Loop-closure candidates selection by exploiting structure in vehicle trajectory

Juan I. Nieto, Gabriel Agamennoni and Teresa Vidal-Calleja
Australian Centre for Field Robotics
The University of Sydney
j.nieto@acfr.usyd.edu.au

***Abstract*— One of the most important problems in robot localisation is the detection of previously visited places (loops). When a robot closes a loop, the association between observed features and present ones can be used to update its position. The computational cost involved in the association process makes exhaustive loop search intractable. Most of the current techniques use observations of the environment as their main features to produce loop hypotheses. In this paper, we investigate the feasibility of producing loop candidates from features of the robot trajectory. We propose a new method for selecting loop-closure candidates based on an alignment likelihood function, which measures similarity between trajectory sequences. The algorithm is validated with data gathered in the city with our experimental platform. Positive results show that the trajectory has, indeed, features that can be extracted and applied to robot localisation. The resulting loop hypotheses may be regarded, for example, as a initialisation step to aid current methods.**

I. Introduction

To navigate autonomously, a mobile robot needs to know where it is within an environment. One of the main problems in robot localisation is the detection of previously visited places, commonly known as the loop-closure detection problem. Traditional approaches to find loop-closure candidates were based only on innovation gate distance metrics between individual poses [1]. More recent approaches using range information also incorporate landmark shape descriptors [2], [3], [4], [5]. The main disadvantage of range-based methods is the poor environment information obtained with 2D sensors. This problem can be eliminated by using 3D range sensors, which render a better characterisation of the robot's surrounding. The main two limitations of 3D lasers are (a) the cost, they are still relatively expensive, and (b) the extra complexity added to the algorithm in order to devise how to handle the greatly increased amount of data [5].

Cameras have also been used to generate loop-closure hypotheses [6]. Adding visual descriptors facilitates the detection of previously visited places by capturing more information of the environment [7], [8], [9]. This comes at the price of an increasing computational complexity when different observations need to be matched. An exhaustive search to compare the current view with the stored ones is not feasible due to the high computational load involved. Consequently, regardless of the sensor and strategy used, generating a reduced number of loop-closure candidates is critical for any real-time implementation. One of the latest common alternatives to exhaustive search is building a dictionary of the environment in the form of a vocabulary tree [10]. Then, every time a new observation is acquired, the algorithm searches the stored sensed data for the most feasible candidates by comparing their words.

This paper presents a new method for producing best candidates for loop-closure by making use of the structure in vehicle trajectory. We exploit the fact that in most of outdoor environments vehicles are limited to drive on constrained paths, and hence, the trajectory possesses geometric features that can be used to generate loop-hypotheses. Note that the method is not meant to be used as a stand-alone data association technique due to the reduced amount of information being used. The approach presented here should be used to complement methods using environment features for loop detection. For example, our framework can generate hypotheses in a SLAM implementation, and images from the hypotheses can be inspected to confirm the loops.

Fig. 1 shows an example to illustrate the main idea. The figure shows data extracted from our experiments where the vehicle revisits an area after accumulating a large uncertainty. Four positions in the second loop are shown with their respective candidates selected. Note that, for positions 3 and 4, the algorithm does not simply select the nearest neighbours, but rather it chooses the ones that are the nearest in terms of distance and shape (uncertainty ellipses are not included to maintain clarity in the example).

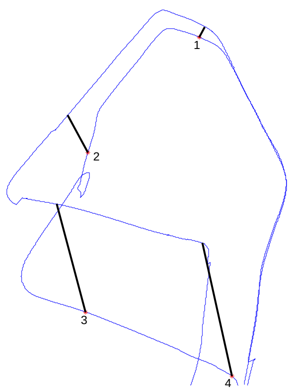

Fig. 1. Example of loop candidate selection. The blue line represents the vehicle trajectory. The vehicle's estimated position for the second time the area is visited is the larger loop. The method presented in this paper selects loop candidates for positions 1,2,3,4. Only the candidate with the highest likelihood is shown for each position (solid line).

