Lab 2: SLAM for Karel in 2D

Ramón Júlvez, Ubaldo
696497
Cano Andrés, Lorenzo
736078

March 17, 2021

1 Introduction

In this document, we present the work we have developed to achieve the proposed objectives in of the second lab session of the SLAM course. The proposed tasks can be broken down into:

- Implement a SINGLES approach for the data association problem.
- Comparison of the SINGLES and NN approaches, and evaluation of them in different circumstances.
- Improvement of the computational cost of the data association algorithm.
- Implementation of a map maintenance utility.
- Implementation of the JCBB (Joint Compatibility Branch and Bound) data association approach.
- Combination of JCBB with RANSAC.

2 Nearest Neighbor and SINGLES

2.1 Implementation

The nearest neighbor (NN) approach chooses, for a given observation, the feature whose predicted observation is closest to the actual observation. Then, if the Mahalanobis distance between the observation and the selected prediction passes the chi2 test, it is considered a match. This approach has many drawbacks, one of them is the possibility of one feature being assigned to more than one measurement. SINGLES is an approach that avoids that specific issue.

The SINGLES approach avoids pairing a feature with multiple measurements by ensuring that, in each match, the selected feature is only compatible with the selected measurements, and vice-versa. To implement this, we have used the individual compatibility matrix.

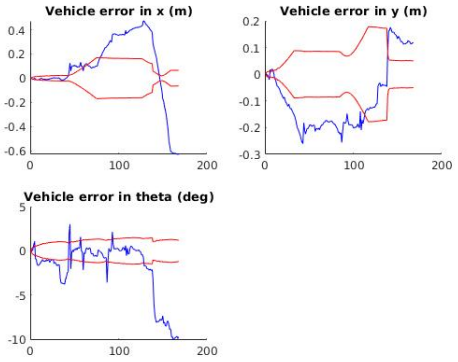
As a refresher, the individual compatibility matrix (ICM) encodes the compatible feature-measurements pairs as ones and the incompatible ones as zeros. Compatible feature-measurement pairs are understood as those whose Mahalanobis distance passes the chi-squared test.

In our implementation, we loop over all observations, and, for the corresponding row in the ICM, we check that there is only one feature that is compatible. Then, we check if that feature

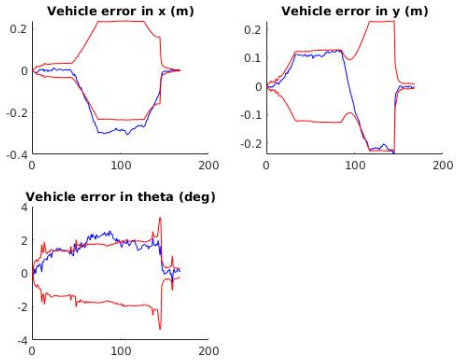
(column in the ICM matrix) has only one compatible measurement. If both requirements are met, the pair is registered as a match, otherwise, the measurement is discarded.

The comparison of both approaches are displayed in fig.1. Specifically, comparing fig.1c and fig.1d, the number of false positives of the SINGLES approach is zero for the whole run. For NN, this number increases as the error increases, which leads to further degeneration of the estimation of the pose of the robot (fig.1a.

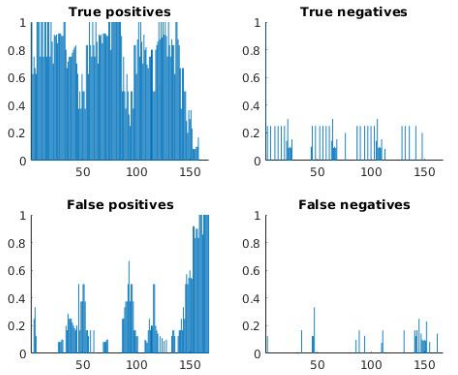
This concrete example makes SINGLES be far superior to NN. However, this approach is far from perfect. For example, one can easily imagine a more cluttered scenario, where all measurements could match more than one possible prediction. In this case, the SINGLES approach would result in no features being added to the map, and so, no SLAM.



