Image Segmentation for Robots: Fast Self-Adapting Gaussian Mixture Model

Nicola Greggio[1,2], Alexandre Bernardino[2], José Santos-Victor[2]

[1] ARTS Lab - Scuola Superiore S.Anna, Polo S.Anna Valdera
Viale R. Piaggio, 34 - 56025 Pontedera, Italy
[2] Instituto de Sistemas e Robótica, Instituto Superior Técnico - 1049-001 Lisboa, Portugal

ngreggio@isr.ist.utl.pt

Abstract. *Image segmentation is a critical low-level visual routine for robot perception. However, most image segmentation approaches are still too slow to allow real-time robot operation. In this paper we explore a new method for image segmentation based on the expectation maximization algorithm applied to Gaussian Mixtures. Our approach is fully automatic in the choice of the number of mixture components, the initialization parameters and the stopping criterion. The rationale is to start with a single Gaussian in the mixture, covering the whole data set, and split it incrementally during expectation maximization steps until a good data likelihood is reached. Singe the method starts with a single Gaussian, it is more computationally efficient that others, especially in the initial steps. We show the effectiveness of the method in a series of simulated experiments both with synthetic and real images, including experiments with the iCub humanoid robot.*

Keywords - image processing, unsupervised learning, self-adapting gaussians mixture, expectation maximization, machine learning, clustering

1 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, for instance considered to time consuming only a few years ago. 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].

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. Particularly interesting is the Expectation Maximization algorithm applied to gaussians mixtures which allows to model complex probability distribution functions. 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 [7].

Expectation-Maximization (EM) algorithm is the standard approach for learning the parameters of the mixture model [8]. It is demonstrated that it always converges to a local optimum. However, it also presents some drawbacks. For instance, EM requires an *a-priori* selection of model order, namely, the number of components (M) to be incorporated into the model, and its results depend on initialization. The more gaussians there are within the mixture, the higher will be the total log-likelihood, and more precise the estimation. Unfortunately, increasing the number of gaussians will lead to overfitting the data and it increases the computational burden. Therefore, finding the best compromise between precision, generalization and speed is a must. A common approach to address this compromise is trying different configurations before determining the optimal solution, e.g. by applying the algorithm for a different number of components, and selecting the best model according to appropriate criteria.

1.1 Related Work

Different approaches can be used to select the best number of components. 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) [9] and the Rissanen Minimum Description Length (MDL) [10]). These, in order to be effective, have to be evaluated for every possible number of models under comparison. Therefore, it is clear that, for having a sufficiently exhaustive search the complexity goes with the number of tested models, and 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 [11]. 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 been explored.

An example are the greedy algorithms. 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. The first formulation was originally proposed in 2000, by Li and Barron [12]. Subsequently, in 2002 Vlassis and Likas introduced a greedy algorithm for learning Gaussian mixtures [13]. Nevertheless, the total complexity for the global search of the element to be splitted $O(n^2)$. Subsequently, Verbeek et al. developed a greedy method to learn the gaussians mixture model configuration [14]. However, the big issue in these kind of algorithm is the insertion selection criterion: Deciding when inserting a new component and how can determine the success or failure of the subsequent computation.

Ueda *et Al.* proposed a split-and-merge EM algorithm to alleviate the problem of local convergence of the EM method [15]. Subsequently, Zhang *et Al.* introduced another split-and-merge technique [16]. Merge and split criterion is efficient in reducing number of model hypothesis, and it is often more efficient than exhaustive, random or genetic algorithm approaches. Particularly interesting is the method proposed by Figueiredo and Jain, which uses only merge operations, therefore starting with a high number of mixture parameters, merging them step by step until convergence [17], making then no use of splitting operations. This method can be applied to any parametric mixture where the EM algorithm can be used. However, the higher the number of mixture components is, the more expensive the computation will be. Therefore, since the idea of Figueiredo and Jain starts with a very high number of mixture components, greatly slowing the computation from the first steps.

