Bisio Isabella-Sole
Roncagliolo Serena

Localization Lab

OVERESTIMATING VS UNDERESTIMATING

Standard deviation sigmaTuning

We have observed that underestimating the value *sigmaTuning,* the end position and the start position are different.

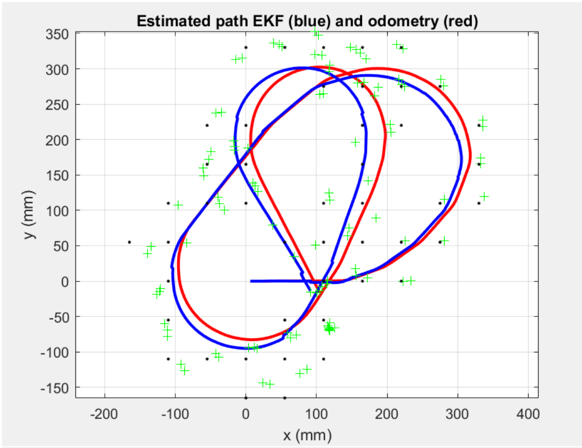
This behaviour can be explained looking at the Malahanobis distances graph which helps to understand the origin of the problem. When *sigmaTuning* is underestimated, it means that we are overconfident of our odometry, so we consider the odometry to be correct when it is instead incorrect.

The variance of Y^ (expected measurement) which is equal to (C*P*CT) results too low. This causes the sum (C*P*CT + Qgamma) is also low, which means its inverse (C*P*CT)^(-1) is very high. Since this is the central term of the Mahalanobis distance, the Mahalanobis distance is going to be a high value and the algorithm is going to reject measurements. We may say that we are "over trusting" our X^ (estimated position) and our Y^.

Over trusting the estimated values means that we are very "strict" in judging the matching between measurements and expected measurements and this leads me to reject many measurements.

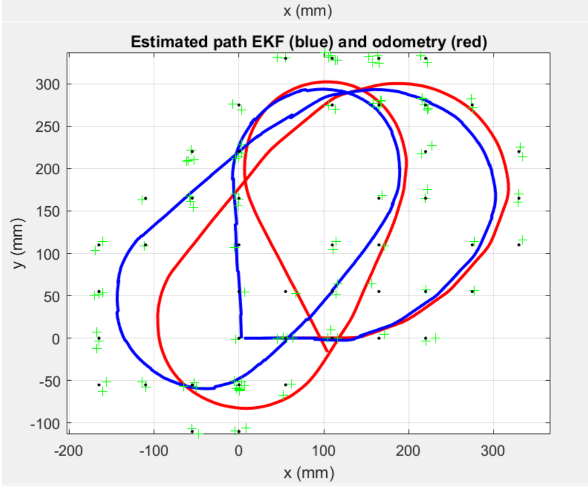
When the algorithm rejects possible correct measurements, the robot gets lost. By doing so, we keep taking subsequent measurements, but they will not be the correct ones anymore and the path followed will be incorrect. We can observe that much more than the 5% (expected value of rejected measurements) will be rejected. We know it would be 5, if everything were correctly tuned and every noise were gaussian.

Regarding the path figure (for the one that run correctly), the black dots are the sensors that have been detected and the green crosses are the estimated positions of the magnet in the abs frame when a measurement occurs. When the robot is lost these green crosses have no sense since the program continues to run.

