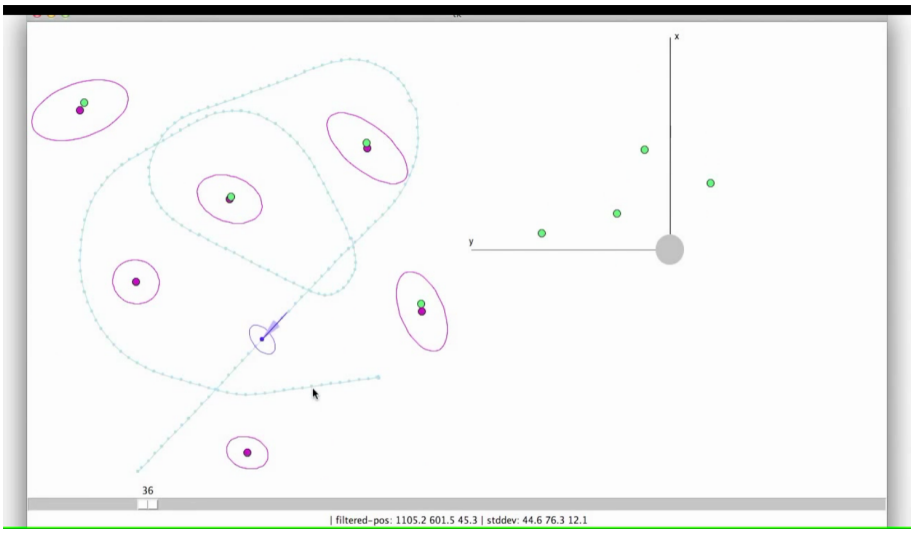
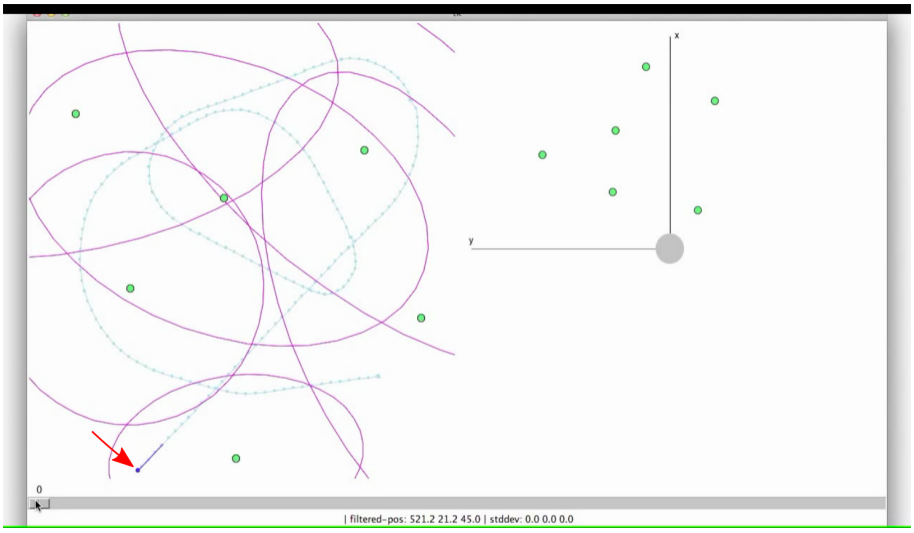
The robot's position is a big blue dot. Around the robot's position is the position uncertainty ellipse, in blue, and finally, the$\pm\,\sigma_{\theta_t}$ of uncertainty in heading. At this position the robot sees 4 landmarks relative to its own coordinate system. These landmarks are depicted in green in the robot's coordinate system and they are also depicted in green in the world coordinate system. So, these green dots are simply the measurements transformed into the world coordinate system according to the robot's current location and orientation. Remember, the world's coordinate system is the robot's first coordinate system. The estimated position for each landmark is depicted in magenta. These positions are also part of our system state now, in addition to the robot's estimated pose. The landmarks' position uncertainty ellipses are also depicted in magenta.



In the beginning the robot starts at an arbitrary position and orientation in the world. At this moment, we have defined that the robot's uncertainty in position and heading is zero. In the SLAM algorithm the built map has its origin at the robot's initial position.



The robot observes six cylinders from its initial pose. The discovered landmarks' coordinates are added to the

algorithm's state. Besides, the position initial uncertainty of each discovered landmark is added to the algorithm's covariance matrix. All this data is added to the algorithm using the function `add_landmark_to_state (initial_coords)`.

$$\vec{\mu}_t = \begin{pmatrix} \mu_{x_t} \\ \mu_{y_t} \\ \mu_{\theta_t} \end{pmatrix} \implies \vec{\mu}_t = \begin{pmatrix} \mu_{x_t} \\ \mu_{y_t} \\ \mu_{\theta_t} \\ \mu_{x_{W\,1}} \\ \mu_{y_{W\,1}} \\ \vdots \\ \mu_{x_{W\,N}} \\ \mu_{y_{W\,N}} \end{pmatrix}$$

$$\Sigma_t = \begin{pmatrix} \sigma^2_{x_t} & \sigma_{x_t y_t} & \sigma_{x_t \theta_t} \\ \sigma_{x_t y_t} & \sigma^2_{y_t} & \sigma_{y_t \theta_t} \\ \sigma_{x_t \theta_t} & \sigma_{y_t \theta_t} & \sigma^2_{\theta_t} \end{pmatrix}$$

$$\Downarrow$$

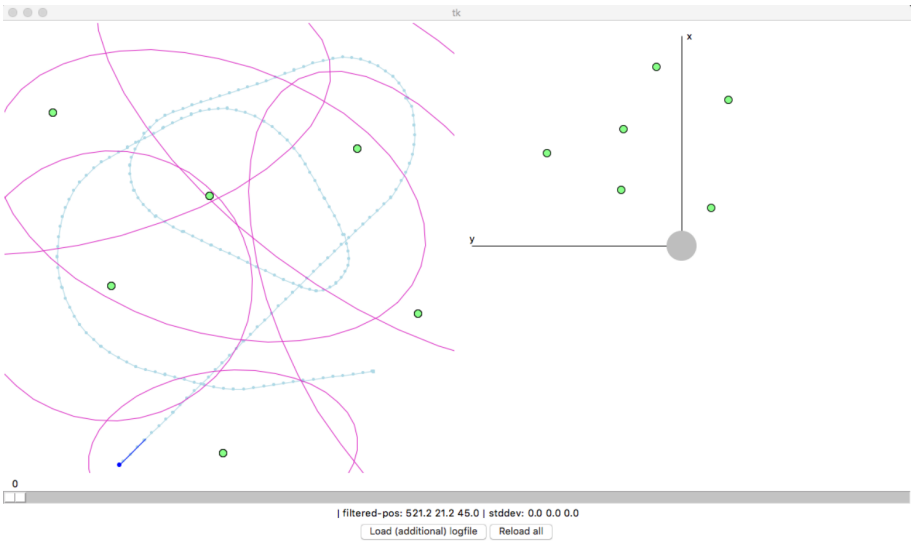
$$\Sigma_t = \begin{pmatrix} \sigma^2_{x_t} & \sigma_{x_t y_t} & \sigma_{x_t \theta_t} & 0 & 0 & \dots & 0 & 0 & \dots & 0 & 0 \\ \sigma_{x_t y_t} & \sigma^2_{y_t} & \sigma_{y_t \theta_t} & 0 & 0 & \dots & 0 & 0 & \dots & 0 & 0 \\ \sigma_{x_t \theta_t} & \sigma_{y_t \theta_t} & \sigma^2_{\theta_t} & 0 & 0 & \dots & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & 0 & \sigma^2_{x_{W\,0}} & 0 & \dots & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & 0 & 0 & \sigma^2_{y_{W\,0}} & \dots & 0 & 0 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & \dots & \sigma^2_{x_{W\,j}} & 0 & \dots & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \dots & 0 & \sigma^2_{y_{W\,j}} & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & \dots & 0 & 0 & \dots & \sigma^2_{x_{W\,N}} & 0 \\ 0 & 0 & 0 & 0 & 0 & \dots & 0 & 0 & \dots & 0 & \sigma^2_{y_{W\,N}} \end{pmatrix}$$

where $\sigma^2_{x_{W\,0}} = \sigma^2_{y_{W\,0}} = \dots = \sigma^2_{x_{W\,N}} = \sigma^2_{y_{W\,N}} = \infty$ when these values are added to the algorithm's covariance matrix (for practical purposes instead of ∞ we use 100 meters). These initial uncertainty ellipses (in fact when $\sigma^2_{x_{W\,j}} = \sigma^2_{y_{W\,j}}$, the major axis' length is equal the minor axis' length, so we have a circle) are so big that we can't observe them in the GUI.