1.2 Our contribution

In this paper, we propose an algorithm that automatically learns the number of components as well as the parameters of the mixture model. The particularly of our model is that we approach the problem contrariwise than Figueiredo and Jain did, i.e. by starting from only one mixture component instead of several ones and progressively adapting the mixture by adding new components when necessary. Therefore, in order to accomplish this we needed to define a precise split and stopping criteria. The first is essential to be sure to introduce a new component (and therefore new computational burden) only when strictly necessary, while the second one is fundamental to stop the computation when a good compromise has been obtained (otherwise the algorithm will continue to add components indefinitely, until the maximum possible likelihood is obtained). Our formulation guarantees the following advantages. First, it is a deterministic algorithm; we avoid the different possibilities in the components initializations that greatly affect the standard EM algorithm, or any EM technique that starts with more than one component, by using a unique initialization independently from the input data. Therefore, by applying the same algorithm to the same input data we will get always the same results, Second, it is a low computationally expensive technique - in fact, new components will be added only when strictly necessary.

1.3 Outline

The paper is organized as follows. In sec. 2 we introduce the proposed algorithm. Specifically, we describe the insertion of a new gaussians in sec. 2.4, the initializations in sec. 2.2, the decision thresholds update rules in sec. 2.5, and the stopping criterion 2.6. Furthermore, in sec. 3 we describe our experimental set-up for testing the validity of our new technique and the results. Finally, in sec. 4 we conclude and propose directions for future work.

2 FASTGMM: FAST Self-Adapting Gaussian Mixture Model

We distinguish two main important features for our algorithm: The splitting criterion and the stopping criterion. The key issue of our algorithm is looking whether one or

more gaussians are not increasing their own likelihood during optimization. In other words, if they are not participating in the optimization process, they will be split into two new gaussians. We will introduce a new concept related to the state of a gaussians component:

– Its age, that measures how long the component's own likelihood does not increase significantly (see sec. 2.1);

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

– One adaptive threshold L_{TH} for determining a significant increase in likelihood (see sec. 2.5);
– One adaptive threshold Age_{TH} for triggering the merge or split process based on the component's own age (see sec. 2.5);
– One adaptive threshold S_{TH} for deciding to split a gaussians based on its area (see sec. 2.4).

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

These parameters will be fully detailed within the next sections.

2.1 FASTGMM Formulation

Our algorithm's formulation can be summarized within three steps:

– Initializing the parameters;
– Adding a gaussians;
– Updating decision thresholds.

Each mixture component i is represented as follows:

$$\bar{\vartheta}_i = \varrho(w_i, \bar{\mu}_i, \Sigma_i, \xi_i, \Lambda_{last(i)}, \Lambda_{curr(i)}, a_i) \tag{1}$$

where w_i is the *a-priori* probabilities of the class, $\bar{\mu}_i$ is its mean, Σ_i is its covariance matrix, ξ_i its *area*, $\Lambda_{last(i)}$ and $\Lambda_{curr(i)}$ are its last and its current log-likelihood value, and a_i its *age*. Here, we define two new elements, the area (namely, the covariance matrix determinant) and the age of the gaussians, which will be described later.

During each iteration, the algorithm keeps memory of the previous likelihood. Once the re-estimation of the vector parameter $\bar{\vartheta}$ has been computed in the EM step, our algorithm evaluates the current likelihood of each single gaussians as:

$$\Lambda_{curr(i)}(\vartheta) = \sum_{j=1}^{k} log(w_i \cdot p_i(\bar{x}^j)) \tag{2}$$

If a_i overcomes the age threshold Age_{TH} (i.e. the gaussians i does not increase its own likelihood for a predetermined number of times significally - over L_{TH}), the algorithm decides whether to split this gaussians or merging it with existing ones depedending on whether their own single area overcome S_{TH}.

Then, after a certain number of iterations the algorithm will stop - see sec. 2.6. The whole algorithm pseudocode is shown in Fig. 2.1.

