Robotic Color Image Segmentation by Means of Finite Mixture Models

Nicola Greggio $^{\oslash,\dagger}$- *IEEE Member*, Alexandre Bernardino † - *IEEE Member*
José Santos-Victor † - *IEEE Member*

$^{\oslash}$ *ARTS Lab - Scuola Superiore S.Anna, Polo S.Anna Valdera*
Viale R. Piaggio, 34 - 56025 Pontedera, Italy

† *Instituto de Sistemas e Robótica, Instituto Superior Técnico*
1049-001 Lisboa, Portugal

$^{\oslash}$*nicola.greggio@sssup.it* - †*ngreggio@isr.ist.utl.pt*

***Abstract*— Image segmentation for robots requires to be fast, in order to deal with ever more powerful processors. Moreover, it is assumed to be robust to environmental changes, such as light conditions. In this paper we propose the application of a couple of unsupervised learning algorithms for the estimation of the number of components and the parameters of a mixture model for image segmentation. These serve for the unsupervised identification of multiple different objects in a visual scene, such as for a subsequent localization and tracking. We compare our previous technique against the new one. The distinctive aspect of our new approach is related to a top-down hierarchical search for the number of components by means of a binary tree decision structure. This work analyzes both approaches, two previous work of ours, in terms of applicability to object detection for robotic applications. Besides, we propose the computational burden evaluation for the two algorithms.**

Index Terms - *Robotics, Computer Vision, Object Segmentation, Unsupervised Learning, Self-Adapting Expectation Maximization*

I. Introduction

Nowadays, computer vision and image processing are involved in many practical applications. The constant progress in hardware technologies leads to new computing capabilities, and therefore to the possibilities of exploiting new techniques. Image segmentation is a key low level perceptual capability in many robotics related application, as a support function for the detection and representation of objects and regions with similar photometric properties. Several applications in humanoid robots [1], rescue robots [2], or soccer robots [3] rely on some sort on image segmentation [4]. Additionally, many other fields of image analysis depend on the performance and limitations of existing image segmentation algorithms: video surveillance, medical imaging and database retrieval are some examples [5], [6].

A. Related Work

Two main principal approaches for image segmentation are adopted: Supervised and unsupervised. The latter one is the one of most practical interest. It may be defined as the task of segmenting an image in different regions based on some similarity criterion among each region's pixels.

Several techniques have been proposed in the literature for unsupervised learning, from Kohonen maps [7], Growing Neural gas [8], [9], k-means [10], to Independent component analysis [11], [12], etc. Particularly successful is the Expectation Maximization algorithm applied to finite mixture models. Fitting a mixture model to the distribution of the data is equivalent, in some applications, to the identification of the clusters with the mixture components [13].

One of the most widely used distributions is the normal, or Gaussian, distribution. The normal distribution can be used to describe, at least approximately, any variable that tends to cluster around the mean. If data is generated by a mixture of Gaussians, the clustering problem will reduce to the estimation of the number of Gaussian components and their parameters. Expectation-Maximization (EM) algorithm is well known and attractive approach for learning the parameters of mixture models [13], [14]. It always converges to a local optimum [15], especially for the case of Normal mixtures [13], [16]. However, it also presents some drawbacks. For instance, if requires the *a-priori* specification of the model order, namely, the number of components and its results are sensitive to initialization. The selection of the right number of components is a critical issue. The more components there are within the mixture, the better the data fit will be. Unfortunately, increasing the number of components will lead to data overfitting and to increase in the computational burden. Therefore, finding the best compromise between precision, generalization and speed is an essential concern. A common approach to address this compromise is to try different hypothesis for the number of components, and then selecting the best model according to some appropriate model selection criteria.

Different techniques can be used to select the best number of components in a mixture distribution. These can be divided into two main classes: *off-line* and *on-line* techniques.

The first ones evaluate the best model by executing independent runs of the EM algorithm for many different initializations, and evaluating each estimate with criteria that penalize complex models (e.g. the Akaike Information Criterion (AIC) [17], the Schwarz's Bayesian Information Criterion [18], the Rissanen Minimum Description Length (MDL) [19], and Wallace and Freeman Minimum Message Length (MML) [20]). All of these criteria, in order to be effective, have to be evaluated for every possible number of models under comparison. Therefore, it is obvious that, for having a sufficient search range the complexity goes with the number of tested models as well as the model parameters.

The second ones start with a fixed set of models and sequentially adjust their configuration (including the number of components) based on different evaluation criteria. Pernkopf and Bouchaffra proposed a Genetic-Based EM Algorithm capable of learning gaussians mixture models [21]. They first selected the number of components by means of the minimum description length (MDL) criterion. A combination of genetic algorithms with the EM has also been explored. Simulating annealing has also been explored as a possible solution for mixture selection, with Ueda and Nakano who proposed the deterministic annealing (DAEM), in which a modified posterior probability parametrized by *temperature* is derived to avoid local maxima [22]. Ueda *et Al.* proposed a split-and-merge EM algorithm (SMEM) to alleviate the problem of local convergence of the EM method [23].

