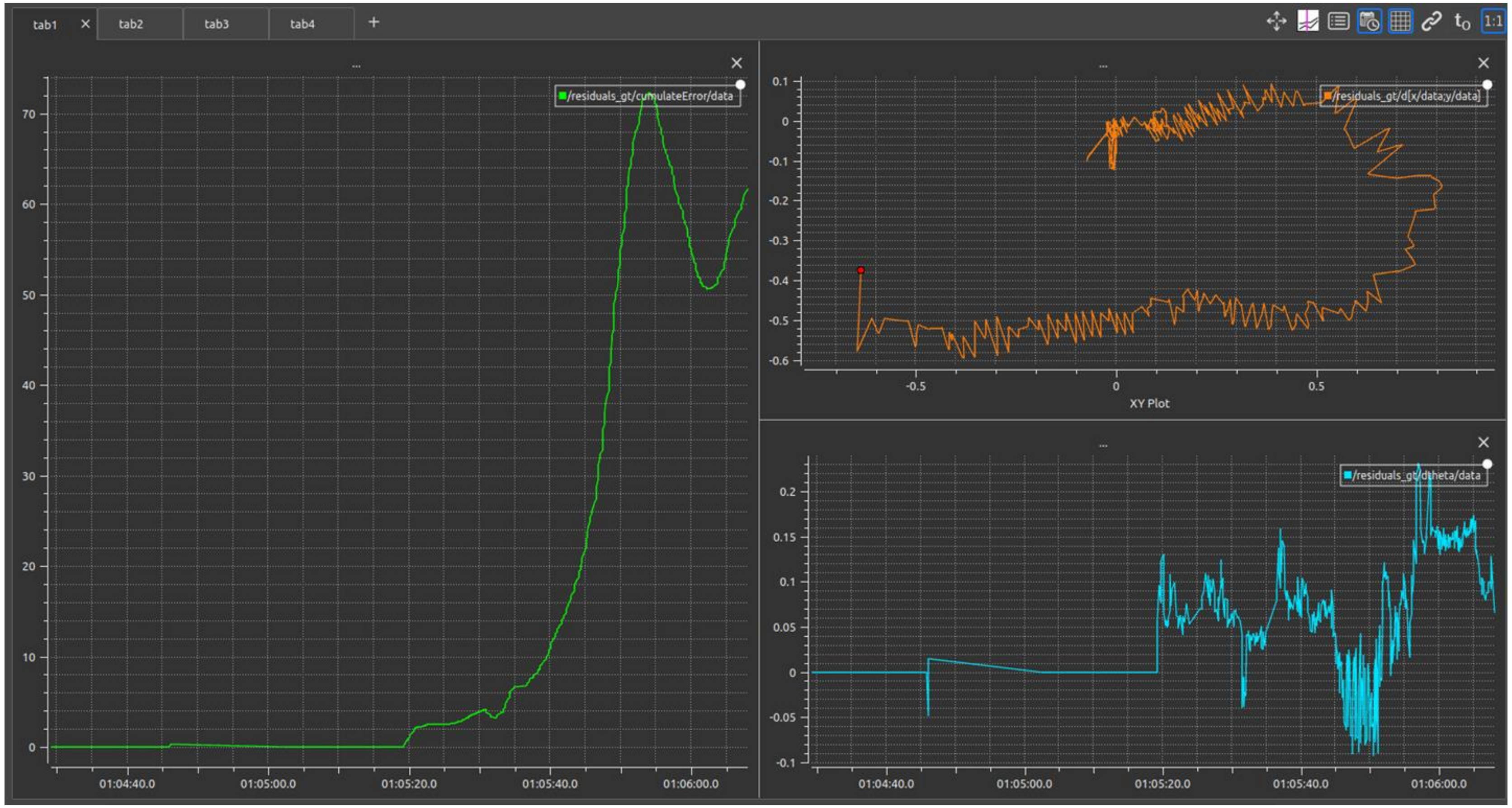
Notice that there is an almost constant offset in the theta plot (on the right), which is the orientation offset mentioned above.



CHI: 1.75, RATIO: 1:38.7, Simulation time: 40"

EXPERIMENT 3 - /gt_pose

This is the cost function we would like to improve with a better parameter tuning.