Algorithm 2.1 Pseudocode

- Parameter initialization;
while (stopping criterion is not met) **do**
  $\Lambda_{curr(i)}$, evaluation, for $i = 0, 1, \ldots, c$;
  $L(\bar{\vartheta})$ evaluation;
  Re-estimate priors w_i, for $i = 0, 1, \ldots, c$;
  Recompute center $\bar{\mu}_i^{(n+1)}$ and covariances $\Sigma_i^{(n+1)}$, for $i = 0, 1, \ldots, c$;
  - Evaluation whether changing the gaussians distribution structure -
  for ($i = 0$ to c) ***do***
    if *(*$a_i > Age_{TH}$*)* ***then***
      if *(*$(\Lambda_{curr(i)} - \Lambda_{last(i)}) < L_{TH}$*)* ***then***
        $a_i += 1$;
        - General condition for changing satisfied; checking those for each gaussians -
        if *(*$\Sigma_i > S_{TH}$*)* ***then***
          if *(*$c < maxNumgaussians$*)* ***then***
            split gaussians $\rightarrow$ *split ;*
            $c += 1$;
            reset $S_{TH} \leftarrow \frac{S_{M-INIT}}{ng}$;
            reset $L_{TH} \leftarrow L_{INIT}$;
            reset $a_A, a_B \leftarrow 0$*, with A, B being the new two gaussians;*
            return
          end if
        end if
        $S_{TH} = S_{TH} \cdot (1 + \alpha \cdot \xi)$;
      end if
    end if
  end for
end while

2.2 Parameters initialization

At the beginning, S_{TH} will be automatically initialized to the Area of the covariance of all the data set - i.e. the determinant of the covariance matrix relative to the whole data set. The other decision thresholds will be initialized as follows:

$$\begin{aligned} L_{INIT} &= k_{LTH} \\ Age_{INIT} &= k_{ATH} \end{aligned} \tag{3}$$

with k_{LTH} and k_{ATH} (namely, the minimum amount of likelihood difference between two iterations and the number of iterations required for taking into account the lack of a likelihood consistent variation) relatively low (i.e. both in the order of 10, or 20). Of course, higher values for k_{LTH} and smaller for k_{ATH} give rise to a faster adaptation, however adding instabilities.

2.3 Gaussians components initialization

The algorithm starts with just only one gaussians. Its mean will be the whole data mean, while its covariance matrix will be those of the whole data set. Of course, one may de-

sire to start with more than one gaussians in case that *a-priori* the gaussians components of the data set are more than one, for sake of convergence speed. In that case means and covariances will be as follows.

2.4 Splitting a gaussian

If the covariance matrix determinant of the examined gaussians at each stage overcomes the maximum area threshold S_{TH}, then another gaussians is added to the mixture.

More precisely, the original gaussians with parameters $\bar{\vartheta}_{old}$ will be split within other two ones. The new means, A and B, will be:

$$\begin{aligned} \bar{\mu}_A &= \bar{\mu}_{old} + \frac{1}{2}(\Sigma_{i=j})^{1/2} \\ \bar{\mu}_B &= \bar{\mu}_{old} - \frac{1}{2}(\Sigma_{i=j})^{1/2} \quad i,j = \{1,2,\ldots,d\} \end{aligned} \tag{4}$$

where d is the input dimension.

The covariance matrixes will be updated as:

$$\Sigma_{A(i,j)} = \Sigma_{B(i,j)} = \begin{cases} \frac{1}{2}\Sigma_{old(i,j)}, & \text{if } i = j; \\ 0, & \text{othrewise.} \end{cases} \tag{5}$$

The *a-priori* probabilities will be

$$w_A = \frac{1}{2}w_{old} \quad w_B = \frac{1}{2}w_{old} \tag{6}$$

The decision thresholds will be updated as follows:

$$S_{TH} = \frac{S_{M-INIT}}{ng} \quad L_{TH} = L_{INIT} \tag{7}$$

where ng_{old} and ng are the previous and the current number of mixture components, respectively. Finally, their ages, a_A and a_B, will be reset to zero.

2.5 Updating decision thresholds

The thresholds L_{TH}, and S_{TH} vary at each step with the following rules:

$$\begin{aligned} L_{TH} &= L_{TH} - \frac{\lambda}{ng} \cdot L_{TH} = L_{TH} \cdot (1 - \frac{\lambda}{ng}) \\ S_{TH} &= S_{TH} - \frac{\alpha_{Max}}{ng} \cdot S_{TH} = S_{TH} \cdot (1 - \frac{\alpha_{Max}}{ng}) \end{aligned} \tag{8}$$

with ng is the number of current gaussians, λ, and α_{Max} Using high values for λ, and α_{Max} results in high convergence speed. However, with faster convergence comes significant instability around the optimal desidered point. Following this rules L_{TH} will decrease step by step, approaching the current value of the global log-likelihood increment. This is the same for S_{TH}, which will become closer to the maximum *area* of the gaussians, allowing splitting. This will allow the system to avoid some local optima, by varying its configuration if a stationary situation occurs.