Greedy algorithms take part within the second class of unsupervised classification techniques. They are characterized by making the locally optimal choice at each stage with the hope of finding the global optimum. Applied to the EM algorithm, they usually start with a single component (therefore side-stepping the EM initialization problem), and then increase their number during the computation. At the time, no precise solution has been posted to address this drawback. In 2002 Vlassis and Likas introduced a greedy algorithm for learning Gaussian mixtures [24]. They start with a single component covering all the data. Then they split an element and perform the EM locally for optimizing only the two modified components. Nevertheless, the total complexity for the global search of the element to be split $O(n^2)$. Subsequently, Verbeek et al. developed a greedy method to learn the gaussians mixture model configuration [25]. Their search for the new components is claimed to take $O(n)$, while our recursive search by means of the binary tree only $O(\log n)$.

B. *Our contribution*

We want to find a procedure for the unsupervised identification of multiple different objects in a visual scene, for a further localization and tracking. Therefore, we first need to segment the color image in a unsupervised way in order to detect the different targets and distinguishing them from the background. Then, we need to identify them. We decided to use Gaussian mixture models due to their accuracy and general applicability. Besides, in order to sidestep the shortcoming of high computational burden together with the need of the $a-priori$ decision of the number of components, we opted for a greedy self-organizing approach.

However, greedy algorithms mostly (but not always) fail to find the globally optimal solution, because they usually do not operate exhaustively on all the data. Our new technique tries to overcome this limitation by using a binary tree for deciding which component has to be replicated in an exhaustive way. The optimization of the parameters is done with expectation maximization (EM) and the search for the best number of parameters is done in a top-down manner, by starting with a single component and progressively adapting the mixture by adding new elements according to a binary tree structure. We compare a previous greedy algorithm we developed in [26], and then refined in [27], versus our new technique, presented in [28]. The restriction of the previous approach relies on the excessive number of input parameters to be tuned before the computation, and the heuristic stopping criterion. The latter may cause that more components than those effective necessary may be employed, resulting in an excessive mixture complexity and long computation.

C. *Outline*

In sec. II we introduce the analyzed algorithms. Besides, in sec. II-F and II-G we propose a computational complexity analysis. Then, in sec. III we describe our experimental set-up for testing the validity of our new technique and we compare our results against our previous alternative. Finally, in sec. V we conclude.

II. The Finite Mixture Learning Algorithms

In this section we provide a description about the differences between the two approaches. Since now, we will refer to these as:

- FASTGMM: The previous approach [27];
- FSAEM: The new algorithm [28].

In the following we present a brief overall description of both approaches, and a deeper analysis of the adding a new component process, together with the stopping criterion.

A. *FASTGMM Overall Description*

The basic idea is to incrementally estimate the mixture parameters and the number of components simultaneously. This algorithm starts with a single component and only increments its number as the optimization procedure progresses. The number of components is incremented at certain stages of the optimization procedure but the values of the mixture parameters are incrementally changed and not reinitialized.

The key issue of our technique is looking whether one or more Gaussians are not increasing their own likelihood during optimization.

For this algorithm we need to introduce a state variable related to the state of the gaussian component:

- Its age, that measures how long the component's own likelihood does not increase significantly;

Then, the split process is controlled by the following adaptive decision thresholds:

- One adaptive threshold Λ_{TH} for determining a significant increase in likelihood;
- One adaptive threshold A_{TH} for triggering the split process based on the component's own age;
- One adaptive threshold ξ_{TH} for deciding to split a gaussian based on its area.

It is worth noticing that even though we consider three thresholds to tune, all of them are adaptive, and only require a coarse initialization.

However, one of the biggest limitations is that this is suitable only for Gaussian mixtures, and not for generic ones.

B. FSAEM Overall Description

This algorithm starts with a single component. Then, by following a binary tree structure new classes are added by means of replicating existing ones, once a time. Subsequently, our modified EM algorithm is run in order to achieve the current mixture best configuration [28]. Furthermore, the cost function is evaluated in order to decide whether keeping or discarding the current mixture. In the first case, the binary tree will be updated with the new solution. When a new mixture element is added, it will become a child together with the original one. Therefore, within our representation, its father dies, and only the two children survive. Otherwise, in the second case (i.e. when the new mixture configuration is discarded), the previous mixture will be restored as a starting point for a new component replication, and that node will never be proposed to have children anymore. Finally, when there will no node eligible to have children (i.e. when all the combinations have been tried), the algorithm terminates.

It is worth noticing that this technique can be applied to any mixture, rather than Gaussian ones, merely.

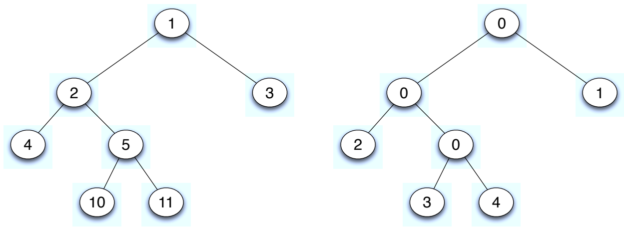
C. Adding a new mixture class

One the one hand, FASTGMM splits the component with highest covariance matrix determinant (and therefore that covering the highest number of data), when this overcome a threshold. The latter one follows a decreasing law, in order to avoid stationary situations. However, both the splitting process itself is ill-posed (there are infinite solutions [23]), and the chosen decision criterion is arbitrary. Finally, the law leading the thresholds variation is heuristic.

On the other hand, FSAEM employs a replication process rather than a splitting procedure. The new component will be the exact copy of that candidate to be replicated. This allows to a unique solution, with the only exception of a variation in the mean of these components (in order to not have them exactly superimposed, situation not allowing the EM procedure to escape the current local minimum [28]). Besides, instead of relying on empirical thresholds for determining the most suitable component to be replicated, all the classes are replicated in sequence, in order to exploit all the mixture combinations possibilities. This is achieved thanks to the binary tree decision structure, showed in Fig. 1.