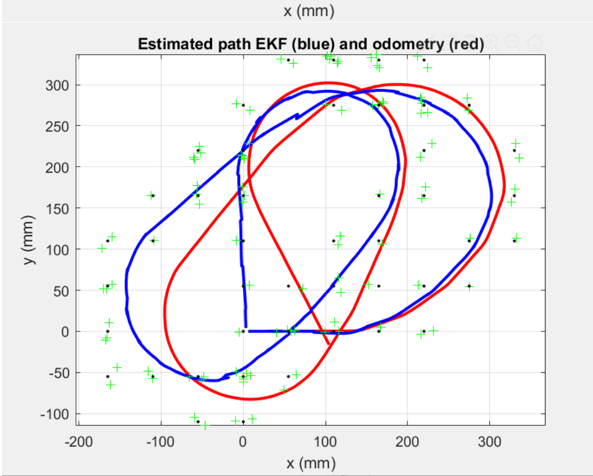


Underestimated *(sigmaTuning = 0.01)*

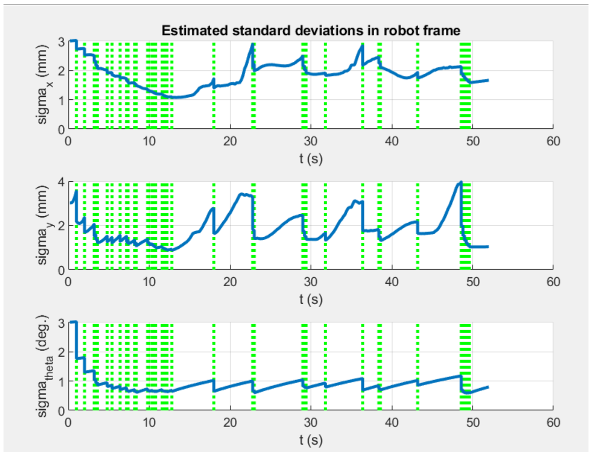
End position does not coincide with start position



Correct value *(sigmaTuning = 0.1)*

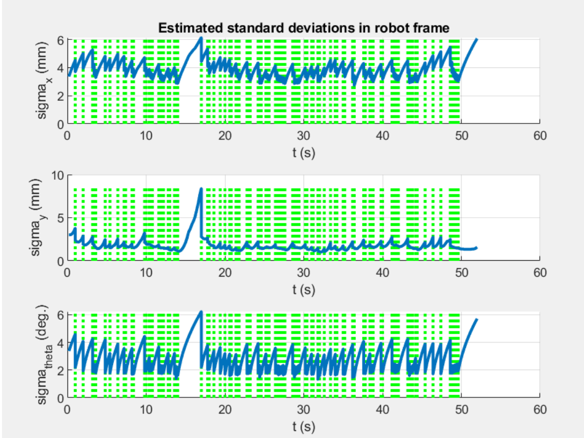


Overestimated *(sigmaTuning = 1)*

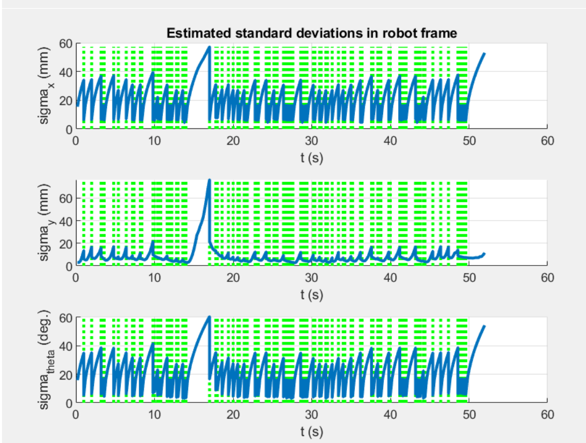


Underestimated *(sigmaTuning = 0.01)*

The green lines represent the detected magnet. As explained before, when we underestimate the value of sigmaTuning, we detect less magnets, which means the green lines are scarcer

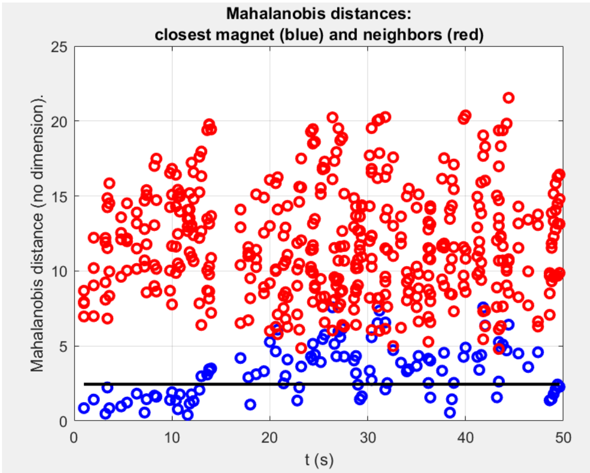


Correct value *(sigmaTuning = 0.1)*



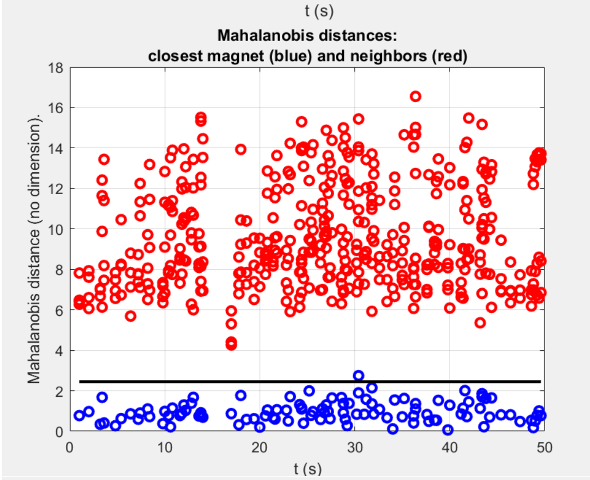
Overestimated *(sigmaTuning = 1)*

In this case we detect more magnets, which means there will be more green lines

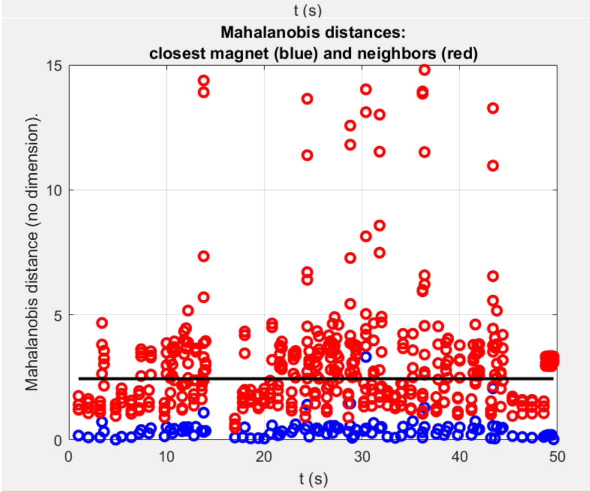


Underestimated *(sigmaTuning = 0.01)*

Understimating *sigmaTuning,* we reject many correct measurements (blue dots are above the threshold).



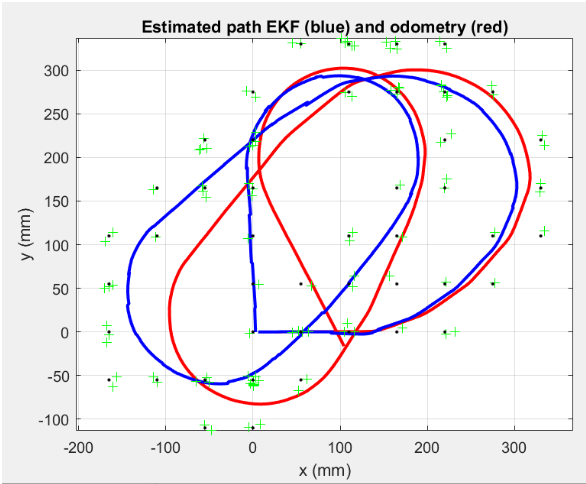
Correct value *(sigmaTuning = 0.1)*



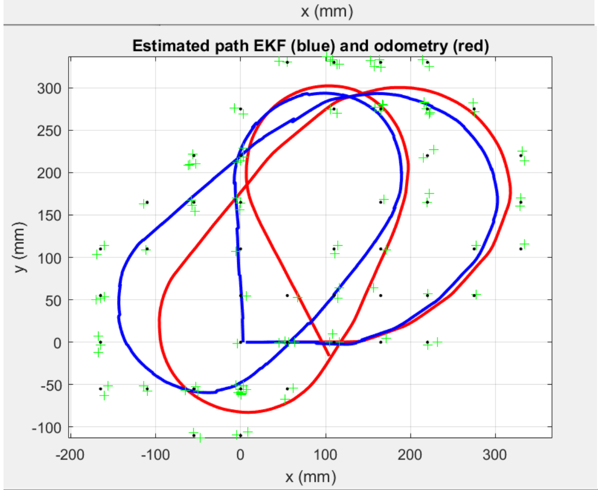
Overestimated *(sigmaTuning = 1)*

This is a dangerous situation, because almost all the red dots are below the threshold. This means the robot would locate magnets incorrectly and follow a wrong trajectory

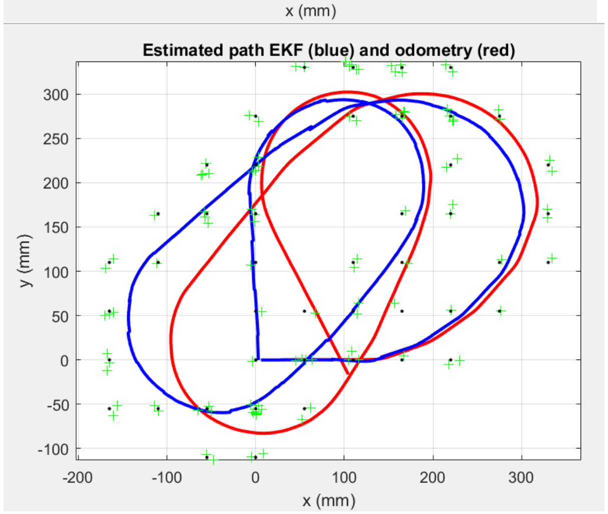
Initial robot position variance



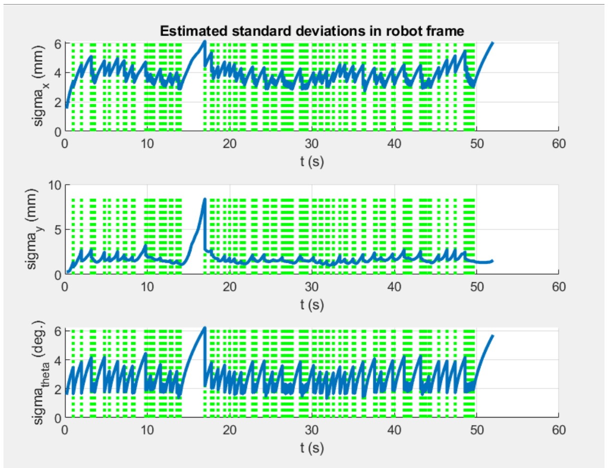
sigmaX = 0.3
sigmaY = 0.3
sigmaTheta = 0.3*pi/180



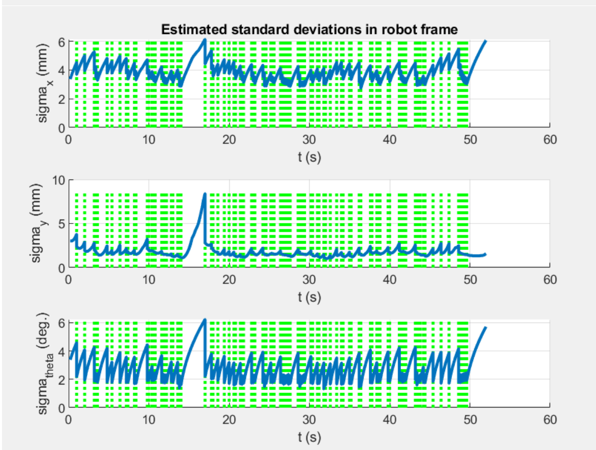
sigmaX = 3
sigmaY = 3
sigmaTheta = 3*pi/180