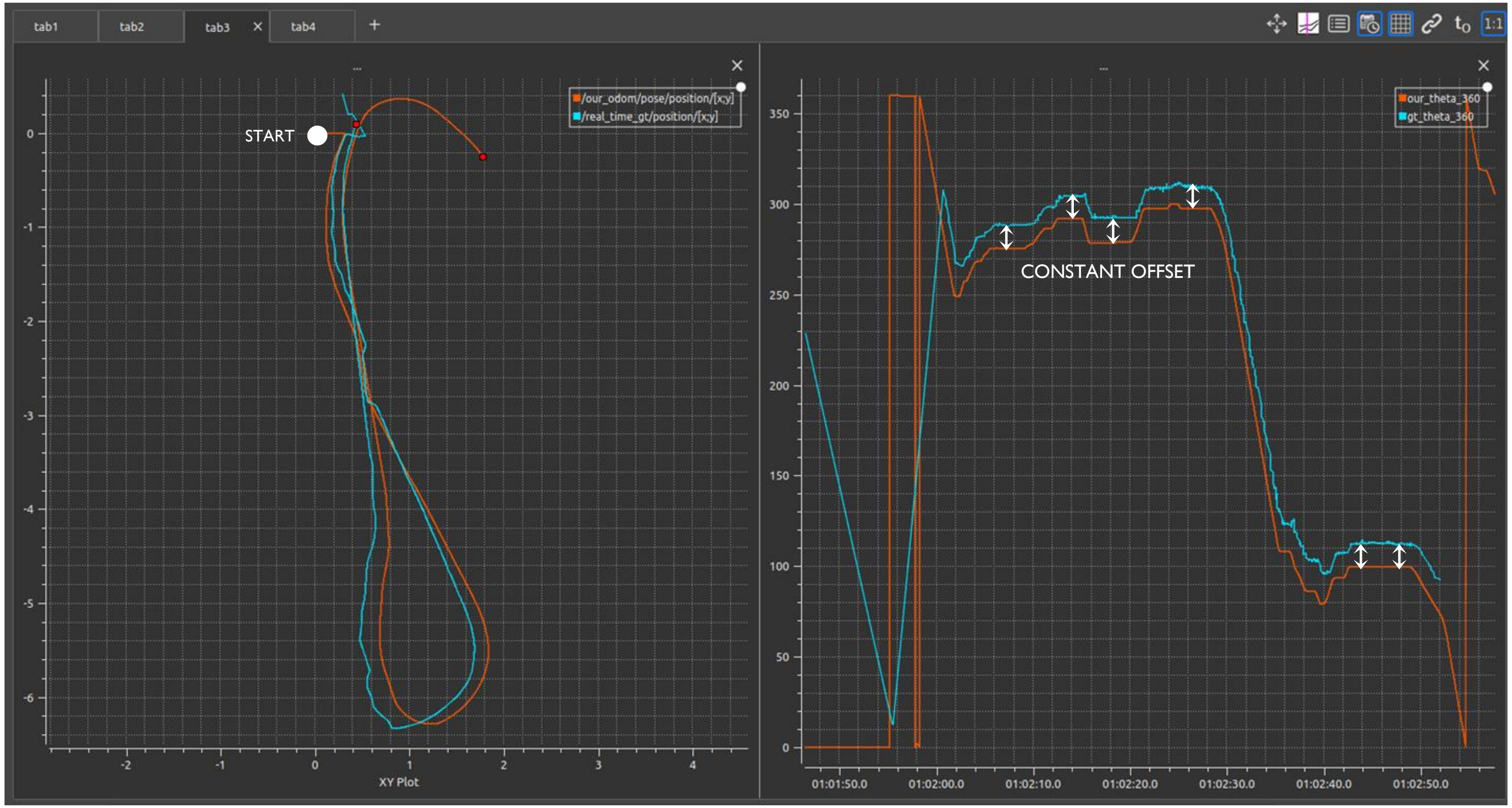


CHI: 1.75, RATIO: 1:38.7, Simulation time: 40"

EXPERIMENT 4 - /gt_pose

After optimizing the parameters we managed to get a better result.