Main part of the algorithm:

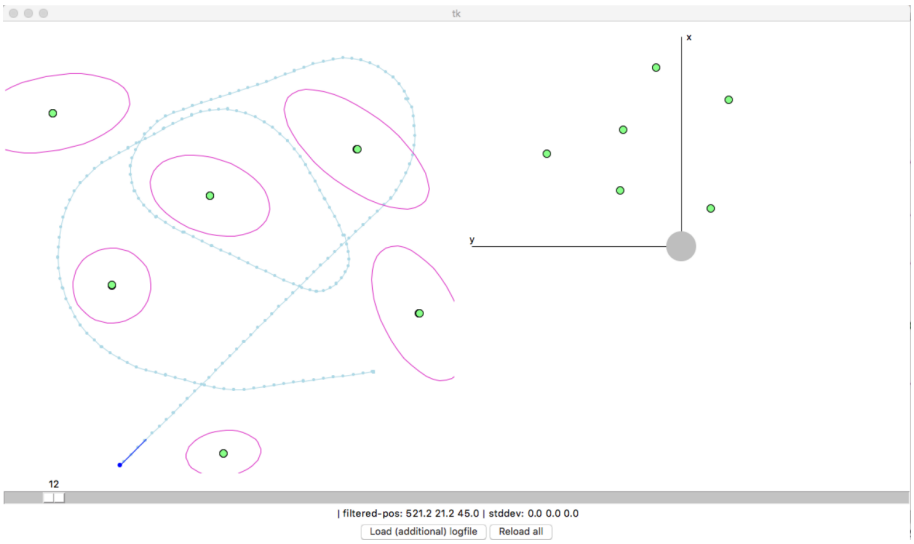
```
# Give values to algorithm's parameters
# Read data from files
for i in xrange(len(logfile.motor_ticks)): # Loop A
    # Prediction.
    control = array(logfile.motor_ticks[i]) * ticks_to_mm
    kf.predict(control)
    # Correction.
    observations = get_observations(logfile.scan_data[i], depth_jump,
      minimum_valid_distance, cylinder_offset, kf, max_cylinder_distance)
    for obs in observations:              # Loop B
        measurement, cylinder_world, cylinder_scanner, cylinder_index = obs
        if cylinder_index == -1:
            cylinder_index = kf.add_landmark_to_state(cylinder_world)
        kf.correct(measurement, cylinder_index)
    # Write data  to file
```

After the for loop labeled as B finishes its execution for the first time the algorithm has obtained:

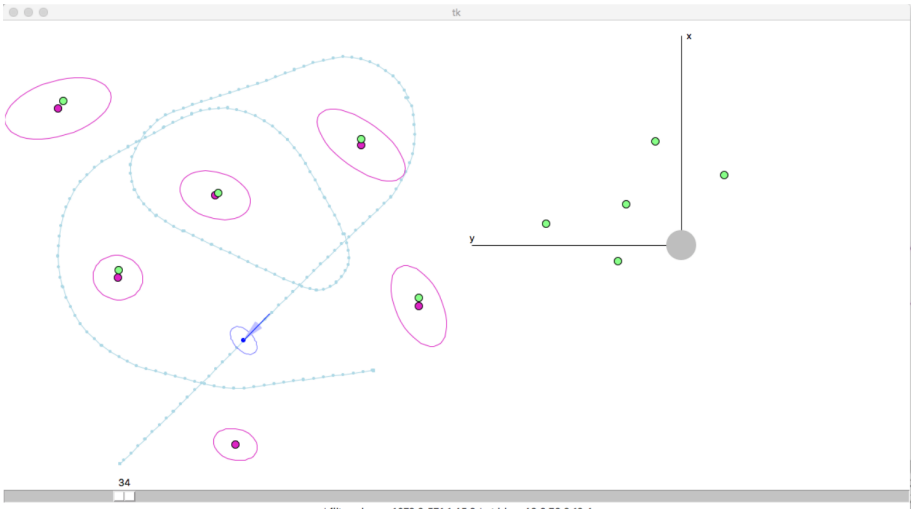


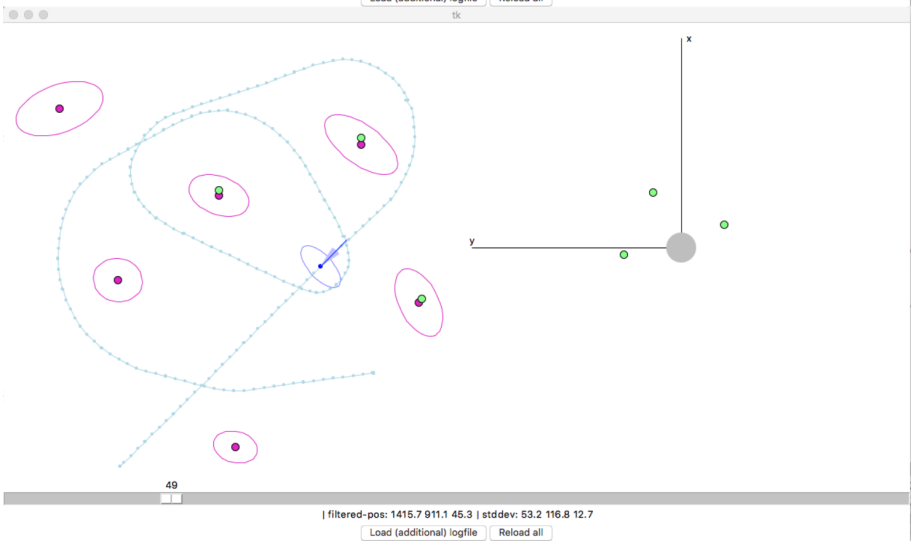
From its initial pose the robot observes 6 landmarks. So the algorithm adds the first landmark to the specific state and also to the covariance matrix. Then the algorithm executes the correction stage using the first landmark added previously. Then the algorithm adds the second landmark to the specific state and also to the covariance matrix. Then it executes the correction stage using two landmarks. Finally when the algorithm has added the six landmarks and has executed 6 corrections stages (one per cylinder observation) the result is the one shown in the previous figure.

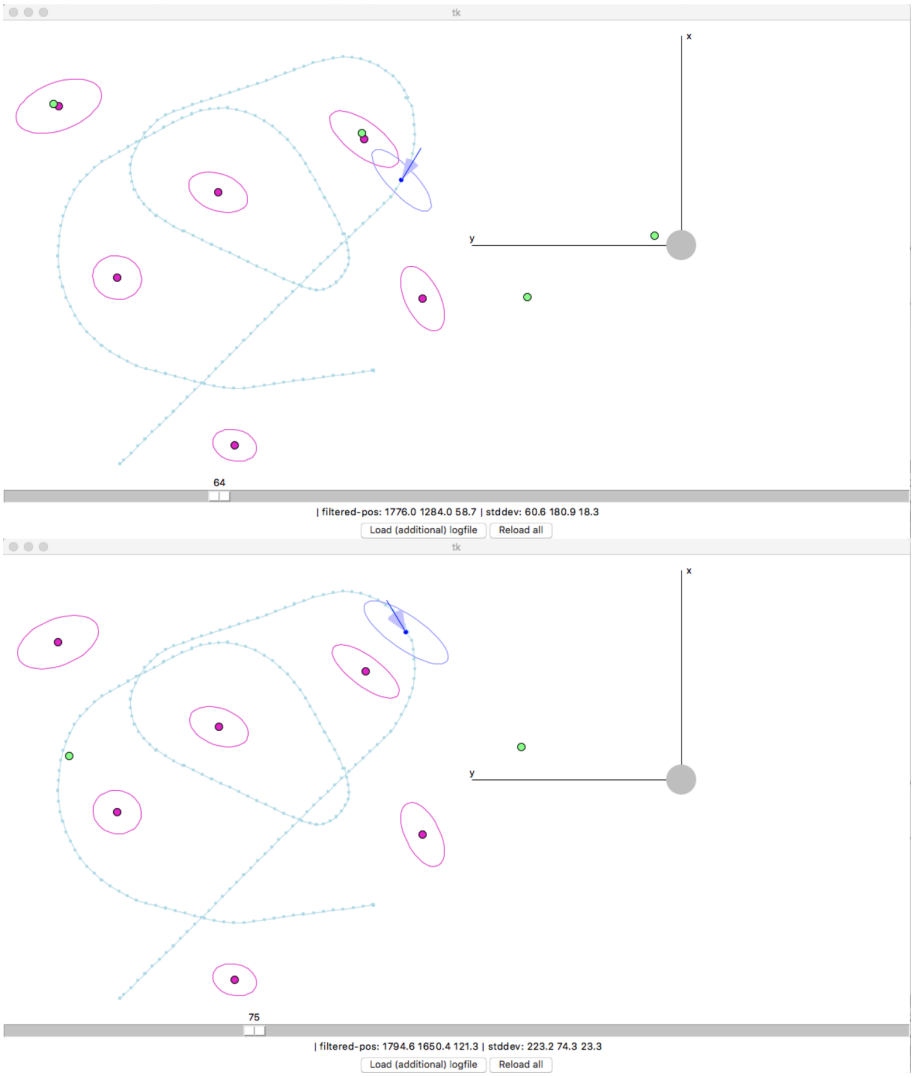
The robot is stopped for a while in its initial pose. So, the prediction stage doesn't modify the algorithm's specific state. From this pose the robot observes 6 cylinders in several scans. Each one of those scans with 6 cylinders implies the execution of the for loop labeled as B with 6 iterations (one iteration per observed cylinder), therefore, 6 correction stages. In this occasion, all the cylinders observed are already present in the algorithm's specific state. So, after the last correction stage, for the last cylinder observed in the last scan taken from the same robot's pose, the algorithm has reduced the landmarks' uncertainty ellipses considerably, as it is show below.