Finally, every time a gaussians is added these thresholds will be reset to their initial value.

2.6 Stopping criterion

Analyzing the behavior of the whole mixture log-likelihood emerges a common trend: It always acts like a first order system. In fact, it produces a curve with a high derivate at beginning that decreases going on with the number of iterations, reaching the log-likelihood maximum value asymptotically. We know from the theory that the rate at which the response approaches the final value is determined by the time constant. When $t = \tau$ (in our case $i = \tau$), L has reached 63.2% of its final value. When $t = 5\tau$, L has reached 99.3% of its final value. Again, we know from the theory that the time constant τ is the angular coefficient of the output curve at the time $t = 0$.

We know from the EM theory that at each iteration it has to grow, or at least remaining the same. However, spikes during the splitting operations that make the log-likelihood decreasing abruptly are present. Moreover, in order to avoid local optima-like situations, we average the log-likelihood increments by sampling it with a fixed sampling rate (e.g. $T_s = 25$ iterations).

For each $i = n \cdot T_s$, with n an integer number, we store the current log-likelihood within an array. The first time the log-likelihood increment between two consecutive sampled value increases less than 0.7% we store the relative number of iterations $i_{first} = n_{stop}T_s$. Then, we stop after the log-likelihood does not increase over 0.7% for a number of times equal to n_{stop}.

2.7 Computational complexity evaluation

Within this section we will use the following convention: ng is the number of the mixture gaussians components, k is the number of input vectors, d is the number of input dimension, and it is the number of iterations.

The computational burden of the EM algorithm is, referring to the pseudocode in tab. 2.1 as follows:

- the original EM algorithm (steps 3 to 6) take $O(k \cdot d \cdot ng)$ for 3 and 6, while step 4 and step 5 take $O(1)$ and $O(k \cdot ng)$;
- our algorithm takes $O(ng)$ for evaluating all the gaussians (step 8 to 26);
- our split (step 15) operation requires $O(d)$.
- the others take $O(1)$.

Therefore, the original EM algorithm takes $O(k \cdot d \cdot ng)$, while our algorithm adds $O(d \cdot ng)$ on the whole, giving rise to $O(k \cdot d \cdot ng)$ + $O(d \cdot ng)$ = $O(k \cdot d \cdot ng + d \cdot ng)$ = $(ng \cdot d \cdot (k + 1))$. Considering that usually $d << k$ and $ng << k$ this does not add a considerable burden, while giving an important improvement to the original computation in terms of self-adapting to the data input configuration at best.

3 Experimental Validation

3.1 Experimental set-up

To compare our algorithm other EM-based methods we choose three techniques, BIC, AIC, and MDL, as the most common used selection criteria. In order to reduce the artifact of the initialization on the standard EM algorithm, we adopted a standard approach:

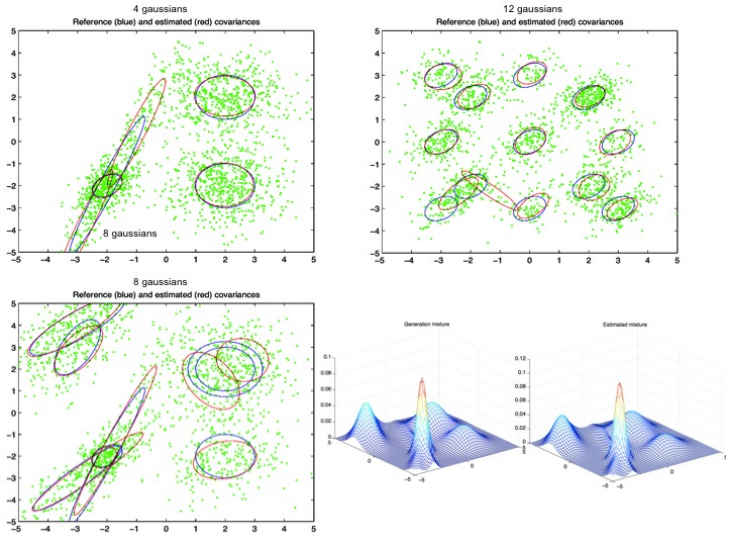

Fig. 1. The 2D representation of the final gaussians mixture generated by our algorithm vs. the real one and the relative log-likelihood outputs as function of the iterations number, for different input mixtures of data (4, 8, 12 gaussians components). Moreover the 8-gaussians case comparison between the generated and computed mixtures is shown on the bottom right.

