Generating Optimized Trajectories for Robotic Spray Painting

Daniel Gleeson, Stefan Jakobsson, Raad Salman, Fredrik Ekstedt, Niklas Sandgren, Fredrik Edelvik, Johan S. Carlson, and Bengt Lennartson, *Fellow, IEEE*

***Abstract*— In the manufacturing industry, spray painting is often an important part of the manufacturing process. Especially in the automotive industry, the perceived quality of the final product is closely linked to the exactness and smoothness of the painting process. For complex products or low batch size production, manual spray painting is often used. But in large scale production with a high degree of automation, the painting is usually performed by industrial robots. There is a need to improve and simplify the generation of robot trajectories used in industrial paint booths. A novel method for spray paint optimization is presented, which can be used to smooth out a generated initial trajectory and minimize paint thickness deviations from a target thickness. The smoothed out trajectory is found by solving, using an interior point solver, a continuous non-linear optimization problem. A two-dimensional reference function of the applied paint thickness is selected by fitting a spline function to experimental data. This applicator footprint profile is then projected to the geometry and used as a paint deposition model. After generating an initial trajectory, the position and duration of each trajectory segment are used as optimization variables. The primary goal of the optimization is to obtain a paint applicator trajectory, which would closely match a target paint thickness when executed. The algorithm has been shown to produce satisfactory results on both a simple 2-dimensional test example, and a non-trivial industrial case of painting a tractor fender. The resulting trajectory is also proven feasible to be executed by an industrial robot.**

***Note to Practitioners*—The work is motivated by the need to generate well performing robot trajectories in robotized spray-painting booths. The described method applies to cases where robotic spray painting is to be used for painting a surface with an even layer of paint at a specified thickness. The method generates and optimizes robot trajectories and is shown to be able to generate a satisfactory paint cover for simple test cases as well as more realistic industrial cases. For a user it could be implemented as is, or be obtained as a standalone service, but there are some prerequisites that need to be fulfilled to make use of the optimization method. It is assumed that the surface to be painted is available as a CAD-model and that physical testing has been performed to determine the characteristics of the paint and nozzle. These physical tests amount to spraying paint on a flat piece of material at a few different distances from the surface and measuring the cross section of the paint thickness. The resulting trajectory can be executed on any industrial painting robot that can handle linear motion commands.**

***Index Terms*— Industrial robots, manufacturing automation, robot motion, spray painting, trajectory optimization.**

Manuscript received 8 October 2021; revised 7 January 2022; accepted 15 February 2022. Date of publication 16 March 2022; date of current version 5 July 2022. This article was recommended for publication by Associate Editor W. Shen and Editor M. Dotoli upon evaluation of the reviewers' comments. This work was supported in part by the Project SelfPaint funded by the Fraunhofer Gesellshaft and the Internal Program for Business Oriented Strategic Alliances [Wissenschaftsorientierte Strategische Allianzen (WISA)], in part by the Formas—a Swedish Research Council for Sustainable Development and the Project Roboclean (2019-02264), and in part by the Project Sustainable Motions—SmoothIT funded by the Swedish Governmental Agency for Innovation Systems (Vinnova). *(Corresponding author: Daniel Gleeson.)*

Daniel Gleeson is with the Geometry and Motion Planning Department, Fraunhofer-Chalmers Centre, 412 88 Gothenburg, Sweden, and also with the Automation Research Group, Department of Electrical Engineering, Chalmers University of Technology, 412 96 Gothenburg, Sweden (e-mail: daniel.gleeson@fcc.chalmers.se).

Stefan Jakobsson is with Arcam EBM, GE Additive Company, 435 33 Mölnlycke, Sweden.

Raad Salman, Fredrik Ekstedt, and Johan S. Carlson are with the Geometry and Motion Planning Department, Fraunhofer-Chalmers Centre, 412 88 Gothenburg, Sweden.

Niklas Sandgren and Fredrik Edelvik are with the Computational Engineering and Design Department, Fraunhofer-Chalmers Centre, 412 88 Gothenburg, Sweden.

Bengt Lennartson is with the Automation Research Group, Department of Electrical Engineering, Chalmers University of Technology, 412 96 Gothenburg, Sweden.

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TASE.2022.3156803.

Digital Object Identifier 10.1109/TASE.2022.3156803

I. INTRODUCTION

SPRAY painting has been used for more than a century as an efficient way to apply an even layer of paint to details and parts of all sizes. Initially this was a purely manual process, using airbrushes or other handheld devices. Manual spray painting is still used, even in industries which are otherwise highly automated. However, a significant part of industrial spray painting is now automated. The degree of automation is bound to increase, since it is one of the key drivers for increasing quality and lowering costs [1]. Early automation developments included automated linear spray systems, which have advanced to the fully automated and robotized spray-painting booths used in many industries today.

Robotic spray painting is a commonly used painting technique in highly automated production processes, such as in the automotive industry. Often a manual teaching pendant is used to time-consumingly generate robot trajectories. By simulating the process offline, it is possible to create tools that simplify the work for a spray-painting engineer. Accurately simulating the resulting paint thickness for a given robot trajectory and providing this as feedback to the operator, makes it possible to modify and improve robotic trajectories before, or ideally instead of, physical testing.

After an accurate forward simulation of a process is achieved, the focus is generally shifted towards solving the inverse problem. In this case, where the forward simulation finds a resulting paint thickness for a given robotic painting trajectory, the inverse problem becomes finding a painting trajectory given a desired paint thickness. More precisely, the problem could be stated as follows: given a desired paint thickness and a triangulated model of a fixed geometry to be painted, find a painting trajectory that when executed produces a paint thickness that as closely as possible matches the target thickness. The optimization problem was described in [2] where an initial trajectory was assumed to be previously generated. This paper aims to provide a more complete description of the process of going from a specified surface to be painted, to executable robot trajectories. To achieve this, the initial trajectories, which are an important starting point of the optimization, need to be generated with a focus on the feasibility and structure of the trajectory. The initial curve generation is done by covering a rectangular area with a sweeping curve using a specified sweep width, projecting this curve onto the specified surface, and finally modifying the curve with extra waypoints and extended sweeps.

Apart from the paint thickness objective, there are some additional requirements placed on the final trajectory. These mainly relate to executability. It is also important that the generated trajectory lies within the set of parameter combinations, where the chosen projection-based paint deposition model gives accurate results. This is an accuracy problem, which is not easy to specify or include in the optimization problem, since there are no strict borders of the feasible region. Some parameters behave quite regularly and predictably. For example, the linear scaling of the projection profile with increasing perpendicular distance to the target has been experimentally verified. For other parameters the accuracy degrades rapidly, for example when approximating the paint deposition on parts of the geometry with very high curvature. These problems are handled in the cost function by penalising several deviation measures from the feasible initial trajectory.

Related problems within the area of robotics that have been extensively studied include trajectory generation for point-like features, such as spot welding [3]. It also includes curve following applications, such as lay down of sealant material, where the curve is already defined by the seam between the sheet metal parts [4]. The added complexity of features with higher degrees of freedom and higher model uncertainty makes the optimization of paint applicator trajectories a comparatively harder problem to solve. The overall goal is to formulate a paint thickness optimization problem that accurately captures the behaviour of the physical system.

The main contributions of the work presented in this paper can be summarized as follows:

- An algorithm for generating initial spray painting trajectories, that cover the surface with sufficient paint, while having a simple topology with few or no intersections and twists.
- Incorporating an accurate paint deposition model in the trajectory optimization based on experimental data.
- Generating solution trajectories that are feasible to directly execute on industrial painting robots.

