Automatic and Flexible Robotic Drawing on Complex Surfaces With an Industrial Robot

Thomas Weingartshofer, Christian Hartl-Nesic, *Member, IEEE*, and Andreas Kugi, *Senior Member, IEEE*

***Abstract*— In industrial applications, planning and executing robot motions are crucial steps for manufacturing processes. Following the trend for customization, more flexible production systems are needed to quickly adapt the planned robot motion to new user inputs. In this work, a user-defined 2-D input pattern has to be drawn by a robot on a given 3-D object in an automated workflow. For this, two projection methods to map the 2-D input pattern to the 3-D object are presented, and robot trajectories are automatically generated based on the result of the projection methods. Furthermore, two control concepts, i.e., a pure motion control and a hybrid force/motion control, are investigated and validated by experimental results. In addition, a precise force estimation is performed to guarantee a constant normal contact force during the drawing process. The proposed automated workflow is applicable to various industrial processes, e.g., spray painting, cutting, and engraving, and provides an easy way to plan and execute robot motions based on user inputs.**

***Index Terms*— Force control concept, manufacturing process, mapping, path planning, robotic application, user interaction.**

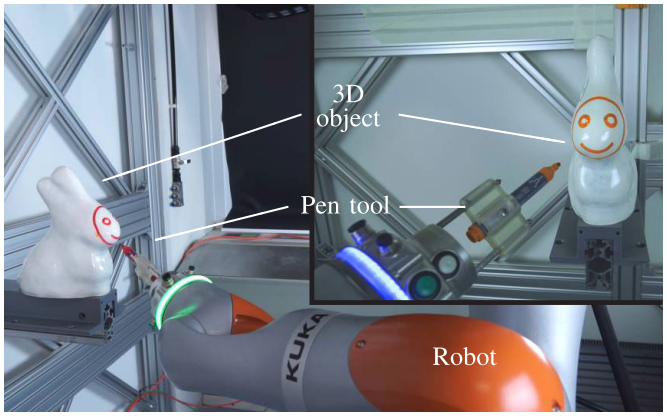


Fig. 1. Experimental setup of the drawing process with a KUKA LBR iiwa 14 R820.

I. Introduction

In automated production lines, an increasing number of industrial robots are put into operation every year [1]. The main driver for this trend is the growing product diversity in the industry, which goes up to full individualization [2]. In some production sectors, such as clothing, shoe, and apparel industry, end consumers can customize and personalize products during ordering. Custom labels, logos, and symbols' size, appearance, and location can be specified. For automation, this raises the demand for flexible production systems and workflows.

In order to keep up with these trends, manual online programming of robots becomes infeasible and Computer-Aided Design (CAD)-based offline programming using fully automated Computer-Aided Manufacturing (CAM)-based production workflows is required [3]. In most manufacturing processes, the workpiece geometry is already known, e.g., [4], or state-of-the-art 3-D scanners and algorithms can be employed to obtain the shape of the workpiece [5]. In this way, a 2-D user-generated pattern could be projected on a 3-D workpiece to automatically generate robot programs without human intervention.

In the literature, automatic generation of robot programs from a user input has been considered in several works, as discussed in the following. In many publications, a drawing process serves as a demonstration example to show the capabilities of the proposed algorithms. Note that in most cases, the considered task can be easily adapted to a different manufacturing process, e.g., engraving, laser cutting, or painting.

In [6] and [7], an edge detection algorithm is used to extract features from portraits of humans, which are drawn on a flat canvas by a humanoid robot. In order to perform this task, a task-space path is generated first, for which a joint-space path is computed using inverse kinematics, and then, it is executed by the robot. Furthermore, Jean-Pierre and Said [8] considered an industrial robot drawing on a flat whiteboard, where robot programs are automatically generated for drawing the edges and important features of ordinary photographs and images. Similarly, in [9], [10], [11], and [12], robotic drawing is presented with an emphasis on artistic and stylistic algorithms for path generation. Drawing on 2.5-D (terrain-like) objects is demonstrated in [13], where a KUKA LBR iiwa 7 R800 is utilized. The drawing is interpolated with Bézier curves and an impedance control is used to draw on unknown nonplanar objects. In [14], a force/torque sensor is mounted on the end-effector of the drawing robot. With this information, the robot is able to draw on objects, whose

Manuscript received 30 November 2022; revised 6 July 2023; accepted 13 July 2023. The authors acknowledge TU Wien Bibliothek for financial support through its Open Access Funding Programme. Recommended by Associate Editor A. Serrani. *(Corresponding author: Thomas Weingartshofer.)*

Thomas Weingartshofer and Christian Hartl-Nesic are with the Automation and Control Institute (ACIN), TU Wien, 1040 Vienna, Austria (e-mail: weingartshofer@acin.tuwien.ac.at; hartl@acin.tuwien.ac.at).

Andreas Kugi is with the Automation and Control Institute (ACIN), TU Wien, 1040 Vienna, Austria, and also with AIT Austrian Institute of Technology GmbH, 1210 Vienna, Austria (e-mail: kugi@acin.tuwien.ac.at).

Digital Object Identifier 10.1109/TCST.2023.3345209

relative position to the robot is unknown. Even on 3-D surfaces with small curvature, drawing is successful because the pen orientation is controlled to remain normal to the surface. Because the force information is only available during the drawing process and the geometry of the object is unknown, the projection can cause distortions and is not predictable. Most of the works discussed so far focus on the artistic aspects of robotic drawing on planar surfaces, but the manufacturing aspects on 3-D objects are not considered in detail.

Automatic path planning to directly work on 3-D workpieces is examined for processes such as spraying [15], polishing [16], or draping [17]. In those works, algorithms generate 3-D paths automatically based on the CAD data and no user input is required.

Another way of planning and executing robot motions from a user input is interactive teach-in methods. A state-of-the-art concept based on an instrumented tool is developed by Wandelbots [18]. With the so-called *TracePen*, robot motions are demonstrated by the user and a robot program is generated automatically. In [19], an instrumented tool is used to record the position, orientation, and force of a rope winding task. This demonstrated trajectory is then performed with an industrial robot. Inspired by the gaming industry, a representation of a robotic environment in a virtual reality could also be used to teach-in an industrial task [20]. However, those teach-in methods need a trained person to generate the robot program and this is especially cumbersome for frequently changing user inputs or small lot sizes.

A different way to customize the visual design of products is to use inkjet printer heads mounted on the end-effector of industrial robots to directly perform 2-D printing on 3-D surfaces. Thereby, the printer head is used to print custom designs on shoes [21] or cars [22] and provides an easy way for customization. In this process, the robot motion is generated for each product only once because the robot trajectory stays the same if different 2-D inputs have to be printed on the same surface area. The different designs result from different print jobs. Note that the quality of the print on curved surfaces increases for small 2-D inputs because the print head has to be in constant distance from the product.

The goal of this work is to demonstrate the full customization of products in an industrial manufacturing process. To this end, a robotic drawing task for known arbitrary 3-D objects is considered, see Fig. 1, which is the main contribution of this work. This process has high demands on the robotic system to achieve the required flexibility and replication accuracy on the 3-D surface. A user specifies the drawing and its exact location, size, and orientation on the object. Subsequently, the robot trajectory is planned using a mapping and a drawing procedure is executed. To improve the drawing quality, the contact force of the pen is adjusted using a hybrid force/torque controller, which is further extended to control the position and orientation of the pen simultaneously.

This automatic pipeline is applicable to different manufacturing processes. Robot-assisted additive manufacturing processes, such as material extrusion, fused deposition modeling (FDM), material jetting, and directed energy deposition as shown in [23], can also be performed using the proposed automatic trajectory planning and robot execution pipeline. Another example is automated milling or laser engraving, where a user input pattern has to be engraved into large 3-D objects. In subtractive processes such as laser or ultrasonic cutting, this pipeline can also be employed as well as in spray painting processes.

A preliminary version of this work is published in [24]. The main contributions beyond [24] are as follows. First, the automatic planning pipeline utilizing a least-squares conformal mapping (LSCM) is evaluated and compared to a standard parallel projection method. Second, both projection methods are investigated in new experiments, where the robot draws on challenging areas with more complex geometry and significantly higher curvature, in contrast to the plane-like area of [24]. Therefore, an adapted version of the hybrid force/motion controller, including adaptions to the null-space controller, was derived to be able to execute these challenging experiments. Third, the previously used optical measurement system to calibrate the experimental setup is replaced by an inexpensive passive mechanical method. In order to keep this work self-contained, parts of the preliminary work [24] are summarized and some aspects are elaborated in more detail.

This article is organized as follows. In Section II, two projection methods to transfer a 2-D path onto a 3-D object are introduced. Subsequently, robot trajectories are generated from the projected 3-D patterns. The control concept to control the pose (i.e., the position and orientation) of the pen and the contact force are explained in Section III. The experimental results are presented and described in Section IV. Finally, this work is concluded in Section V.

II. Path Projection and Trajectory Generation

In this section, two path projection methods to transfer a user-provided 2-D input pattern onto a 3-D object are presented and the robot trajectory generation is explained. First, a LSCM is introduced and explained in detail. Next, a simple parallel projection method is presented. Finally, the robot trajectory is generated based on the result of the two projection methods.

A. Path Projection Using LSCM [25], [26]

A flowchart illustrating the individual steps of the path projection with the LSCM is shown in Fig. 2. In order to handle meshes of high complexity, the mesh of the 3-D object is first decomposed into multiple mesh segments, which is explained in Section II-A1. This preparation step limits the local distortions of the transferred 2-D input pattern in the subsequent workflow. Next, the mesh segments are flattened using a LSCM approach, introduced in Section II-A2. The 2-D input pattern is projected on the flattened surface, and then, an inverse mapping transfers the pattern back onto the 3-D object, which is discussed in Section II-A3. The methods in this section are summarized from [25] and [26] and are tightly integrated in the path-planning workflow. Subsequently, these methods are utilized to compare the two projection methods, i.e., the LSCM and the parallel projection method.

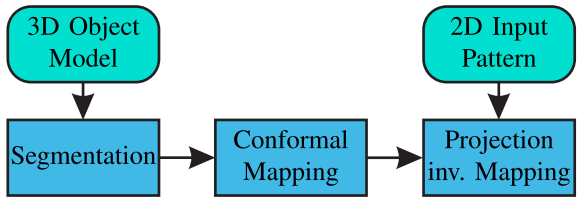
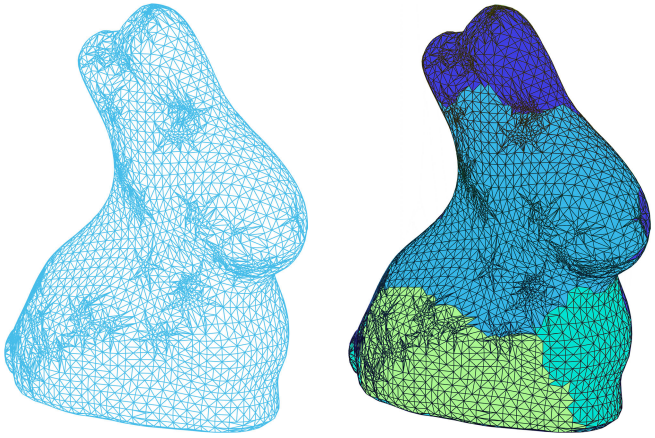


Fig. 2. Flowchart of the LSCM to project the 2-D input pattern.



(a) (b)

Fig. 3. Segmentation of a 3-D object before flattening. (a) Mesh of the 3-D object. (b) Result of the segmentation using [25].

1) Segmentation: As a first preparation step, the mesh segmentation algorithm presented in [25] is applied to the mesh of the 3-D object before flattening the individual mesh segments. In this way, distortions in the projected 2-D paths are minimized and the replication accuracy on the 3-D object is improved.

The algorithm [25] first finds the boundaries between two segments by computing the so-called sharpness criterion

$$w_{i,j} = \arccos\left(\frac{\mathbf{n}_i^{\mathrm{T}}\mathbf{n}_j}{\|\mathbf{n}_i\|_2\,\|\mathbf{n}_j\|_2}\right) \tag{1}$$

for each edge of the mesh, where $\mathbf{n}_i$ and $\mathbf{n}_j$ denote the normal vectors of two adjacent triangles T_i and T_j, respectively, see [27], and $\|\cdot\|_2$ is the Euclidean 2-norm. Edges for which the angle $w_{i,j}$ in (1) is over a certain threshold value are combined to feature curves. Next, the triangles with the maximum geodesic distance to a feature curve are determined. These triangles are subsequently used as seeds for a region growing algorithm to obtain the individual mesh segments. An example of a segmented 3-D object is shown in Fig. 3.