We selected 10 different initial random conditions, keeping those giving the highest likelihood. The stopping criteria we adopted for the EM computation is the most common used, i.e. it requires that the log-likelihood increment goes below a threshold ϵ. We used $\epsilon = 10 \cdot e^{-5}$. We evaluated our technique's performances by applying it both to synthetic data (artificially generated with a known mixture) and with different kind of pictures, i.e. some well known pictures and some real images (taken by a webcam or by our robotic platform iCub's cameras).

Mixture precision estimation It is possible to see that FASTGMM usually achieves a higher final log-likelihood than the other techniques, although running more iterations. This suggests a better approximation of the data mixture. However, a higher log-likelihood does not strictly imply that the extracted mixture covers the data better than another one. This is because it is based on the probability of each component, which may be more or less exact, being not deterministic. Nevertheless, it is not a good index on the probability that such mixture would be better.

A deterministic approach is to adopt a unique distance measure between the generation mixture and the evaluated one. In [18] Jensen *et Al.* exposed three different strategies for computing such distance: The Kullback-Leibler, the Earh Mover, and the Normalized L2 distance. The first one is not symmetric, even though a symmetrized version is usually adopted in music retrival. However, this measure can be evaluated in a close form only with mono-dimensional gaussians. The second one also suffers ana-

log problems of the latter. The third choice, finally is symmetric, obeys to the triangle inequality and it is easy to compute, with a comparable precision with the other two. We then used the last one. Its expression states [19]:

$$\begin{aligned} z_c N_x(\bar{\mu}_c, \bar{\Sigma}_c) &= N_x(\bar{\mu}_a, \bar{\Sigma}_a) \cdot N_x(\bar{\mu}_b, \bar{\Sigma}_b) \\ & \quad where \\ \bar{\Sigma}_c =& (\bar{\Sigma}_a^{-1} + \bar{\Sigma}_b^{-1})^{-1} \quad and \quad \bar{\mu}_c = \bar{\Sigma}_c(\bar{\Sigma}_a^{-1}\bar{\mu}_a + \bar{\Sigma}_b^{-1}\bar{\mu}_b) \\ z_c =& |2\pi\bar{\Sigma}_a\bar{\Sigma}_b\bar{\Sigma}_c^{-1}|^{\frac{1}{2}} \cdot \exp\left\{-\frac{1}{2}(\bar{\mu}_a - \bar{\mu}_b)^T\bar{\Sigma}_a^{-1}\bar{\Sigma}_c\bar{\Sigma}_b^{-1}(\bar{\mu}_a - \bar{\mu}_b)\right\} \\ =& |2\pi(\bar{\Sigma}_a + \bar{\Sigma}_b)|^{\frac{1}{2}} \cdot \exp\left\{-\frac{1}{2}(\bar{\mu}_a - \bar{\mu}_b)^T(\bar{\Sigma}_a + \bar{\Sigma}_b)^{-1}(\bar{\mu}_a - \bar{\mu}_b)\right\} \end{aligned} \tag{9}$$

Therefore, we evaluated the Normalized L2 distance as a measure of synthetic data estimation precision, and we reported our result in tab. 3.3.

3.2 Synthetic data

Actual number of Gaussian components	Algorithm	Detected number of Gaussian components	Total number of iterations	Final log-likelihood	Normalized L2 distance
4	AIC	4	91	-7403.656573	6.595441
	BIC	4	91	-7405.021887	6.382962
	MDL	6	98	-7460.206259	13.715347
	FASTGMM	4	268	-7405.078438	0.075190
8	AIC	9	120	-8400.626025	34.796101
	BIC	7	91	-8428.323612	18.092732
	MDL	8	111	-8554.125701	22.052649
	FASTGMM	8	650	-8446.063794	6.184175
12	AIC	14	103	-7475.658908	45.687874
	BIC	12	124	-7547.612061	2.811907
	MDL	13	161	-7613.774605	21.293496
	FASTGMM	12	393	-7511.032752	2.658803

Table 1. Experimental results on synthetic data.