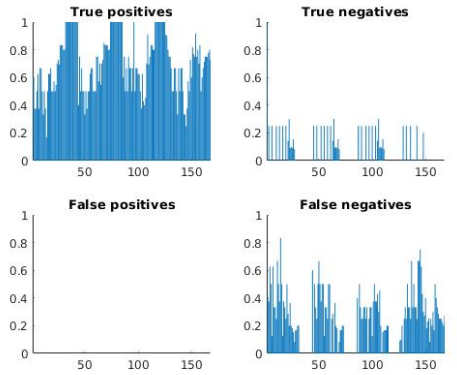
(a) Error with NN approach.



(b) Error with SINGLES approach.



(c) Error count with NN approach.



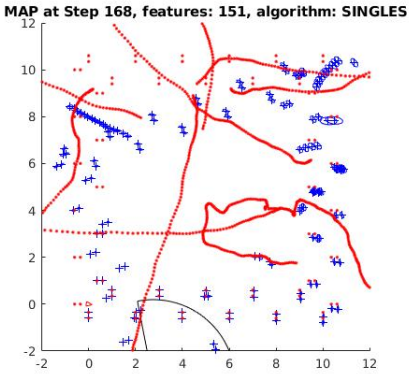
(d) Error count with SINGLES approach.

Figure 1: Comparison of NN and SINGLES approaches. All runs lack any map maintenance and moving elements.

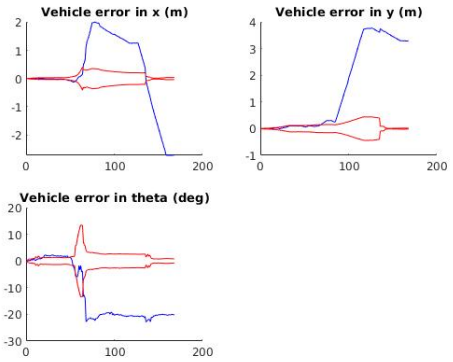
2.2 Proposed Tests

The first test is to run the SINGLES algorithm with dynamic elements on the environment (fig2. These dynamic elements are especially troublesome for this approach. This is because, as these dynamic elements are usually quite separated from other elements, the SINGLES test is passed easily when they are first encountered. When dynamic elements are taken as a

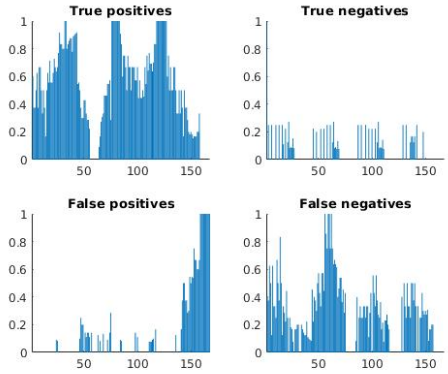
match, they deviate the estimations of the static features, and as static features are close to each other, they start to fail the SINGLES test. This means that while dynamic obstacles are in the sensor's range the estimation breaks completely. After they leave the sensor's range, the system works again, but the accumulated error remains.



(a) Result.



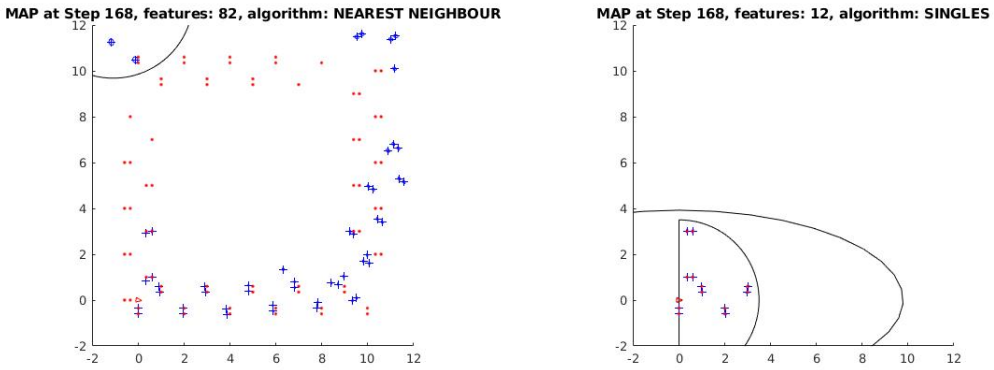
(b) Error.