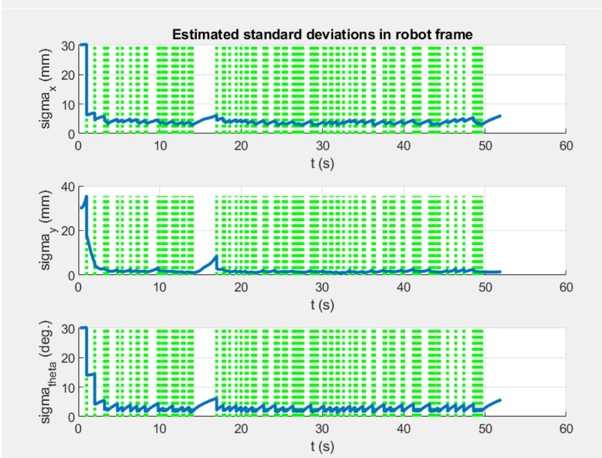
sigmaX = 30
sigmaY = 30
sigmaTheta = 30*pi/180



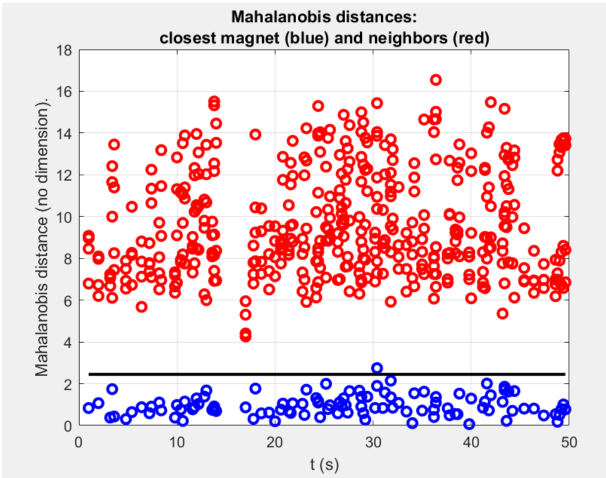
sigmaX = 0.3
sigmaY = 0.3
sigmaTheta = 0.3*pi/180



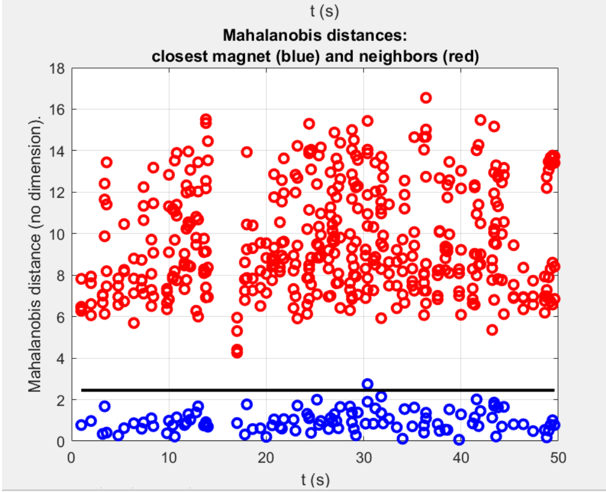
sigmaX = 3
sigmaY = 3
sigmaTheta = 3*pi/180



