Environment Identification by Alignment of Abstract Sensory Flow Representations

FREDRIK LINÅKER*,†  KIM LAURIO∗‡

*Department of Computer Science, University of Skövde
P.O. Box 408, SE-541 28 Skövde, SWEDEN
†Department of Computer Science, University of Sheffield
Regent Court, 211 Portobello Street, Sheffield S1 4DP, UNITED KINGDOM
‡Department of Computer Science, University of Exeter
Old Library, Prince of Wales Road, Exeter EX4 4PT, UNITED KINGDOM
fredrik.linaker@ida.his.se  http://www.his.se/ida/~fredrik/
kim.laurio@ida.his.se  http://www.his.se/ida/~kim/

Abstract: We present an approach for robot environment identification based on abstract sensory flow representations. The approach consists of an unsupervised mechanism for forming a growing set of concepts from the sensory signal, and an alignment algorithm for comparing concept sequences. Each environment will lead to a different concept sequence, i.e., different abstract sensory flow, which can be efficiently stored and compared with new sensory data as the robot moves about. One advantage of using this approach is that the environments need not contain any unique landmarks nor does the robot need any sort of compass or odometry tracking. Instead, it relies solely on the sequence of sensory inputs and a systematic pattern of movement, namely wall following around the environment. Finally, we show that this simple approach is also able to correctly identify situations where the robot has been placed in previously unencountered environments as the sequence alignment technique used will yield lower matching scores in these situations.

Key Words: localization, concept formation, segmentation, alignments, robotics, growing knowledge

1 Introduction

Consider an automated window cleaner which works its way through an office building and cleans the windows. Each room may have its own cleaning specifications, e.g., windows in front of the building, near the entrance, should be cleaned more frequently than others. Personnel may at any time do a manual override and move the robot to their own office, or any other part of the building, and command it to immediately clean their windows. When finished, the robot should resume its normal duties. This will however require that the robot finds out where it is in the office environment. This problem is also known as the lost robot problem [3], i.e., the robot needs to gather evidence about where it presently is located in the environment.

As shown in Fig. 1, it may, however, be very difficult to find out where the robot is located based on the information available from the sensors as the environment can look very similar. That is, the robot is experiencing the problem of perceptual aliasing—there are many different locations in the environment which correspond to any given input.

In fact, the environments may even be so similar that they contain no unique landmarks at all, but still, the robot should somehow be able to find out where it is located. We will show, in the following, that taking into account sensor inputs over a long time, the robot will be able to identify in which environment it has been placed.

Consider the environments in Fig. 1. Having sensed one of the corridors, it will take at least 130 time steps (sensor samples) at maximum speed to move to a location at which the other corridor can be sensed. (In these experiments the robot is wall following which means that the robot will in fact not reach the location of the possible other corridor until after 800 time steps.) The problem is that the memory capacity of mobile robots is often very limited so storing all these inputs may be difficult, and then finding the relevant information for localization among hundreds, or thousands, of stored inputs is an intractable task.

But, instead of storing each individual input, the inputs can be filtered and a very compact abstract representation can be stored of the entire input sequence. In doing this, the space/time differences between relevant inputs can in fact be arbitrarily large without affecting the environment identification performance of our approach. That is, the distance can just as easily be 130,000 inputs or even 13 million inputs between the relevant inputs. Our technique for extracting the ab-

1This is similar to the approach by Neuhäuser and Smithe [4], but here we show how an arbitrarily large number of inputs can be taken into account, instead of just a limited [pre-defined] window of past events.
are stored in an input buffer $X(t)$:

$$X(t) = \{x(t), \ldots, x(t - n + 1)\}. \quad (1)$$

The values in the input buffer are averaged to create a more reliable, filtered, input $\mathcal{F}(t)$ to the rest of the network:

$$\mathcal{F}(t) = \frac{1}{n} \sum_{i=1}^{n} x_i; x_i \in X(t), \quad (2)$$

where $x_i$ is the $i$th member of the input buffer, i.e. $x(t - i + 1)$.