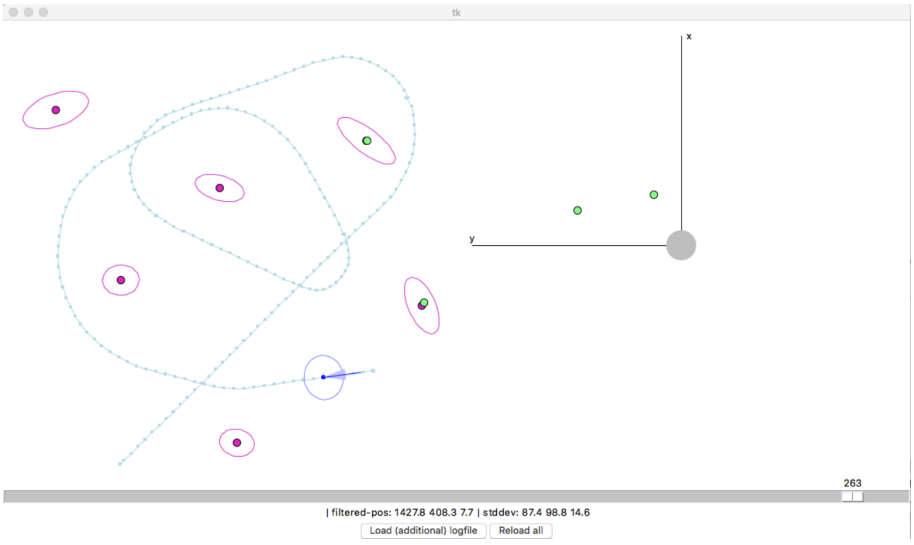
If the robot moves then its pose uncertainty (position uncertainty and heading uncertainty) starts to get larger and larger, and as usual, it is especially large when the robot turns left for the first time because it doesn't observe many landmarks at that point.


tk
x
y
34
| filtered-pos: 1073.2 571.1 45.6 | stddev: 43.6 73.2 12.4
Load (additional) logfile
Reload all

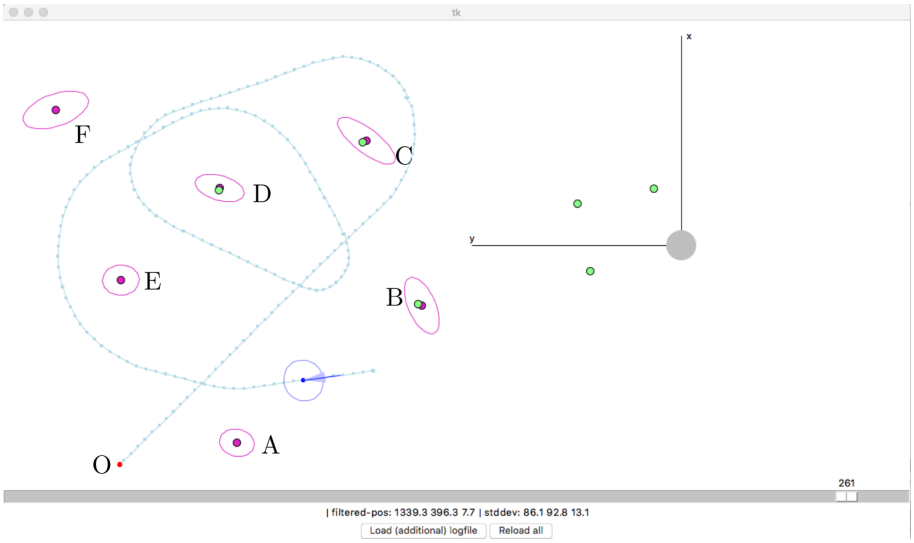

tk
x
y
49
| filtered-pos: 1415.7 911.1 45.3 | stddev: 53.2 116.8 12.7
Load (additional) logfile
Reload all



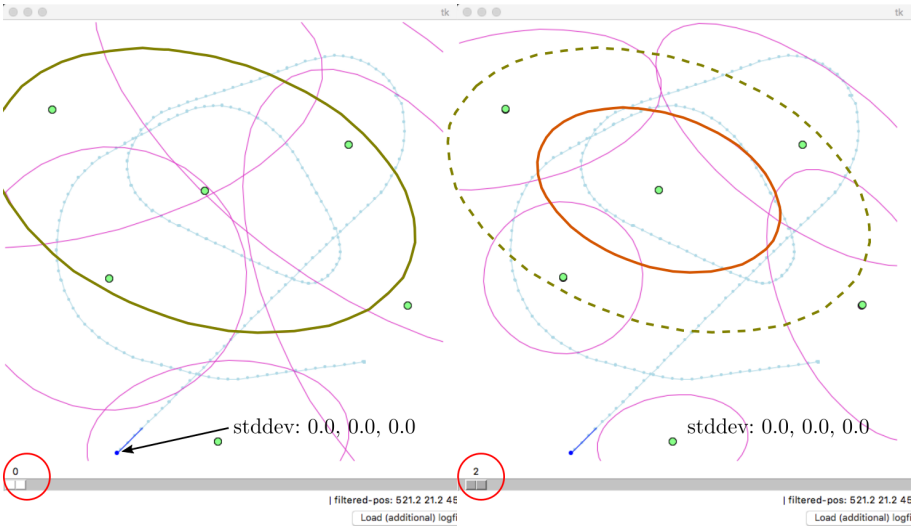
If the robot goes on its uncertainty pose gets smaller again.

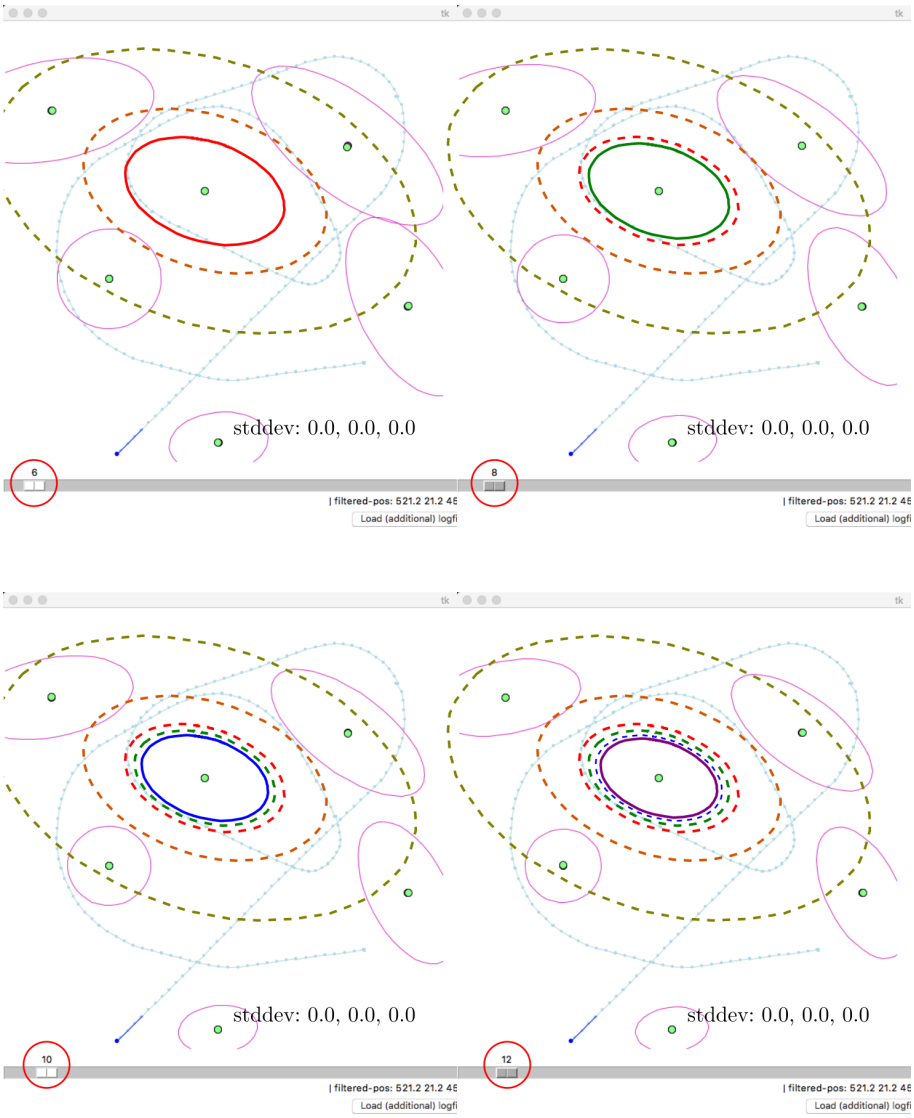


Now let's look at the landmarks' uncertainties. The error ellipse for the landmarks A and E is small but the error ellipse for the landmarks B, C and F is larger. The reason for this is that our coordinate system is actually anchored to the point O. This point has 0 variance in position and orientation and the further away a cylinder is from the robot the first time it is observed the larger the error will be, and even though the robot drives around for a while these uncertainties will stay the same.