D. Model Selection Criterion: Minimum message length (MML)

Since no merging or annihilating operation ha been envisioned, it is worth being sure about a new component insertion. FSAEM integrates a derivation of the MML criterion in order to evaluate whether the new mixture configuration (i.e. that after the last component replication) provides a better data description in terms of the MML evaluation. In the affirmative case, the current mixture configuration is kept, a new replication is performed following the updated binary tree. Otherwise, the old mixture configuration is resumed (therefore voiding the last component insertion). The binary tree is updated in order to not replicate the same component anymore once the correspondent mixture has been discarded.



(a) Binary tree indexes (b) Binary tree contents

Fig. 1. Binary tree mixture distribution structure example: On the left the binary tree indexes representation, used as decision structure; on the right the contents of the binary tree used as look up table. The array correspondent representation is: $[0, 0, 1, 2, 0, 3, 4]$. The 0 contents are relative to the parents have been eliminated after creating their children.

We adopted the minimum message length (MML) criterion developed in [29], which formulation is:

$$\bar{\vartheta}_{opt} = arg \min_{\bar{\vartheta}} \Big\{ -L\left(X|\bar{\vartheta}\right) + \frac{N}{2}\sum_{i=1}^{c}\ln\left(\frac{n\cdot w_i}{12}\right) + \frac{c}{2}\left(N+1-\ln 12n\right)\Big\} \tag{1}$$

In eq. (1):

- $-L\left(X|\bar{\vartheta}\right)$ is the log-likelihood of the whole distribution;
- N is the number of parameters specifying each component (e.g. in case of a normal distribution they are the mean and the covariance matrix, counted as 1 parameter for each input dimension for the mean and 1 for each the covariance among each dimension, resulting in $N = d + (d+1) * d/2$);
- c is the number of mixture components;
- n is the number of input data;
- w_i is the *a-priori* class, or component, probability, therefore resulting $n \cdot w_i$ the number of components of the class i.

E. Stopping criterion

On the one side, FASTGMM relies on an empirical stopping criterion. This, together with the splitting procedure may result in having the best mixture data description.

On the other side, FSAEM stops when all the mixture replication combinations have been exploited. Besides, the MML criterion described in sec. II-D ensure that when the insertion of a new component do not improve the data description, this is discarded.

F. FASTGMM Computational complexity evaluation

Taking D as the input dimension, the computational burden of each iteration is:

- the original EM algorithm takes $O(N \cdot D \cdot nc)$ for each step, for a total of $O(4 \cdot N \cdot D \cdot nc)$ operations;
- the algorithm takes $O(nc)$ for evaluating all the single Gaussians log-likelihood;
- the split operation requires $O(D)$.
- the others take $O(1)$.
- the optional procedure of optimizing the selected mixture takes $O(4 \cdot N \cdot D \cdot nc)$, being the original EM.

Therefore, the original EM algorithm takes:

- $O(4 \cdot N \cdot D \cdot nc)$, while our algorithm adds $O(D \cdot nc)$ on the whole, or $O(4 \cdot N \cdot D \cdot nc)$, giving rise to $O(4 \cdot N \cdot D \cdot nc) + O(D \cdot nc) = O(4 \cdot N \cdot D \cdot nc + D \cdot nc) = (nc \cdot D \cdot (4N+1))$ in the first case;
- $2 \cdot O(4 \cdot N \cdot D \cdot nc) + O(D \cdot nc) = O(8 \cdot N \cdot D \cdot nc + D \cdot nc) = (nc \cdot D \cdot (8N+1))$ in the second case, with the optimization procedure.

G. *FSAEM Computational complexity evaluation*

The computational burden of each iteration of the algorithm is:

1. The original EM algorithm takes $O(k \cdot D \cdot nc)$ for the whole mixture log-likelihood evaluation, $O(k \cdot D \cdot nc)$ for the E-step, and approximatively the same amount for the M-step, and $O(nc)$ for the prior re-estimation, therefore resulting in $O(k \cdot D \cdot (nc+1))$ for each step;
2. The binary tree, if complete, takes $O(\log nc)$ for the insertion, the selection, and the removal operation.
3. Our algorithm takes $O(nc)$ for evaluating all the components possible ill-conditioning (this would result in evaluating the covariance matrix determinants in case of Gaussian components, which requires $O(D!)$ additionally, and another $O(D)$ for replicating along all the input dimensions whether necessary);
4. Our replication operation requires $O(D!)$ for the SVD decomposition, plus other $O(nc)$ for the components reallocation.

This gives rise to the total computation: $O(k \cdot D \cdot (nc+1)) + O(\log nc) + O(nc) + O(D!) + O(D) + O(D!) + O(nc) + O(1) = O(k \cdot (D+1) \cdot (nc+3) + \log nc) + 2O(D!))$.

Considering that usually $D << k$ and $nc << k$, we can assume that the computational complexity does not differ considerably between the general case and the application to the mixtures of Gaussians. Therefore, we assume that the total computational burden goes with $O(k \cdot D \cdot (nc+1) + \log nc)$.

III. Experiments

Due to its robotic application, we tested our algorithm on camera images taken from our robotic platform, the iCub. The iCub cameras are two DragonFly with VGA resolution and 30 fps speed. The acquired images have a resolution of $320x240$. We will segment these images by means both algorithms, in order to compare them both in terms of accuracy and, processing speed.