2) Least-Squares Conformal Mapping: The LSCM approach [25] is used to flatten the mesh segments of the 3-D object into 2-D meshes. In general, a locally isotropic conformal map $\mathcal{X} : (u, v) \mapsto (x, y)$ preserves the local angles and therefore the shape of small figures, but generally not their size.

In this section, the considered mesh segment $\mathcal{T}$ consists of n triangles T_i, $i = 1, \ldots, n$, and $\hat{n}$ vertices. Each triangle T_i has a local orthonormal basis and the coordinates of the vertices are given by $(x_{i,1}, y_{i,1})$, $(x_{i,2}, y_{i,2})$, and $(x_{i,3}, y_{i,3})$. Then, the conformal mapping for a single triangle fulfills the condition

$$\frac{\partial \mathcal{X}}{\partial u} - \mathrm{i}\frac{\partial \mathcal{X}}{\partial v} = 0 \tag{2}$$

with the complex number $\mathcal{X} = x + \mathrm{i}y$, where i denotes the imaginary unit. The inverse conformal mapping $\mathcal{U} : (x, y) \mapsto (u, v)$ reads as [28]

$$\frac{\partial \mathcal{U}}{\partial x} + \mathrm{i}\frac{\partial \mathcal{U}}{\partial y} = 0 \tag{3}$$

which is a formulation of the Cauchy–Riemann equations for complex numbers $\mathcal{U} = u + \mathrm{i}v$. Since (3) cannot be satisfied for all triangles T_i of the mesh segment $\mathcal{T}$ simultaneously, the minimization problem

$$\min_{\mathcal{U}} \sum_{T_i \in \mathcal{T}} C(T_i) \tag{4a}$$

$$C(T_i) = 2\int_{T_i} \left|\frac{\partial \mathcal{U}}{\partial x} + \mathrm{i}\frac{\partial \mathcal{U}}{\partial y}\right|^2 \mathrm{d}A_i \tag{4b}$$

is formulated, where $|\cdot|$ denotes the magnitude of the complex number $\cdot$ and A_i is the area of the triangle T_i, $i = 1, \ldots, n$.

The gradients in (4b) for a single triangle T_i read as

$$\begin{bmatrix} \frac{\partial u_i}{\partial x_i} \\ \frac{\partial u_i}{\partial y_i} \end{bmatrix} = \frac{1}{2A_i}\mathbf{B}\begin{bmatrix} u_{i,1} \\ u_{i,2} \\ u_{i,3} \end{bmatrix}, \quad \begin{bmatrix} \frac{\partial v_i}{\partial x_i} \\ \frac{\partial v_i}{\partial y_i} \end{bmatrix} = \frac{1}{2A_i}\mathbf{B}\begin{bmatrix} v_{i,1} \\ v_{i,2} \\ v_{i,3} \end{bmatrix} \tag{5}$$

with

$$\mathbf{B} = \begin{bmatrix} y_{i,2} - y_{i,3} & y_{i,3} - y_{i,1} & y_{i,1} - y_{i,2} \\ x_{i,3} - x_{i,2} & x_{i,1} - x_{i,3} & x_{i,2} - x_{i,1} \end{bmatrix} \tag{6}$$

and

$$2A_i = \left(x_{i,1}y_{i,2} - y_{i,1}x_{i,2}\right) + \left(x_{i,2}y_{i,3} - y_{i,2}x_{i,3}\right) \tag{7}$$

$$+\left(x_{i,3}y_{i,1} - y_{i,3}x_{i,1}\right) \tag{8}$$

while satisfying the Cauchy–Riemann equations

$$\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}\begin{bmatrix} \frac{\partial u_i}{\partial x_i} \\ \frac{\partial u_i}{\partial y_i} \end{bmatrix} = \begin{bmatrix} \frac{\partial v_i}{\partial x_i} \\ \frac{\partial v_i}{\partial y_i} \end{bmatrix} \tag{9}$$

see [29]. The compact formulation

$$\frac{\partial \mathcal{U}}{\partial x} + \mathrm{i}\frac{\partial \mathcal{U}}{\partial y} = \frac{\mathrm{i}}{2A_i}\begin{bmatrix} W_{i,1} & W_{i,2} & W_{i,3} \end{bmatrix}\begin{bmatrix} U_{i,1} \\ U_{i,2} \\ U_{i,3} \end{bmatrix} = 0 \tag{10}$$

is found with $U_{i,j} = u_{i,j} + \mathrm{i}v_{i,j}$ using (5) and

$$W_{i,1} = (x_{i,3} - x_{i,2}) + \mathrm{i}(y_{i,3} - y_{i,2}) \tag{11a}$$

$$W_{i,2} = (x_{i,1} - x_{i,3}) + \mathrm{i}(y_{i,1} - y_{i,3}) \tag{11b}$$

$$W_{i,3} = (x_{i,2} - x_{i,1}) + \mathrm{i}(y_{i,2} - y_{i,1}). \tag{11c}$$

Inserting (10) into the minimization problem (4), the optimization problem for the whole mesh segment $\mathcal{T}$ with all triangles $T_i \in \mathcal{T}$ is reformulated as

$$\min_{\mathbf{U}} C(\mathbf{U}) = \min_{\mathbf{U}} \sum_{T_i \in \mathcal{T}} C(T_i) \tag{12a}$$

$$C(T_i) = \frac{1}{2A_i}\left|\begin{bmatrix} W_{i,1} & W_{i,2} & W_{i,3} \end{bmatrix}\begin{bmatrix} U_{i,1} \\ U_{i,2} \\ U_{i,3} \end{bmatrix}\right|^2 \tag{12b}$$

with the coordinates of the vertices $U_{i,1}$–$U_{i,3}$ of the corresponding triangle $T_i, i = 1, \ldots, n$. Finally, to formulate the conformal mapping for the complete mesh segment $\mathcal{T}$, all vertices $\mathbf{U}^{\mathrm{T}} = \begin{bmatrix} U_1 & \cdots & U_{\hat{n}} \end{bmatrix}$ of this segment have to be considered. Note that a single vertex may be contained in multiple triangles.

In order to solve the optimization problem (12), the cost function $C(\mathbf{U})$ is decomposed in the form

$$C(\mathbf{U}) = \mathbf{U}^{*}\mathbf{N}^{*}\mathbf{N}\mathbf{U} \tag{13}$$

with the sparse index matrix $\mathbf{N} \in \mathbb{R}^{n \times \hat{n}}$ and their Hermitian conjugated matrix $\mathbf{N}^{*}$. Thereby, the elements are calculated as

$$(\mathbf{N})_{ij} = \begin{cases} \dfrac{W_{i,j}}{\sqrt{2A_i}}, & \text{if vertex } w_j \text{ belongs to triangle } T_i \\ & \text{(consisting of the vertices } w_j, w_k, w_l) \\ 0, & \text{otherwise} \end{cases} \tag{14}$$

with $W_{i,j} = (x_{i,l} - x_{i,k}) + \mathrm{i}(y_{i,l} - y_{i,k})$, cf. (11). Equation (13) is rewritten in the quadratic form

$$C(\mathbf{U}) = \|\mathbf{N}\mathbf{U}\|_2^2 = \left\|\mathbf{N}_f\mathbf{U}_f + \mathbf{N}_p\mathbf{U}_p\right\|_2^2. \tag{15}$$

The vector of mapped vertex coordinates $\mathbf{U}$ is decomposed into $\mathbf{U}^{\mathrm{T}} = [\mathbf{U}_f^{\mathrm{T}}\ \mathbf{U}_p^{\mathrm{T}}]$, where $\mathbf{U}_f$ denotes the vector of free (unknown) vertex coordinates and $\mathbf{U}_p$ are the pinned (given) vertex coordinates. As suggested in the original work [25], the two vertices with the maximum distance to each other are pinned for each mesh segment. In this way, (12) is reformulated as a least-squares problem in the unknown variables

$$\mathbf{x}^{\mathrm{T}} = \begin{bmatrix} \left(\mathbf{U}_f^R\right)^{\mathrm{T}} & \left(\mathbf{U}_f^I\right)^{\mathrm{T}} \end{bmatrix} \tag{16}$$

and

$$C(\mathbf{x}) = \|\mathbf{A}\mathbf{x} - \mathbf{b}\|^2 \tag{17}$$

with

$$\mathbf{A} = \begin{bmatrix} \mathbf{N}_f^R & -\mathbf{N}_f^I \\ \mathbf{N}_f^I & \mathbf{N}_f^R \end{bmatrix}, \quad \mathbf{b} = -\begin{bmatrix} \mathbf{N}_p^R & -\mathbf{N}_p^I \\ \mathbf{N}_p^I & \mathbf{N}_p^R \end{bmatrix} \begin{bmatrix} \mathbf{U}_p^R \\ \mathbf{U}_p^I \end{bmatrix} \tag{18}$$

where $\mathbf{N}_f$ and $\mathbf{N}_p$ denote the sparse index matrices for the free and pinned vertices, respectively, and the superscripts ^{R} and ^{I} in (16) and (18) refer to the real and imaginary parts of the complex-valued vectors and matrices, respectively.

Finally, the least-squares problem (17) is solved in MATLAB using a numerical solver in order to find the coordinates of the free vertices $\mathbf{U}_f$ under the inverse conformal mapping $\mathcal{U}$. For a more detailed explanation of the mapping algorithm, the reader is referred to [25].

3) 2-D Path Projection and Inverse Mapping: In this section, the user-provided 2-D input pattern is transferred to a user-specified location and size onto the 3-D object. To this end, the 2-D input pattern is projected on the corresponding flattened mesh segment and is subsequently mapped back onto the 3-D object by the inverse conformal mapping $\mathcal{U}$, see Section II-A2, using barycentric coordinates [26].

The 2-D input pattern is created by the user with a touchscreen, digitizer, or mouse interface and is given by the discrete path points $\mathbf{p}_{d,k}^{\mathrm{T}} = \begin{bmatrix} x_{d,k} & y_{d,k} \end{bmatrix}$, $k = 1, \ldots, K$. Furthermore, the user defines the location $\begin{bmatrix} \Delta x & \Delta y \end{bmatrix}^{\mathrm{T}}$, rotation θ, and scale s of the desired pattern on the flattened mesh segment. The resulting path points of the 2-D path $\mathbf{p}_{t,k}$, $k = 1, \ldots, K$, are computed using the transformation

$$\mathbf{p}_{t,k} = s\,\mathbf{R}(\theta)\mathbf{p}_{d,k} + \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} \tag{19}$$

with the rotation matrix

$$\mathbf{R}(\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}. \tag{20}$$

Note that if the 2-D path covers multiple mesh segments, those segments are combined and flattened again.

Next, the correspondence between the flattened mesh segment and the 3-D mesh segment is established via the triangles of both meshes. For each 2-D path point $\mathbf{p}_{t,k}$, $k = 1, \ldots, K$, from the flattened mesh segment, the associated triangle T_k is determined. This is performed using the efficient bin-based algorithm published in [26].