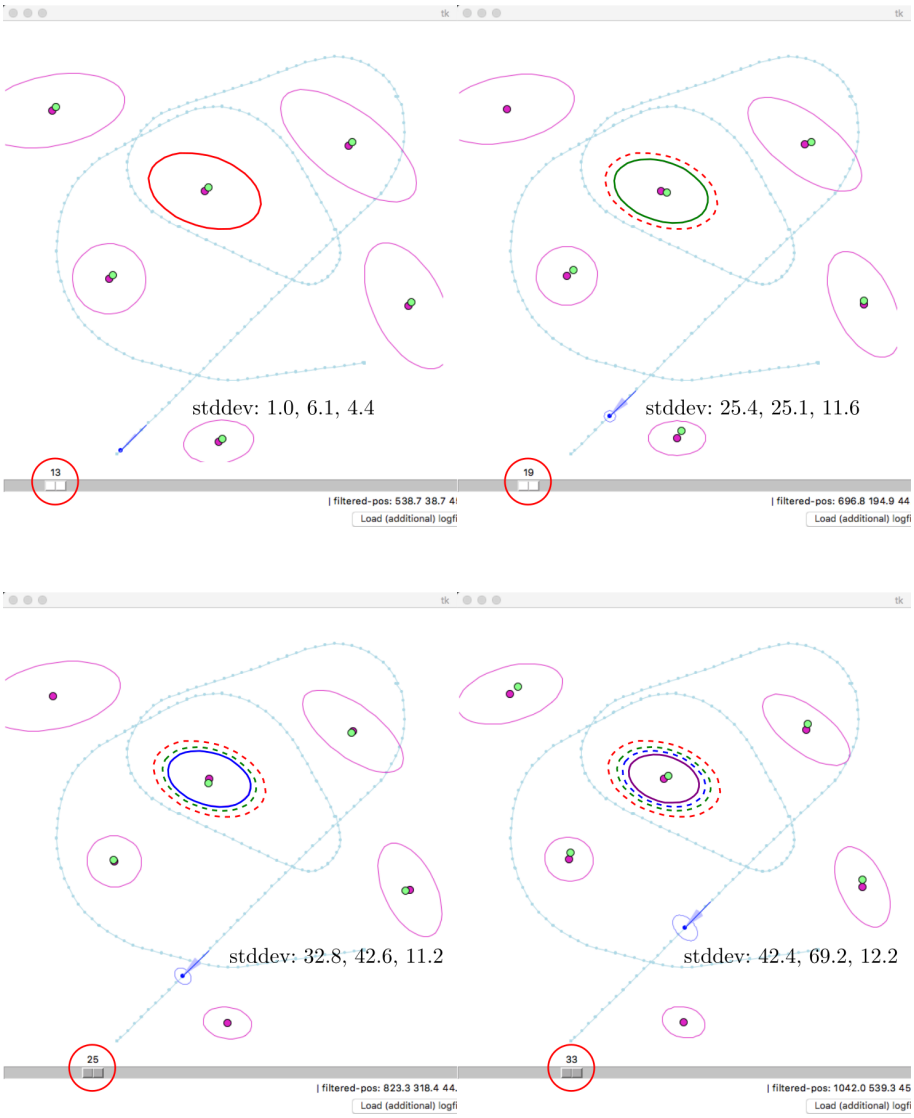


Now, let's look at landmark D. In the beginning, while the robot is stopped in its initial pose, the robot observes the landmark D for a while, so the uncertainty ellipse of this landmark gets smaller, relatively fast.

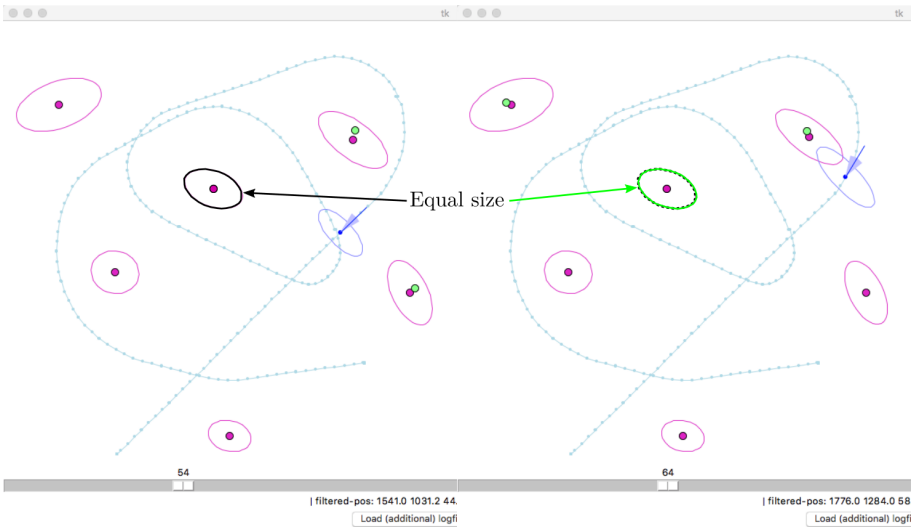




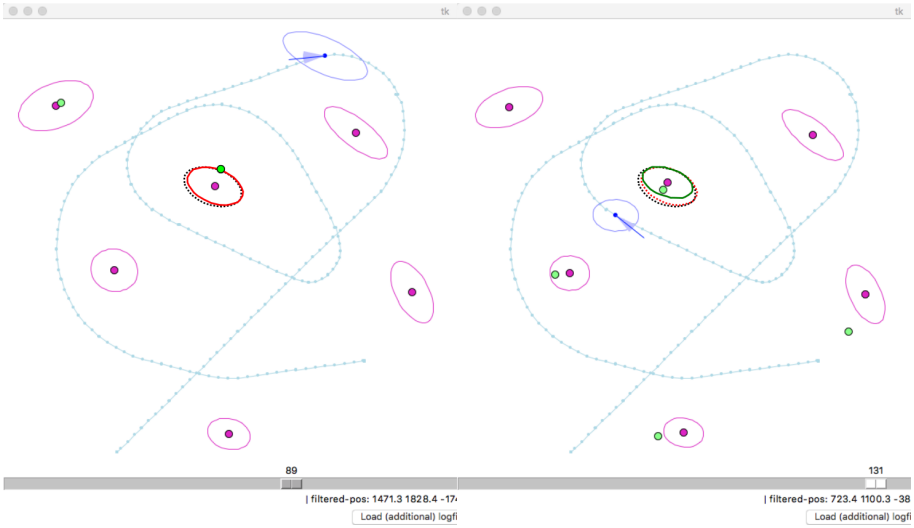
Now, as the robot moves, it still observes the landmark D, but the uncertainty ellipse of that landmark doesn't get smaller as fast as it did previously because now the robot's pose has also accumulated some uncertainty.



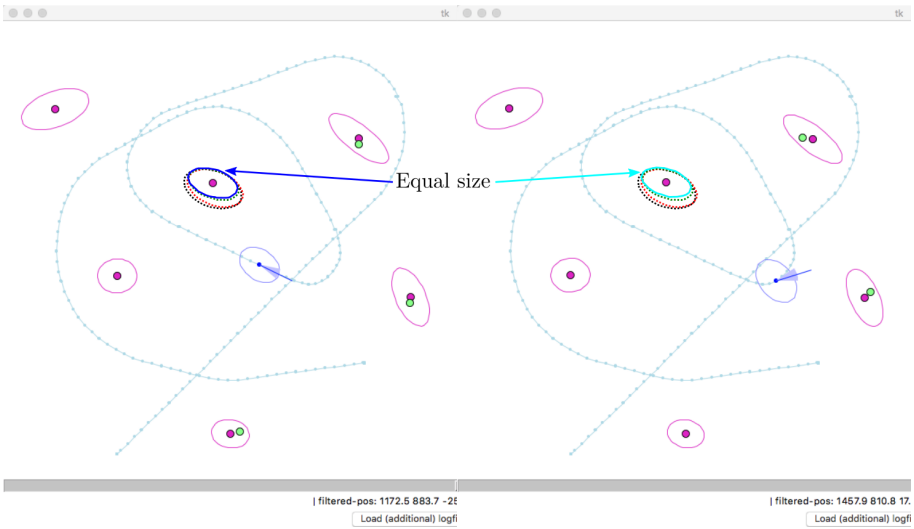
As long as the robot has an observation that is tied to a landmark (the green point close to each landmark) the uncertainty ellipse of that landmark gets smaller. When the robot doesn't have an observation of a landmark anymore the size of the uncertainty ellipse for that landmark stays the same but that landmark's position will move a little bit when the landmarks in the vicinity move. But again, the size of the uncertainty ellipse of the non-observed landmarks will stay the same.



