Robot localization based on KS-FAM

October 13, 2010

1 Description

The objective is mobile robot vision based localization using associative memories. The map stores a path previously followed by the robot in the form of several view "landmarks" representing points of interest in the path. Those landmarks will identify a section of the path, dividing it in a sequence of locations without gaps between them. These landmarks are stored as gray-scale patterns in a Kosko Subsethood Fuzzy Associative Memory (KS-FAM) [1]. Localization will be performed by feeding the KS-FAM with the images that the robot acquires in its movement, obtaining from it the recognized position.

2 Experiment details

For the experiment, the optical image database already recorded is used [5, 4, 6, 7, 2, 3]. Results shown here are obtained from the first recorded path. As in other experiments, the first walk is used for training and the remaining 5 for testing. The sample path contains 11 relevant positions, that will be the number of associations stored in the memories.

The code for the KS-FAM was provided by prof. Peter Sussner[1].

Available example uses of KS-FAM are as Auto-Associative memories. In this experiment, the Auto-Associative type has the additional problem of estimating which position is the one recalled by the memory. Visual examination of results with both Auto-Associative and Hetero-Associative memories seemed to give very similar results. So, in a first approach, Hetero-Associative memories are used and after evaluating their results, the same experiment will be performed with Auto-Associative memories to compare their performance.

In the first experiment, the reference path map used identified each localization with a single landmark, corresponding to the position of the interest location in the map. In further experiments, each of the locations was identified by several views arround the landmark, well distributed along the path segment corresponding to it.

[1] http://www.ehu.es/ccwintco/groupware/webdav.php/apps/phpbrain/142/KSFAM%20-%20Code.rar

2.1 Hetero-Associative case

In the pairs (x,y), x will be the pattern (gray-scale image corresponding to the landmark that is going to be stored) and y will be a vector of size n = # of patterns to store. The vector will be composed of 0's, except for one 1 in the vector position corresponding to the map position of the stored pattern. e.g:

Being $X = \{x_1, x_2, x_3, x_4, x_5\}$ the patterns that we want to encode in the KS-FAM. The pair y_2 of pattern x_2 (second pattern in the path) will be $y_2 = [01000]$. Y (the matrix of outputs) will be then (vectors stored column-wise):

$$Y = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix},$$

which corresponds to an identity matrix of size nxn.

Initially, a simpler approach was used, being y_i a scalar identifying the position (i.e. '2' for the second position instead of [01000]). However, results obtained with that method were much worse.

For validation purposes, the same ground division based on the odometry data of previous experiments has been used.

2.2 View selection

In the case of maps with several views representing each location, those views are selected according to the number of views to choose and the size of the segment, in order to get views well spread along the segment. The cental view of the segment is selected and views are selected ahead and behind it, with steps equal to the number of views in the segment divided by the number of views we want to select. For example, if we want to select 5 views from a segment of 25 views, first we choose the central view (view #13) and two views ahead and behind it at spaces of 5 views (# of views in the segment / # of views we want to select). In this way, the selected views will be 3, 8, 13, 18 and 23.

Every selected view for each segment will be encoded in the memory with the same pair: the vector corresponding to the reference position.

3 Implementation details

3.1 Hetero-Associative case

First, the image database is transformed to gray-scale [0,1], as is done in the sample code provided by Sussner.

```
for i = 1:nWalks
        for j = 1:tamsBD(i);
                bdImagenes{i}(j,:) = mat2gray(bdImagenes{i}(j,:));
```

```
        end
end
```

The patterns matrix is built using the images of the selected landmark positions from the first walk.

```
X = zeros(tamVec, nSitios); % reservo espacio para matriz de patrones
% obtengo los patrones (imagenes de los landmaks)
for i = 1:nSitios
        X(:,i) = bdImagenes{1}(sitios(i), :);
end
```

Output patterns matrix is built as the identity matrix.

```
Y = eye(nSitios); % cada vector tendrá un 1 en la posición correspondiente
```

Mxz and Wzy memories are built using the input and output pattern matrices.

```
Mxz = BoxMax2(eye(nSitios), -1*X',-Inf);
Wzy = BoxMin2(Y, -1*eye(nSitios), Inf);
```

For each test walk i, the images are put in an input matrix and feed to the memories. Some of the code is redundant or unnecessary, but was done like that to make sure that it was being done correctly.

```
Xin = zeros(tamVec, tamsBD(i));
for j = 1:tamsBD(i)
        Xin(:,j) = bdImagenes{i}(j,:);
end
[Yout,u]  =  AMM_Nova(Xin,Mxz,Wzy);
```

Output vectors are translated to scalars identifying the positions ('find' returns the nonzero position in the vector) .

```
posLoc(j) = find(Yout(:,j));
```

