Active Constraints/Virtual Fixtures: A Survey

Stuart A. Bowyer, *Student Member, IEEE*, Brian L. Davies, and Ferdinando Rodriguez y Baena, *Member, IEEE*

***Abstract*—Active constraints, also known as virtual fixtures, are high-level control algorithms which can be used to assist a human in man–machine collaborative manipulation tasks. The active constraint controller monitors the robotic manipulator with respect to the environment and task, and anisotropically regulates the motion to provide assistance. The type of assistance offered by active constraints can vary, but they are typically used to either guide the user along a task-specific pathway or limit the user to within a "safe" region. There are several diverse methods described within the literature for applying active constraints, and these are surveyed within this paper. The active constraint research is described and compared using a simple generalized framework, which consists of three primary processes: 1) constraint definition, 2) constraint evaluation, and 3) constraint enforcement. All relevant research approaches for each of these processes, found using search terms associated to "virtual fixture," "active constraint" and "motion constraint," are presented.**

***Index Terms*—Haptics and haptic interfaces, impedance/admittance control, medical robots and systems, physical human–robot interaction, telerobotics.**

I. Introduction

FULLY autonomous robotic systems are currently used in many varied applications. Modern control algorithms allow these robots to operate in increasingly unstructured environments [1]. They can offer many advantages over humans, such as their resistance to harmful conditions, untiring strength, and superior accuracy and precision.

Nevertheless, there is a limit to the abilities of autonomous robots to perceive and evaluate their environments, making them less suitable than manual systems for some tasks. Highly unstructured environments, where objects within the robot's environment are partially unknown due to sensing or data processing limitations, are a particular challenge for autonomous robots, as any decision-making processes will be based on incomplete or inaccurate information. Humans are particularly adept at dealing with this type of situation.

An additional limitation inherent in current autonomous robots is that, when compared with human operators, they have a limited knowledge base concerning the task that they are performing. A robot may be programmed to perform one set of motions very accurately, but without a comprehensive understanding of the task's purpose and environment, robots are less likely to respond correctly to unusual or unexpected occurrences.

A final important reason for autonomous robots being unsuitable for some applications is that culpability for damage or harm is unclear. In some cases this may not be an issue; however, there often needs to be a distinction regarding who is responsible for the safety of people and objects within a robot's workspace.

By using a robot to intelligently regulate the motion of a human user, rather than replace the human entirely, many of the challenges above can be overcome. "Virtual fixtures," as conceived by Rosenberg, are the first published disclosure of such a control concept, whereby "abstract sensory information [is] overlayed on top of reflected sensory feedback from remote environments" [2], [3]. As the name implies, virtual fixtures are loosely based on mechanical fixtures, which anisotropically limit tool motion, but, as they are enforced in software, they are considerably more flexible with respect to positioning, real-time modification, and feel. Rosenberg's method includes both auditory and haptic virtual fixtures, but the fundamental concept is that by augmenting feedback from the slave to the master in a teleoperation system (see Section II-B), virtual fixtures can reduce mental workload, task time, and errors.

Shortly after Rosenberg's initial publication, in the field of robotic surgery there was little consensus on how best to ensure system safety [4], [5], which had previously been approached using fully autonomous robotic manipulators [6]. It was in this climate of exploring possible requirements for a more widely acceptable and safer form of control that Davies *et al*. devised the parallel concept they termed "active constraints" for robot-assisted knee replacement surgery [7], [8]. Under this concept, surgical tools continue to be manually operated by the surgeon, however, they are attached to a back-drivable robotic manipulator. When the surgeon is cutting a tissue, which requires removal, the robot complies; however, as the surgeon moves toward material, which should not be removed, the stiffness in that direction is increased to prevent overcutting.

Since the conception of this form of control,[1] many research institutions have developed and diversified it greatly. Some of the methods researched are for medical applications, but many are not, and there is currently no definitive concept which unifies the field. Previous survey papers of medical robotics (for example [13]) provide brief discussions of the area; however, they do not approach a complete and structured representation

Manuscript received March 12, 2013; revised July 24, 2013; accepted September 19, 2013. Date of publication October 16, 2013; date of current version February 3, 2014. This paper was recommended for publication by Associate Editor S. Régnier and Editor W. K. Chung upon evaluation of the reviewers' comments. This work was supported by EU Grant FP7-ICT-2009-6-270460.

S. A. Bowyer and F. Rodriguez y Baena are with the Mechatronics in Medicine Laboratory, Department of Mechanical Engineering, Imperial College London, London, SW7 2AZ, U.K. (e-mail: s.bowyer10@imperial.ac.uk; f.rodriguez@imperial.ac.uk).

B. L. Davies is with the Mechatronics in Medicine Laboratory, Department of Mechanical Engineering, Imperial College London, London, SW7 2AZ, U.K., and also with the Department of Advanced Robotics, Istituto Italiano di Tecnologia, Genova, 16163, Italy (e-mail: b.davies@imperial.ac.uk).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TRO.2013.2283410

[1]Independently conceived as "synthetic fixtures" [9], "position assist shared contro" [10], "task oriented virtual tools" [11] and "motion constraints" [12].

of the broad range of research. This survey aims to achieve such a depiction for the research carried out in these areas under the general title of "active constraints." The articles cited represent all currently relevant works, which contain a significant conceptual advancement, found using search terms associated to "virtual fixture," "active constraint" and "motion constraint" in scientific publication databases such as IEEE Xplore, ScienceDirect, CiteSeer, and Google Scholar.

It should be noted that this survey focuses exclusively on publications associated with the terms above. There has been parallel research that utilizes similar concepts, primarily in the field of rehabilitation robotics, which is not covered here. Much of this other work involves active trajectory following [14], which does not conform to the concept of active constraints as portrayed here; however, some of these methods do assist a user in a way similar to that of guidance constraints (Section II-D) [15]. The interested reader is directed toward the extensive survey papers already available for more information on this research [16].

The structure of this survey paper is as follows. In Section II, the primary classifications for active constraint properties are listed. In Section III, a generalized active constraint framework is presented which will be used to frame the discussion of the research within the following three sections. In Section IV, techniques for computationally defining constraint geometries are discussed. Methods for evaluating the relative configuration of robot and constraint geometries are introduced in Section V. In Section VI, approaches to generating robot control commands, which enforce the required constraints, are explained. Section VII discusses some advanced active constraint extensions, which span the methods from the previous sections. A discussion of the survey is given in Section VIII. Finally, conclusions are drawn in Section IX.

II. Definitions of Terms

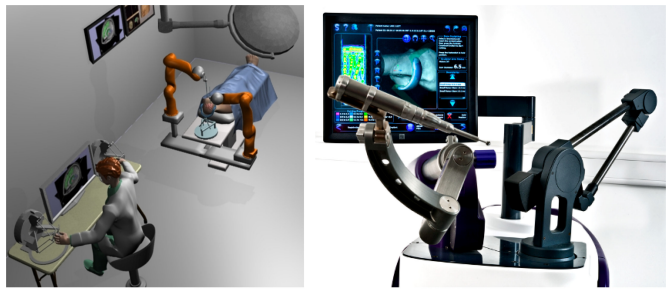
There are several terms that are used to describe active constraint algorithms within the literature. These terms are introduced here so that they can be referred to freely in the following descriptions and explanations.

A. Active Constraints

We formally define "active constraints," and synonymously "virtual fixtures," as collaborative control strategies, which can be used in human manipulation tasks to improve or assist by anisotropically regulating motion. Motion regulation is achieved by attaching tools to a robotic arm, which is primarily controlled by a human user, under either hands-on or teleoperation control. Throughout operation, the robot controller monitors tool motion, and analyzes it with respect to preplanned trajectories and known restricted regions. The active constraint controller then attenuates or nullifies any user command, which will cause the manipulator to digress from a plan, or enter a restricted region.

B. Teleoperated Versus Hands-on Devices

Devices implementing active constraints fall into two categories that depend on where the human user interacts with the system. "Teleoperated" devices are those where the human user is physically removed from the tool-carrying slave manipulator and controls it via a separate "master" device [17] [see Fig. 1(a)]. Telemanipulation offers benefits such as operation in restricted and unsafe environments and motion scaling. However, a key challenge with telemanipulation is that any environmental information that is required by the human user has to be interpreted at the slave device, transmitted to the master, and then presented in some way.



(a) (b)

Fig. 1. Example teleoperated and hands-on robots. (a) Prototypical surgical suite for teleoperation in the FP7 ACTIVE project [20] and (b) the hands-on Sculptor Robotic Guidance Arm. Image courtesy of Stanmore Implants Worldwide Ltd. [21].

"Hands-on" devices are those where the human user interacts directly with the tool-carrying manipulator [18] [see Fig. 1(b)]. As the human user is holding the actual tool, the benefits such as improved access and motion scaling cannot be provided; however, the user directly feels the tool's interaction with the environment. Coupling this direct feedback with the fact that the user is manipulating a "real" tool, means that they will feel well integrated into the procedure, and will likely have a more intuitive ability to perform tasks. For applications where erroneous or uncontrolled tool positions present a significant danger, a hands-on robot is likely to be safer than a teleoperated one because the user can regulate it and the actuation system can be manufactured to be more compliant.

If the actuators of a hands-on device are used to regulate the motion of a tool, then this will constrain both the user and the tool with perfect correspondence between them. However, for a telemanipulation system, the constraint could potentially be applied to the master and/or slave devices, and depending on the teleoperation control architecture, these can have considerably different properties. An analysis of these aspects in teleoperation is given in [19].

C. Impedance Versus Admittance Device Control

Controllers that implement active constraints in master, slave, or hands-on devices generally implement a form of either impedance or admittance control. These are used because they both provide natural frameworks for responding to changes in the device's environment as caused by tool interactions or the human user. As their names suggest, they are both based on the reciprocal mechanical properties of impedance and admittance/compliance. "Impedance" control maps the motion of the device caused by environmental interactions to actuator forces/torques [22], whereas "admittance" control maps

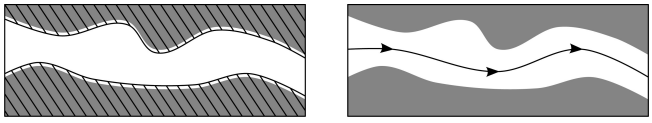


Fig. 2. Example (a) regional and (b) guidance constraints. Obstacles are shown in gray and the constraints in black.

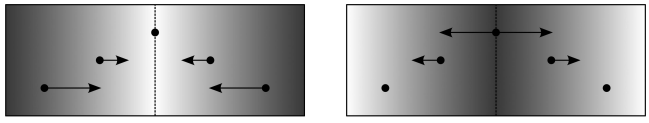


Fig. 3. Example (a) attractive and (b) repulsive constraints. The constraint acts at the dashed line, with permitted regions shown in lighter colors and restricted regions shown in darker colors. Arrows show the direction of "encouraged" motion.

forces/torques being applied to the device by the environment to manipulator motion.

An important distinction between impedance and admittance controlled devices is that those that are impedance controlled must be highly backdriveable so that the environment (or human user) is able to move them freely when there is no controller force; however, admittance controlled devices are highly stiff in response to environmental forces and only move when commanded by the controller.

Although the intended effect of an active constraint controller is the same, an impedance constraint conceptually acts quite differently to an admittance constraint. In an impedance constraint, the controller seeks to apply a force to the user which nullifies, to some degree, the motion which violates a constraint. Conversely, an admittance constraint acts like a filter on the user's intended motion, which only allows the component of motion which does not violate a constraint to be passed to the robot's actuators.

D. Regional Versus Guidance Active Constraints

Most of the active constraints considered within this survey have one of two purposes. "Regional" constraints (commonly called "forbidden region virtual fixtures" [23]) bound a manipulator's tool to certain regions within its task or joint space. That is to say, the constraints prevent the tool from entering certain poses and configurations. Regional active constraints offer benefits such as task simplification [3], [24], reduced risk of tools damaging protected regions/obstacles [25], and avoidance of kinematic singularities [26]. "Guidance" constraints (commonly called "guidance virtual fixtures" [23]) encourage the user to move the tool along a specific pathway or toward a specific target [27], [28]. By their nature, guidance constraints are more intrusive upon the user than regional constraints in normal operation; nevertheless, the ability to constrain positions precisely is invaluable in many applications. Examples of these two constraint types are shown in Fig. 2.

There is an extension to standard guidance constraints, which sometimes appears within the literature, commonly in relation to potential fields. We refer to these as "bound guidance" constraints and they can be thought of as guidance constraints limited to only a specific region of the robot's task space. They are primarily used in assembling curved guidance constraints from multiple linear guidance sections. Here, the limited bounds on each section allow the user to pass from one to another [29].

The distinction between regional and guidance constraints is often vague with respect to their control implementations as, for example, very restrictive regional constraints could be considered guidance constraints [30]. The division is, however, important from an application perspective.

There are certain active constraint implementations where a guidance constraint actively "pushes" the user toward a specific pose as a form of trajectory control [31]–[33]. In general, this type of control is rare within the active constraint literature; however, it is a fundamental feature of the so-called "impedance-based assistance" which is commonly used within rehabilitation robotics. The extensive rehabilitation robotics literature, such as [16], can provide more information if it is required.

E. Attractive Versus Repulsive Active Constraints

The way in which active constraints act is considered in the literature as either "attractive" (encourages motion toward the constraint) or "repulsive" (encourages motion away from the constraint) [34]. The difference between these two types of constraint is best exemplified by considering a unilateral (one-sided), regional active constraint. If an attractive constraint was used, then the user's motion would only be modulated once the tool had already passed into the restricted region, whereas for a repulsive constraint the modulation would come into effect before the tool reached the constraint. As with regional/guidance constraints, the difference between attractive and repulsive constraints is largely superficial rather than inherent in the control. This similarity is seen when considering an attractive constraint that encourages the user toward a linear pathway. This can, in fact, act identically to a large repulsive cylindrical constraint, which has been centered on the pathway. Examples of these two constraint types are shown in Fig. 3.

F. Unilateral Versus Bilateral Active Constraints

Although only relevant to some constraint geometries, the directionality of an active constraint is referred to in the literature as either "unilateral" (acting on only one side) or "bilateral" (acting on both sides) [35]. Conventionally, a regional constraint would be constructed from unilateral surfaces, which prevent tool motion into the restricted space, whereas a guidance constraint would typically be bilateral, always encouraging tool motion toward the guidance path or point. Examples of these two constraint types are shown in Fig. 4.