II. Related Work

As already mentioned, most of the current approaches for generating loop-closure hypotheses, make exclusive use of observations from the environment, ignoring information associated with the robots trajectory history. For example, the work presented in [11] builds a vocabulary tree of visual descriptors to suggest candidate views. A similar approach is proposed in [7] where the authors use a bag-of-words model to build the vocabulary tree. An approach that incorporates information from vehicle position is presented in [12]. The authors evaluate the likelihood that images acquired at each pose overlap by using the joint distribution between the current and candidate views. The main difference between this work and ours, is that we use the history of vehicle poses instead of the current one only. The authors in [13] present a system that uses both appearance and relative position of local visual features to calculate the probability of a loop-closure.

The aforementioned works are based on images. The latest approaches using range data are based on map matching techniques to generate loop candidates. The work presented in [2] extracts geometric features from sensor scans, which are fed to train a binary classifier. The classifier learns a mapping from features to loop-closure events, which is then used to decide whether there is a loop or not. The main disadvantage of learning approaches is that, in general, a mapping function must be previously learnt and may not generalise well for different environments. A cost function similar to the one proposed is used in [4] to compare shape between laser scans; however, unlike the similarity function presented here, the work in [4] does not account for uncertainty.

The key difference between our work and previously presented ones, is that, instead of using information about the environment to generate loop candidates, we explore here the practicability of generating loop-closure candidates by exploiting the shape of the vehicle's trajectory. Our work can be complementary to the ones mentioned above. It is important to note that the algorithm will be useful in environments where the vehicle motion is constrained to certain paths (e.g. roads). Nevertheless, we argue that most of the environments where vehicles navigate are of these kind. We believe that the main advantages of our approach are its simplicity and complementary use with previously presented solutions.

III. Probabilistic matching of sequential data

A. Likelihood function

The foundations of probabilistic sequence matching is data association. We need to determine which of all the past poses is most similar to the current one. The measure of similarity is defined in terms of the poses, which we assume are available in the form of probability distributions. Specifically, we define the log-similarity of poses $\mathbf{x}_i$ and $\mathbf{x}_j$ as

$$\lambda\left(i,j\right) = \ln P\left(\mathbf{x}_i = \mathbf{x}_j\right). \tag{1}$$

The poses will typically be estimated by an extended Kalman or unscented filter; hence they will be jointly Gaussian-distributed. In this case, the log-similarity takes the following mathematical form

$$\lambda\left(i,j\right) = -\frac{1}{2}\ln\left|\mathbf{\Gamma}_{ij}\right| - \left(\mu_i - \mu_j\right)^T \mathbf{\Gamma}_{ij}^{-1}\left(\mu_i - \mu_j\right), \tag{2}$$

where μ_i and μ_j are the estimated means of poses i and j, respectively. The matrix $\mathbf{\Gamma}_{ij}$ is equal to

$$\mathbf{\Gamma}_{ij} = \mathbf{\Sigma}_{ii} + \mathbf{\Sigma}_{jj} - \mathbf{\Sigma}_{ij} - \mathbf{\Sigma}_{ji}, \tag{3}$$

where $\mathbf{\Sigma}_{ii}$, $\mathbf{\Sigma}_{jj}$ are the estimated covariances of i, j and $\mathbf{\Sigma}_{ij}$, $\mathbf{\Sigma}_{ji}$ are their cross-correlation matrices.

The value $\lambda(i,j)$ is a measure of how similar pose i is to pose j. This stems from the formal definition of λ in (1). If the ith pose is close to the jth pose, then $\lambda(i,j)$ will be large; conversely, if poses i and j are far apart, then $\lambda(i,j)$ will be small. Note that $\mathbf{\Gamma}_{ij} = \mathbf{\Gamma}_{ji}$ and thus $\lambda(i,j) = \lambda(j,i)$, i.e. the likelihood function is symmetric.

B. Sequential data

Applying the likelihood function to individual poses does not exploit temporal correlations. In structured environments, the trajectory travelled by a mobile platform tends to exhibit regular patterns. These patters in the trajectory can be used to disambiguate between multiple similar poses when there are several loop-closure candidates to choose from. In order to consider sequences of poses, the sequential ordering must be encoded, for example, by means of a composite likelihood function.