Success rate is calculated for each walk ($i+1$ because the first walk was used for training) using the path division based on odometry.

```
aciertos(i) = sum(posLoc{i}(:) == gruposOdo{i+1}(:))/tamsBD(i+1);
```

3.2 Auto-Associative case

First, the image database is transformed to gray-scale [0,1], as is done in the sample code provided by Sussner.

```
for i = 1:nWalks
        for j = 1:tamsBD(i);
                bdImagenes{i}(j,:) = mat2gray(bdImagenes{i}(j,:));
        end
end
```

The patterns matrix is built using the images of the selected landmark positions from the first walk.

```
X = zeros(tamVec, nSitios); % reservo espacio para matriz de patrones
% obtengo los patrones (imagenes de los landmaks)
for i = 1:nSitios
        X(:,i) = bdImagenes{1}(sitios(i), :);
end
```

Output patterns matrix is the same than the patterns matrix.

```
Ya = X; % salida en el caso de las autoasociativas
```

Mxz and Wzya memories are built using the input and output pattern matrices.

```
Mxz = BoxMax2(eye(nSitios), -1*X',-Inf);
Wzya = BoxMin2(Ya, -1*eye(nSitios), Inf);
```

For each test walk i, the images are put in an input matrix and feed to the memories. Some of the code is redundant or unnecessary, but was done like that to make sure that it was being done correctly.

```
Xin = zeros(tamVec, tamsBD(i));
for j = 1:tamsBD(i)
        Xin(:,j) = bdImagenes{i}(j,:);
end
[Yout,u] = AMM_Nova(Xin,Mxz,Wzya);
```

The obtained output is compared with the stored patterns. The recognized position is the closest pattern. Since the memory always retrieves one of the stored patterns, the lowest difference will be equal to 0.

```
for i = 2:nWalks
        % Compruebo a que posición corresponde cada imagen devuelta
        outWalk = grupos{i-1};
        for j = 1:size(outWalk,2)
                % Compruebo cual es el landmark mas cercano
                difsAc = zeros(tamVec, nSitios);
                for k = 1:nSitios
                        difsAc(:,k) = X(:,k) - outWalk(:,j); %distancia entre
                end
                [ordenados, orden] = sort(sum(abs(difsAc))); %sumo diferencias
                posLoc{i-1}(j) = orden(1); %me quedo con el menor
        end
end
```

Success rate is calculated for each walk ($i+1$ because the first walk was used for training) using the path division based on odometry.

```
aciertos(i) = sum(posLoc{i}(:) == gruposOdo{i+1}(:))/tamsBD(i+1);
```

4 Results

Obtained results are rather poor, as can be appreciated in tables 1 and 2. Surprisingly, the best results were obtained using the smallest images. Also, exactly the same results were obtained with both Hetero-Associative and Auto-Associative memories.

Table 3 shows the results obtained using several views to represent each position, using the smallest images. We can appreciate some improvement, being the best results with 5 and 9 views, and degrading with higher number of views by position. The view selection algorithm was automatic, so with higher number of images we can not garantee that one view could be selected several times when the number of views to select is greater than the number of views in the segment. The table shows also the results obtained storing all the images of the path in the KS-FAM.

The computation times of the Auto-Associative memories are much higher (figures 1 and 2) with no appreciable improvement in the obtained results (Note: the higher computation time of the 2nd walk is probably due the program reserving memory for the first time for the Xin variable).

Image size	Walk 2	Walk 3	Walk 4	Walk 5	Walk 6	Mean
242x314	0.3221	0.3812	0.2883	0.3264	0.246	0.3128
121x157	0.2969	0.3193	0.2909	0.3107	0.2086	0.28528
61x79	0.4678	0.4629	0.4494	0.389	0.4171	0.43724

Table 2: Position recognition success rates obtained using Auto-Associative KS-FAM, with images of different sizes.

# of views	Walk 2	Walk 3	Walk 4	Walk 5	Walk 6	Mean
5	0.619	0.5891	0.4727	0.4517	0.4733	0.52116
7	0.4734	0.3911	0.3481	0.3551	0.3556	0.38466
9	0.6162	0.4975	0.4987	0.4674	0.5214	0.52024
11	0.521	0.3738	0.3403	0.3629	0.3503	0.38966
13	0.5042	0.4158	0.3506	0.3681	0.3663	0.401
15	0.4818	0.3614	0.3455	0.3316	0.3102	0.3661
17	0.4566	0.3787	0.3299	0.3577	0.3529	0.37516
All	0.5462	0.4356	0.4078	0.3525	0.4064	0.4297

Table 3: Position recognition success rates using different number of views for each position.

Image size	Walk 2	Walk 3	Walk 4	Walk 5	Walk 6	Mean
242x314	0.3221	0.3812	0.2883	0.3264	0.246	0.3128
121x157	0.2969	0.3193	0.2909	0.3107	0.2086	0.28528
61x79	0.4678	0.4629	0.4494	0.389	0.4171	0.43724

Table 1: Position recognition success rates obtained using Hetero-Associative KS-FAM, with images of different sizes.