(c) Error count.

Figure 2: SINGLES method with dynamic obstacles

The second test is to compare how NN and SINGLES perform without any odometry (fig.3). Both methods break completely, but in very different ways. For the NN method, the problem is as expected. The lack of odometry makes the predictions very bad, and so, the error is accumulated constantly but with no real updating of the uncertainty. For the SINGLES method, the uncertainty of the pose of the robot increases continually, since from the start no features are correctly associated, because of the strict conditions of association of the SINGLES algorithm.



(a) NN. (b) SINGLES.

Figure 3: NN and SINGLES with no odometry

3 Improving the computational Cost

Originally, we intended to implement some kind of map tessellation, so that we could access features directly by giving some coordinates and a radius. However, this turned out to be quite complicated, so a different approach was taken.

Before any optimization, the algorithm computes the distances and compatibility of all prediction-measurement pairs. Our optimization basically consists of only calculate the distance and individual compatibility matrices for the features that are within a certain range of the robot (the sensor range). This way, we avoid computations related to features that are on the other side of the map. This way, we optimize both NN and SINGLES in one action.

There is also a minor optimization we have implemented in the NN algorithm. Instead of double-looping over the distance matrix, some MATLAB indexing features have been using so that it is faster. The computational complexity, however, stays the same.

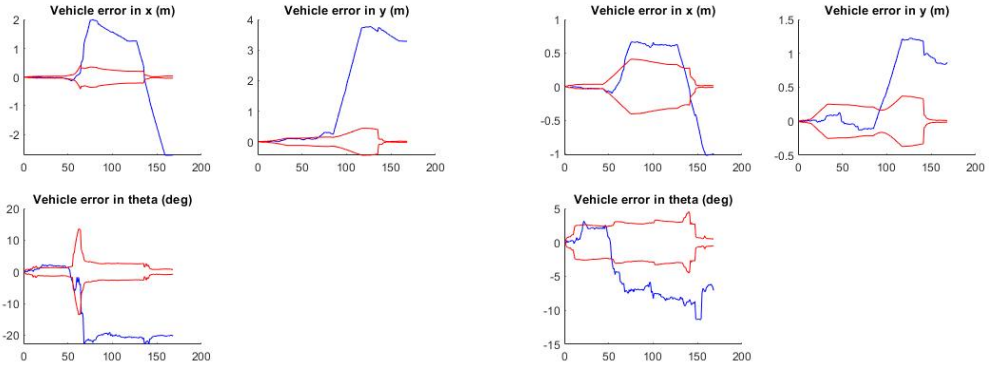
4 Map maintenance

The next concept we will take a look at is map maintenance. This is the act of being selective with the features we actually store in the map, with the hopes of removing those which could be mistaken or not useful.

The conditions for detecting a feature as unreliable and thus removing it from the map we implement are actually rather simple, we will remove features that have only been observed once, and that was first observed at least two steps ago. The reasoning behind this is that we consider that for a feature to be reliable, it has to be identifiable from a reasonable range of positions, and with some consistency. If a is seen on a certain step, but not the several next, it is very unlikely we will be able to re-observe this feature in the future. This could happen for many reasons, the feature could be a moving object and simply not be there anymore, it could be due to a large error on the measurement, or depending on the data association algorithm some failure situation could occur where we incorrectly associate measurements of that feature.

This condition has one exception, and that is for features seen in the first steps. This is the case because we can't be sure of the previous assumptions since it could happen that a feature

is only detected once because it started on the range of our sensor range and therefore shouldn't be erased. To control this, we simply decide to not remove any features that were first seen before the third step. This is also especially important for loop closing since the features from the first steps are the ones with less uncertainty and therefore incorrectly removing one could greatly affect the final result of the map after closing a loop.



(a) Error without map maintenance.