Finally, the 2-D path points $\mathbf{p}_{t,k}$ are mapped onto the 3-D mesh segment using barycentric coordinates inside the corresponding triangles T_k. In general, barycentric coordinates map between two arbitrary triangles based on the corresponding vertices, see, e.g., [30]. A given point $\mathbf{p}_t$ is mapped from one triangle $(\mathbf{g}_1, \mathbf{g}_2, \mathbf{g}_3)$ to another triangle $(\mathbf{w}_1, \mathbf{w}_2, \mathbf{w}_3)$ in the form, see Fig. 4

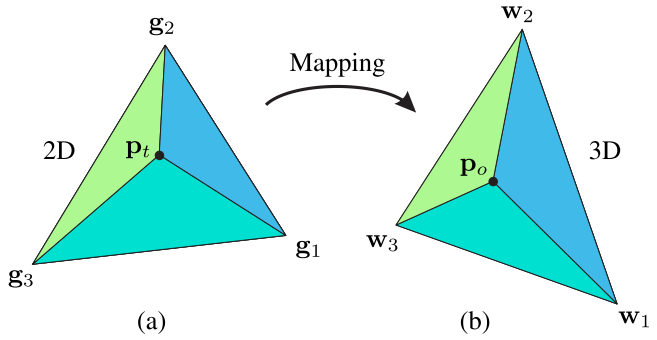


Fig. 4. Mapping from the triangle of the flattened object to the 3-D object, compare [26]. (a) Triangle on the flattened object. (b) Triangle on the 3-D object.

$$\mathbf{p}_o = \tilde{A}_1\mathbf{w}_1 + \tilde{A}_2\mathbf{w}_2 + \tilde{A}_3\mathbf{w}_3 \tag{21}$$

with

$$\tilde{A}_1 = \frac{\mathrm{S}(\mathbf{p}_t, \mathbf{g}_2, \mathbf{g}_3)}{\mathrm{S}(\mathbf{g}_1, \mathbf{g}_2, \mathbf{g}_3)} \tag{22a}$$

$$\tilde{A}_2 = \frac{\mathrm{S}(\mathbf{g}_1, \mathbf{p}_t, \mathbf{g}_3)}{\mathrm{S}(\mathbf{g}_1, \mathbf{g}_2, \mathbf{g}_3)} \tag{22b}$$

$$\tilde{A}_3 = \frac{\mathrm{S}(\mathbf{g}_1, \mathbf{g}_2, \mathbf{p}_t)}{\mathrm{S}(\mathbf{g}_1, \mathbf{g}_2, \mathbf{g}_3)}. \tag{22c}$$

In (22), the function $\mathrm{S}(\cdot, \cdot, \cdot)$ calculates the area of the enclosed triangle. In this way, all 3-D path points $\mathbf{p}_{o,k}$, $k = 1, \ldots, K$, are determined. The vertices $(\mathbf{w}_1, \mathbf{w}_2, \mathbf{w}_3)$ are represented with respect to an object frame $\mathcal{O}$, e.g., located at the origin of the 3-D object. Therefore, the points $\mathbf{p}_{o,k}$ are also described with respect to this object frame $\mathcal{O}$. For each 3-D path point,

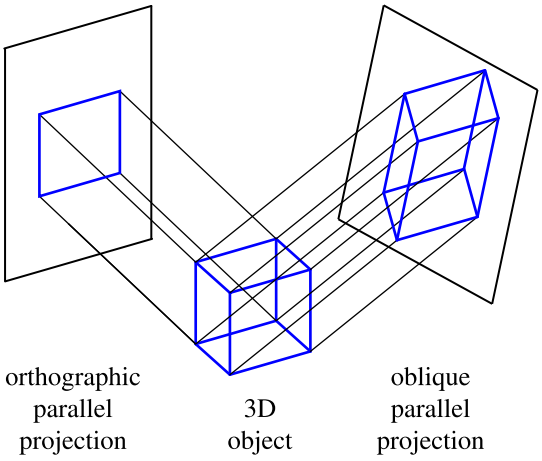


Fig. 5. Examples of parallel projections from a 3-D object to a 2-D plane, compare [31].

a coordinate frame on the surface with the origin $\mathbf{p}_{o,k}$ and the orientation $\mathbf{R}_{o,k}$ is constructed such that the z-axis, i.e., the pen, is parallel to the local surface normal vector. The remaining orientation is derived from a suitable reference orientation.

B. Parallel Path Projection

The parallel projection method is explained in this section, which is simple and serves as a comparison approach for the performance evaluation of the experimental results. The parallel projection method is mostly used to project 3-D objects on 2-D planes by projecting the points along parallel projection rays, see Fig. 5. If the 2-D plane is perpendicular to the projection rays, this projection method is called orthographic parallel projection and otherwise oblique, see [31].

In this work, the inverse projection is needed, i.e., the 2-D pattern of the user input has to be mapped to the 3-D object. Therefore, the points of the user input $\mathbf{p}_{d,k}$ are given in a 2-D plane. This 2-D plane can be either defined in front of the 3-D object by the user or it can be automatically computed based on the mean value of the normal vectors of the corresponding triangles. Then, parallel projection rays perpendicular to this 2-D plane are generated from each position $\mathbf{p}_{d,k}$. At the intersection points of the parallel rays with the surface of the 3-D object, the 3-D path points $\mathbf{p}_{o,k}$ of the 2-D input, described in the object frame $\mathcal{O}$, are found. The intersection points can be computed with, e.g., the *ray-triangle intersection* algorithm [32]. Analogous to Section II-A, local coordinate frames $\mathbf{R}_{o,k}$ are generated for each 3-D path point $\mathbf{p}_{o,k}$. The direction of the parallel projection rays is used as the z-axis of this frame. The remaining orientations are again computed based on a reference orientation. Note that the orientations of all path points $\mathbf{p}_{o,k}$ are equal since all projection rays are parallel.

C. Cartesian Robot Trajectory

The projection result of Sections II-A and II-B is described by the sequence of 3-D path points $\mathbf{p}_{o,k}$, $k = 1, \ldots, K$, which is located on the surface of the 3-D object. Along this 3-D path, the robot has to draw the user-defined pattern. Furthermore, also the corresponding orientations $\mathbf{R}_{o,k}$ are known from the projection. Based on this pose sequence, the executable robot trajectory is generated. A flowchart of the necessary computation steps is shown in Fig. 6.

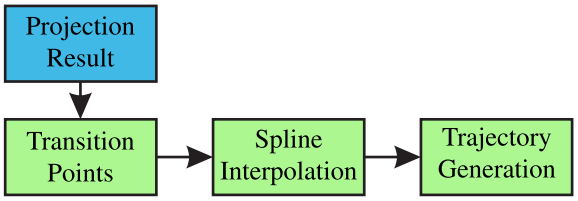


Fig. 6. Flowchart of generating a robot trajectory based on the projection result.

If a user-provided 2-D input pattern contains multiple disconnected path segments, additional transition points are added to the sequence of 3-D path points. Between two segments, the robot retracts the pen from the surface by performing a linear motion. Subsequently, the robot moves to the starting point of the next path segment and approaches the surface again. The individual sequences of 3-D path points are interpolated according to [33], where a 5th-order polynomial is used for the time parametrization. In this way, the drawing process of the robot starts and ends smoothly.

III. Control Concept

In order to realize the robotic drawing process with a pen on the 3-D object, the generated trajectory is executed by the robot using a suitable control concept, which is detailed in this section. First, a standard task-space controller is described to control the motion of the pen on the surface of the 3-D object. Second, contact force estimation [34] is introduced. Third, a hybrid force/motion controller is adapted from [35] and [36] to control the contact force during the drawing process and improve the robotic drawing task.

A. Motion Control

The dynamic robot model reads as

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) = \boldsymbol{\tau} + \boldsymbol{\tau}_{\text{ext}} \tag{23}$$

with the joint position $\mathbf{q}$, the positive definite mass matrix $\mathbf{M}(\mathbf{q})$, the Coriolis matrix $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$, and the vector of gravitational forces $\mathbf{g}(\mathbf{q})$, see, e.g., [35]. The generalized torque $\boldsymbol{\tau}$ is considered as control input and the external torques acting on the robot are denoted by $\boldsymbol{\tau}_{\text{ext}}$. The forward kinematics of the robot

$$\begin{bmatrix} \mathbf{p}_{\mathcal{B}}^{\mathcal{C}} \\ \mathbf{o}_{\mathcal{B}}^{\mathcal{C}} \end{bmatrix} = \mathbf{h}_{\mathcal{B}}^{\mathcal{C}}(\mathbf{q}) \tag{24}$$

computes the pose of the pen tip, i.e., the contact frame $\mathcal{C}$ with respect to the robot base frame $\mathcal{B}$, composed of the position $\mathbf{p}_{\mathcal{B}}^{\mathcal{C}}$ and the unit quaternion $\mathbf{o}_{\mathcal{B}}^{\mathcal{C}}$. In (24) and the remainder of this work, the notation $(\cdot)_{\mathcal{X}}^{\mathcal{Y}}$ refers to mathematical objects describing the geometric relation of the frame $\mathcal{Y}$ with respect to the frame $\mathcal{X}$, expressed in $\mathcal{X}$.

Applying inverse dynamics control, see [35]

$$\boldsymbol{\tau} = \mathbf{M}(\mathbf{q})\mathbf{v}_m + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) \tag{25}$$

to (23) and neglecting the external torques $\boldsymbol{\tau}_{\text{ext}}$ yields the new linear system dynamics in the form

$$\ddot{\mathbf{q}} = \mathbf{v}_m \tag{26}$$

with the new input $\mathbf{v}_m$. The relation between the task-space velocities of the pen tip, $\dot{\mathbf{p}}_{\mathcal{B}}^{\mathcal{C}}$ and $\boldsymbol{\omega}_{\mathcal{B}}^{\mathcal{C}}$, and the joint velocities $\dot{\mathbf{q}}$ is given by the geometric Jacobian $\mathbf{J}_{\mathcal{B}}^{\mathcal{C}}(\mathbf{q})$ as

$$\begin{bmatrix} \dot{\mathbf{p}}_{\mathcal{B}}^{\mathcal{C}} \\ \boldsymbol{\omega}_{\mathcal{B}}^{\mathcal{C}} \end{bmatrix} = \begin{bmatrix} \mathbf{J}_{\mathcal{B},v}^{\mathcal{C}}(\mathbf{q}) \\ \mathbf{J}_{\mathcal{B},\omega}^{\mathcal{C}}(\mathbf{q}) \end{bmatrix} \dot{\mathbf{q}} = \mathbf{J}_{\mathcal{B}}^{\mathcal{C}}(\mathbf{q})\dot{\mathbf{q}}. \tag{27}$$

In (27), the geometric Jacobian $\mathbf{J}_{\mathcal{B}}^{\mathcal{C}}(\mathbf{q})$ consists of the Jacobian related to the linear velocity $\mathbf{J}_{\mathcal{B},v}^{\mathcal{C}}(\mathbf{q})$ and the Jacobian related to the angular velocity $\mathbf{J}_{\mathcal{B},\omega}^{\mathcal{C}}(\mathbf{q})$. A standard task-space controller is implemented by choosing the new input $\mathbf{v}_m$ as

$$\mathbf{v}_m = \left(\mathbf{J}_{\mathcal{B}}^{\mathcal{C}}(\mathbf{q})\right)^{\dagger} \left(\mathbf{v}_c - \dot{\mathbf{J}}_{\mathcal{B}}^{\mathcal{C}}(\mathbf{q})\dot{\mathbf{q}}\right) \tag{28}$$

with the control input

$$\mathbf{v}_c = \begin{bmatrix} \ddot{\mathbf{p}}_{\mathcal{B}}^{\mathcal{D}} + \mathbf{K}_D \dot{\tilde{\mathbf{p}}} + \mathbf{K}_P \tilde{\mathbf{p}} + \mathbf{K}_I \int \tilde{\mathbf{p}}\, \mathrm{d}t \\ \dot{\boldsymbol{\omega}}_{\mathcal{B}}^{\mathcal{D}} + \mathbf{K}_\omega \tilde{\boldsymbol{\omega}} + K_o \mathbf{e}_o \end{bmatrix} \tag{29}$$

where $\mathbf{A}^{\dagger} = \mathbf{A}^{\mathrm{T}}(\mathbf{A}\mathbf{A}^{\mathrm{T}})^{-1}$ is the right pseudoinverse of the matrix $\mathbf{A}$. In (29), the desired Cartesian trajectory $[(\mathbf{p}_{\mathcal{B}}^{\mathcal{D}}(t))^{\mathrm{T}}\ (\mathbf{o}_{\mathcal{B}}^{\mathcal{D}}(t))^{\mathrm{T}}]$, with the desired frame $\mathcal{D}$, is introduced by transforming the robot trajectory from the object frame $\mathcal{O}$, see Section II-C, to the robot base frame $\mathcal{B}$. The geometric relation of the object frame $\mathcal{O}$ with respect to the robot base frame $\mathcal{B}$ is determined using a calibration procedure before the experiment, see, e.g., [37]. Hence, the position error of the controller (29) is computed as $\tilde{\mathbf{p}} = \mathbf{p}_{\mathcal{B}}^{\mathcal{D}} - \mathbf{p}_{\mathcal{B}}^{\mathcal{C}}$, the angular velocity error is computed as $\tilde{\boldsymbol{\omega}} = \boldsymbol{\omega}_{\mathcal{B}}^{\mathcal{D}} - \boldsymbol{\omega}_{\mathcal{B}}^{\mathcal{C}}$, and the quaternion error $\mathbf{e}_o$ is the vector part of the quaternion product $\mathbf{o}_{\mathcal{B}}^{\mathcal{D}} \otimes (\mathbf{o}_{\mathcal{B}}^{\mathcal{C}})^{-1}$, see [35], [38]. The control matrix $\mathbf{K}_\omega$ is diagonal and positive definite, the gain $K_o > 0$, and the choice of the gain matrices $\mathbf{K}_D$, $\mathbf{K}_P$, and $\mathbf{K}_I$ and the proof of the asymptotic stability of (29) are given in [39].