Suppose we are given an undirected graph $G = (V, E)$ of n poses. The vertex set $V = \{1, \ldots, n\}$ indexes all poses stored so far in the sub-sampled metric map; the edge set contains links between consecutive poses and loop-closure links. Let p and q be paths of length k on the graph G; i.e. suppose $p = \{p_1, \ldots, p_k\}$ such that $(p_i, p_{i+1}) \in E$ for all $i = 1, \ldots, k-1$, and similarly for q. We define the *alignment* likelihood of paths p and q as follows,

$$\lambda\left(p, q\right) = \sum_{i=1}^{k} \lambda\left(p_i, q_i\right). \tag{4}$$

The value $\lambda(p, q)$ is a measure of how well the two paths align with each other.

Now we can impose sequential coherence as follows. First, we set p to be last k most recent poses, i.e. $p = \{n-k+1, \ldots, n\}$. Then, q is restricted to the rest of the graph, i.e. q is a path in $\bar{G} = (\bar{V}, \bar{E})$, where

$$\begin{aligned}\bar{V} &= V - \{n-k+1, \ldots n\} = \{1, \ldots, n-k\}, \\ \bar{E} &= E \bigcap \left\{\bar{V} \times \bar{V}\right\} = \{(i,j) \in E : 1 \le i, j \le n-k\}.\end{aligned} \tag{5}$$

The path p is aligned to the graph by searching for the path q in $\bar{G}$ that maximizes $\lambda(p, q)$, that is

$$q^{\star} = \arg\max_{q \in \bar{G}} \lambda\left(\{n-k+1, \ldots, n\}, q\right). \tag{6}$$

Figure 2 illustrates the sequence alignment process. At sample n the robot revisits the same curve that was traversed previously between i and j. This causes λ to be maximum for the path $q = \{i, \ldots, j\}$, causing poses n and j to be matched to one another.

C. Sequence alignment

While the task of searching for the optimal sequence alignment can be simple in principle, a naïve implementation of Eq. (6) can be prohibitively expensive when the current pose uncertainty is large and the graph is highly populated. Fortunately, the search in (6) has optimal substructure, and hence can be implemented efficiently via dynamic programming. We present an efficient implementation based on the Viterbi algorithm, which has linear computational cost.

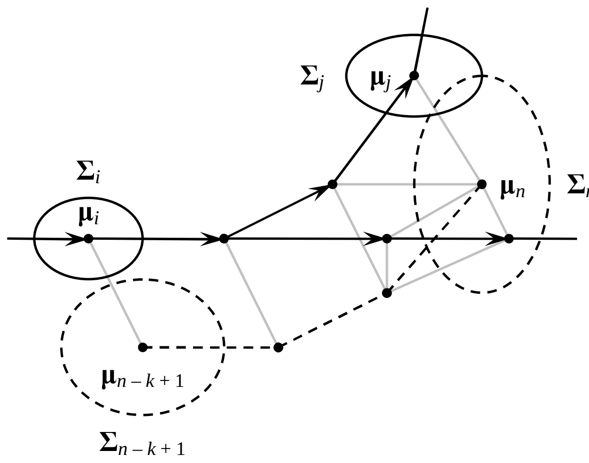

Fig. 2. Illustration of the sequence alignment process. The current trajectory is represented with dashed while the history is depicted with solid line. Using sequence alignment the current position is associated to position j.

1) Viterbi decoding: To illustrate the algorithm, think of a first-order hidden Markov model with k hidden states and transition matrix A with elements

$$\begin{aligned} a_{ij} > 0, (i,j) \in E, \\ a_{ij} = 0, (i,j) \notin E. \end{aligned}$$

(We will typically set a_{ij} so that non-zero elements are identical along each row). Define the conditional probability of emitting symbol j given state i as

$$b_{ij} \propto \exp\left(\lambda\left(i, j\right)\right),$$

Then solving (6) is equivalent to performing Viterbi decoding on this hidden Markov model. Therefore, an efficient implementation will have a complexity of $\mathcal{O}(k|\bar{E}|)$ in the worst case (recall $\bar{E}$ from (5)).

The Viterbi decoder keeps track of the optimal path by means of a trellis. It starts by assigning an equal probability to all nodes in the trellis. At every step of the algorithm, each node is expanded according to all possible transitions to the corresponding state. The transition with the highest likelihood is stored and the node probability is updated according to this likelihood. By keeping track of the nodes that produced the transitions with the highest likelihood values, upon reaching the end of the sequence we can backtrack to find the alignment with the overall maximum likelihood.