We segmented the images as 5-dimensional input in the (R,G,B) space and (x,y), i.e. a generic input point was of kind: $p \in (R, G, B, x, y)$. Then, we applied a simple Gaussian blur and the connected component labeling.

The color image segmentation results are shown in Fig. III. We highlight the blob findings, centering them on their mean and surrounded by their covariance matrix (represented as an ellipse in 2D, green for the binary images and red for the color ones). Input $(1), (2), (3)$ have been chosen to have a considerably lower light contrast than the last two ones, $(4), (5)$. For each row, the first three images on the left, for each column, are obtained with the FASTGMM algorithm, while the other three on the right with the FSAEM approach. Here, the original image, the mixture learning segmented image, and the connected component resultant image are shown, respectively.

We made the same experiments with the same input images both with the RGB color segmentation and the HSV one. Then, we also show the results for the same images in the (H, S, V, x, y) representation in Fig. IV.

Finally, we also measure the elapsed time of both algorithms. This has been performed by means of the *time profile* matlab function. However, although this is claimed to be not sensitive to the other running applications (it counts the number of float operations), we experimented that this is not true, indeed. Therefore, this test cannot be considered as precise ad exhaustive, but indicative, only.

IV. Discussion

The accuracy of the image segmentation greatly depends on the number of employed mixture components. The higher it is, the more accurate the image reconstruction will be. However, using too many components may lead to an overfitting of the input set, together with an excessive increase of the computational burden. Therefore, finding the $best$ compromise is a must. With FSAEM the $best$ compromise is decided by the MML criterion of eq. (1).

In tab. I the results of FASTGMM and FSAEM applied to the selected images with the RGB color segmentation are shown, while in tab. II there are the results for the HSV color space. In each table we report:

- The number of detected components;
- The actual number of components, i.e. that of the generation mixture;
- The number of total iterations;
- The elapsed time (this is relative to the image segmentation only, and not to the connected component labeling);
- The percentage difference in time for the new algorithm $(Time_{FSAEM})$ with respect to the old formulation $(Time_{FSAEM})$, evaluated as $\frac{Time_{FSAEM} - Time_{FSAEM}}{Time_{FSAEM}} \cdot 100$;
- The final log-likelihood;
- The percentage difference in final log-likelihood for the new algorithm versus the previous approach.

A. RGB *versus* HSV *color space*

Comparing the results shown in in Fig. 2 and in Fig. 3 it can be seen that generally both algorithm perform better under