B. Contact Force Estimation

In this work, no external force/torque sensor is available to directly measure the interacting forces between the pen and the 3-D object. Instead, the contact force estimation [34] is employed and summarized in this section.

Based on the dynamic robot model (23), Magrini et al. [34] showed that using the residual vector

$$\mathbf{r}(t) = \mathbf{K}_c \left(\mathbf{M}(\mathbf{q})\dot{\mathbf{q}} - \int_0^t \boldsymbol{\tau} + \mathbf{C}^{\mathrm{T}}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} - \mathbf{g}(\mathbf{q}) + \mathbf{r}\, \mathrm{d}s \right) \tag{30}$$

leads to the residual dynamics

$$\dot{\mathbf{r}}(t) = \mathbf{K}_c(\boldsymbol{\tau}_{\text{ext}} - \mathbf{r}(t)) \tag{31}$$

from which the external torques $\boldsymbol{\tau}_{\text{ext}} \approx \mathbf{r}$ are estimated with a large positive definite gain matrix $\mathbf{K}_c$.

In general, the relation between the contact forces and moments $\mathbf{h}^{\mathrm{T}} = [\mathbf{f}^{\mathrm{T}}\ \mathbf{m}^{\mathrm{T}}]$ and the external torques $\boldsymbol{\tau}_{\text{ext}}$ is established with the geometric Jacobian $\mathbf{J}_{\mathcal{B}}^{\mathcal{C}}(\mathbf{q})$ of the contact frame $\mathcal{C}$ as

$$\boldsymbol{\tau}_{\text{ext}} = \left(\mathbf{J}_{\mathcal{B}}^{\mathcal{C}}(\mathbf{q})\right)^{\mathrm{T}} \mathbf{h}. \tag{32}$$

Assuming a point contact of the pen with the object's surface, the moment $\mathbf{m}$ in the contact frame $\mathcal{C}$ vanishes and the estimated force $\hat{\mathbf{f}}$ results from

$$\begin{bmatrix} \hat{\mathbf{f}} \\ \hat{\mathbf{m}} \end{bmatrix} = \left((\mathbf{J}_{\mathcal{B}}^{\mathcal{C}}(\mathbf{q}))^{\mathrm{T}} \right)^{\dagger} \mathbf{r}. \tag{33}$$

C. Hybrid Force/Motion Control

In this section, the hybrid force/motion controller proposed in [35] and [36] is extended for the robotic drawing task. To this end, the lateral position and the orientation of the pen are controlled simultaneously, while the estimated contact force $\hat{\mathbf{f}}$ at the pen tip along the surface normal vector is regulated to the desired value.

Since a point contact is assumed, $\mathbf{m} = \mathbf{0}$ holds for (32), and the inverse dynamics control law for (23) reads as

$$\boldsymbol{\tau}_1 = \mathbf{M}(\mathbf{q})\,\mathbf{a} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\,\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) - \left(\mathbf{J}_{\mathcal{B},v}^{\mathcal{C}}(\mathbf{q})\right)^{\mathrm{T}} \hat{\mathbf{f}} \tag{34}$$

with a new control input $\mathbf{a}$. Note that the external torque $\boldsymbol{\tau}_{\text{ext}}$ is assumed to be compensated with the last term in (34). The new control input $\mathbf{a}$ is chosen as

$$\mathbf{a} = \left(\mathbf{J}_{\mathcal{B}}^{\mathcal{C}}(\mathbf{q})\right)^{\dagger} \mathbf{T}_{\mathcal{B}}^{\mathcal{C}}(\mathbf{q}) \mathbf{Y}_v \check{\mathbf{v}}_c + \mathbf{M}^{-1} \left(\mathbf{J}_{\mathcal{B}}^{\mathcal{C}}(\mathbf{q})\right)^{\mathrm{T}} \mathbf{T}_{\mathcal{B}}^{\mathcal{C}}(\mathbf{q})\, \mathbf{Y}_f \mathbf{v}_f - \left(\mathbf{J}_{\mathcal{B}}^{\mathcal{C}}(\mathbf{q})\right)^{\dagger} \dot{\mathbf{J}}_{\mathcal{B}}^{\mathcal{C}}(\mathbf{q})\, \dot{\mathbf{q}} \tag{35}$$

with the motion control input $\check{\mathbf{v}}_c$ and the force control input $\mathbf{v}_f$ to be defined later. In order to apply the control concept with respect to the contact frame $\mathcal{C}$, the transformation matrix given by

$$\mathbf{T}_{\mathcal{B}}^{\mathcal{C}}(\mathbf{q}) = \begin{bmatrix} \mathbf{R}_{\mathcal{B}}^{\mathcal{C}}(\mathbf{q}) & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \tag{36}$$

is utilized in (35). Moreover, the constant selection matrices

$$\mathbf{Y}_f = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad \mathbf{Y}_v = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{37}$$

are introduced and $\check{\mathbf{v}}_c$ from (35) is similar to $\mathbf{v}_c$ from (29) by replacing $\tilde{\mathbf{p}}$, $\mathbf{e}_o$, $\dot{\tilde{\mathbf{p}}}$, and $\tilde{\boldsymbol{\omega}}$ with

$$\begin{bmatrix} \check{\mathbf{p}} \\ \check{\mathbf{e}}_o \end{bmatrix} = \left(\mathbf{T}_{\mathcal{B}}^{\mathcal{C}}(\mathbf{q})\right)^{\mathrm{T}} \begin{bmatrix} \tilde{\mathbf{p}} \\ \mathbf{e}_o \end{bmatrix}, \quad \begin{bmatrix} \dot{\check{\mathbf{p}}} \\ \check{\boldsymbol{\omega}} \end{bmatrix} = \left(\mathbf{T}_{\mathcal{B}}^{\mathcal{C}}(\mathbf{q})\right)^{\mathrm{T}} \begin{bmatrix} \dot{\tilde{\mathbf{p}}} \\ \tilde{\boldsymbol{\omega}} \end{bmatrix}. \tag{38}$$

Furthermore, also the feedforward terms $\ddot{\mathbf{p}}_{\mathcal{B}}^{\mathcal{D}}$ and $\dot{\boldsymbol{\omega}}_{\mathcal{B}}^{\mathcal{D}}$ have to be transformed into the contact frame $\mathcal{C}$ with (36) and $\dot{\mathbf{R}}_{\mathcal{B}}^{\mathcal{C}}$ is neglected in (35) and (38), see [36]. The force control input $\mathbf{v}_f$ is chosen as

$$\mathbf{v}_f = \mathbf{f}_d + \mathbf{K}_{Pf} \left(\mathbf{f}_d - \left(\mathbf{R}_{\mathcal{B}}^{\mathcal{C}}(\mathbf{q})\right)^{\mathrm{T}} \hat{\mathbf{f}} \right) + \mathbf{K}_{If} \int \mathbf{f}_d - \left(\mathbf{R}_{\mathcal{B}}^{\mathcal{C}}(\mathbf{q})\right)^{\mathrm{T}} \hat{\mathbf{f}}\, \mathrm{d}t \tag{39}$$

with the desired force $\mathbf{f}_d^{\mathrm{T}} = \begin{bmatrix} 0 & 0 & f_{d,z} \end{bmatrix}$ and suitable positive definite diagonal gain matrices $\mathbf{K}_{Pf}$ and $\mathbf{K}_{If}$. The contact force $\hat{\mathbf{f}}$ is assumed to appear only in the z-direction of the contact frame of the pen tip $\mathcal{C}$, i.e., $\hat{f}_{\mathcal{C},z}$, cf., (37). Because the selection matrices $\mathbf{Y}_f$ and $\mathbf{Y}_v$ are complementary, the force and motion control loops are decoupled, see [35], [40]. A proof of the asymptotic stability of (34) and (35) is given in [35].

Kinematically redundant robots exhibit an additional null space, which is stabilized using the null-space control law

$$\boldsymbol{\tau}_2 = \mathbf{M}(\mathbf{q})\mathbf{P}\left(-\mathbf{b}(\mathbf{q}) - \mathbf{K}_{dn}\dot{\mathbf{q}} - \mathbf{K}_{pn}(\mathbf{q} - \overline{\mathbf{q}})\right) \tag{40}$$

with the projection matrix $\mathbf{P} = \mathbf{I} - (\mathbf{J}_{\mathcal{B}}^{\mathcal{C}}(\mathbf{q}))^{\dagger}(\mathbf{J}_{\mathcal{B}}^{\mathcal{C}}(\mathbf{q}))$, see [41], and the positive definite diagonal gain matrices $\mathbf{K}_{dn}$ and $\mathbf{K}_{pn}$. In (40), the barrier function $\mathbf{b}(\mathbf{q}) = \begin{bmatrix} b_1(q_1) & b_2(q_2) & \cdots & b_m(q_m) \end{bmatrix}^{\mathrm{T}}$ is defined by

$$b_h(q_h) = \frac{b_{\max}}{(q_h - q_{h,\max})^2} - \frac{b_{\min}}{(q_{h,\min} - q_h)^2}, \quad h = 1, \ldots, m \tag{41}$$

to prevent a robot with m degrees of freedom from reaching joint positions near the mechanical axis limits $\mathbf{q}_{\max}^{\mathrm{T}} = [q_{1,\max} \;\; q_{2,\max} \;\; \cdots \;\; q_{m,\max}]$ and $\mathbf{q}_{\min}^{\mathrm{T}} = [q_{1,\min} \;\; q_{2,\min} \;\; \cdots \;\; q_{m,\min}]$. The parameters $b_{\max}$ and $b_{\min}$ are used to tune the barrier functions and $\overline{\mathbf{q}}$ denotes the middle angle between the axis limits

$$\overline{\mathbf{q}} = \frac{1}{2}\begin{bmatrix} q_{1,\max} + q_{1,\min} & q_{2,\max} + q_{2,\min} & \cdots & q_{m,\max} + q_{m,\min} \end{bmatrix}. \tag{42}$$

The control torque of the null-space control law $\boldsymbol{\tau}_2$ is added to (34), resulting in the final control input $\boldsymbol{\tau} = \boldsymbol{\tau}_1 + \boldsymbol{\tau}_2$.

IV. Experimental Results

In this section, the experimental setup and the results for the robotic drawing process are presented and discussed. First, the parallel path projection and the proposed path projection using a LSCM from Section II are evaluated in simulation with a user-provided 2-D input pattern. Second, the properties of these projection methods are compared. Third, drawing trajectories are generated and executed experimentally on the robot using the motion controller from Section III-A, and fourth, the hybrid force/motion controller from Section III-C is evaluated experimentally. Finally, the drawing results and measurements are compared directly. A video of the experimental drawing process is provided at `www.acin.tuwien.ac.at/c1eb`.

A. Experimental Setup

The experimental setup for the drawing process is shown in Fig. 1. In this setup, the ceiling-mounted industrial robot KUKA LBR iiwa 14 R820 is equipped with a pen tool attached to the end-effector. The tool comprises a passive compliance mechanism to account for absolute positioning errors and model uncertainties of the robot and its environment. The workpiece in the drawing process is a 3-D-printed rabbit model. The optimal robot base placement relative to the 3-D object is computed offline using the optimization-based algorithm proposed in [42]. The actual robot base placement in the experimental setup is determined using a calibration procedure before the experiment can start, see Fig. 7. During this calibration procedure, the robot is equipped with calibration pins at the end-effector (red rectangles in Fig. 7), which tightly fit into holes at the base of the 3-D-printed rabbit (red circles in Fig. 7). The actual robot base placement is obtained from the measurement of the robot configuration $\mathbf{q}$ and the forward kinematics (24), which accurately calibrates the pose of the robot base for subsequent path planning and trajectory execution.

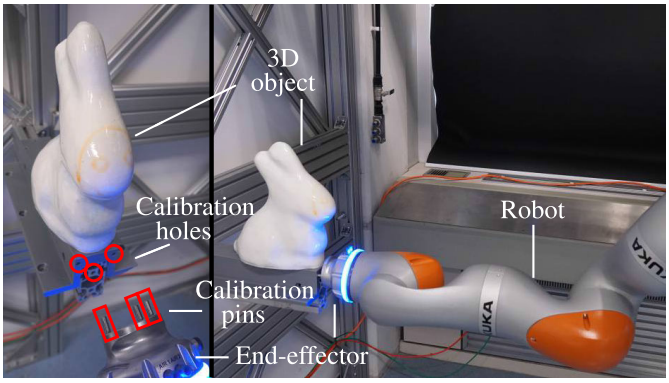


Fig. 7. Calibration method of the experimental setup with the KUKA LBR iiwa 14 R820.

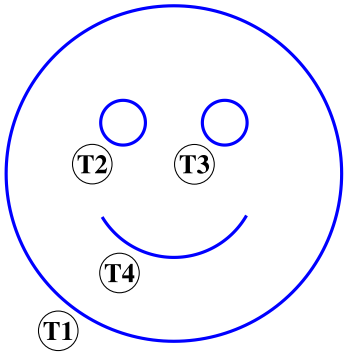


Fig. 8. User-provided 2-D input pattern for the experimental drawing process with the disconnected segments Ⓣ1–Ⓣ4.

The 2-D input pattern is provided by the user either simply using a computer mouse, a touchscreen, or drawing patterns with a digitizer on a tablet device. Furthermore, a path may also be generated from parametric equations. In the following experimental drawing process, the 2-D input pattern with four disconnected segments Ⓣ1–Ⓣ4 shown in Fig. 8 is used, i.e., a smiley symbol.

B. Path Projection

In the following, the user-provided 2-D input pattern in Fig. 8 is projected at a user-specified location on the 3-D object, which is the face of the 3-D-printed rabbit, see Fig. 9. Both projection methods introduced in Section II, i.e., the parallel projection method and the proposed projection method based on the LSCM, are used. Subsequently, the projection results are compared and robot trajectories for the process execution are generated.

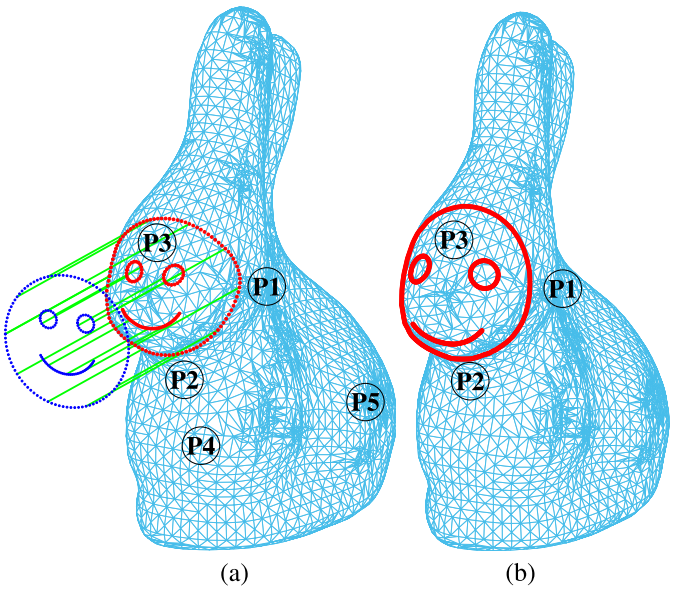


Fig. 9. 2-D input path projected on the 3-D object with (a) parallel projection method and (b) LSCM.

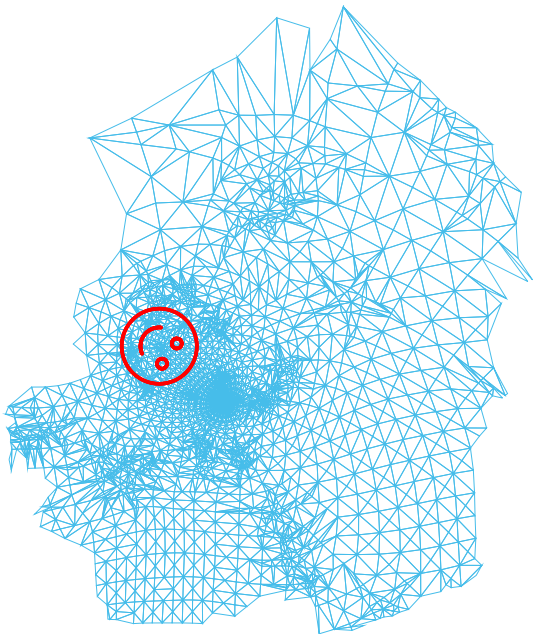


Fig. 10. Projection of the 2-D input pattern on the flattened segments of the 3-D object.

1) Parallel Projection Method: For the simple parallel projection method from Section II-B, parallel green rays are generated originating at each path point $\mathbf{p}_{t,k}, k = 1, \ldots, K$, from the blue 2-D input pattern, see Fig. 9(a). Then, the path points $\mathbf{p}_{o,k}, k = 1, \ldots, K$, on the 3-D object are found at the intersection points of the green rays with the 3-D object's mesh using the implementation of the *ray-triangle intersection* from [43]. At each path point $\mathbf{p}_{o,k}, k = 1, \ldots, K$, a desired end-effector orientation is generated from path planning. Thereby, the z-axis of the end-effector is aligned with the green ray and the x- and y-axes are chosen according to a reference orientation. Note that the orientation is equal for every path point. Finally, transition paths are added and a spline interpolation is computed based on the 3-D path points with a suitable time parametrization, see Section II-C.

Examining the parallel projection result in Fig. 9(a), a large distortion can be seen at point (P1) due to the high curvature of the 3-D object. In comparison, the eyes of the smiley at (P3) are less distorted because of the smaller curvature. Nevertheless, the eyes are in an elliptic shape and not properly placed. If the parallel projection method is used to project a 2-D input pattern to areas with a small curvature, e.g., (P4) or (P5) in Fig. 9(a), the drawing result of the whole 2-D input would be significantly better without notable distortions.

Due to the simplicity of the projection method, the computation is finished after approximately 50 ms on an Intel Core i7-8700K at 3.70 GHz.

2) Path Projection Using LSCM: The path projection on the rabbit according to Section II-A is computed by segmenting the object first, see Fig. 3, and then flattening the individual segments using the conformal mapping. In the next step, the 2-D input pattern is projected on the flattened segments, i.e., the face of the rabbit, see Fig. 10. Note that the two segments located at the face of the rabbit in Fig. 3(b) have to be combined to be able to apply the LSCM for the whole face area. The 2-D input pattern is transferred back to the 3-D object using the barycentric coordinates introduced in Section II-A3. The result of the 2-D input pattern on the 3-D surface of the workpiece is shown in Fig. 9(b). As explained in Section II-C, the robot trajectory is computed by inserting transition points and interpolating the sequence of 3-D path points using a spline with suitable time parametrization. The orientation of the z-direction, i.e., the pen, is chosen based on the normal vector of the surface and the remaining directions are derived from a reference orientation.

Note that the shape of the pattern, in particular the outer circle of the smiley at (P1) and the circular eyes at (P3) in Fig. 9(b), is projected on the 3-D object with minimum distortions.

The path projection using conformal mapping is performed on the 3-D-printed rabbit object, which comprises approximately 13 000 faces and 6500 vertices. The computation is executed on an Intel Core i7-8700K with 3.70 GHz base clock frequency. The most time-consuming computation of this offline planning is the segmentation from Section II-A1 taking around 5 s and the LSCM from Section II-A2 can be calculated in 0.3 s. Note that those steps have to be executed only once for each 3-D object, see Fig. 2.

C. Properties of Projection Methods

In this section, further simulations to compare the parallel projection method with the LSCM are conducted, see Fig. 11. The 2-D input pattern from Fig. 8 is projected on the ear of the 3-D-printed rabbit. Due to the high curvature of the area around the ear, projections are challenging and local distortions can occur.

In the first experiment, the parallel projection method is employed. In this experiment, large distortions due to the high curvature occur, see Fig. 11(a). Note that if the 2-D pattern would be larger than the 3-D object, the green rays

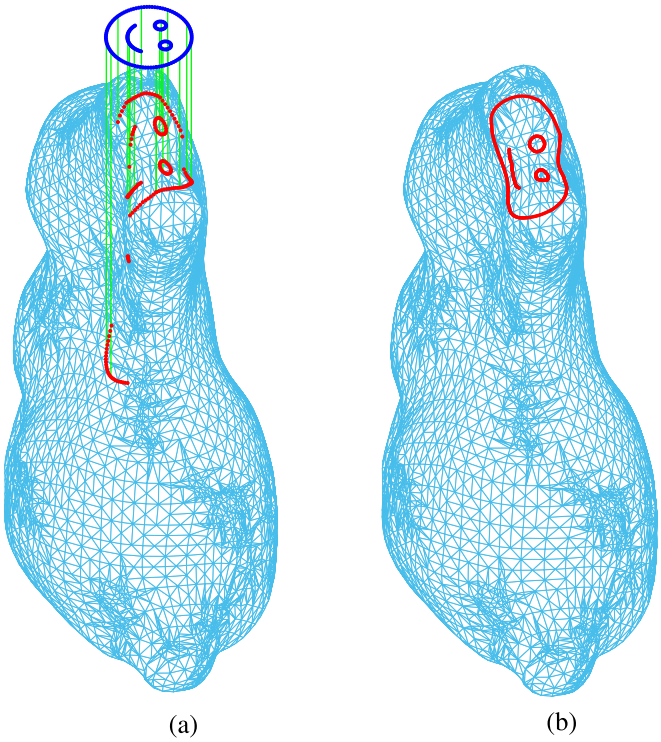


Fig. 11. 2-D input path projected on the ear of the 3-D-printed rabbit with (a) parallel projection method and (b) LSCM.

would not intersect with the 3-D object at all, and therefore, the projection could not be fully computed and the trajectory generation fails.

The second experiment shows the projection of the 2-D input pattern from Fig. 8 on the ear area of the 3-D-printed rabbit with the LSCM, see Fig. 11(b). Using this mapping, the 2-D input pattern is projected visually proper on the 3-D workpiece. Note that due to flattening the segments of the 3-D object into a 2-D form, the 2-D input pattern is wrapped around the ear area at the 3-D object, see also Fig. 10. Using this method, the projection result contains the complete user-provided pattern and the trajectory for the robotic drawing process can be calculated.

D. Drawing Process With Motion Control

In this section, the drawing process using the two presented path projection results from Section IV-B is executed with the pure motion control as introduced in Section III-A. In the following, the planning and measurement results are discussed in terms of the pen motions as 3-D paths, the position control errors, and the joint-space paths.

1) Planned and Executed 3-D Paths: The drawing process is planned with both projection methods and executed on the KUKA LBR iiwa 14 R820 using pure motion control. The resulting 3-D paths are shown in Fig. 12 for the parallel projection and in Fig. 13 for the LSCM projection. Figs. 12 and 13 show the desired trajectory $\mathbf{p}_{\mathcal{B}}^{\mathcal{D}}$ of the contact frame $\mathcal{C}$, i.e., the pen tip, the corresponding pen orientations $\mathbf{n}_{\mathcal{B}}^{\mathcal{D}}$, and the actual trajectory $\mathbf{p}_{\mathcal{B}}^{\mathcal{C}}$, computed using the forward kinematics (24), for the respective projection method. The automatically generated transition paths are also visible.