Algorithm 1 shows pseudo-code for finding the optimal alignment via Viterbi decoding. The trellis structure is stored in the matrices node and parent; the former contains the node probabilities and the latter keeps track of the optimal transitions. The maximization function in line 5 returns the maximum value of the expression as well as the corresponding index; these two arguments are assigned to element i, j of node and element i, $j-1$ of parent, respectively. Scalar $\mathrm{accumulator}$ is an auxiliary accumulator variable. The alignment indices for the sequence are returned in vector index; i.e. element $n-k+1$ of index is the index of the pose corresponding to pose $n-k+1$, and so on.

Algorithm 1 Sequence alignment using Viterbi.

```
1:  for $j = n-k+1, \ldots, n$ do                                  ▷ Main loop
2:      $\mathrm{accumulator} \leftarrow 0$
3:      for $i = 1, \ldots, n-k$ do
4:          if $j > n-k+1$ then
5:              $\mathrm{node}(i,j), \mathrm{parent}(i,j-1) \leftarrow \max_{h:(h,i)\in\bar{E}} \mathrm{node}(h,j-1)\, a_{hi} \exp(\lambda(h,i))$
                ▷ Search for most likely transitions
6:          else
7:              $\mathrm{node}(i,j)(i) \leftarrow \exp(\lambda(i,j))$
8:          end if
9:          $\mathrm{accumulator} \leftarrow \mathrm{accumulator} + \mathrm{node}(i,j)$
10:     end for
11:     for $i = 1, \ldots, n-k$ do
12:         $\mathrm{node}(i,j) \leftarrow \mathrm{node}(i,j) / \mathrm{accumulator}$
13:     end for
14: end for
15: $i \leftarrow \arg\max \mathrm{node}(i,n)$
16: for $j = n, \ldots, n-k+1$ do                                  ▷ Backtrack
17:     $\mathrm{index}(j) \leftarrow i$
18:     if $j > n-k+1$ then
19:         $i \leftarrow \mathrm{parent}(i,j-1)$
20:     end if
21: end for
22: return index
```

D. Simulation

The main idea is illustrated with a simple simulated vehicle trajectory (navigation code available in [14]). To make the order of the sequence independent of the vehicle speed, in all the implementations presented we first subsample the vehicle trajectory at evenly-spaced distances. Therefore the sequence length is specified as a distance. For this example, the length was defined as 30 metres. Note that for simple graphs like this, a naïve implementation is enough, even for real-time performance.

Fig. 3 shows the simulated trajectory and loop candidates selected by our approach. The vehicle moves counter-clockwise. Only the candidate with the highest likelihood is shown for each pose. The loop detection algorithm was run automatically every 5 metres. There are two important points to note from the results. First, note how the loop candidates are consistent with the shape of the uncertainty ellipses. This is more clearly observed at either the top or bottom parts of the trajectory, where the uncertainty ellipses change their orientation as the vehicle travels from one corner to the next one. Second, the results also illustrate the importance of the shape of the vehicle trajectory in order to reduce ambiguity in the matches. The association just before the vehicle traverses the upper-left corner is not as accurate due to lack of shape in the past sequence. Once the vehicle incorporates the corner into its trajectory, the algorithm selects a better candidate.

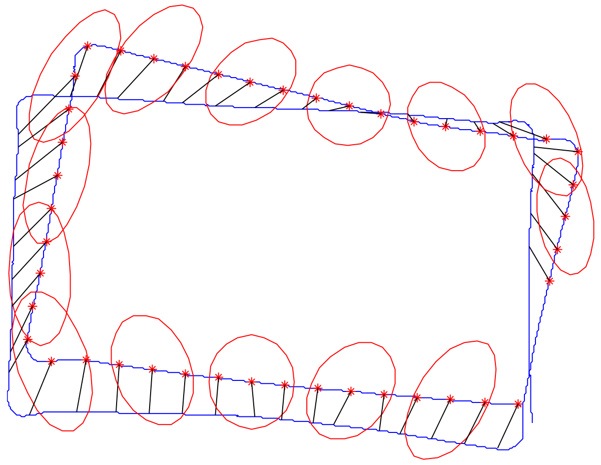

Fig. 3. Loop candidates selection for a simulated trajectory. The figure shows the vehicle trajectory estimated by the motion model. The vehicle travels counter-clockwise. To reduce clutter in the figure, uncertainty is shown only in some of the points where loops were detected. Stars denote these positions, and straight lines connect them to their corresponding loop candidates from the trajectory history.

