Chapter 2
Contact-State (CS) Modeling

2.1 Problem Statement

Consider the robot systems shown in Figs. 2.1 and 2.2. The system of Fig. 2.1 is composed of a KUKA LWR manipulating a rigid cube object that interacts with an environment of three orthogonal planes. This process is called a robotic cube-in-corner assembly of rigid objects. Figure 2.1a shows a human operator programming the KUKA LWR for doing the given task through using a gravitational compensation mode (see [96] for more details about the gravitational compensation mode and other modes of operation of the KUKA LWR). For other versions of robots that do not have the feature of the gravitational compensation mode, the robot needs to be guided through the teach pendant for the positions of the considered task. The gravitational compensation mode is used in guiding the robot to the required positions that have different CS's between the manipulated object and the surrounding environment. As soon as these positions are taught, then the robot is changed into position control mode in order to execute the taught positions and in this stage the considered signals are captured for the taught CS's. For instance when the robot moves to *v-f* CS position (that was already taught in the gravitational compensation mode), then the considered signals corresponding to this CS are captured in order to develop the *v-f* CS model and the same is done for all other CS's. Figure 2.2 shows a similar robotic cube-in-corner assembly process but for a flexible rubber object. For both cases of the rigid and flexible objects, one can see that as soon as the robot is programmed to assemble the cube in the corner, different possible CS's would be generated as the task is executed in a position control mode in which the robot moves to the programmed positions without any intervention from the human operator. Therefore, the signals captured during the task execution are not influenced by the human operator. In order to model these CS's, the overall motion is segmented according to the corresponding CS's. For each segment, the wrench signals of the manipulated object are collected and the models, that realize the desired input–output mapping, are developed. For

I.F.J. Ghalyan, *Force-Controlled Robotic Assembly Processes of Rigid and Flexible Objects*, DOI 10.1007/978-3-319-39185-4_2

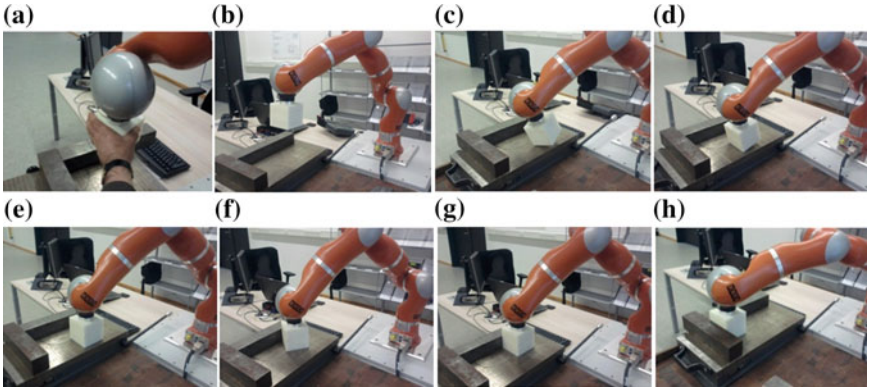

Fig. 2.1 Cube-in-corner assembly task: **a** a human operator programs the robot, **b** free space (*fs*) CS, **c** vertex-face (*v-f*) CS, **d** edge-face (*e-f*) CS, **e** face–face (*f–f*) CS, **f** edge face–2faces (*ef–2f*) CS, **g** 2faces–2faces (*2f–2f*) CS, **h** 3faces–3faces (*3f–3f*) CS

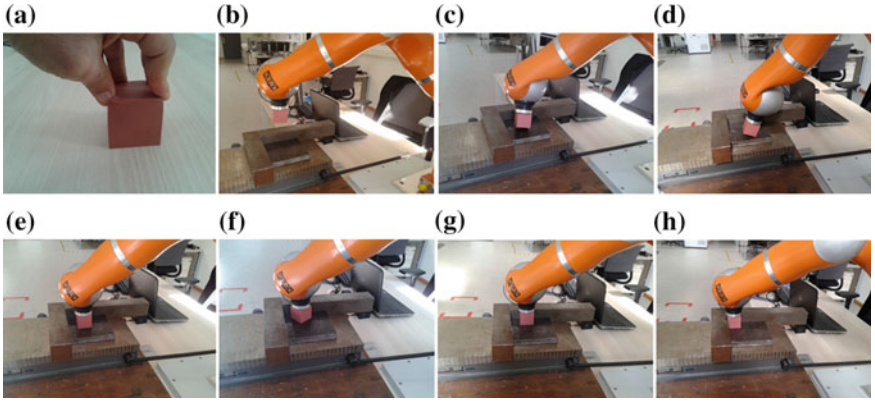

Fig. 2.2 Flexible rubber object cube-in-corner assembly task: **a** the manipulated object deformation when grasped, **b** free space (*fs*) CS, **c** vertex-face (*v-f*) CS, **d** edge-face (*e-f*) CS, **e** face–face (*f–f*) CS, **f** edge face–2faces (*ef–2f*) CS, **g** 2faces–2faces (*2f–2f*) CS, **h** 3faces–3faces (*3f–3f*) CS

the manipulated object, one can describe the wrench signals W to be

$$W = [f_x, f_y, f_z, \tau_x, \tau_y, \tau_z] \tag{2.1}$$

where f_x, f_y, and f_z are the forces along the x, y, and z axes respectively. τ_x, τ_y, and τ_z are the torques around the x, y, and z axes, respectively.

Hence, we have 6 input signals for the classifier, say $\boldsymbol{x}_k = [x_{k,1}, x_{k,2}, \ldots, x_{k,6}]$ with k to be the sample index. The CS classification problem can be formulated as

$$y_k = \begin{cases} 1 & \text{if } (\boldsymbol{x}_k \in \text{current CS}) \\ 0 & \text{Otherwise} \end{cases} \tag{2.2}$$

y_k is the output of the CS classifier. It can be seen that (2.2) represents a nonlinear mapping between $\boldsymbol{x}_k$ and y_k and the goal of almost all modeling and classification researches is to approximate or realize this mapping as accurate as possible. Even though excellent performance was reported in the literature to model (2.2), the signals non-stationarity (non-normal distribution) was not considered resulting in modeling performance degradation. For instance, Fig. 2.6a illustrates the distribution of the torque signal around the x axis and its non-stationary distribution is clear. If one models the signal shown in Fig. 2.6a using a normal Gaussian distribution, then a significant modeling error is expected resulted from the poor modeling. Furthermore, the object flexibility, for the case of Fig. 2.2, would even bring about higher non-stationarity to the captured signals that makes their modeling process to be more difficult. Moreover, the available CS modeling schemes consider either the rigid or the flexible object manipulation and till now there is no CS modeling scheme that is efficiently applicable to model (2.2) for both rigid and flexible objects. Thus, the objective of the CS modeling part is to propose a task sequence-free CS modeling scheme that accommodates the signals non-stationarity and applicable to rigid/flexible objects. The next sections explain the methodology proposed for realizing (2.2).

2.2 Gravitational Search–Fuzzy Clustering Algorithm (GS–FCA)

In this section, a CS recognition system is proposed that relies on building a Takagi–Sugeno (T–S) fuzzy model with multiple if-then rules for each CS. A T–S fuzzy model is composed of two main blocks; the antecedent part (or called the If part) and the consequent part (or called the Then part). In the antecedent part, the membership functions of the input variables are specified. Hence, the parameters of the membership functions are called the antecedent part parameters. In the consequent part of T–S fuzzy models, linear blending of the antecedent part outputs is employed and the parameters of the linear blending are called consequent part parameters (see [3, 43, 135] for more details about the antecedent and consequent parts of the fuzzy systems). The antecedent part parameters for each model are computed by the Gravitational Search–Fuzzy Clustering Algorithm GS–FCA approach [59, 104]. The Least Mean Square (LMS) is used in tuning the consequent part parameters for each if-then rule of each CS model. The main contribution of this strategy is to have a CS recognition system with the following features:

1. The suggested approach does not require knowing the CS's sequence or graph.
2. Enhanced input–output mapping through using:
   i. GS–FCA and LMS in tuning the T–S fuzzy models.
   ii. Multiple rules are used for each CS model.

2.2.1 T–S Fuzzy Modeling

Suppose that we are given a data set $\boldsymbol{x}_k = [x_{k,1}, x_{k,2}, \ldots, x_{k,D}]$, D is the width of the data and consider that $X = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N)$. Since it is required to approximate (2.2), then one can consider it as an unknown nonlinear equation that is described as:

$$y_k = f(\boldsymbol{x}_k) \tag{2.3}$$

(2.3) maps the available input–output data, say $\boldsymbol{x}_k$ and y_k. The nonlinear equation (2.3) can be efficiently modeled using a T–S fuzzy system that breaks the nonlinear system into sum of linear models each one of them is described by the following if-then rules [165]:

$$R_i : \text{If } x_{k,1} \text{ is } A_{i1}(x_{k,1}) \text{ and } x_{k,2} \text{ is } A_{i2}(x_{k,2}) \text{ and} \ldots \text{and } x_{k,D} \text{ is } A_{iD}(x_{k,D}) \text{ Then } \hat{y} = \mathbf{a}_{Gi}^T \boldsymbol{x}_k + b_{Gi} \tag{2.4}$$

$A_{ij}(x_{k,j})$ is a membership function that quantifies for the input $x_{k,j}$. $\mathbf{a}_{Gi}$ and b_{Gi} are the parameters of the ith local linear model. Equation (2.4) can be aggregated using the fuzzy mean approach and the output can be computed as:

$$\hat{y}_k = \frac{\sum_{i=1}^{c_F} \beta_{Gi}(\boldsymbol{x}_k)(\mathbf{a}_{Gi}^T \boldsymbol{x}_k + b_{Gi}))}{\sum_{i=1}^{c_F} \beta_{Gi}(\boldsymbol{x}_k)} \tag{2.5}$$

where:

$$\beta_{Fi}(\boldsymbol{x}_k) = \Pi_{j=1}^{n} A_{ij}(x_{k,j}) \tag{2.6}$$

c_F is the number of rules for each model. If we use Gaussian membership functions for the antecedent part of (2.4), then we have

$$A_{ij}(x_j) = exp\left(-\frac{(x_{k,j} - \delta_{ij})^2}{(2\rho_{ij}^2)}\right) \tag{2.7}$$

δ_{ij} and ρ_{ij}^2 are the membership function center and variance respectively. For the robot system under consideration, it is required to classify the CS's according to the available wrench readings given in (2.1). In the framework of function approximation, T–S fuzzy system is proposed to approximate (2.2) from the given input–output data and relying on (2.3)–(2.7). However, the approximation accuracy would highly rely on the choice of the antecedent and consequent parts parameters for each if-then rule. The following subsection describes how one can choose those parameters for an optimal classification process.

2.2.2 T–S Fuzzy Models Parameters Estimation

The estimation of the T–S fuzzy models involves the estimation of the antecedent part and the consequent part of the if-then rules of the system. Hence in the following subsections, the estimation of both of the antecedent and consequent part parameters are detailed.

2.2.3 Antecedent Part Parameters Estimation

The Gravitational Search–Fuzzy Clustering Algorithm (GS–FCA) is used in computing the antecedent part parameters, say the membership functions parameters. Using the clustering process, the data set $\boldsymbol{x}$ can be grouped into clusters in which the data group of each cluster share a certain attribute, i.e., grouping the data set X into $c \in \{2, \ldots, N-1\}$ clusters. Fuzzy clustering performs the function of data clustering using the concept of fuzzy sets theory. Each element of the set X is assigned with a membership function that quantifies the degree of its affiliation to one of the given clusters. Consider that μ_{Gij} represents the membership function that quantifies the strength in which the ith data vector of X belongs to the jth cluster, then $\mu_{Gij} \in [0, 1]$, $(i = 1, 2, \ldots, N; j = 1, \ldots, c)$. Extending μ_{Gij} for all data vectors and over all clusters, a partition matrix, denoted as $U \in M$, is obtained which can be described by:

$$M = \{U : U \in [0, 1]^{N \times c}\} \tag{2.8}$$

However, the following constraints are necessary to be satisfied for the selection of the partition matrix:

$$\sum_{j=1}^{c} \mu_{Gij} = 1; i = 1, 2, \ldots, N \tag{2.9}$$

$$\sum_{i=1}^{N} \mu_{Gij} \geq 0; j = 1, \ldots, c \tag{2.10}$$

The fuzzy clustering problem can be solved by finding the clusters centers and the partition matrix. Fuzzy c-means (FCM) clustering is a widely used approach in which both the clusters centers and the partition matrix are found through solving the following constrained optimization problem:

$$\text{minimize } J_f = \sum_{j=1}^{c} \sum_{i=1}^{N} \mu_{Gij}^{m} \|v_j - x_i\|_2 \tag{2.11}$$

subject to (2.9) and (2.10).

With $m > 1$, v_j is the jth cluster center, and $\|.\|_2$ represents a norm on R, frequently Euclidean norm is utilized. It is worth noting that x_i could be a vector of signals (which is the case of our application) and the clustering is then performed in a vector wise for the clusters centers, that is each cluster center would be a vector of dimension equal to the number of columns of the vector x_i. Alternating Optimization (AO) was successfully used to solve the constrained optimization above and the following solution was obtained [123]:

$$\mu_{ik} = \frac{1}{\sum_{j=1}^{c} \left(\frac{\|v_k - x_i\|_2}{\|v_j - x_i\|_2} \right)^{\frac{2}{m-1}}} \tag{2.12}$$

and

$$v_j = \frac{\sum_{i=1}^{N} \mu_{Gij}^m x_i}{\sum_{i=1}^{N} \mu_{Gij}^m} \tag{2.13}$$