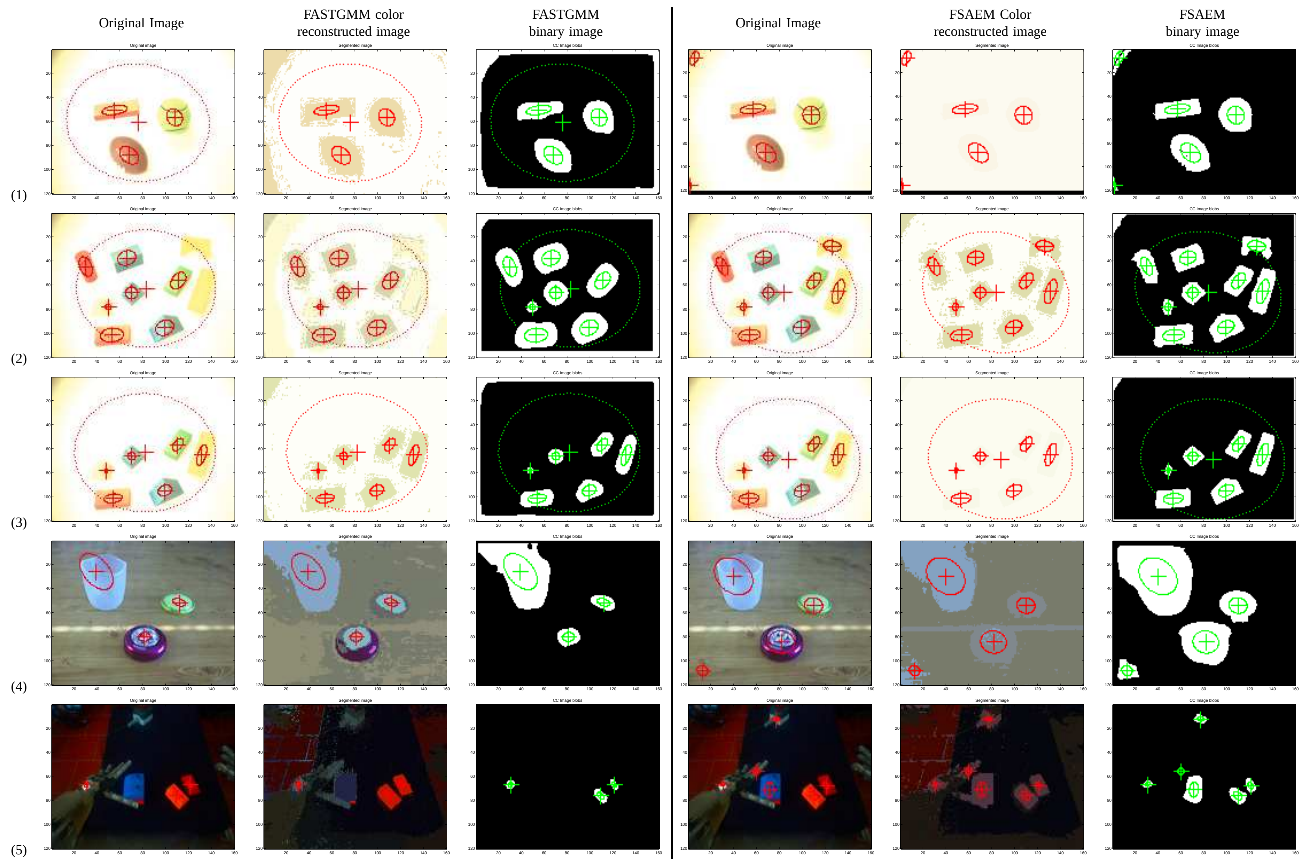


Fig. 2. Image segmentation in the RGB color space. FASTGMM results are shown on the left, while FSAEM outcomes are represented on the right. For each image, there is a subset composed by the original image, the color image reconstruction, and the binary image labeled with the connected components, respectively. Each image contains the objects of interest highlighted in red for the color output, and green for the binary one superimposed. The objects have been marked with their mean and covariance, represented as a regular ellipse in 2D, obtained with the connected components labeling.

RGB Color Segmentation Results							
Image number	Algorithm	Detected number of Gaussian components	Number of iterations	Elapsed Time [s]	Percentage time difference	Final log-likelihood	Percentage difference on log-likelihood
(1)	FASTGMM	2	38	1.244413	-37.05064155	-278959.4334	-55.54316696
	FSAEM	2	26	1.705476		-433902.3373	
(2)	FASTGMM	3	21	0.933944	-129.1852616	-313336.7632	-12.9245556
	FSAEM	2	38	2.140462		-353834.1474	
(3)	FASTGMM	2	62	2.244048	-101.9897524	-285480.1774	-54.44871057
	FSAEM	2	121	4.532747		-440920.453	
(4)	FASTGMM	11	332	46.743716	95.54896534	-403884.4389	-2.432889246
	FSAEM	3	31	2.080579		-413710.5	
(5)	FASTGMM	11	285	34.017004	82.4117315	-387677.3402	-4.231222296
	FSAEM	4	106	5.983002		-404080.8302	

TABLE I

EXPERIMENTAL RESULTS ON REAL ROBOTIC IMAGES. SEGMENTATION PERFORMED IN THE RGB COLOR SPACE.

the HSV color segmentation than the RGB one. It is well-known that the HSV representation is more robust to light changes, for instance. Images $1, 4$, and 5 are better segmented in terms of number of effective objects detected. Specifically, input 3 gives rise to better results for the FSAEM approach, while resulting less precise with the FASTGMM segmentation. However, image 2 is confused for both algorithms within the HSV color space. Besides, image 4 is better segmented by FASTGMM in both color spaces. Finally, it is wort noticing that segmenting within HSV color space requires less iterations.

Comparing the segmented images, it may seem at a first glance that FASTGMM performs better than FSAEM. This is because generally FSAEM uses less mixture components for representing the image (this is clear both in Fig. 2 and in Fig. 3, where the FSAEM reconstructed images - the middle column on the right set - are less precise). However, the final results in terms of object recognition may seem similar. Nevertheless, FSAEM is capable of extracting the important features as well as FASTGMM, while requiring less computational resources.

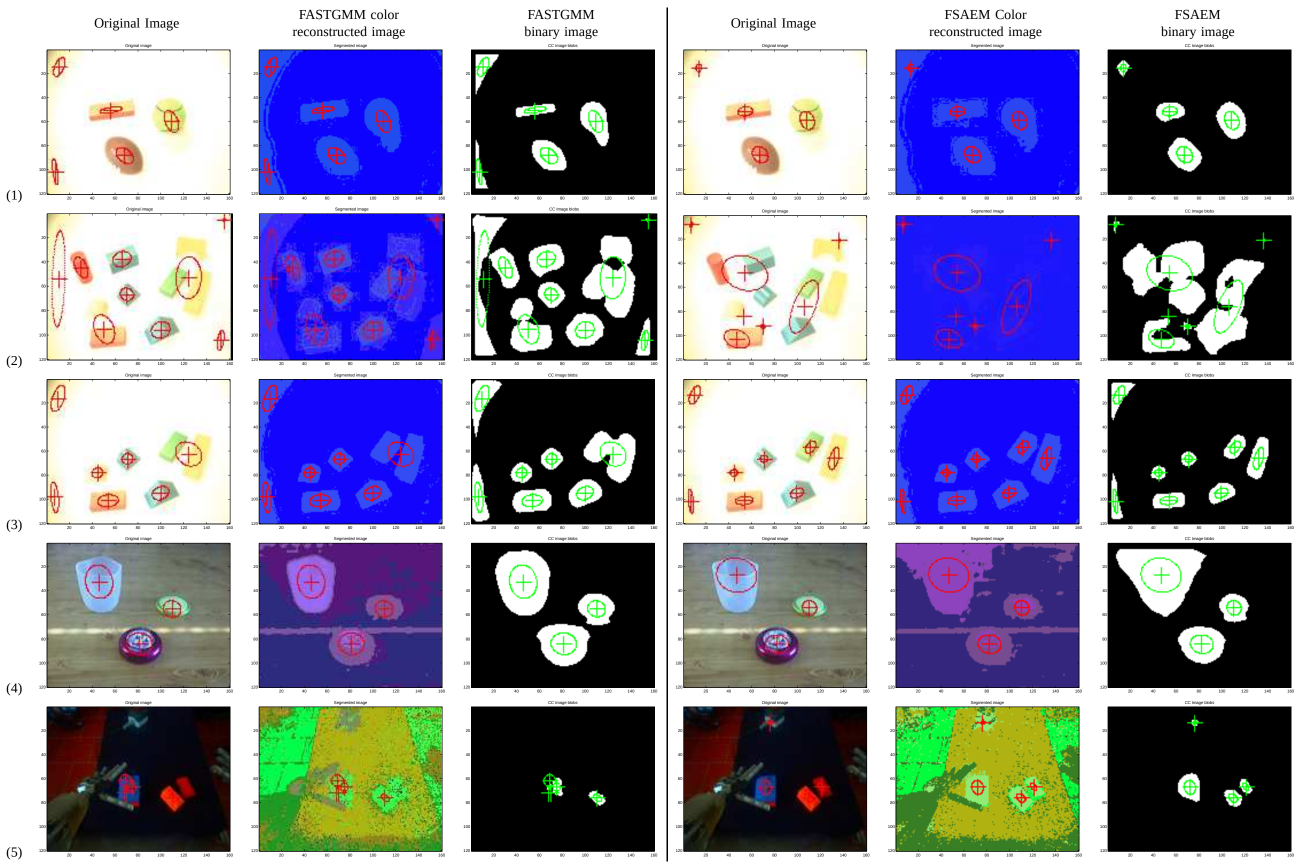


Fig. 3. Image segmentation in the HSV color space. FASTGMM results are shown on the left, while FSAEM outcomes are represented on the right. The input images are the same as in Fig. III, and so are the output subsets organization. As for the images in the previous figure, the recognized objects of interests have been highlighted and superimposed to the original pictures.

HSV Color Segmentation Results								
Image number	Algorithm	Detected number of Gaussian components	Number of iterations	Elapsed Time [s]	Percentage time difference	Final log-likelihood	Percentage difference on log-likelihood
(1)	FASTGMM	2	25	0.755909	-171.7325763	-338069.989677	-33.18992954
	FSAEM	2	48	2.054051		-450275.181	
(2)	FASTGMM	3	22	0.877972	-288.2384632	-340969.8623	-17.78399023
	FSAEM	2	76	3.408625		-401607.9093	
(3)	FASTGMM	2	25	0.754844	-117.6407311	-352858.824	-29.54999667
	FSAEM	2	33	1.642848		-457128.5947	
(4)	FASTGMM	7	80	6.484346	0.1370223	-455537.6481	-3.387910601
	FSAEM	4	117	6.475461		-470970.8564	
(5)	FASTGMM	10	274	48.567681	93.45860059	-435224.6992	-2.347807229
	FSAEM	3	56	3.177006		-445442.9361	

TABLE II

EXPERIMENTAL RESULTS ON REAL ROBOTIC IMAGES. SEGMENTATION PERFORMED IN THE HSV COLOR SPACE.

B. *Log-Likelihood*

Fig. 4 shows the final log-likelihood of both approaches. Here it is possible to see when the FASTGMM or FSAEM add a component, i.e. corresponding to the spikes that lower the curve. Then, when the log-likelihood does not increase any-more significantly the FASTGMM computation stops, while FSAEM stops when there are no more components to be replicated.

C. *Cost Function*

Finally, we provide the cost function evolution of the MML information criterion as function of the number of components for the FSAEM algorithm (FASTGMM do not provide an information criterion). Fig. 5 shows a couple of examples, namely those of the image no. 4 and, both present in Fig. 2 and in Fig. 3.

D. *Elapsed time*

Generally FSAEM performs faster than FASTGMM. The results in tab. I and in tab. II demonstrate that generally