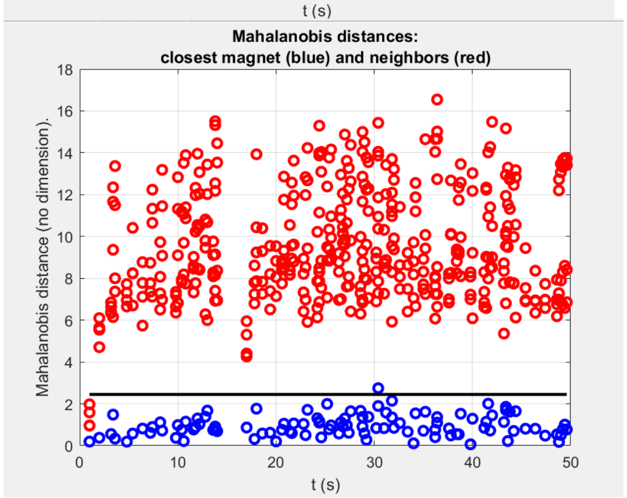
sigmaX = 30
sigmaY = 30
sigmaTheta = 30*pi/180



sigmaX = 0.3
sigmaY = 0.3
sigmaTheta = 0.3*pi/180

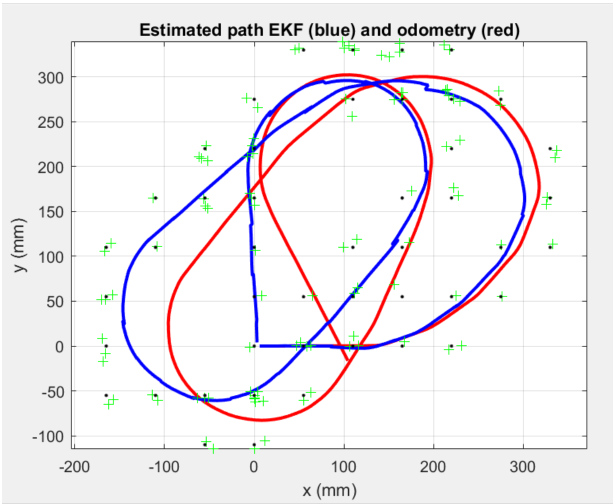


sigmaX = 3
sigmaY = 3
sigmaTheta = 3*pi/180

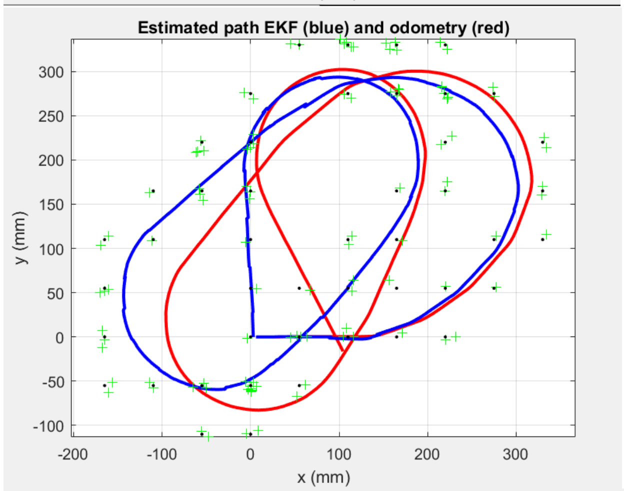


sigmaX = 30
sigmaY = 30
sigmaTheta = 30*pi/180

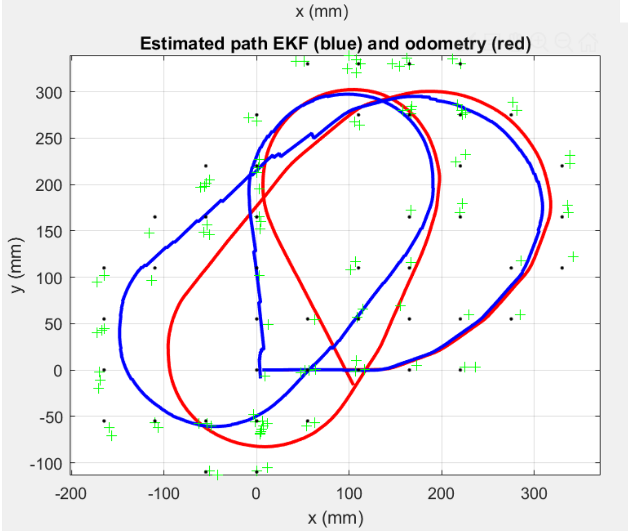
Measurement noise variance