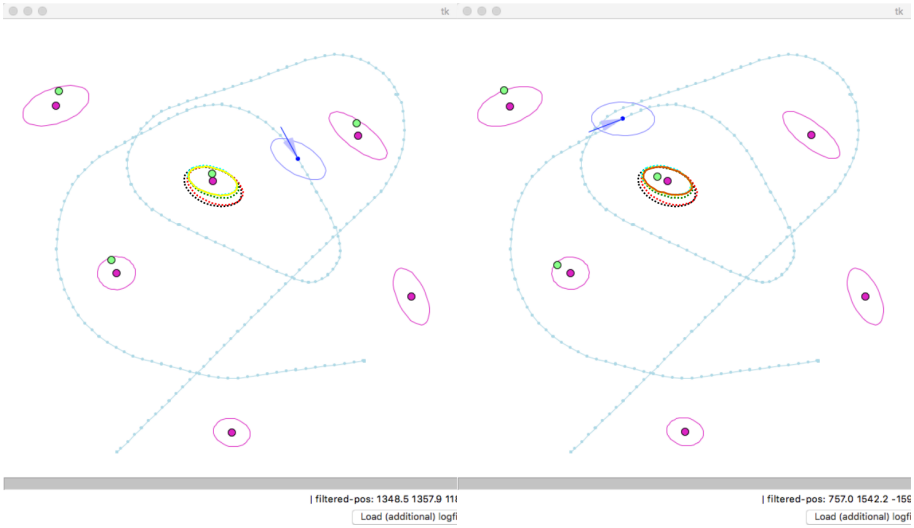
Now, the robot starts observing the landmark D again, and its uncertainty ellipse gets smaller again.



Now, the landmark D is out of the robot's field of view again and the uncertainty ellipse of this landmark stays the same.



Now, the robot starts observing the landmark D again, and its uncertainty ellipse gets smaller again.



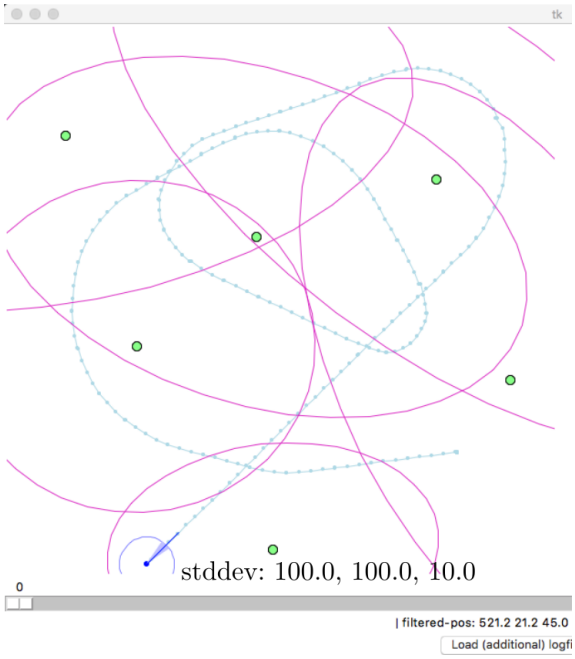
Therefore, basically, whenever the robot has a landmark observation, i.e, whenever there is a green point tied to a landmark in the GUI, the algorithm adds information and when the information increases the uncertainty decreases.

Now let's change the covariance matrix:

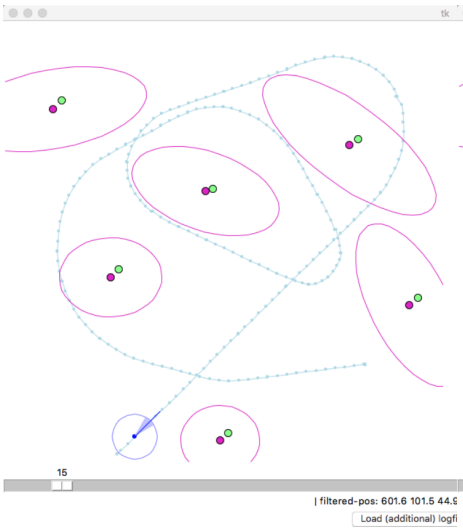
$$\Sigma_t = \begin{pmatrix} \sigma_{x_t}^2 = 0 & \sigma_{x_t y_t} = 0 & \sigma_{x_t \theta_t} = 0 \\ \sigma_{x_t y_t} = 0 & \sigma_{y_t}^2 = 0 & \sigma_{y_t \theta_t} = 0 \\ \sigma_{x_t \theta_t} = 0 & \sigma_{y_t \theta_t} = 0 & \sigma_{\theta_t}^2 = 0 \end{pmatrix}$$

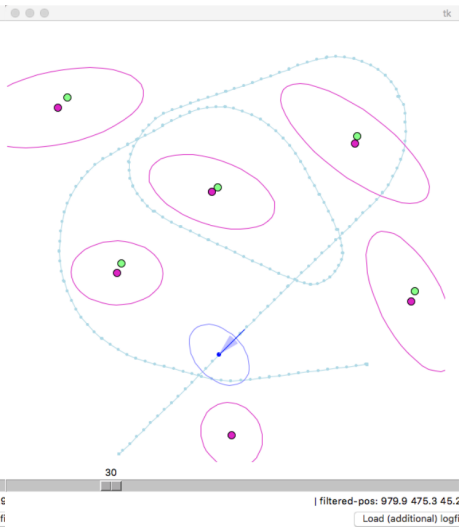
$$\Downarrow$$

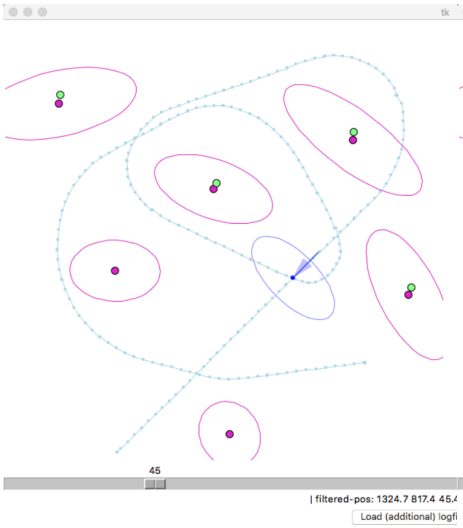
$$\Sigma_t = \begin{pmatrix} \sigma_{x_t}^2 = 10 & \sigma_{x_t y_t} = 0 & \sigma_{x_t \theta_t} = 0 \\ \sigma_{x_t y_t} = 0 & \sigma_{y_t}^2 = 10 & \sigma_{y_t \theta_t} = 0 \\ \sigma_{x_t \theta_t} = 0 & \sigma_{y_t \theta_t} = 0 & \sigma_{\theta_t}^2 = \left(\frac{10}{180}\pi\right)^2 \end{pmatrix}$$

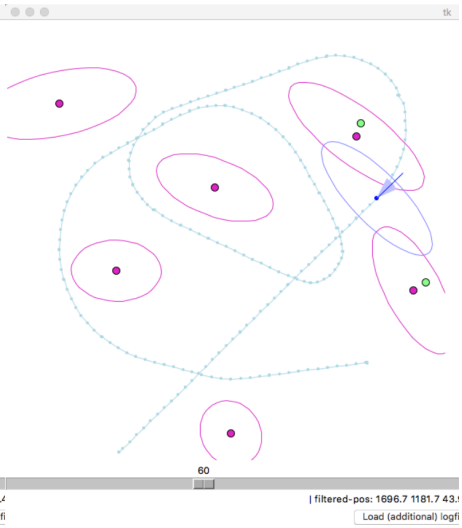


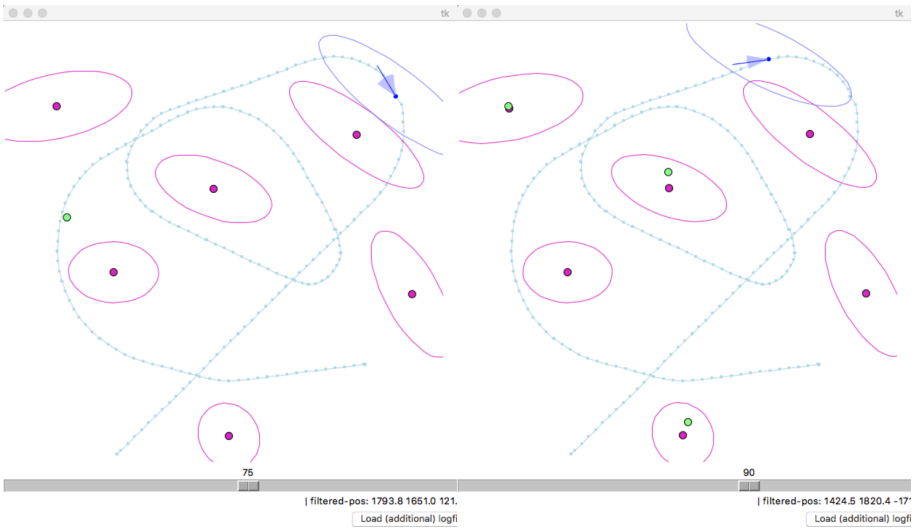
After the robot drives for a while we´ll see the size of the uncertainty ellipse of each landmark doesn't get smaller anymore.

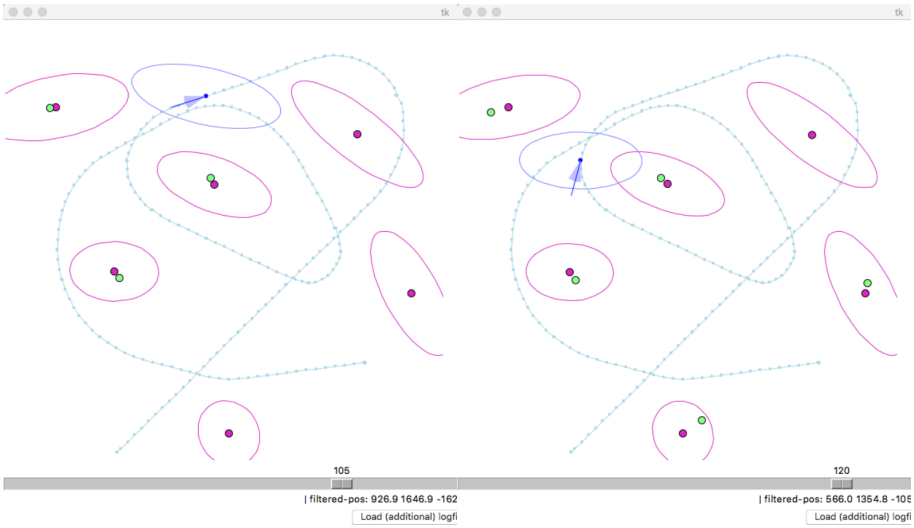

tk
15
| filtered-pos: 601.6 101.5 44.9
Load (additional) logfi


tk
30
| filtered-pos: 979.9 475.3 45.2
Load (additional) logfi


tk
45
| filtered-pos: 1324.7 817.4 45.4
Load (additional) logfi


tk
60
| filtered-pos: 1696.7 1181.7 43.5
Load (additional) logfi



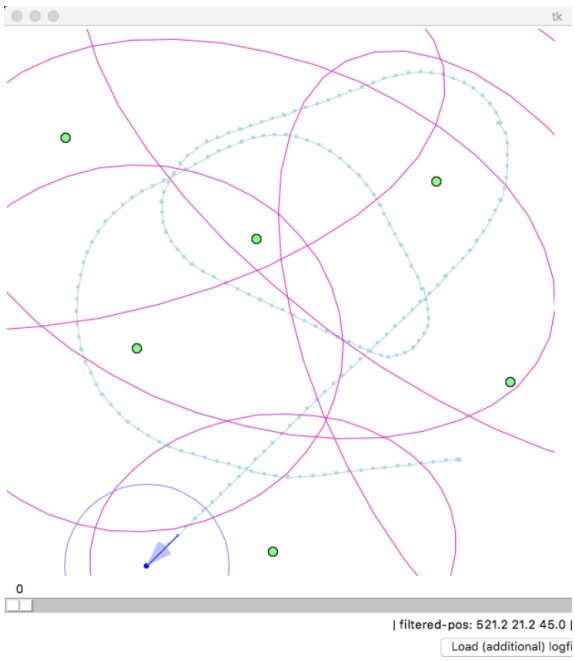


In the end, the size of the landmarks' uncertainty ellipses, the size of the position uncertainty ellipse for the robot and the size of the heading uncertainty sector for the robot are larger than in the previous case. Why is this the case? Simply because the overall uncertainty is tied to the robot's uncertainty in the initial pose, i.e, the pose at the point labeled as O. Therefore, if you have a large variance in the beginning the algorithm never be able to obtain small variances for the landmarks' positions, no matter how many relative measurements the robot obtains. So, the uncertainty ellipses for the landmarks won't get smaller and since the robot uses the landmarks in the correction step of the EKF, the uncertainty for the robot won't get smaller either.

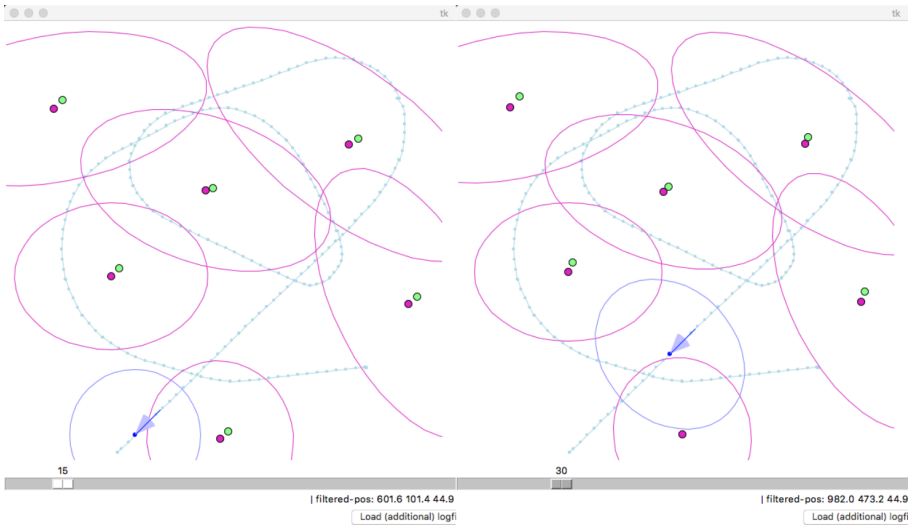
If we make the variances even larger:

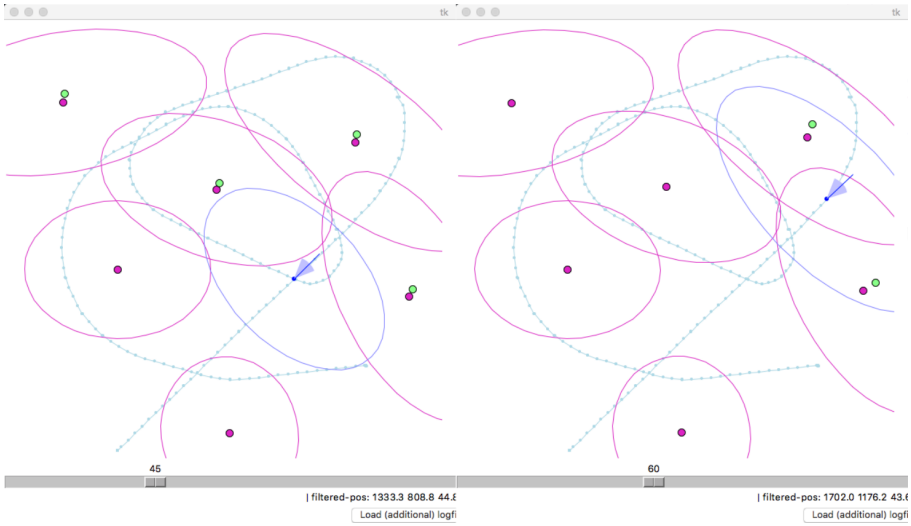
$$\Sigma_t = \begin{pmatrix} \sigma_{x_t}^2 = 30 & \sigma_{x_t y_t} = 0 & \sigma_{x_t \theta_t} = 0 \\ \sigma_{x_t y_t} = 0 & \sigma_{y_t}^2 = 30 & \sigma_{y_t \theta_t} = 0 \\ \sigma_{x_t \theta_t} = 0 & \sigma_{y_t \theta_t} = 0 & \sigma_{\theta_t}^2 = \left(\frac{20}{180}\pi\right)^2 \end{pmatrix}$$

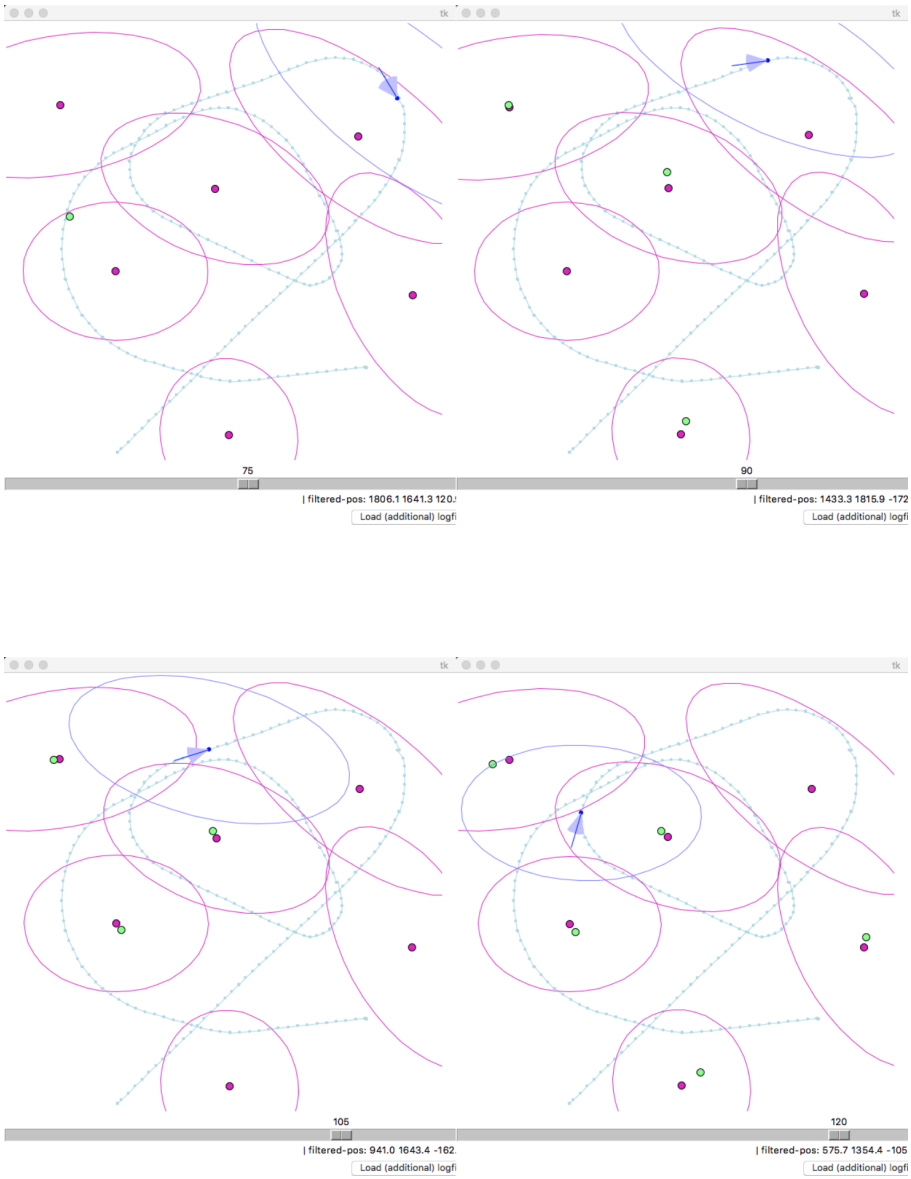
we will easily see the effect.



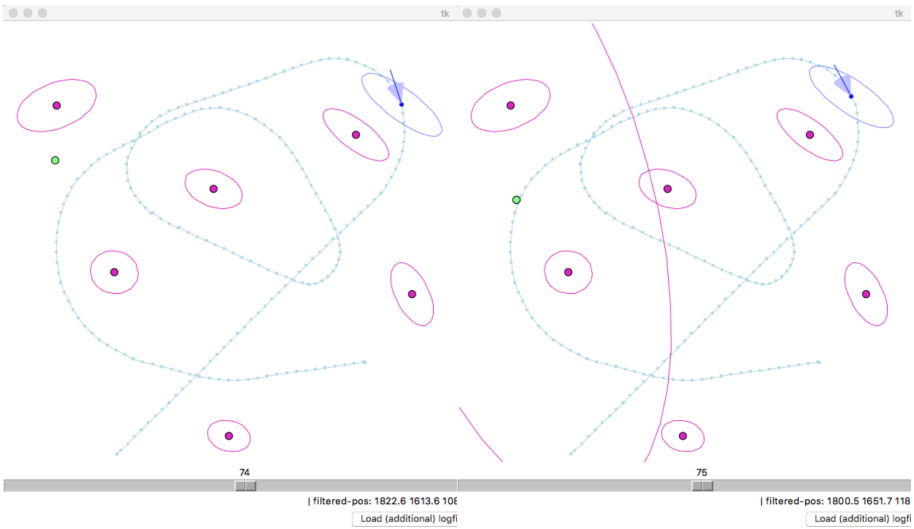
So, the robot starts now with a larger associated uncertainty and as it moves the larger variances in the initial pose translates to a larger variance of the landmarks' positions, of the robot's position and of the robot's heading, no matter how many measurements the algorithm integrates during the trip.


tk
15
| filtered-pos: 601.6 101.4 44.9
Load (additional) logfi
tk
30
| filtered-pos: 982.0 473.2 44.9
Load (additional) logfi

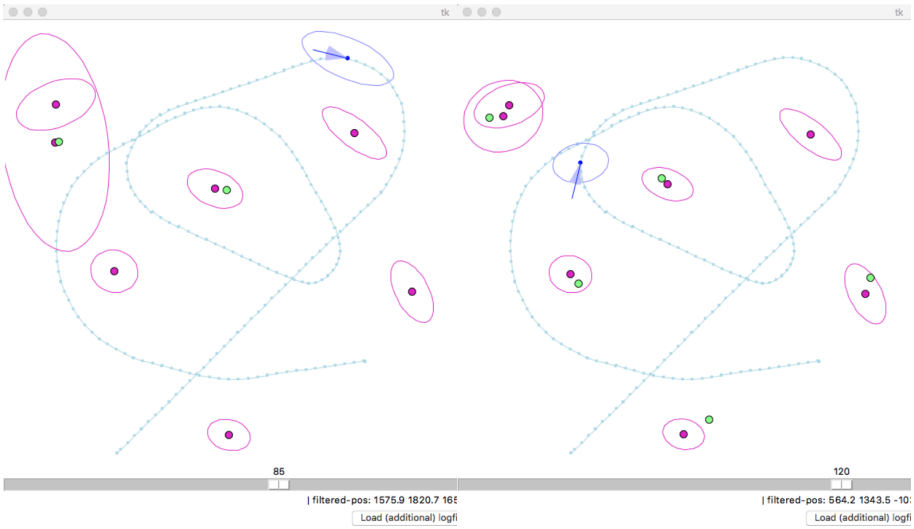

tk
45
| filtered-pos: 1333.3 808.8 44.
Load (additional) logfi
tk
60
| filtered-pos: 1702.0 1176.2 43.
Load (additional) logfi

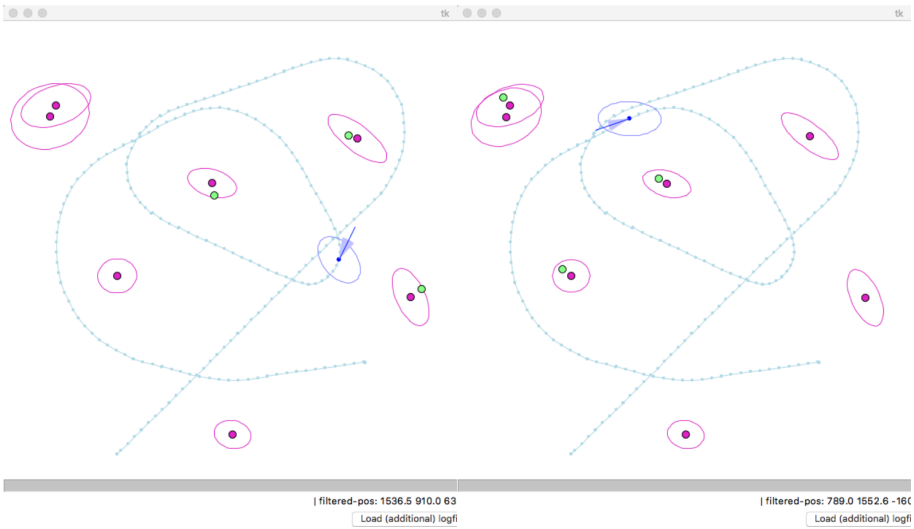


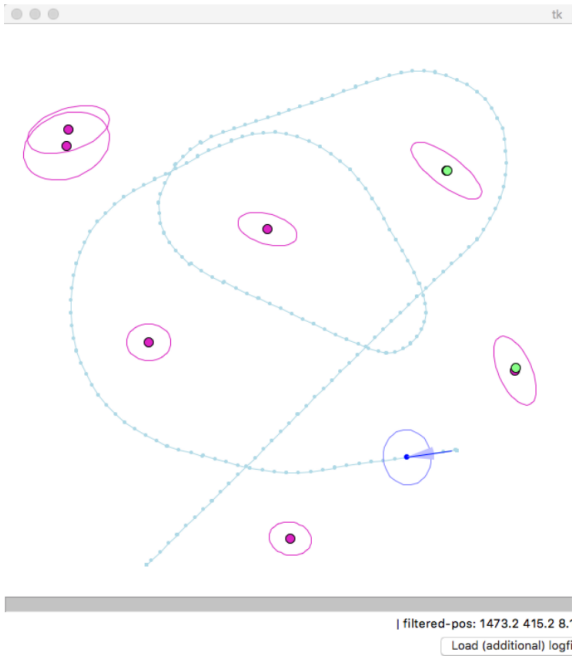
Finally let's have a look at the maximum cylinder distance. This distance is set, by default, to 50 cm. Now let's set this distance to 40 cm. Initially the algorithm gets exactly the same results as previously. But after a while, the robot has the wrong heading and so the cylinder that it observes is considerably far away from the landmark that should be associated to it. Anyway, the cylinder is close enough to the proper landmark according to the 40 cm threshold, so the cylinder is assigned to the correct landmark.
But in the next step the observed cylinder is too far away from any of the landmarks, therefore, the algorithm sets up a new landmark with position variances set to infinity.