IV. Experiments

A. Sydney CBD dataset

This section presents experimental results with data gathered with our sensor platform. The platform is equipped with GPS, inertial measurement unit (IMU), lasers and cameras. For our experiments, the position estimated from a Synchronised Position Attitude Navigation (SPAN) system is used. The system outputs the raw data and fusion of a NovAtel GPS and a Honeywell IMU at an average rate of 1 Hz. A Ladybug3 spherical camera mounted on top of the platform collects images for validating the loop candidates.

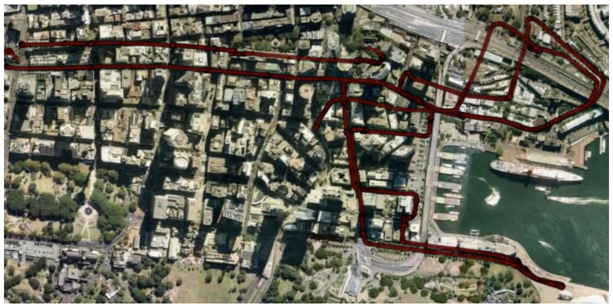

Fig. 4. Vehicle trajectory estimated by our GPS-IMU system, superimposed on an aerial image (rotated 90 deg. clockwise to reduce space). The total trajectory is 9.95 km long. The largest accumulated uncertainty occurs on the top-right part, where the vehicle vehicle estimated position possesses almost 200 metres of error. GPS visibility is extremely low in this area of the city.

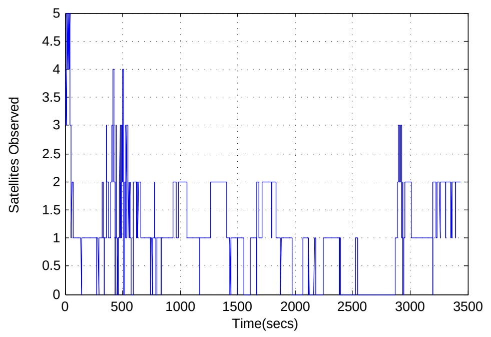

Fig. 5. Number of satellites observed during the experiments. The low satellite visibility is due to the large number of buildings in the city.

Fig. 4 illustrates the trajectory superimposed on an aerial image. The total distance travelled is 9.95 km. Fig. 5 shows the satellite visibility during the experiments. As seen in the figure, the reception is very poor due to the surrounding buildings. This highlights the need for complementary methods for robust localisation.

1) Loop-closure candidates: The loop-closure detection algorithm was set to check for loops every 50 metres. The sequence length was defined as 150 metres. A total of 54 positions were checked. The candidate with the highest likelihood is tested for consistency against a χ^2 gate validation test. The threshold value for the gate was set to 9.49. This is 95% quantile of a χ^2 distribution with 4 degrees of freedom (3D position plus heading).

Figs. 6 and 7 illustrate the results obtained for the northern and a zoom-in of the centre part of the trajectory respectively. The central part is the most interesting in terms of loops. In this part, the navigation system accumulates an uncertainty of over 200 metres due to the poor GPS visibility. Points where the approach found candidates are denoted with starts, whereas points where no candidates were found are represented with circles. The lines depict the best candidate position for each point.

Out of the 54 points, 46 are actual loops. The algorithm detected a total of 30 loops. The main reason for the false negatives (undetected loops), can be attributed to an overconfident estimate of the covariance generated by the navigation filter.

2) Candidates validation: The 30 points where candidates were found were verified by applying SIFT matching combined with RANSAC between images corresponding to current and candidate positions [1]. Notice that only the candidate with the highest likelihood was selected, and thus only one image per position needs to be matched. This is the main advantage of applying our method prior to image matching. Particularly, in areas with large uncertainty in position, trajectory matching as a means for pre-selecting candidates can drastically reduce the computational cost of a vision-only loop-closure detection.

