Robust sensing for force-controlled assembly

H. Bruyninckx*
Dept. of Mechanical Engineering
Katholieke Universiteit Leuven
Leuven, Belgium

G. Hovland†, B. J. McCarragher
Faculty of Engineering and Information Technology
The Australian National University
Canberra, Australia

Abstract

This workshop introduces a number of statistical tools that can be used to increase the reliability and robustness of force-controlled assembly tasks. The emphasis is on efficient, model-based sensor processing methods, and not on the force control aspects proper. The statistical tools are applied at different levels of the control hierarchy: set-point control, contact state monitoring, and action planning. All techniques are illustrated by experiments. The interpretation and critical assessment of the performance of the presented stochastic methods prevails over their detailed theoretical developments (for which the interested reader should consult the references).

1 Introduction

The major reason to use force control in a robotic assembly system is to be able to cope with *uncertainty*: the robot is not a perfectly accurate motion device, the tools and objects the robot has to work with are not exactly known, and are not at exactly the location the robot expects them to be. Hence, sensor feedback is necessary in order not to break objects or loose guiding contacts during the assembly task. This text discusses "assembly" operations only, because, typically, in this context uncertainties are bounded, initial estimates are available, executions are repetitive. Tasks may vary from "once-only" with "large" uncertainties (hence inherently slow) to repetitive tasks with "small" uncertainties, for which "training" or "learning" allow to make execution faster and/or better.

The obvious way to deal with uncertainties in a system is to apply *statistical tools*. Surprisingly enough, only very little research has been done in applying statistics to robotic assembly, certainly in comparison with other sensor-based applications such as mobile navigation and computer vision. One possible explanation is that most research work emphasized the *force control* aspects of the assembly process, and the fact that the robot physically *interacts* with its environment makes this control problem inherently much more difficult than in the other mentioned areas.

This workshop presents the current state of the art in using statistical tools during force-controlled robotic assembly. The authors know that many different and possibly better statistical approaches exist in the vast literature. (Wherever possible, pointers to promising techniques and tools are given.) However, this text presents implementations of *simple* and *efficient* tools that allow for the increase the reliability of the execution. In their current form, these tools are "just" *low-level* interfaces to the on-line task controller in the sense that (i) they work with the same inputs as the controller (i.e., forces and motions), and (ii) they cannot guarantee 100% reliability and robustness since their ability to *reason* about the evolution of the task is limited. Basically, the presented tools implement *process monitoring* (basically *change detection*) and *uncertainty identification*. One indispensable module in any autonomous robotic system is "error detection and recovery." Although the change detection modules of this text can (and should) be used to start error recovery, this error recovery still requires a level of reasoning and (re)planning beyond the current state of the art [71].

Overview. The following sections of this Introduction sketch in more detail the context of using statistical tools in force-controlled assembly: What are the sources of uncertainty, and how do they affect the task execution? (Sect. 1.1) Where in the classical control loops can statistical methods be integrated? (Sect. 1.2) What exactly are they good for? (Sect. 1.3) Then, Section 2 describes the basic concepts used to integrate uncertainties into the *modeling* of force-controlled assembly tasks. Section 3 introduces the statistical tools used in this text; the next sections then illustrate them with experimental results and applications in the area of robotic assembly.

*Postdoctoral Fellow of the Fund for Scientific Research–Flanders (F.W.O.) in Belgium.

†Sponsored by the Research Council of Norway.

1.1 Sources of uncertainty

Look at a typical set-up for force-controlled assembly: a grasping tool is attached to the end-effector of an (industrial) manipulator; the tool has grasped one part of the object to be assembled; the planned motion of the robot is adapted on-line by a control loop that interprets the sensed forces exerted on the robot's wrist force/torque sensor. Hence, the following sources of uncertainty are immediately recognized:

1. *Noise* Force sensors inherently produce noisy signals; many industrial robots estimate velocity by differentiating encoder position signals, which gives quantization noise especially at the low velocities typical for force-controlled tasks.

2. *Inaccurate models* In the first place, accurate dynamic or kinematic models do not exist, e.g. for the relative positions and orientations between the robot end-effector, the force sensor, the grasping tool, the grasped object and the assembly fixture; for the friction in the robot as well as in the contacts; for the impedance of the interaction with the environment; for the impact phenomenon. Moreover, users don't want to do the effort to identify all model parameters before task execution. And probably these parameters change during the task, or between subsequent executions due to production tolerances and imprecise processes such as welding or casting.

3. *Inaccurate control* No robot system can operate completely in open loop, and, by definition, force-controlled systems certainly cannot. Hence, regulation and/or tracking errors are inevitable.

Another major problem comes from modeling the uncertainty itself: finding an accurate probability density function ("pdf") describing the uncertainty in a given assembly process is practically speaking impossible, *and* almost no real-time algorithms exist to work with a pdf more complicated than a Gaussian distribution, i.e., taking into account more than the mean and (co)variance only, [58]. Moreover, some task-determining parameters are usually not known or not sensed directly, but must be *estimated* ("observed," "identified"). Hence, the noisy sensor signals are to be transformed through "measurement equations" linking the sensor signals with the estimated parameters. Extra "uncertainty" creeps in during these transformations, since (i) they are based on uncertain models, and (ii) linearizations are often required.

Hence, the reader should evaluate all statistical methods (including the ones in this text) very critically, and be aware of the limiting assumptions underlying their applicability.

1.2 Control architecture

A force-controlled assembly task in general consists of the following modules:

1. **Planning** (or "specification"). This module determines what contact situations are desired, in what sequence, and what action (i.e., force and/or velocity set-points, in combination with sensor processing algorithms) must realise this sequence. A typical assembly task can be represented by a *Discrete Events System*, since contact situations and contact transitions (gain or loss of contact) are discrete, [72, 73, 70].

2. **Execution** (or "control"). Between contact transitions (and starting in a given contact situation) a continuous controller is used, regulating the forces and motions prescribed by the task specification in this state.

3. **Monitoring** ("sensing"): the controller uses the model and the specifications given to it by the planner, and works within this context. The monitoring module checks (i) whether the model and the actual sensor data (still) correspond, (ii) at what time instant a transition has taken place, and (iii) which new contact state this transition has generated.

4. All three modules above rely heavily on the same (geometric) **models**: (rigid) contacts constructed from a set of primitive ("elementary") contacts, e.g., vertex-surface, edge-surface, edge-edge [71].

The planning, monitoring and execution modules can all use statistical tools to increase their performance. The functionalities one wants to achieve by using statistical tools are described in the following paragraphs.

1.3 Functionalities

This section gives an overview of the major tasks for which statistical methods have been used or proposed in the literature. Examples for most of these functionalities are given in the rest of the text. However, this set of examples and the list of cited references are certainly not exhaustive, and many more applications are to be expected in the near future.

1. **Change detection,** also called "segmentation," [8, 9, 13, 29, 86, 90]. The goal is to detect when the sensed process undergoes a change in its state. This change can be: the mean of a sensed variable; its first difference; its variance; the model in which the sensed variable is interpreted; etc.

2. **Randomization,** or "perturbation," [5, 4, 30, 45, 46, 51, 64, 75, 84, 87]. Random motions have long been recognized as a good approach to solving

small uncertainties in assembly tasks. Moreover, random motion is a simple way to implement *active sensing*, in order to "(persistently) excite" all uncertainties in the system.

3. **Data fusion,** [3, 24, 26, 48, 60]. Information from different sensors are combined into one single outcome, with (hopefully) better resolution and reliability than could be produced by the single sensors individually.

4. **Classification,** also known as "(template/pattern) matching," "labelling" or "event/object/pattern recognition," [10, 22, 21, 20, 35, 49, 74, 83, 79]. A (spatial or temporal) sequence of sensor signals is compared to logs of previous executions, or to models of the task, in order to find the "best" correspondence.

5. **Sequence planning**, or "decision making," [2, 1, 14, 17]. If the task controller has the option to choose between different alternative subsequent subtasks, the one with highest probability is chosen. Sequence planning is often done on the basis of a lower-level classification module that assigns the probabilities to the different alternatives.

6. **Sensor selection,** [36, 48, 62]. Based on the current "state" of the system (model, parameters, past and current sensor readings), a decision is made as to which sensor to use next, and to measure which feature. Again, the choice is based on which sensing action has the highest probability for success.

7. **Tracking,** [7, 28, 34, 39, 53, 69, 77, 82, 89]. This is probably the most actively investigated aspect of stochastic sensor processing. Its goal is to estimate a set of (time-varying) parameters on line. The parameters can be time-varying, and hence also the estimation module must work on line, i.e., in the form of a *recursive filter*. This means that each new measurement is taken into account at the time it is produced, in order to give new parameter estimates as soon as possible.

2 Contact models

This text considers contacts between rigid bodies only. Of course, this is an approximation, but priority is given to simple models, and any modeling error is assumed to be "noise." In this context, a contact model has several layers: a state level, a topological level and a geometric level (Sect. 2.1). The last two layers are most intimately connected with the controller, i.e., they cope with the uncertainties in the model (Sect. 2.2). The geometric contact model layer works directly with the set-point specifications and measurements of the forces generated in a contact and the motions allowed by the contact (Sect. 2.3). Reciprocity is an important modeling concept at this layer (Sect. 2.4).

2.1 Contact model layers

Three layers can be defined in a general contact model:

1. *Contact states plus transitions.* In general, an assembly task brings the object manipulated by the robot in different contacts states, each determined by the local geometry of the surfaces involved in the contact. The transitions between these states, however, are not random, i.e., given the specified motion and force commands at a given instant in time, the robot can only switch to a limited number of other contact states (or remain in the same state). Finite state automata or Petri nets are often-used models for this kind of system behavior.

2. *Contact topology.* Contact states are classified according to what kind of geometric primitives on the contacting surfaces are involved in the contact. Typically, polyhedral contacts between rigid bodies can be modeled by simple *elementary* contacts such as vertex-surface, edge-edge, edge-surface. The topology of a contact state then consists of a list of entries (i.e., different elementary contacts acting on the manipulated object), each containing (i) the type of the contact, and (ii) (a pointer to) the two geometric features of the objects involved in the contact.

3. *Contact geometry.* At each instant in time, the actual motion freedom of the robot, as well as the set of possible contact forces that the robot can feel, are determined not only by the contact state and topology, but also by the instantaneous geometry of the contact. That is, the instantaneous value of the parameters that determine the relative position and orientation of the contacting primitives, with respect to each other and with respect to the robot end effector.

2.2 Contact uncertainties

Discrepancies between the modeled and sensed actions of the robot indicate inaccuracies in the contact models. However, not every discrepancy should immediately raise an alarm flag, indicating to the task controller that a contact state transition has taken place: indeed, an alarm is raised only if the discrepancy is *statistically* significant. Roughly speaking, two levels of contact state monitoring are described in this text:

1. *Identification of contact geometry.* A discrepancy between the modeled and sensed actions is, in

the first place, considered as a result of the time-variance of the *geometric* contact parameters, e.g. the contact normal is incorrect because one of the surfaces is curved, or because the current model parameters are not yet correct.

2. *Monitoring of contact topology.* Only when the identification module above cannot adapt the contact geometry in order to explain the measured errors (or whenever the required adaptation would become statistically too large), a possible change of contact state is flagged. Of course, nothing prohibits the parallel or serial execution of several contact transition monitors, each "specialised" in particular cases.

Identification requires a contact model that incorporates the effects of uncertainty. One such modeling approach targeted at force-controlled assembly has been presented in [16]. It relies on the concept of *virtual contact manipulators*, [88]. (Other contact modeling tools are possible, but virtual manipulators lead to important simplifications in a nonlinear Kalman Filter, see Sect. 5.1.) Each elementary contact between the manipulated object and its environment is modeled by a virtual kinematic chain that give the manipulated object the same instantaneous degrees-of-freedom as the contact, and whose "mechanical" structure passively resists all possible ideal contact forces that can be generated by the contact. Geometric uncertainties in the contact situation are then modeled by uncertain joint angles in the virtual contact manipulator. Not only contact uncertainties but any set of uncertainties between two rigid bodies can be modeled by a virtual kinematic chain; e.g. the uncertainties in how the robot has grasped the manipulated object, or the uncertainties between the end effector frame and the frame in which the force sensor expresses its measurements. One of the major advantages of the virtual manipulator contact model is that *any* set of contact situations can be modeled, even multiple simultaneous contacts (which require *parallel* virtual manipulators).

2.3 Twist and wrench spaces

It is well known that the ideal contact force is perpendicular to the surface tangent plane in a frictionless contact. However, when looking at the *six-dimensional* space of spatial forces and motions, orthogonality is not a well-defined concept anymore. The appropriate way to model these spatial velocities and forces is *screw theory*, [6, 38, 40, 50, 65, 80]. Geometrically speaking, a screw corresponds to "two vectors on a line." Screws represent rigid body motion as well as force in the following way:

1. *Motion.* The line of the screw is the line of application of the body's angular velocity. The magnitude and direction of the angular velocity are represented by a three-vector ω on the line. Similarly, the translational velocity of a reference point of the body on the line is represented by a three-vector v. The couple of parallel vectors (ω, v) is called a *twist*, [6], or *generalized velocity*.