The set $M(t)$ of concepts is initially empty (the ARAVQ does not start working until the input buffer is filled, i.e., until time step $n - 1$):

$$M(n - 1) = \emptyset. \quad (3)$$

For convenience, we define the following general distance measure between a set of (model/filtered input) vectors $V$ and a set of actual inputs $X$:

$$d(V, X) = \frac{1}{|X|} \min_{1 \leq j \leq |V|} \{||x_i - v_j||; x_i \in X, v_j \in V\}, \quad (4)$$

where $|| \cdot ||$ denotes the Euclidean distance measure.

The distance between the filtered input and the actual inputs is defined as:

$$d_{\mathcal{F}(t)} = d(\{\mathcal{F}(t)\}, X(t)), \quad (5)$$

and the distance between the existing model vectors and the actual inputs is:

$$d_{M(t)} = \begin{cases} d(M(t), X(t)) & |M(t)| > 0 \\ \epsilon + \delta & \text{otherwise}. \end{cases} \quad (6)$$

If both the stability and novelty criteria are met, i.e. the mismatch between the filtered input and the current inputs are less than $\epsilon$, and the mismatch with the existing model vectors compared to the filtered input is larger than $\delta$, the filtered input is incorporated as an additional model vector:

$$M(t + 1) = \begin{cases} M(t) \cup \mathcal{F}(t) & d_{\mathcal{F}(t)} \leq \min(\epsilon, d_{M(t)} - \delta) \\ M(t) & \text{otherwise}. \end{cases} \quad (7)$$

Each time step, a winning model vector $\text{win}(t)$ is selected, indicating which concept the (filtered) input currently matches:

$$\text{win}(t) = \arg \min_{1 \leq j \leq |M(t)|} \{||\mathcal{F}(t) - m_j||; m_j \in M(t)\}. \quad (8)$$

If the winning model vector matches the (filtered) input very closely, the (filtered) input is considered to represent a ‘typical’ instance of the concept, and the model vector is modified to match the input even closer:

$$\Delta m_{\text{win}(t)} = \begin{cases} \alpha [\mathcal{F}(t) - m_{\text{win}(t)}] & ||\mathcal{F}(t) - m_{\text{win}(t)}|| < \epsilon \\ 0 & \text{otherwise}, \end{cases} \quad (9)$$

where $\alpha$ is a user defined learning rate. The structure of the ARAVQ network is depicted in Fig. 2.

---

Fig. 1: Three of the environments used in these simulations. Rooms 0, 1, and 2 have corresponding numbers of short corridors, or ‘doorways’, placed in different locations. Other than this, there are no unique landmarks in any of the environments; the only way to differentiate between the different environments is to incorporate sensor readings with long time in between (at least 130 time steps).

2 The ARAVQ

The adaptive resource allocating vector quantization (ARAVQ) network [2] is a vector quantization network which is able to dynamically allocate additional model vectors to represent any novel and stable situations the input signal reflects.

The amount of resources (model vectors) which are allocated depend on the characteristics of the input signal. (Which in turn reflects the characteristics of the environment, or the agent-environment interaction, which underlies the sensory flow which we apply the ARAVQ network to here.)

The ARAVQ has four user defined parameters: a novelty criterion $\delta$, a stability criterion $\epsilon$, an input buffer size $n$ and a learning rate $\alpha$. These parameters are described below.

The last $n$ input vectors $x(t), x(t - 1), \ldots, x(t - n + 1)$
2.1 Sensory flow segmentation

The ARAVQ network was set to segment the incoming sensory flow of a (simulated) mobile Khepera robot. The robot was controlled using a fixed wall following behaviour, moving counter-clockwise around the environment. At each time step, the sensor inputs of the Khepera’s eight distance sensors and two proprioceptive motor sensors were normalized to the range [0, 0.9] and combined into a 10 dimensional vector and fed as input to an ARAVQ network. The robot moved about in the environment and at each time step, the index of the winning model vector was plotted using a colour of its own, making the trail shown in Fig. 3. As originally noted by Tani & Nolfi [5], an interesting result is that the inputs are split into concepts which to a distal observer correspond to walls, corridors and corners, respectively.