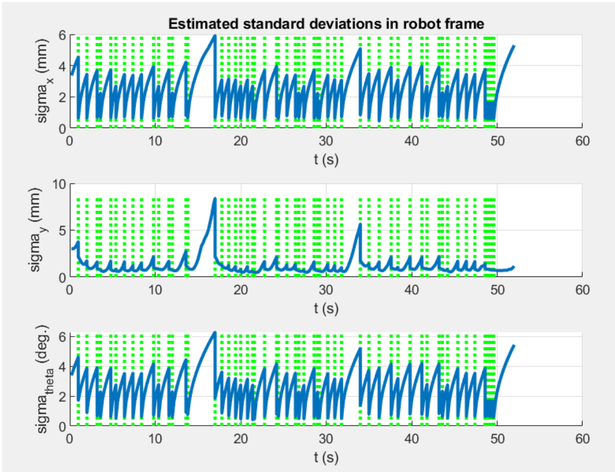
sigmaXmeasurement = (sqrt(22^2/12))*0.1
sigmaYmeasurement = (sqrt(10^2/12))*0.1



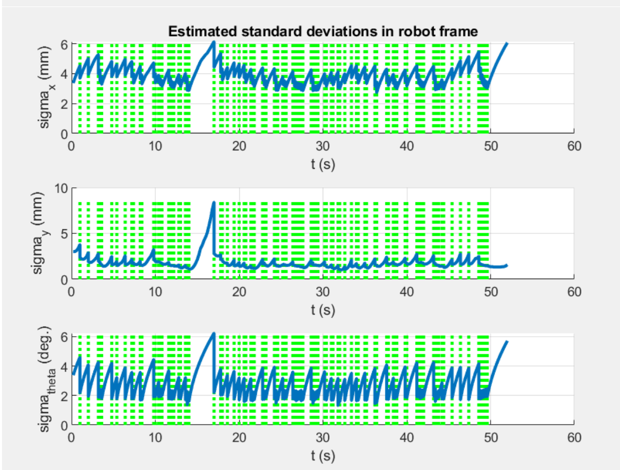
sigmaXmeasurement = (sqrt(22^2/12))
sigmaYmeasurement = (sqrt(10^2/12))



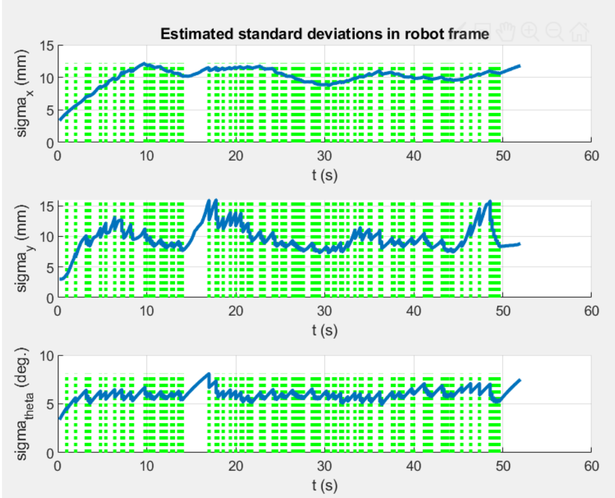
sigmaXmeasurement = (sqrt(22^2/12)) *10
sigmaYmeasurement = (sqrt(10^2/12))*10