One important consideration with respect to unilateral constraints is that in some configurations they become discontinuous such that, if they are applied carelessly, they can cause unsmooth, unstable, or erratic robot motion. This problem arises when the constraint enforcement function acts particularly strongly close to the constraint geometry so that, as the

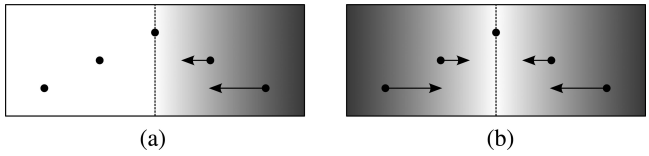


Fig. 4. Example (a) unilateral and (b) bilateral constraints. The constraint acts at the dashed line, with permitted regions shown in lighter colors and restricted regions shown in darker colors. Arrows show the direction of "encouraged" motion.

tool passes from the unconstrained to constrained sides, the constraint algorithm's influence increases abruptly. A common method for overcoming these problems is to introduce a transition region between the two sides, which gradually increases the effect of the constraint as the user approaches the boundary [27], [36], [37]. Temporal functions have also been used for this purpose, where the effect of the constraint is gradually increased over time when the tool is within the constrained region [29], [38].

G. Static Versus Dynamic Active Constraints

A final distinction between active constraint implementations is as to whether they consider changing constraint geometries during operation. Of the literature surveyed, very few papers make a significant consideration of "dynamic active constraints," where the constraint geometry continuously moves as a result of environmental changes [39]–[44]. The overwhelming majority of the literature assumes that the environment is stable enough for the use of static active constraints. It is important to constrain deforming or moving geometries in some applications; however, the sensing and computational demands of dynamic constraints are a significant challenge.

III. Generalized Active Constraint Framework

From the initial concept, active constraints have been developed and modified for use in a wide range of robotic devices. To present the entirety of the field in a systematic way, a generalized active constraint framework is first introduced. This framework is shown in Fig. 5, and is based on three primary processes, which are required in all practical active constraint implementations.

The first stage in applying an active constraint is the input of the geometry, which will constitute the constraint itself. In the generalized active constraint framework, this task is carried out by the "constraint definition" module. The input to this module can be via human user supervision or autonomous sensing and the output will be some computational representation of the geometry.

Before any constraint can be enforced, the relative configuration of the constraint and the robot must first be evaluated so that the anisotropy in the motion regulation can be established. This relative configuration can take a variety of forms; however, it is typically a pair of colliding or closest points, with one on the constraint and one on the robot. This stage of the active constraint computation is carried out by the "constraint evaluation" module.

Once the relative configuration of the constraint geometry and the robot is known, the active constraint can be applied. This is undertaken by the "constraint enforcement" module, which converts the relative robot-constraint configuration into hardware specific commands, which will regulate the motion of the human user.

Methods in the literature for these three stages are described in the following sections. The three modules will be considered separately for clarity; however, there are some methodological interdependences, which will be identified as necessary.

IV. Constraint Definition Methods

The vast majority of the literature assumes that the geometries that are used as active constraints are defined *a priori* by a separate process, which often involves the supervision of a human operator. There are some methods that are described within the literature for autonomously generating constraint geometries based on medical image processing [42], [45]–[48]; however, these are not deemed to be general enough for detailed discussion in this survey.

Three-dimensional (3-D) reconstruction of surfaces is a general method for generating active constraints around physical objects within a robot's workspace. However, with the exception of the B-spline method in [49], the use of un-reconstructed raw data (i.e., point clouds) is currently more widespread within the literature. These point cloud methods are described in Section IV-G, and the reader is directed to the literature [50], [51] for more abstract information on 3-D reconstruction.

Although most of the methods for computing constraint geometries are too specific for inclusion in this survey, the methods for computationally representing them are widely applicable. Additionally, the chosen constraint representation directly affects the geometrical form that constraints can take and the subsequent choice of the constraint evaluation method.

The descriptions of various methods for computationally representing active constraints are presented in this section. Simple methods using points, lines, and planes are first introduced, and then increasingly flexible surface representations are described. Finally, volumetric and explicit representations are explained. Simple examples of each type of constraint are illustrated in Fig. 6.

A. Point Constraints

Although they are simple, points in the robot's task space have been widely used in the literature for the definition of guidance constraints to assist with tool positioning [28], [38], [52]. Proximities to points can be computed using simple closed-form methods, and they can be represented efficiently in memory. Point constraints are not restricted to purely Cartesian space and can be used for rotation and configuration control; however, their potential applications are limited to simple positioning-like tasks.

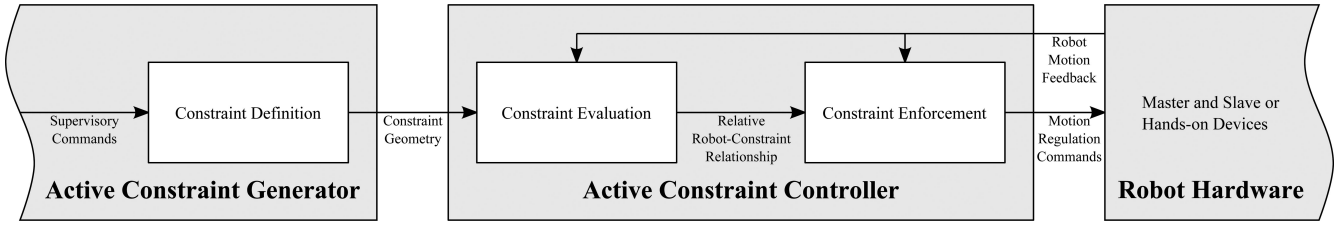


Fig. 5. Generalized active constraint implementation framework.

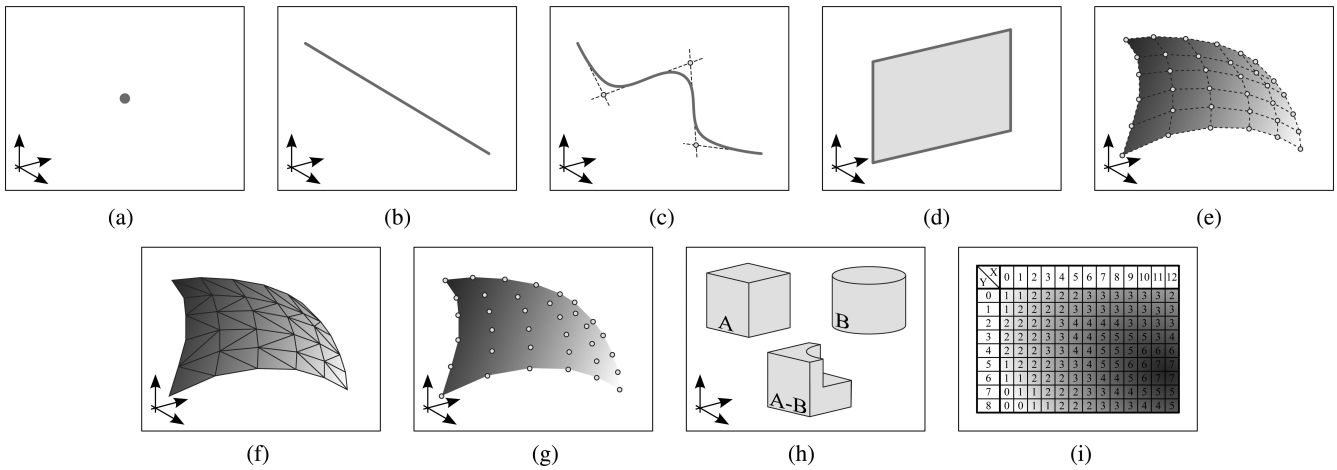


Fig. 6. Example illustrations of the constraint representations described within the literature. (a) Point. (b) Linear. (c) Parametric curve. (d) Planar. (e) Parametric surface. (f) Polygonal mesh. (g) Point cloud. (h) Volumetric primitive. (i) Explicitly described.

B. Linear Constraints

Linear constraints that are formed from vectors within a robot's task space can be used for straightforward guidance constraints [27], [53]. As with point constraints, they offer closed-form constraint evaluation methods and small memory demands, but the number of path following tasks which can be represented as straight lines is limited. It has been shown however, that multiple short, linear, constraints can be combined for more complex tasks [54]. An important consideration with linear (and indeed point and parametric curve) constraints is that close to the line itself, small changes in position lead to large changes in constraint direction. This can be an issue for stability and therefore adding a radius to the line to form a tubular constraint has also been investigated [30].

C. Parametric Curve Constraints

Guidance constraints are commonly represented in the literature as parametric curves ranging in complexity from sinusoidal functions to splines [30], [43], [55], [56]. Parametric curves can be geometrically flexible and are able to describe complex tool paths, but the cost of this is that they do not typically have closed-form constraint evaluation methods. It is shown in [42] how guidance constraints in the form of parametric curves can be generated in real time from cardiac MR images. By "extruding" 2-D curves into a 3-D space (so-called 2.5-D), volumetric constraints can be represented [36]. This is a computationally efficient method for creating 3-D constraints, as any geometry can be projected onto the plane of the 2-D curve for constraint evaluation.

D. Hyperplanar Constraints

Hyperplanes can be used to divide a robot's task space into separate subspaces, and are an effective method for defining both regional and guidance constraints [3], [57]. Hyperplanes are compact to represent in memory, and relative robot-constraint configurations can be evaluated using the closed-form techniques. Hyperplanes are geometrically rigid, making it difficult to apply them to complex environments; however, in [58], it is shown that they can be used to assist in the task of artery dissection. In this study, Park *et al*. used medical images to align a single plane between an artery that requires dissection and the chest wall to which it was attached. By constraining the surgical tool so that it could only move within this plane, the surgeon was assisted in accurate resection. By combining multiple hyperplanes, constraints can be built to represent semi-complex geometries. However, it has been shown that once a certain level of complexity is reached, the efficiency degrades and more complex methods become faster to evaluate [59].

E. Parametric Surface Constraints

Parametric surfaces are the extension of parametric expressions from curves into 3-D surfaces. Little research has been found where parametric surfaces have been used for active constraints [49]; however, nonuniform rational B-splines (NURBS) and other parametric surfaces have been used successfully within the field of haptic rendering [60]. By varying the unilateral/bilateral and attractive/repulsive properties, a parametric

surface can be used to define both regional and guidance constraints.

F. Polygonal Mesh Constraints

Polygonal meshes provide great flexibility for representing constraint geometries, but they are considerably more complex to construct, evaluate, and store than the more simple representations. For some applications, the degree of surface conformity offered by meshes is not worth the increased complexity; however, for applications where the constraint profiles are extracted from "real world" surfaces, they are a useful way to ensure tight regional conformance [39], [61], [62]. Uniquely, in [26], course polygonal meshes were instead used to represent robot-link geometries for checking against workspace obstacle constraints, although analytical primitives could also be used for this purpose. Methods for performing proximity queries with mesh-based geometries are shown in Section V.

G. Point Cloud Constraints

By sampling a set of Cartesian points on the surface of a geometry, a representative point cloud can be constructed. Point clouds are commonly produced from 3-D scanners, range cameras, or fiducial tracking systems, and can be produced and maintained effectively in real time. Within the literature, point clouds have been used for both static [63] and dynamic [41] active constraints.

H. Volumetric Primitive Constraints

Analytical geometric primitives, such as spheres, cylinders, cones, and cuboids (and their higher dimensional equivalents), can be defined using simple mathematical expressions, allowing them to be evaluated in a computationally efficient way. They can be used individually for representing regional constraints, but some researchers have shown that small primitives can be combined such that they form guiding constraint pathways [30]. As combining volumetric primitives is an effective way of representing more complex regional and guidance constraints, several groups have proposed standardized and systematic methods for assembling them [29], [34], [64].

I. Explicitly Described Constraints

Although less common in modern research, some implementations in the literature use explicit descriptions of constraint configurations. These representations can be considered analogous to lookup tables, which describe the constraint's proximity, or effect, at every pose within the robot's workspace. During operation, the robot controller "looks up" the current pose within this table, and retrieves a value which is used to decide if, and how, constraints should be applied. Rather than storing a large lookup table covering the whole workspace, the field is often described using a numerical function, which can be efficiently evaluated during operation [26], [65]. As many constraint methods only depend upon the current pose within the task space, the constraint enforcement can also be precomputed across the workspace and included in the lookup table. This is the case with force fields, where the controller looks up a force vector for the current pose, and applies it without any "in-loop" computation.

It has been shown in the literature that explicitly represented tabular constraints can be tailored for specific applications in order to reduce memory requirements. The regional constraint profiles required for knee surgery were found by Ho *et al*. to have a loosely circular profile, and this allowed them to be represented as a 1-D lookup table of polar coordinates, necessary due to the slow processing speeds of the time [36].

A common limitation to all explicit methods is that, as they are defined before operation, they are generally inflexible with respect to unpredictable variation in their environment. Rigid transformations can be applied to keep explicitly defined constraints in the correct location relative to environmental features, but to date, this has not been demonstrated for complex nonrigid applications.

J. Artificial Neural Network Constraints

Cretu *et al*. researched a method for representing 3-D geometries in a memory efficient manner by training a neural network to output surface proximity values for any supplied Cartesian location [66]. Their neural network is taught the object that it is required to represent using backpropagation with an explicit lookup table of the proximity values in the Cartesian volume. The main advantage of this method over the explicit representation methods is that it is comparatively memory efficient as the neural network is actually equivalent to an implicit surface function. Only the set of interconnection weights must be stored and relatively few calculations are required at runtime.

This technique has been used for the representation of a dynamic active constraint based on the surface of a periodically beating heart [39]. A method for morphing geometries represented in this way was investigated by Cretu *et al*. [66]; however, their method was based on linear interpolation between known start and finish network representations. Neural networks are typically "black boxes" and the effect of changing weighting values is extremely difficult to predict. This means that using their method for nonperiodic deformation, which cannot be predicted and preprocessed, is likely to be challenging.

V. Constraint Evaluation Methods

To decide on the necessity, and direction of anisotropy, of an active constraint, the relative configuration of the robot and the constraint must be evaluated. Conventionally, the geometric relationship of interest is the proximity between the robot's tool tip and the active constraint geometry, or simply whether the two have collided. There are also some implementations in the literature where proximities to constraint features such as apexes and centeres are required [29], [34]. In addition to the tool tip, there are a small number of methods in the literature where tool bodies or robot links [26], [43], [62] are constrained and therefore the relative configuration of the constraint and these parts is also required. Within this survey the term constrained tool geometry (CTG) will be used to abstractly refer to the tool geometry that is to be constrained, whether this is tool tip, tool body, or robot.

The majority of constraint evaluation methods within the literature are based on simple constraint representations and, subsequently, closed-form expressions from linear algebra are sufficient. In most of the remaining literature, either explicit constraint representations are used, making constraint evaluation trivial, or the method used is not discussed. In the few other cases, the two key methods used, namely, spatial partitioning/bounding volume hierarchies and feature tracking, are described next. This is in no way a complete review of proximity querying, nearest neighbor searches, or collision detection; only methods currently used within the active constraint literature are discussed. These areas are well covered within fields such as robot motion planning and computer graphics, and therefore the reader is guided to [67]–[70] if more detail is required.

A. Spatial Partitioning and Bounding Volume Hierarchies

Space partitioning trees and bounding volume hierarchies are widely used to allow efficient searching of geometric objects for intersecting and closest points. These structures are constructed so that during traversal, large sections of the objects can quickly be eliminated without needing to compare every low-level surface primitive [67].

Covariance trees are a variation on spatial partitioning k-d trees [71], where the splitting planes are oriented based on an object's covariance so that the splitting of the surface is more efficient, minimizing recursions during the search [72]. These covariance trees were used in [62] for potential collision detection in regional constraint-assisted endoscopy. In this implementation, potential collisions between line segments defining the tools and a mesh representing the patient's anatomy are checked using a covariance tree. To predict, rather than just identify collisions, each node's bounding volume was inflated by a proximity threshold. This expansion means that, when tree collisions are detected, the tool is actually only close to the constrained tissue, and the constraint can be used to prevent it from moving any closer. Bounding volume hierarchies are efficient to search, but their initial generation can be time consuming and therefore their application to dynamic constraints would require the investigation of real-time modification such as discussed in [73]–[75].

B. Feature-Tracking Algorithms

Sequential constraint evaluations within an active constraint controller will tend to be reasonably similar if the control frequency is high with respect to the speed of robot movement, as is generally the case. The temporal coherence between evaluations means that incremental feature-based algorithms are well suited. Methods such as Lin–Canny use the closest points from the preceding computation step to identify a small neighborhood, where the closest points at the current step should be found [76]. This small neighborhood means that searches at each stage are considerably reduced from searching the complete objects. Voronoi-clip (V-Clip) [77], a modified version of Lin–Canny able to process concave objects, was used in [61] to evaluate the proximity between a user's hand and a cylindrical guidance constraint for peg placement tasks.

VI. Constraint Enforcement Methods

There is a wide variety of methods that are described within the literature for enforcing active constraints. In general, the input to these methods is the current relative configuration of the CTG and the constraint geometry. Based on this relative configuration, the active constraint enforcement module decides if, and how, anisotropic motion regulation should be applied.

Seven primary approaches for enforcing active constraints were found within the literature and each of these is presented in this section. The constraint enforcement methods have significant differences between them and each have attributes which lend themselves to certain applications over others. To introduce the constraint enforcement methods, and aid in the selection of a method for a specific application, a decision tree, the layout of which was optimized for improved discriminatory power [78] is presented in Fig. 7. This decision tree is based upon several key criteria from the target application and can be used to recommend a method from the following sections for further investigation.

A. Simple Functions of Constraint Proximity

The most common constraint enforcement methods described in the literature are those where the effect of a constraint on a device is computed using a simple function of the proximity between the CTG and the constraint.

Linear functions, generally based on minimum constraint proximity, have been widely used to enforce both regional [40], [57] and guidance [29], [38] constraints. The mechanical analogy of an elastic spring between the CTG and constraint (with stiffness k_p) is often used when describing linear functions, as is the classic control concept of a proportional position controller. A linear function for a Cartesian pure spring impedance controller takes the form

$$\mathbf{f}_p = k_p(\mathbf{p}_{\mathrm{const}} - \mathbf{p}_{\mathrm{ctg}}) \tag{1}$$

where $\mathbf{f}_p$ is the constraint force vector, k_p is a linear gain, $\mathbf{p}_{\mathrm{const}}$ is the closest point on the constraint geometry to the CTG, and $\mathbf{p}_{\mathrm{ctg}}$ is the closest point on the CTG to the constraint. Alternating the value of k_p between positive and negative changes the constraint between attractive and repulsive, respectively, and changing its magnitude affects how strongly the constraint acts upon the robot and a piecewise function can be incorporated to implement unilateral constraints. Using the Jacobian and the principle of virtual work [79], the constraining force $\mathbf{f}_c$ can be applied to a device via its joint actuators.

The stability of linear constraints for teleoperation was investigated at static-force equilibrium by Abbott and Okamura [80]. They initially found that using Hannaford's two-port model with Llewelyn's criteria for unconditional stability [81] was unsuitable for predicting constraint stability, and it led to unstable systems being misclassified. Using the eigenvalues of a system in its discrete state-space form was more successful, giving a "good rule-of-thumb" for predicting stability. In [82], they developed their discrete formalization and showed that analysis of the system's eigenvalues was sufficient for stability investigations, even with a unilateral (i.e., discontinuous) constraint.

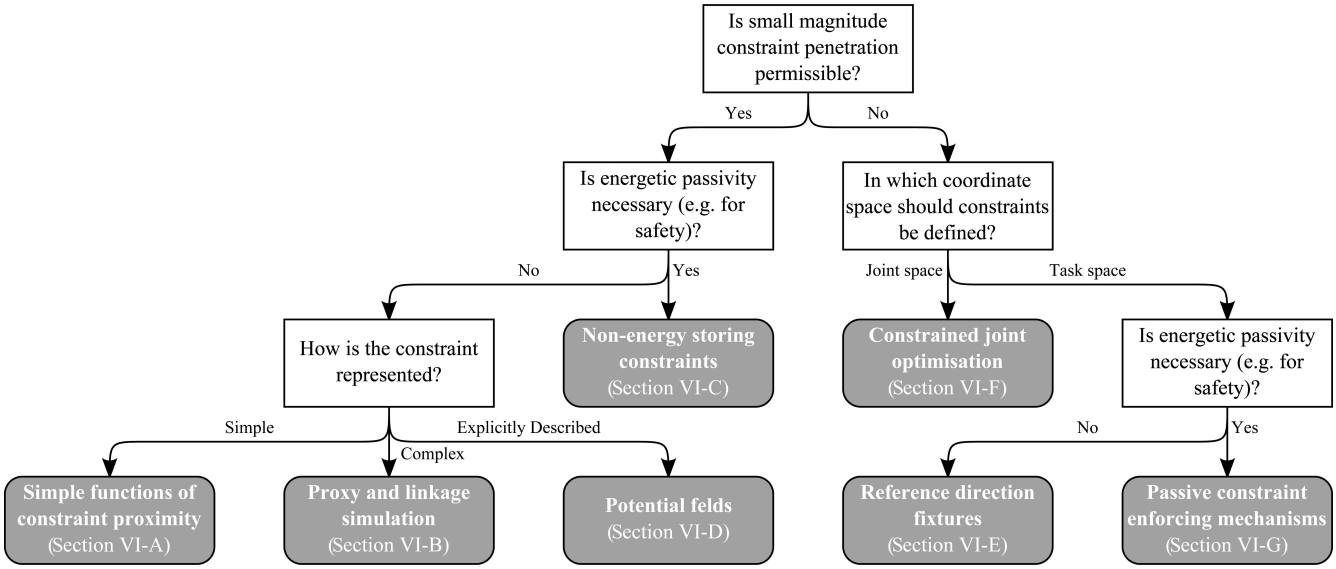


Fig. 7. Simplified decision tree for recommending a constraint enforcement method based upon the key decision criteria. Shown in gray are the seven constraint enforcement methods with reference to the sections in which they are described.

Based on their discrete state space representation of this system, and a linear time invariant model of the user, they found that they were able to predict experimental stability bounds for k_p with a reasonable consistency. The numerical stability of linear constraints is also discussed in [83]. Here, Joli *et al*. define an implicit constraint formulation based on the augmented Lagrangian method which, it is claimed, avoids numerical instabilities.

Derivative terms have been used to extend linear enforcement functions and improve their properties for various applications. If a user moves against a constraint at a higher velocity differential, then it will take a larger force from the controller in order to slow them down and minimize any nonconformance. By introducing a derivative (or viscous) term to the enforcement function, the controller is able to respond to this situation, for example

$$\mathbf{f}_{pd} = k_p(\mathbf{p}_{\text{const}} - \mathbf{p}_{\text{ctg}}) + k_d(\dot{\mathbf{p}}_{\text{const}} - \dot{\mathbf{p}}_{\text{ctg}}) \qquad (2)$$

where $\mathbf{f}_{pd}$ is the constraint force vector, and k_d is a derivative gain. Clearly, if the constraint geometry in this formulation is static (i.e., $\dot{\mathbf{p}}_{\text{const}} = 0$), then the derivative term will be isotropic so that it opposes motion conforming to the constraint as much as it opposes motion that does not. This may be suitable for some applications [84]–[86]; however, if $\mathbf{p}_{\text{const}}$ is set to move, along a pathway for example, then the robot will be constrained to follow its motion. By considering the rotational penetration of a constraint using "eigenscrews," viscoelastic constraints such as these have been extended into six-degrees of freedom (DOF) control [87].

B. Proxy and Linkage Simulation

Proxy techniques were originally conceived for haptic rendering of surfaces in virtual environments on impedance controlled devices [88], [89]. At the time, most techniques for surface rendering were proximity based and therefore disregarded the path taken by a CTG. Often, the path is irrelevant, but when a regional constraint is thin, for example, and the user forces the CTG to reach the region's midpoint, the closest edge will switch to the opposite side and the constraint will act in the opposite direction. There are two potentially detrimental effects arising from this behavior: it can cause harmful control discontinuities and it will encourage the user to pass completely through a constrained region, rather than restricting them.

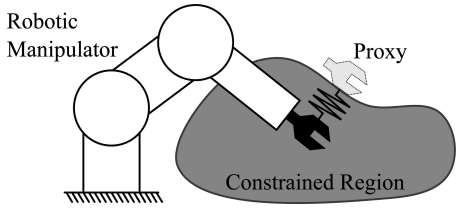


Fig. 8. Illustration of a proxy-based constraint enforcement, where an elastic linkage between a proxy and a robot's end effector is simulated.

In its simplest form, a proxy is an entirely simulated virtual object, which can have any dynamic properties required by the controller and constraints. In the simulation, the proxy is virtually attached to the CTG of a force reflecting device by some form of virtual linkage (as shown in Fig. 8). The linkage model is typically elastic or viscoelastic, although it can take any form to give the constraint properties desired. If the CTG violates a constraint, then the proxy will remain on the surface causing them to separate, and the virtual linkage to generate a force. Although proxies are often mentioned within the literature, there have been a few significant attempts at utilizing them for active constraints in this standard form. Proxies are, however, the foundation of the more complex methods described within this section.

Implicit-force and modified-damping control are two proxy-based algorithms that were formulated by Ho *et al*. for the application of regional constraints to hands-on knee replacement surgery [36]. In these approaches, a proportional-derivative position controller (i.e., viscoelastic linkage) is used to drive the CTG toward a desired position (i.e., a proxy), which is controlled so as to avoid constraint penetration. Two parallel controllers are used to implement this architecture: one defines the motion of the proxy and the control parameters, and the other computes and applies constraint forces based on the viscoelastic linkage.

The proxy motion controller in this constraint formulation generally acts in the conventional way, whereby the proxy is free to follow the CTG in the unconstrained region, but is prevented from following it into the constrained region. Uniquely, this method also considers an additional configuration close to the constraint boundary. In this configuration, the proxy is moved up onto the constraint boundary ensuring that smooth surfaces are produced during contouring cuts. The difference between the "implicit-force" and "modified-damping" variants of this method lies in how the velocity of the proxy is computed. In the "implicit-force" method, the proxy's motion is quasi-static, such that the derivative term within the control law always opposes CTG motion away from the proxy, irrespective of whether the CTG is violating a constraint. In the "modified-damping" method, the velocity for the proxy is computed from the force applied by the user to the end effector, in a way similar to that in reference direction fixtures (see Section VI-E). The velocity consideration in this method means that the derivative term is anisotropic and treats motion which either conforms to or violates a constraint differently.

Virtual mechanisms were conceived by Joly and Andriot as a novel method for constraining the motion of a proxy used in teleoperation [12]. Their control algorithm operates by simulating the kinematics of an ideal (i.e., inertia free) mechanism, which is attached, via simulated linkages, to both the master and slave device. By selecting an appropriate kinematic structure for the virtual mechanism, the motion of its end effector is restricted to a limited task space, thus constraining the master and slave's motions via the linkages. Joly and Andriot showed how the kinematics for various functional virtual mechanisms can be constructed; however, the requirement for these kinematic descriptions of constraints can be complex or restrictive. Compound versions of virtual mechanisms were later used to decouple constrained and unconstrained DOFs with greater control [90]. In this method, secondary virtual mechanisms are used to remove the isotropy of the conventional spring-damper linkages in Joly and Andriot's method. These secondary mechanisms mean that the dynamics and behavior of master and slave motion's in both constrained and unconstrained directions are fully controllable.

Pseudoadmittance control is a method for applying constraints developed for admittance controlled robots (specifically reference direction fixtures, as in Section VI-E) on impedance controlled teleoperation systems, thus giving quasi-static transparency and hand tremor attenuation [91]. To achieve this, Abbott and Okamura conceived a proxy-based method, which uses three control laws to define the forces, which should be applied to master and slave devices, and the simulated proxy. The master forces are based on its separation from the slave device, the slave forces are based on its separation from the proxy, and the proxy forces are based on the error between the master and slave. The proxy will respect any admittance constraints applied and as the slave follows the proxy, it will also be constrained. By controlling the velocity of the proxy, Pezzementi *et al*. [32] were able to use this method for applying active constraints to a robot's velocity rather than just position.

C. Nonenergy Storing Constraints

A challenge with many enforcement methods is that the constraints have the effect of storing potential energy when the CTG is forced to move against them. Once the energy has been stored, releasing the device sharply can cause unwanted and potentially dangerous motion if not controlled correctly. Another limitation in common proximity-based approaches is that they require at least a small amount of constraint violation before any motion regulation is applied.