However, it was found that such a solution can be easily trapped in local minima because they depend on derivatives in their optimization process [145]. Using a more powerful stochastic optimization algorithm can highly improve the fuzzy clustering algorithm above. Gravitational Search Algorithm (GSA) is a recently developed efficient stochastic optimization algorithm that depends on the concept of Newton laws of gravity and motion. It is assumed that all agents within a certain population have certain gravitational forces between them, and the agent with higher mass exerts higher force and becomes more optimal (see [140] for more details about the GSA and its performance). Suppose that we have n agents and the pth agent is described by:

$$z_p = (z_p^1, \ldots, z_p^m);\ p = 1, 2, \ldots, n \tag{2.14}$$

z_p^k is the position of the pth agent in the kth dimension. Lets define the gravitational force between the pth and the qth agents to be:

$$F_{pq}^d = \frac{M_p(t) \times M_q(t)}{\|z_p - z_q\|} \left(z_q^d - z_p^d \right) \tag{2.15}$$

where $M_p(t)$ and $M_q(t)$ are the masses, z_p^d and z_q^d are the positions of the pth and qth agents respectively, and $G(t)$ is the gravitational constant. The gravitational constant can be described by the following expression:

$$G(t) = G_o exp\left(-\frac{\alpha t}{\max_t} \right) \tag{2.16}$$

G_o is the initial value of the gravitational constant, α is a constant, t is the current iteration, and $\max_t$ is the maximum iteration. The inertial mass for the pth agent can be computed as:

$$M_p(t) = \frac{m_p(t)}{\sum_{q=1}^{N} m_q(t)} \tag{2.17}$$

m_p can be found as:

$$m_p(t) = \frac{fit_p - worst}{best - worst} \tag{2.18}$$

fit_p is the value of the fitness function (objective function) for the pth agent, $best$ and $worst$ have different expressions depending on the nature of the optimization problem in hand, i.e., minimization or maximization optimization problems. For minimization problems $best = \min(fit_p)$ and $worst = \max(fit_p)$, and for maximization problems $best = \max(fit_p)$ and $worst = \min(fit_p)$. The force exerted on the pth agent can be computed as a random weighted sum of all attraction forces from other agents, i.e.,

$$F_p^d(t) = \sum_{p \neq q} \text{rand}_q \cdot F_{pq}^d(t) \tag{2.19}$$

rand_q is a random number. Using the Newton's law of motion, we can find the acceleration of the pth agent movement as:

$$a_p^d(t) = \frac{F_p^d(t)}{M_p(t)} \tag{2.20}$$

The new velocities and positions can be computed according to the following equations:

$$v_p^d(t+1) = \text{rand}_p \cdot v_p^d(t) + a_p^d(t) \tag{2.21}$$

$$z_p^d(t+1) = z_p^d(t) + v_p^d(t+1) \tag{2.22}$$

The GSA optimization above has proved to be efficient in solving the clustering problems [59, 104]. Furthermore, the GSA is a random search method that does not depend on the computations of derivatives and consequently the problem of trapping in local minima can be avoided. Moreover, the GSA can be used to solve optimization problems in which the objective function is non-differentiable, and this opens the door to its applicability to discontinuous functions identification. However, the dimension of the data to be clustered may restrict the usability of the GSA algorithm for finding only the clusters centers since using the GSA in finding the partition matrix would be tedious for high dimensional data. Therefore, in order to minimize the objective function given in (2.11), we will use (2.12) for computing the partition matrix, but for clusters centers, instead of using (2.13), we will utilize the GSA in minimizing the objective function (2.11). The algorithm below details the GS–FCA through which the centers of clusters are computed using the GSA optimization:

GS–FCA Algorithm

Step 1: Set $l = 1$, initialize the centers and code them into positions of agents. Initialize the tolerance ϵ. Initialize U^l.
Step 2: Compute the objective function for each agent using (2.11). Update l as $l = l + 1$.
Step 3: Update G using (2.16) and find the best and worst of the agents.
Step 4: Compute the mass M_p and then calculate the gravitational force for each agent F_p using (2.19).
Step 5: Compute the acceleration for each agent using (2.20).
Step 6: Update the velocity and position for each agent according to (2.21) and (2.22) respectively.
Step 7: Compute the partition matrix $U^l = [\mu_{Gij}]^{n\times c}$ using (2.12).
Step 8: If $|U^l - U^{l-1}| \leq \epsilon$ then stop. Otherwise repeat **Steps** 2–8.