- our computed odometry
- /gt_pose

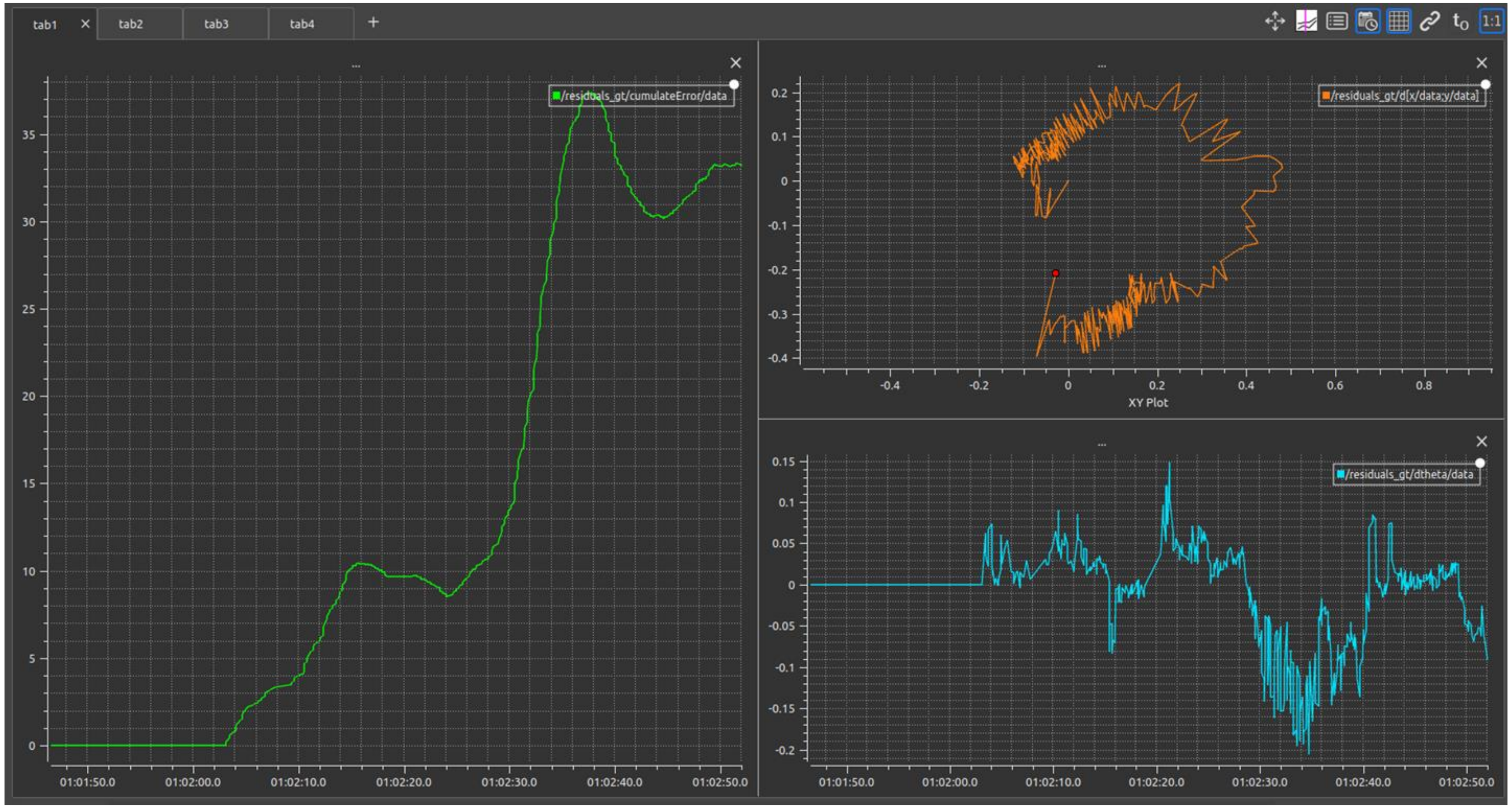


CHI: 1.64, RATIO: 1:40.0, Simulation time: 40”

EXPERIMENT 4 - /gt_pose

POLITECNICO MILANO 1863

With the optimized parameters the cost function decreased a lot (the maximum value decreases from about 75 to about 38). Also the XY and the theta error plot shows that we are much closer to /scout_odom.