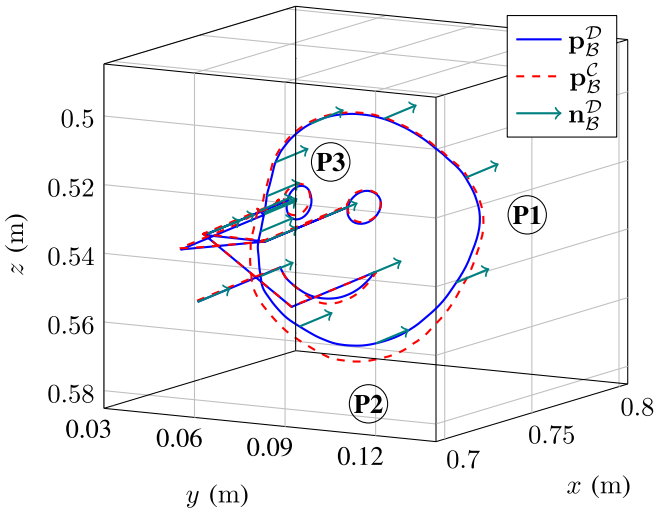


Fig. 12. Desired and executed 3-D path $\mathbf{p}_{\mathcal{B}}^{\mathcal{D}}$ and $\mathbf{p}_{\mathcal{B}}^{\mathcal{C}}$, respectively, and the corresponding surface normal vectors $\mathbf{n}_{\mathcal{B}}^{\mathcal{D}}$ using the parallel projection method with pure motion control.

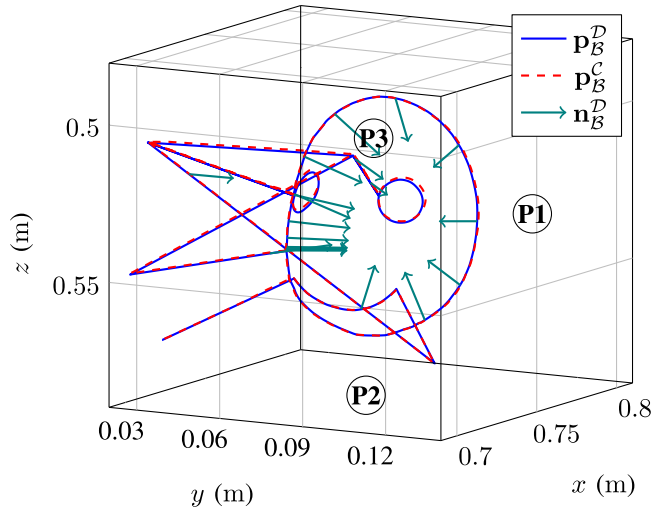


Fig. 13. Desired and executed 3-D path $\mathbf{p}_{\mathcal{B}}^{\mathcal{D}}$ and $\mathbf{p}_{\mathcal{B}}^{\mathcal{C}}$, respectively, and the corresponding surface normal vectors $\mathbf{n}_{\mathcal{B}}^{\mathcal{D}}$ using the LSCM method with pure motion control.

For the parallel projection, the pen orientation $\mathbf{n}_{\mathcal{B}}^{\mathcal{D}}$ is constant and aligned with the projection direction, while it remains aligned with the surface normal vector using the LSCM projection. Consequently, for the parallel projection, a high position deviation between the desired and the actual trajectory becomes clearly visible around (P2). In this area, the local curvature of the 3-D object is very high, see also Fig. 9(a). Due to the constant orientation $\mathbf{n}_{\mathcal{B}}^{\mathcal{D}}$, the angle between the surface normal vector and the pen at the contact point $\mathcal{C}$ becomes large. Due to this large approach angle, the pen slips on the object's surface, which significantly deviates the pen tip from the desired trajectory $\mathbf{p}_{\mathcal{B}}^{\mathcal{D}}$ and results in a large position error. In comparison, the approach angle around the eyes at point (P3) is much smaller, and therefore, this effect is less pronounced, see Fig. 9.

For the LSCM projection, see Fig. 13, the pure motion control exhibits a good closed-loop control performance when

following the desired trajectory in the experimental setup. Even at the points (P1) and (P2) with high curvature, compare Fig. 12, the actual trajectory $\mathbf{p}_{\mathcal{B}}^{\mathcal{C}}$ visually does not deviate from the desired trajectory $\mathbf{p}_{\mathcal{B}}^{\mathcal{D}}$. Because the pen axis $\mathbf{n}_{\mathcal{B}}^{\mathcal{D}}$ is aligned with the surface normal vector, the pen does not slip on the surface due to the perpendicular approach angle. Thus, no position errors appear at areas with high curvature.

2) Position Control Error: In Fig. 14, the position control errors $\breve{\mathbf{p}}$ in the contact frame $\mathcal{C}$ for both experiments are presented, see (38). The topmost graph shows the contact state of the pen, where intervals with the value 1 indicate that the pen is in contact with the 3-D object. The variable l denotes the progress of the path from 0 % to 100 %. During the first interval (T1), where the pen is in contact with the 3-D object, the outer circle of the smiley, at (T2) and (T3) the eyes and during (T4) the mouth is drawn, cf. Fig. 8. In between those intervals, the robot's end-effector moves from the end of the previous pattern to the starting point of the new pattern.

Examining the position control errors $\breve{\mathbf{p}}$ during the intervals (T1) and (T4) for the parallel projection experiment, the position deviation due to the large approach angle of the pen on the object can be clearly seen for $l = 5\ \% - 20\ \%$ and $l = 80\ \% - 90\ \%$ in Fig. 14. Small position errors emerge during the intervals (T2) and (T3) due to the small local curvature and the small approach angle. Note that due to the passive compliance of the pen holder in the z-direction, the position error in this direction is small. In contrast, the position control errors $\breve{\mathbf{p}}$ of the LSCM experiment in Fig. 14 (red lines) remain small during the entire execution time. Quantitatively, the position control error $\breve{\mathbf{p}}$ with respect to the contact frame $\mathcal{C}$ remains below 1 mm for all Cartesian position coordinates.

Note that the lengths of the intervals (T1)–(T4) for the two projection methods differ. This is due to the fact that the range of the axis motion during the drawing process depends on the planned trajectory. Using the LSCM, the pen orientation is always normal to the surface; therefore, the robot has to perform wide joint motions and, consequently, the time intervals are longer. Note that the positive definite diagonal gain matrices $\mathbf{K}_{\omega}, \mathbf{K}_{pn}, \mathbf{K}_{dn}$, and $K_o > 0$ are chosen empirically and the gain matrices $\mathbf{K}_D$, $\mathbf{K}_P$, and $\mathbf{K}_I$ are found by pole placement.

3) Joint-Space Paths: The joint-space paths $\bar{\mathbf{q}}^{\mathrm{T}}(l) = \begin{bmatrix}\bar{q}_1 & \bar{q}_2 & \cdots & \bar{q}_7\end{bmatrix}$ for the process execution with both projections are shown in Fig. 15, where the individual joint angles $\bar{q}_h$ are normalized to their axes limits $q_{h,\min}$ and $q_{h,\max}$ in the form

$$\bar{q}_h = \frac{2q_h - (q_{h,\max} + q_{h,\min})}{q_{h,\max} - q_{h,\min}}, \quad h = 1, \ldots, 7. \tag{43}$$

As the orientation of the pen during the parallel projection experiment remains constant, only small changes in the joint angles are required to follow the corresponding desired path, see the blue lines in Fig. 15. The joint-space path $\bar{\mathbf{q}}(l)$ for the LSCM experiment (red lines in Fig. 15) shows that in this case, significantly larger robot movements are performed. Most of the joints come close to the respective mechanical axis limit at one point during process execution. Therefore, it is necessary to use the advanced null-space control law (40) and (41) to be able to execute this process. Consequently, this shows that

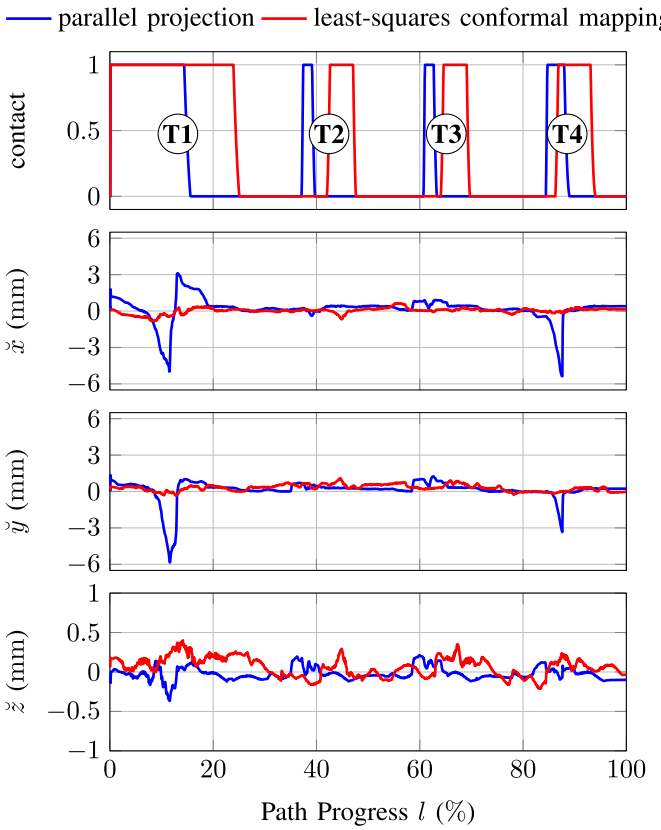


Fig. 14. Evolution of the pen/surface contact (top) and position control errors $\breve{\mathbf{p}}$ using the parallel projection method and the LSCM using pure motion control.

the execution of the planned trajectory is more challenging for the LSCM, but it yields more accurate results for the drawing process, compare Figs. 12 and 13.

E. Drawing Process With Hybrid Force/Motion Control

In this experiment, the hybrid force/motion controller from Section III-C is employed to perform the robotic drawing process based on the planned trajectory with the LSCM. The results are discussed in terms of the planned and executed 3-D paths, the position control error, and the contact force.

1) Planned and Executed 3-D Paths: The desired and actual trajectories of the pen tip $\mathbf{p}_{\mathcal{B}}^{\mathcal{D}}$ and $\mathbf{p}_{\mathcal{B}}^{\mathcal{C}}$, respectively, are shown in Fig. 16. It can be clearly seen that the actual trajectory $\mathbf{p}_{\mathcal{B}}^{\mathcal{C}}$ deviates in the z-direction from the desired trajectory $\mathbf{p}_{\mathcal{B}}^{\mathcal{D}}$ because the force control law (39) is applied in this direction. Therefore, the position in the z-direction deviates from the planned trajectory to guarantee a constant contact force. Note that during the transition phase from the end of one pattern to the starting point of the next pattern, the pure motion controller as in Section IV-D is used. To achieve a smooth transition between the hybrid force/motion control and the pure motion control, slightly smaller diagonal entries of the positive definite gain matrix $\mathbf{K}_{\omega}$ and $K_o > 0$ and smaller diagonal entries of the matrices $\mathbf{K}_D$, $\mathbf{K}_P$, and $\mathbf{K}_I$ compared to the experiment with pure motion control in Section IV-D were chosen. In the force control law (39), the first term, i.e., the feedforward term $\mathbf{f}_d$, is neglected and the term $-\mathbf{K}_{Df}\left(\mathbf{R}_{\mathcal{B}}^{\mathcal{C}}(\mathbf{q})\right)^{\mathrm{T}}\dot{\mathbf{p}}_{\mathcal{B}}^{\mathcal{C}}(\mathbf{q})$ is added with the positive definite diagonal matrix $\mathbf{K}_{Df}$ to damp

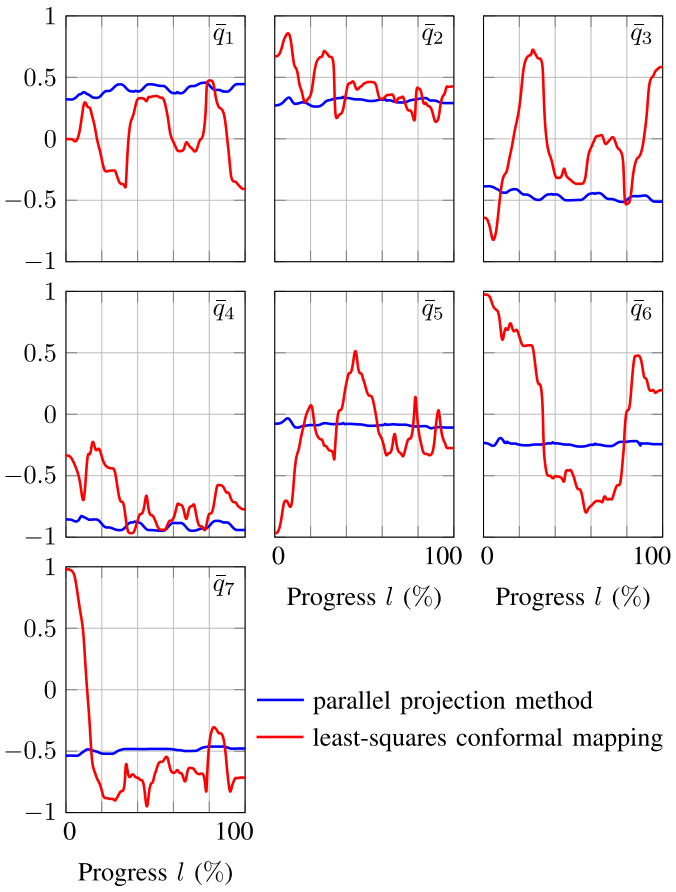


Fig. 15. Joint-space path $\bar{\mathbf{q}}(l)$ using the parallel projection method and the LSCM with pure motion control. The paths of the individual joints are normalized to their respective axes limits $\mathbf{q}_{\min}$ and $\mathbf{q}_{\max}$.