As per accomplishing the algorithm above, the center and variance for each membership function can be computed as [123]:

$$\delta_{ij} = \frac{\sum_{k=1}^{L} \mu_{kj} x_{kj}}{\sum_{k=1}^{L} \mu_{kj}} \tag{2.23}$$

$$\rho_{ij} = \sqrt{\frac{\sum_{k=1}^{L} \mu_{kj} (x_{kj} - \delta_{ij})}{\sum_{k=1}^{L} \mu_{kj}}} \tag{2.24}$$

2.2.4 Consequent Part Parameters

The consequent parts parameters, say a_{Gi} and b_{Gi}, are tuned using the Least Mean Square (LMS) algorithm. Suppose that $\theta_{Gi} = [a_{Gi} b_{Gi}]$, then the value of θ_{Gi} can be computed as [2]:

$$\theta^*_{Gi} = \arg\min_{\theta_{Gi}} \frac{1}{N} (\mathbf{y} - \mathbf{X}\theta_{Gi})^T \Phi_i (\mathbf{y} - \mathbf{X}\theta_{Gi}) \tag{2.25}$$

where $\mathbf{X} = [\mathbf{x}1]$ and Φ_i is a diagonal matrix with the membership grades are the elements of the main diagonal:

$$\Phi_i = \begin{pmatrix} \mu_{i1} & 0 & \dots & 0 \\ 0 & \mu i2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 \dots & 0 & \mu iL \end{pmatrix} \tag{2.26}$$

Φ_i is obtained from the GS–FCA algorithm explained above. Using the LMS, the parameters can be updated as:

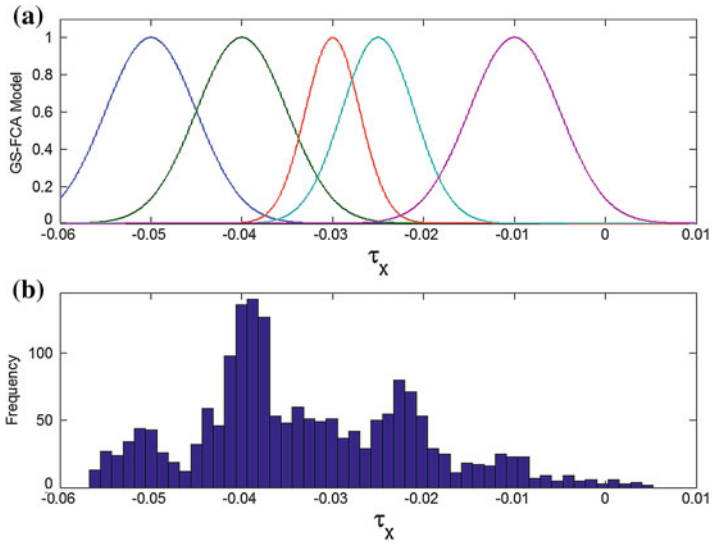

Fig. 2.3 **a** The shape of the GS–FCA model for modeling the τ_x signal; **b** The distribution of the τ_x signal

$$\theta_{Gi} = (\mathbf{X}^T \Phi_i \mathbf{X})^{-1} \Phi \mathbf{X}^T \Phi_i \mathbf{y} \tag{2.27}$$

Therefore, the models are tuned to be fitted more to their desired input–output mappings. Despite the excellent recognition performance reported when using the GS–FCA for the CS recognition process [74], there are two drawbacks noted during the implementation of the experiments. The first drawback is the high computational cost required for developing the models and the second one is the fixed fuzzy sets amplitude of 1 as a maximum value that makes it not reflecting the precise signals distribution. As an example let's contemplate the GS–FCA model of the τ_x signal shown in Fig. 2.3a. One can see that the constant peaks of the fuzzy sets makes it significantly dissimilar to the real distribution shown in Fig. 2.3b. In the next section both drawbacks are overcome by proposing a more efficient modeling strategy with enhanced performance and reduced computational cost.

2.3 Expectation Maximization-Based Gaussian Mixtures Model (EM-GMM)

The main motivation behind employing the EM-GMM in modeling the captured wrench signals, of the force-guided robotic assembly processes, is the ability in capturing the non-stationarity in the signals distribution that would give more accurate modeling process. For instance, if one examines Fig. 2.4 that shows the distribution

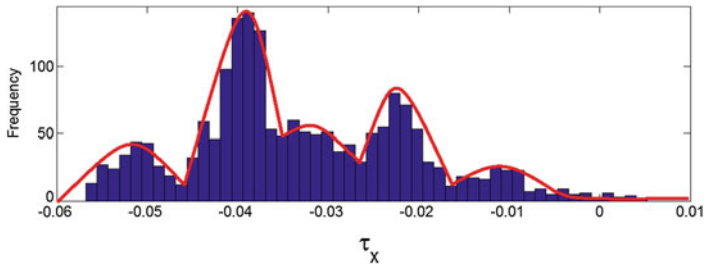

Fig. 2.4 The distribution of the τ_x signal (in *blue*) and the Gaussian Mixtures Model of τ_x (in *red*)

of the τ_x signal and its GMM model, then one can see that the GMM is more precise in capturing the different peaks of the distribution of the τ_x signal that would enhance the given data modeling process. Before explaining the details of the EM-GMM CS modeling scheme, the principles of the Bayesian modeling (or classification) are clarified.