5 Discussion

The dense nature of the map can cause “overlapping” recognition areas. That is, the views at both sides of the boundaries of two consecutive positions are very similar, and they may be recognised as belonging to the neighbour and not to the actual position they belong to. This problem would be responsible for a small percentage of the error rate. There is also a problem when different positions have views quite similar. This problem can be appreciated in the long corridor. However there are other recognition problems whose source must be necessarily

```
>> localizacionKSFAM
Creando matrices de entrada y salida.
Elapsed time is 0.003206 seconds.
Calculando Mxz.
Elapsed time is 0.632443 seconds.
Calculando Wzy.
Elapsed time is 0.004929 seconds.
Calculando localizacion walk 2.
Elapsed time is 4.189256 seconds.
Calculando localizacion walk 3.
Elapsed time is 0.903991 seconds.
Calculando localizacion walk 4.
Elapsed time is 0.968646 seconds.
Calculando localizacion walk 5.
Elapsed time is 0.982962 seconds.
Calculando localizacion walk 6.
Elapsed time is 0.856521 seconds.

tTotal =

        8.5421
```

Figure 1: Hetero-Associative run with smallest images.

```
>> localizacionKSFAMAA
Creando matrices de entrada y salida.
Elapsed time is 0.003120 seconds.
Calculando Mxz.
Elapsed time is 0.593087 seconds.
Calculando Wzya.
Elapsed time is 0.013334 seconds.
Calculando localizacion walk 2.
Elapsed time is 7.779774 seconds.
Calculando localizacion walk 3.
Elapsed time is 5.628681 seconds.
Calculando localizacion walk 4.
Elapsed time is 5.091582 seconds.
Calculando localizacion walk 5.
Elapsed time is 5.044541 seconds.
Calculando localizacion walk 6.
Elapsed time is 4.827368 seconds.
Calculando la posición devuelta.
Elapsed time is 1.303573 seconds.

tTotal =

   30.2852
```

Figure 2: Auto-Associative run with smallest images.

different. For instance, several views of the first segment are recognised as members of the last segment (maybe due the windows in the upper part of the image). Also, the sixth and seventh positions are completely missed.

References

[1] Peter Sussner and Estevão Esmi. An introduction to the Kosko Subsethood FAM. In Emilio Corchado, Manuel Graña Romay, and Alexandre Manhaes Savio, editors, *Hybrid Artificial Intelligence Systems*, volume 6077 of *Lecture Notes in Computer Science*, pages 343–350. Springer Berlin / Heidelberg, 2010.

[2] Ivan Villaverde, Alicia D'Anjou, and Manuel Graña. Morphological Neural Networks and vision based simultaneous localization and maping. *Integrated Computer-Aided Engineering*, 14(4)(14):355–363, 2007. IOS Press.

[3] Ivan Villaverde, Borja Fernandez-Gauna, and Ekaitz Zulueta. Lattice Independent Component Analysis for mobile robot localization. In Emilio Corchado, Manuel Graña Romay, and Alexandre Manhaes Savio, editors, *Hybrid Artificial Intelligence Systems*, volume 6077 of *Lecture Notes in Computer Science*, pages 335–342. Springer Berlin / Heidelberg, 2010.

[4] Ivan Villaverde, Manuel Graña, and Alicia D'Anjou. Morphological Neural Networks and vision based mobile robot navigation. In *Artificial Neural Networks - ICANN 2006*, volume 4131 of *Lecture Notes in Computer Science*, pages 878–887. Springer-Verlag, 2006.

[5] Ivan Villaverde, Manuel Graña, and Alicia D'Anjou. Morphological Neural Networks for localization and mapping. In *Proceedings of the IEEE International Conference on Computational Intelligence for Measurement Systems and Applications (CIMSA 06)*, pages 9–14. IEEE Press, 2006.

[6] Ivan Villaverde, Manuel Graña, and Alicia D'Anjou. Morphological Independence for landmark detection in vision based SLAM. In F. Sandoval, A. Prieto, J. Cabestany, and M. Graña, editors, *Proc. of IWANN 2007*, volume 4507 of *Lecture Notes in Computer Science*, pages 847–854. Springer-Verlag, 2007.

[7] Ivan Villaverde, Manuel Graña, and Jose Luis Jiménez. Lattice Independence and vision based mobile robot navigation. In Bruno Apolloni, Robert J. Howlett, and Lakhmi Jain, editors, *Knowledge-Based Intelligent Information and Engineering Systems, KES 2007*, volume 4693/2009 of *Lecture Notes in Artificial Inteligence*, pages 1196–1203. Springer-Verlag, 2007.