(b) Error with map maintenance.t

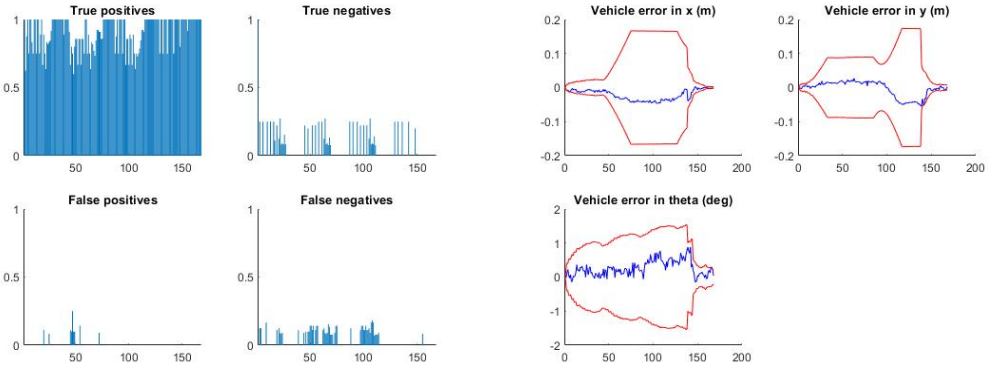
Figure 4: Robot position and orientation errors for a map with moving persons, using the SINGLES data association algorithm.

In figure 4 we can observe the errors in the position and orientation of the robot for two experiments run on a map with moving persons, with the SINGLES algorithm. See how the error is around half for the version with map maintenance on the right. Another advantage of performing map maintenance is that the cost will be reduced since we reduce the number of features on which we know many of the SLAM's algorithm cost depends. for this same example, the number of features of the final map without map maintenance was 150, while with map maintenance was 110, which is around a 30% reduction and definitely significant.

5 JCBB

As an additional data association algorithm, we implement Joint Compatibility Branch and Bound (JCBB), as described in [1]. The basis for this algorithm is taking into account the joint compatibility of all the feature associations, and perform exploration on the tree of possible data association hypothesis, selecting the one with a higher number of total pairings that are jointly compatible.

Overall this algorithm should provide more robust results than NN or SINGLES, since checking the joint compatibility should help with the false positives NN or false negatives SINGLES produces. Also, though the algorithm is definitely more complex than any of the former and would require more computation, each measurement should still have a constant number of feature candidates and thus the overall cost will remain constant.



(a) Classification results of the data association for JCBB.

(b) Robot position and orientation errors for JCBB.

Figure 5: Results of an experiment run with moving objects (persons), JCBB data association algorithm and map maintenance.

On figure 5 the results of an experiment for a map with moving persons can be seen. The classification results are very good, and feature almost no false positives, with only a few false negatives. This is reflected in the obtained robot position and orientation errors, which are the smallest so far. On figure 6 the resulting map can be seen, which looks very precise.

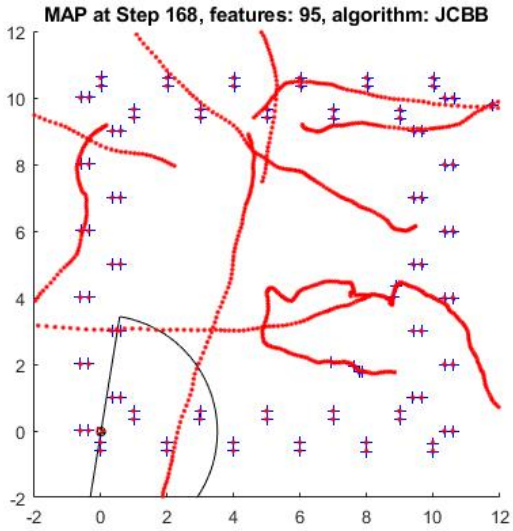


Figure 6: Resulting map of an experiment run with moving objects (persons), JCBB data association algorithm and map maintenance.

6 RJC

Though as we discussed previously, the computational complexity for JCBB should remain constant because of a constant number of features, if this number is very large it could potentially be too slow, if, for example, a moving object generates many close-by features. For this,

we implement the Randomize Joint compatibility, again adapted from [1], with the hopes of obtaining a more efficient algorithm with comparable results to JCBB.