2.3.1 Bayesian Classification

Suppose that one is given the data vector $\boldsymbol{x}_k = [x_{k,1}, x_{k,2}, \ldots, x_{k,D}]$ where D is the width of the vector (for the CS recognition problem addressed in this book, it is clear that $D = 6$ and each model has 6 inputs). Suppose that the vector $\boldsymbol{x}_k$ belongs to one of the classes $[c_1, c_2, \ldots, c_C]$. In the framework of the Bayesian classification, one can say that the vector $\boldsymbol{x}_k$ belongs to a class c_i implies that [14]:

$$p(c_i|\boldsymbol{x}_k) \geq p(c_j|\boldsymbol{x}_k) \tag{2.28}$$

for $i \neq j$. $p(c_i|\boldsymbol{x}_k)$ is called the posterior probability of class c_i given the vector $\boldsymbol{x}_k$ and can be computed using the Bayes rule as:

$$p(c_i|\boldsymbol{x}_k) = \frac{p(\boldsymbol{x}_k|c_i)\,p(c_i)}{p(\boldsymbol{x}_k)} \tag{2.29}$$

where $p(\boldsymbol{x}_k|c_i)$ is the Probability Density Function (PDF) of class c_i in the vector space of $\boldsymbol{x}_k$, $p(c_i)$ is the priori probability that represents the probability of class c_i, and $p(\boldsymbol{x}_k)$ is the probability of the vector space $\boldsymbol{x}_k$ that can be expressed as:

$$p(\boldsymbol{x}_k) = \sum_{i=1}^{C} p(\boldsymbol{x}_k|c_i)\,p(c_i) \tag{2.30}$$

From (2.30), one can notice that for equal class priori $p(c_i)$, the term $p(\boldsymbol{x}_k)$ of (2.29) would be merely a scaling factor. Therefore, it can be deduced that the vector $\boldsymbol{x}_k$ belongs to a class c_i implies that:

$$p(\boldsymbol{x}_k|c_i)p(c_i) \geq p(\boldsymbol{x}_k|c_j)p(c_j) \tag{2.31}$$

for $i \neq j$. Hence, the best approximation of the term $p(\boldsymbol{x}_k|c_j)$ results in the best classification for the pattern $\boldsymbol{x}_k$. In the conventional Bayesian classifier, a Gaussian distribution is used in approximating the term $p(\boldsymbol{x}_k|c_j)$, that is:

$$p(\boldsymbol{x}_k|c_i) = \frac{1}{(2\pi)^{D/2}|\Sigma|^{\frac{1}{2}}} exp\left(-\frac{1}{2}(\boldsymbol{x}_k - \mu)^T \Sigma^{-1}(\boldsymbol{x}_k - \mu)\right) \tag{2.32}$$

where $\mu \in R^D$ is the mean, $\Sigma \in R^{D\times D}$ is the covariance matrix, and $|\Sigma| \in R$ is the determinant of Σ. It was shown that the approximation (2.32) performs well for cases of normal signals distribution. However, in many cases, one may face situations in which the vector space signals, or several signals of the vector space, have non-normal distribution and consequently the use of (2.32) results in significant modeling errors.

2.3.2 Gaussian Mixtures Model (GMM)

In order to accommodate the possible non-normal distribution of the signals, Gaussian mixtures is employed in modeling the features (input signals), i.e., assigning more than a Gaussian component for each feature. Suppose that a single Gaussian distribution is represented as

$$N(\boldsymbol{x}_k, \mu, \Sigma) = \frac{1}{|2\pi|^{D/2}|\Sigma|^{\frac{1}{2}}} exp\left(-\frac{1}{2}(\boldsymbol{x}_k - \mu)^T \Sigma^{-1}(\boldsymbol{x}_k - \mu)\right) \tag{2.33}$$

Then, a Gaussian Mixtures Model (GMM) can be described as:

$$p(\boldsymbol{x}_k|c_i) = \sum_{q=1}^{M} \omega_q N_q(\boldsymbol{x}_k, \mu_q, \Sigma_q) \tag{2.34}$$

M is the total number of the GMM components, ω_q, μ_q, and Σ_q are the weight, mean, and covariance of the qth Gaussian component. Suppose that $\theta_q = (\omega_q, \mu_q, \Sigma_q)$ and consider the parameter vector $\theta = [\theta_1, \theta_2, \ldots, \theta_M]^T$. It is clear that finding the values of the parameters is very important in having a precise modeling of the given features. Therefore, one can write the model of (2.34) in terms of the parameters θ as:

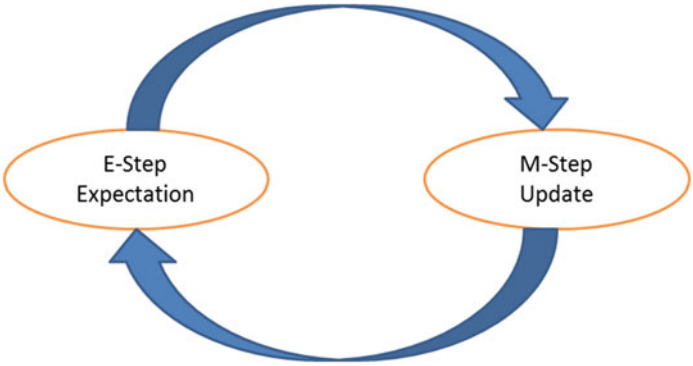

Fig. 2.5 The main two steps of the Expectation Maximization (EM) algorithm

$$p(\boldsymbol{x}_k|c_i;\theta) = \sum_{q=1}^{M} \omega_q N_q(\boldsymbol{x}_k, \mu_q, \Sigma_q) \tag{2.35}$$

Finding the parameter vector θ that optimizes the models from the available measurements would enhance the performance of the classification process.

2.3.3 Expectation Maximization (EM)

One of the most efficient approaches in finding these parameters is the Expectation Maximization (EM) algorithm. The EM algorithm is composed of two steps; the E-step in which the log-likelihood is estimated for the current parameters, and the M-step in which the parameter θ is updated such that a maximized log-likelihood would result. Figure 2.5 shows the block diagram of the phases of the EM algorithm. In order to explain the EM algorithm, let's consider the overall data $X = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N]^T$. The likelihood function of the data X given the parameter θ can be defined as:

$$\ell(X;\theta) = \prod_{n=1}^{N} p(\boldsymbol{x}_n;\theta) \tag{2.36}$$

Define the logarithm of $\ell(X;\theta)$ to be $L(X;\theta)$ which is called the log-likelihood. Taking the logarithm for both sides of (2.36), then the log-likelihood can be expressed as:

$$L(X;\theta) = \sum_{n=1}^{N} \ln(p(\boldsymbol{x}_n;\theta)) \tag{2.37}$$

The parameter θ that maximizes (2.37) can be described as:

$$\theta(t) = \arg\max_{\theta} L(X; \theta(t)) \tag{2.38}$$

subject to:

$$\sum_{q=1}^{M} \omega_q = 1$$

(2.38) is a constrained optimization problem and the analytical solutions can be intractable. Therefore, iterative solutions, like the EM algorithm, were suggested to solve such a problem. An important quantity that plays a vital role in the EM algorithm is the conditional probability of y_k given $\boldsymbol{x}_k$ and let's denote $p(c_i = 1|\boldsymbol{x}_k)$ as $\gamma(c_{ik})$. The value of $\gamma(c_{ik})$ can be computed using Bayes rule as:

$$\gamma(c_{ik}) = \frac{p(c_i = 1)p(\boldsymbol{x}_k|c_i = 1)}{\sum_{j=1}^{M} p(c_j = 1)p(\boldsymbol{x}_k|c_j = 1)} \tag{2.39}$$

that leads to:

$$\gamma(c_{ik}) = \frac{w_i N_i(\boldsymbol{x}_k, \mu_i, \Sigma_i)}{\sum_{j=1}^{M} w_j N_j(\boldsymbol{x}_k, \mu_j, \Sigma_j)} \tag{2.40}$$

$\gamma(c_{ik})$ is called the responsibility that the ith component takes for explaining $\boldsymbol{x}_k$ [14]. Suppose that the parameter θ at a certain iteration is θ and that of the next iteration is θ^{new}. One can say that the convergence of θ is achieved if:

$$|\theta^{new} - \theta| \leq \varepsilon_m \tag{2.41}$$

Likewise to the log-likelihood, its convergence is achieved if:

$$|\ln p(X; \theta^{new}) - \ln p(X; \theta)| \leq \epsilon_m \tag{2.42}$$

where ε_m and ϵ_m are small positive constants that quantify for the convergence of θ and $\ln p(X; \theta)$ respectively. The steps below summarize the EM algorithm:

Step 1: Initialize the parameter vector $\theta_i = (\omega_i, \mu_i, \Sigma_i)$. Initialize the convergence parameters ε_m and ϵ_m.
Step 2: (E-Step) For the current parameter vector θ_i compute the responsibilities using (2.40).
Step 3: (M-Step) Re-estimate the parameters using the current responsibilities:

$$\mu_i^{new} = \frac{1}{N_i} \sum_{n=1}^{N} \gamma(c_{in}) \boldsymbol{x}_n \tag{2.43}$$

$$\Sigma_i^{new} = \frac{1}{N_i} \sum_{n=1}^{N} \gamma(c_{in})(\boldsymbol{x}_n - \mu_i^{new})(\boldsymbol{x}_n - \mu_i^{new})^T \tag{2.44}$$

$$\omega_i^{new} = \frac{N_i}{N} \tag{2.45}$$

with:

$$N_i = \sum_{n=1}^{N} \gamma(c_{in}) \tag{2.46}$$

Step 4: Compute the log-likelihood:

$$\ln p(X; \theta) = \sum_{n=1}^{N} \ln \left\{ \sum_{i=1}^{M} \omega_i N(\boldsymbol{x}_n, \theta) \right\} \tag{2.47}$$

Step 5: Check for the convergence:

If (2.41) or (2.42) are satisfied then stop. Otherwise, go to **Step 2**.