Visual-matching verified 24 out of the 30 loops detected. Figs. 6 and 7 depict correct candidates with straight lines and wrong candidates with dashed lines. To avoid clutter in the figure, we set the algorithm to start searching for loops after the whole trajectory has been visited once.

Fig. 8 shows examples of images from the LadyBug3 camera corresponding to the current and the most likely candidate positions. The first 4 were accepted by visual-matching whereas the last 4 were rejected (dashed lines in Fig. 6). Note that the visual matching rejected the 6 wrong candidates.

V. Discussion

Many methods for loop-closure candidates selection have been presented in the literature (see Section II). Some of them have been shown to give very good results. The main drawbacks we found in previously presented approaches is that they either require a previous training process or they are expensive to compute. Another disadvantage is the general complexity involved in the implementation, sometimes requiring a bigger effort than the one needed to implement the navigation filter itself.

This paper studied the practicability of using a simple probabilistic cost function to detect loops based on the shape of the trajectory. The results show that this very simple idea can be a powerful tool to complement

[1]Since visual matching is not the aim of this paper, we compare all the images from the spherical camera. A better implementation would compare, for example, only the ones with overlap in the field of view.

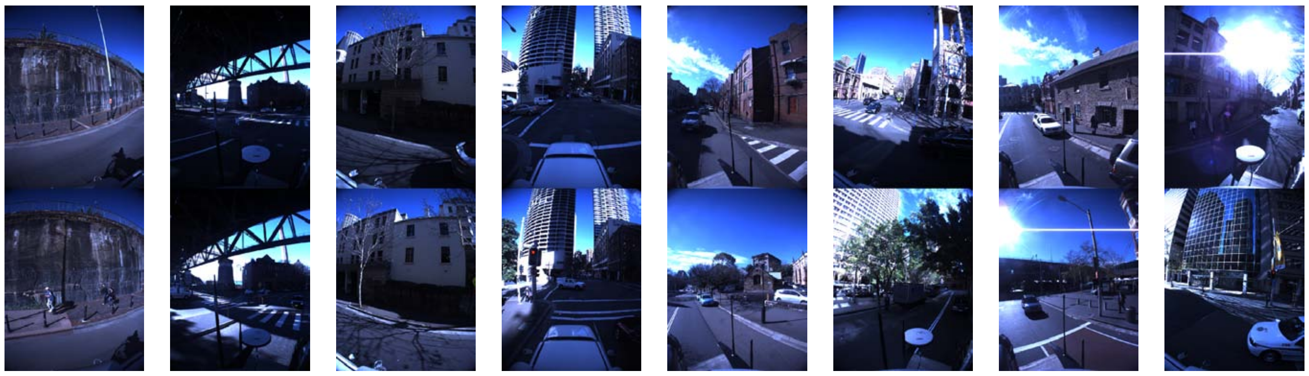
Fig. 8. The figure shows the current (top images) and candidate (bottom images) for positions 4,36,50,54,39,48,49 and 52. The first 4 are images with correct associations while the last 4 correspond to candidates selected by the sequence matching approach but rejected by visual matching.

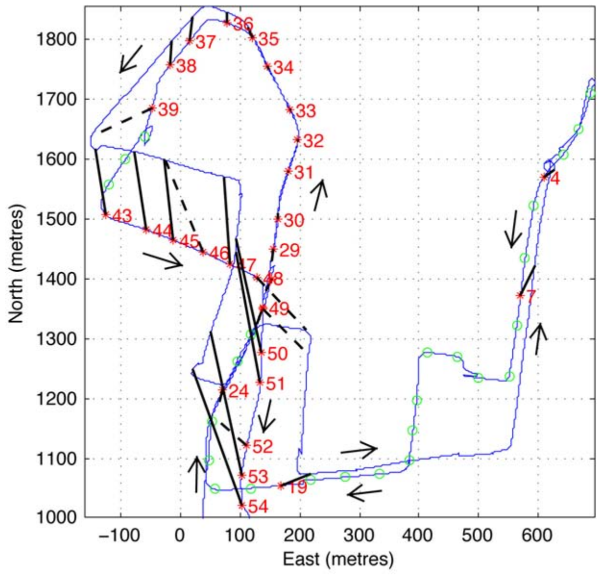

Fig. 6. Loop candidates selected from the experimental data. Positions where loop candidates were found are denoted with '*'. Associations are represented with lines. Dashed lines denote incorrect ones. Verification was done by visual matching. Arrows indicate vehicle direction.