CHI: 1.64, RATIO: 1:40.0, Simulation time: 40''

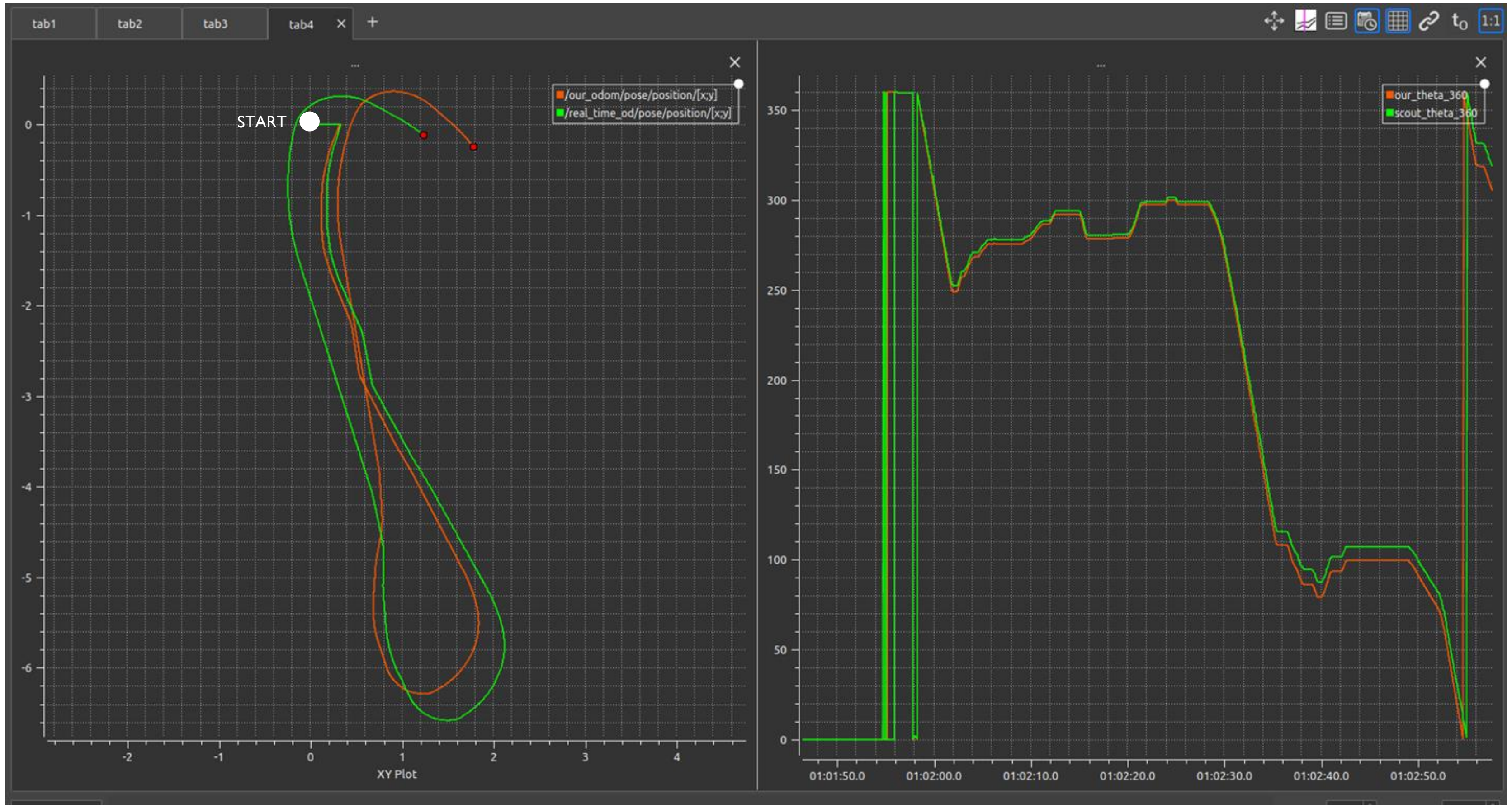
EXPERIMENT 5 - /scout_odom

POLITECNICO MILANO 1863

Here is a comparison with /scout_odom with the optimal parameter for /gt_pose.