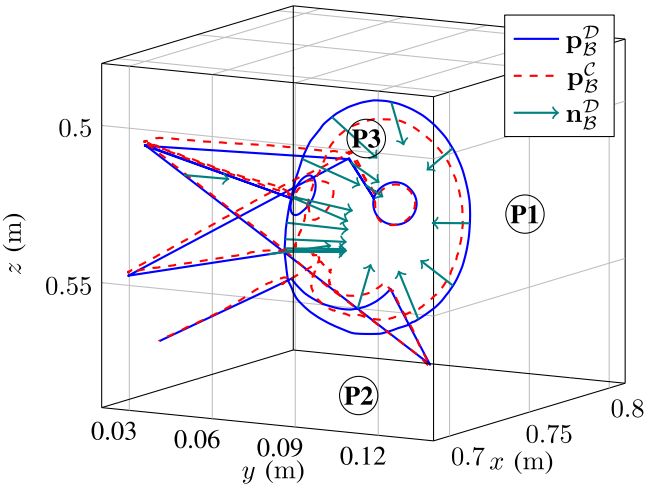


Fig. 16. Desired and executed 3-D path $\mathbf{p}_{\mathcal{B}}^{\mathcal{D}}$ and $\mathbf{p}_{\mathcal{B}}^{\mathcal{C}}$, respectively, and the corresponding surface normal vectors $\mathbf{n}_{\mathcal{B}}^{\mathcal{D}}$ using the LSCM method with hybrid force/motion control.

the motion along the z-direction of the contact frame $\mathcal{C}$ and generate smooth robot motions during the experiment. Note that the direct force feedback from the last term of (34) is omitted due to noise in the contact force estimation $\hat{\mathbf{f}}$ and a disturbance observer is added for friction compensation in the robot joints.

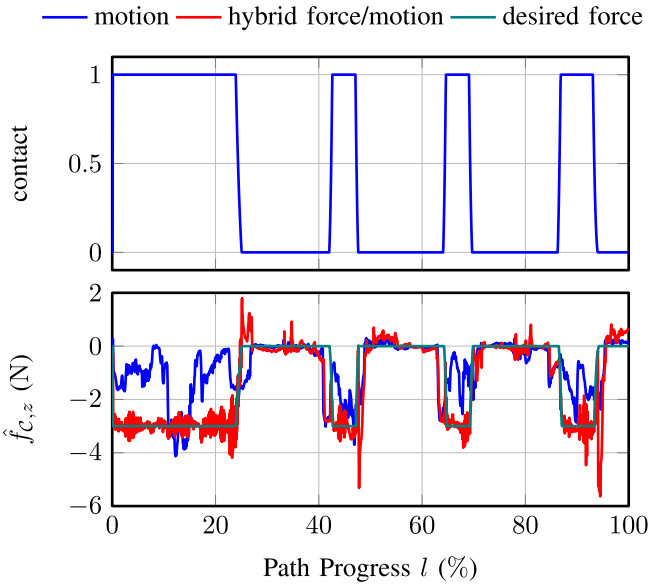


Fig. 17. Evolution of the pen/surface contact state (top) and the estimated contact forces $\hat{f}_{\mathcal{C},z}$ in the z-direction (bottom) during the drawing process with the pure motion control and the hybrid force/motion control.

2) Contact Force: The estimated contact forces $\hat{\mathbf{f}}$ are shown in Fig. 17 for the process execution of the LSCM trajectory with the hybrid force/motion control and the pure motion control of Section IV-D. Clearly, the contact force parallel to the surface normal $\hat{f}_{\mathcal{C},z}$ is controlled to the desired value of -3 N by the hybrid force/motion controller (red line). At every change of the pen/surface contact state, the controller is switched from the hybrid force/motion controller to the pure motion controller and vice versa. The small contact force peaks in $\hat{f}_{\mathcal{C},z}$, see Fig. 17, originate from these controller switching operations. In contrast, the contact force $\hat{f}_{\mathcal{C},z}$ is not controlled by the pure motion controller, and hence, it varies between -4 N and nearly zero for this experiment (blue line). In addition, a loss of the pen/surface contact can occur due to misalignment of the workpiece and/or inaccuracies of the robot. For both controllers, the estimated forces are approximately zero in phases without contact. The stick-slip effect is observed between the pen tip and the surface if the normal contact force becomes too high. Other friction effects are not perceivable in the estimation because the pen is controlled to be always normal to the surface, and only the normal contact force is estimated.

Remark 1: Note that the contact force estimation [34] requires a precisely calibrated dynamic robot model (23). Otherwise, significant estimation errors may occur, even if no pen/surface contact is present. Alternatively, the contact force estimation can be calibrated for a specific robot trajectory by performing the experiment without any pen/surface contact as a reference motion with $\hat{\mathbf{f}} = \mathbf{0}$.

3) Position Control Error: In order to evaluate the performance of the hybrid force/motion control, the position control error $\breve{\mathbf{p}}$ of the drawing process is shown in Fig. 18. Overall, small control errors are observed for $\breve{x}$ and $\breve{y}$, whereas the small peaks originate from the controller switching, compare Fig. 17. During the intervals with pen/surface contact, the

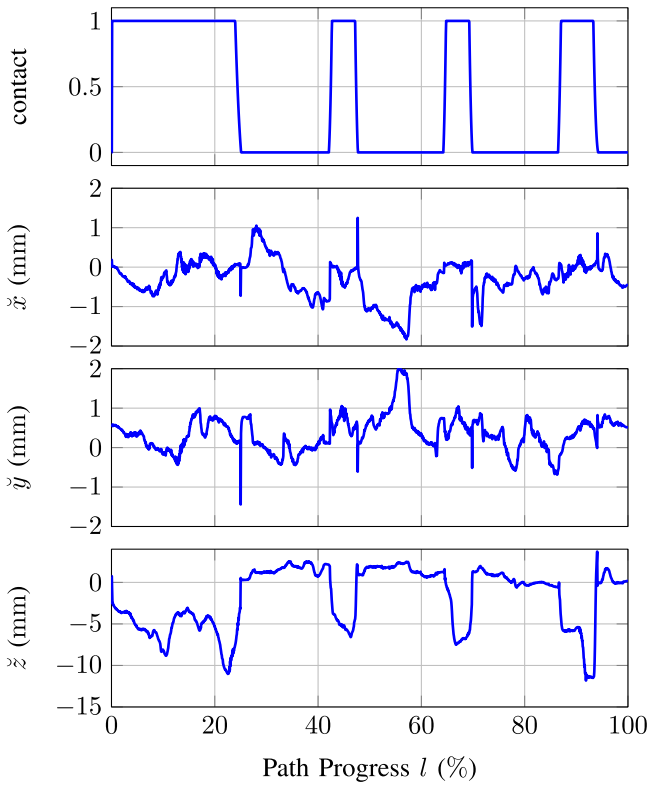


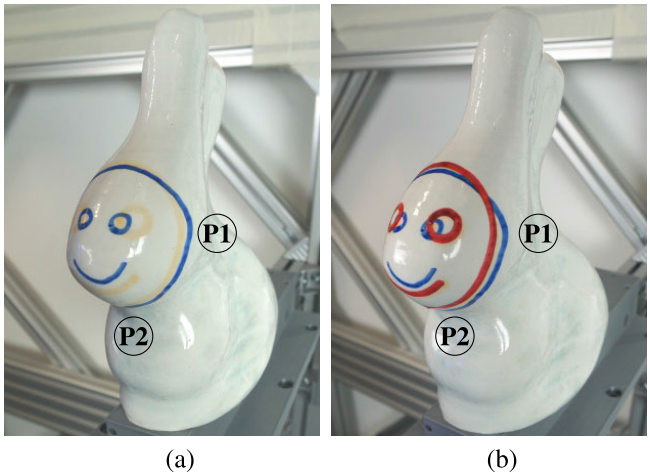
Fig. 18. Evolution of the pen/surface contact (top) and position control error $\check{\mathbf{p}}$ using the LSCM with the hybrid force/motion control.

position control error $\check{\mathbf{p}}$ in z-direction is higher because the force controller adapts the position in the z-direction to control the contact force $\hat{f}_{\mathcal{C},z}$.

F. Comparison of the Drawing Results

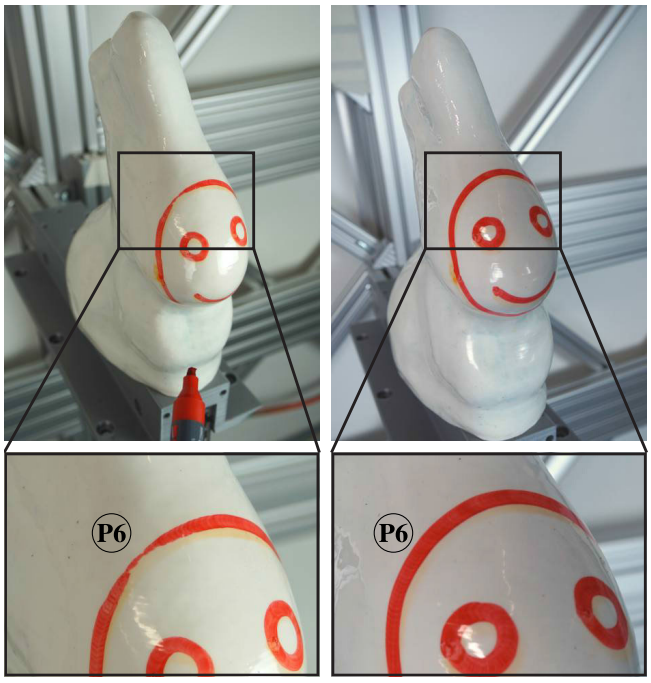
In this section, the experimental drawing results of both projection methods introduced in Section II and the two control concepts from Section III are shown and compared. The resulting drawing pattern on the 3-D-printed rabbit using the parallel projection method with pure motion control is shown in Fig. 19(a). For comparison, the LSCM pattern is drawn on top of the drawing pattern for the parallel projection. Comparing the two projection methods, distortions from the parallel projection method are seen, especially at areas with high curvature, e.g., at point (P1). The robot is not able to accurately follow the desired trajectory at areas with a large approach angle using the parallel projection method with pure motion control, e.g., (P2), compare Figs. 12 and 13.

Next, the drawing results with the two presented control concepts are compared for the LSCM trajectory, see Fig. 20. In the drawing result using the pure motion controller in Fig. 20(a), a fluctuating line thickness is observed. In particular, at point (P6), nearly no pen/surface contact is present, emerging from inaccuracies of the robot kinematics. This thin line results from a low contact force of the pen tip, which can also be seen in the estimated contact force $\hat{f}_{\mathcal{C},z}$ in Fig. 17 at $l \approx 10\%$. A uniform line thickness is achieved with the hybrid force/motion control in Fig. 20(b). Note that the experiments for this drawing process are executed without an absolute calibration of the robot and without optical measurement of the actual Cartesian end-effector pose.



(a) (b)

Fig. 19. Resulting patterns on the 3-D object with pure motion control (a) using the parallel projection and (b) together with the LSCM.



(a) (b)

Fig. 20. Resulting patterns on the 3-D object with the LSCM using (a) motion control and (b) hybrid force/motion control.

V. Conclusion

In this work, an automated workflow for full customization of products using robotic manufacturing is presented on the basis of a drawing process on a complex surface of a 3-D object. In this process, a 2-D input pattern is provided by a user together with the desired size, location, and rotation on the 3-D object and then drawn on the 3-D object using an industrial robot.