Just as it is important for the optimization problem to accurately capture the physical system, it also has to be solvable within reasonable execution times. This is something that will be application dependent. The trajectory optimization has initially been used in an offline stage, meant to be used in a setup stage of an industrial painting line. In an offline stage, short execution times are not critical. Instead, computationally expensive droplet tracing algorithms can be used in a post processing step to verify or improve the solution trajectories. Using the optimization problem formulation presented in this paper, solutions are found in a matter of minutes even for reasonably complex industrial cases. This opens up the possibility of using the solution trajectories in low batch size painting booths, where incoming objects are scanned in connection with their arrival at the painting booth.

The proposed iterative optimization procedure is shown to produce an even paint thickness for typical non-trivial industrial use cases of robotic paint coating, and the proposed paint deposition model has been tuned to closely match physical experiments of rotary bell applicators. Additionally, later stages of the development have been performed in parallel with physical testing in order to be able to handle practical considerations for typical cases. The optimization problem is specialized for spray painting applications, but the proposed problem formulation is quite general and could be applied in similar situations with a defined goal function over a continuous surface. Similar applications might include automated cleaning of non-trivial surfaces or refining a laser measurement mapping of a complex geometry.

Some background and motivation for the work as well as a brief summary of previous work and contributions can be found in Section II. A description of the overall problem to be solved is given in Section III, along with an overview of how the work presented here fits into a modularized workflow to find a solution. The method for generating initial painting trajectories is described in Section IV, while the paint optimization problem is defined in Section V. Plots of the resulting paint thickness when running the optimization on a simple test-geometry, as well as an industrial case of a tractor fender to be painted, are presented in Section VI. A discussion about the method, model and the results can be found in Section VII. Finally, some conclusions and recommendations for further research are given in Section VIII.

II. Background

Paint deposition modelling is a field which has received attention from both industry and academia, and multiple different models have been developed and are used in different settings. These models include simple cover checking, projection methods using for example a composite Gaussian formulation [5], or even full dynamic simulation of electrically charged paint droplets dispersed from a rotary bell nozzle [6]–[9].

In the general case, to accurately simulate the paint thickness produced by a given paint applicator trajectory, a set of coupled non-linear differential equations needs to be solved. Such a computationally expensive simulation is infeasible to be used directly in an optimization routine. But it is possible to simplify the paint deposition problem, and we are satisfied to find a reasonable approximation to the resulting paint thickness. This is done by fitting parameters to experimentally found thickness measures of simple test trajectories. The simplified deposition model is based on projecting a footprint profile onto a target geometry. This experimentally verified model is reasonably accurate in a region of parameter values, as long as they do not deviate too much from the different sets of parameters used to tune the model. The initial trajectory is assumed to be in this feasible region. During the optimization the deviations from the initial trajectory is limited to ensure that the final solution is within a region of acceptable model accuracy. The main parts of the paint projection method is described in Section V, and a thorough explanation can be found in [10]. The necessary calculations can be divided into a paint flow calculation and a paint projection scaling.

Optimizing or creating a painting trajectory, given a representation of a geometry, has been a research topic for many years. Examples of contributions in the area from the early '90s include both trajectory planning for uniform circular paint deposition on early CAD-models [11], and more analytical treatment of the underlying functions in the optimization problem [12]. Later work has expanded this to more general surfaces, while still focusing on finding an optimal overlap for adjacent sweeps. A common method, where intersecting planes parallel to the bounding box of the surface generate the painting trajectory, is outlined in [13]. The same technique is used in [14], but an additional post-processing step is introduced in order to achieve uniform overlap between adjacent passes on surfaces with heavy curvature. In [15] the proposed method produces a painting trajectory on a triangulated 2D surface by computing curves with minimized geodesic curvature, and a comparison to the method of intersecting planes is performed. An integer programming approach is described in [16], while [17] outlines a constrained multi-objective optimization problem for achieving uniform thickness while minimizing both cycle time and material waste.

More recent work on optimizing trajectories for thermal spraying [10] has directly influenced the formulation of the trajectory optimization problem presented here. However, in this paper a more accurate paint projection profile is directly included into the optimization problem, with the parameters of the deposition model tuned to physical experiments. The objective function is also extended to model more precise behaviour of the physical robot system, which will execute the solution trajectory.

III. Problem Description

The painting trajectory optimization approach presented in this paper has been developed and refined during a project called SelfPaint, where the goal is to automate the full process of spray-painting parts in a painting booth [18]. The overall problem is to handle an incoming geometry to be painted, with the goal of automatically finding a robot trajectory that can be executed in the painting booth. The resulting paint coverage

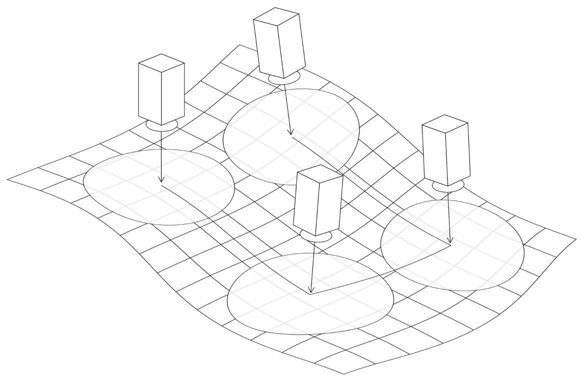


Fig. 1. Sketch of a paint applicator trajectory showing the applicator at four different waypoints. At each of these waypoints a simplified visual representation of the applicator is shown, as well as the area of the projected paint thickness profile onto the static geometry to be painted.

should closely match the target thickness and always be within specification.

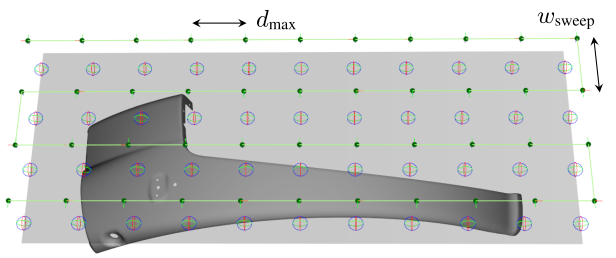
The trajectory optimization has been included as one of the modules of the self-programming painting cell. The full SelfPaint workflow [19] also includes earlier modules such as point cloud scanning with CAD model fitting, and later modules that include physics-based simulations as well as physical testing and verification.

The following two sections describe the initial curve generation and the optimization problem definition, but first some notations and concepts that will be used later are briefly introduced. The applicator trajectory that is generated and then optimized is assumed to be piecewise linear. The sketch of a paint applicator trajectory seen in Fig. 1, shows how a typical robotic paint trajectory is defined. The trajectory consists of trajectory segments, each one starting and ending in a waypoint with specified values for applicator position and orientation. It is these waypoints along with the segment durations that are modified in order to obtain a satisfactory paint coating.

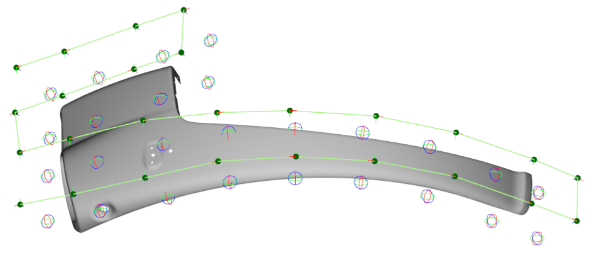
IV. Initial Curve Generation

The trajectory optimization can be initialized using any defined curve, which could be manually specified. But in order to automate the process of finding a trajectory for a specified surface, this initial curve needs to be algorithmically generated. The initial curve generation does not need to find a perfect applicator trajectory, but the generated initial curve still needs to have some positive traits, all of which might be difficult or impossible for the subsequent optimization to fix. The main points in this regard are: i) It needs to cover the surface with sufficient paint. ii) It should have a simple topology with few or no intersections and twists.

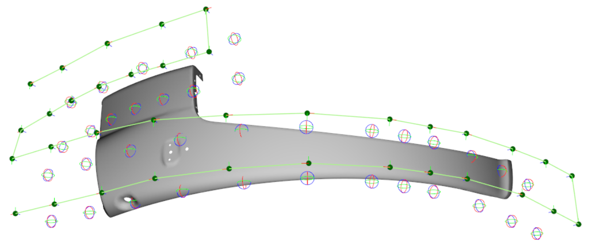
For the latter, we decided to follow the typical paint program pattern of having a sequence of sweeps. Each sweep has relatively low curvature, starts at one edge of the surface and ends at the opposite edge. After turning around at the edge, possibly both outside the surface and with the painting turned off, a new sweep follows parallel to the previous sweep but in the opposite direction. The distance between subsequent sweeps should be kept around a user specified value, based



(a) Raster curve generated on the rectangle R. The two arrows indicate how the user parameters d_{max} and w_{sweep} influence the generated raster curve. The sweep width is determined by w_{sweep}, while d_{max} is an upper limit on the distance between consecutive points, each sweep being evenly divided into the fewest numer of segments that will fulfill this inequality.



(b) Raster curve projected down onto the surface.



(c) Curve modified with extra waypoints and extended sweeps.

Fig. 2. Illustration of the steps when generating the initial curve. The spherical pivot markers, ⊕, are used to indicate the position and orientation of the projected points. In the first figure the points are initialized on the initial rectangle, shown in light gray. In the next two figures they are projected onto the triangulated surface, S, or its extension. Green points indicate paint nozzle positions at each waypoint, given by moving projected points a certain offset in the surface normal direction. The light green piecewise linear curve indicates the path the paint nozzle would follow when executing the painting instruction.

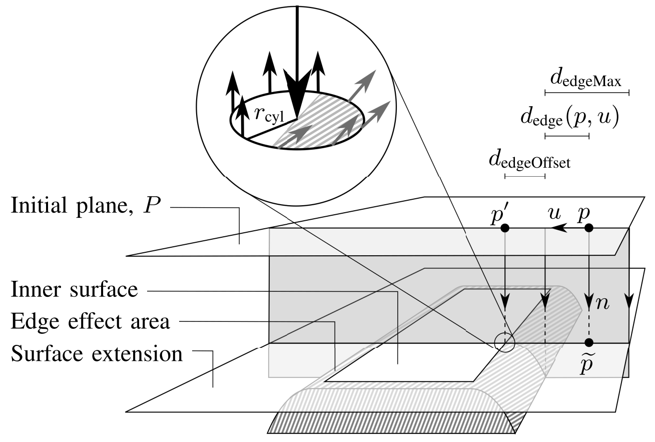


Fig. 3. Sketch of a simplified case of the pseudo projection method for a point, p, with parallel planes and right angles for the ray tracing direction, n, and the search direction, u. The surface, S, to be painted is here divided into an inner part, with small variations of the surface normal, and the edge effect area, which could have large surface normal variation and is avoided for surface normal calculations. The magnified circle represents the calculation of stable surface normals. Sampled surface normals shown as gray arrows in the shaded edge effect area are too close to the surface boundary and are not included in the calculation. The remaining samples, shown as black arrows, are used to calculate the new normal, $\tilde{n}$.

on the paint dispersion pattern of the bell. If the distance between sweeps becomes too large, it might be difficult, or even impossible, to get enough paint everywhere. Having too tight sweeps is not acceptable either.

We have strived for an initial curve generation algorithm that is simple and robust and can handle surfaces with low to moderate complexity. More specifically, we assume low to moderate curvature with no sharp corners. Further we assume a simple topology with a simply connected surface without holes, even though some number of smaller holes seems to be handled well. Using the trajectory optimization for more complex surfaces might require more information from the user when selecting an initial curve.

A. Algorithm Description

We are given a triangulated surface S to be painted, and a user-defined point and normal direction that together specify a plane P. We further have access to an efficient implementation of a ray tracing projection operator $\mathcal{P}_S(p, v)$, that given any point p and direction v computes the point on S closest to p along the line $\{p + \alpha v, \alpha \in \mathbb{R}\}$, if any such points exist. If no ray tracing projection onto S is found, we say that p is outside S.

The user may also influence the algorithm with the following parameters, which also are represented in Fig. 2a and Fig. 3.

- w_{sweep}, the required distance between consecutive sweeps.
- d_{max}, the maximum allowed segment length.
- d_{edgeMax}, the maximum distance to the surface edge for keeping waypoints outside the surface
- $d_{\text{edgeOffset}}$, the required distance to the surface edge for interior waypoints to be considered stable for normal estimation.
- r_{cyl}, a radius/distance used for surface normal smoothing.

With these input data and tools, the following steps, which are described in detail below, outline the method:

1) Compute a rectangle R on the plane P, onto which all surface points can be orthogonally projected. Generate a regular raster-based curve C_R on R.
2) Project each sweep of C_R onto S in a generalized sense to be defined in Section IV-B.
3) Modify the sweep to properly fit S.

The results after each of these steps are shown for a model example in Fig. 2. A simplified representation of the second step of the algorithm, along with a schematic description of most parameters can be found in Fig. 3.

Step 1: All surface mesh vertices of S are orthogonally projected onto the plane P, and a rectangle R is created on P containing all projected points with a certain margin.

A curve C_R is created as a simple raster, i.e., the sweeps are straight lines, spaced w_{sweep} apart, along one of the rectangle directions. The segment length is uniform and chosen as large as possible without exceeding $d_{\max}$.

Step 2: Each sweep of C_R is projected onto S in a generalized sense. A key concept is whether a point is *feasible*. This means that it is possible to compute a *pseudo-projection* onto S, as described in Section IV-B. For now, it is enough to point out that it does not only include points that actually project directly onto the surface using $\mathcal{P}_S$, but the pseudo-projection also extends outside the surface. The reason to allow points that in effect project outside the surface, is that these points also contribute paint to the surface, in particular close to surface edges.

Step 3: In steps 3a-d the sweep is modified to fix cases where consecutive waypoints are too close or too far apart, and to move the endpoints to the boundary of the feasible area.

Step 3a: Sequences of infeasible waypoints are removed at both ends of the sweep, while feasible points are projected onto S or its extension using pseudo projection. Sweeps where less than two points are feasible are discarded altogether. For interior non-feasible points, due to holes for instance, a new position is computed from neighbouring feasible points by linear interpolation.

Step 3b: The end points of the sweep are maximally extended tangentially while maintaining feasibility, i.e. extending them slightly further would render them infeasible.

Step 3c: Waypoints closer than $d_{\max}/2$ to each other are recursively merged until no such too-short segments remain. Merging of two waypoints is done by first replacing them with their average. If this new point is feasible, its pseudo projection is used, otherwise it is kept unchanged. An exception applies when one of the original points is an endpoint of a sweep. In that case, the endpoint is used, and the other waypoint point is discarded.

Step 3d: Segments longer than $d_{\max}$ are recursively divided into two segments until no such segments exist. Segment division is done by inserting a new point in the middle of the segment, and pseudo projection is applied to the new point if feasible.

B. Computing Normals and Pseudo-Projection

When a ray tracing projection is successful, the underlying surface model provides a surface normal at the projected point. A rather abrupt change in normal direction could be caused by natural irregularities of the surface, or at points close to the surface edges, where there is usually a flange, or an area with large curvature. This will lead to possibly large fluctuations of the normal, which from a painting perspective is something we would like to avoid. As previously discussed, we would also like to be able to project down points even if they are outside the surface. These are the main motivations for the pseudo-projection operation described below and shown in Fig. 3; to stabilize the surface normal computation against edge effects and small surface variations, and to extend the computation outside the surface.

Conceptually, the aim of the pseudo-projection and normal calculation is to reliably handle the projection of points, even if they like point p in Fig. 3 project outside the surface onto the surface extension. The calculation should also handle points that project onto an area close to the edge, referred to in the figure as the edge effect area, and shown as a gray shaded area. For these cases, we want to find a point p' that projects onto the inner part of the surface, where a stable normal can be calculated, and use this normal instead. The three surfaces seen in the figure, the inner surface, the edge effect area, and the surface extension, are not well-defined. They are rather meant to give a simplified, but useful, mental picture of how the user parameters specified in Section IV-A affect the ray tracing and line search steps presented in this section.

For each point p that we try to project onto the surface during the various phases of the method, its orientation is described by a right-handed orthonormal frame (t, b, n). The tangent, t, points along the direction of the curve, and the normal, n, typically points toward the surface. The *bi-direction*, b, is uniquely determined by the other two; $b = n \times t$. For points of the initial raster curve C_R, n is given by the user-defined normal direction of P and t is given as the sweep direction. For all subsequent points, n and t are outputs of the pseudo-projection algorithm as described below.

The main idea behind computing stable normals is to avoid edge effects by only using surface points on the inner part of S, or to otherwise use the closest such point. To do this we introduce a concept of edge distance $d_{\text{edge}}(p, u)$ to the surface edge ∂S, given a point p and a direction u. The edge distance is calculated using a simple line search along u based on ray tracing projections along the direction n, see the simplified sketch in Fig. 3. The search finds the transition point between being on or outside S. For simplicity we only consider the search directions $u = \pm t$ and $u = \pm b$. For a point outside S, we require that $d_{\text{edge}} < d_{\text{edgeMax}}$ for some of the four directions, otherwise the point is deemed infeasible. If $d_{\text{edge}} < d_{\text{edgeMax}}$ for some directions, or p is on the surface but $d_{\text{edge}} < d_{\text{edgeOffset}}$ for some directions, we search along these directions to the surface interior for a new point p' such that $d_{\text{edge}} > d_{\text{edgeOffset}}$ for all directions. If this is not possible, p is deemed infeasible.

The new interior point p' is the basis for the surface normal computation, but to make the calculation more robust, we also sample 8 points on the tangent plane spanned by p', t and b, uniformly sampled at distance r_{cyl} from p'. Points that are closer than $d_{\text{edgeOffset}}$ to the edge are discarded, see Fig. 3. The remaining normals are fed into a simple cluster algorithm, with the aim to avoid the influence of outlier normals when averaging.

After a new normal $\widetilde{n}$ has been computed according to above, a new tangent is given by

$$\widetilde{t} = t - (t \cdot \widetilde{n})\widetilde{n}.$$

The projected point is given by

$$\widetilde{p} = p + d\,\widetilde{n},$$

where d is the distance between p' and its projection on S. The point $\widetilde{p}$ and the unit vector $\widetilde{n}$ combined are the result of a successful *pseudo projection* of the right-handed orthonormal frame (t, b, n) onto S.

C. A Modified Method

The procedure described in Section IV-A has one major drawback. If the surface has a slope with respect to P in the direction perpendicular to the sweep directions, the distance between sweeps will increase after projection. This is illustrated in Fig. 4a. The way we deal with this is to use the projection method to create only a single, stable sweep. This sweep is then propagated sideways in both directions at proper distances. Each waypoint in the new sweep is moved in a direction perpendicular to both the normal direction and the mean of the direction of the incoming and outgoing segments. Thereafter, the sweep is pseudo projected and modified according to Section IV-A. The result of the modified method is shown in Fig. 4b. Here the sweeps cover the full surface, and they are more evenly spaced. This is a simpler version of the same basic method employed by [15], where an advanced geometrical analysis is used to pick the first sweep.

V. Optimization Problem Definition

After generating initial trajectories, the next step is to formulate an optimization problem that modifies the trajectory waypoints, in order to produce an even paint cover matching the target thickness. The problem is formulated as a continuous non-linear optimization problem, and the general form of the optimization problem can be stated using the following notation:

$$\begin{aligned} &\min_{x \in X} \; J(x), \\ &\text{such that } g(x) \geq 0, \\ &\qquad\qquad h(x) = 0. \end{aligned}$$

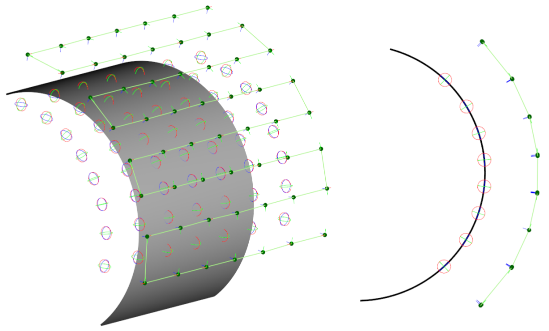
The cost function, J, is minimized over the variable space, X, to find optimal variable values, x, while fulfilling both inequality constraints, g, and equality constraints, h.

A. Variables

The variables in the optimization problem consist of all segment durations and all waypoints of the painting trajectory, including the direction of the applicator nozzle. This is described as a quaternion along with time durations of each segment between two consecutive waypoints. The notation used is x for waypoint positions, q for quaternion applicator directions and $\Delta\tau$ for segment durations. Here the notation is fully indexed for both n number of applicator waypoints and the variable dimensions at each waypoint, but often a simplified notation will be used, where one or more indices are dropped.

$$\begin{aligned} &x_{i,j}, &&i \in \{1, \ldots, n\}, \; j \in \{1, \ldots, 3\}, \\ &q_{i,j}, &&i \in \{1, \ldots, n\}, \; j \in \{1, \ldots, 4\}, \\ &\Delta\tau_i, &&i \in \{1, \ldots, n-1\} \end{aligned}$$

The index j is used for the three spacial dimensions of the position and the four quaternion values that specify the orientation.


(a) Curve generated using a raster curve projected onto the curved surface as described in Section IV-A.

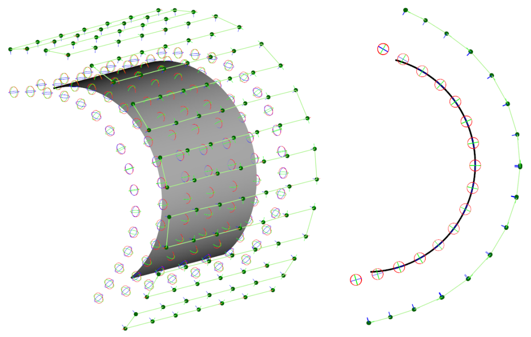

(b) Curve generated with the modified method where sweeps are added incrementally taking local curvature into account as described in Section IV-C.

Fig. 4. Illustration highlighting the differences in the initial curve when using the modified method described in Section IV-C. The example geometry is a simple curved geometry, with a side view shown on the right-hand side. Spherical pivot markers, $\oplus$, are used to indicate the position and orientation of waypoints projected on the surface or its extension. The light green curve shows the path of the paint nozzle, with green points marking its position at each waypoint.

B. Goal Function

The goal function combines different aspects of what an optimal trajectory should fulfill. The main consideration is to obtain a paint thickness that matches the target thickness as closely as possible for all areas of the painted geometry. The paint projection used to calculate the paint thickness is exemplified by Fig. 5 and consists of a reference paint thickness which is projected to the target geometry. The reference thickness is a radially symmetric spline curve that can be fitted to experimental data of spray painting along a straight line, by reversing the integration exemplified by the projection distributions of Fig. 6.

Apart from the paint thickness, there are other aspects of the applicator trajectory which also need to be included in the goal function, in order to obtain a trajectory that is executable. The goal function is therefore a sum of J_{paint}, which relates to the paint thickness objective, and J_{path}, which includes all costs related to the applicator trajectory.

The part of the objective function related to paint thickness is the sum of squared paint differences compared to the desired paint thickness T_{ref}

$$J_{\text{paint}} = \sum_k A_k \left(T(p_k) - T_{\text{ref}}\right)^2 .$$

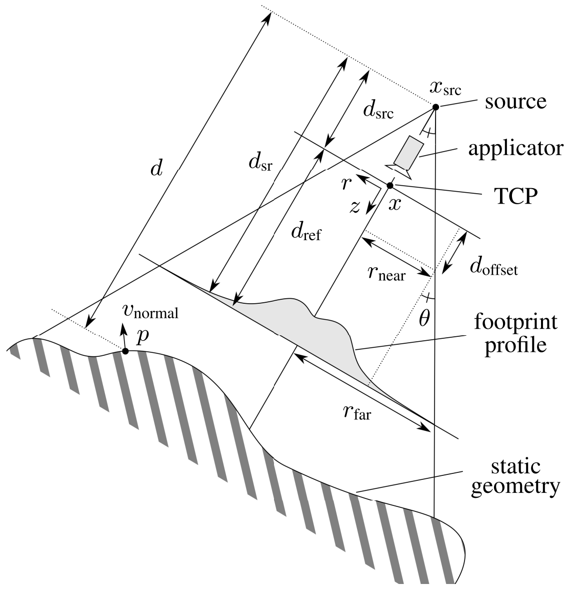


Fig. 5. Representation of the main components of the paint projection. The position of the modeled point source effects how the footprint profile is projected to the static geometry. The figure shows notations for spray cone distances and angles, a TCP-centered coordinate system as well as the position and normal vector of a point on the static geometry.

Here $T(p_k)$ is the paint thickness at node p_k of the surface, and A_k is the surface area represented by p_k. The regions A_k must partition the surface, and there are multiple such representations that could be used. The partitioning has been implemented using triangle vertices as nodes, and a third of the sum of the area of all triangles connected to the vertex as the corresponding area.

The goal function concerning the applicator path, J_{path}, consists of a weighted sum of objectives. It includes the squared difference of consecutive waypoints for the position and orientation of the applicator nozzle as well as the total duration of the application motion,

$$J_{\mathrm{path}} = w_x \sum_{i=1}^{n-1} \sum_{j=1}^{3} \left(x_{i+1,j} - x_{i,j}\right)^2 + w_q \sum_{i=1}^{n-1} \sum_{j=1}^{4} \left(q_{i+1,j} - q_{i,j}\right)^2 + w_t \sum_{i=1}^{n-1} \Delta\tau_i.$$

The positional and orientational objectives act similar to spring forces, evenly distributing points along the trajectory. The weights w_x and w_q determine how much changes in position and orientation along the trajectory are penalised. These minimization terms guide the search towards a trajectory that will be easier for a robot to execute. Smoothing changes between segments is also beneficial for the accuracy of the projection model, which is calibrated using long sweeps with constant velocity.

Including durations in the objective function makes it possible to prioritize short execution time by increasing the corresponding weight w_t. This constraint also has the added benefit that it will directly affect the amount of paint consumed when executing the trajectory, since the primary use case is an applicator with constant paint flow. Handling changes of paint flow at discrete trajectory points has also been implemented, but will not be further addressed here.

The paint thickness calculation is performed for each node of the target geometry, by integrating the amount of paint that each node receives over the span of the painting trajectory. Assuming the velocity of the paint applicator nozzle is constant over a single segment between waypoints, the paint thickness calculation is a sum of local paint flow contributions

$$T(p_k) = \sum_{i=1}^{n-1} \left(\Delta\tau_i \sum_{\ell} \Delta T(x_\ell, p_k) \right).$$

Here $\Delta\tau_i$ is the segment time and $\Delta T(x_\ell, p_k)$ is the calculated paint thickness contribution at a point, p_k, on the geometry for a paint applicator with its tool center point (TCP) at x_ℓ. The summation over index ℓ consists of the subdivision of each segment according to a user supplied time step. This thickness calculation can be seen as a discrete version of integrating a brush function $B(x, p)$ over the continuously parametrized path $x(s)$,

$$T(p) = \int B(x(s), p)ds.$$

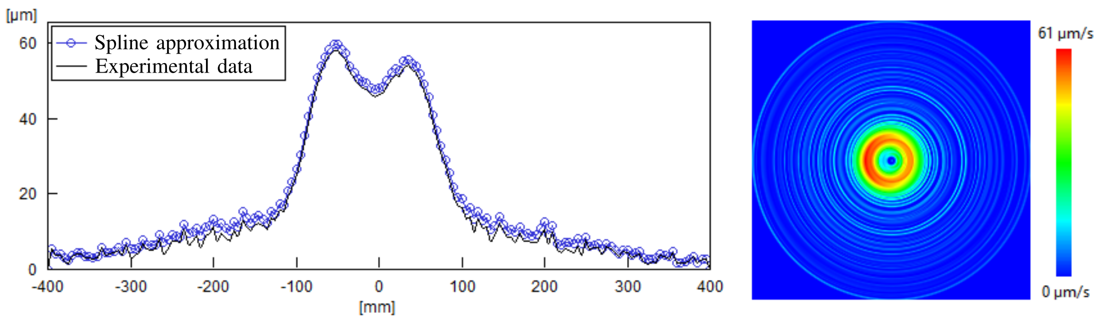
The brush function, $B(x, p)$, uses the current position of the applicator TCP, x, calculates the projection source point, x_{src}, and projects a footprint profile, $f(r)$, down on a specified point, p, on the fixed geometry. The footprint profile used here is an experimentally fitted curve of spline functions that tries to approximate the paint distribution obtained when spraying in a line over a flat surface at a specified distance, see Fig. 6. The brush function calculation consists of three parts, the footprint profile, a scalar product of the surface normal and the direction of the applied paint, and finally a scaling factor,

$$B(x(s), p) = f(r)(-v_{\mathrm{normal}} \cdot \overrightarrow{px}) \left(\frac{d_{\mathrm{sr}}}{d}\right)^2.$$

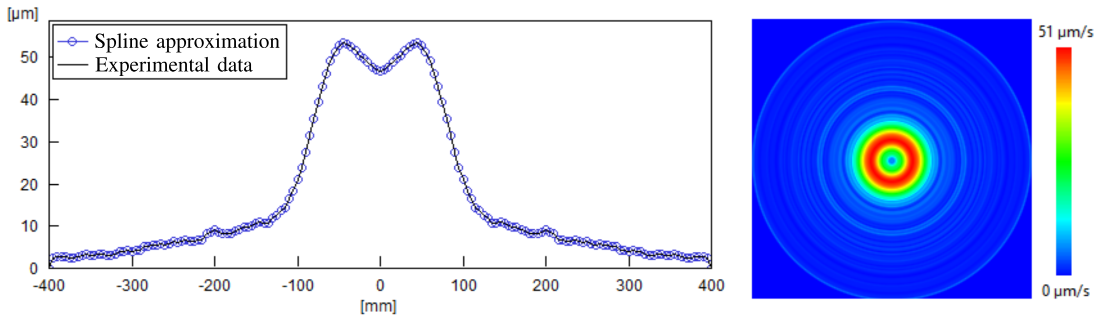
Here, d is the distance in the z-direction from the projection source to the current point on the geometry, with surface normal v_{normal}, see Fig. 5. The vector $\overrightarrow{px}$ represents the vector of the applied paint, which is a scaled vector from the position of the applicator projection source $x_{\mathrm{src}}(s)$ to the point p on the geometry,

$$\overrightarrow{px} = \frac{p - x_{\mathrm{src}}(s)}{d}.$$

Describing the projection method in a bit more detail, it starts by defining a projection cone. This gives a reasonable approximation to the deposition of paint for a range of distances to the target geometry. The paint applicator is modelled as a point source a distance d_{src} behind the TCP of the applicator, see Fig. 5. This distance d_{src} is calculated by using two defined points on the cone, one at the edge of the footprint profile and one point closer to the TCP. Using an (r, z)-plane with the TCP as origin, these two points are described by their displacements from the TCP, $(r_{\mathrm{far}}, d_{\mathrm{ref}})$ and $(r_{\mathrm{near}}, d_{\mathrm{offset}})$ respectively. The displacement values can be used to calculate the angle of the



(a) Spline approximation from the original experimental data.



(b) Spline approximation after original data is smoothed and made symmetric.

Fig. 6. A footprint profile has been obtained experimentally by measuring the paint thickness across a stroke of paint on a flat surface from a specified height. A cross section of the footprint profile of the brush function is shown for a) a fitted spline to the experimental data and b) after smoothing is applied and the data is made symmetric. The resulting projection distribution is also shown to the right using a color map range covering the obtained values.

projection cone,

$$\tan\theta = \frac{r_{\text{far}} - r_{\text{near}}}{d_{\text{ref}} - d_{\text{offset}}},$$

which makes it possible to find the distance from the TCP to the point source,

$$d_{\text{src}} = \frac{r_{\text{near}}}{\tan\theta} - d_{\text{offset}}.$$

The distance d_{sr} from the point source to the reference footprint profile is calculated and is used when projecting the footprint profile to the target geometry,

$$d_{\text{sr}} = d_{\text{src}} + d_{\text{ref}}.$$

C. Constraints

The solution is improved in an iterative procedure with an upper bound on how much each variable can change between iterations. These limitations are set using *box bounds*, lower and upper limits of the variables $x_{i,j}$, $q_{i,j}$ and $\Delta\tau_i$ that are modified at each iteration. There are also absolute upper and/or lower limits on most included variables, for example segment durations, along with constraints limiting a number of derived properties. These include limits on velocity, step length, deviations from a target distance from the geometry, applicator angle deviations from the surface normal and constraints to keep all quaternions normalized.

Selecting in what way the variables should be constrained is important, since the solver will be able to exploit inaccuracies in the model or cost function. As an example, the accuracy of the deposition model is only satisfactory in a region of parameter values around the combination of parameters where it was experimentally verified. If it was experimentally tested at a painting distance of 400 mm, then it might only be sufficiently accurate for distances in the range 300-600 mm. Another, more exploitable, issue is that the model is most accurate when the applicator direction is perpendicular to the surface. Without constraining the trajectory, the optimizer will find solutions with the applicator down close to and almost parallel to the surface, where the deposition cone is spread out over a larger area.

D. Optimization

With the problem formulated as a continuous non-linear optimization problem, a solution trajectory is found by using an interior point non-linear solver, IPOPT [20]. Ideally the optimization would be run to convergence, since this would represent a trajectory where, given user supplied preferences of weights, the different parts of the objective function are optimally balanced against each other. The weights in the objective function can be used to guide the optimization, but we also want to make use of information contained in the initial trajectory. Assuming the initial trajectory is generated to be feasible, both in terms of executability and model accuracy, limiting deviations from it is beneficial with respect to both of these aspects. The solver is run for a fixed number of iterations, in order to improve the initial solution, while keeping the solution in the feasible region. The deviation limits of each variable are reset in each run of the optimization, allowing

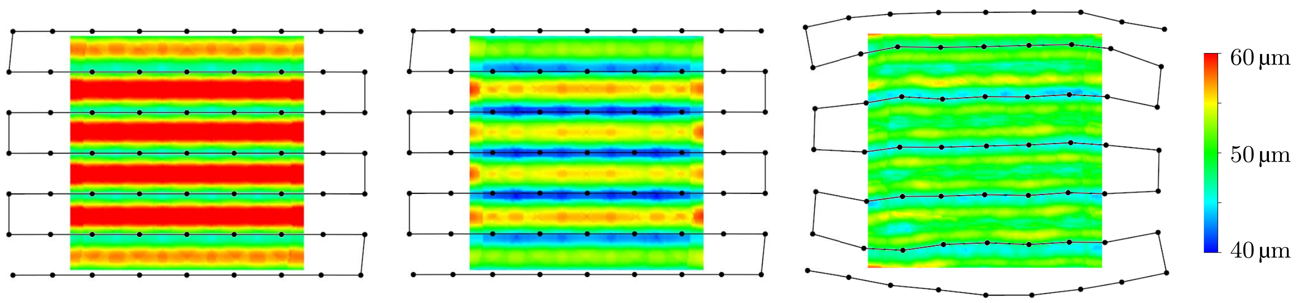


Fig. 7. Test case of a square plate with a side length of 1.0 m, an initial path with a sweep width of 180 mm, and a target thickness of 50 μm. The figures show from left to right the resulting paint thickness for the initial trajectory with a constant velocity, the thickness after applying a speed optimization but leaving the path unchanged, and after performing a full optimization where the trajectory points are also allowed to move.

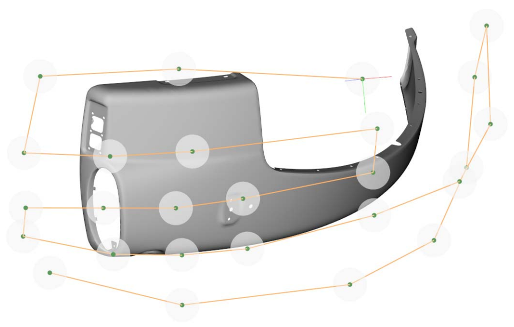
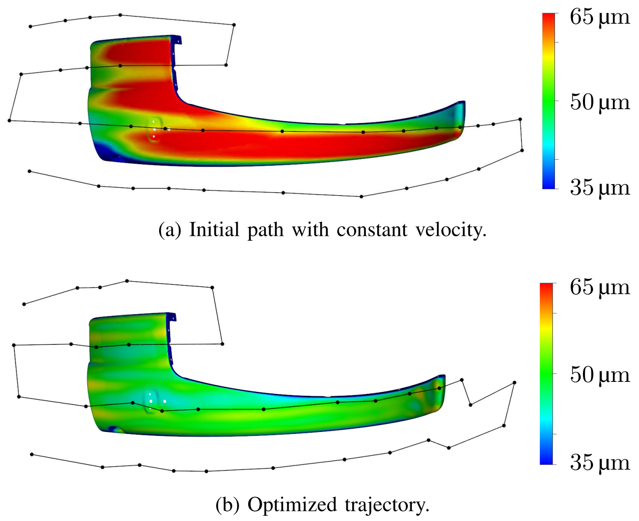


Fig. 8. Industrial test case of tractor fender to be painted, along with a representation of an example of an initial path.