Fig. 3: Acquired segmentation of the sensory flow in room 1 using the ARAVQ. The number of extracted concepts depends on the parameter settings of the network, as discussed below. (For convenience, the model vectors have been labeled alphabetically in this and all of the following pictures, instead of \( m_1, m_2, m_3 \), etc.)

Tani & Nolfi [5] used a more complicated mechanism for extracting these concepts, where the user manually had to specify how many concepts the system should extract. As shown in [2], the ARAVQ has the advantage of being able to determine this by itself, depending on the complexity of the input signal. As we will show, however, while the number is no longer explicitly defined by the operator, the number is implicitly imposed through the choice of ARAVQ parameters. (The ARAVQ parameter settings used to acquire the segmentation depicted in Figs. 3 and 4 were \( \delta = 0.80, \epsilon = 0.20, n = 10 \) and \( \alpha = 0.05 \).) The parameter settings can however be evaluated through their discriminative ability in the environment identification task and can thus be compared using quantitative rather than just qualitative comparisons, as shown below.

2.2 ARAVQ parameter settings

The ARAVQ has four parameters, of which the mismatch reduction requirement \( \delta \) and the stability criterion \( \epsilon \) have the greatest influence on the number and type of concepts which are incorporated. Decreasing the mismatch reduction criterion \( \delta \) means that it becomes easier for inputs to qualify as being novel, and thus it becomes easier to be incorporated as a new model vector. Increasing \( \epsilon \) will on the other hand relax the stability criterion so that even very different inputs will be considered to be similar enough for incorporation as a new concept.

Several ARAVQs with different \( \delta \) and \( \epsilon \) parameters were put to segment the sensory flow of several laps in room 2. The resulting number of concepts are depicted in Fig. 5.

A small set of parameter settings, yielding different numbers of concepts, were selected and are presented in Fig. 6. The different parameter settings in Fig. 6 yielded the segmentations depicted in Fig. 7.

When six or more concepts were extracted, the resulting segmentation was unstable, in that different concepts could be used for the same location (actually for the inputs at the same location) during different laps. This is because the model vectors, representing each concept, share the same input space. When additional model vectors are incorporated, they steal space from the existing model vectors, whose uptake regions consequently become smaller. The smaller the regions representing each concept, the lower the probability that similar inputs are classified in the same manner.
3 Environment classification
A signature for each environment can be generated by storing the sequence of model vector winners for one lap. Depending on the parameter settings of the ARAVQ, it may turn out that these signatures are identical for different environments, see e.g., Fig. 7 where using only one concept will make the signatures for all rooms identical. (In that case, the room identification will be impossible.)

Once these signatures have been extracted (an example of this is shown in Fig. 8), they can be used for identification. If the robot is placed randomly in one of the environments, it should be able to match its current abstract sensory flow, or evidence sequence, (see Fig. 9) with the extracted signatures, and thereby find out where it is.

The environment with the best matching signature is most likely where the robot actually is. Such a comparison is analogous to a well known and much studied problem in many application areas: Finding the optimal alignment for a pair of sequences. In our case the problem is complicated by the fact that the evidence can either be an incomplete sequence “taken” from an already known signature (with the possible added complication of circular permutation), or be data collected from a previously unseen environment. No matter the situation, we want the algorithm to find the best alignment between the evidence sequence and any subsequence of the signature sequence. This is commonly referred to as local alignment [1].

4 Local alignments
The local alignment algorithm creates the best possible alignment between subsequences of the compared pair and if the evidence and signature sequences are identical, the alignment will eventually extend to the full length of the sequences. Together with a quantitative scoring method for alignments this gives us the possibility to form hypotheses about which environment the robot currently is in, and also to detect completely new environments.