The workflow starts with a mapping procedure of the 2-D input pattern to the 3-D object, for which two different

mapping methods are presented and explained, i.e., a conformal mapping and a simple parallel projection approach. The conformal mapping procedure comprises a segmentation step, a LSCM to flatten the segments, and an inverse map using barycentric coordinates. In this way, distortions of the 2-D input pattern are minimized and a 3-D path on the 3-D object is obtained. Although the parallel projection method can be used in areas with small curvature and low complexity, only with the more advanced conformal mapping procedure, a visually proper projection is achieved. Based on the result of the two presented projection methods, robot trajectories are planned to be executed in an experimental setup with an industrial robot. For the planned trajectories, two different control concepts, i.e., pure motion control and hybrid force/motion control, are presented and employed in the experiments. In addition, the contact force during the drawing process is estimated. The pure motion controller is able to execute the trajectory with small errors, however, only the hybrid force/motion controller is able to maintain the desired contact force normal to the surface during the whole task execution, which is necessary for achieving a high production quality. This is demonstrated in this work by visually comparing the drawing process results using both mapping procedures and the two control concepts.

In industry, this approach can be used to automatically map 2-D manufacturing paths to different 3-D objects while maintaining the required accuracy in position and contact force during the whole process. This automated workflow is directly applicable to many other manufacturing processes such as automated laser engraving, milling, or ultrasonic cutting.

REFERENCES

[1] International Federation of Robotics. (2020). *World Robotics 2020 Report*. Accessed: Nov. 23, 2022. [Online]. Available: http://reparti.free.fr/robotics2000.pdf

[2] M. T. Fralix, "From mass production to mass customization," *J. Textile Apparel, Technol. Manage.*, vol. 1, no. 2, pp. 1–7, 2001.

[3] G. Ye and R. Alterovitz, *Robotics Research: Demonstration-Guided Motion Planning* (Springer Tracts in Advanced Robotics), vol. 100. Cham: Springer, 2017, pp. 291–307.

[4] D. Ding et al., "Towards an automated robotic arc-welding-based additive manufacturing system from CAD to finished part," *Comput.-Aided Des.*, vol. 73, pp. 66–75, Apr. 2016.

[5] M. Daneshmand et al., "3D scanning: A comprehensive survey," 2018, *arXiv:1801.08863*.

[6] S. Calinon, J. Epiney, and A. Billard, "A humanoid robot drawing human portraits," in *Proc. IEEE-RAS Int. Conf. Humanoid Robots*, Dec. 2005, pp. 161–166.

[7] C.-Y. Lin, L.-W. Chuang, and T. Thoa Mac, "Human portrait generation system for robot arm drawing," in *Proc. IEEE/ASME Int. Conf. Adv. Intell. Mechatronics*, Jul. 2009, pp. 1757–1762.

[8] G. Jean-Pierre and Z. Saíd, "The artist robot: A robot drawing like a human artist," in *Proc. IEEE Int. Conf. Ind. Technol.*, Mar. 2012, pp. 486–491.

[9] P. Tresset and F. F. Leymarie, "Portrait drawing by Paul the robot," *Comput. Graph.*, vol. 37, no. 5, pp. 348–363, 2013.

[10] X. Huang, S. Bi, M. Dong, H. Chen, S. Fang, and N. Xi, "Automatic feature extraction and optimal path planning for robotic drawing," in *Proc. IEEE Int. Conf. Cyber Technol. Autom., Control, Intell. Syst.*, Jun. 2016, pp. 19–24.

[11] T. Xue and Y. Liu, "Robot portrait rendering based on multi-features fusion method inspired by human painting," in *Proc. IEEE Int. Conf. Robot. Biomimetics*, Dec. 2017, pp. 2413–2418.

[12] M. Pichkalev, R. Lavrenov, R. Safin, and K.-H. Hsia, "Face drawing by KUKA 6 axis robot manipulator," in *Proc. 12th Int. Conf. Develop. eSyst. Eng.*, 2019, pp. 709–714.

[13] D. Song, T. Lee, and Y. J. Kim, "Artistic pen drawing on an arbitrary surface using an impedance-controlled robot," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2018, pp. 4085–4090.

[14] S. Jain, P. Gupta, V. Kumar, and K. Sharma, "A force-controlled portrait drawing robot," in *Proc. IEEE Int. Conf. Ind. Technol.*, Mar. 2015, pp. 3160–3165.

[15] W. Sheng, H. Chen, N. Xi, and Y. Chen, "Tool path planning for compound surfaces in spray forming processes," *IEEE Trans. Autom. Sci. Eng.*, vol. 2, no. 3, pp. 240–249, Jul. 2005.

[16] A. Kharidege, D. T. Ting, and Z. Yajun, "A practical approach for automated polishing system of free-form surface path generation based on industrial arm robot," *Int. J. Adv. Manuf. Technol.*, vol. 93, nos. 9–12, pp. 3921–3934, Dec. 2017.

[17] R. K. Malhan et al., "Automated planning for robotic layup of composite prepreg," *Robot. Comput.-Integr. Manuf.*, vol. 67, 2021, Art. no. 102020.

[18] Wandelbots. *TracePen*. Accessed: Nov. 23, 2022. [Online]. Available: https://wandelbots.com/

[19] M. Schwegel, H. Augschöll, C. Hartl-Nesic, and A. Kugi, "Teach-in for force sensitive automatic rope winding in three dimensions," in *Digital-Fachtagung VDI Mechatronik*, T. Bertram, B. Corves, K. Janschek, S. Rinderknecht, Eds. Darmstadt: Universitäts- und Landesbibliothek, 2021, pp. 193–198. [Online]. Available: https://publications.rwth-aachen.de/record/819234

[20] D. Ehmann and C. Wittenberg, "The idea of virtual teach-in in the field of industrial robotics," in *Proc. IEEE 14th Int. Conf. Control Autom.*, Jun. 2018, pp. 680–685.

[21] C. Wögerer, M. Mühlberger, M. Ikeda, J. Kastner, N. C. Chitturi, and A. Pichler, "Inkjet Printings on FFF printed curved surfaces," in *Proc. Fraunhofer Direct Digit. Manuf. Conf.*, 2018, pp. 1–4.

[22] ABB. *PixelPaint*. Accessed: Nov. 23, 2022. [Online]. Available: https://new.abb.com/products/robotics/de/funktionspakete/pixelpaint

[23] P. Urhal, A. Weightman, C. Diver, and P. Bartolo, "Robot assisted additive manufacturing: A review," *Robot. Comput.-Integr. Manuf.*, vol. 59, pp. 335–345, 2019.

[24] T. Weingartshofer, A. Haddadi, C. Hartl-Nesic, and A. Kugi, "Flexible robotic drawing on 3D objects with an industrial robot," in *Proc. IEEE Conf. Control Technol. Appl.*, Aug. 2022, pp. 29–36.

[25] B. Lévy, S. Petitjean, N. Ray, and J. Maillot, "Least squares conformal maps for automatic texture atlas generation," *ACM Trans. Graph.*, vol. 21, no. 3, pp. 362–371, 2002.

[26] J. Xu, X. Zhang, S. Wang, and J. Wu, "Tool path generation for pattern sculpting on free-form surfaces," *Int. J. Adv. Manuf. Technol.*, vol. 67, nos. 9–12, pp. 2469–2476, 2013.

[27] A. Hubeli and M. Gross, "Multiresolution feature extraction for unstructured meshes," in *Proc. Visualizat.*, 2001, pp. 287–294.

[28] D. Alpay, *A Complex Analysis Problem Book*. Basel, Switzerland: Birkhäuser, 2016.

[29] M. Botsch, L. Kobbelt, M. Pauly, P. Alliez, and B. Levy, *Polygon Mesh Processing*. Natick, MA, USA: A K Peters, 2010.

[30] A. A. Ungar, *Barycentric Calculus in Euclidean and Hyperbolic Geometry: A Comparative Introduction*. Singapore: World Scientific, 2010.

[31] P. Shirley, M. Ashikhmin, and S. Marschner, *Fundamentals of Computer Graphics*, 3rd ed. Boca Raton, FL, USA: Taylor & Francis, 2009.

[32] T. Möller and B. Trumbore, "Fast, minimum storage ray-triangle intersection," *J. Graph. Tools*, vol. 2, no. 1, pp. 21–28, 1997.

[33] E. T. Y. Lee, "Choosing nodes in parametric curve interpolation," *Computer-Aided Design*, vol. 21, no. 6, pp. 363–370, 1989.

[34] E. Magrini, F. Flacco, and A. De Luca, "Estimation of contact forces using a virtual force sensor," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Sep. 2014, pp. 2126–2133.

[35] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics Modelling, Planning and Control*. London, U.K.: Springer, 2009.

[36] E. Magrini and A. De Luca, "Hybrid force/velocity control for physical human-robot collaboration tasks," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2016, pp. 857–863.

[37] F. Leali, M. Pellicciari, F. Pini, A. Vergnano, and G. Berselli, "A calibration method for the integrated design of finishing robotic workcells in the aerospace industry," in *Proc. Int. Workshop Robot. Smart Manuf.*, 2013, pp. 37–48.

[38] J. S. Yuan, "Closed-loop manipulator control using quaternion feedback," *IEEE J. Robot. Autom.*, vol. 4, no. 4, pp. 434–440, Aug. 1988.

[39] T. Weingartshofer, M. Schwegel, C. Hartl-Nesic, T. Glück, and A. Kugi, "Collaborative synchronization of a 7-axis robot," *IFAC-PapersOnLine*, vol. 52, no. 15, pp. 507–512, 2019.

[40] B. Siciliano and O. Khatib, Eds., *Springer Handbook of Robotics*. Berlin, Germany: Springer, 2008.

[41] C. Ott, *Cartesian Impedance Control Redundant Flexible-Joint Robots* (Springer Tracts in Advanced Robotics), vol. 49. Berlin, Germany: Springer, 2008.

[42] T. Weingartshofer, C. Hartl-Nesic, and A. Kugi, "Optimal TCP and robot base placement for a set of complex continuous paths," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2021, pp. 9659–9665.

[43] J. Tuszynski. (2018). *Triangle/Ray Intersection*. MATLAB Central File Exchange. Accessed: Nov. 23, 2022. [Online]. Available: https://www.mathworks.com/matlabcentral/fileexchange/33073-triangle-ray-intersection

Thomas Weingartshofer received the Dipl.-Ing. degree in electrical engineering from TU Wien, Vienna, Austria, in 2018.

He is currently working as a Ph.D. Researcher at the Automation and Control Institute (ACIN), TU Wien. His main research interests include constraint motion and path planning algorithms, automated and user-friendly robot programming, optimal robot base, and tool center point (TCP) placement and control concepts for industrial robots.

Christian Hartl-Nesic (Member, IEEE) received the Dipl.-Ing. and Ph.D. (Dr.techn.) degrees in electrical engineering from TU Wien, Vienna, Austria, in 2016 and 2020, respectively.

He is currently working as a Post-Doctoral Researcher at the Automation and Control Institute (ACIN), TU Wien. His main research interests include novel control, teach-in, and usage concepts for robotics in a wide range of industrial applications, as well as human–robot interaction.

Andreas Kugi (Senior Member, IEEE) received the Dipl.-Ing. degree in electrical engineering from TU Graz, Graz, Austria, in 1992, and the Ph.D. degree in control engineering and the Habilitation degree in automatic control and control theory from Johannes Kepler University (JKU), Linz, Austria, in 1995 and 2000, respectively.

He was an Associate Professor with JKU from 2000 to 2002 and a Full Professor with Saarland University, Saarbrücken, Germany, from 2002 to 2007. Since 2007, he has been a Full Professor of complex dynamical systems at the Automation and Control Institute (ACIN), TU Wien, Vienna, Austria. From 2017 to 2023, he was the Head of the Center for Vision, Automation and Control, AIT Austrian Institute of Technology GmbH, Vienna. Since July 2023, he has been the Scientific Director of the AIT Austrian Institute of Technology GmbH. His main research interests include modeling, control, and optimization of complex dynamical systems; mechatronic system design; and robotics and process automation.

Dr. Kugi is a full member of the Austrian Academy of Sciences and a member of the German National Academy of Science and Engineering (acatech).