Simulated plasticity was proposed by Kikuuwe *et al*. to overcome both of these difficulties with constraints [92]. The property of plasticity is advantageous for active constraints because it involves a stiff initial collision and then, if penetration does occur, the constraint only dissipates, and never stores, energy. A challenge with implementing active constraints, which exhibit this behavior, is that it is highly discontinuous when the tool crosses the constraint boundary and this can cause instabilities or tactile inconsistencies in discrete time implementations.

The solution Kikuuwe *et al*. offered to the discontinuity problem was to apply the plasticity (modeled as coulomb friction) to a proxy, as discussed in their previous work [35]. They claim that their discretized proxy model can apply nonenergy storing static boundary constraints that allow the user to sense boundaries without crossing them. The aim of their research was to show that simulated plasticity constraints could be implemented on both impedance and admittance devices. This means that they did not compare their results with other conventional techniques and quantification of the improvement from plasticity was not given.

Continuous impulsive forces represent a similar approach to the implementation of constraints, which imitate hard inelastic collisions in impedance devices [93]. The continuous impulsive forces technique is based on the previous use of impulse forces in haptic rendering [94]. Impulse forces are applied for a single controller time step with the aim of eliminating the constraint-violating component of a CTG's motion. The continuous nature of this method comes from the fact that, for each time-step in which the tool is violating a constraint, the controller applies a new impulse force $\mathbf{f}_{\mathrm{CIF}}$ to prevent any further incursion. The impulse force is calculated from the equivalent CTG mass violating the constraint M_{vc}, the CTG velocity component violating the constraint $\dot{\mathbf{p}}_{\mathrm{vc}}$, and time step Δt as follows:

$$\mathbf{f}_{\mathrm{CIF}} = \frac{-M_{\mathrm{vc}}\dot{\mathbf{p}}_{\mathrm{vc}}}{\Delta t}. \tag{3}$$

Although it has the effect of storing energy, the implementation in [93] combines the impulse force with the output from a PD

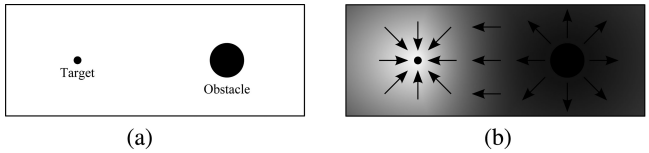


Fig. 9. (a) A task space with a target position and obstacle and (b) a potential field which could be constructed to constrain it. In the field, dark colors represent high potential values and light colors represent low potential values. The arrows show the direction of "encouraged" motion.

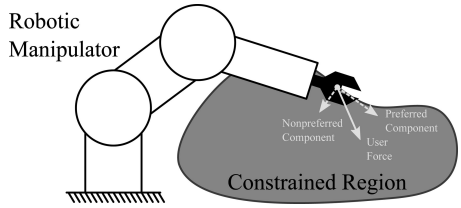


Fig. 10. Reference direction fixture where a user's force is decomposed into preferred (permitted by the constraint) and nonpreferred (resisted by the constraint) components.

controller to prevent slow "drift" to which impulse forces will not respond. The authors state that by using this combination of impulse force and PD control, they provide high initial stiffness, reduce incursion magnitude, and prevent PD bouncing.

D. Potential Fields

Potential fields were devised as a method for real-time collision avoidance in robotic systems [95]. In the initial formulation, each point within a robot's discretized Cartesian workspace is allocated a potential value based on the distribution of obstacles and targets that surround it (as illustrated in Fig. 9). Areas where the robot is actively encouraged to be (such as targets) are allocated low potential values, and areas from which the robot is restricted (such as obstacles) are allocated high potential values. By calculating the negative gradient of the field, a force vector can be extracted for any robot position, which will point away from obstacles and toward targets. By applying this force at the robot's end effector, this force will actively move the robot as required by the obstacles and targets. It is not only workspace obstacles, which can be considered by this method, it is also able to incorporate joint limits if a potential field is implemented in joint rather than task space.

It should be noted that the terms "force fields" and "repulsion fields" are sometimes used in relation to potential fields. These can be considered negative gradient representations of potential fields and can be computed directly using the Cartesian partial derivatives. In, for example, [34] and [84], the terms force and repulsion fields are also used to refer to any impedance constraint method, where the output is a force vector, irrespective of how the force is computed. To ensure clarity, this convention will be avoided in this survey.

Using potential fields for obstacle avoidance in autonomous robots is a well-researched area. Aigner and McCarragher extended autonomous potential field controllers so that a human user was able to guide the manipulator's motion between obstacles using a non-force-reflecting master device [65]. They proposed that a standard task space potential field should be built for the robot, with attractive "wells" and repulsive "peaks," as necessary for the robot's environment and task. During operation, the gradient of the potential field was used to compute a robot velocity command which was sent to the slave device. Two methods for combining the user commanded motion with the potential field were proposed. In one approach, the velocity of the master device is added to the field velocity before it is sent to the slave robot. In the other, the motion of the master device is used to translate and scale obstacles within the potential field.

Later research by Turro *et al.* showed how potential field active constraints could be used in teleoperated systems with force-reflecting master devices [26]. Their method is conceptually similar to the conventional application of potential fields to autonomous robots with the difference being that forces from the potential field are applied on the master device, to regulate the motion of the human user. A position controller on the slave follows the position of the master, and therefore the effect of the potential field is on both the master and slave. An important development in [26] is that each link within the robot is considered in the potential field and the obstacle avoidance forces for each of them are transformed and combined with the force at the end effector before application to the master device. This method ensures that, while the master respects guidance constraints, it does not cause the body of the slave robot to collide with obstacles within its environment. It is this constraint formulation, which has subsequently been used in practical constraint applications such as microscale telemanipulation [86] and intracellular injection [96].

E. Reference Direction Fixtures

Active constraints on admittance controlled robots require that the robot is more compliant when moved in a direction that respects a constraint and less compliant when not. Burghart *et al.* proposed an initial approach to generating this compliance anisotropy by geometrically scaling the velocity vectors produced from a linear admittance function [27]. However, the more recent "reference direction fixtures" described by Hager *et al.* [28], [30], [97], [98] offer a more systematic formalization and are widely used within the literature. It is this approach that will be focused upon in this section.

In reference direction fixtures, the scalar admittance gain of Burghart *et al.* is replaced with a compliance matrix, which accedes to the user applied force in an anisotropic way. The key component of this approach is the method employed for specifying the compliance matrix which is based on the concept of "reference directions." For a given time and position, the entire space of possible directions of motion is divided into two complementary subspaces: "preferred" (i.e., those which correspond to a direction permitted by the active constraints) and "nonpreferred" (i.e., those which correspond to a direction restricted by the active constraints). An example of the subspace decomposition used in reference direction fixtures is shown in Fig. 10.

The preferred directions can be computed in a variety of different ways, allowing different constraints to be represented. In their general form, they are represented as a $6 \times n$ matrix $\mathbf{D}$, defining n linearly independent directions in which motion is preferred. Taking the span $[\mathbf{D}]$ and the kernel $\langle \mathbf{D} \rangle$ projections of $\mathbf{D}$, any user applied force vector $\mathbf{f}$ can be decomposed into independent components in preferred and nonpreferred directions. The span and kernel can be weighted and combined to construct an anisotropic compliance matrix that can convert user forces $\mathbf{f}$ into constrained velocities $\dot{\mathbf{p}}_{rdf}$

$$\dot{\mathbf{p}}_{rdf} = c([\mathbf{D}] + \mathbf{c}_\tau \langle \mathbf{D} \rangle)\mathbf{f} \tag{4}$$

where c is the overall system compliance and c_τ is the compliance in nonpreferred directions.

By varying c_τ between 0 and 1, the compliance matrix can be tailored so that nonpreferred directions are noncompliant ($c_\tau = 0$), fully compliant ($c_\tau = 1$), or partially compliant to any degree in-between. The partial compliance for nonpreferred motion means that admittance active constraints can be constructed which can be overpowered if the user requires.

Guidance constraints are implemented on reference direction fixtures by functionally generating the preferred direction matrix $\mathbf{D}$ based on the relative constraint-robot configuration. For positioning constraints, a single-task space vector can be used, which points from the current to the desired positions. This method has been used for both 3-DOF [28] and 6-DOF [98] manipulators; however, some research has shown that when the robot is close to a target, it can become erratic or unstable due to discretization and noise that causes large discontinuities in $\mathbf{D}$. Several solutions were proposed in [30], but autonomous fine positioning and fixing the constraint direction were found to be the most successful. In [29], this problem was addressed for impedance devices, where it was proposed that a damping factor could be introduced when the tool becomes close to the target. This could equally be extended to admittance controllers by scaling the compliance matrix as a function of the CTG velocity.

Path following constraints can be implemented in this framework by setting $\mathbf{D}$ as a single vector, which is tangential to the closest point on the path. However, if the CTG does move away from the path, then the controller will make no distinction and the motion will be constrained to a parallel pathway [28]. To overcome this, positional feedback can be incorporated into $\mathbf{D}$ so that rather than only directing tangentially to the path, it includes the vector between the CTG and the closest point on the pathway $\mathbf{u}$ as a correction component [98]

$$\mathbf{D}_c = (1 - k_d)[\mathbf{D}]\mathbf{f} + k_d \|\mathbf{f}\| \langle \mathbf{D} \rangle \mathbf{u} \tag{5}$$

where $\mathbf{D}_c$ is the modified preferred direction matrix, and k_d is a constant which controls the degree of offset compensation.

For applications where the path does not need to be followed precisely, the constant motion regulation can be intrusive. Bettini *et al*. created a method whereby a 3-D "tube" is created around the path, inside which the end effector's motion is unregulated [30]. Here, conical constraints are also proposed for relaxing the approach on point positioning constraints.

Regional constraints based on reference direction fixtures have not been explicitly discussed within the literature, however computing $\mathbf{D}$ based on surface collisions is not believed to be overly complex. The initial paper by Berghart *et al*. [27] and subsequent papers such as [37] and [99] all enforce regional admittance constraints by selectively scaling the robot velocity based on its direction and proximity with respect to regional boundaries. The effect of these methods is the same, but their formulation is less general.

Autonomous error compensation was used by Castillo–Cruces and Wahrburg to overcome human difficulties in simultaneously controlling the position and orientation of a 6-DOF robot under reference direction fixtures [33]. They found that, in a system with both position and orientation reference direction fixtures, only position or orientation would be effectively constrained at any one time. Castillo–Cruces and Wahrburg stated that this is due to the translational and rotational components of motion being decoupled from each other in such a way that the user, focusing on moving one, will not notice an error in the other.

To ensure that errors are always corrected, they modified (4) to include a proportional controller with gain k_v, which autonomously compensates for error $\mathbf{e}$

$$\dot{\mathbf{p}}_{\text{aec}} = c([\mathbf{D}] + c_\tau \langle \mathbf{D} \rangle)\mathbf{f} + k_v \mathbf{e} \tag{6}$$

where $\dot{\mathbf{p}}_{\text{aec}}$ is the constrained and autonomous error compensated velocity. This formulation was found to reduce off motion-axis error, although the cost of this increased accuracy is that a component of the robot's motion is no longer fully controlled by the human user, which is somewhat inconsistent with the conventional definition of active constraints.

F. Constrained Joint Optimization

Constrained quadratic optimization of robot's joint velocities was one of the earliest active constraint enforcement strategies proposed in the literature. Unlike the majority, this method considers the robot in joint space and can therefore efficiently constrain the motion of individual links as well as the end effector. Funda *et al*. [100] demonstrated the application of this method for the control of a teleoperated robotic endoscope, where 1) the end effector position was constrained to prevent tissue damage; 2) the endoscope focal position was constrained to a location controlled by the user; and 3) the robot joints were constrained to be within their mechanical limits. Expressing these three constraints as algebraic inequalities and combining them into a single-matrix expression allows them to be solved numerically for joint displacements.

There can potentially be many joint displacements, which are considered acceptable by the set of constraints, some of which may be "more suitable" than others. Funda *et al*. establish the best solution by minimizing the rotational error of the endoscope focal position, the end effector displacement, and the

joint displacements. This is equivalent to

$$\Delta \overset{*}{\mathbf{q}} = \underset{\Delta \mathbf{q} \in Q(\mathbf{q})}{\arg\min} \sum_{i=1}^{N} C_i(\mathbf{q}, \Delta \mathbf{q}) \tag{7}$$

where $\Delta \overset{*}{\mathbf{q}}$ is a vector containing the optimized displacement at each joint; N is the number of controllable joint frames; $\mathbf{q}$ is a vector containing the current joint positions; $\Delta \mathbf{q}$ is a vector containing joint displacements; $C_i(.)$ is a weighted objective cost function for joint i based on the three criteria above, and $Q(\mathbf{q})$ is the set of constraint conforming joint displacements in the current position. By representing (7) as a linear least-squares problem, the value of $\mathbf{q}$ is iteratively optimized using Lawson and Hanson's method [101].

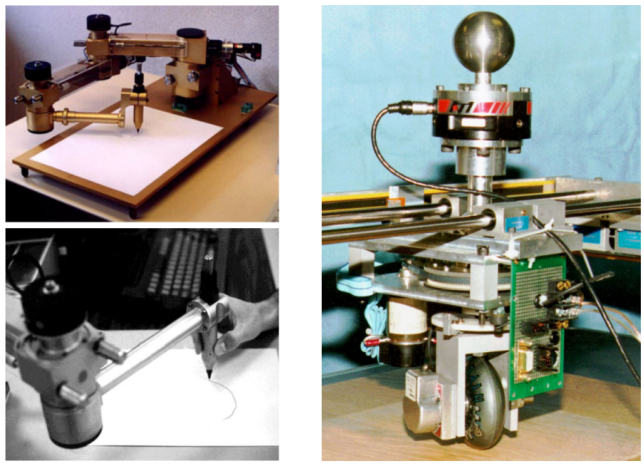
Enforcing constraints with this technique ensures that commands are never sent to the robot which will cause it to move into a constrained configuration. This is a problem which will reduce with developments in computing power; however, the limitation of this method is that the iterative optimization algorithm is computationally demanding and inconsistent for some constraint and cost functions. In addition, the master device, which commands the focal position, is non-force-reflecting, and therefore only visual feedback can be relied upon to inform the user about the constraint interactions.

Haptic constraint feedback can be incorporated into a constrained joint optimization controller by implementing it on a hands-on, as opposed to teleoperated, robot. Under hands-on control, when the constrained joint optimization prevents the manipulator from moving in a certain direction, the user holding the robot will directly feel this. As the robot under constrained joint optimization must be admittance controlled, a force sensor is required on a handle at the end effector to interpret the user's desired motion. Li *et al*. used such haptic constraints with a 7 DOF admittance controlled robot for simulated endoscopic sinus surgery [45], [62].

The sinus geometry considered by Li *et al*. is complex and therefore a covariance tree (as described in Section V-A) was preoperatively constructed from it, to allow efficient identification of when tool–tissue collisions were impending, for the application of constraints. Intraoperatively, a linear admittance control law was used to compute the required CTG velocity and the covariance tree was used to identify when and where the CTG is closer than permitted to the tissue. Based on these pieces of information, the constrained optimization controller utilized terms to prevent tissue penetration and minimize the discrepancy between the user desired and the actual CTG velocity. By using their iTaSC constrained optimization controller, Borghesan *et al*. applied similar optimization-based active constraints to a PR2 robot [102].

In [56], a comparison was made between the applications of constrained joint optimization using teleoperation (without haptic feedback) and hands-on (with haptic feedback) control. It was found that with haptic feedback the tasks were faster and more intuitive than without.

Guidance constraints can be incorporated into the constrained joint optimization by including additional constraint inequalities [25], [52], or by modifying the admittance control law to incorporate reference direction fixtures, as described previously [62].



(a) (b)

Fig. 11. Two passive constraint enforcing mechanisms. (a) PADyC [110]. (b) Cobot [110].

G. Passive Constraint Enforcing Mechanisms

Constraints on active robots have been the main focus of this survey; however, several groups have researched passive devices, which use physical mechanisms to constrain the motion of an end effector. The core concept in these devices is that they are hands-on and their primary motive force comes directly from a human user. In each case, the control architecture is only able to limit or redirect the motion. The main advantage cited for passive mechanisms is that they are safer because errors or malfunctions cannot cause harmful movements.

Mechanical brakes were used by Taylor *et al*. to apply "passive manipulation aids" as a method for improving the placement accuracy of bone segments during craniofacial osteotomies [103], [104]. Passive manipulation aids were implemented on a mechanical device consisting of six orthogonally decoupled passive axes with a brake attached to each. By actuating each brake when the associated manipulator axis was correctly aligned with the target, the task of manual placement could be considered one axis at a time by the surgeon. Sub-millimetre/degree accuracy was achieved in this way.

Mechanical clutches have also been used to passively enforce active constraints. "Passive arm with dynamic constraints" (PADyC), as shown in Fig. 11(a), is a "mechanical guiding system," which can enforce both regional and guidance constraints [105]. It consists of modular mechanisms constructed from clutched, contra-rotating freewheels, which allow the controller to place bounds on the angular velocity of each joint. Based on the relative configuration of the end effector position and the constraint geometry in the joint space, PADyC's controller identifies what angle each joint can move through before it will violate a constraint or pass a target point. Using these joint displacement limits, the angular velocity bounds can be set for each joint so that during the following time step the end effector will respect all constraints. Rendering constraint

profiles, which are not parallel to the axes within PADyC's joint space, is a significant challenge, as it is with most brake actuated mechanisms. When moving along constraints that are not parallel to joint space axes it is not the absolute change in each joint that must be limited, rather it is the ratio between joint velocities which must be controlled. As a result, surfaces can be rough, frictional or possibly even unenforceable by PADyC.

Both brakes and clutches are combined to apply guidance constraints in the "passive trajectory enhancing robot" (P-TER) [106]. The P-TER was designed with brakes on each joint within the mechanism and clutches between the joints. The intention was that clutches can enforce relative velocity constraints between joints so that smooth guidance pathways could be enforced.

Continuously variable transmissions (CVT) were used within "Cobots," as shown in Fig. 11(b), to constrain the relative joint velocities in a passive mechanism [107], [108]. Mechanisms based on brakes and clutches all dissipate the energy that users introduce to the system and therefore constraints cannot be applied which are both smooth and stiff. CVTs are used within a Cobot to link the angular velocity of all joints to the angular velocity of all others such that at any one moment, the chain of joints and CVTs limits the end effector to a single DOF. By varying the gear ratios of the CVTs, a Cobot controller can orient the single free DOF so that the user follows a smooth pathway. Cobots approach the problem of constraining motion from the opposite perspective to most methods covered in this survey. Rather than having a free device, which when interacting with a constraint, has some DOFs restricted, a Cobot is a constrained device, which when not interacting with a constraint, has some DOFs simulated. The DOF simulation is achieved using a force sensor mounted on the end effector. The controller uses the force sensor to interpret the intended direction of motion and, if this direction is permitted by the constraints, the CVTs are reconfigured to align the single free DOF with this direction. By attaching a motor to the CVT chain, so-called 'intelligent assist devices" can be created, although these are no longer passive [109]. These devices actively assist a user working with heavy payloads, while still allowing for the enforcement of constraints.

VII. Advanced Constraint Extensions

The three previous sections described the core concepts behind the most common active constraint approaches within the literature. Several research groups have built upon these fundamental elements to create advanced active constraints that overcome specific limitations, or assist in more complex tasks. An overview of such "advanced constraint extensions" found within the literature is given in this section. In each case considered, the research does not involve an entirely new control method, rather, each is based upon one or more of the methods described in the preceding sections. These advanced constraint extensions have been separated from the methods upon which they are based, because they can be considered higher level concepts, which have the potential to be applied across a wider range of methods and applications.

A. Adaptive Constraints

Various methods have been proposed in the literature for applying active constraints in an adaptive manner. In each of these methods, the constraint is enforced using one of the previously discussed methods; however, the controller decides when and how to apply the constraints based on some knowledge of the task, hardware or users.

Selective application of active constraints has been achieved by several groups using hidden Markov models (HMM). By using HMMs to classify and interpret user commands, these groups are able to adaptively apply active constraints to situations that have not been explicitly preprogrammed. In [54], complex high level tasks are broken down into simple subtasks, each of which has a straight line guidance constraint of its own. Here, the current state is identified using an HMM, so that the controller can apply a guidance constraint appropriate to the subtask associated with it. Yu *et al*. applied a similar approach based on subtasks, but rather than just straight lines, they also considered curved constraints for positioning and obstacle avoidance [111]. Procedure specific adaptive constraints are shown in [112] where an HMM-based controller is trained on expert procedures from which it learns tool positions which should be taken during the procedure. Guidance constraints are then implemented by clustering the tool positions into discrete states and constructing an HMM for transitions between them. Based on the possible state transitions at any moment, the constraint guides the user toward the next state which was learnt from the expert.

When the user needs to depart from a constraint due to unpredicted obstacles or unplanned targets, it can be problematic with many constraint implementations. If, for example, a particularly stiff guidance constraint is in use, and a physical obstacle blocks the robot's path, the robot can become immobilized. In [55], a process for selecting the optimal constraint compliance value with respect to this problem is reported. The method described is predicated on finding a compromise between the level of permissible accidental violation and the need for freedom of intentional violation. For some applications it might not be possible to make concessions on either of these aspects. In [113], a solution is proposed. In this study, three methods for scaling constraint effects ("toggle," "fade," and "hold") and three methods for deciding when to apply the scaling ("explicit," "implicit," and "autonomous") were studied. In [114] and [115], HMMs are used to identify when a user was consciously intending to leave a guidance constraint as a result of an obstacle blocking the path. Once a user's motion was classified as intentionally violating the constraint, the anisotropy on the reference direction fixture was removed and user motion was smoother and faster as a result. Passenberg *et al*. used "interaction forces" between the user and robot to achieve a similar autonomous response [116]. By monitoring how much "agreement" there is between the user and robot they classify intentional violation cases and react accordingly.

B. Compliance in Tools and Robot Mechanisms

Compliance within a robot can reduce the positional accuracy of active constraints because the CTG will not be in the exact

position reported by the robot joint sensors. Beasley and Howe compared the effect of tool-tip tracking and flexion modeling for improving the accuracy of long thin surgical instruments under regional active constraints [117]. They found that incorporating the Euler–Bernoulli beam bending model could significantly reduce maximum constraint penetration over a rigid tool assumption. It was also found that feeding back the tracked tool tip could significantly reduce the maximum constraint penetration further.

Marayong *et al*. investigated potential methods, which could be used to accommodate for admittance controlled compliant robot mechanisms. They initially proposed an open-loop control architecture based on a model of human hand dynamics under "equilibrium-point control" [118]. Using a unified mass-spring-damper model of the hand, robot, and constraint, they were able to adaptively position the constraint so that the user's hand would be stopped before it penetrated a restricted region, irrespective of compliance between the robot actuators and the hand. The method was successful in preventing region penetrations; however, it was conservative and generally stopped the user's hand well before the region. They later implemented a closed-loop controller using a camera to measure and feedback the deviation between the rigid kinematic and actual end effector positions [119]. By interpolating between low-frequency images with a Kalman filter, Marayong *et al*. were able to incorporate a linear dynamic model of the manipulator compliance in their controller so that the actual end effector position could be accurately constrained.

C. Dynamic Active Constraints

Dynamic active constraints are those where the constraint geometry moves continuously, as a result of changes in the physical environment or task being undertaken.

An abstract investigation of dynamic active constraints was carried out using simple proximity based constraints (as discussed in Section VI-A) by Gibo *et al*. [40]. They constructed an experiment where a linear actuator was used to move a soft-tissue phantom in 1 DOF, while a human user attempted to depress the phantom by a fixed amount using a teleoperated robot. By constructing a regional dynamic active constraint, which followed the tissue phantom at the required tissue depth, the user was assisted while the tissue was moved periodically and randomly. Gibo *et al*. used two methods for computing the necessary position of the dynamic constraint; one based on the current tissue position and one based on its predicted position. They found that the two methods were comparable and both were significant improvements on unconstrained teleoperation and static constraints.

Much of the remaining dynamic active constraint research has focused on applications for surgery on a beating heart. Navkar *et al*. considered the heart's left ventricle and generated multidimensional dynamic active constraints based on a simple proximity function [42]. They constrained the end effector of a simulated surgical tool to follow a "dynamic guidance curve" between the inner walls of a beating heart which were generated in real time. They used a haptic master device to render a constraint force to the user. The results of experiments with this system show that off-path error was reduced compared with cases with only visual guidance and cases without any guidance.

Potential fields were used by Ren *et al*. to constrain a simulated surgical tool to the heart's external surface [39]. In their method, a series of preoperative images are taken of the heart throughout its pulsatile cycle. Each 3-D time slice is segmented to extract the geometry of the surface of the heart, and image dilation is used to create an explicit distance map from the heart's surface. The distance map is then used to train a B-spline-based artificial neural network, as described previously, allowing for highly efficient real-time evaluation. Intraoperatively, the time slices are synchronized with the patient using electrocardiography and the controller uses the neural network to efficiently establish the distance of the CTG from the closest point on the heart's surface. Values for the potential are computed from the proximity values using either a generalized sigmoid function (for regional constraints) or a Gaussian function (for guidance constraints), allowing the constraint forces to be computed in the standard way.

The preoperative acquisition and processing of medical images within the aforementioned method means that it is unable to adapt to intraoperative changes in the heart's motion. By acquiring a real-time model of a beating heart using a range camera, Rydén and Chizeck were able to overcome this limitation [41]. By considering each point within a point cloud as an individual spherical constraint, Rydén and Chizeck's method creates implicit surfaces, which the constrained tool cannot penetrate.

A hyper-redundant "snake" robot was dynamically constrained using joint optimization for camera positioning by Kwok *et al*. in cardiac surgery [120] and colonoscopy [43], and by Vitiello *et al*. in liver surgery [121]. They defined a method for creating tubular dynamic active constraint pathways by placing cspline control points, which move relative to a tissue's deforming surface. By creating an exponential cost function based on the robot body's infractions outside of the tubular constraint, joint optimization was used to position the body of the snake, while the camera was focused on the user-defined target. They also showed how GPU acceleration could allow for complex proximity queries between dynamic surfaces in real time.

In [44], the properties of the haptic interaction between a user and a dynamic active constraint were considered. As the boundary of a dynamic active constraint can move, it is possible for it to move past a static tool so that it switches from unconstrained to constrained regions. When this happens, conventional constraint enforcement methods can lead to undesirable active tool motion. In this research, a dynamic constraint based on a friction model is proposed, which remains passive and avoids autonomous motion.

D. Multihanded Active Constraints

When multiple manipulators are operating within the same workspace, multihanded active constraints can be used [122]. The challenge for multihanded tasks is in enforcing temporally and spatially dependent relationships between the manipulators while still ensuring that each one conforms to its individual

environmental and physical constraints. The method in [122] uses constrained joint optimization with a supervisory optimization function, which enforces interrobot constraints, while seeking to minimize the weighted sum of the cost functions for each individual robot. This method was used for a simplified knot positioning task, where two robots held opposite ends of the thread. This application is a good demonstration of multirobot constraints because the relative angle and velocity with which the robots pull on their ends of the thread determines how close the knot is to the desired position. An interesting effect of these constraints is that by removing the user guidance from one of the robots, it essentially becomes autonomous and follows the constrained optimum position based on the position of the other. In [123], it is shown how this method could be implemented on a commercial teleoperation system for surgery.

E. Active Constraints on Handheld Manipulators

It is shown by Becker *et al*. that constraints can be applied via handheld manipulators, which are not physically attached to the ground in the conventional way [124]. In this research, a handheld micromanipulator is used, where actuators are located between the user's hand and the tool-tip, such that the controller can compensate for unintentional motion like hand tremor. Visually tracking the micromanipulator, Becker *et al*. computed the relative position of the tool-tip to a constraint geometry. Using this relationship with an admittance control law similar to the reference direction fixtures above, they could use the handheld actuators to minimize the error from preplanned positions and pathways. Subsequent research by the same group demonstrates how a vision system can be used to estimate the surface geometry of a retinal membrane allowing dynamic active constraints, which prevent over-penetration during a surgical procedure [125].

VIII. Discussion

Active constraints and virtual fixtures are robot control strategies that, through physical human–robot interaction, assist users performing manipulation tasks. The assistance is achieved by anisotropically regulating the user's motion, such that deviations from a plan are attenuated and conformance to a plan is encouraged. This survey presents the current state of published literature regarding these synonymous methods. To portray the wide range of research in a systematic way, a generalized active constraint framework has been proposed which consists of three primary components. Initially, in a process called "constraint definition," the geometries, which constitute the constraints, are input to the system and are represented computationally. At each time step during operation, the relative configuration of the constraint geometry and the robot's current location is computed. This stage is termed "constraint evaluation." Finally, "constraint enforcement" is a process whereby the relative robot-constraint configuration is converted into commands that can be sent to the robot hardware to enact the constraint. Although all of these elements are required for a complete implementation, each publication tends to focus on certain elements. A breakdown of the contribution to methods for each of these stages from the literature is given in Table I.