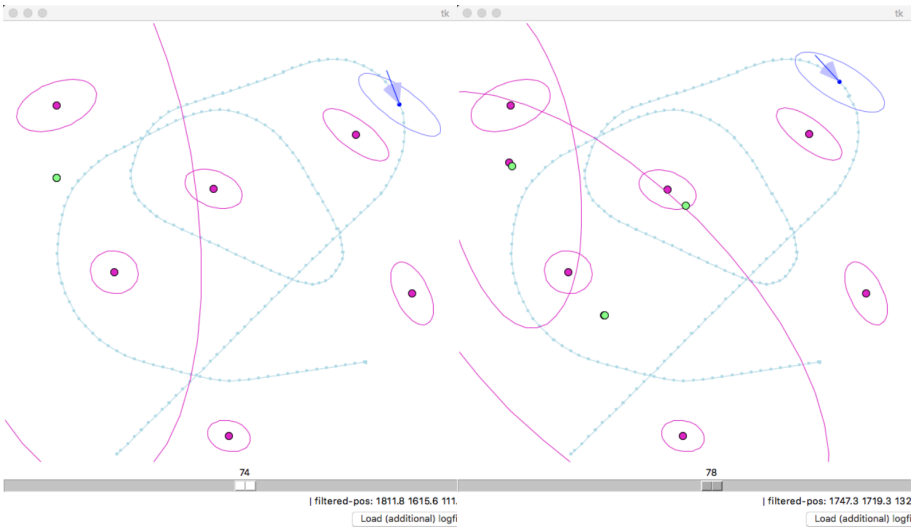
The robot observes more landmarks again, so it corrects the landmarks' positions and what we see is that after a while the newly inserted landmark is moved very close to the old landmark. What we see is that the landmark assignment is pretty simple and it is also brittle.

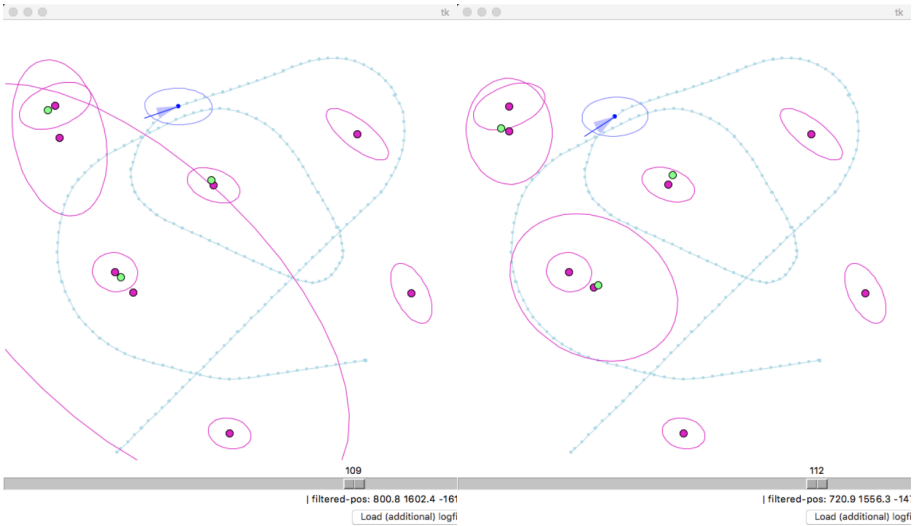


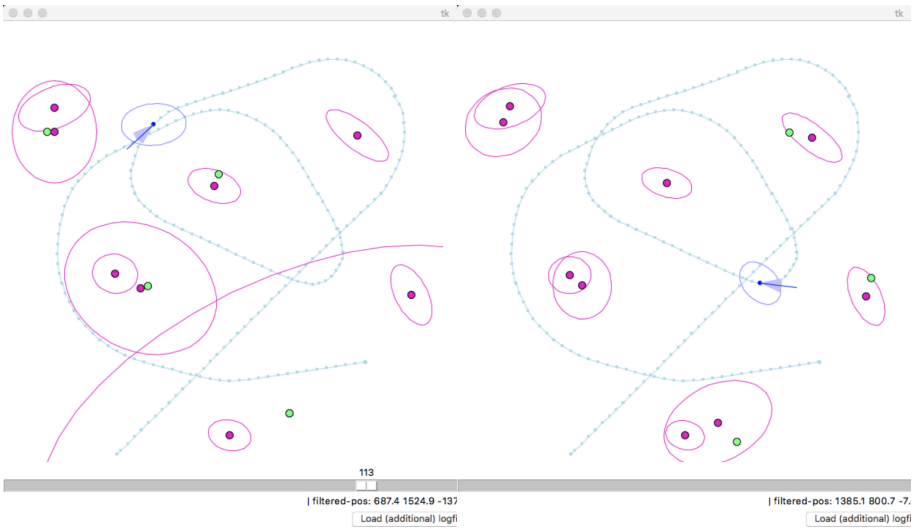


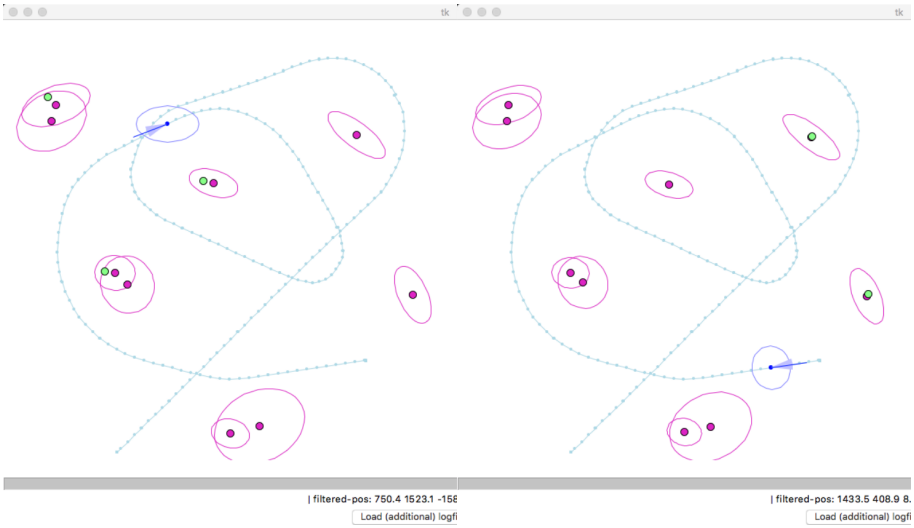


Let's make the maximum cylinder distance even smaller: 30 cm.
The algorithm starts perfectly, as in the previous case. When the robot moves around the first corner the algorithm makes new landmark and a bit more later it creates one more landmark. In the end the algorithm obtains ten landmarks instead of six. This behavior shows us how brittle this landmark assignment process actually is and obviously you could try to improve the algorithm. For example, we could think about unifying landmarks which turn out to be close after observing them for a while but there is other techniques.


tk
74
| filtered-pos: 1811.8 1615.6 111.
Load (additional) logfi
tk
78
| filtered-pos: 1747.3 1719.3 132
Load (additional) logfi


tk
109
| filtered-pos: 800.8 1602.4 -161
Load (additional) logfi
tk
112
| filtered-pos: 720.9 1556.3 -14
Load (additional) logfi


tk
tk
113
| filtered-pos: 687.4 1524.9 -137
Load (additional) logfi
| filtered-pos: 1385.1 800.7 -7.
Load (additional) logfi


tk
tk
| filtered-pos: 750.4 1523.1 -158
Load (additional) logfi
| filtered-pos: 1433.5 408.9 8.
Load (additional) logfi