- our computed odometry
- /scout_odom

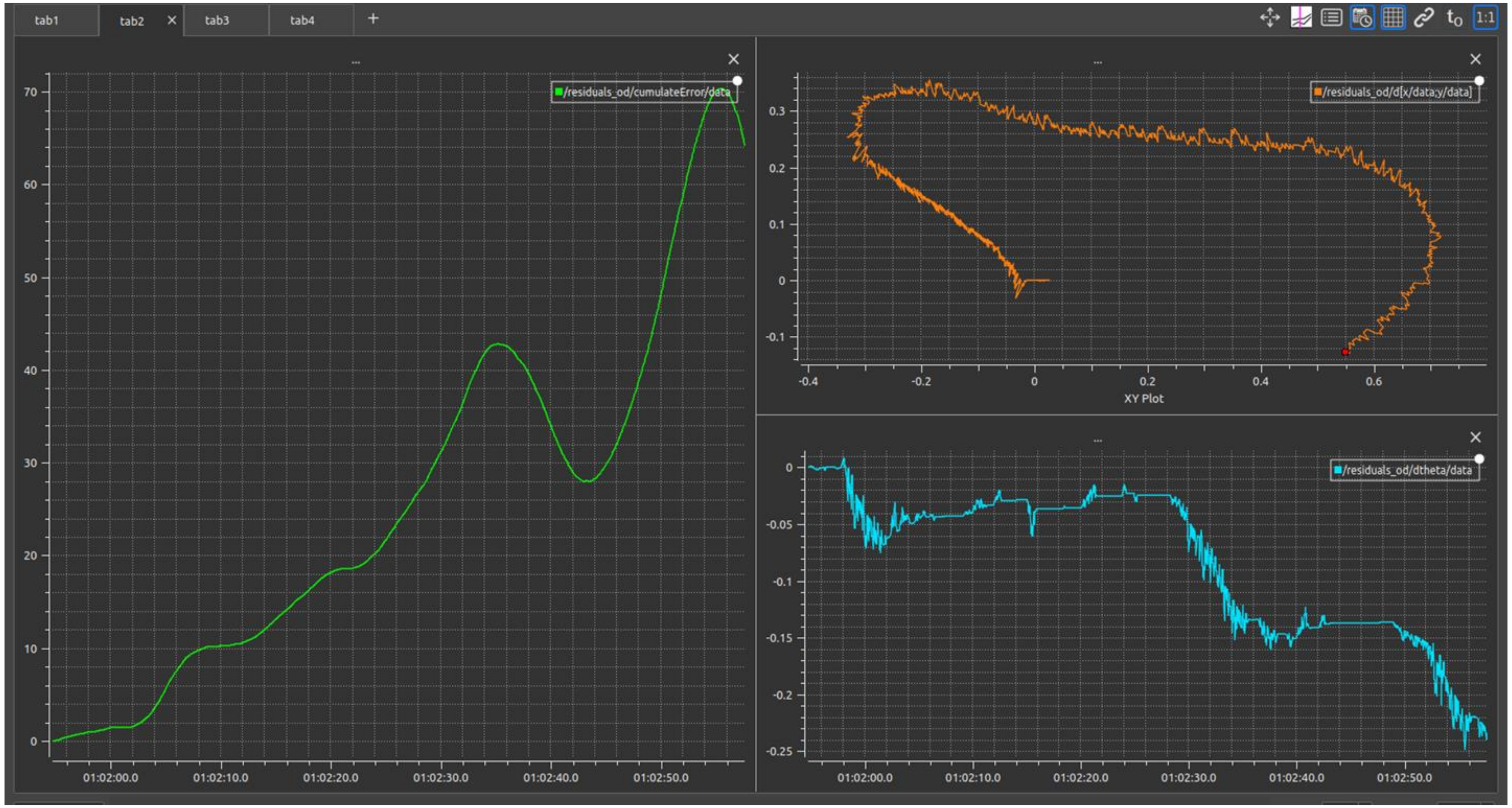
The result is acceptable but worse than the one optimized for /scout_odom.



CHI: 1.64, RATIO: 1:40.0, Simulation time: 40”