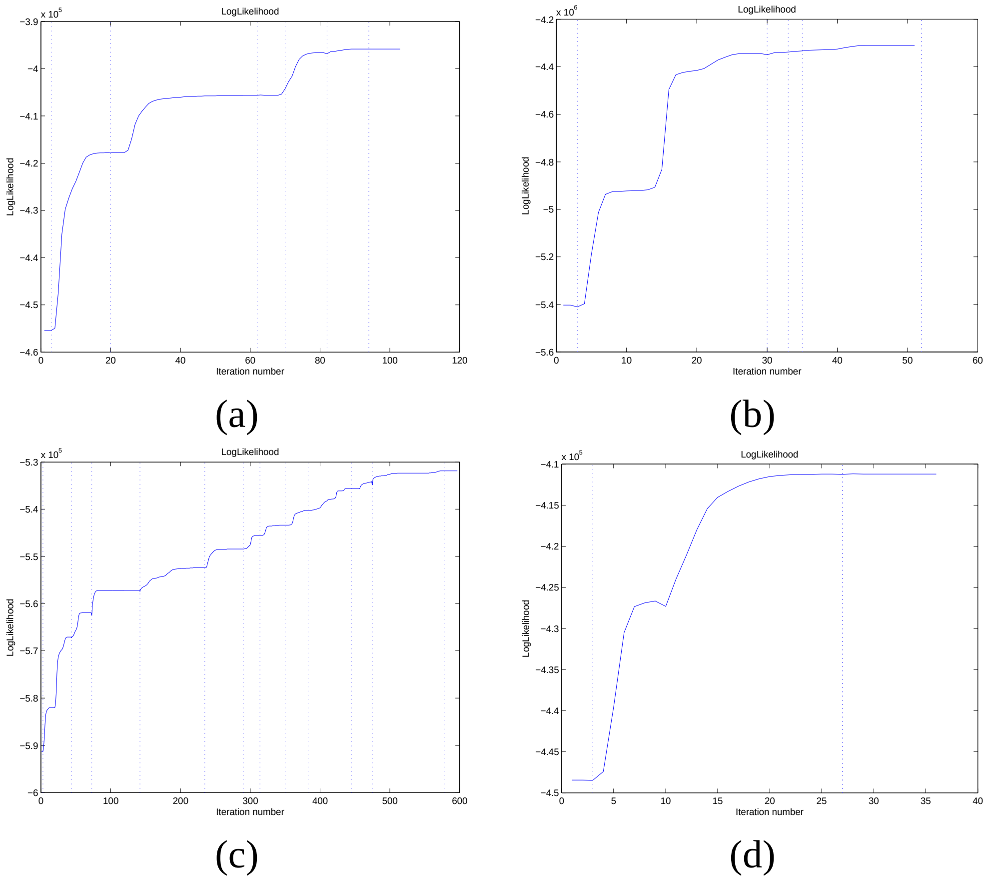


Fig. 4. The final log-likelihood evolution as function of the number of iterations of two different kinds of input data used within the experiments: The image (4) - (a) FASTGMM and (b) FSAEM, and the image (5) - (c) FASTGMM and (d) FSAEM.

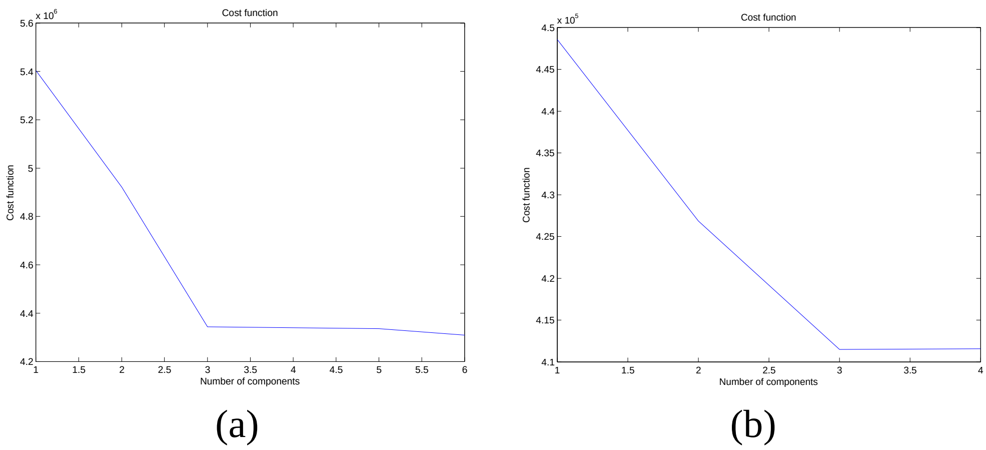


Fig. 5. The cost function evolution as function of the number of components of two different kinds of input data used within the experiments for the FSAEM algorithm: (a) The image (4), (b) and the image (5).

FSAEM runs faster. Indeed, for the first three images the elapsed time can be considered comparable, if not double for FSAEM with respect to FASTGMM. Nevertheless, the elapsed time is so low that it makes this comparison not accurate. As a confirmation the last two images, 4 and 5, that require a longer computation, advantage FSAEM.

Our explanation relies both on the splitting procedure and the stopping criterion, which are more heuristic in FASTGMM. The FSAEM replication process, that exploit all the mixture classes by means of the binary tree, may be the reason for the slower computation with the first three images. However, this also provides a better accuracy in the choice of the component to be replicated, and this in long term gives rise to a fewer EM iterations, and then to a lower elapsed time.

Moreover, FSAEM does not depend on the FASTGMM heuristic parameters and thresholds.

Finally, it can be argued that the approaches take always more than 750 ms to process each image. In a robotics framework, this time could be excessively high (it is near to 1-1.2 fps) and it could be a source of security problems. However, the algorithms have been implemented under Matlab herein, which is a interpreted language rather than C++, which is a compiled one. This means that the first one will result much slower than the second one. Generally, and this is for the iCub repository software, for this reason robotics applications are written in C/C++, and not in Matlab. We choose the latter for a sake of practicality.

V. Conclusion

In this paper we compared two unsupervised algorithms that on-line learns a finite mixture model from multivariate data, presented in a couple of previous work of ours. We briefly summarized the algorithms, with respect to their more salient characteristics. Besides, we also presented an accurate computational complexity analysis of both approaches. Finally, we discuss our results, arguing for a more general validity of the more recent approach.

Acknowledgements

This work was supported by the European Commission, Project IST-004370 RobotCub and FP7-231640 Handle, and by the Portuguese Government - Fundação para a Ciência e Tecnologia (ISR/IST pluriannual funding) through the PIDDAC program funds and through project BIO-LOOK, PTDC / EEA-ACR / 71032 / 2006.

References

[1] L. Montesano, M. Lopes, A. Bernardino, and J. Santos-Victor, "Learning object affordances: From sensory motor maps to imitation," *IEEE Trans. on Robotics*, vol. 24, no. 1, 2008.