The procedures used to define constraints are specific to each application, and therefore this aspect has not been discussed in detail. However, the methods available to computationally represent constraint geometries are universally applicable and should be considered prior to constructing an active constraint system. Ten constraint representation methods have been found within the literature, ranging from simple points and lines, up to meshes and point clouds. At present, the simple methods are far more widespread within the literature than the complex ones (85% of references in Table I). By applying them creatively, it has been shown how simple constraint representations can be used for a range of practical tasks. However, they are inevitably less flexible than those which are more complex, and this can lead to difficulties in unstructured environments. It is likely that the pervasiveness of simple constraint representations is due to the focus within the literature on methods for constraint enforcement. Using simple constraint representations for abstract tasks facilitates the unbiased validation of constraint enforcement methods. Additionally, when complex constraint representations are used, the subsequent step of constraint evaluation also becomes more demanding.

For constraint evaluation, the process of taking a robot pose and a constraint geometry and computing their relative configuration, methods involve a similar range of complexity. Simple methods, which rely on linear algebra, lookup tables, or exhaustive searches are also much more common within the literature than those that involve geometric algorithms for collision detection and closest point searches (90% of references in Table I). The choice of constraint evaluation method is based upon the geometric properties of the chosen constraint representation and methods available from within the computer graphics literature.

The published approaches to enforcing constraints are largely independent of the methods selected for the preceding stages of constraint representation and evaluation. In general, constraint enforcement algorithms consider the relative robot-constraint configuration as an input parameter, and are indifferent to how it was produced. Nonetheless, constraint enforcement is intrinsically linked to the robot hardware through which constraints will be applied, and therefore this aspect requires consideration. The main factor dictating potential enforcement methods is whether the constrained device is impedance controlled, admittance controlled, or constructed with some other mechanism. Some methods within the literature have embodiments for more than one of these options, however, most are specialized. Although constraints on all hardware have the same overall goal, the haptic properties of each are different, and they interact with users in a different way. Admittance controlled devices, for example, can implement very stiff constraints, which never permit access to constrained regions; however, the mechanism's rigidity can stifle any tactile feedback from the environment.

In addition to the generalized active constraint framework, this survey has detailed several advanced extensions that expand the core active constraint concepts to overcome limitations or increase functionality.

TABLE I
TABULATED SUMMARY OF RESEARCH CONTRIBUTIONS FROM THE KEY LITERATURE WITHIN THIS SURVEY (81 PAPERS)

		Representation										Evaluation				Enforcement							Admittance	Impedance	Hands-on	Teleoperated	Regional	Guidance	Static	Dynamic
First Author (Year)	References	Points	Lines	Parametric Curves	Planar	Parametric Surface	Polygonal Mesh	Point Cloud	Volumetric Primitives	Explicitly Described	Artificial Neural Network	Closed Form	Explicit	Hierarchical Representations	Feature Tracking	Simple Functions of Proximity	Proxy and Linkage	Non-Energy Storing	Potential Fields	Reference Direction Fixtures	Constrained Joint Optimisation	Passive Mechanisms								
Aarno (2005)	[54]		●									●								●				●		●	●		●	
Abbott (2003)	[19]				●							●				●							●	●		●	●		●	
Abbott (2006)	[82]				●							●				●								●		●	●		●	
Aigner (1997)	[65]								●	●			●						●							●	●	●	●	
Aleotti (2005)	[61]						●								●											○		○	●	
Ammi (2007)	[86]			●												●			●					●		●	●	●	●	
Arata (2010)	[64]								●			●					●							●			●		●	
Beasley (2009)	[117]				●							●								●			●			●	●		●	
Becker (2011)	[124]	●		●		○			○			●								●					●		●		●	○
Becker (2012)	[125]				●							●								●					●		●		●	○
Bettini (2001)	[28]	●		●								●								●			●		●		●	●		
Bettini (2004)	[30]	●		●					●			●								●			●		●		○	●		
Bowyer (2013)	[44]			●								●						●												●
Burschka (2005)	[49]					●														●			●		●			●		
Davis (1997)	[106]			●																		●			●			●	●	
Borghesan (2012)	[102]				●							●									●			●		●	●		●	
Burghart (1999)	[27]		●						●			●				○				●			●		●		○	●		
Castillo-Cruces (2010)	[33]			●																●			●		●		●	●		
Chong (2001)	[84]	●										●				●			●					●			●	●		
Funda (1996)	[100]				●							●									●			○			●		●	
Ghanbari (2010)	[96]				●	●			●			●							●					●			●	●	●	
Gibo (2009)	[40]				●							●				●								●			●	●		●
Gillespie (2001)/Peshkin (2001)	[107], [108]			●								○										●			●		●	●		
Hager (2002)	[97]	○																		●			●				○	●		
Hennekens (2008)	[93]				●							●						●						●			●	●		
Ho (1994/1995)	[8], [36]			●						●		○	●				●			●					●		●	●	●	
Jang (2012)	[48]						●							●													○	●		
Joli (2007)	[83]	●	●		●							●								●	●							●		
Joly (1995)/Micaelli (1998)	[12], [90]			●					●			○				●	●							●		●	●	●		
Kapoor (2005)	[52]	●	●									●									●		●		●			●		
Kapoor (2006)	[59]	●	●		●							●									●						●	●		
Kapoor (2008)	[122]											●									●							●		●
Kikuuwe (2008)	[92]			●	●				●			●						●					●	●		●	●	●		
Kosuge (1995)	[11]	●										○				●	○							●			●	●		
Kragic (2005)	[115]	○		●																●			●		●			●	●	
Kuang (2004)	[34]								●			●							●						●			●	●	
Kumar (2003)	[46]	●										●								●			●		●			●		
Kwok (2010/2013)/Vitiello (2011)	[43], [120], [121]			●					●			●									●					●	●			●
Li (2003/2007)	[45], [62]						●							●							●		●		●		●	●		
Li (2003)	[114]			●								●								●			●		●		●	●		
Li (2005)	[25]	●		●	●							●									●		●		●		●	●		
Li (2005)	[56]			●			●							●						●	●		●		●	●	●	●		
Marayong (2004)	[55]			●								●								●			●		●			●		
Marayong (2003)	[98]				●							●								●			●		●		●	●		
Marayong (2006)	[118]				●							●								●			●		●			●		○
Marayong (2008)	[119]				●							●				●							●		●		●		●	○
Matinfar (2007)	[99]				●							●								●			●		●		●		●	
Navkar (2012)	[42]			●												●								●		○		●		●
Nolin (2003)	[113]				●							●				●								●		●	●		●	
Ortmaier (2006)	[85]	●	●									○				●								●		●		●	●	
Park (2001)	[58]				●																						●	●		
Park (2011)	[47]			●								●				○								●			●	●		
Passenberg (2011)	[116]		●									○				●				●			●					●		
Payandeh (2002)	[24]								●				●							●				●			●	●		
Pezzementi (2007)	[32]											●				●				●			○	●	●			●		
Prada (2005/2009)	[29], [38]	●							●			●		●		●								●			●	●	●	
Ren (2008)	[39]			●							●								●					●			●	●		●
Rosenberg (1992/1993)	[2], [3]				●																						●	●		
Ryden (2012)	[41]							●				●					●							●			○	●		●
Sayers (1992)	[31]	●	●		●							●				●							●				●		●	
Seung (2010)	[57]				●							●				●											●	●		
Srimathveeravalli (2007)	[53]		●	●								●				●								●			○		●	
Taylor (1991)	[103]	●										●								○		●		○		●			●	
Troccaz (1996)	[105]	●		●		○			●			●								○		●			●	●			●	
Troccaz (1998)	[110]															●						●		●	●		●	●	●	○
Turro (2001)	[26]						●			●		●		●					●					●			●	●	●	
Unruh (1993)	[10]	●										●				●				●				●			●	●	●	
Weede (2011)	[112]								●			●											○		○		○	●		
Xia (2008)	[37]				●	●						●				●				●			●		●		●		●	
Xia (2011)	[123]		●									●									●			●		●		●	●	
Yamamoto (2012)	[63]							●				●				●								●			●		●	○
Yu (2005)	[111]								●			●				●			●						●			●		●	
Zhang (2012)	[87]			●								●				●								●		○			●	
Total occurrences (only ●)	81	17	10	28	24	3	6	2	18	4	1	53	4	7	1	25	6	3	8	26	14	6	27	36	34	35	54	51	72	9

For a given paper, the symbol '●' identifies constraint methods which are an appreciable component of the work, and the symbol '○' identifies methods which are used or discussed, but are of less significance.

IX. Conclusion

Active constraints, virtual fixtures, and several other similar robot control strategies were first developed during the early 1990s. Rather than replacing a human operator with an autonomous robot, these strategies sought to augment the human by anisotropically regulating their motion, so as to prevent access to certain regions within a robot's environment, or encourage conformance to preproduced plans.

The primary benefit of collaborative robot control strategies, such as active constraints, is that as a human remains within the control loop, they can safely cope with more complex and unstructured tasks than would be possible under full automation. Nevertheless, in the early embodiments of the concept, conveniently well-structured tasks were considered. Indeed, in the first research on virtual fixtures, the forces were not generated through software; rather, they were the result of the master device physically touching a "fixture board." At the initiation of a concept, comparatively simple applications will be considered; however, there were several limitations at the time making wider application more difficult. For example, the custom robotic hardware was used in most of the early research and complete controllers had to be created. Additionally, certain components of an active constraint controller, specifically constraint evaluation, can involve computationally nontrivial algorithms (often $O(n^2)$) which, at the time, would have been very unlikely to run sufficiently fast for online use.

Since this early research, many of the barriers to wider active constraint use have been overcome. Robots suitable for physical human robot interaction, that can operate under Cartesian control "out of the box," can now be purchased. Computational power has also drastically increased since the 1990s and most algorithms pertinent to active constraints can readily be run at control rates over 1 kHz.

Due to these, and other background developments, active constraint research has grown significantly since its conception. There are currently over 200 publications in the field, with a diverse range of methods and applications considered. It is shown within the literature that constraint geometries can be computationally represented in a variety of ways and they are no longer tied to simple inflexible shapes. The use of flexible structures such as meshes and point clouds allows the constraint generator, human, or otherwise, to be much more expressive, making constraints of greater use during a task.

The fast computation of relative robot-constraint configurations is an important factor for robust and stable control of a constrained robot. In practice, the choice of a method for constraint representation is dependent on the availability of fast computational methods for evaluating it, and it is likely to have been this problem that limited early research. Modern publications have shown that collision detection algorithms for the most complex of freeform constraint surfaces can operate fast enough for stable control; however, this problem has generally been disregarded. The field of computer graphics contains numerous methods for collision detection and proximity queries; therefore, if required for an active constraint implementation, this field could provide a solution.

The key component of an active constraint controller takes the relative robot-constraint configuration and computes a set of robot commands that enforce the required constraint. This aspect has been the primary focus of publications over the years, and seven core methods for enforcing active constraints have been proposed. These methods, along with their extensions and adaptations, provide a range of functional and haptic properties for constraint implementations.

By combining these options for constraint enforcement with those for representation and evaluation, modern active constraints can incorporate many assistive properties, making them useful in a variety of practical tasks. One area where active constraints have seen significant development is in computer-assisted surgery, notably with the commercial development of the hands-on Sculptor Robotic Guidance Arm (Stanmore Implants Worldwide Ltd., Elstree, U.K.) and the RIO Robotic Arm Interactive Orthopedic System (MAKO Surgical Corp., Fort Lauderdale, FL, USA). In these embodiments, not only do active constraints supply the ability to constrain cuts for accurate sculpted surfaces, but they can also physically constrain surgeons to avoid cutting into critical areas that need protecting. In this way, active constraints provide additional safety in complex surgery.

With the maturity of methods for representing, evaluating, and enforcing constraints, the most significant challenge now seems to be the initial step of generating the constraint geometries in an effective way. In addition to more effective constraint generation methods, future research into active constraints will move toward a more complete representation of the working environment with fewer simplifications and assumptions if the current trends continue. This is expected to include the widened use of freeform geometries for representing constraints; tool/robot bodies rather than just the tool tip and dynamic active constraints to respond to changes in the working environment. As active constraint controllers are developed that can simultaneously make use of all of these elements, it is likely that they will become increasingly relevant to practical commercial tasks resulting in more widespread usage.

References

[1] E. Garcia, M. A. Jimenez, P. G. de Santos, and M. Armada, "The evolution of robotics research," *IEEE Robot. Autom. Mag.*, vol. 14, no. 1, pp. 90–103, Mar. 2007.

[2] L. B. Rosenberg, "The use of virtual fixtures as perceptual overlays to enhance operator performance in remote environments," USAF Armstrong Lab., TX, USA, Tech. Rep. AL/CF-TR-1994-0089, 1992

[3] L. B. Rosenberg, "Virtual fixtures: Perceptual tools for telerobotic manipulation," in *Proc. IEEE Virt. Real. Annual Int. Symp.*, Sep. 1993, pp. 76–82.

[4] R. H. Taylor, H. A. Paul, P. Kazanzides, B. D. Mittelstadt, W. Hanson, J. F. Zuhars, B. Williamson, B. L. Musits, E. Glassman, and W. L. Bargar, "Taming the bull: Safety in a precise surgical robot," in *Proc. IEEE Int. Conf. Adv. Robot.*, 1991, pp. 865–870.

[5] B. L. Davies, "A discussion of safety issues for medical robots," in *Computer-Integrated Surgery—Technology and Clinical Applications*, R. H. Taylor, S. Lavaliée, G. C. Burdea, and R. Mösges, Eds. Cambridge, MA, USA: MIT Press, 1996, ch. 19, pp. 287–296.

[6] Y. S. Kwoh, J. Hou, E. A. Jonckheere, and S. Hayati, "A robot with improved absolute positioning accuracy for CT guided stereotactic brain surgery," *IEEE Trans. Biomed. Eng.*, vol. 35, no. 2, pp. 153–160, Feb. 1988.

[7] B. Davies, M. Jakopec, S. J. Harris, F. Rodriguez y Baena, A. Barrett, A. Evangelidis, P. Gomes, J. Henckel, and J. Cobb, "Active-constraint robotics for surgery," *Proc. IEEE*, vol. 94, no. 9, pp. 1696–1704, Sep. 2006.