(a) Initial path with constant velocity.

(b) Optimized trajectory.

Fig. 9. The figures show the simulated paint thickness that is obtained when using the same paint projection as is used in the optimization. The results are for an initial path with constant velocity, and an optimized trajectory, where the target thickness is 50 μm.

the trajectory to iteratively improve, even if the deviations are strictly limited in each iteration. The execution time for a single iteration increases with the size of the trajectory and the complexity of the geometry, which is why limiting the number of optimization steps is included as an option. This non-convergence in each iteration is not optimal but can be seen as a tolerable trade-off, especially when viewed as a local update of deviation limits during the iterative improvement of the trajectory.

VI. Results

A number of test cases have been setup to validate the accuracy of the process simulation and optimization. As an academic test example, to exemplify the performance of the optimization, a simple square plate is used. Fig. 7 shows an initial trajectory with a sweep width of 180 mm, where the typical banded thickness of the dual peak applicator function can be clearly seen. After optimizing the speed along the trajectory, the thickness differences are clearly reduced, but the banded structure is still clearly visible. Finally utilizing full optimization, where the trajectory points are also allowed to be moved and reoriented during the optimization, the result is further improved and most of the banded structure of the resulting paint thickness is smoothed out. As previously discussed, points outside the geometry can still contribute paint to the surface. Here we can see that the optimization solver has moved the top and bottom sweep away from the geometry. The sweeps are still contributing paint to the surface, but this might be a sign that it is possible to achieve a similar paint coverage even if the number of sweeps is reduced.

An example of an industrial case is seen in Fig. 8 where a CAD-model of a tractor fender is used as the target to be painted. Also visible in the figure is an example of an initial path, which consists of a few sweeps, curved around the geometry. The paint thickness obtained for an initial trajectory with four sweeps can be seen in Fig. 9 along with the thickness obtained after running the optimization. The optimized trajectory has a similar shape as the initial trajectory with most of the points only moved short distances. It also produces a paint thickness that for almost all regions of the target geometry is close to the target thickness of 50 μm.

A comparison between the projection-based deposition model described in this article and physical painting experiments is provided in Fig. 10. We refer to [19] for an explanation of the physical testing setup, while providing an overview of the results here. The graphs show that the optimization model has, based on the projection-based deposition model, provided a painting trajectory which gives a fairly uniform paint thickness close to the target of 50 μm. However, for measurement lines L4, L5, and L6 the agreement with the physical experiments is less than desirable. This was found to be due to the fact that the applicator gave a radically different

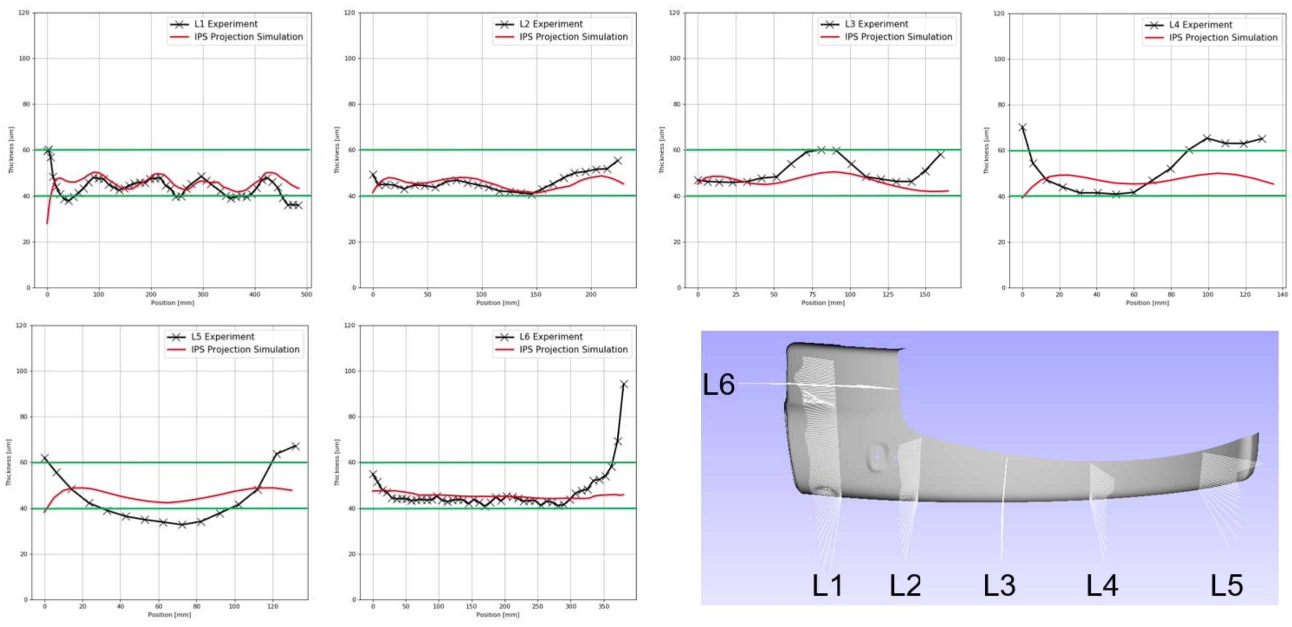


Fig. 10. Paint thickness measured on a physically painted tractor fender and a virtually simulated fender along six lines of measurement points. The red line in the graphs is the paint thickness from the projection-based deposition model and the black line from the physical experiments. The green lines represent the acceptable deviation from the target thickness 50 μm.

footprint when painting on a much narrower surface than the one painted when obtaining the footprint profile (which was a 1×1 metre plate). Furthermore, in the experiments there is an electrostatic effect which draws the paint towards the edge of the surface even when painting outside of it. This effect is also not captured by the projection-based deposition model and therefore the optimization model is unable to compensate for it. This effect can be observed clearly at the end of measurement line L6. This is an area where the painting trajectory turns outside of the surface such that the projection footprint does not hit the fender, but in the physical experiments the paint is accumulated on the edge of the fender due to the electrostatic effect. A discussion about these edge cases can be found in Section VII, and handling them is one of the main goals of later physics-based simulation modules in the SelfPaint workflow [18].

VII. Discussion

Both physical and virtual experiments found the results to be fairly insensitive to changes to the positional and orientational penalty weights w_x and w_q, or to the box bounds in the optimization model. Given an initial path that covers the surface and a normal industrial painting robot, these parameters do not need to be tuned in order to achieve an executable trajectory that gives uniform thickness. However, the parameter w_t, which penalizes the total time of the trajectory, may affect the result of the optimization drastically. A value that is too large will give a shorter execution time for the path, but at the cost of paint uniformity, and therefore this parameter should be tuned depending on the desired result.

How trajectories are formulated and how the initial curve is generated will affect the final result. The optimization step cannot alter the main outline of the initial curve. If the sweep width is chosen too large for instance, there will not be enough paint everywhere on the surface after optimization. Choosing too long segments will provide too few waypoints and too little flexibility for the optimization step. Too short segments, on the other hand, may lead to erratic behaviour. There is a fairly big window for the maximum segment length where the final result is not significantly affected. The same can be said for other moderate variations of the initial curve.