In order to evaluate the performance of our algorithm, we tested it by classifying different input data randomly generated by a known gaussians mixture, and subsequently saved to a file. We choose to show the results for 2-dimensional input because they are easier to show than multidimensional ones (for instance, a 2-dimensional gaussians is represented in 2D as an ellipse).

The output of the two algorithms is shown in Fig. 1. Each distribution has a total of 2000 points, but disposed differently. The first one has been generated by a 4 gaussian mixture, the second one by a 8 gaussian mixture and the third one by a 12 gaussian mixture. The generation mixture (blue) and the evaluated one (red) are represented in each subfigure. Finally, the 3D histogram representation of the 8-components generated gaussians mixture data and the estimated one. Due to space limitations, we choose to show only the one that gave rise to the worst log-likelihood estimation plot, i.e. the one with 8 components.

We can see that our algorithm is capable to learn the input data mixture starting from only one component with a good accuracy.

3.3 Colored real images

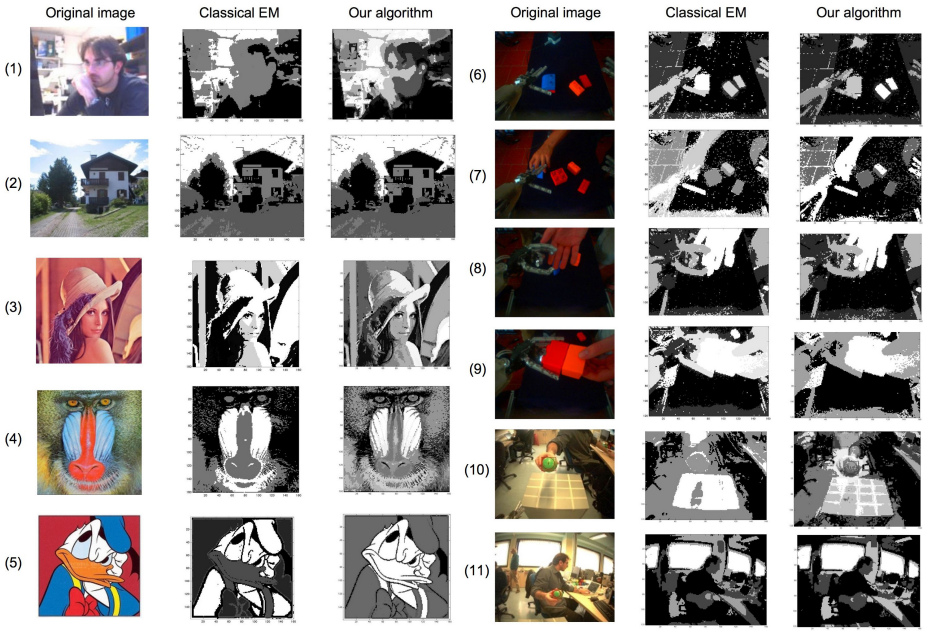

Fig. 2. Color image segmentation results. We divide these images into two groups: Some general images, on the left (from (1) to (5)), and some images taken by the iCub's cameras, on the right (from (6) to (11)). For each group we show the original images, those obtained with the standard EM algorithm initialized with the BIC/AIC/MDL criteria, and those obtained with our algorithm on the left, in the middle, and on the right, respectively.

Learning the right number of color components (i.e. mixture components) within a colored image is a difficult task. This is because an general image contains several of the three fundamental color combinations. Therefore, it is clear that the number of mixture components needed to represent the image at best rapidly rises up excessively, becoming too high.

The color image segmentation results are shown in Fig. 2. The set of images is divided into two groups: Some general images, on the left (from (1) to (5)), and some images taken by the iCub's cameras, on the right (from (6) to (11)). For each group we show the original images, those obtained with the standard EM algorithm initialized with the BIC/AIC/MDL criteria, and those obtained with our algorithm on the left, in the middle, and on the right, respectively.