[8] S. C. Ho, B. L. Davies, R. D. Hibberd, and J. Cobb, "Robotic knee surgery—Implicit force control strategy with active motion constraint," in *Proc. Eur. Robot. Intell. Sys. Conf.*, 1994, vol. 3, pp. 1235–1248.

[9] C. Sayers and R. Paul, "Synthetic fixturing," in *Proc. ASME Winter Annu. Meet.*, 1993, vol. 49, pp. 37–46.

[10] S. G. Unruh, T. N. Faddis, and B. G. Barr, "Position-assist shared control of a force-reflecting telerobot," in *Proc. SPIE*, H. Das, Ed., vol. 1833, no. 1, 1993, pp. 113–121.

[11] K. Kosuge, T. Itoh, T. Fukuda, and M. Otsuka, "Tele-manipulation system based on task-oriented virtual tool," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 1995, vol. 1, pp. 351–356.

[12] L. D. Joly and C. Andriot, "Imposing motion constraints to a force reflecting telerobot through real-time simulation of a virtual mechanism," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 1995, vol. 1, pp. 357–362.

[13] G. Hager, A. Okamura, P. Kazanzides, L. Whitcomb, G. Fichtinger, and R. Taylor, "Surgical and interventional robotics—Part III," *IEEE Robot. Autom. Mag.*, vol. 15, no. 4, pp. 84–93, Dec. 2008.

[14] B. Volpe, H. Krebs, N. Hogan, L. Edelstein, C. Diels, and M. Aisen, "A novel approach to stroke rehabilitation robot-aided sensorimotor stimulation," *Neurology*, vol. 54, no. 10, pp. 1938–1944, 2000.

[15] V. Crocher, A. Sahbani, J. Robertson, A. Roby-Brami, and G. Morel, "Constraining upper limb synergies of hemiparetic patients using a robotic exoskeleton in the perspective of neuro-rehabilitation," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 20, no. 3, pp. 247–257, May 2012.

[16] L. Marchal-Crespo and D. J. Reinkensmeyer, "Review of control strategies for robotic movement training after neurologic injury," *J. NeuroEng. Rehab.*, vol. 6, no. 20, 2009.

[17] T. B. Sheridan, "Telerobotics," *Automatica*, vol. 25, no. 4, pp. 487–507, 1989.

[18] M. Jakopec, S. J. Harris, F. Rodriguez y Baena, P. Gomes, and B. L. Davies, "The acrobot system for total knee replacement," *Ind. Robot: Int. J.*, vol. 30, no. 1, pp. 61–66, 2003.

[19] J. J. Abbott and A. M. Okamura, "Virtual fixture architectures for telemanipulation," in *Proc. IEEE Int. Conf. Robot. Autom.*, Sep. 2003, vol. 2, pp. 2798–2805.

[20] (2012). "Active Constraint Technologies for Ill-defined or Volatile Environments (ACTIVE)," [Online]. Available: http://www.active-fp7.eu/

[21] Stanmore Implants Worldwide Ltd. (2012). "Stanmore Sculptor RGA (Robotic Guidance Arm)," [Online]. Available: http://www.stanmoreimplants.com/

[22] N. Hogan, "Impedance control: An approach to manipulation—Part 1—Theory," *J. Dyn. Sys., Measure., Cont.*, vol. 107, no. 1, pp. 1–7, 1985.

[23] J. J. Abbott, P. Marayong, and A. M. Okamura, "Haptic virtual fixtures for robot-assisted manipulation," in *Robot. Res., Springer Tracts Adv. Robot.* Berlin, Germany: Springer, 2007, vol. 28, pp. 49–64.

[24] S. Payandeh and Z. Stanisic, "On application of virtual fixtures as an aid for telemanipulation and training," in *Proc. Symp. Haptic Interf. Virt. Env. Teleop. Sys.*, 2002, pp. 18–23.

[25] M. Li, A. Kapoor, and R. H. Taylor, "A constrained optimization approach to virtual fixtures," in *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, Aug. 2005, pp. 1408–1413.

[26] N. Turro, O. Khatib, and E. Coste-Maniere, "Haptically augmented teleoperation," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2001, vol. 1, pp. 386–392.

[27] C. Burghart, J. Keitel, S. Hassfeld, U. Rembold, and H. Woern, "Robot controlled osteotomy in craniofacial surgery," presented at the Int. Worksh. Haptic Devices Med. Appl., Paris, France, Jun. 1999

[28] A. Bettini, S. Lang, A. M. Okamura, and G. D. Hager, "Vision assisted control for manipulation using virtual fixtures," in *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, 2001, pp. 1171–1176.

[29] R. Prada and S. Payandeh, "On study of design and implementation of virtual fixtures," *Virtual Reality*, vol. 13, no. 2, pp. 117–129, Jun. 2009.

[30] A. Bettini, P. Marayong, S. Lang, A. M. Okamura, and G. D. Hager, "Vision-assisted control for manipulation using virtual fixtures," *IEEE Trans. Robot.*, vol. 20, no. 6, pp. 953–966, Dec. 2004.

[31] C. Sayers, R. Paul, and M. Mintz, "Operator interaction and teleprogramming for subsea manipulation," presented at the 4th IARP Worksh. Underwater Robot., Genova, Italy, Nov. 1992.

[32] Z. Pezzementi, A. M. Okamura, and G. D. Hager, "Dynamic guidance with pseudoadmittance virtual fixtures," in *Proc. IEEE Int. Conf. Robot. Autom.*, Apr. 2007, pp. 1761–1767.

[33] R. Castillo-Cruces and J. Wahrburg, "Virtual fixtures for autonomous error compensation for human-robot cooperative tasks," *Robotica*, vol. 28, pp. 267–277, 2010.

[34] A. B. Kuang, S. Payandeh, B. Zheng, F. Henigman, and C. MacKenzie, "Assembling virtual fixtures for guidance in training environments," in *Proc. Symp. Haptic Interf. Virt. Env. Teleop. Sys.*, Mar. 2004, pp. 367–374.

[35] R. Kikuuwe, N. Takesue, A. Sano, H. Mochiyama, and H. Fujimoto, "Admittance and impedance representations of friction based on implicit Euler integration," *IEEE Trans. Robot.*, vol. 22, no. 6, pp. 1176–1188, Dec. 2006.

[36] S. C. Ho, R. D. Hibberd, and B. L. Davies, "Robot assisted knee surgery," *IEEE Eng. Med. Biol. Mag.*, vol. 14, no. 3, pp. 292–300, May/Jun. 1995.

[37] T. Xia, C. Baird, G. Jallo, K. Hayes, N. Nakajima, N. Hata, and P. Kazanzides, "An integrated system for planning, navigation and robotic assistance for skull base surgery," *Int. J. Med. Robot. Comput.-Assisted. Surg.*, vol. 4, no. 4, pp. 321–330, 2008.

[38] R. Prada and S. Payandeh, "A study on design and analysis of virtual fixtures for cutting in training environments," in *Proc. IEEE World Haptics Conf.*, Mar. 2005, pp. 375–380.

[39] J. Ren, R. V. Patel, K. A. McIsaac, G. Guiraudon, and T. M. Peters, "Dynamic 3-D virtual fixtures for minimally invasive beating heart procedures," *IEEE Trans. Med. Imag.*, vol. 27, no. 8, pp. 1061–1070, Aug. 2008.

[40] T. L. Gibo, L. N. Verner, D. D. Yuh, and A. M. Okamura, "Design considerations and human-machine performance of moving virtual fixtures," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2009, pp. 671–676.

[41] F. Rydén and H. Chizeck, "Forbidden-region virtual fixtures from streaming point clouds: Remotely touching and protecting a beating heart," in *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, Oct. 2012, pp. 3308–3313.

[42] N. V. Navkar, Z. Deng, D. J. Shah, K. E. Bekris, and N. V. Tsekos, "Visual and force-feedback guidance for robot-assisted interventions in the beating heart with real-time MRI," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2012, pp. 689–694.

[43] K.-W. Kwok, K. H. Tsoi, V. Vitiello, J. Clark, G. Chow, W. Luk, and G.-Z. Yang, "Dimensionality reduction in controlling articulated snake robot for endoscopy under dynamic active constraints," *IEEE Trans. Robot.*, vol. 29, no. 1, pp. 15–31, Feb. 2013.

[44] S. A. Bowyer and F. Rodriguez y Baena, "Dynamic frictional constraints for robot assisted surgery," in *Proc. IEEE World Haptics Conf.*, 2013, pp. 319–324.

[45] M. Li and R. H. Taylor, "Optimum robot control for 3D virtual fixture in constrained ENT surgery," in *Proc. Int. Conf. Med. Imag. Comput. Comput.-Assisted Int.* Berlin, Germany: Springer-Verlag, 2003, vol. 2878, pp. 165–172.

[46] R. Kumar, A. Kapoor, and R. H. Taylor, "Preliminary experiments in robot/human cooperative microinjection," in *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, Oct. 2003, vol. 3, pp. 3186–3191.

[47] J. W. Park, J. Choi, Y. Park, and K. Sun, "Haptic virtual fixture for robotic cardiac catheter navigation," *Artif. Organs*, vol. 35, no. 11, pp. 1127–1131, 2011.

[48] J. Jang, H. W. Kim, and Y. S. Kim, "Efficient neurosurgical monitoring method for operative safety on image-guided brain tumor resection," *Int. J. Comp. Assisted-Radiol. Surg.*, vol. 7, Suppl. 1, pp. S128–S130, Jun. 2012.

[49] D. Burschka, J. J. Corso, M. Dewan, W. Lau, M. Li, H. Lin, P. Marayong, N. Ramey, G. D. Hager, B. Hoffman, D. Larkin, and C. Hasser, "Navigating inner space: 3-D assistance for minimally invasive surgery," *Robot. Auton. Syst.*, vol. 52, no. 1, pp. 5–26, 2005.

[50] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle, "Surface reconstruction from unorganized points," *Comput. Graph.*, vol. 26, no. 2, pp. 71–78, Jul. 1992.

[51] T. K. Dey, *Curve and Surface Reconstruction: Algorithms with Mathematical Analysis*. Cambridge, U.K.: Cambridge Univ. Press, 2006.

[52] A. Kapoor, M. Li, and R. H. Taylor, "Spatial motion constraints for robot assisted suturing using virtual fixtures," in *Proc. Int. Conf. Med. Img. Comput. Comput.-Asst. Intervention*, 2005, vol. 8, no. pt. 2, pp. 89–96.

[53] G. Srimathveeravalli, V. Gourishankar and T. Kesavadas, "Comparative study: Virtual fixtures and shared control for rehabilitation of fine motor skills," in *Proc. IEEE World Haptics Conf.*, Mar. 2007, pp. 304–309.

[54] D. Aarno, S. Ekvall, and D. Kragic, "Adaptive virtual fixtures for machine-assisted teleoperation tasks," in *Proc. IEEE Int. Conf. Robot. Autom.*, Apr. 2005, pp. 897–903.

[55] P. Marayong and A. M. Okamura, "Speed-accuracy characteristics of human-machine cooperative manipulation using virtual fixtures with