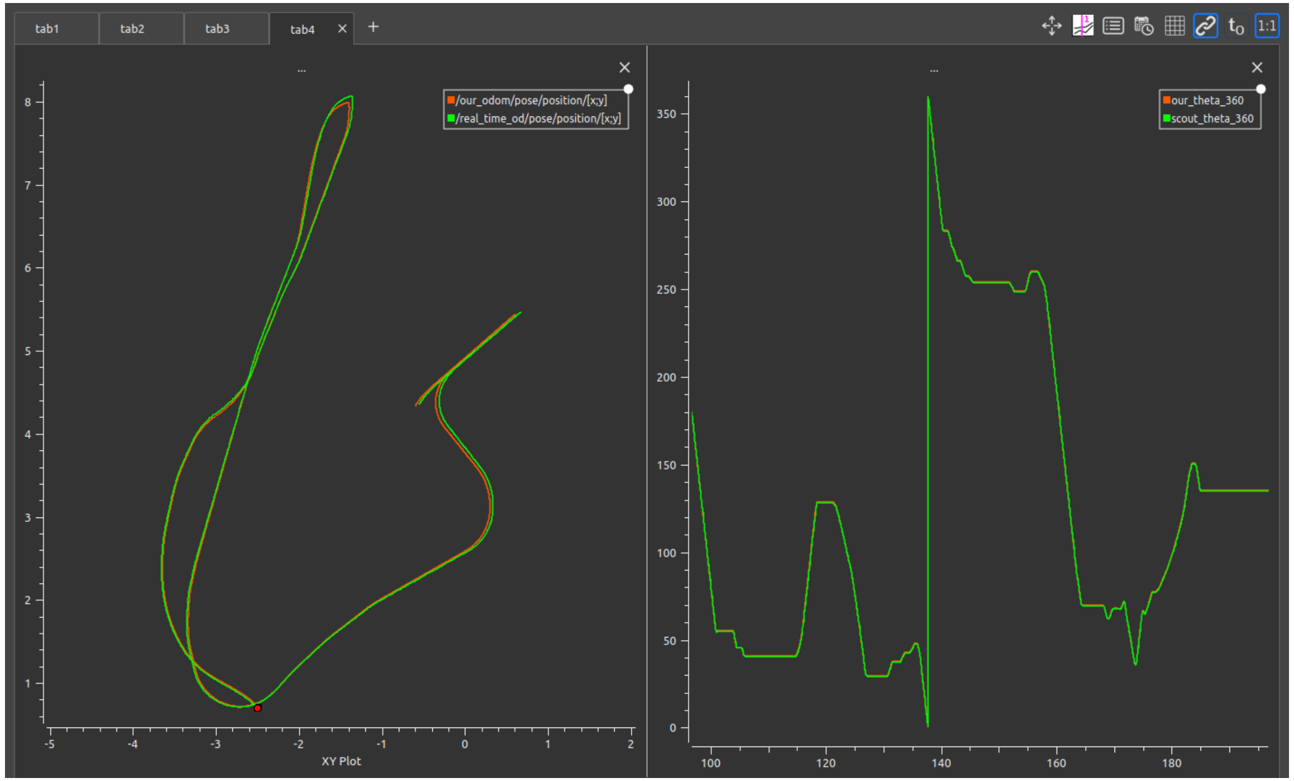
EXPERIMENT 5 - /scout_odom

This is the cost function for the last experiment, it can be noticed that the cost is much higher than the optimal one (which had a maximum value of about 13).



CHI: 1.75, RATIO: 1:38.7, Simulation time: 40”