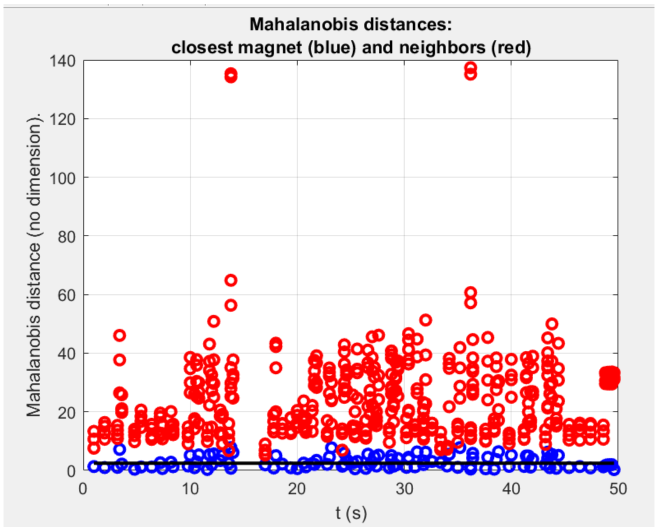
sigmaXmeasurement = (sqrt(22^2/12))*0.1
sigmaYmeasurement = (sqrt(10^2/12))*0.1



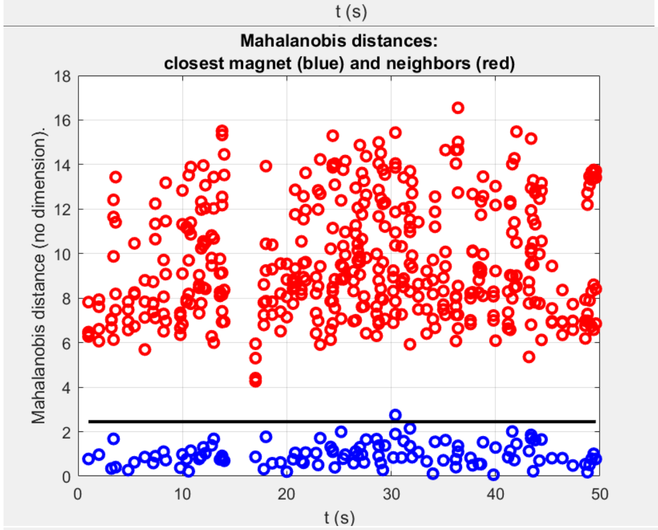
sigmaXmeasurement = (sqrt(22^2/12))
sigmaYmeasurement = (sqrt(10^2/12))



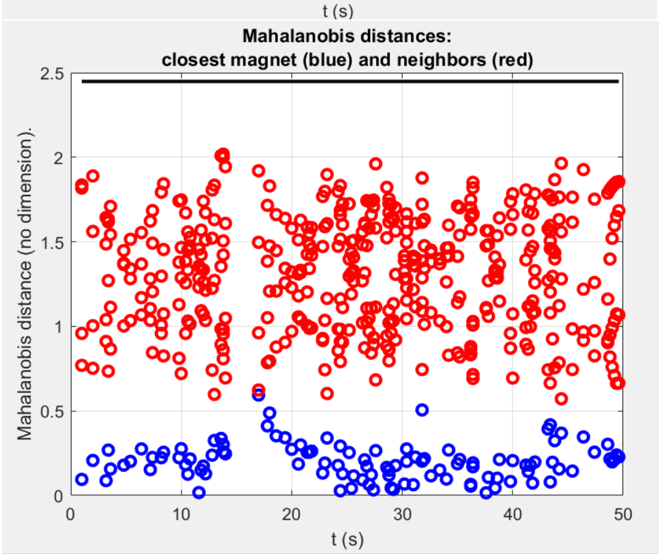
sigmaXmeasurement = (sqrt(22^2/12)) *10
sigmaYmeasurement = (sqrt(10^2/12))*10



sigmaXmeasurement = (sqrt(22^2/12))*0.1
sigmaYmeasurement = (sqrt(10^2/12))*0.1



sigmaXmeasurement = (sqrt(22^2/12))
sigmaYmeasurement = (sqrt(10^2/12))



sigmaXmeasurement = (sqrt(22^2/12)) *10
sigmaYmeasurement = (sqrt(10^2/12))*10