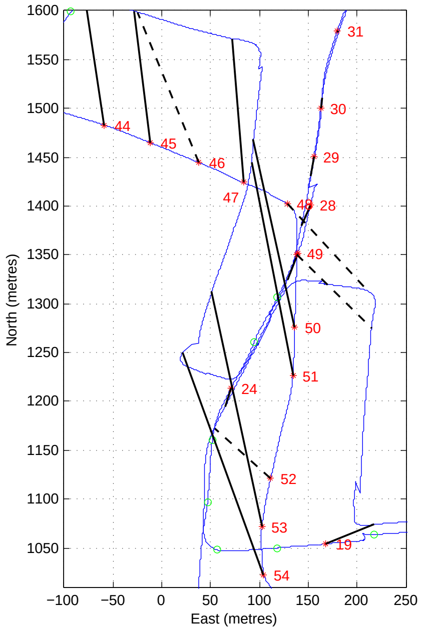

Fig. 7. Loop candidates selected in the north-central part of the trajectory.

current navigation approaches. Among the 30 best loop candidates selected, only 6 of them were wrong. The wrong associations in points 39, 45 and 52 were due to the lack of features. Since the vehicle travels in a straight line, the approach associates the nearest point according to the shape of the covariance. Although points 48 and 49 were matched to the wrong candidates due to the symmetry of the trajectory (two corners near the current position), this serves to illustrate the robustness of the approach in the sense of finding similar shapes in the trajectory. As soon as the vehicle travels further, ambiguity is reduced and the algorithm managed to select good candidates (50,51). Points 53 and 54 were assigned good candidates thanks to the good alignment of the sequence and the relatively low heading uncertainty.

The algorithm missed some loops. Positions after point 7 should have been associated with the nearest ones. Due to the very low heading uncertainty in this sector, the associations failed the probabilistic test. The two positions before point 19 were not matched due to the curve present when that street is revisited. After this curve is removed from the reference sequence, the algorithm correctly associates the parallel lines (point 19).

The method presented has only one parameter that must be specified by the user, namely, the sequence length k. After processing our experimental data with different values, we found that, although loop candidates vary, the algorithm still detects good candidates as long as k has "sensitive" values. For example, if the length ranges between 100 and 400 metres, the algorithm extracts good loop candidates from the city data set. The range of values will have to change, if for instance, the vehicle travels along country roads. In that case, the sequence length will be in the order of kms.

VI. Conclusions

This paper presented an evaluation of an approach that uses the vehicle trajectory sequence to generate loop-closure candidates. A probabilistic cost function for obtaining maximum likelihood estimates was presented. The cost function is straightforward to implement and has linear computational cost. Every time a loop-closure is evaluated, the algorithm selects a sequence of vehicle poses leading up to the current pose, and compares this against the trajectory history. Experimental results with real data taken in the city showed that the algorithm can be a very powerful tool to complement current techniques.

Acknowledgements

This work has been supported by the Australian Centre for Field Robotics and the Rio Tinto Centre for Mine Automation. The authors would like to thank James Underwood and Laura Merry for their help with the dataset.

Appendix: Derivation of the Likelihood Function

Let $\mathbf{x}_i$, $\mathbf{x}_j$ be jointly-distributed Gaussian random variables

$$\begin{bmatrix} \mathbf{x}_i \\ \mathbf{x}_j \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu_i \\ \mu_j \end{bmatrix}, \begin{bmatrix} \mathbf{\Sigma}_{ii} & \mathbf{\Sigma}_{ij} \\ \mathbf{\Sigma}_{ji} & \mathbf{\Sigma}_{jj} \end{bmatrix}, \right).$$

Define the random variable

$$\mathbf{y}_{ij} = \mathbf{x}_i - \mathbf{x}_j = \begin{bmatrix} \mathbf{I} \\ -\mathbf{I} \end{bmatrix}^T \begin{bmatrix} \mathbf{x}_i \\ \mathbf{x}_j \end{bmatrix}.$$