2. *Force.* The line of the screw is the line of application of the linear force acting on the rigid body. The magnitude and direction of the force are represented by a three-vector f on the line. Similarly, the moment acting on the body is represented by a three-vector m. The couple of parallel vectors (f, m) is called a *wrench*, or *generalized force*.

Different but fully equivalent coordinate representations exist for twists:

$$t_{4\times4} = \begin{pmatrix} 0 & -\omega_z & \omega_y & v_x \\ \omega_z & 0 & -\omega_x & v_y \\ -\omega_y & \omega_x & 0 & v_z \\ 0 & 0 & 0 & 0 \end{pmatrix} \triangleq \begin{pmatrix} [\omega] & v \\ 0\,0\,0 & 0 \end{pmatrix}, \tag{1}$$

or

$$t_6 = \begin{pmatrix} \omega \\ v \end{pmatrix}, \tag{2}$$

with $\omega = (\omega_x, \omega_y, \omega_z)^T$ the angular velocity coordinate three-vector, and $v = (v_x, v_y, v_z)^T$ the linear velocity coordinate three-vector. The same physical twist or wrench has different coordinates when expressed with respect to different world reference frames, i.e., the numbers in Eq. (1) depend on the chosen reference frame. Transformation of the world reference frame, by a displacement represented by the homogeneous transformation matrix $T = \begin{pmatrix} R_{3\times3} & p \\ 0\,0\,0 & 1 \end{pmatrix}$, transforms the velocity components v and ω into

$$\omega' = R\,\omega, \quad \text{and} \quad v' = v + p \times R\,\omega. \tag{3}$$

The linear velocity three-vector v in $t_{4\times4}$ represents the velocity of the origin of the frame fixed to the moving body. The linear velocity three-vector v in t_6 is sometimes given a different interpretation, i.e., it represents the linear velocity of the point moving together with the body and *instantaneously coinciding with the origin* of the world reference frame.

Forces exerted on a rigid body can be mathematically treated in exactly the same way as velocities, i.e., by means of screws. "Force" stands for "force and torque," often also called a *wrench*, [6]. Possible coordinate representations are:

$$w_{4\times4} = \begin{pmatrix} 0 & -f_z & f_y & m_x \\ f_z & 0 & -f_x & m_y \\ -f_y & f_x & 0 & m_z \\ 0 & 0 & 0 & 0 \end{pmatrix} \triangleq \begin{pmatrix} [f] & m \\ 0\,0\,0 & 0 \end{pmatrix}, \tag{4}$$

or

$$w_6 = \begin{pmatrix} f \\ m \end{pmatrix}, \tag{5}$$

with $m = (m_x, m_y, m_z)^T$ the torque (or moment of force) three-vector, and $f = (f_x, f_y, f_z)^T$ the linear force three-vector. Wrenches transform similarly to twists, Eq. (3):

$$f' = R\, f, \quad \text{and} \quad m' = m + p \times R\, f. \tag{6}$$

2.4 Reciprocity

The space of all wrenches is a vector space, *dual* to the vector space of twists. Duality means the following: each wrench w can serve as a *linear mapping* from the space of twists to the real line:

$$w : t \mapsto w(t) \triangleq f \cdot v + m \cdot \omega. \tag{7}$$

The physical interpretation of this mapping is the *instantaneous work* done by the force on the moving body.

Often-used coordinate representations of twists and wrenches are six-vectors of real numbers, as for example t_6 and w_6 in Eqs (2) and (5):

$$\text{twist} \leftrightarrow \begin{pmatrix} \omega_x \\ \omega_x \\ \omega_x \\ v_x \\ v_y \\ v_z \end{pmatrix}, \qquad \text{wrench} \leftrightarrow \begin{pmatrix} f_x \\ f_y \\ f_z \\ m_x \\ m_y \\ m_z \end{pmatrix}. \tag{8}$$

However, the following coordinate representation is equally valid:

$$\text{twist} \leftrightarrow \begin{pmatrix} v_x \\ v_y \\ v_z \\ \omega_x \\ \omega_x \\ \omega_x \end{pmatrix}, \qquad \text{wrench} \leftrightarrow \begin{pmatrix} m_x \\ m_y \\ m_z \\ f_x \\ f_y \\ f_z \end{pmatrix}. \tag{9}$$

There is a good reason to emphasize these seemingly trivial facts: any ideal rigid body contact induces the interesting kinematic property of *reciprocity*, i.e., any motion (twist) allowed by the constraint does no work against any ideal reaction force (wrench) generated by the constraint. This means that the right-hand side of Eq. (7) vanishes:

$$f \cdot v + m \cdot \omega = 0, \tag{10}$$

or, in terms of twists and wrenches,

$$w^T t = 0, \forall w = \begin{pmatrix} f \\ m \end{pmatrix}, t = \begin{pmatrix} v \\ \omega \end{pmatrix}. \tag{11}$$

Note that this last equation is *only* valid for the given, particular *ordering choice* of the twist and wrench coordinate six-vectors. The reciprocity condition in Eq. (11) has often been incorrectly interpreted as "orthogonality," [23].

2.5 Experimental contact "models"

The basic purpose of a model is to be able to connect inputs (desired motions or forces) to outputs (measured motions or forces), or vice versa. The previous paragraphs used *explicit, geometry-based* models for this purpose. However, a "model-less" approach is possible too. Basically, this means that the model consists of the set of *measurements* in a given contact situation and with a given task specification. This gives rise to a wide range of diverse implementations, from "behavior-based" controls (the system reacts to inputs the way it reacted successfully in a previous execution of the same task), over stochastic automatons (Sect. 4), to Hidden Markov Models (Sect. 3.7).

3 Statistical tools

This section discusses the basics of the statistical tools used in the rest of this text. The emphasis is not on a rigourous mathematical treatment, but on an intuitive introduction of the major ideas. The interested reader finds more details in the cited references. But this same reader should be warned: the statistics literature is enormous, to say the least, and it's far from straightforward to objectively compare different methods, or to assess whether a particular method can be implemented efficiently or not. Moreover, innumerable variations exist for every basic statistical tool, so the following introduction certainly does not claim to be exhaustive.

3.1 Probability models

The *probability density function* (pdf) $p(y)$ of a random variable is a function that indicates the probability that the random variable has the value y. In general, a pdf can have an arbitrary shape, limited only by the fact that the area under its curve is unity, i.e., the random variable has probability one to have one of the values in the pdf. The *mean* μ and *variance* σ of a pdf $p(y)$ are classically defined as

$$\mu = \mathrm{E}(y) = \int_{-\infty}^{+\infty} y p(y)\, dy, \tag{12}$$

$$\sigma = \mathrm{E}\left((y-\mu)(y-\mu)^T\right) \tag{13}$$

$$= \int_{-\infty}^{+\infty} (y-\mu)(y-\mu)^T p(y)\, dy. \tag{14}$$

In reality, assigning pdfs to random variables faces two major problems:

1. It is very difficult to find the "real" pdf for all random variables in a system.

2. Arbitrary pdfs are difficult to do calculations with (i.e., combining pdfs, finding means and covariances) since the integrals in Eqs (12)–(13) can be very complicated.

Hence, one very often opts for one particular class of pdfs, namely *Gaussian* distributions. These distributions are efficient to work with because they are completely determined by their mean and variance:

$$p(y) = \frac{1}{\sqrt{(2\pi)^n (det\Sigma)}} e^{-\frac{1}{2}(y-\mu)^T \Sigma^{-1}(y-\mu)}, \qquad (15)$$

with y an n-vector of random variables, with vector of mean values μ, and *covariance matrix* Σ. This covariance matrix is diagonal only if all random variables are *independent*, i.e., a change in one of them does not influence the measurement of any of the other variables.

If the pdfs of the random variables in the system are not known, one has to approximate the mean and variance by the *sample mean* and *sample (co)variance* extracted from the N measurements that are available:

$$\mu = \sum_{i=1}^{N} \frac{y_i}{N}, \quad \sigma = \sum_{i=1}^{N} \frac{(y_i - \mu)(y_i - \mu)^T}{N-1}. \qquad (16)$$

One important design aspect will show up in all subsequent statistical tools: the choice of the *observation window*, i.e., the length N of the measurement sequence. Should one take into account *all* measurements since the start of the (sub)task? Should one give less weight to older measurements (e.g., if the system under observation is time-varying)? If so, how should one do the weighting?

A second important design consideration is that *many random variables don't have Gaussian distributions*. For example, a Gaussian distribution has infinite "tails" (implying non-zero probabilities for measurements with arbitrary values!) while many uncertain parameters in assembly have can *never* have values outside of well-known bounds, [60].

3.2 Whiteness test

The second most common probability distribution is the χ^2 ("chi-square") distribution. The reason of its popularity is a consequence of the popularity of the Gaussian distribution: if $y_1, \ldots, y_n$ have independent Gaussian distributions, then the random variable $\eta = y_1^2 + \cdots + y_n^2$ is "χ^2 *with n degrees of freedom*," [7, 33]. (The "degrees of freedom" of a random variable that serves as an estimator are defined as the number of observations minus the number of estimated parameters.) This text will not go into the details of what the χ^2 distribution looks like, but note that it can be used to check whether a sample of measurements has a given mean and/or variance, within statistically significant bounds. For example, a signal that contains no information (so-called "white noise") should have zero mean (its variance need not be zero, of course). Such a "whiteness" test is one of the basic tools to determine whether a sequence of measurements corresponds to a given model within statistically significant bounds. (Note that real white noise requires more than just zero mean: all frequencies should be equally present in the signal. However, checking this requires much more computations than checking the mean.) If the model is accurate, it will be successful in predicting the new measurements. This means that the difference between the predicted and measured values forms a "white noise" random variable. Tests for whiteness have existed for almost a century already, one of the most popular ones being the "Student's" test, using the χ^2 distribution on the random variable representing the sum of the squared errors between predictions and measurements, [31, 33, 42, 78, 85].

3.3 Likelihood tests

Likelihood tests compare two (or more) possible hypotheses and determine which one has the highest probability, see e.g. [7, 8, 9, 13, 29, 33, 41, 42, 44, 76, 86, 90]. The tests work with the probability distributions $p_\theta(y)$, with y a vector of measurements and θ the parameter (vector) defining the pdf of a hypothesis. Two hypotheses are tested with the *ratio* of the likelihood of both hypotheses; most often one uses the logarithm of the likelihoods (the ***log-likelihood*** $s(y)$), in order to keep the numbers within smaller ranges:

$$s(y) = \ln \frac{p_{\theta_1}(y)}{p_{\theta_0}(y)}. \qquad (17)$$

$s(y)$ is negative if hypothesis "0" is satisfied (statistically speaking), and $s(y)$ is positive if hypothesis "1" is satisfied, i.e.,

$$\mathrm{E}_{\theta_1}(s) > 0 \quad \text{and} \quad \mathrm{E}_{\theta_0}(s) < 0. \qquad (18)$$

Likelihood tests for jump detection. One of the most standard application of likelihood ratio tests is *jump detection* in the *mean* of a stochastic signal: the likelihood of a change is compared to the likelihood of no change, and the time instant where the jump took place is estimated. For example, assume the signal can belong to two classes, each with the *same* variance σ but with different means μ_0 and μ_1. The idea is to look at an (efficient and recursively computable) function of the signal during the last N measurements, and to check whether it statistically belongs to one of both expected signals. The CUSUM ("cumulative sum ") is such a function, defined as, [9, 76]:

$$S_k = \sum_{i=1}^{k} s_i, \quad \text{with} \quad s_i = \ln \frac{p_{\theta_1}(y_i)}{p_{\theta_0}(y_i)}. \qquad (19)$$

As long as the change from hypothesis "0" to hypothesis "1" has not occurred, S_k is constantly decreasing. This means that

$$g_k := S_k - m_k, \quad \text{with} \quad m_k := \min_{1 \le j \le k} S_j, \tag{20}$$

will be "small" until the change occurs. Hence, the change time (or "alarm time") t_a is given by

$$t_a = \min \{k : g_k \ge h\}, \tag{21}$$

with h a user-specified threshold. CUSUM tests with Gaussian distributions are of course very popular. For a change in the mean $\mu : \mu_0 \to \mu_1$ the CUSUM terms become

$$S_k = \frac{\Delta\mu}{\sigma^2} \sum_{i=1}^{k} (y_i - \overline{\mu}), \tag{22}$$

with $\Delta\mu = \mu_1 - \mu_0$ and $\overline{\mu} = (\mu_1 + \mu_0)/2$. This means that the signal is compared with the level halfway between both means. The coefficients in front of the summation sign make S_k dimensionless.

CUSUM functions as in Eq. (19) can be generalized for other jump detections, such as jumps in the variance, or when one or both means and/or variances are not known. However, such extensions have only scarcely been applied in the robotics literature, [29].

3.4 Markov chain and stochastic automaton

A Markov chain is a statistical generalization of the well-known concept of a state-space description of a dynamic system: (i) all (statistical) information about the system is captured in the state, and (ii) the transition from the state at instant k to the state at instant $k+1$ is determined by the transition probability matrix $p(k+1|k)$. The ijth element in the matrix gives the probability of a transition from state i to state j.

A stochastic automaton is a Markov chain whose state transition probability matrix $p(k+1|k)$ can be *adapted* on line. This "learning from experience" needs (i) an evaluation function to quantitatively assess the "goodness" of the lastly executed action, and (ii) (heuristic) functions that adapt $p(k+1|k)$.

It is obvious that Markov chains and stochastic automata are very flexible tools, of which many different varieties can be imagined.

3.5 Brownian motion

A Brownian motion is a random variable, whose increments come from independent Gaussian distributions. That means that the next state of the random variable cannot be predicted, but nevertheless the randomness of the evolution has a statistically simple model. The Brownian motion model is a popular tool to implement "randomization" applications (Sect. 4).

3.6 Kalman filter

A Kalman filter (KF) is a statistical extension of the concept of an observer (or "estimator") used in control theory, see e.g. [32, 68]. The extension consists in the fact that the calculation of the "gain" of the observer (i.e., the weighting between the contributions of (i) the state *prediction* based on the *model*, and (ii) the *update* of the state based on the *measurements*) takes place dynamically, based on the relative weights of the uncertainties ("covariances") of the state and measurement vectors. It is a classical textbook proof that, if all the variables (states, process noise, measurement noise, ...) have Gaussian distributions, the KF generates the *minimum mean square error estimate*. The *Extended Kalman filter* (EKF) uses the KF ideas for systems of random variables with *non-linear* evolution; such a linearized filter is in general *suboptimal*, at least it cannot be proven that no better filters exist.

The following paragraphs give a summary of the KF procedure, without proofs or detailed derivations. (Consult, for example, [7, 37, 55] for more details.) Let predicted variables carry a tilde ($\tilde{}$), and estimated variables a hat ($\hat{}$). Let k be the time index, x the random vector of the state to be estimated, and y the random vector of measurements. (x, y) is a Markov chain, in the sense that

$$\widetilde{x}_k = f_k(\widehat{x}_{k-1}) + v_k, \tag{23}$$

$$y_k = h_k(x_k) + w_k. \tag{24}$$

v_k is the *process noise* with covariance Q_k; w_k is the *measurement noise* with covariance R_k. The state covariance $\widetilde{P}_k$ obviously depends on the process noise only, and propagates as

$$\widetilde{P}_k = \widehat{F}_k \widehat{P}_{k-1} \widehat{F}_k^T + Q_k, \tag{25}$$

with $\widehat{F}_k$ the linearization of the state evolution function f_k. The *innovation* (or *measurement residual*) $\boldsymbol{\nu}_k$ is the difference between the actually measured output y_k and the predicted output $h_k(\widetilde{x}_k)$:

$$\nu_k = y_k - h_k(\widetilde{x}_k). \tag{26}$$

It contains the information that could not be derived from the current knowledge of the system. The covariance matrix S_k of the innovations obviously depends on the process as well as on the measurement:

$$S_k = \widetilde{H}_k \widetilde{P}_k \widetilde{H}_k^T + R_k, \tag{27}$$

with $\widetilde{H}_k$ the linearization of the measurement function h_k. (Obtaining $\widetilde{H}_k$ is in practice often the toughest job in programming a nonlinear Kalman Filter, [57].) The first term represents the influence of the state, transformed through the measurement function; the second

term is the covariance of the measurement itself. The innovation contains new information, and hence should be used to *correct* the predicted state $\widetilde{x}_k$:

$$\widehat{x}_k = \widetilde{x}_k + K_k \nu_k, \tag{28}$$

where the *Kalman gain matrix* K_k is given by

$$K_k = \widetilde{P}_k \widetilde{H}_k^T S_k^{-1}. \tag{29}$$

The interpretation of Eq. (29) is quite intuitive: the innovation (i.e., the new measurement) is given higher weight when (i) the uncertainty on the state estimate is high (i.e., large $\widetilde{P}_k$), and/or (ii) the uncertainty on the measurement is small (i.e., small $\widetilde{S}_k$, transformed to state vector space by means of the measurement function). Finally, the uncertainty of the state estimate is adapted too:

$$\widehat{P}_k = \widetilde{P}_k - K_k^T S_k K_k. \tag{30}$$

This update takes into account the previous uncertainty and the uncertainty induced by the new measurement. If the model is very good, the innovations ν_k in Eq. (26) are "small." An often-used statistical tool to quantify this "smallness" is the sample average NIS (*Normalized Innovations Squared*, [7]) which is the average of the *magnitude* of the errors over a window of the last N observations:

$$s_N^k = \frac{1}{N} \sum_{j=k-(N-1)}^{k} \nu_j^T S_j^{-1} \nu_j. \tag{31}$$

A magnitude of a statistical parameter vector is calculated with the covariance as weighting matrix. The NIS is the sum of squares of random variables. Hence, the χ^2 tests (Sect.3.2) can be applied to it in order to check whether the mean of the innovations lies indeed around zero. Note that many other statistical tests exist, using variations and/or extensions of Eq. (31); this text refers to all of them with the same term "NIS." This section ends with some closing remarks of practical importance:

1. The Kalman gain K_k is found from the *linearized* measurement function evaluated in the predicted state. Hence, an error in the predicted state makes this linearization incorrect too.

2. Kalman filters also often use a "sliding window," (or "forgetting factor," or "limited memory," [54]), in order to give less weight to older measurements.

3. Some links between Kalman filters and likelihood tests are discussed in [41, 61, 90].

4. Using a Kalman filter requires quite some a priori inputs from the user:

   (a) The dynamic model f_k.

   (b) The measurement function h_k.

   (c) The process noise covariance Q_k.

   (d) The measurement noise covariance R_k.

   (e) An initial estimate $\widehat{x}_0$.

   (f) The covariance $\widehat{P}_0$ of the initial state.

3.7 Hidden Markov Models

A contact between the manipulated object and its environment is a "hidden process," in the sense that it consists of a number of random variables (the contact forces, friction forces and the motion of the manipulated object *at the contact point*) that can only be observed *indirectly*, by looking at the random variables of measured forces and instantaneous relative velocities measured at the robot's *end effector*. Such a hidden stochastic process, observable through stochastic dependent variables, can be modeled by so-called Hidden Markov Models (HMM), [43, 81]. Also the state transitions in the contact can only be detected indirectly.

A HMM is a powerful tool for describing how stochastic signals evolve in time. The force measurements are a rich source of information for contact tasks. Notice from Figure 1 that the frequency band is broad and occurs within a short time scale of the event.

A HMM is a doubly stochastic process. Each individual discrete transition is represented by a hidden stochastic process which can not be observed directly but only through another observable stochastic process (Fig. 2). The HMM is able to estimate the behavior of the hidden process (discrete transitions) from the observable process. Motivated by Figure 1, the force signals are mapped to the frequency-domain by a Fast Fourier Transform (FFT) algorithm and used as the observable stochastic process.

3.8 Multilayer Perceptron Network

A multilayer perceptron network (MLP) is well suited for monitoring of assembly processes, especially when a number of different measurements are available. A network with three layers of nodes is used. No more than three layers are required in perceptron-like feed-forward nets because a three-layer net can generate arbitrarily complex decision regions, [66]. The network has one output node for each contact transition or state of interest. Hence, when using the network as a classifier, the contact transition or state with the largest output value is chosen.

The output of any node in layer n is given by the outputs from the previous layer. The network is fully connected, hence the output of node i in layer $n \geq 1$

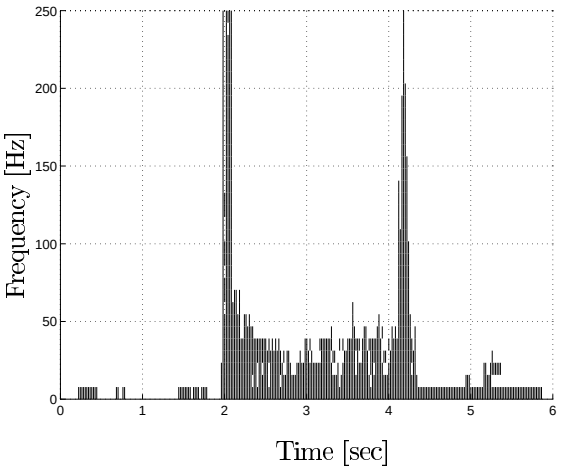

Figure 1: Spectrogram of the F_y force measurements for two transitions. The first transition (dynamic forces) starts at $t = 2.0sec$ followed by another relatively static transition at $t = 4.2sec$. The measured force was sampled at 500 Hz.

is given by

$$x_{n,i} = f\left(\sum_{j=1}^{N_{n-1}} w_{n-1,ij} x_{n-1,j}\right), \qquad (32)$$

where N_{n-1} is the total number of nodes for layer $n-1$ and w_{ij} is the network weight between node i in layer n and node j in layer $n-1$. The inputs to the nodes in the input layer are given by the sensing measurements. The training algorithm assumes the sigmoid function to guarantee that the node outputs are between 0 and 1.

The following sections present examples of applications to force-controlled assembly of the statistical tools introduced in this Section.

4 Randomization

This is probably the oldest "stochastic tool" used in robotics applications, especially for insertion tasks ("peg-in-hole"), see e.g. [5, 30, 45, 46, 56, 75, 84, 87]. Two complementary applications of randomization exist for insertion tasks:

1. *Finding the hole.* The idea is to make the robot execute a random motion ("Brownian motion") whenever the controller has no clear idea about how to move closer to its goal, or whenever the robot's positioning accuracy is worse than the tolerances required by the task. The monitoring of a randomized task is quite simple: the insertion succeeds or not, but no qualitative or quantitative information about the "convergence" of the task can be given. Except perhaps when the random motion has brought the tool "too far" from its goal, in which case the random motion can be "reflected back" towards the expected goal position, [46].

2. *Coping with friction during the insertion proper.* In assembly tasks with high friction, shaking

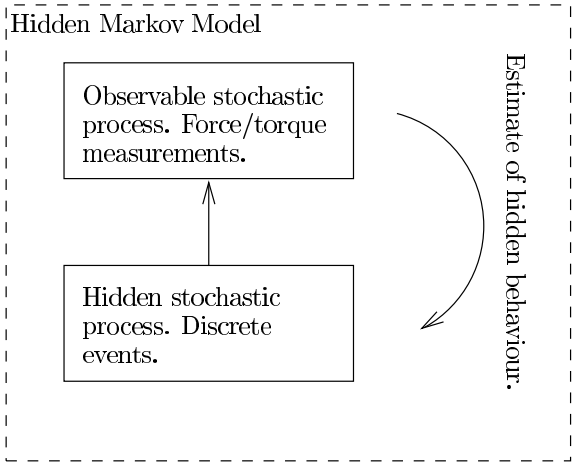

Figure 2: Illustration of a HMM. The underlying stochastic process can only be observed through another stochastic process.

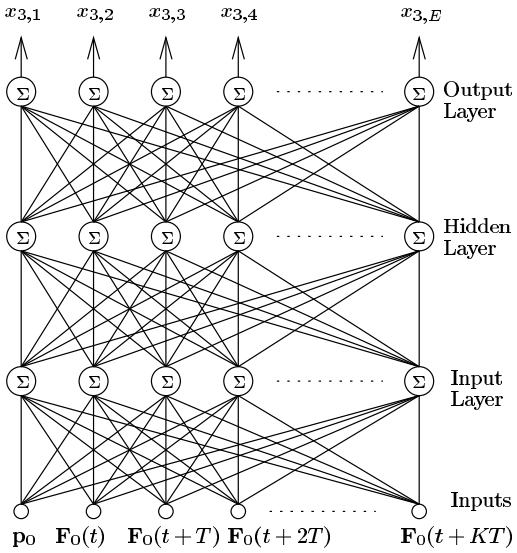

Figure 3: MLP neural network with one hidden layer. Each node in the input layer has $K+2$ input vectors. The first input vector, $\mathbf{p}_0$, contains the distance functions. The following inputs, $\mathbf{F}_0$, contain the frequency-domain force symbols at time $t = 0, T, 2T, \cdots, KT$. E is the number of discrete transitions.

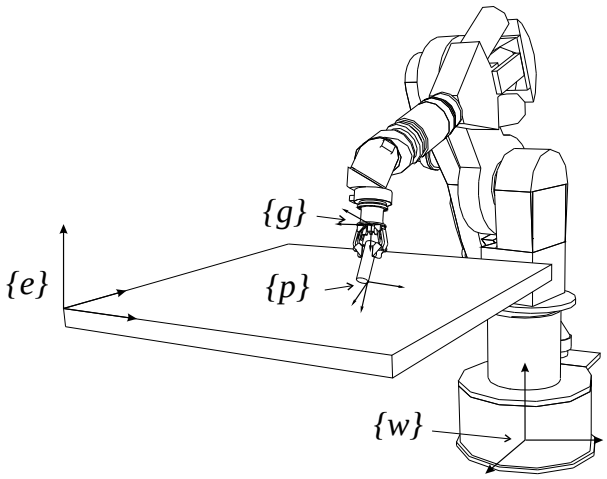

Figure 4: Robot holding a peg, and moving it over a surface. The uncertainties in the system are modeled by uncertainty in the relative position and orientation of the reference frames.

the tool at "high" frequencies (dithering) usually helps a lot, [51, 56, 87].

The random motion can be generated in hardware (e.g., by a vibrating tool, similarly in prinicple to bowl feeders) or in software.

5 Tracking

The purpose of a tracking module is to process the sensor measurements and deduce from them an estimate of a (time-varying) system parameter, see e.g. [28, 52, 53, 67, 69, 89]. In the context of force-controlled assembly, system parameters the on-line controller would want to track are:

1. *Geometric contact parameters.* If the interaction between robot tool and environment is very stiff, the instantaneous degrees of freedom of the tool with respect to the environment are appropriately modeled by, for example, a virtual contact manipulator as in Sect. 2.2. The "joint angles" of these virtual manipulators are in general time-varying, such that continuous tracking is required.

2. *Contact impedance parameters.* If there is some compliance between robot end-effector and environment, the controller can benefit from estimating the instantaneous value of this compliance. In this way, the controller can adapt its motion such that the "most compliant" route is followed. Estimating the compliance of an interaction boils down, conceptually at least, to taking the derivative of an executed incremental motion with respect to the corresponding change in force. However, since both motion and force measurements

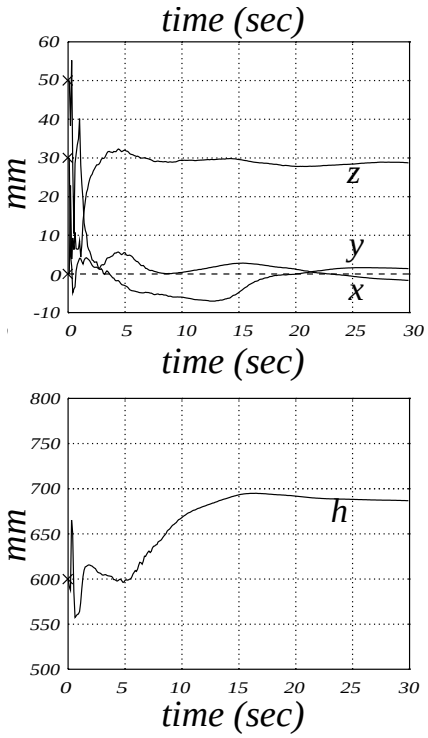

Figure 5: Parameter tracking results of an EKF that monitors the motion of the peg over the surface (Fig. 4). x, y and z are the *errors* between the initial estimates given to the EKF and the on-line estimates of the position of the contact point with respect to the robot end effector. This results corresponds to the real situation: the peg's initial length (z) was about 30mm too short. h is the estimated height of the surface with which the peg makes contact: the initial estimate of 600mm is corrected to about 700mm, which corresponds to reality quite well.

can be inaccurate and/or noisy, this direct differentiation approach is very badly conditioned. However, a statistical tool can offer a way out, [64]: (i) make the controller apply a "persistently exciting" motion on top of the nominal insertion motion; (ii) calculate the *correlation* between force and motion during each cycle of excitation; (iii) the correlation factor is a statistical estimate of the compliance.

5.1 Extended Kalman Filter for tracking

This section further describes a task in which geometric contact parameters have been tracked, using an Extended Kalman Filter (Sect. 3.6), [28]. The set-up is as in Fig. 4: the robot holds a peg, and moves it over a surface under force control. The relative positions and orientations of the peg reference frame $\{p\}$ with respect to the grasp reference frame $\{g\}$ is uncertain. Also the position and orientation of the surface (i.e., the reference frame $\{e\}$) are inaccurately known. These uncertainties are modeled by uncertainty "joints" in the virtual manipulators between robot, peg and environment surface (Sect. 2.2).

The link between these virtual manipulators and the EKF is as follows. The Kalman Filter has as its *states* the joint angles in the virtual manipulators that correspond to *geometric uncertainties* in the model. The state prediction is simple: the uncertainties do not change between two successive sample instants. The nominal evolution of the virtual contact model is straightforwardly derived from the forward kinematics routines of the virtual manipulators, especially if this same model is used to *specify* the motion. Hence, the bases of the instantaneous twist and wrench spaces of the contact are known, as well as the nonlinear measurement function h_k and its linearization $\widetilde{H}_k$, Eqs (24) and (27). Using virtual manipulators, this linearization can be derived in *closed-form*, [16, 27], which is an *enormous* advantage when using a nonlinear Kalman Filter.

The results of the nonlinear Kalman Filter are shown in Fig. 5; the robot controller made the peg rotate about its own axis according to a sinusoidal velocity profile, and made the (nominal) contact point execute another sinusoidal motion over the environment surface. Since the initial estimates are quite inaccurate and numerous, the filter needs some time to converge. (Note that no convergence properties can be *proven* in the case of a nonlinear KF.)

6 Matching

Assume the on-line controller performs model-based estimation of the parameters in the system, and these estimates converge. Assume also that an off-line model

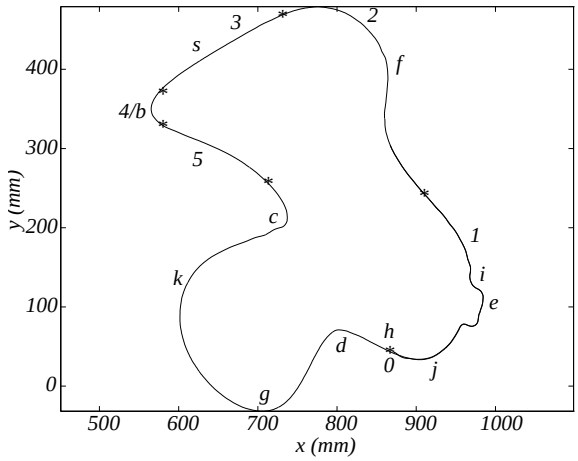

Figure 6: Outline of a planar contour, which is followed under force control with a known probe. The letters are for further reference. A "disturbance" between model and real object has been introduced at point "*e*." (Figure courtesy of S. Demey.)

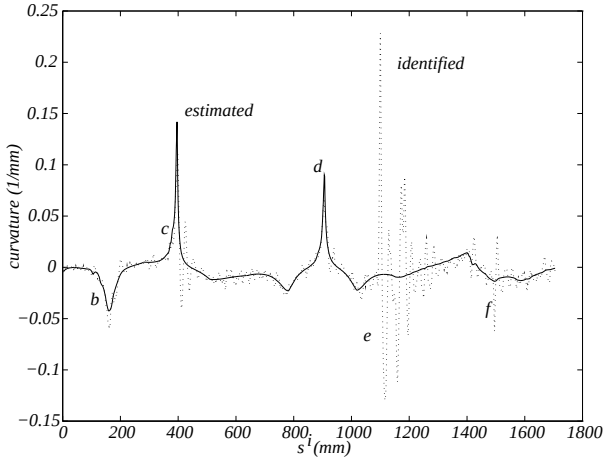

Figure 7: On-line estimates of the curvature along the contour of Fig. 6. Two curves are shown: the "identified" results, obtained by using the raw measurements, and the "estimated" curvature, resulting from applying Kalman Filter tracking to the identification data. (Figure courtesy of S. Demey.)

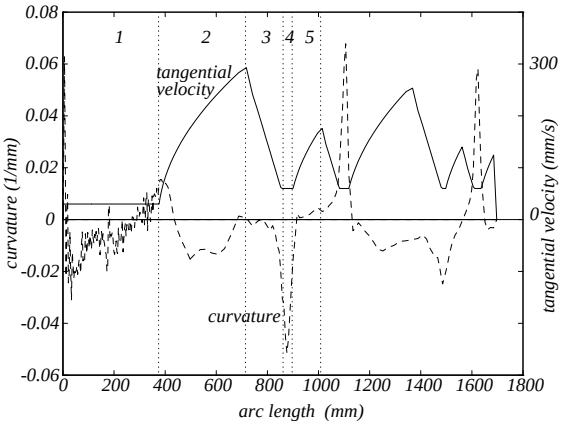

Figure 8: The matching results based on the data in Fig. 7 allow to adapt the tangential velocity along the contour: when a match is reached, the velocity can increase, until a region of high curvature (and hence badly conditioned estimates) is reached. (Figure courtesy of S. Demey.)

of (part of) the system is available. The purpose of a matching module is then to try to find the elements in the off-line model that correspond to (clustered sets of) the estimates, [11, 12, 19, 20, 21, 59, 63, 79]. In the context of force-controlled assembly, matching can improve the quality of the task. "Quality" means two things: (i) the task can be executed under the same specification with smaller force errors, and (ii) the task can be executed faster while maintaining the same level of force errors.

This section describes some experiments in which a Kalman Filter is used to match *curvature* data between a CAD model of the environment on the one hand and on-line measurements coming from moving a measurement probe with known shape around a (planar) contour on the other hand, [19, 20], (Fig. 6). Figure 7 shows curvature estimates generated by an (Extended) Kalman Filter that performs matching between measurements and model: the curvature "estimate" corresponds to the curvature value *in the CAD model* at the arc length estimated by the EKF. This EKF is constructed as follows:

1. The state of the EKF is the *shift* between the arc length parameters along the modeled curve and the actually (partially) measured curve.
2. The contour is cut in a number of spans, at the inflection points. This means that each span is either completely convex or completely concave.
3. For each span a separate EKF is started. The convexity or concavity of each span helps the convergence of the linearizations used in the EKF.

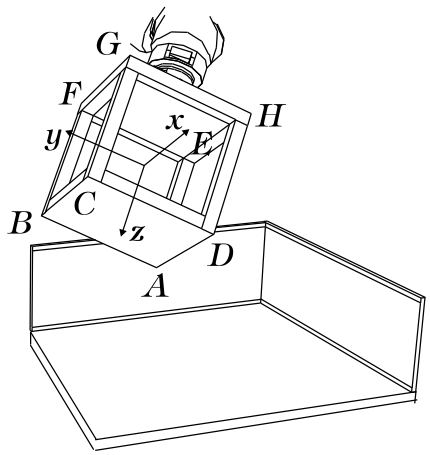

Figure 9: Set-up of a "palletizing" task. The cube is to be placed in the corner, going through a sequence of contact situations with increasing constraints.

4. The different filters that do converge are ordered according to the magnitude of the innovations, Eq. (31) in Sect. 3.6.

Figure 8 shows that the results of this EKF-based matching allows to adapt the speed of the execution to the local curvature: the robot can accelerate when a match is reached, and slows down before regions of high curvature. These regions are inherently badly conditioned for curvature estimations, since a second derivative of the position is required. Moreover, taking sharp corners at high speeds could require motions beyond the manipulator's bandwidth.

7 Transition monitoring

It is not difficult to extend an on-line filter that is used for tracking and/or matching, in such a way that it raises an alarm signal whenever it "thinks" the current model is not valid anymore. In a statistical context, the alarm should be raised as soon as the deviations between model and measurements become statistically significant. For this purpose, the CUSUM and NIS test can be used. This section describes the application of these tests in assembly tasks, [18, 19, 25, 29].

7.1 Transition monitoring with CUSUM test

The simplest force-controlled action is probably the *guarded motion*, i.e., the robot moves the manipulated object in free space until a contact is encountered, [15]. Similarly, a guarded motion can consist of moving in a "steady state" contact situation (e.g., sliding over a surface with constant relative orientation with respect to the surface) until a change in the contact situation occurs (gain or loss of contact). Figure 10 shows the raw force measurements for a *palletizing* task, [15, 25],

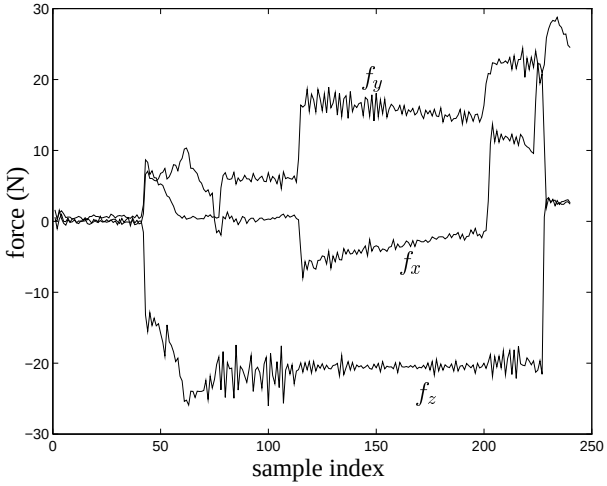

Figure 10: Force signals measured during a "palletizing" task. The transitions in the contact states are quite noticeable.

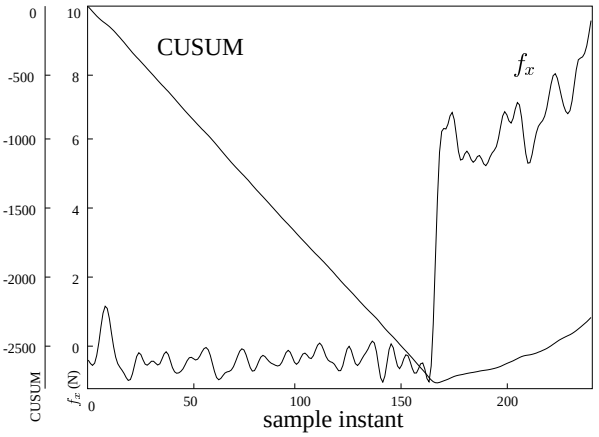

Figure 11: CUSUM test on the first part of the f_x force signal of Fig. 10. The transition (i.e. the "guarded motion" from free space to vertex-surface contact) is clearly detectable.

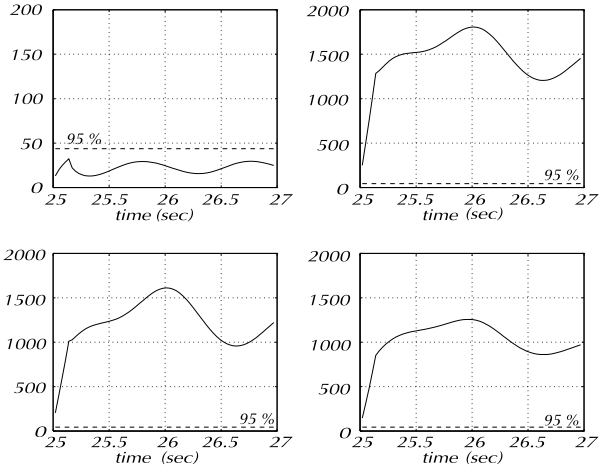

Figure 12: NIS tests on the innovations from four parallel Kalman Filters that each track the relative orientation between one of the bottom edges of the cube and the planar surface in the environment. The transition from vertex-surface contact (Fig. 13) to edge-surface contact is detected at the vertex with the correct model: only here the test remains within the statistical significance bounds.

i.e., a block is placed in a corner, by making contact with the three sides of the corner successively (Fig. 9). The transitions are rather obvious from visual inspection; however, the automation of this transition monitoring is less obvious. A basic tool to detect a transition is the CUSUM test, as introduced in Sect. 3.3, or variants described in the cited references: the measurements are constant up to noise, until an abrupt change is encountered. The "noise" is not just sensor noise, but has also components due to friction ("stick-slip") and variations in the force control action.

Figure 11 shows a typical plot for a CUSUM test, assuming Gaussian signals with known mean and variance. The threshold value used to decide when the CUSUM has "sufficiently" changed its direction (e.g., from decreasing to increasing, as in Fig. 11) influences the moment on which the "alarm flag" is raised. Making this threshold larger increases the *detection delay*; making it smaller increases the *false alarm rate*.

7.2 Transition monitoring with Kalman filter

The same "palletizing" task has been monitored by means of a Kalman Filter, using virtual contact manipulators in each of the different contact situations. As mentioned in Sect. 3.6, the innovations sequence of the KF should be "small" until the contact model changes abruptly. Figure 12 shows the KF innovations around the transition from a vertex-surface contact to an edge-surface contact.

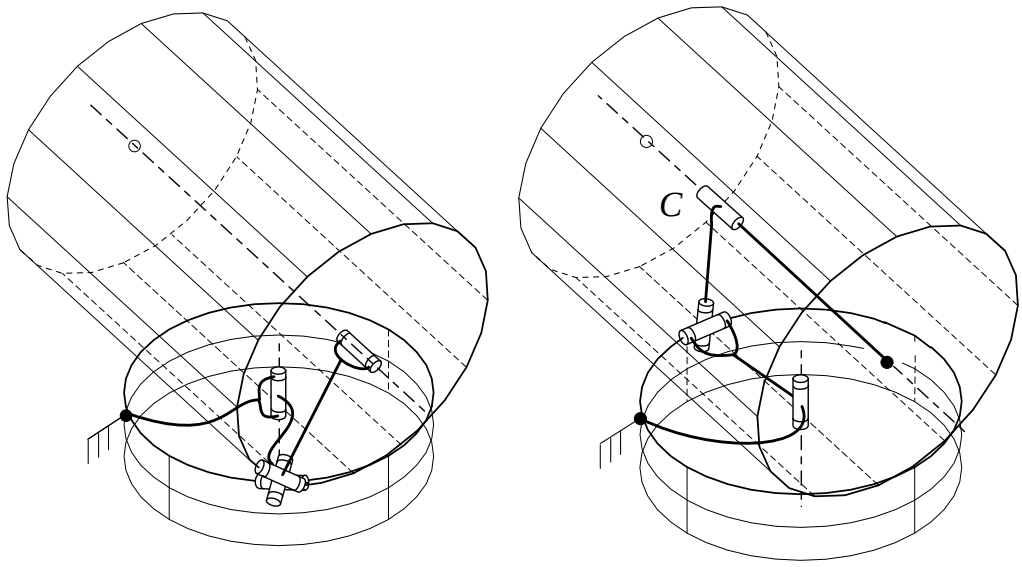

Figure 15: The "peg-on-hole" contact situation, with virtual manipulators for each of the three contact "points." (The leftmost model is used in symmetrical form for the third contact point too.)

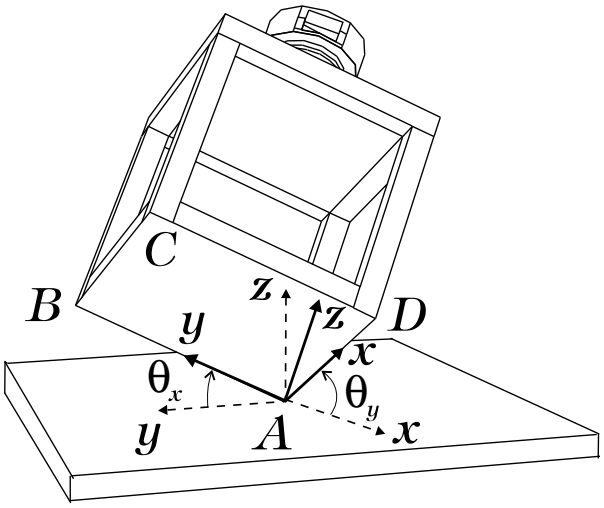

Figure 13: Vertex-surface contact during palletizing task. The angles θ_x and θ_y are to be estimated.

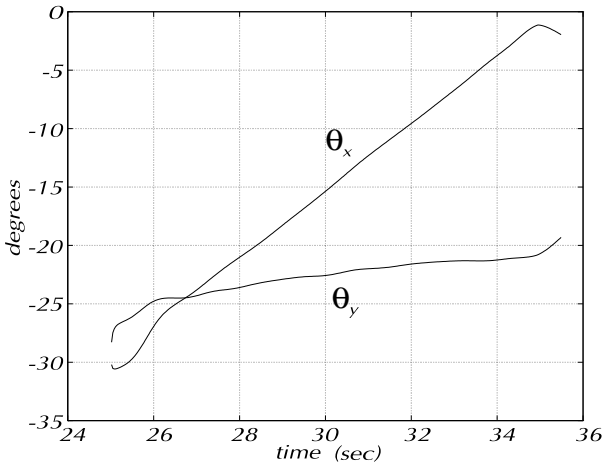

Figure 14: Kalman Filter estimates of the angles θ_x and θ_y (Fig. 13).

8 Data fusion

It is normal to expect that the use of more than one sensor increases the information available about a task, both qualitatively as quantitatively. Combining the information from different sensors (different both in location as in physical measurement principles), using their statistical properties (i.c., measurement variances), is called *sensor fusion*. This section describes a sensor fusion application during a force-controlled assembly task, using *velocity* and *force* measurements in order to estimate the geometrical parameters in the contact, [26].

The task consists of aligning the axis of a cylindrical peg with the axis of the hole in which the peg must be inserted; during the alignment motion, the bottom rim of the peg remains in full contact with the (chamferless) top rim of the hole. "Full contact" means that three "point contacts" occur, Fig. 15. This figure also shows the three virtual contact manipulators used to model the total contact situation. Each virtual manipulator has five degrees of freedom, leaving three degrees of freedom between peg and hole. The virtual joint angles of all virtual manipulators are tracked with a nonlinear Kalman Filter, similar to the

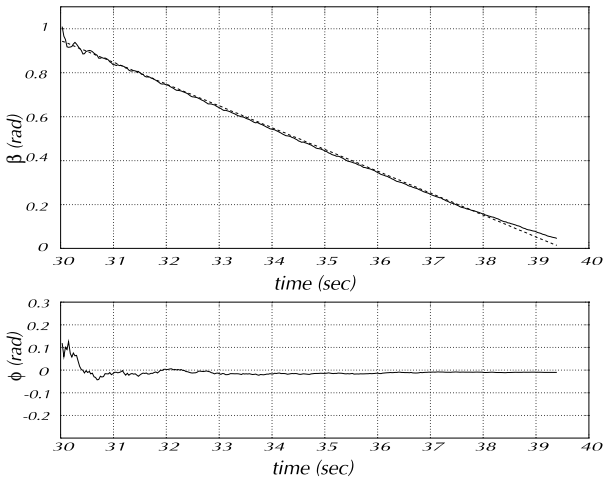

Figure 16: Top: Real (dashed line) and estimated (solid line) values of the angle β between the axes of the peg and the hole of Fig. 15 during an alignment motion. Bottom: real and estimated values of rotation angle ϕ of contact point around peg's axis.

task described in Sect. 5. The measurement vector of the KF is the concatenation of the measured twist of the manipulated object and the measured wrench on the object. Hence, the outputs of the KF (i.e., the estimates of the geometric uncertainties) are found by *fusing* force and motion information. Figure 16 shows the estimates of the alignment angle between peg and hole.

9 Transition monitoring with HMMs

A process monitor, based on recognizing contact transitions using Hidden Markov Models (HMMs) and force measurements, is also well suited to robust sensing for force controlled assembly.

The HMM approach for the monitoring of assembly is given by three steps as shown in Figure 17. The training of the HMMs is performed off line. In real-time operation the probabilities of the measured forces matching the trained models are calculated. Finally, the discrete transition with the best match is chosen.

1. **Training**—Each individual discrete transition τ_k is described by a HMM Λ_k. The model parameters of Λ_k have to be estimated from several training examples of the force signals. The parameters are estimated using the Baum-Welch re-estimation formula, see [49] for the details. The formula is an iterative solution and at each iteration the probability of the HMM matching the force signals in the training set is increased. The training is performed only once and off line.

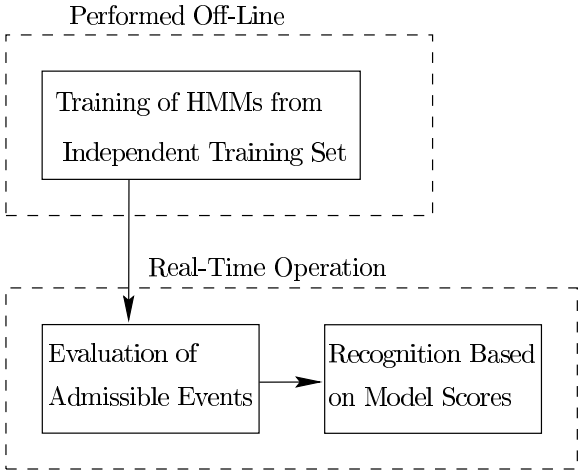

Figure 17: The HMM approach to the monitoring of assembly.

2. **Evaluation**—When a discrete transition occurs, every HMM corresponding to an admissible discrete transition is evaluated. This is called *scoring* the HMMs. The score P_k is the probability of the model Λ_k matching frequency-domain force signals resulting from the event.

3. **Recognition**—The HMM with the largest score is chosen and the corresponding discrete transition τ^* is sent to the task-level controller, i.e.,

$$\tau^* = \tau_k \quad \text{where} \quad k = \text{argmax}_k P_k. \tag{33}$$

By comparing the individual HMM scores, a confidence level of the recognised transition is calculated. The confidence levels are useful when incorporating a sensory perception controller, see section 11. A confidence level of the final decision can be defined as follows:

$$C = \frac{P_{k^*}}{\sum_k P_k} \in (0, 1] \tag{34}$$

where P_k is the individual model score for HMM Λ_k and P_{k^*} is the largest HMM model score.

The stochastic model for the hidden process (for each individual discrete transition) is shown in Figure 18. Several left-to-right models have been proposed for describing dynamic signals, see for example [81]. Note that there are several internal states q_i for each discrete transition Λ_k. The states q_i describe the internal behavior of the discrete transitions. For example, if the force signals for transition τ_k are static, then it is likely that the state of Λ_k will be q_1 most of the time, i.e., a_{11} is large. On the other hand, if the force signals are dynamic for transition τ_k, it is likely

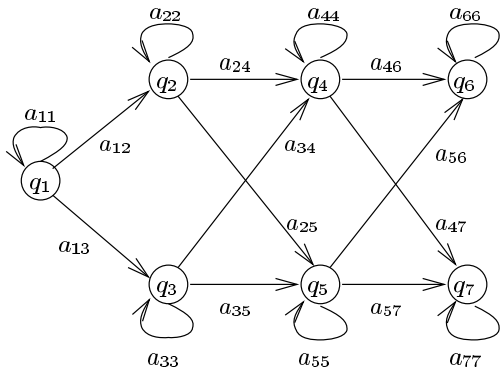

Figure 18: HMM Λ_k for each transition. q_i are the states and a_{ij} are the hidden stochastic process state change probabilities.

that the hidden state will proceed from left-to-right in the model Λ_k, i.e., a_{ii} is small, where $i = \{1, 2, \cdots, 7\}$. The estimation of the a_{ij} parameters is exactly how the force signals are used to estimate the behavior of the discrete transitions, as illustrated in Figure 2. The internal behavior of the discrete transitions is influenced by for example friction between the workpiece and the environment, sensor noise and dynamics of the manipulator. The HMM process monitor was implemented for discrete event contact recognition of an L-shaped assembly. The experimental setup consists of a 5-DOF Eshed Scorbot, and a 6-axis JR3 force/torque sensor as shown in Figure 19.

Figure 19: Eshed Scorbot 5-DOF manipulator and JR3 force/torque sensor used in the experiments.

For this particular assembly task we define 11 contact states as shown in Figure 20. If all physically possible contact state transitions are considered, we get a total of 25 discrete events.

Figure 21 shows the measured force/torque signals

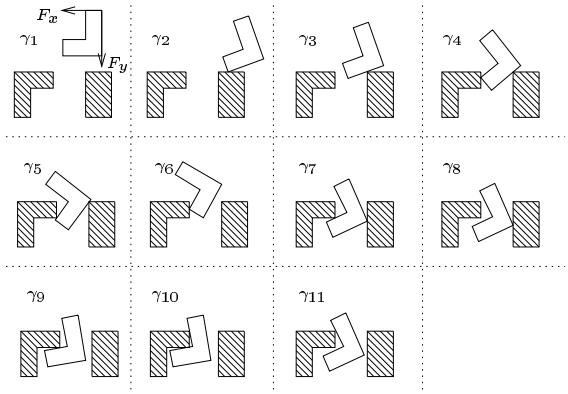

Figure 20: Model of contact states for a planar assembly task.

for a complete assembly. The training of the process monitor involves collecting several force/torque measurements, F_x, F_y and M_z for each of the 25 discrete events. Figure 21 shows the force/torque measurements for a few of these discrete events. When a discrete event occurs in real-time operation, characterised by a sudden change in at least one of the forces, all the HMMs corresponding to admissible events are evaluated. The recognised discrete event is the one with the highest HMM score, see Eq. (33).

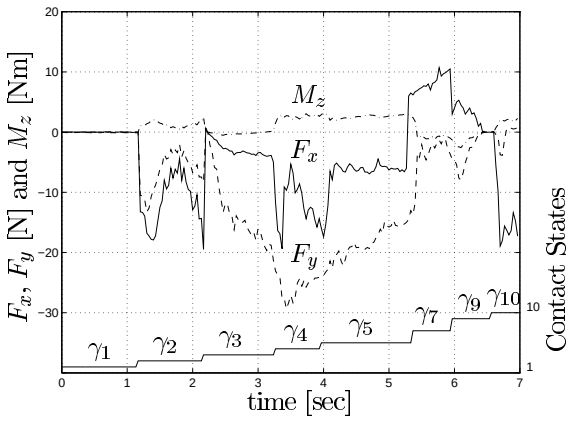

Figure 21: Force measurements for contact state sequence $\gamma_1 \rightarrow \gamma_2 \rightarrow \gamma_3 \rightarrow \gamma_4 \rightarrow \gamma_5 \rightarrow \gamma_7 \rightarrow \gamma_9 \rightarrow \gamma_{10}$.

Figure 22 shows how the successful recognition rate is a function of the training set size. Some of the contact state transitions are easy to recognise, for example $\gamma_2 \rightarrow \gamma_3$. Even with a training set size of 4 examples, the HMM process monitor successfully recognises this discrete event every time. One of the most difficult transitions to recognise is $\gamma_7 \rightarrow \gamma_9$. With a training

set size as high as 20 examples, the recognition rate is only 85%. The average recognition rate, however, is as high as 97% with a training set size of 20 examples.

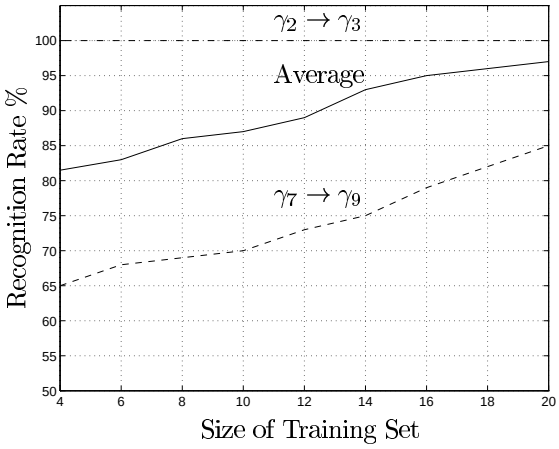

Figure 22: The performance rate vs. the training set size.

10 Fusing force and position sensing with a ANN

A method for combining dynamic force and static position measurements for the monitoring of assembly is presented. A multilayer perceptron (MLP) network is used as a classifier where the individual network outputs correspond to contact state transitions occurring during the assembly process. When a contact state transition occurs, the MLP output with the largest value is chosen. The recognised contact state is sent to a task-level controller which guides the workpiece through a series of contact states to the final desired configuration.

The proposed monitoring method combines both dynamic force and static position measurements and hence both static and dynamic behavior are modeled by the MLP. The change of workpiece position from a gain of contact continues until it is recognised as small and static position measurements suffice. The force measurements on the other hand change significantly due to friction between the workpiece and the environment, sensor noise and robot dynamics. The MLP successfully incorporates both types of sensory information.

We again use frequency-domain force measurements for two reasons. First, the frequency band is broad and occurs within a short period of time when a contact is gained or lost and hence a lot of information is available in the frequency-domain. Second, the observation symbols are easily normalised which eliminates the difference between gentle and hard gains or losses of contact.

The distance functions (position measurements) are defined as in [1]. The distance between a surface ψ and an edge ϕ, denoted by $h_{\psi\phi}$, is defined as the nearest distance between ϕ and coordinates on the surface ψ. The different surfaces and edges for the L-shaped assembly considered in this paper are shown in Figure 23.

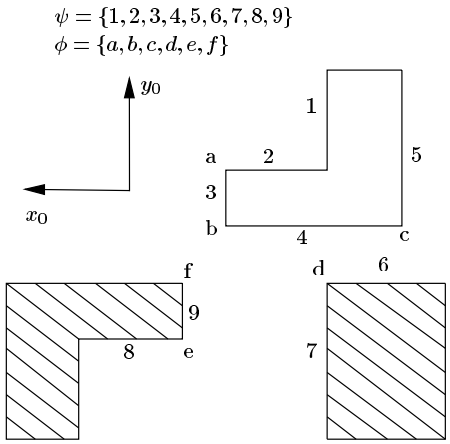

Figure 23: Labeling the surfaces and the edges of the planar assembly task.

In presence of model uncertainties and inaccurate measurements, the exact positions of the surfaces and the edges are not known. However, the distance functions still contain useful information. With a reasonably accurate geometrical model an edge and a surface with a small distance are more likely to be in contact than an edge and a surface with a large distance. By combining the position and the force measurements more reliable and robust monitoring of assembly is expected.

A multilayer perceptron network (MLP) admirably combines measurements of different types (such as position and force) for the robust monitoring of assembly. A network with three layers of nodes is used. No more than three layers are required in perceptron-like feed-forward nets because a three-layer net can generate arbitrarily complex decision regions, [66]. The network has one output node for each change in contact. Hence, when using the network as a classifier, the contact transition with the largest output value is chosen.

The output of any node in layer n is given by the outputs from the previous layer (Fig. 24). The network is fully connected, hence the output of node i in layer $n \geq 1$ is given by

$$x_{n,i} = f\left(\sum_{j=1}^{N_{n-1}} w_{n-1,ij} x_{n-1,j}\right), \qquad (35)$$

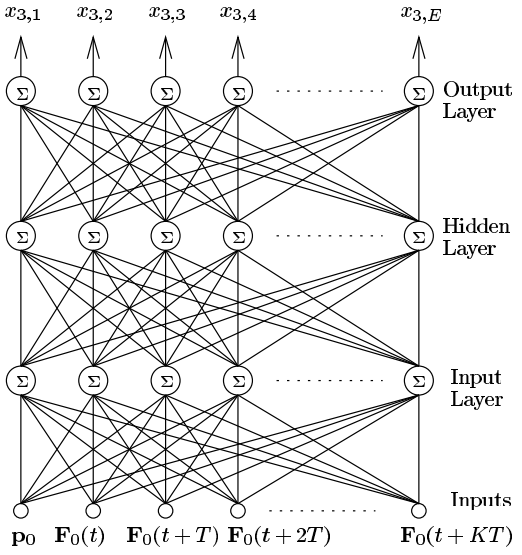

Figure 24: MLP neural network with one hidden layer. Each node in the input layer has $K+2$ input vectors. The first input vector, $\mathbf{p}_0$, contains the distance functions. The following inputs, $\mathbf{F}_0$, contain the frequency-domain force symbols at time $t = 0, T, 2t, \cdots, KT$. E is the number of discrete transitions.

where N_{n-1} is the total number of nodes for layer $n-1$ and w_{ij} is the network weight between node i in layer n and node j in layer $n-1$. The inputs to the nodes in the input layer are given by the force and position measurements:

$$x_{0,j} \in \mathcal{X}_0 = \{\mathbf{p}_0, \mathbf{F}_0(t), \cdots, \mathbf{F}_0(t+KT)\}. \quad (36)$$

Once the MLP network has been trained, it can be used quite efficiently in real-time operation. First, a change detection algorithm is needed. For assembly, a change in the constraint equations is characterised by a sudden change in at least one of the force measurements. The jump detector used here is described by the following threshold test:

$$[F_x(t) - F_x(t-\Delta T)]^2 + [F_y(t) - F_y(t-\Delta T)]^2 \geq \Delta F, \quad (37)$$

where ΔT and ΔF are design parameters. If the threshold test is able to detect gentle contacts, it will also detect brutal contacts. Hence, the design parameters are found by looking at the force measurements for gentle contacts. Of course, statistical jump detectors based on CUSUM (Sect. 7.1) or NIS (Sect. 7.2) could be used in cases with worse signal to noise ratios.

Once a discrete transition is detected, it is classified by using the MLP network previously described. Given the real-time force and position measurements, the forward pass calculates the outputs of all the network outputs $x_{3,i}$. Given the current contact state γ, only discrete transitions in the set of admissible transitions $\mathcal{E}_\gamma$ can occur. Hence, the discrete transition $\tau^* \in \mathcal{E}_\gamma$ with the largest output value is chosen.

$$\tau^* = \text{argmax}_{\tau \in \mathcal{E}_\gamma} x_{3,\tau} \quad (38)$$

The different values of the MLP outputs corresponding to the admissible transitions can be used to calculate a confidence level C of the recognised transitions τ^*, for example

$$C = \frac{x_{3,\tau^*}}{\sum_{\tau \in \mathcal{E}_\gamma} x_{3,\tau}} \in [0,1] \quad (39)$$

The confidence levels are very useful for decision making by a sensory perception controller, see Section 11.

The MLP has been successfully implemented on the five degrees-of-freedom Eshed Scorbot manipulator using a JR3 force/torque sensor (Fig. 19). The Cartesian position measurements were obtained from the manipulator's joint angles and the forward kinematics. The shape of the workpiece and the environment were known. A successful recognition rate of 92.0% was achieved on an independent test set (i.e., non-training set). The data collection time was approximately 0.6 seconds. The successful recognition rates were found (i) when both force and position were used as inputs, (ii) when only force measurements were used and (iii) when only position measurements were used. As expected, the best results were obtained for case (i), as shown in the following figures.

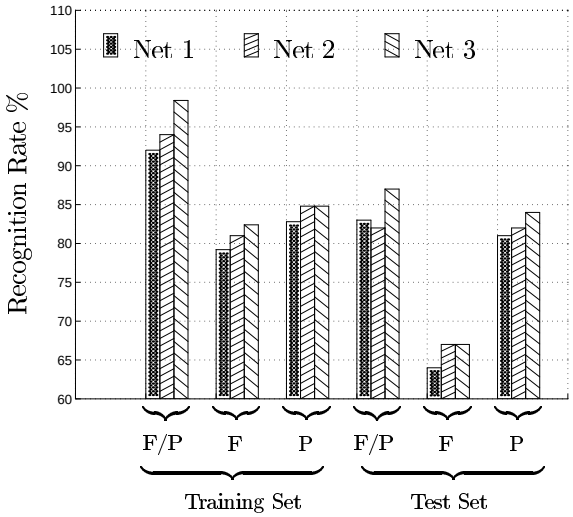

Figure 25: Recognition rates for the three different nets for both the training set and an independent test set. 10 training examples were used. F/P means that both force and position input symbols were available to the MLP while F or P means force or position only.

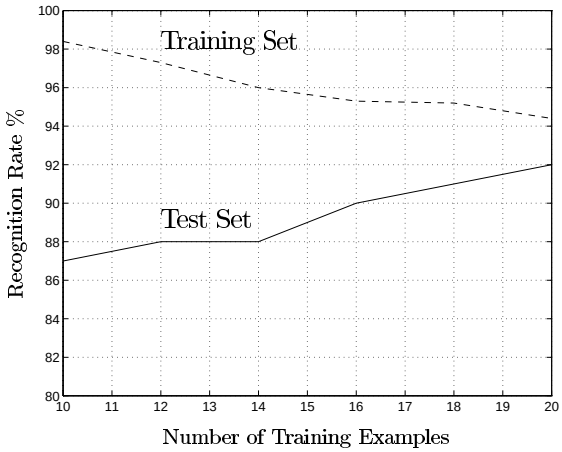

Figure 26: Recognition rates vs. number of training set examples. The MLP has 50 input nodes, 50 hidden nodes and 25 output nodes.

To increase the recognition rates demonstrated in Figure 25 more than 10 training examples are required. Figure 26 shows the recognition rates for both the training set and an independent test set as a function of the number of training set examples when Net 3 was used. The independent test set contains 4 sets of force/position measurements for each of the 25 discrete transitions which gives a total test set size of 100 examples. Note that as the number of the training set examples increases, it becomes harder for the MLP to classify the examples in the training set set, while the recognition rate for the independent test set is increasing.

One advantage of the proposed method compared to other existing solutions for recognition of contact state transitions (including the methods described in previous sections of this text) is that it models *both dynamic and static* behavior. The dynamic force measurements depend on a number of system parameters, such as workpiece and environment stiffnesses, sensor noise and dynamics, and the individual joint PID gains of the robot manipulator. These factors may vary from one contact state to another and hence help to improve the performance of the discrete transition recognition. The position measurements, on the other hand, contain mostly static information during the short time from the instant that a change in contact occurs until it is recognised by the monitor. The MLP network successfully incorporates both static and dynamic measurements.

11 Dynamic Sensor Selection with Stochastic Dynamic Programming

Dynamic sensor selection (DSS) is the process of seeking new sensory information when the current available information is inadequate. Sensors are used when there is a need for additional information. The long-term needs of a DSS system are apparent as the degree of system uncertainty and ambiguity increases. One sensor type alone may not provide sufficient information for the controllers in such systems. For systems with a high degree of uncertainty, it is desirable to collect as much sensory information as possible. However, a trade-off has to be made between sensing and control. Dynamic sensor selection aims at collecting high quality sensory information while keeping the sensory processing time low.

This section presents a method for the real-time selection of different sensor types using stochastic dynamic programming (SDP). A sensor selection controller (SSC) collects information from different process monitoring techniques. In the experiments the process monitoring uses position and force measurements for a planar assembly task like in Fig. 20. The different monitoring techniques have different characteristics and the sensor selection is based on a real-time reliability measure and the CPU time spent by each monitor. After the sensor selection is completed, the SSC selects a recognised contact transition which is sent to the controller.

A block diagram of the implemented hierarchical control system with sensory perception capabilities is shown in Figure 27. Three process monitoring techniques are available to the sensory perception controller.

Process Monitor 1 uses measurements of the manipulator's joint angles and the forward kinematics to find the 3-DOF Cartesian positions P_x, P_y and α. The recognition rate of the monitor is relatively low while the main advantage of the method is a low computational effort.

Process Monitor 2 is based on force measurements and qualitative template matching. The monitor has a reasonably high recognition rate and the computational effort is low.

Process Monitor 3 is the HMM-based force/torque method described in Section 3.7. The planar forces F_x, F_y and M_z are logged for approximately 0.5 seconds. With this substantial amount of force information, process monitor 3 is very reliable. However, the monitor requires a significant amount of computational effort to analyse the information and hence is relatively slow.

The method used for the dynamic sensory percep-

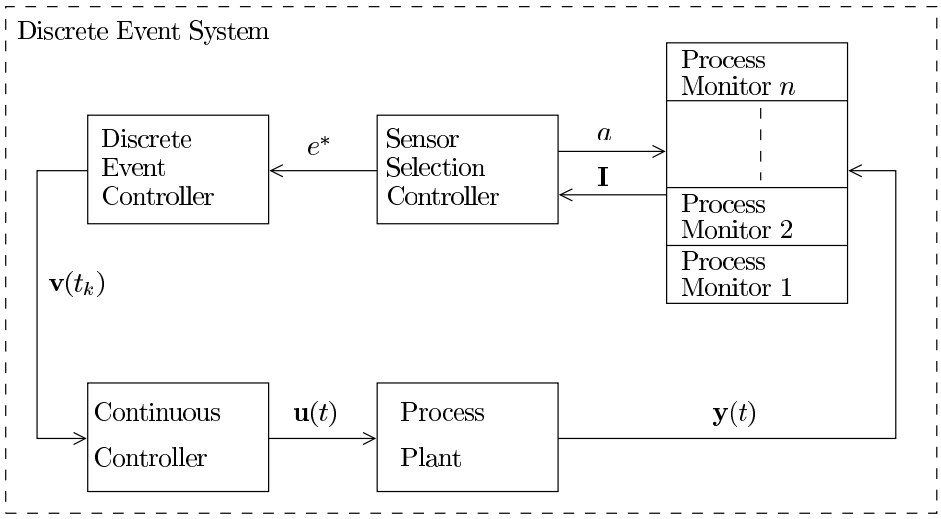

Figure 27: Discrete event system with dynamic sensory perception capabilities. $\mathbf{v}(k)$ is a vector of discrete controller commands occurring at time k, $\mathbf{u}(t)$ and $\mathbf{z}(t)$ are continuous plant inputs and outputs, $\mathbf{I}$ is a vector of symbolic process plant information, a is a command (or action) from the sensory perception controller and $\tau(k)$ is the final output from the SPC.

tion is based on stochastic dynamic programming and is described in detail in [47]. The method starts with an initial confidence level of zero and all monitors enabled. Then the sensory perception consists of two parts. First, an iterative dynamic programming (DP) algorithm evaluates all possible orderings of enabled process monitors by calculating the DP value function V. The dynamic programming model is formulated as an optimal stopping problem. At each iteration two actions are evaluated; a_1 (terminate the sensory perception), or a_2 (consult another process monitor). Second, a sensory perception controller (SPC) selects the ordering of enabled process monitors with the highest V. If the optimal action for this ordering is a_2, then the first monitor in the ordering is consulted. The confidence level output from the monitor is recorded. Next the monitor is disabled and the sensory perception problem is repeated with the new initial confidence level. The SPC terminates when the optimal action is a_1 or all monitors are disabled. The final recognised discrete event is sent to the discrete event controller.

Monitor	Rate	CPU Time
Position	79%	0.10
Template Matching	85%	0.08
Hidden Markov Models	87%	0.87
SPC	97%	0.38

Table 1: Evaluation of different process monitoring techniques and the sensory perception controller.

The performance of the SPC was evaluated from a sample set of 100 discrete events and compared with the individual process monitoring techniques. The results are given in Table 1. The table shows the average successful recognition rates and the costs of the individual process monitors. With individual average recognition rates of 79%, 85% and 87% the average recognition rate of the sensor selection controller is as high as 97% which is better than any individual process monitor. The CPU time spent by the SPC depends on the number of monitors used for each event. The average CPU time for the sample set of 100 discrete events was found to be 0.38 seconds. These results clearly show the benefits of fusing several sensing techniques for the process monitoring of robotic assembly.

The main advantage of the sensory perception control structure is an improved transition recognition rate compared to individual process monitors while the total cost is kept low. For cases where it is too expensive to use all the process monitors simultaneously, the maximum total cost can be limited and hence the proposed solution is well suited for use in real-time control systems with bandwidth requirements.

One advantage of the dynamic programming approach to the control of sensory perception is the ease with which new process monitoring techniques can be added to the system. The dynamic programming model remains unchanged, while more iterations are required by the DP algorithm to evaluate the new monitoring techniques. The real-time sensory perception controller uses lookup tables generated by the

dynamic programming algorithm. The lookup tables are generated off-line. The SPC does not require any heavy computations and uses only the lookup tables generated by the DP algorithm. Hence, the proposed method is well suited to fast real-time operation.

12 Conclusions

This workshop has introduced a number of statistical tools that can help to increase the quality of force-controlled assembly tasks. Increased quality is obtained in such "lower-level" areas as identification and tracking of model parameters, detection of contact state transitions, decision making in task sequencing, fusion of motion and force data, etc. The experiments illustrating the statistical tools clearly show this increased performance, but also indicate that they cannot be relied on blindly: a "higher-level" task supervisor is required, that can schedule sequences of actions monitored by the "lower-level" tools, and, especially, that can *reason* about their outcome and appropriateness.

Acknowledgments

HB gratefully acknowledges the scientific and practical contributions of J. De Geeter, S. Demey, J. De Schutter and S. Dutré.

References

[1] P. Astuti and B. McCarragher. The stability of discrete event systems using Markov processes. *International Journal of Control*, 64(3):391–408, 1996.

[2] D. Austin and B. J. McCarragher. Force control command synthesis for assembly using a discrete event framework. In *IEEE Int. Conf. Robotics and Automation*, pages 933–938, Albuquerque, NM, 1997.

[3] P. G. Backes. Generalized compliant motion with sensor fusion. In *IEEE Int. Conf. Robotics and Automation*, pages 1281–1286, Sacramento, CA, 1991.

[4] F. Badano, M. Bétemps, R. Clavel, A. Jutard, and M. O. Hongler. Random exploration strategy: a new paradigm in robotics. comparison with deterministic approaches. *IFAC Control Engineering Practice*, 3(9):1301–1306, 1995.

[5] F. Badano, M. Bétemps, T. Redarce, and A. Jutard. Robotic assembly by slight random movements. *Robotica*, 9:23–29, 1991.

[6] R. S. Ball. *Theory of screws: a study in the dynamics of a rigid body.* Hodges, Foster and Co, Dublin, Ireland, 1876.

[7] Y. Bar-Shalom and X.-R. Li. *Estimation and Tracking, Principles, Techniques, and Software.* Artech House, 1993.

[8] M. Basseville. Detecting changes in signals and systems—a survey. *Automatica*, 24(3):309–326, 1988.

[9] M. Basseville and I. Nikiforov. *Detection of Abrupt Changes: Theory and Application.* Information and Systems Science Series. Prentice Hall, 1993.

[10] M. Berkemeier and R. Fearing. Determining the axis of a surface of revolution using tactile sensing. *IEEE Trans. Pattern Anal. Machine Intell.*, 15(10):1079–1087, 1993.

[11] M. Betke and L. Gurvits. Mobile robot localization using landmarks. *IEEE Trans. Rob. Automation*, 13(2):251–263, 1997.

[12] R. M. Bolle and D. B. Cooper. On optimally combining pieces of information, with application to 3D complex object position from range data. *IEEE Trans. Pattern Anal. Machine Intell.*, 8:619–638, 1986.

[13] B. E. Brodsky. Asymptotically optimal methods in a problem of the fastest detection of a change point. Part 2. Studies of fastest detection methods. *Automation and Remote Control*, 56(10):1394–1401, 1995.

[14] M. L. Brown and D. E. Whitney. Stochastic dynamic programming applied to planning of robot grinding tasks. *IEEE Trans. Rob. Automation*, 10(5):594–604, 1994.

[15] H. Bruyninckx and J. De Schutter. Specification of force-controlled actions in the "Task Frame Formalism": A survey. *IEEE Trans. Rob. Automation*, 12(5):581–589, 1996.

[16] H. Bruyninckx, S. Demey, S. Dutré, and J. De Schutter. Kinematic models for model based compliant motion in the presence of uncertainty. *Int. J. Robotics Research*, 14(5):465–482, 1995.

[17] J. De Geeter, H. Van Brussel, J. De Schutter, and M. Decréton. Local world modelling for teleoperation in a nuclear environment using a Bayesian multiple hypothesis tree. In *Int. Conf. Intel. Robots and Systems*, Grenoble, France, 1997.

[18] J. De Schutter, H. Bruyninckx, S. Demey, S. Dutré, J. De Geeter, and J. Katupitiya. Force control. In C. Melchiorri and A. Tornambè, editors, *Modelling and Control of Mechanisms and Robots*, chapter 7, pages 227–264. World Scientific Publishing, 1996.

[19] S. Demey. *Shape identification and shape matching for compliant motion based on invariant differential shape descriptions.* PhD thesis, Mechanical Engineering, Katholieke Universiteit Leuven, Belgium, 1996.

[20] S. Demey, H. Bruyninckx, and J. De Schutter. Model-based planar contour following in the presence of pose and model errors. *Int. J. Robotics Research*, 16, 1997. To appear.

[21] S. Demey and J. De Schutter. Identification of second order surface geometry with a force controlled robot. In *Int. Conf. Intel. Robots and Systems*, pages 473–480, Osaka, Japan, 1996.

[22] R. O. Duda and P. E. Hart. *Pattern classification and scene analysis.* Wiley, New York, NY, 1973.

[23] J. Duffy. The fallacy of modern hybrid control theory that is based on "orthogonal complements" of twist and wrench spaces. *Journal of Robotic Systems*, 7(2):139–144, 1990.

[24] H. F. Durrant-Whyte. Consistent integration and propagation of disparate sensor observations. *Int. J. Robotics Research*, 6(3):3–24, 1987.

[25] S. Dutré. *Identification and monitoring of contacts in force controlled robotic manipulation.* PhD thesis, Mechanical Engineering, Katholieke Universiteit Leuven, Belgium, 1997.

[26] S. Dutré, H. Bruyninckx, and J. De Schutter. Contact identification and monitoring based on energy. In *IEEE Int. Conf. Robotics and Automation*, pages 1333–1338, Minneapolis, MN, 1996.

[27] S. Dutré, H. Bruyninckx, and J. De Schutter. The analytical Jacobian and its derivative for a parallel manipulator. In *IEEE Int. Conf. Robotics and Automation*, pages 2961–2966, Albuquerque, NM, 1997.

[28] S. Dutré, H. Bruyninckx, S. Demey, and J. De Schutter. Solving contact and grasp uncertainties. In *Int. Conf. Intel. Robots and Systems*, Grenoble, France, 1997.

[29] B. S. Eberman and J. K. Salisbury, Jr. Application of change detection to dynamic contact sensing. *Int. J. Robotics Research*, 13(5):369–394, 1994.

[30] M. Erdmann. Randomization of robot tasks. *Int. J. Robotics Research*, 11(2):399–436, 1992.

[31] R. A. Fisher. *Statistical methods for research workers.* Oliver and Boyd, Edinburgh, Scotland, 13th edition, 1967.

[32] G. F. Franklin, J. D. Powell, and A. Emami-Naeini. *Feedback Control of Dynamic Systems.* Addison-Wesley, Reading, MA, 3rd edition, 1990.

[33] J. E. Freund. *Mathematical Statistics.* Prentice Hall International, London, England, 2nd edition, 1972.

[34] B. Friedland. Treatment of bias in recursive filtering. *IEEE Trans. Automatic Control*, 14(4):359–367, 1969.

[35] K. Fukunaga. *Introduction to statistical pattern recognition.* Academic Press, New York, NY, 2nd edition, 1990.

[36] H. P. Gadagkar and M. M. Trivedi. Active exploration using contact and noncontact sensors: An integrated approach. In *Int. Conf. Intel. Robots and Systems*, pages 399–406, Raleigh, NC, 1992.

[37] A. E. Gelb. *Optimal Estimation.* The Analytic Sciences Corporation, 3rd edition, 1978.

[38] C. G. Gibson and K. H. Hunt. Geometry of screw systems—1. screws: Genesis and geometry. *Mechanism and Machine Theory*, 25(1):1–10, 1990.

[39] N. Gordon, D. Salmond, and C. Ewing. Bayesian state estimation for tracking and guidance using the bootstrap filter. *J. Guid. Cont. Dynamics*, 18(6):1434–1443, 1995.

[40] M. Griffis and J. Duffy. Kinestatic control: A novel theory for simultaneously regulating force and displacement. *Trans. ASME J. Mech. Design*, 113:508–515, 1991.

[41] F. Gustafsson. The marginalized likelihood ratio test for detecting abrupt changes. *IEEE Trans. Automatic Control*, 41(1):66–78, 1996.

[42] A. Hald. *Statistical theory with engineering applications.* John Wiley & Sons, New York, NY, 1962.

[43] B. Hannaford and P. Lee. Hidden Markov Model analysis of force/torque information in telemanipulation. *Int. J. Robotics Research*, 10(5):528–539, 1991.

[44] D. V. Hinkley. Inference about the change-point in a sequence of random variables. *Biometrika*, 57(1):1–17, 1970.

[45] B. Hoffmann, H. Pollak, and B. Weissman. Vibratory insertion process: A new approach to nonstandard component insertion. In *Robot 8*, pages 8.1–8.10, 1985.

[46] M.-O. Hongler, F. Badano, M. Bétemps, and A. Jutard. A random exploration approach for automatic chamferless insertion. *Int. J. Robotics Research*, 14(2):161–173, 1995.

[47] G. Hovland and B. J. McCarragher. Sensory perception and dynamic programming. In *Proceedings of the 1st Australian Data Fusion Symposium*, pages 196–201, 1996.

[48] G. Hovland and B. J. McCarragher. Dynamic sensor selection for robotic systems. In *IEEE Int. Conf. Robotics and Automation*, pages 272–277, Albuquerque, NM, 1997.

[49] G. Hovland and B. J. McCarragher. Hidden Markov models as a process monitor in robotic assembly. *Int. J. Robotics Research*, 17, 1998.

[50] K. H. Hunt. *Kinematic Geometry of Mechanisms*. Oxford Science Publications, Oxford, England, 2nd edition, 1990.

[51] S. L. Ipri and H. Asada. Force-guided robotic assembly using tuned dither and recursive parameter estimation. In *ASME Dynamic Systems and Control meeting*, pages 857–862, 1994.

[52] A. J. Isaksson, F. Gustafsson, and N. Bergman. Pruning versus merging in Kalman filter banks for manoeuvre tracking. *IEEE Transactions on Aerospace and Electronic Systems*, AFS-33, 1997.

[53] M. Isard and A. Blake. Contour tracking by stochastic propagation of conditional density. In *Proc. European Conf. Computer Vision*, pages 343–356, Cambridge, UK, 1996.

[54] A. H. Jazwinski. Limited memory optimal filtering. *IEEE Trans. Automatic Control*, 13:558–563, 1968.

[55] A. H. Jazwinski. *Stochastic processes and filtering theory*, volume 64 of *Mathematics in science and engineering*. Academic Press, New York, NY, 1970.

[56] K. Jeong and H. Cho. Development of a pneumatic vibratory wrist for robotic assembly. *Robotica*, 7:9–16, 1989.

[57] S. J. Julier and J. K. Uhlmann. A new extension of the Kalman filter to nonlinear systems. In *Int. Symp. Aerospace/Defense Sensing, Simul. and Controls*, Orlando, FL, 1997.

[58] G. Kamberova, R. Mandelbaum, and M. Mintz. Statistical decision theory for mobile robotics: Theory and applications. In *Int. Conf. on Multisensor Fusion and Integration for Intel. Syst.*, pages 17–24, Washington, DC, 1996.

[59] K. Kanatani. Analysis of 3-D rotation fitting. *IEEE Trans. Pattern Anal. Machine Intell.*, 16(5):543–549, 1994.

[60] R. Kendall and M. Mintz. Robust fusion of location information. In *IEEE Int. Conf. Robotics and Automation*, pages 1239–1244, Philadelphia, PA, 1988.

[61] T. Kerr. Decentralized filtering and redundancy management for multisensor navigation. *IEEE Trans. on Aerospace and Electronic Systems*, 23:83–119, 1987.

[62] S. Kristensen and H. I. Christensen. Decision-theoretic multisensor planning and integration for mobile robot navigation. In *Int. Conf. on Multisensor Fusion and Integration for Intel. Syst.*, pages 517–524, Washington, DC, 1996.

[63] H. Le and D. Kendall. The Riemannian structure of Euclidean shape space. *Annals of Statistics*, 21:1225–1271, 1993.

[64] S. Lee and H. Asada. Assembly of parts with irregular surfaces using active force sensing. In *IEEE Int. Conf. Robotics and Automation*, pages 2639–2644, San Diego, CA, 1994.

[65] H. Lipkin and J. Duffy. Hybrid twist and wrench control for a robotic manipulator. *Trans. ASME J. Mech. Transm. Automation Design*, 110:138–144, 1988.

[66] R. Lippmann. An introduction to computing with neural nets. *IEEE ASSP Magazine*, pages 4–22, 1987.

[67] J. Lu, N. A. Duffie, and J. G. Bollinger. Two dimensional tracing and measurement using touch trigger probes. In *Annals of the CIRP*, volume 31/1, pages 415–419, 1982.

[68] D. G. Luenberger. Observing the state of a linear system. *IEEE Transactions on Military ELectronics*, pages 74–80, 1964.

[69] B. Maqueira, C. I. Umeagukwu, and J. Jarzynski. Application of ultrasonic sensors to robotic seam tracking. *IEEE Trans. Rob. Automation*, 5(3):337–344, 1989.

[70] B. McCarragher. Petri net modelling for robotic assembly and trajectory planning. *IEEE Transactions on Industrial Electronics*, 41(6):631–640, 1994.

[71] B. McCarragher. Task primitives for the discrete event modeling and control of 6-dof assembly tasks. *IEEE Trans. Rob. Automation*, 12(2):280–289, 1996.

[72] B. McCarragher and H. Asada. The discrete event control of robotic assembly tasks. *Trans. ASME J. Dyn. Systems Meas. Control*, 117(3):384–393, 1995.

[73] B. McCarragher and H. Asada. The discrete event modelling and trajectory planning of robotic assembly tasks. *Trans. ASME J. Dyn. Systems Meas. Control*, 117(3):394–400, 1995.

[74] B. J. McCarragher and H. Asada. Qualitative template matching using dynamic process models for state transition recognition of robotic assembly. *Trans. ASME J. Dyn. Systems Meas. Control*, 115:261–269, 1993.

[75] W. Paetsch and G. von Wichert. Solving insertion tasks with a multifingered gripper by fumbling. In *IEEE Int. Conf. Robotics and Automation*, pages 173–179, Atlanta, GA, 1993.

[76] E. S. Page. Continuous inspection schemes. *Biometrika*, 41:100–115, 1954.

[77] N. P. Papanikolopoulos, B. J. Nelson, and P. K. Khosla. Six degree-of-freedom hand/eye visual tracking with uncertain parameters. *IEEE Trans. Rob. Automation*, 11(5):725–732, 1995.

[78] K. Pearson. On the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling. *The London, Edinburgh and Dublin philosophical magazine and journal of science*, Series V-50:157–175, 1900.

[79] X. Pennec and J.-P. Thirion. Validation of 3-D registration methods based on points and frames. In *IEEE Int. Conf. Robotics and Automation*, pages 557–562, Nagoya, Japan, 1995.

[80] J. Phillips. *Introducing screw theory*, volume 1 of *Freedom in machinery*. Cambridge University Press, Cambridge, England, 1984.

[81] L. R. Rabiner. A tutorial on Hidden Markov Models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[82] M. Sato and J. K. Aggarwal. Estimation of position and orientation from image sequence of a circle. In *IEEE Int. Conf. Robotics and Automation*, pages 2252–2257, Albuquerque, NM, 1997.

[83] P. Sikka and B. McCarragher. Monitoring contact using clustering and discriminant functions. In *IEEE Int. Conf. Robotics and Automation*, pages 1351–1356, Minneapolis, MN, 1996.

[84] J. Simons, H. Van Brussel, J. De Schutter, and J. Verhaert. A self-learning automaton with variable resolution for high precision assembly by industrial robots. *IEEE Trans. Automatic Control*, AC-27(5):1109–1113, 1982.

[85] Student. The probable error of a mean. *Biometrika*, 6:1–25, 1908.

[86] J. K. Tugnait. Detecting and estimation for abruptly changing systems. *Automatica*, 18(5):607–615, 1982.

[87] H. Warnecke, B. Frankenhauser, D. Gweon, and H. Cho. Fitting of crimp contacts to connectors using industrial robots supported by vibrating tools. *Robotica*, 6:123–129, 1988.

[88] H. West and H. Asada. A method for the design of hybrid position/force controllers for manipulation constrained by contact with the environment. In *IEEE Int. Conf. Robotics and Automation*, pages 251–259, St. Louis, MS, 1985.

[89] D. E. Whitney and E. F. Junkel. Applying stochastic control theory to robot sensing, teaching, and long term control. In *Int. Symp. Industrial Robots*, pages 445–457, Tokyo, Japan, 1981.

[90] A. S. Willsky and H. L. Jones. A generalized likelihood ratio approach to the detection and estimation of jumps in linear systems. *IEEE Trans. Automatic Control*, pages 108–112, 1976.