Image (Fig. 2)	Algorithm	Detected number of mixture components	Total number of iterations	Image (Fig. 2)	Algorithm	Detected number of mixture components	Total number of iterations
(1)	AIC	8	85	(6)	AIC	8	234
	BIC	7	91		BIC	7	180
	MDL	7	106		MDL	8	213
	FASTGMM	10	400		FASTGMM	8	350
(2)	AIC	4	120	(7)	AIC	5	92
	BIC	4	91		BIC	4	90
	MDL	4	134		MDL	4	104
	FASTGMM	4	175		FASTGMM	3	150
(3)	AIC	20	213	(8)	AIC	4	92
	BIC	18	192		BIC	4	90
	MDL	18	221		MDL	4	94
	FASTGMM	22	475		FASTGMM	4	175
(4)	AIC	18	145	(9)	AIC	4	78
	BIC	17	126		BIC	3	94
	MDL	16	153		MDL	3	97
	FASTGMM	20	325		FASTGMM	3	150
(5)	AIC	4	86	(10)	AIC	16	121
	BIC	4	93		BIC	16	112
	MDL	4	91		MDL	15	146
	FASTGMM	4	175		FASTGMM	18	300
(11)	AIC	7	131				
	BIC	7	124				
	MDL	6	156				
	FASTGMM	8	350				

Table 2. Experimental results on real images.

Here we will find some differences in the number of mixture components detected by our algorithm and those detected by the BIC/AIC/MDL techniques. Our approach tends to use more components than BIC/AIC/MDL do. This is more evident on the real images (which of course contain more color variations than the artificial ones). In table 3.3 the results of our algorithm and the BIC/AIC/MDL criteria applied to the selected images are shown.

4 Conclusion and Future work

In this paper we proposed a unsupervised algorithm that learns a finite mixture model from multivariate data on-line. We approached the problem starting from a single mixture component and sequentially *growing* both increases the number of components and adapting their means and covariances. Therefore, its initialization is unique, and it is not affected by different possible starting points like the original EM formulation. Moreover, by starting with a single component the computational burden is low at the beginning, increasing only whether more components are required. We also defined a precise stopping criteria, otherwise the algorithm continues to split indefinitely. Finally, we presented the effectivity of our technique in a series of simulated experiments with synthetic data, artificial, and real images.

- Future work: At the moment we tested our algorithm with synthetic data and static images. As future work, we will improve our algorithm by implementing also a merge technique. So far, it will be possible to remove unused components, too. Our final aim is to apply it to moving objects, online adapting the mixture description with varying input.

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, "earning 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. G. McLachlan and D. Peel, "Finite mixture models." *John Wiley and Sons*, 2000.
8. 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.
9. Y. Sakimoto, M. Iahiguro, and G. Kitagawa, "Akaike information criterion statistics," *KTK Scientific Publisher, Tokio*, 1986.
10. J. Rissanen, "Stochastic complexity in statistical inquiry." *Wold Scientific Publishing Co. USA*, 1989.
11. 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.
12. J. Li and A. Barron, "Mixture density estimation," *NIPS, MIT Press*, vol. 11, 2000.
13. N. Vlassis and A. Likas, "A greedy em algorithm for gaussian mixture learning," *Neural Processing Letters*, vol. 15, pp. 77–87, 2002.
14. J. Verbeek, N. Vlassis, , and B. Krose, "Efficient greedy learning of gaussian mixture models," *Neural Computation*, vol. 15, no. 2, pp. 469–485, 2003.
15. N. Ueda, R. Nakano, Y. Ghahramani, and G. Hiton, "Smem algorithm for mixture models," *Neural Comput*, vol. 12, no. 10, pp. 2109–2128, 2000.
16. Z. Zhang, C. Chen, J. Sun, and K. Chan, "Em algorithms for gaussian mixtures with split-and-merge operation," *Pattern Recognition*, vol. 36, pp. 1973 – 1983, 2003.
17. A. Figueiredo and A. Jain, "Unsupervised learning of finite mixture models," *IEEE Trans. Patt. Anal. Mach. Intell.*, vol. 24, no. 3, 2002.
18. J. H. Jensen, D. Ellis, M. G. Christensen, and S. H. Jensen, "Evaluation distance measures between gaussian mixture models of mfccs," *Proc. Int. Conf. on Music Info. Retrieval ISMIR-07 Vienna, Austria*, pp. 107–108, October, 2007.
19. P. Ahrendt, "The multivariate gaussian probability distribution," http://www2.imm.dtu.dk/pubdb/p.php?3312, Tech. Rep., January 2005.