Then, $\mathbf{y}_{ij}$ is Gaussian-distributed with mean

$$\begin{bmatrix} \mathbf{I} \\ -\mathbf{I} \end{bmatrix}^T \begin{bmatrix} \mu_i \\ \mu_j \end{bmatrix} = \mu_i - \mu_j$$

and covariance matrix

$$\begin{bmatrix} \mathbf{I} \\ -\mathbf{I} \end{bmatrix}^T \begin{bmatrix} \mathbf{\Sigma}_{ii} & \mathbf{\Sigma}_{ij} \\ \mathbf{\Sigma}_{ji} & \mathbf{\Sigma}_{jj} \end{bmatrix} \begin{bmatrix} \mathbf{I} \\ -\mathbf{I} \end{bmatrix} = \mathbf{\Sigma}_{ii} + \mathbf{\Sigma}_{jj} - \mathbf{\Sigma}_{ij} - \mathbf{\Sigma}_{ji},$$

which is equal to $\mathbf{\Gamma}_{ij}$ in (3). Then, the probability of the event $\mathbf{x}_i = \mathbf{x}_j$ is given by

$$P(\mathbf{x}_i = \mathbf{x}_j) = P(\mathbf{y}_{ij} = 0) = p(\mathbf{0}),$$

where p is the probability density function of $\mathbf{y}_{ij}$. Taking the logarithm in the equation above yields the formal definition of the likelihood function in (2).

References

[1] T Bailey and H Durrant-Whyte, "Simultaneous localization and mapping (slam): Part ii," *IEEE Robotics & Automation Magazine*, Jan 2006.

[2] K Granström, J Callmer, F Ramos, and J Nieto, "Learning to detect loop closure from range data," in *IEEE International Conference on Robotics and Automation*, 2009.

[3] M Bosse and R Zlot, "Map matching and data association for large-scale two-dimensional laser scan-based slam," *The International Journal of Robotics Research*, Jan 2008.

[4] J Nieto, T Bailey, and E Nebot, "Recursive scan-matching slam," *Robotics and Autonomous systems*, Jan 2007.

[5] M Magnusson, H Andreasson, A Nüchter, and A Lilienthal, "Automatic appearance-based loop detection from 3d laser data using the normal distributions transform," *aass.oru.se*, Jan 2009.

[6] B Williams, M Cummins, J Neira, P Newman, I Reid, and J Tardós, "A comparison of loop closing techniques in monocular slam," *Robotics and Autonomous Systems*, 2009.

[7] P Newman, G Sibley, M Smith, M Cummins, A Harrison, C Mei, I Posner, R Shade, D Schroeter, L Murphy, W Churchill, D Cole, and I Reid, "Navigating, recognizing and describing urban spaces with vision and lasers," *The International Journal of Robotics Research*, vol. 28, no. 11-12, pp. 1406–1433, Nov 2009.

[8] JMM Montiel, J Civera, and AJ Davison, "Unified inverse depth parametrization for monocular slam," *Proceedings of Robotics: Science and Systems*, 2006.

[9] E Mouragnon, M Lhuillier, M Dhome, F Dekeyser, and P Sayd, "Monocular vision based slam for mobile robots," *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, vol. 3.

[10] D Nister and H Stewenius, "Scalable recognition with a vocabulary tree," *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2, 2006.

[11] K Konolige, J Bowman, J. D Chen, P Mihelich, M Calonder, V Lepetit, and P Fua, "View-based maps," *The International Journal of Robotics Research*, pp. 1–17, May 2010.

[12] Ian Mahon, Stefan Williams, Oscar Pizarron, and Matthew Johnson-Roberson, "Efficient view-based slam using visual loop closures," *Robotics, IEEE Transactions on*, vol. 24, no. 5, pp. 1002 – 1014, 2008.

[13] Roy Anati and Kostas Daniilidis, "Constructing topological maps using markov random fields and loop-closure detection," .

[14] T. Bailey and J. Nieto, "Slam simulations," `http://www-personal.acfr.usyd.edu.au/tbailey/software/slam_simulations.htm`, 2010.