variable admittance," *Human Factors: J. Human Fact. Ergon. Soc.*, vol. 46, no. 3, pp. 518–532, 2004.
[56] M. Li and R. H. Taylor, "Performance of surgical robots with automatically generated spatial virtual fixtures," in *Proc. IEEE Int. Conf. Robot. Autom.*, Apr. 2005, pp. 217–222.
[57] S. Seung, B. Kang, W. Kim, J. Park, and S. Park, "Development of image guided master-slave system for minimal invasive brain surgery," in *Proc. 41st Int. Symp. Robot. 6th German Conf. Robot.*, Jun. 2010, pp. 1–6.
[58] S. Park, R. D. Howe, and D. F. Torchiana, "Virtual fixtures for robotic cardiac surgery," in *Proc. Int. Conf. Med. Img. Comput. Comput.-Assisted. Intervention,* Springer-Verlag, 2001, pp. 1419–1420.
[59] A. Kapoor, M. Li, and R. H. Taylor, "Constrained control for surgical assistant robots," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2006, pp. 231–236.
[60] M. Ikits, J. D. Brederson, C. D. Hansen, and C. R. Johnson, "A constraint-based technique for haptic volume exploration," in *Proc. IEEE Int. Conf. Vis.*, Oct. 2003, pp. 263–269.
[61] J. Aleotti, S. Caselli, and M. Reggiani, "Evaluation of virtual fixtures for a robot programming by demonstration interface," *IEEE Trans. Syst., Man, Cybern. A*, vol. 35, no. 4, pp. 536–545, Jul. 2005.
[62] M. Li, M. Ishii, and R. H. Taylor, "Spatial motion constraints using fixtures generated by anatomy," *IEEE Trans. Robot.*, vol. 23, no. 1, pp. 4–19, Feb. 2007.
[63] T. Yamamoto, N. Abolhassani, S. Jung, A. M. Okamura, and T. N. Judkins, "Augmented reality and haptic interfaces for robot-assisted surgery," *Int. J. Med. Robot. Comput. Assisted-Surg.*, vol. 8, no. 1, pp. 45–56, 2012.
[64] J. Arata, H. Kozuka, H. Kim, N. Takesue, B. Vladimirov, M. Sakaguchi, J. Tokuda, N. Hata, K. Chinzei, and H. Fujimoto, "Open core control software for surgical robots," *Int. J. Comput.-Assisted Radiol. Surg.*, vol. 5, pp. 211–220, 2010.
[65] P. Aigner and B. McCarragher, "Human integration into robot control utilising potential fields," in *Proc. IEEE Int. Conf. Robot. Autom.*, Apr. 1997, vol. 1, pp. 291–296.
[66] A.-M. Cretu, E. M. Petriu, and G. G. Patry, "Neural network architecture for 3D object representation," in *Proc. IEEE Int. Workshop Haptic, Audio Visual Env. Appl.*, Sep. 2003, pp. 31–36.
[67] C. Ericson, in *Real-Time Collision Detection*, T. Cox, Ed. San Francisco, CA, USA: Morgan Kaufmann, 2004.
[68] E. Larsen, S. Gottschalk, M. C. Lin, and D. Manocha, "Fast proximity queries with swept sphere volumes," Dept. Comput. Sci., Univ. North Carolina, Chapel Hill, NC, USA, Tech. Rep. TR99-018, 1999
[69] P. Jimenez, F. Thomas, and C. Torras, "3D collision detection: A survey," *Comput. Graph.*, vol. 25, pp. 269–285, 2000.
[70] K. L. Clarkson, "Nearest neighbour searching and metric space dimensions," in *Nearest-Neighbor Methods for Learning and Vision: Theory and Practice*, G. Shakhnarovich, T. Darrell, and P. Indyk, Eds. Cambridge, MA, USA: MIT Press, 2005, pp. 15–59.
[71] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Commun. ACM*, vol. 18, no. 9, pp. 509–517, Sep. 1975.
[72] J. P. Williams, R. H. Taylor, and L. B. Wolff, "Augmented k-d techniques for accelerated registration and distance measurement of surfaces," in *Computer-Aided Surgery: Computer-Integrated Surgery of the Head and Spine*, Linz, Austria, 1997.
[73] G. van den Bergen, "Efficient collision detection of complex deformable models using AABB trees," *J. Graph. Tools*, vol. 2, no. 4, pp. 1–13, Jan. 1998.
[74] T. Larsson and T. Akenine-Möller, "Collision detection for continuously deforming bodies," in *Eurographics*. Ascona, Switzerland: Eurographics, 2001.
[75] J. Spillmann, M. Becker, and M. Teschner, "Efficient updates of bounding sphere hierarchies for geometrically deformable models," *J. Vis. Commun. Image Rep.*, vol. 18, no. 2, pp. 101–108, Apr. 2007.
[76] M. Lin and J. Canny, "A fast algorithm for incremental distance calculation," in *Proc. IEEE Int. Conf. Robot. Autom.*, Apr. 1991, vol. 2, pp. 1008–1014.
[77] B. Mirtich, "V-clip: Fast and robust polyhedral collision detection," *ACM Trans. Graph.*, vol. 17, pp. 177–208, 1998.
[78] J. R. Quinlan, *C45: Programs for Machine Learning*. San Mateo, CA, USA: Morgan Kaufmann, 1993.
[79] J. J. Craig, *Introduction to Robotics: Mechanics and Control*, 3rd ed., M. J. Horton, Ed. Upper Saddle River, NJ, USA: Pearson, 2005.
[80] J. J. Abbott and A. M. Okamura, "Analysis of virtual fixture contact stability for telemanipulation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, Oct. 2003, pp. 2699–2706.
[81] R. J. Adams and B. Hannaford, "Stable haptic interaction with virtual environments," *IEEE Trans. Robot. Autom.*, vol. 15, no. 3, pp. 465–474, Jun. 1999.
[82] J. J. Abbott and A. M. Okamura, "Stable forbidden-region virtual fixtures for bilateral telemanipulation," *J. Dyn. Sys., Meas., Cont.*, vol. 128, no. 1, pp. 53–64, 2006.
[83] P. Joli, S. Payandeh, M. Chan, and B. Bayart, "A new approach to solve constraint forces of virtual fixtures in haptic rendering," in *Proc. 12th World Congr. Mech. Mach. Sci. Conj. IFToMM*, 2007.
[84] N. Y. Chong, T. Kotoku, K. Ohba, and K. Tanie, "Virtual repulsive force field guided coordination for multi-telerobot collaboration," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2001, vol. 1, pp. 1013–1018.
[85] T. Ortmaier, H. Weiss, U. Hagn, M. Grebenstein, M. Nickl, A. Albu-Schaffer, C. Ott, S. Jorg, R. Konietschke, L. Le-Tien, and G. Hirzinger, "A hands-on-robot for accurate placement of pedicle screws," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2006, pp. 4179–4186.
[86] M. Ammi and A. Ferreira, "Robotic assisted micromanipulation system using virtual fixtures and metaphors," in *Proc. IEEE Int. Conf. Robot. Autom.*, Apr. 2007, pp. 454–460.
[87] D. Zhang, L. Wang, J. Gu, Z. Li, and K. Chen, "Realization of spatial compliant virtual fixture using eigenscrews," in *Proc. IEEE Int. Conf. Eng. Med. Biol. Soc.*, 2012, pp. 1506–1509.
[88] C. B. Zilles and J. K. Salisbury, "A constraint-based god-object method for haptic display," in *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, Aug. 1995, vol. 3, pp. 146–151.
[89] D. C. Ruspini, K. Kolarov, and O. Khatib, "The haptic display of complex graphical environments," in *Proc. 24th Annu. Conf. Comp. Graph. Inter. Tech.*, 1997, pp. 345–352.
[90] A. Micaelli, C. Bidard, and C. Andriot, "Decoupling control based on virtual mechanisms for telemanipulation," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 1998, vol. 3, pp. 1924–1931.
[91] J. J. Abbott and A. M. Okamura, "Pseudo-admittance bilateral telemanipulation with guidance virtual fixtures," in *Proc. Symp. Haptic Interf. Virt. Env. Teleop. Sys.*, Mar. 2006, pp. 169–175.
[92] R. Kikuuwe, N. Takesue, and H. Fujimoto, "A control framework to generate nonenergy-storing virtual fixtures: Use of simulated plasticity," *IEEE Trans. Robot.*, vol. 24, no. 4, pp. 781–793, Aug. 2008.
[93] D. Hennekens, D. Constantinescu, and M. Steinbuch, "Continuous impulsive force controller for forbidden-region virtual fixtures," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2008, pp. 2890–2895.
[94] D. Constantinescu, S. E. Salcudean, and E. A. Croft, "Haptic rendering of rigid contacts using impulsive and penalty forces," *IEEE Trans. Robot.*, vol. 21, no. 3, pp. 309–323, Jun. 2005.
[95] O. Khatib, "Real time obstacle avoidance for manipulators and mobile robots," *Int. J. Robot. Res.*, vol. 5, pp. 90–98, Apr. 1986.
[96] A. Ghanbari, H. Abdi, B. Horan, S. Nahavandi, X. Chen, and W. Wang, "Haptic guidance for microrobotic intracellular injection," in *Proc. IEEE RAS EMBS Int. Conf. Biomed. Robot. Biomech.*, Sep. 2010, pp. 162–167.
[97] G. Hager, "Human-machine cooperative manipulation with vision-based motion constraints," in *Visual Servoing via Advanced Numerical Methods*. Lecture Notes in Control and Information Sciences, vol. 401, G. Chesi, and K. Hashimoto, Eds. Berlin, Germany: Springer, 2010, pp. 55–70.
[98] P. Marayong, M. Li, A. M. Okamura, and G. D. Hager, "Spatial motion constraints: Theory and demonstrations for robot guidance using virtual fixtures," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2003, pp. 1954–1959.
[99] M. Matinfar, C. Baird, A. Batouli, R. Clatterbuck, and P. Kazanzides, "Robot-assisted skull base surgery," in *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, Nov. 2007, pp. 865–870.
[100] J. Funda, R. H. Taylor, B. Eldridge, S. Gomory, and K. G. Gruben, "Constrained cartesian motion control for teleoperated surgical robots," *IEEE Trans. Robot. Autom.*, vol. 12, no. 3, pp. 453–465, Jun. 1996.
[101] C. L. Lawson and R. J. Hanson, *Solving Least Squares Problems*. Philadelphia, PA, USA: Soc. Ind. Appl. Math., 1995.
[102] G. Borghesan, B. Willaert, and J. De Schutter, "A constraint-based programming approach to physical human-robot interaction," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2012, pp. 3890–3896.
[103] R. H. Taylor, C. B. Cutting, Y. Kim, A. D. Kalvin, D. Larose, B. Haddad, D. Khoramabadi, M. Noz, R. Olyha, N. Bruun, and D. Grimm, "A model-based optimal planning and execution system with active sensing and passive manipulation for augmentation of human precision in computer-integrated surgery," in *Proc. Int. Symp. Exp. Robot.*, 1991, pp. 177–195.
[104] R. H. Taylor, H. A. Paul, C. B. Cutting, B. D. Mittelstadt, W. Hanson, P. Kazanzides, B. L. Musits, Y.-Y. Kim, A. Kalvin, B. Haddad, D. Khoramabadi, and D. Larose, "Augmentation of human precision in

computer-integrated surgery," *Innovat. Tech. Biol. Med.*, vol. 13, no. 4, pp. 450–468, 1992.
[105] J. Troccaz and Y. Delnondedieu, "Semi-active guiding systems in surgery. A two-dof prototype of the passive arm with dynamic constraints (PADyC)," *Mechatronics*, vol. 6, no. 4, pp. 399–421, 1996.
[106] H. Davis and W. Book, "Torque control of a redundantly actuated passive manipulator," in *Proc. Amer. Control Conf.*, Jun. 1997, vol. 2, pp. 959–963.
[107] M. A. Peshkin, J. E. Colgate, W. Wannasuphoprasit, C. A. Moore, R. B. Gillespie, and P. Akella, "Cobot architecture," *IEEE Trans. Robot. Autom.*, vol. 17, no. 4, pp. 377–390, Aug. 2001.
[108] R. B. Gillespie, J. E. Colgate, and M. A. Peshkin, "A general framework for cobot control," *IEEE Trans. Robot. Autom.*, vol. 17, no. 4, pp. 391–401, Aug. 2001.
[109] J. Colgate, M. Peshkin, and S. Klostermeyer, "Intelligent assist devices in industrial applications: A review," in *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, Oct. 2003, vol. 3, pp. 2516–2521.
[110] J. Troccaz, M. Peshkin, and B. Davies, "Guiding systems for computer-assisted surgery: Introducing synergistic devices and discussing the different approaches," *Med. Image Anal.*, vol. 2, no. 2, pp. 101–119, 1998.
[111] W. Yu, R. Alqasemi, R. Dubey, and N. Pernalete, "Telemanipulation assistance based on motion intention recognition," in *Proc. IEEE Int. Conf. Robot. Autom.*, Apr. 2005, pp. 1121–1126.
[112] O. Weede, H. Monnich, B. Muller, and H. Worn, "An intelligent and autonomous endoscopic guidance system for minimally invasive surgery," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2011, pp. 5762–5768.
[113] J. T. Nolin, P. M. Stemniski, and A. M. Okamura, "Activation cues and force scaling methods for virtual fixtures," in *Proc. Symp. Haptic Interf. Virtual Env. Teleop. Sys.*, 2003, pp. 404–409.
[114] M. Li and A. M. Okamura, "Recognition of operator motions for real-time assistance using virtual fixtures," in *Proc. Symp. Haptic Interf. Virtual Env. Teleop. Sys.*, Mar. 2003, pp. 125–131.
[115] D. Kragic, P. Marayong, M. Li, A. M. Okamura, and G. D. Hager, "Human–machine collaborative systems for microsurgical applications," *Int. J. Robot. Res.*, vol. 24, no. 9, pp. 731–741, 2005.
[116] C. Passenberg, R. Groten, A. Peer, and M. Buss, "Towards real-time haptic assistance adaptation optimizing task performance and human effort," in *Proc. IEEE World Haptics Conf.*, Jun. 2011, pp. 155–160.
[117] R. A. Beasley and R. D. Howe, "Increasing accuracy in image-guided robotic surgery through tip tracking and model-based flexion correction," *IEEE Trans. Robot.*, vol. 25, no. 2, pp. 292–302, Apr. 2009.
[118] P. Marayong, G. D. Hager, and A. M. Okamura, "Effect of hand dynamics on virtual fixtures for compliant human–machine interfaces," in *Proc. Symp. Haptic Interf. Virt. Env. Teleop. Sys.*, Mar. 2006, pp. 109–115.
[119] P. Marayong, G. D. Hager, and A. M. Okamura, "Control methods for guidance virtual fixtures in compliant human–machine interfaces," in *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, Sep. 2008, pp. 1166–1172.
[120] K.-W. Kwok, V. Vitiello, and G.-Z. Yang, "Control of an articulated snake robot under dynamic active constraints," in *Proc. Int. Conf. Med. Img. Comput. Comput.-Asst. Int.*, 2010, vol. 6363, pp. 229–236.
[121] V. Vitiello, K.-W. Kwok, C. Payne, and G.-Z. Yang, "DOF minimization for optimized shape control under active constraints for a hyper-redundant flexible robot," in *Inf. Process. Comput.-Assisted Interventions*, 2011, vol. 6689, pp. 67–78.
[122] A. Kapoor and R. H. Taylor, "A constrained optimization approach to virtual fixtures for multi-handed tasks," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2008, pp. 3401–3406.
[123] T. Xia, A. Kapoor, P. Kazanzides, and R. Taylor, "A constrained optimization approach to virtual fixtures for multi-robot collaborative teleoperation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, Sep. 2011, pp. 639–644.
[124] B. C. Becker, R. A. MacLachlan, G. D. Hager, and C. N. Riviere, "Handheld micromanipulation with vision-based virtual fixtures," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2011, pp. 4127–4132.
[125] B. Becker, R. MacLachlan, L. Lobes, and C. Riviere, "Vision-based retinal membrane peeling with a handheld robot," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2012, pp. 1075–1080.

Stuart A. Bowyer (S'11) received the B.Eng. degree in mechanical engineering from Loughborough University, Leicestershire, U.K., in 2007 and the M.Sc. degree in computer science from the University of Oxford, U.K., in 2010. He is currently working toward the Ph.D. degree in medical robotics with the Mechatronics in Medicine Laboratory at Imperial College London, U.K.

Brian L. Davies received the Master's degree in mechanical engineering from University College London, London, U.K., in 1970, the Ph.D. degree in medical robotics from Imperial College London, in 1995, and the D.Sc. degree in 2001 for his international contribution to Robotic and Computer Aided Surgery systems.

He is an Emeritus Professor of medical robotics and Senior Research Investigator at Imperial College London. He developed the world's first surgical robot to remove quantities of tissue from a human patient in a clinical trial in April 1991, called the Probot. He subsequently pioneered the concept of active constraints and was a co-founder of the spin-off company Acrobot Ltd. Since 2006, he has been a Consultant for the Advanced Robotics Group Italian Institute of Technology, Genoa, Italy.

Ferdinando Rodriguez y Baena (M'09) received the M.Eng. degree in mechatronics and manufacturing systems engineering from King's College London, U.K., in 2000 and the Ph.D. degree in medical robotics from Imperial College London, in 2004.

He is currently a Reader in medical robotics with the Department of Mechanical Engineering at Imperial College, where he leads the Mechatronics in Medicine Laboratory. His research interests include mechatronic systems for diagnostics, surgical training, and surgical intervention.