[2] S. Carpin, M. Lewis, J. Wang, S. Balakirsky, and C. Scrapper, "Bridging the gap between simulation and reality in urban search and rescue"," in *Robocup 2006: Robot Soccer World Cup X*, 2006.

[3] N. Greggio, G. Silvestri, E. Menegatti, and E. Pagello, "Simulation of small humanoid robots for soccer domain." *Journal of The Franklin Institute - Engineering and Applied Mathematics*, vol. 346, no. 5, pp. 500–519, 2009.

[4] M. Vincze, "Robust tracking of ellipses at frame rate," *Pattern Recognition*, vol. 34, pp. 487–498, 2001.

[5] J. G. G. Dobbe, G. J. Streekstra, M. R. Hardeman, C. Ince, and C. A. Grimbergen, "Measurement of the distribution of red blood cell deformability using an automated rheoscope," *Cytometry (Clinical Cytometry)*, vol. 50, pp. 313–325, 2002.

[6] H. Shim, D. Kwon, I. Yun, and S. Lee, "Robust segmentation of cerebral arterial segments by a sequential monte carlo method: Particle filtering," *Computer Methods and Programs in Biomedicine*, vol. 84, no. 2-3, pp. 135–145, December 2006.

[7] T. Kohonen, "Analysis of a simple self-organizing process." *Biological Cybernetics*, vol. 44, no. 2, pp. 135–140, 1982.

[8] B. Fritzke, "A growing neural gas network learns topologies." *Adv ances in Neural Inform ation Processing Systems 7 (NIPS'94), MIT Press, Cambridge MA*, pp. 625–632, 1995.

[9] J. Holmström, "Growing neural gas - experiments with gng, gng with utility and supervised gng," 2002.

[10] J. B. MacQueen, "Some methods for classification and analysis of multivariate observations." *Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability.*, pp. 281–297, 1967.

[11] P. Comon, "Independent component analysis: a new concept?" *Signal Processing, Elsevier*, vol. 36, no. 3, pp. 287–314, 1994.

[12] A. Hyvärinen, J. Karhunen, and E. Oja, "Independent component analysis," *New York: John Wiley and Sons*, vol. ISBN 978-0-471-40540-5, 2001.

[13] G. McLachlan and D. Peel, "Finite mixture models." *John Wiley and Sons*, 2000.

[14] H. Hartley, "Maximum likelihood estimation from incomplete data." *Biometrics*, vol. 14, pp. 174–194, 1958.

[15] A. Dempster, N. Laird, and D. Rubin, "Maximum likelihood estimation from incomplete data via the em algorithm," *J. Royal Statistic Soc.*, vol. 30, no. B, pp. 1–38, 1977.

[16] L. Xu and J. M., "On convergence properties of the em algorithm for gaussian mixtures," *Neural Computation*, vol. 8, pp. 129–151, 1996.

[17] Y. Sakimoto, M. Iahiguro, and G. Kitagawa, "Akaike information criterion statistics," *KTK Scientific Publisher, Tokio*, 1986.

[18] G. Schwarz, "Estimating the dimension of a model," *Ann. Statist.*, vol. 6, no. 2, pp. 461–464, 1978.

[19] J. Rissanen, "Stochastic complexity in statistical inquiry." *Wold Scientific Publishing Co. USA*, 1989.

[20] C. Wallace and P. Freeman, "Estimation and inference by compact coding," *J. Royal Statistic Soc. B*, vol. 49, no. 3, pp. 241–252, 1987.

[21] F. Pernkopf and D. Bouchaffra, "Genetic-based em algorithm for learning gaussian mixture models," *IEEE Trans. Patt. Anal. Mach. Intell.*, vol. 27, no. 8, pp. 1344–1348, 2005.

[22] N. Ueda and R. Nakano, "Deterministic annealing em algorithm," *Neural Networks*, vol. 11, no. 2, pp. 271–282, 1998.

[23] N. Ueda, R. Nakano, Y. Ghahramani, and G. Hiton, "Smem algorithm for mixture models," *Neural Comput*, vol. 12, no. 10, pp. 2109–2128, 2000.

[24] N. Vlassis and A. Likas, "A greedy em algorithm for gaussian mixture learning," *Neural Processing Letters*, vol. 15, pp. 77–87, 2002.

[25] J. Verbeek, N. Vlassis, , and B. Krose, "Efficient greedy learning of gaussian mixture models," *Neural Computation*, vol. 15, no. 2, pp. 469–485, 2003.

[26] N. Greggio, A. Bernardino, and J. Santos-Victor, "Image segmentation for robots: Fast self-adapting expectation maximization." *International Conference on Image Analysis and Recognition (ICIAR), Povoa de Varzim, Portugal, June 21-23*, 2010.

[27] N. Greggio, A. Bernardino, C. Laschi, P. Dario, and J. Santos-Victor, "Fast estimation of gaussian mixture models for image segmentation," *Machine Vision and Application*, Accepted for publication 2011.

[28] N. Greggio, A. Bernardino, C. Laschi, J. Santos-Victor, and P. Dario, "Unsupervised greedy learning of finite mixture models." *IEEE 22th International Conference on Tools with Artificial Intelligence (ICTAI 2010), Arras, France*, October 27-29 2010.

[29] A. Figueiredo and A. Jain, "Unsupervised learning of finite mixture models," *IEEE Trans. Patt. Anal. Mach. Intell.*, vol. 24, no. 3, 2002.