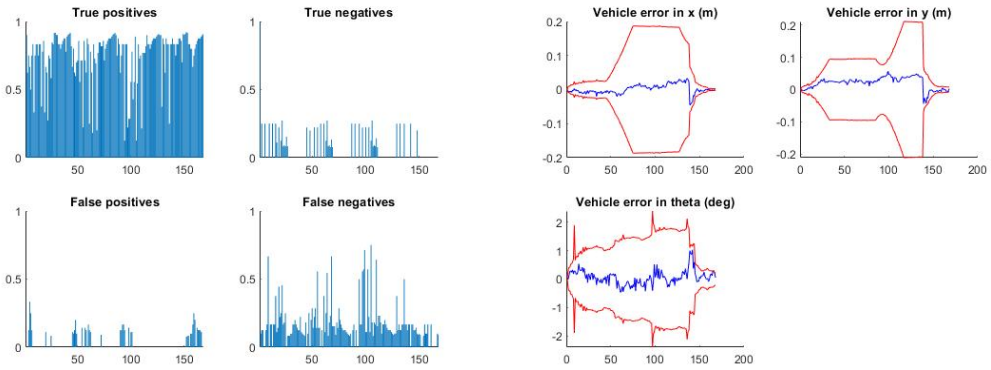
The premise of this algorithm is to combine the ideas behind RANSAC, the generation of multiple random hypotheses which are voted by the data so as to select the best, and the previous JCBB algorithm. In practice, this means that for each iteration a hypothesis of b measurements is randomly selected, which will return a result of n jointly compatible associations. This is ran a number k of times, after which the result with a higher number of associations is chosen. k is calculated just as in RANSAC from the inlier percentage w and the probability of the algorithm producing a correct result for any given iteration, p, just as follows:

$$k = \frac{\log(1-p)}{\log(1-w^b)} \tag{1}$$

As for selecting the associations from a hypothesis, JCBB* is used, which is a modification of JCBB where there is no star node, meaning that the first b measurements are always expected to be fit into a jointly compatible hypothesis. Then, the rest of the associations are done with a modified NN algorithm which also checks for the joint compatibility with the hypothesis.

The values chosen were $b = 4$, as proposed, though a hypothesis from only 2 measurements could be constructed. The rest of the parameters were modified to fit our purposes, and we choose initial values for $w = 0.5$ and $p = 0.1$. This means the algorithm will execute at most 35 iterations, but this number rapidly decreases if a good hypothesis is found, down to 2 if the estimated inlier percentage $w \approx 0.9$.

In figure 7 the results of an experiment with moving persons, just as for JCBB, can be observed. Though the number of false negatives has certainly increased, the false positives are still kept very low and the error and uncertainty for the position and orientations of the robot are almost identical to the previously obtained with JCBB. In figure 8 the final obtained map can be seen, which again looks very precise.



(a) Classification results of the data association for RJC algorithm.

(b) Robot position and orientation errors for RJC algorithm.

Figure 7: Results of an experiment run with moving objects (persons), RJC data association algorithm and map maintenance.

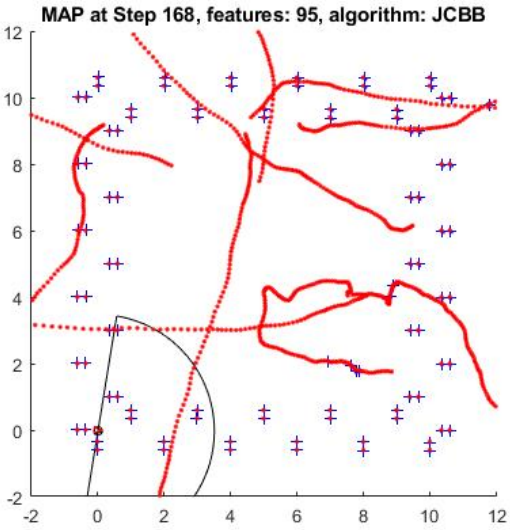


Figure 8: Resulting map of an experiment run with moving objects (persons), RJC data association algorithm and map maintenance.

References

[1] L. M. Paz, J. D. Tardós, and J. Neira. "Divide and Conquer: EKF SLAM in $O(n)$". In: *IEEE Transactions on Robotics* 24.5 (2008), pp. 1107–1120. DOI: 10.1109/TRO.2008.2004639.