VALIDATION 1 – bag2

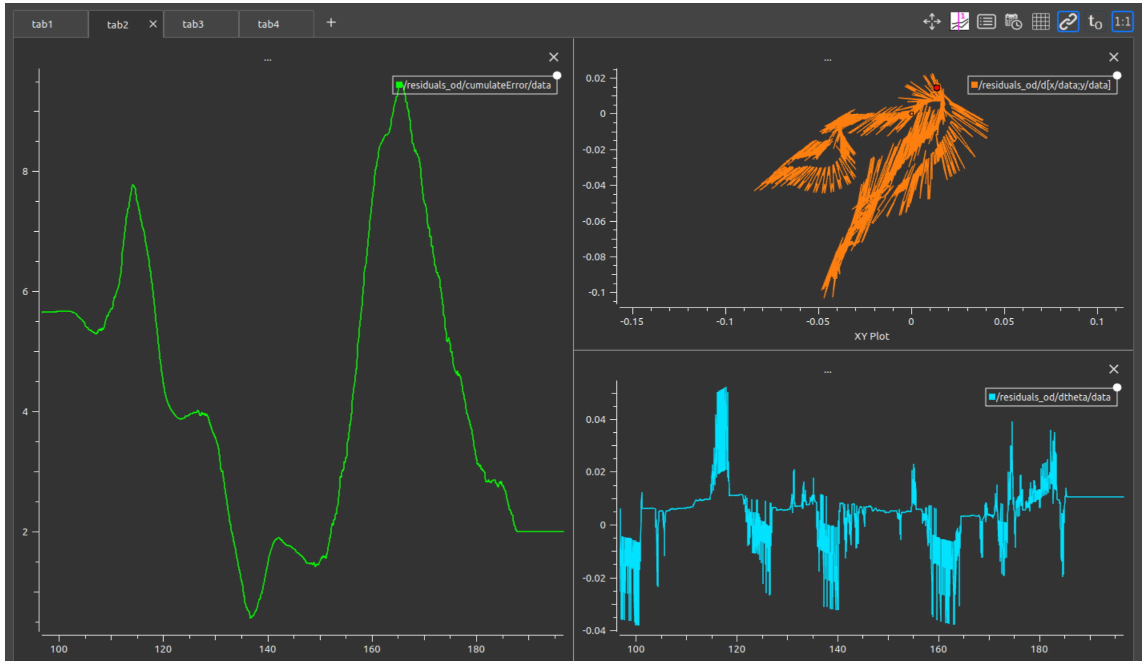


CHI: 1.75, RATIO: 1:38.7

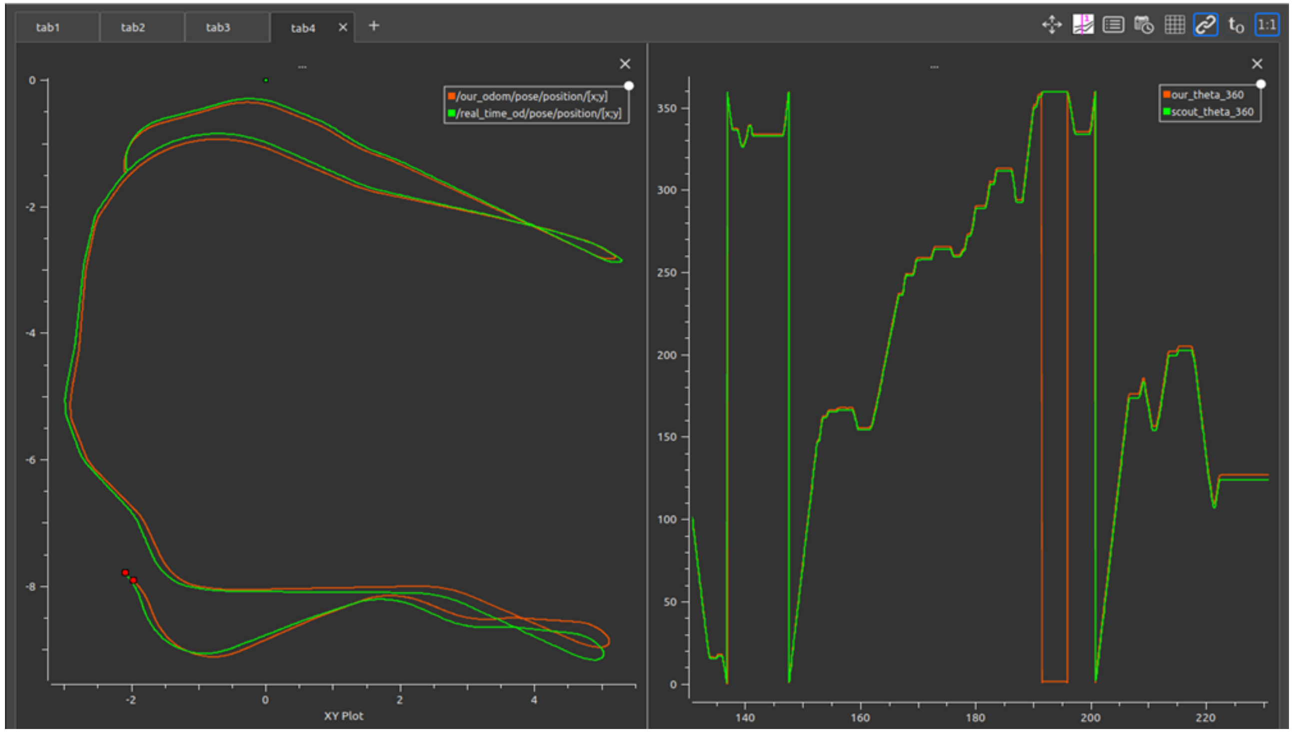
This is the trajectory and orientation of the second bag (the parameters are the ones optimized for /scout_odom)

- our computed odometry
- /scout_odom

Cost function



VALIDATION 2 – bag3



This is the trajectory and orientation of the second bag (the parameters are the ones optimized for /scout_odom)

- our computed odometry
- /scout_odom

CHI: 1.75, RATIO: 1:38.7

Cost function