Briefly, finding the optimal alignment consists of two parts: An additive scoring scheme for each aligned symbol vector (with terms for each gap), and a dynamic programming algorithm for constructing the optimal alignment given a specific scoring scheme. Initial experiments
showed that a penalty of 8 for gaps and the following scoring function for two symbols \(x\) and \(y\) was adequate:

\[
s(x, y) = \begin{cases} 
  10 & x = y \\ 
  -5 & \text{otherwise} 
\end{cases}
\]  

(10)

Note that it is possible to have several different alignments that return equal scores.

In order to simplify our analysis of the system we chose to normalize all scores by dividing them with the maximum possible score for an ungapped alignment (which would be obtained if the two sequences were identical).

5 Results

One example of the alignment technique in action is given in Fig. 10 where the evidence sequence \(b\ a\ c\) is gradually aligned to the known environments (rooms 0, 1 and 2) as the robot moves. When only one model vector winner has been observed (in this case the one a distal observer could denote corner), it is impossible to know which room the robot is in since all rooms have corners. A corner followed by a wall is also not enough to discriminate between the three environments.

It is first when the model vector for corridor is the winning one that it is possible to decide that it cannot be in room 0. The score for room 0 does not go to zero because it still matches the \(b\ a\), which is 2/3 of \(b\ a\ c\).

In case the signature and evidence signatures do not originate from the same starting point in the room, we have to be able to handle circular permutations. We have solved this by adding at least \(k - 1\) extra symbols at the end of the signature sequences (where the evidence sequence is of length \(k\)). These extra symbols are taken from the beginning from the same signature, when needed, so that we get a sort of wrap around effect. This can be seen in Fig. 10 in the form of the shaded symbols at the rightmost end.

The complete run for one experiment is shown in Fig. 11, and clearly it is not possible to discriminate between rooms 1 and 2 until the second corridor is encountered. There is a temporary increase in the score for room 0 as the robot moves along and this happens because the cost of introducing a mismatch is eventually compensated by the additional matches which it leads to.

Since we normalize the score from the alignment with the maximum possible score, we can also easily detect new rooms. To show this, room 3 was created with three corridors, see Fig. 12.

The robot was placed in room 3 and it started comparing its evidence sequence with the known signatures as it moved. As depicted in Fig. 12, the scores indicate
that none of the known environments are consistent with the gathered evidence.

The overall performance using the parameter settings (Fig. 6) is summarized in Fig. 13. Note that as the number of concepts reaches a critical limit (here 6 concepts), the identification performance starts to decrease. This is because there are too many concepts which represent basically the same things; small perturbations in the robot’s movement will be reflected in the sensory signal, which is then in turn classified as belonging to a different concept, although there really is no significant change happening.

Fig. 13: Number of correctly identified environments using different numbers of extracted concepts.

6 Conclusions

We have presented a system which is able to correctly identify different environments based on abstract sensory flow representations. It does so without using any sort of compass, Cartesian map, odometric information, nor does it rely on any unique landmarks. The abstract sensory flow representations are compact descriptions of the environments which are independent of the actual sizes involved. (This however means that rooms which differ only in scaling cannot be differentiated.)

The system extracts a set of concepts automatically from the sensory flow and uses these concepts to create individual environment signatures. These can be matched in parallel against a gathered evidence sequence in case the robot gets lost, using a standard alignment technique. As we have shown, this enables the robot to localize itself from random starting positions and also to detect new environments. The alignment technique always generates a hypothesis (best matching score) for the evidence, even if there is no perfect match among the known environments.

The parameter settings of the extraction network will influence the number of concepts formed. This will in turn affect the environment identification performance, and will thus provide a quantitative measurement of the quality of the parameter settings. The results show that paying too much attention to details, i.e. having many different concepts for the same thing, may actually hurt performance in a similar manner as having too few concepts. However, the technique presented here provides help in choosing an appropriate level of granularity. This is important since our robots must not pay too much attention to detail, nor too little, or they may find themselves lost in their struggle to find their place in the world.

Acknowledgements

This paper was supported by grants from The Foundation for Knowledge and Competence Development (1507/97), Sweden, and the University of Skövde, Sweden.

References:


