Chapter 2
Literature Survey: Trajectory Generation in and Control of Robotic Systems

This work spans and combines a wide range of research topics. In order to classify this work within the robotics research landscape, this chapter provides a survey about all adjoining fields. Starting with a short section on terminology, the states of the art in technology as well as in robotics research are surveyed.

2.1 Terminology

Before going ahead with the literature survey, some terms and words are briefly defined for the context of this book in order to prevent misunderstandings and misusages as often happens in literature.

Pose/position/orientation
A *pose* is considered *position and orientation in Euclidian space*. For all other spaces regarded in this book, for example, the joint space or any other multi-dimensional space, we only consider the term *position* for all DOFs. For example, position control in joint space takes all DOFs into account, while position control in Cartesian space only includes the three translational DOFs.

Path planning
A path is a geometric representation of a plan to move from a start to a target pose. The task of planning is to find a collision-free path among a collection of static and dynamic obstacles. Path planning can also include the consideration of dynamic constraints such as workspace boundaries, maximum velocities, maximum accelerations, and maximum jerks. We distinguish between *on-line* and *off-line* path planning algorithms. Off-line planned paths are static and calculated prior to execution. On-line methods enable the path (re-)calculation and/or adaptation during the robot motion in order to react to and interact with dynamic environments. This means that a robot moves along a path that has not necessarily been computed completely, and which may change during the movement. The

T. Kröger: On-Line Trajectory Generation in Robotic Systems, STAR 58, pp. 11–31.
springerlink.com

term *real-time path planning* is a synonym for *on-line path planning*. It would not make sense to use non-real-time planning algorithms for on-line purposes.

Trajectory planning
A trajectory is more than a path: It also includes velocities, accelerations, and jerks along the path. A common task is to find trajectories for a priori specified paths, which fulfill a certain criterion (e.g., minimum execution time). We distinguish between *on-line* and *off-line* trajectory planning methods. An off-line calculated trajectory cannot be influenced during its execution, while on-line trajectory planning methods can (re-)calculate and/or adapt the robot's motion behavior during the movement. The reasons for this (re-)calculation and/or adaptation can vary: improvement of accuracy, better utilization of currently available dynamics, reaction to and interaction with a dynamic environment, or reaction to (sensor) events. Analogous to *real-time path planning*, the term *real-time trajectory planning* is used in the same way as *on-line trajectory planning*.

Motion planning
Motion planning includes the task of *path planning* and the task of *trajectory planning*. On-line and off-line motion planning is defined analogously to on-line path planning and on-line trajectory planning.

Trajectory generation
This term is used as a synonym for *trajectory planning*.

Motion control
A robotic system interacts with its environment only by moving its kinematic chain. The control of the robot's kinematic chain is meant here. It includes trajectory-following control as well as sensor-guided motion control (e.g., force/torque control).

In particular the words *path* and *trajectory* are not consistently used in literature. The meaning of *off-line* is clear, but *on-line* and *real-time* are often used in different contexts, as will be seen in the two following surveying sections.

2.2 Overview

As depicted in Fig. 1.5 (p. 5), the field of on-line trajectory generation adjoins to (low-level) robot motion control and to (higher-level) robot motion planning, such that it exactly lies in-between these fields. Since these fields belong to the most classical ones in robotics research, we can find plenty of technological and scientific contributions within the last four decades that will be surveyed here.

Fig. 2.1 shows an overview of the treated topics and how this work is related to them. After the reviews in Sec. 2.3 and Sec. 2.4, we will explain in Sec. 2.5 how the work is related to fields depicted in Fig. 2.1.

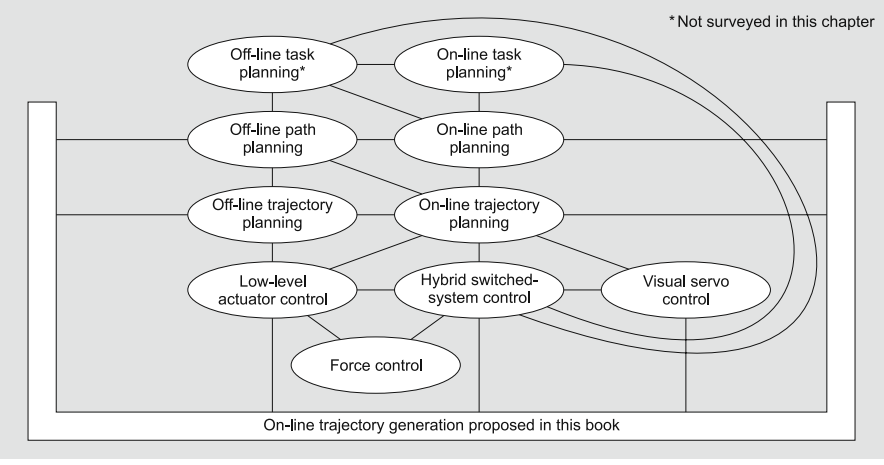

Fig. 2.1 Overview and logical coherence of areas treated in this chapter. The terminology is based on Sec. 2.1.

2.3 State of the Art in Robot Technology

Robot manufacturing companies commonly do not publish detailed information about their control concepts, technologies, methodologies, or philosophies. Hence, this section is very brief, and we predominantly focus on the scientific literature in the next section. To give an overview of the state of the art in robot technology, product specifications of some of the world's leading manufacturers of industrial manipulation robots are surveyed in the following. The text is meant to be neutral and free of bias; manufacturers are named in alphabetical order.

ABB Asea Brown Boveri Ltd. [3] applies a dynamic model in their robot controller called *QuickMove*™ and *TrueMove*™ concept [1, 2], which enable fast and accurate positioning. Sensor integration is possible through digital and/or analog input and output ports or via field bus systems, but these signals cannot be used for an on-line trajectory adaptation, but for search runs. Even when applying the force control options *RobotWare Machining FC*™ or *RobotWare Assembly FC*™ [2], the force sensor is not part of the control loop, and only intermediate path segments are adapted based on the sensor signal. Regarding pick and place tasks, ABB offers the *PickMaster*™ [4], which uses a computer vision system in order to cope with undefined part poses.

COMAU S.p.A. Robotics [60] offers the *C4G*™ robot control unit with two completely different interfaces: 1) The standard *C4G*™ [57] interface with the robot programming language *PDL2*™ designed for field applications. 2) The open interface *C4G OPEN*™ [58, 59] designed for research and development institutions. The programming language *PDL2*™ of the standard interface is a high-level Pascal-like language with typical motion planning instructions, but it does not allow for the addressing of a sensor

in the feedback control loop. Even the *sensor tracking option* only provides the possibility of modifying a pre-planned trajectory, but the robotic arm remains position controlled. Compared to this, the *C4G OPEN*™interface gives users access to the low-level servo control loops [58, 59]. An external PC communicates with the *C4G OPEN*™controller in cycles of one millisecond and can even set torque set-points, such that the user can absolutely freely develop his own control schemes. This system is excellent for research and development purposes, and users can set up sensor-guided motion control schemes having sensors in the feedback control loop.

FANUC Ltd. [79] provides options for computer vision [77] and force/torque sensor [76] integration for the *R-30iA Mate*™controllers [78]. Nevertheless, these sensors are not part of the feedback control loop; the signals are used for planning a trajectory prior to execution.

Kawasaki Heavy Industries Ltd. [123] offers robot control units named *D-Controller*™[124], which are quite closed and only accessible via the individual and plain *AS-Language*™[125]. Embedding sensors in feedback control loops is not possible in any way.

KUKA Roboter GmbH [153] also applies a dynamic model for the robot motion controller *KR C2*™[152] in order to achieve shorter cycle times and a lower trajectory-following control error. KUKA offers the possibility to embed sensor systems via digital and/or analog input and output ports as well via field bus systems [151]. One interesting system is the *Occubot*™system [150], a robot system for car seat testing. Here, forces and torques are measured in six axes during one single test cycle, and these measured values are used to adapt the trajectory of the next test cycle, such that a desired force can be exerted after a certain number of test cycles.

The robotics division of Mitsubishi Heavy Industries Ltd. [186] provides robot painting systems [185], which are generally not open for any kind of sensor integration. Mitsubishi offers a so-called *open architecture*, which enables customers to interface the robot control unit in time steps of seven milliseconds. Reseller companies such as Battenberg Robotic [18] use this interface for individual add-on controllers.

MOTOMAN Inc. [189] provides the MOTOMAN *NX100*™[190] controller for their industrial robots. Neither the previously mentioned manufacturers nor the *NX100*™of MOTOMAN Inc. provide the feature of embedding a sensor in the feedback control loop [191].

The *Kantana*™manipulator [192] of the Neuronics AG [194] is based on the ideas of using mechanical components of very light weight, of using low-power drives, and of mounting foam around all feather-edged parts of the arm, so that the system can be used in human environments without any further protection mechanisms. The programming interface [193] is very open, such that users can develop their own control schemes, but the standard system does not offer possibilities of integration in the feedback control loop.

Stäubli Faverges SCA [250] offers robot control units, for example, the *CS8C*™controller [248], with digital and/or analog inputs and outputs,

which can be used for the retrieval of sensor signals. But as also stated for all other commercial robot manufacturers, it is not possible to put a sensor into the feedback loop of the controller [249]. An interesting option offered by Stäubli is the *Low Level Interface LLI*™ [247], which is comparable to the solution *C4G Open*™ of Comau Robotics. Pertin et al. describe in [207] how the *LLI*™ can be used as an interface for individual control schemes. The *LLI*™ accepts position and velocity set-points as well as velocity and torque values for feedforward control.

2.4 State of the Art in Robotics Research

Fig. 2.2 indicates a first brief classification of this work with regard to adjoining fields, which are surveyed in the following three subsections.

General overviews of *all* fields can be found in some robotic textbooks. The Springer Handbook of Robotics edited by Siciliano and Khatib [240] includes a chapter about off-line motion planning written by Kavraki and LaValle [122] and a chapter about on-line motion planning written by Brock, Kuffner, and Xiao [36]. A very recent and also very excellent textbook on on-line (but mostly off-line) trajectory generation concepts was written by Biagiotti and Melchiorri [24]; in particular the part [25] contains fundamental methods that are also applied in this work. Basic concepts of robot motion control were formulated by Chung, Fu, and Hsu [55]. Shorter summaries are given in the textbooks of Craig [62], of Fu, Gonzalez, and Lee [94], and of Spong, Hutchinson, and Vidyasager [246]. In general, it is of course a voluminous task to classify the whole scientific literature in this field, because we can find contributions from four decades ago and almost every work considers different assumptions and basic conditions, but this classification has to be done in order to place this work correctly within this wide field of literature.

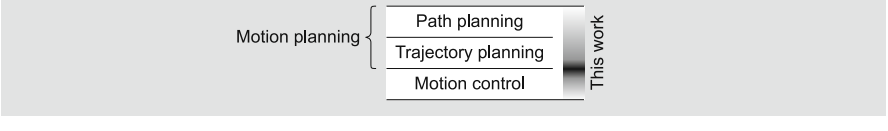

Fig. 2.2 Brief classification of this survey based on the terminology of Sec. 2.1.

2.4.1 Path Planning

Off-Line Path Planning Methods

Off-line path planning methods are not directly related to this work, but as on-line path planning concepts are often based on off-line ones, we refer to a selection of surveys and books: Lozano-Pérez [169], Lindemann and LaValle [166], González-Baños, Hsu, and Latombe [101], and LaValle [156].

On-Line Path Planning Methods

On-line path planning is generally only reasonable if paths *and* trajectories are considered. Hence, we talk about *on-line motion planning* or — to use a new eligible term for this field — real-time adaptive motion planning (RAMP, [266]). This field of research is very young and publications are rare. There are many works assuming dynamic but exactly known environments, which is not meant here. We assume robots that have to act in a dynamic and/or unknown environment, and which are equipped with sensor systems to react to and interact on-line with (unknown) static or dynamic obstacles, events, or (abrupt) changes of task parameters. As a result, RAMP takes place in the *configuration*×*time* (CT) space.

The research groups of Brock and Xiao focus on real-time capable methods for (multi-)robot motion planning. The use of splines is one very common method to represent calculated trajectories; it is the task of a motion planning algorithm to calculate respective sets of spline knots and trajectory parameters during runtime. [266] proposes an on-line motion planning approach; paths and trajectories are calculated on-line in CT-space, such that the system can act in unknown dynamic environments. Yang and Brock [280] use collision-free vertices ("milestones") and edges on a roadmap, which is another kind of representation, to represent currently planned trajectories. These knots or milestones are generated from an overall view onto the robotic system and its environment — it is in particular a kind of motion planning from a global point of view. The PhD thesis of Brock from 1999 [33] as well as follow-up works published together with Khatib and Kavraki [34, 35] introduced the elastic strip framework. The basic idea of this work is to locally deform a previously planned trajectory in order to avoid moving obstacles inside a collision-free "elastic tunnel", such that the robot can move task-consistently from the initial to the desired target pose in a three-dimensional workspace.

Jaillet and Siméon [118] on the contrary use rapidly exploring random trees (RRT) as a local planner to update a global roadmap, which was originally represented by a probabilistic road map (PRM). This way, arbitrarily moving obstacles can be also considered during runtime. Another method, which is based on a potential field approach, avoids unknown and dynamic obstacles and was presented by Ögren, Egerstedt, and Hu [195]. In the work of Mbede et al. [177], a neuro-fuzzy controller is used to locally modify the base motion of a mobile manipulator in order to avoid a moving obstacle. Based on the contribution of Mbede et al., Merchán-Cruz and Morris [180] published a work about motion planning for cooperating manipulators in 2006. Li and Latombe [161, 162] presented works on an on-line planner for the special purpose of planning the motions of two robot arms taking arbitrarily positioned parts from a conveyer belt.

2.4.2 Trajectory Planning Concepts

Off-Line Trajectory Planning Methods

Even if off-line trajectory planning is not a subject in this work, many on-line trajectory planning concepts are rooted in ideas from off-line concepts. Roth and Kahn [120] belong to the pioneers in the field of time-optimal trajectory planning. In 1971, they published methods of optimal, linear control theory and achieved a near-time-optimal solution for linearized manipulators. The resulting trajectories are jerk-limited and lead to smaller trajectory-following errors and to less excitation of structural natural frequencies in the system.

In 1982, the work of Brady [29] introduced several techniques of trajectory planning in joint space and Paul [203, 204] and Taylor [256] published works about the planning of trajectories in Cartesian space in parallel to Brady. Lin, Chang, and Luh [165] published another purely kinematic approach in 1983 as did Castain and Paul [43] in 1984 and Chand and Doty [44] in 1985.

In 1984, one of the early publications of Hollerbach [114] first introduced the consideration of the nonlinear inverse robot dynamics for the generation of manipulator trajectories. The aim here is to exhaust the maximum actuator forces and/or torques as well as possible in order to achieve shorter execution times. The basic idea is to represent the path as well as the trajectory by a set of parametric functions, which are employed in the dynamic model of the manipulator. This way, the optimization problem can be described for an arbitrary number of DOFs. Later works of Hollerbach's group are based on this idea [10, 11, 223].

During the middle of the 1980s, three groups developed techniques for time-optimal trajectory planning for arbitrarily specified paths: Bobrow, Dubowsky, and Gibson [27, 28], Shin and McKay [238, 239], and Pfeiffer and Johanni [208]. Based on the approach of Hollerbach, that is, describing the robot dynamics of dependence on a parametric path representation, a maximum acceleration can be calculated for each point of the trajectory. These maximum acceleration values correspond to maximum actuation forces and/or torques. Furthermore, maximum actuator velocities are taken into account, such that a characteristic curve for the maximum velocity can be calculated. This so-called maximum velocity curve describes the maximum velocity for each trajectory point and has to be regarded during the trajectory planning phase, that is, the time-optimal trajectory must avoid crossing the limit curve in order to minimize execution time. A time-optimal trajectory can be found by determining the switching points between positive and negative maximum acceleration values, which have been calculated beforehand. The algorithms of the groups around Bobrow, Pfeiffer, and Shin differ in their way to find these switching points.

Independently from these three groups, Rajan [218] presented a spline-based approach for minimum-time *motion* planning (cf. Fig. 2.2). Here an initial and a goal position are given in configuration space; path *and* trajectory

are calculated and represented by cubic splines. Geering, Guzzella, Hepner, and Onder [99] discuss the same topic and extend it to different types of robot kinematics.

Shiller and Dubowski [235] used *B-splines* as extension of the basic algorithm of Bobrow, Pfeiffer, and Shin as did Takayama and Kano [254] later. A B-spline curve is an extended version of Bézier curves that consists of segments, each of which can be viewed as an individual Bézier curve with some additions; a detailed introduction can be found in [212].

In 1988, Kyriakopoulos and Saridis [154] extended the basic trajectory planning approach of Bobrow/Pfeiffer/Shin by introducing a minimum-jerk criterion in order to achieve a better trajectory-following behavior. Tan and Potts [255] first transferred the ideas of Bobrow, Pfeiffer, and Shin to a discretely working system that was based on a discrete dynamic model. One year later, in 1989, Slotine and Yang [244] proposed an alternative algorithm for the works of Bobrow, Pfeiffer, and Shin. To abandon the computationally expensive calculation of the maximum velocity curve, a method for the analytical calculation of a limit curve is proposed, and conditions for characteristic switching points on this curve are determined in order to achieve a time-optimal trajectory. In the same year, Olomski [196] presented detailed experimental results and used the off-line generated trajectories as input values for different trajectory-following control schemes.

At the same time, Chen and Desrochers [49, 50, 51] proved that, as long as only dynamic constraints are considered, a time-optimal trajectory can only be achieved if at least one actuator permanently runs in saturation. McCarthy and Bobrow [178] present a similar proof but for the more general case of robots with arbitrary amounts of DOFs.

At the beginning of the 1990s, Shiller and Lu [236, 237] extended the algorithm of Bobrow, Pfeiffer, and Shin and considered dynamic singularities. A dynamic singularity is a part of the trajectory, at which at least one actuator does not contribute to the acceleration along the path.

Apart from the works mentioned above, Simon and Isik [243] presented a concept using trigonometric splines in 1993. The basic idea is to connect knots in joint space that have previously been calculated by the inverse kinematic model, with parameterized trigonometric functions. Some of the parameters can be chosen to minimize the jerk, which results in a very smooth and harmonic motion in joint space.

In 1994, Dahl [63] improved the basic algorithm of Shiller and Lu, such that it becomes more compact. Furthermore, Dahl presented more advanced experimental results. Two years later, Fiorini and Shiller [89] presented a further off-line algorithm for known dynamic environments. It is based on the assumption that the environment is completely known beforehand, that is, all static objects are known, and all moving objects are known with known trajectories.

Also in 1994, von Stryk and Schlemmer [252] evolved experimental results for minimum-time and minimum-energy trajectories for a Manutec r3

industrial robot. The research group of von Stryk published further works on trajectory optimization methods, for example, [109]. In 1996, Žlajpah [270] slightly extended the basic approach of Bobrow, Pfeiffer, and Shin by embedding task constraints. The original algorithm becomes redefined and different areas below the maximum velocity curve are defined and taken into account during the calculation of the switching points.

Based on Lie groups and Riemannian geometry, which were applied together in research on robotics dynamics by Park, Bobrow, and Ploen in 1995 [201], Žefran, Kumar, and Croke [269] suggested an off-line method for the generation of task space trajectories from an initial to a target pose (both with zero velocities); this is an approach that considers multiple DOFs. The key contribution of this work is a measure for the smoothness of a *multi-dimensional* trajectory that is supposed to be optimized by the proposed algorithm. Furthermore, dynamic system constraints such as boundary velocity and acceleration curves are considered.

A method for the generation of minimum-jerk trajectories, but without consideration of robot-dynamics, was contributed by Piazzi and Visioli [209, 210] in 1998 and 2000. In a follow-up work from 2002, Bianco and Piazzi embedded robot dynamics into this approach [26].

A general overview of basic off-line trajectory planning concepts is presented in the textbook of Khalil and Dombre [126]. A more recent approach of Lambrechts, Boerlage, and Steinbuch from 2004 suggests the generation of very smooth trajectories, whose jerk derivatives are limited as well [155], but there, only one DOF-systems are considered, and the method requires initial and goal velocities of zero.

On-Line Trajectory Planning Methods

An on-line modification of a planned trajectory may have several reasons: 1) The trajectory becomes adapted in order to improve the accuracy with a path specified beforehand; 2) The robotic system reacts on sensor signals and/or events that cannot be predicted beforehand, because the robot acts in a (partly) unknown and dynamic environment. These two reasons are rooted in the field of robotics, but there is another field, in which on-line trajectory planning plays an important role: CNC machine tools. The following three paragraphs survey these fields.

Improving Path Accuracy

All previously described off-line trajectory planning methods assume a dynamic model that describes the behavior of the real robot exactly. In practice, this is often not the case, and some robot parameters are only estimated, some dynamic effects remain unmodeled, and system parameters may change during operation. If this is the case, the resulting robot motion is not time-optimal anymore and/or the maximum actuator forces and/or torques are

exceeded, which leads to an undesired difference between the specified and the executed path. In the following, we give a survey of methods that adapt the trajectory on-line in order to improve the path accuracy.

In 1989, Dahl and Nielsen [64] suggested an on-line trajectory adaptation method. The approach is based on the basic algorithm of Bobrow/Pfeiffer/Shin, and adapts the acceleration along the path, and furthermore the parameters of the trajectory-following controller are adapted on-line, such that the underlying trajectory-following controller becomes adapted, depending on the current state of motion.

Independently from Dahl, van Aken and van Brussel [265] proposed a concept that uses one-dimensional parameterized acceleration profiles along the path in joint space instead of adapted splines. These profiles are computed on-line under consideration of the dynamic model of the manipulator.

Also independently from the two latter works, Bestaoui [23] proposed another algorithm for on-line trajectory generation at the same time. This method replans the trajectory based on state-dependent acceleration and velocity constraints if an intermediate trajectory knot is attained, or if the difference between the real and the desired trajectory is greater than a certain threshold.

The approaches of Cao, Dodds, and Irwin from 1994 [41] and 1998 [42] use cubic splines to generate smooth paths in joint space with time-optimal trajectories. A cost function is used to define an optimization problem that considers the execution time and the smoothness of the path. The cost-function can be minimized during the motion by applying the efficient numerical quasi-Newton method proposed by Davidon, Fletcher, and Powell [65, 92]. Thus, new intermediate knots are calculated. Robot dynamics are only considered by a set of parameters that must be determined empirically.

The works of Bazaz and Tondu [20, 21] (1997, 1999) extend the work of Bestaoui [23] and use segments of cubic splines to interconnect trajectory segments instead of fourth-order polynomials to describe the trajectory.

In 1998, Schlemmer and Gruebel [224] suggested an on-line method that embeds a static environmental model, such that the optimization problem becomes extended. Not only the minimum time criterion is applied but also a further function representing the distance between the robot and its environment is taken into account. Hence, the aim is to find a compromise trajectory represented by cubic splines that minimizes the execution time and that maximizes the distance to the environment.

Constantinescu and Croft [61] suggest a further improvement to the approach of Shiller and Lu [234, 236, 237] in the year 2000. The additional limitation of the derivative of actuator forces/torques leads to a limitation of the jerk in joint space and thus to smoother trajectories with better following behavior.

The PhD thesis of Pietsch [214] from 2003 is based on the method of Dahl and Nielsen and improved the concept by combining adaptive control and time-optimal trajectory planning.

In the same year, Macfarlane and Croft presented a jerk-bounded, near-time-optimal, one-dimensional trajectory planner in [174] that uses quintic splines that are computed on-line. Similar to the approach of Andersson [13, 14] from 1988, which will be described in the next paragraph, Macfarlane and Croft use quintic polynomials. As an extension to Andersson, they proposed an additional algorithm, which guarantees that these polynomials will perform a limited jerk along the path. Experimental results underline the usability of this approach, but robot dynamics are not considered.

Together with Owen and Benhabib, Croft publishes a further work on on-line trajectory planning [199, 200]. Here, a cooperating robot task is considered. An off-line planned trajectory becomes adapted on-line in order to respond to unmodeled disturbances and to maintain the desired path.

A work of Kim et al. from 2007 [130] implies improving the algorithm of Macfarlane and Croft [174] by taking robot dynamics into account, but only simulation results are shown.

Sensor-Based Trajectory Adaptation

The last paragraph presented an overview of on-line trajectory generation methods for improving the path accuracy, while this one focuses on the on-line consideration of sensor signals.

The works of Andersson [13, 14] from 1988 and 1989 present a Ping-Pong-playing PUMA 260 manipulator, whose trajectory is generated on the basis of a three-dimensional stereo-vision system. The position progression of a trajectory is represented by quintic polynomials that are parameterized by Andersson's algorithm. One particular property is that the algorithm works with arbitrary initial and goal conditions, in particular target velocities unequal to zero.

In 1993, Lloyd and Hayward [168] proposed a technique to perform transitions between two different path segments. On the one hand, the manipulator dynamics are considered based on the basic trajectory generation approach of Bobrow/Pfeiffer/Shin [27, 28, 208, 238, 239], and on the other hand the transition window technique first proposed by Paul [205] and Taylor [256] is applied. The two path segments, which are to be blended together within the transition window, need not be known in advance. The trajectory-describing blend function for the transition window is calculated on-line and considers acceleration and velocity bounds.

Inspired by human arm movements and based on the works of Lloyd and Hayward, the research group of Flash [98, 222] published further works in the same field in 1999 and 2001. Their superposition principle incorporates position, velocity, and acceleration constraints. As in the method of Lloyd and Hayward, the target position in joint space can change arbitrarily, and within a certain superposition window, the current trajectory "fades out" while the new one "fades in."

In 2002, Liu [167] presented a one-dimensional approach that calculates linear acceleration progressions on-line by parameterizing the classic seven-segment acceleration profile [43]. This way, the system can react to arbitrary target position switchings at any time. Liu's approach is strongly related to a part of this work, but as will be described in Chap. 4, it is incomplete and erroneous.

Two years later, in 2004, Ahn, Chung, and Youm [5] proposed a work for the on-line calculation of one-dimensional trajectories for any given state of motion *and* with arbitrary target states of motion, that is, with target velocities and target accelerations unequal to zero. Sixth-order polynomials are used to represent the trajectory, which is called arbitrary states polynomial-like trajectory (ASPOT). The major drawback of this work is that neither kinematic nor dynamic constraints can be regarded for on-line trajectory generation, such that this method cannot be applied in practice.

In a work from 2005, Chwa, Kang, and Choi [56] presented an advanced visual servo control system, whose purpose is to intercept fast and arbitrarily-moving objects with a robot. The proposed on-line trajectory planner considers the system dynamics of a two-link planar robot and works in two DOFs. A drawback of this work is its missing generality, and only simulation results are shown.

The title of the work of Kang [121] implies the description of an on-line trajectory planning method, but only basic and well-known kinematic and dynamic principles of robotics are presented.

Broquère, Sidobre, and Herrera-Aguilar [38] published a work in 2008 that uses an on-line trajectory generator for an arbitrary number of independently acting DOFs. The approach is very similar to the one of Liu [167] from 2002 and is based on the classic seven-segment acceleration profile [43], but this approach is unfortunately also incomplete (cf. Chap. 4) and can only perform reactions if the current acceleration value of a DOF is zero. An additional very recent work of Haschke, Weitnauer, and Ritter [108] presents an on-line trajectory planner in the very same sense as this book does. The proposed algorithm generates jerk-limited trajectories from arbitrary states of motion, but it suffers from numerical stability problems and only allows target velocities of zero.

On-Line Trajectory Generation for CNC Machine Tools

Besides the field of robotics research, we can also find approaches for trajectory generation in the field of CNC, that is, machines reading G-code instructions and driving machine tools. G-code is the numerical control programming language. We neither give an overview of CNC systems nor introduce this field here. Therefore, please refer to basic textbooks, for example, the one of Suh, Kang, Shung, and Stroud [253]. Another brief survey of the field of machine tools is given in [133]. In the following, only recent CNC-related works that address the field of on-line trajectory generation are briefly surveyed.

Non-uniform, rational B-splines (NURBS) are commonly used as representation of free-form shapes in the field of CAD/CAM systems (computer aided design/computer aided manufacturing). During the process of machining, these free-form shapes are comparable to arbitrarily specified paths in the field of robotics, and — hence — the basic task of accurate path-following, is very similar. In the field of CNC, the path is always specified, and we only have to work on interpolators. On-line interpolators, which can also be regarded as on-line trajectory generators, are the focus of current research in machine tools. An introduction to NURBS can be found, for example, in the textbook of Piegel [212] or in a short survey by him in [211].

Works on on-line NURBS interpolation [48, 52, 54, 132, 157] usually consider the complete system dynamics for setting up CNC control units, and since the path is completely known in advance, the dynamics can also be calculated beforehand, but this way, unforeseen and unmodeled effects are excluded, and the responsibility of proper functioning is passed to the tracking controllers. In order to support these controllers, it is necessary to adapt the desired trajectory on-line in order to prevent them from bringing actuators to saturation and therewith increasing the tracking error. All mentioned real-time NURBS interpolation techniques are suited for a priori-specified paths represented by NURBS, and hence these works are related to the survey about on-line trajectory generation for improving the path accuracy (p. 19).

Besides these on-line methods, we can also find recent works in the field of CNC machine tools that transfer basic (off-line) methods from the field of robotics. Dong, Ferreira, and Stori [69] present an off-line algorithm to calculate a varying feedrate for CNC machine tools in order to achieve a time-optimal trajectory. The approach from 2007 uses the algorithm of Bobrow/Pfeiffer/Shin [27, 28, 208, 238, 239] as the basis for calculating a feedrate progression for a given path.

2.4.3 Robot Motion Control

Regarding Fig. 2.2 (p. 15), the field of robot motion control is the third field besides path and trajectory planning that relates to this book. Although the latter two subsections focused on robotic manipulators, we address a wide field of robotic systems in general and hence do not focus specifically on any concrete motion control scheme or any concrete kinematic. Only brief overviews of robot motion control in general and hybrid switched-system control are given in the following.

Actuator/Joint Control in Robotics

The overview of low-level robot motion control schemes in the Springer Handbook of Robotics [55] was already mentioned earlier. Additional textbooks describing this subject are by de Wit, Siciliano, and Bastin [278], by Kozłowski

[135], by Sciavicco and Siciliano [229], and by Khalil and Dombre [127]. Many of the described approaches are based on the early works of Whitney [274] and Paul [206]. Another important issue is embedding rigid body dynamics to set up a dynamic model of the robot. The works of Featherstone [81, 82, 83] are to be mentioned as fundamental to this field.

In the following part, we will focus on hybrid switched-system control, which constitutes the next control layer above the actuator controller. For a hybrid switched-system controller, it is essential that the underlying actuator control scheme is stable, that is, if we cannot prove stability for the inner control loops, stability for the outer loops will be even harder to prove.

Hybrid Switched-System Control

Hybrid switched-control systems are systems that comprise a family of continuously working subsystems and a supervisory discrete system that switches between them. Fig. 2.3 shows an abstract and simple scheme of a hybrid switched-system control architecture for a one-DOF system, that is, with one actuator only (e.g., a simple linear positioning unit). This system can be controlled on the basis of different sensor signals (distance, vision, and force/torque), and depending on the current task and/or depending on the current system state and/or depending on time, one of the controllers can be selected. The force/torque, the distance, and the visual servo controllers of Fig. 2.3 are closed-loop controllers; the velocity and the position controller are open-loop controllers (i.e., on-line trajectory generators) as they will be proposed in this work. The difference between these latter two modules is

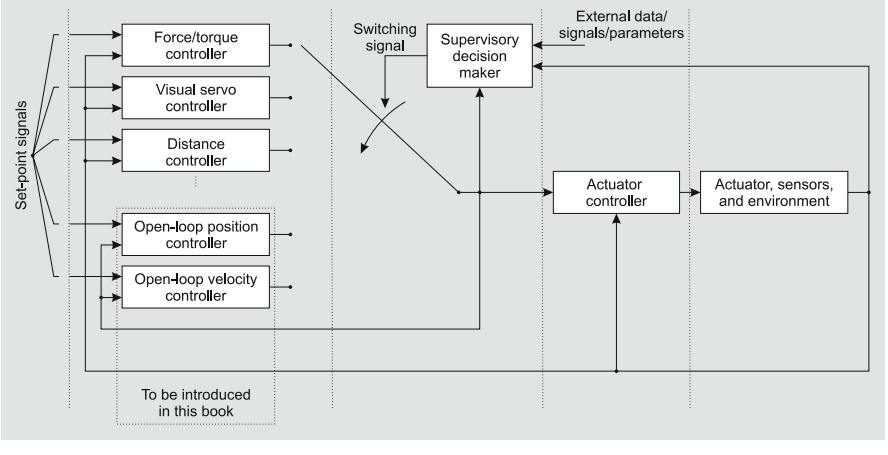

Fig. 2.3 Abstract control scheme of a hybrid switched-system for one single DOF, that is, one single actuator. The dotted vertical lines will be explained in Sec. 2.5 (p. 30).

that the velocity controller does not consider target positions and only leads a single DOF to a certain velocity under given acceleration and/or jerk (additionally also the derivative of the jerk) constraints. Therefore, it is of a much simpler manner than the open-loop position controller. The system depicted in Fig. 2.3 can, of course, be extended to multiple DOFs, as will be shown in Chap. 7 (p. 105).

When we take a multi-DOF robotic manipulation system with many different sensors for granted, and when we consider a system that enables the execution of sensor-guided motion control commands in any DOF, it becomes self-evident that we need to switch discretely between several continuously working (open- and/or closed-loop) controllers at any time. Hence the analysis of hybrid switched-system control is one fundamental part of the work presented here.

Especially the works of Branicky [30, 31] and Liberzon [163, 164] provide elementary concepts to develop and analyze hybrid switched-system control techniques. In particular the stability analysis is of fundamental interest here, because the stability of a switched-system cannot be assured by the stability of each single sub-controller. To prove the stability of hybrid switched-systems can be extremely difficult and many researchers are working on analyzing such stability questions. It may happen that a set of stable subsystems becomes unstable if the switching between them occurs inappropriately [32, 164, 277]. In the field of stability analysis, we can distinguish between techniques for linear [66, 164, 181] and nonlinear [67, 179, 279] switching systems.

The discrete switching system by itself can of course not work without the respective continuously working subsystems. Hence, we also refer to basic works for the two most relevant candidates: force/torque control and visual servo control. Distance control (cf. Fig. 2.3) usually works in one DOF only and is a trivial task.

Force/Torque Control

Robotic applications such as grinding, assembly, or deburring involve an extensive contact between the robot and its environment. In order to establish this contact, force/torque control is applied. In all previously mentioned works in this section, only pose and/or position control was regarded, and this is the first scenario, in which a further sensor is part of the feedback control loop.[1] These sensors can be either wrist-mounted force/torque sensors (e.g., [15, 119]) or joint torque sensors. The latter method was, for example, successfully realized with the DLR light-weight arms [8, 149], which possess joint-integrated torque sensors. Force/torque control, based on sensor signals from wrist-mounted sensors, can be applied to a much wider range of robot systems, since only actuator position feedback — as it is common for

[1] Force control concepts can be distinguished in active and passive methods. Passive ones do not consider a force/torque sensor in the feedback control loop [276] and are excluded here.

industrial robots — is required. The following paragraphs give a brief overview of this field.

Whitney [275] and Mason [176] belong to the pioneers in the field of compliant motion control. Based on their work, numerous approaches have been published in this field. Khatib [129] formulated his key work about the *operational space approach*, and especially the group of De Schutter [226, 227, 228] contributed promising concepts to the community and coined the term *Task Frame Formalism* [40], which enables the development of compliant motion solutions on an abstract programming level.

These works understand force/torque control as one basic element of compliant motion concepts [267]. Three different approaches are known from literature: 1. Impedance control [113], which uses relationships between acting forces/torques and manipulator pose to adjust the mechanical impedance of the end-effector to external forces/torques; 2. Parallel control [53], which enables to control both, force/torque and pose, along the same task space direction. 3. Hybrid force/torque and pose control, which controls force/torque and pose in two orthogonal subspaces [217]. To realize this approach, we have to pay attention to the problem of orthogonality as stated by Duffy [72], who extended the approach such that it is consistent, independent of units, and independent of any origin coordinate system.

General overviews about the field of force/torque control can be found in the Springer Handbook of Robotics [267] and, for example, in the textbooks of Siciliano and Villani [241] or Gorinevsky, Formalsky, and Schneider [102], respectively. The previously mentioned textbooks of de Wit et al. [278], Craig [62], Khalil et al. [128], Kozłowski [135], Sciavicco et al. [229], and Spong et al. [246] also give brief surveys of this field. Another good overview of this field can be found in the work of Vukobratović and Šurdilović [268].

The PhD thesis of Reisinger, who belongs to the same research group as the author does, contains a control framework for the contact transition of force/torque-controlled parallel kinematic machines [220].

Visual Servo Control

If computer vision data is used for robot motion control, we speak about *visual servo control*. Together with force/torque control, powerful robotic systems can be achieved, because — similar to human beings — robots can use two very complementary sensors, one to recognize the global task environment, and one for fine (contact) motions. A camera may be mounted directly on a robot (eye-in-hand) or the camera can be fixed somewhere in the robot's workcell, such that it observes the robot motion from a stationary pose. The field of visual servo control consists of three domains: low-level image processing, computer vision, and control theory. Here, only a brief overview with regard to hybrid switched-system control is given.

Basic overviews were presented by Chaumette and Hutchinson [45, 46, 47], which are regarded as a very good introduction for control engineers, who often are not familiar with the field of computer vision. The origins for most works on visual servo control can be found in the publications of Weiss, Sanderson, and Neuman [273] and of Feddema and Mitchell [84]. Detailed introductions to image processing and computer vision algorithms can be found, for example, in the textbooks of Wahl [271], by Forsyth [93], and of Ma, Soatto, Košecká, and Sastry [171].

Regarding Fig. 2.3, visual servo control is supposed to be applied as one continuously working submodule in a hybrid switched-system. Baeten and De Schutter [16] present a very comprehensive work, in which computer vision and force/torque control become unified in the Task Frame Formalism [40]. Another more recent approach of Gans and Hutchinson [95, 96, 97] suggests two visual servo controllers as submodules in a hybrid switched-system. Assuming an eye-in-hand camera setup, the first control module uses the camera position to calculate an error signal in the feedback loop of the control law and the second submodule uses image features. Stability is proven by means of a state-based switching scheme.

2.4.4 Human-Inspired Motion Analysis

When considering reflexes and human motion patterns as inspiration for the development of robotic systems, as described in the explanation of human neurophysiology (Sec. 1.2, p. 6), we also have to compare both worlds with each other, and we should try to build a bridge between both fields. Flash and Hogan [91] present a mathematical model based on human arm movements. A set of tasks (including compliant motion tasks) was used to observe trajectories taken by a human arm. The observed trajectories, in particular the magnitude of the jerk integrated over the entire movement, were used to rate the performance of an executed trajectory. In a later work of Flash [90], it is analyzed what human arm trajectories would look like if test candidates performed an aimed arm movement to a visually recognized target location, and if this target location suddenly changed. Even if it is not clear how this motor task is performed, it could be stated that after a new visual stimulus appears, an old "plan" is aborted and a new one appears during the movement. A further work of Henis and Flash [110] underlines this book; here time intervals between a visual stimulus and the respective trajectory modification are investigated. Depending on the current trajectory, that is, the current state of motion with regard to the old and the new target, such an interval takes between 10 and 300 milliseconds. Till today, it is not clear how our neurophysiological system exactly works. Human motion patterns are obviously generated on different levels. A monosynaptic reflex, as described in Sec. 1.2 (p. 6), constitutes the lowest level. The motion patterns are furthermore represented in multiple layers in the human motor cortex of

the cerebrum (cf. Fig. 1.6, p. 6), which contains regions that are involved in the planning, control, and execution of human motor skills [232, 263, 272].

We can also find works that consider human-like reflexes in the field of robotics, in particular in the fields of humanoid, biped, or quadruped robots, in which common motion pattern generators are employed to generate set-points for low-level controllers that are responsible for standing upright. After a (sensor) event, this pattern can be adapted by adding a further previously *learned pattern* in order to assure that the system stands upright. Zaier et al. [282, 283] propose such a control scheme for humanoid robots. The learned pattern, which is added to the currently generated one, always fades smoothly in and out, in order to prevent jerky motions. The same idea was applied to prosthetics in [230].

In [106], Haddadin, Albu-Schäffer, De Luca, and Hirzinger present a very impressive work on the detection of unforeseen collisions and respective reaction concepts. Based on previous works of the authors [103, 104, 105, 170], five different collision (= sensor event) reaction strategies are investigated[2]: **1)** The robot shows no reaction at all and continues to follow the reference trajectory; **2)** The robot is stopped as soon as a collision is detected. This is obtained by using the actual joint position, which was measured at the time instant of collision detection, as set-point for the position controller; **3)** Switch from position control to zero-gravity torque control [6, 7], letting the robot behave in a very compliant way; **4)** Switch to torque control with gravity compensation but, in contrast to 3), use joint torque feedback and the signal of the estimated external torque, which is used as a collision signal, to scale down both the motor inertia as well as the link inertia, thus obtaining an even "lighter" robot; **5)** Use the estimated external torque to implement an admittance controller. By defining the desired velocity in the opposite direction of the external torque estimation, the robot "flees" from this disturbance. The strategies 3 – 5 contain switchings from trajectory-following control to sensor-guided robot motion control; this monograph considers the opposite way of switching: from sensor-guided motion back to trajectory-following control (cf. Fig. 1.4, p. 4).

2.4.5 Own Works

This subsection shall briefly classify a selection of previous works of the author with regard to the previously surveyed fields of robotics research. Starting with works on force/torque control for robotic manipulation systems, in particular adaptive control and model-following control schemes [87, 136, 198, 260], especially works on practical implementations of the Task Frame Formalism [137, 138, 139] were done by the author. Another field addresses the fusion of force/torque and acceleration signals in order

[2] The rest of this paragraph is written in accordance with [106].

to separate contact forces/torques and forces/torques caused by inertial effects [143, 144, 145] with the aim to improve force/torque control quality in practical applications [147, 148]. [88] introduces *Manipulation Primitives* as an interface to hybrid switched-systems applied in the field of robot motion control. Besides these objectives, software engineering and real-time aspects commonly play a fundamental role for practical and elegant implementations of the mentioned techniques; over the years, a framework based on a dedicated real-time middleware was conceived and finally presented in [86]. Besides these basic works, experimental achievements in these fields were presented in [140, 141, 142]. The last work to be mentioned here is [146], which is closely related to this book and presents a preliminary stage of on-line trajectory generation in hybrid switched-systems.

2.5 Conclusions and Classification of This Work

After this broad and, in some areas, thorough survey, we are now able to concretize the motivation for this work and to classify it within the robotics research landscape.

The majority of the surveyed concepts for off-line and also on-line motion generation produce a motion along a specified path. But, is this a good approach? — For purely position/pose and/or trajectory-following controlled motions: Sure and without restriction of any kind! But: When we execute sensor-guided motions, for example, by force/torque or by visual servo control, we do not have a predefined path anyway, because the robot motion directly depends on the sensor signal. We have to dismiss the path during sensor-based motion control! As soon as we embed sensor-guided or sensor-guarded motions, there is no predefined path anymore. In particular, we have to say good-bye to trajectory planning and reference trajectories along previously specified paths. There is no path that can be exactly followed, because everything may depend on sensors whose signals cannot be foreseen.

For further advancements in the field of robotics, sensor integration is absolutely indispensable! We can find plenty of approaches in the scientific arena, but when looking at the state of the art in technology (Sec. 2.3, p. 13), almost none of these approaches can be found. The author believes that there is one missing part in the robot control architecture as indicated in Fig. 1.4 (p. 4) that enables switchings from sensor-guided to trajectory-following operations. Furthermore, commercial robot manufacturers have to develop reliable and deterministically working machines. These companies need concepts to react safely to sensor malfunctions. How should a robot behave if a sensor stops delivering a signal during a high-speed sensor-guided motion? If a robot performs a sensor-guided motion in all of its DOFs (e.g., zero-force-control during a teach-in process) how can the controller — except when performing an emergency stop — take over the guidance of the robot if something unforeseen happens?

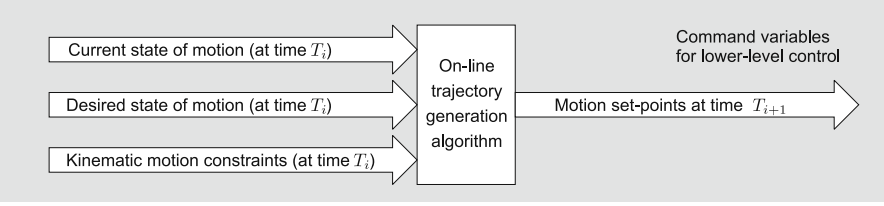

Fig. 2.4 Input and output values of the on-line trajectory generation algorithm.

These are simple questions that are mostly not relevant for research institutions. On the one hand, we (researchers) often complain about inaccessible commercial control units, and that sensor-guided motions are not possible, but on the other hand, we also do not have a general concept that is able to generate set-points for a trajectory from arbitrary initial states of motion to arbitrary target states of motion under consideration of kinematic and dynamic constraints — as would be required for practical realizations.

One of the goals of the author is to contribute to the solutions of these problems. The concept proposed in this monograph does not deliver a complete solution — it is just one further piece of technology to be developed with this work. To place this work in the wide landscape of robotics research, the author suggests considering two different points of view:

Horizontal view
Here, we take a model of multiple horizontal layers for robot motion control into account. As depicted in Fig. 1.5 (p. 5), on-line trajectory generation is considered together with sensor-based motion control as one horizontal layer in an abstract robot control model. This layer constitutes the interface between higher-level (on-line and/or off-line) task and motion planning and lower-level actuator control. This point of view seems to be common in the field of computer science.

Vertical view
Here, we consider vertical layers of a hybrid switched-system control scheme as shown in Fig. 2.3 (p. 24), which shows such a simple scheme and indicates the layered structure via the dotted lines. The on-line trajectory generation algorithm is regarded as one continuously working submodule that is able to take over control from any arbitrary state of motion and to transfer the system to a desired state of motion in the shortest possible time. This figure resembles the view of control engineers.

The aim of this work is to develop an algorithm for time-discrete systems as depicted in Fig. 2.4. This on-line trajectory generation algorithm can also be regarded as feedforward or open-loop controller. One can also consider it as a feedback on the current state. At a time instant T_i the algorithm receives the parameters of the current state of motion, the parameters of the desired state of motion, and the motion constraints, depending on the machine kinematics and dynamics, whereas the dynamics are only represented by constant

kinematic motion constraints in this book (cf. Fig. 2.4). Embedding dynamics while keeping the possibility of instantaneous reactions to unforeseen events alive is considered as a future work, which is discussed in Chap. 9.5 (p. 164). All the algorithm of Fig. 2.4 has to calculate is the respective motion set-points for the time instant T_{i+1}, which is subsequently used as a command variable for lower-level control.