See ([14]: Chap. 9) for more details on the EM-GMM algorithm and the derivations of the equations above. The EM-GMM is used in building the likelihood of each signal for all CS's, and a classifier is developed for each CS in the framework of the Bayesian classification. The nonstationary behavior of the captured force and torque signals is accommodated that enhances the recognition performance.

In spite of the fact that the EM-GMM scheme accommodates the signals nonstationary behavior, an issue remains open which is the determination of the optimal number of GMM components for each CS model. One of the vital and simple methods, to determine the optimal number of the GMM components, is to find a similarity measure between the developed models and the distribution of the captured signals along with invoking to the fact that the optimal number of GMM components results in the highest similarity measure. For instance, if one considers the histogram shown in Fig. 2.6a which is taken for the torque signal around the x-axis of one of the experiments. Then, the best choice of the number of GMM components is realized by finding the number of GMM components resulting in highest similarity compared with the signal distribution shown in Fig. 2.6a. Different approaches were developed in quantifying the similarity between the two distributions, however, Probabilistic Similarity Measure (PSM) is considered one of the most effective similarity measure schemes. The PSM is summarized by using the CSR performance as an index for the similarity measure between the distribution of the signals and the developed model [23]. The best CSR results from the highest similarity between the developed model and the distribution of the signals. Thus, for the example of Fig. 2.6a, the best CSR results with 3 GMM components. Figure 2.6b shows the GMM that can optimally model the given histogram. If one increases or decreases the number of GMM components (from the optimal number of GMM components), then the similarity between the histogram of the signal and the developed model is decreased causing

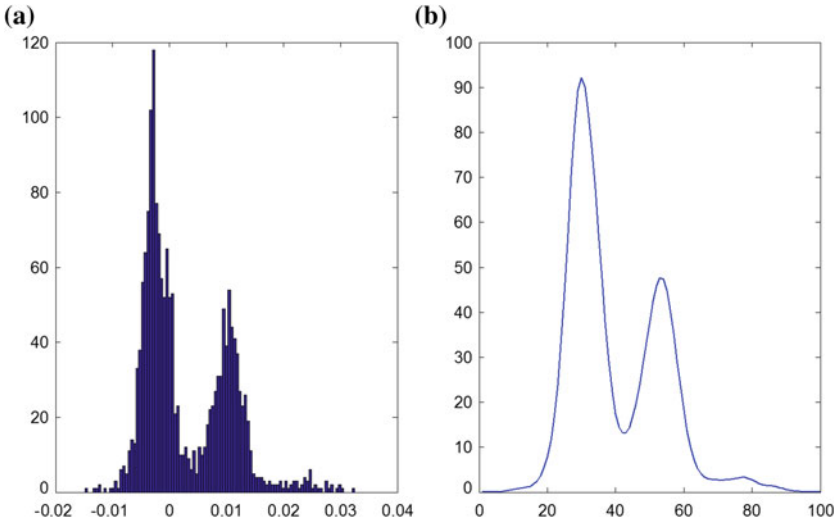

Fig. 2.6 **a** τ_x histogram; **b** τ_x optimal GMM model

CSR degradation. Hence, for any assembly task, the optimal number of GMM components is found by increasing the number of GMM components and continuously observing the CSR values. The number of GMM components resulting in the highest CSR value is considered as the optimal number of GMM components. So, for the assembly processes of flexible rubber objects, one can accommodate the signals extra non-stationarity by finding the optimal number of GMM components resulting with the highest CSR value.

Thus, the EM-GMM CS modeling scheme results in models that are:

1. Independent of the task sequence.
2. Accommodating signals non-stationarity.
3. Applicable to both rigid and flexible objects.

The other improvements brought about by the EM-GMM CS modeling scheme, like high CSR performance and enhanced computational time, will be shown experimentally in Chap. 5.

http://www.springer.com/978-3-319-39184-7

Force-Controlled Robotic Assembly Processes of Rigid and Flexible Objects
Methodologies and Applications
Ghalyan, I.F.J.
2016, XXI, 184 p. 93 illus., 70 illus. in color., Hardcover
ISBN: 978-3-319-39184-7