The modified version of the initial curve generation algorithm suffers from one potential problem. When propagating the first sweep forward and backward, small irregularities, like bending, tend to be accumulated and increased. This problem is addressed in [15], and is the motivation for using a start curve with low geodesic curvature that in effect splits the Gaussian curvature in two equal parts. The same principles could be used to select the best projected sweep. Another possible improvement could be to generate the full set of sweeps, and moving them closer to each other whenever they are too far apart. A few new sweeps might still have to be added at the ends, but clearly fewer than in the present method.

The painting trajectories have been modeled as a piecewise linear curve where the waypoints are used as optimization variables. In reality, a robot trajectory will often be smoothed out in the vicinity of waypoints to avoid the robot stopping at each waypoint. Larger deviations from the piecewise linear trajectory will typically further smooth out the velocity profile of the movement. One possible extension will be to more closely model the robot trajectory and incorporate additional robot code parameters into the trajectory optimization and generation, making use of previous work on robot controller modelling [21].

The projection-based deposition model used in the optimization has accuracy limitations, that in some cases might significantly affect the final result. It is therefore an ongoing work to make use of more accurate physics-based simulations of the paint deposition, to further improve the applicator trajectory. The proposed workflow is to use the projection-based optimization to improve the geometric path of the applicator, and then use results from the physics-based deposition model to further improve the velocity profile along the path. The last step is particularly important in order to compensate for changing footprints along the geometry, due to for example electrostatic effects, which the projection-based deposition model does not capture.

In the presented method, there is no inherent time dependent behaviour of the paint application process. In reality, such application dependent processes must be considered. For example, it is often desirable to paint neighbouring regions within a limited time frame. Also, the number of second passes over an area are often minimized in order to avoid degrading the surface finish. For future work, this is an interesting extension, to try to model the time dependent behaviour.

The presented solution is developed for single robot workstations. For multiple robots working on the same work piece, the time dependent behaviour as well as spatial interactions, for example airflow or electric potential, might warrant special consideration. However, if these effects are deemed minor enough to only marginally effect the paint distribution, the trajectory optimization presented here could be used independently for each robot. By even introducing some terms describing the interaction between robots, it might still be possible to use the presented solution to formulate a largely decoupled optimization problem.

VIII. Conclusion

The proposed optimization method is able to find a solution that satisfies many of the overall goals of a painting instruction. Its search is guided by generated initial trajectories that manage to cover the surface while minimizing intersections and twists. The optimization method makes use of a paint projection method, which has been accurately tuned to physical experiments on flat surfaces. The solution trajectories are feasible and have been executed on robots in an industrial test case to perform a physical evaluation of the results. The evaluation shows good agreement with physical results, especially in low curvature areas, while high curvature and edge effects are not fully captured with the presented method.

Going forward, the goal is to increase the usability and accuracy of the painting trajectory optimization. The presented results show feasibility of the method, and that it can be used to generate implementable solutions. When the resulting trajectory is evaluated using the projection method, the result is very promising. This shows that the optimization itself is not a significant cause of paint thickness deviations. For a given paint deposition model the optimization manages to find good solutions. It is however important to try to limit the possibility of the optimization algorithm taking advantage of weaknesses in the projection method. The accuracy of the paint deposition model will directly affect the resulting paint thickness. Physical testing for a large set of industrial cases will be needed to further evaluate deposition models and design choices in the optimization. Determining and handling problem cases with high paint thickness deviations will aid further development of the optimization workflow.

Acknowledgment

This work was carried out within the Production Area of Advance at the Chalmers University of Technology. The fender geometry was provided by John Deere, Bruchsal, Germany.

References

[1] H.-J. Streitberger and K.-F. Dossel, *Automotive Paints and Coatings*. Hoboken, NJ, USA: Wiley, 2008.

[2] D. Gleeson *et al.*, “Robot spray painting trajectory optimization,” in *Proc. IEEE 16th Int. Conf. Autom. Sci. Eng. (CASE)*, Aug. 2020, pp. 1135–1140.

[3] R. Bohlin and L. E. Kavraki, “Path planning using lazy PRM,” in *Proc. Millennium Conf. IEEE Int. Conf. Robot. Automat. Symposia (ICRA)*, vol. 1, Apr. 2000, pp. 521–528.

[4] A. Mark, R. Bohlin, D. Segerdahl, F. Edelvik, and J. S. Carlson, “Optimisation of robotised sealing stations in paint shops by process simulation and automatic path planning,” *Int. J. Manuf. Res.*, vol. 59, no. 1, pp. 4–26, 2014.

[5] D. C. Conner, A. Greenfield, P. N. Atkar, A. A. Rizzi, and H. Choset, “Paint deposition modeling for trajectory planning on automotive surfaces,” *IEEE Trans. Autom. Sci. Eng.*, vol. 2, no. 4, pp. 381–392, Oct. 2005.

[6] A. Mark *et al.*, “Simulation of electrostatic rotary bell spray painting in automotive paint shops,” *Atomization Sprays*, vol. 23, no. 1, pp. 25–45, 2013.

[7] B. Andersson *et al.*, “A modified TAB model for simulation of atomization in rotary bell spray painting,” *J. Mech. Eng. Automat.*, vol. 3, no. 2, pp. 54–61, 2013.

[8] T. Johnson, S. Jakobsson, B. Wettervik, B. Andersson, A. Mark, and F. Edelvik, “A finite volume method for electrostatic three species negative corona discharge simulations with application to externally charged powder bells,” *J. Electrostatics*, vol. 74, pp. 27–36, Apr. 2015.

[9] F. Edelvik, A. Mark, N. Karlsson, T. Johnson, and J. S. Carlson, “Math-based algorithms and software for virtual product realization implemented in automotive paint shops,” in *Math for the Digital Factory*. Cham, Switzerland: Springer, 2017, pp. 231–251.

[10] D. Hegels, T. Wiederkehr, and H. Müller, “Simulation based iterative post-optimization of paths of robot guided thermal spraying,” *Robot. Comput.-Integr. Manuf.*, vol. 35, pp. 1–15, Oct. 2015.

[11] S.-H. Suh, I.-K. Woo, and S.-K. Noh, “Development of an automatic trajectory planning system (ATPS) for spray painting robots,” in *Proc. IEEE Int. Conf. Robot. Autom.*, Apr. 1991, pp. 1948–1955.

[12] J. K. Antonio, “Optimal trajectory planning for spray coating,” in *Proc. IEEE Int. Conf. Robot. Autom.*, May 1994, pp. 2570–2577.

[13] H. Chen, W. Sheng, N. Xi, M. Song, and Y. Chen, “Automated robot trajectory planning for spray painting of free-form surfaces in automotive manufacturing,” in *Proc. IEEE Int. Conf. Robot. Autom.*, vol. 1, May 2002, pp. 450–455.

[14] Q. Yu, G. Wang, and K. Chen, “A robotic spraying path generation algorithm for free-form surface based on constant coating overlapping width,” in *Proc. IEEE Int. Conf. Cyber Technol. Autom., Control, Intell. Syst. (CYBER)*, Jun. 2015, pp. 1045–1049.

[15] P. N. Atkar, H. Choset, and A. A. Rizzi, “Towards optimal coverage of 2-dimensional surfaces embedded in IR^3: Choice of start curve,” in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, vol. 4, Oct. 2003, pp. 3581–3587.

[16] 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.

[17] W. Chen and D. Zhao, “Path planning for spray painting robot of workpiece surfaces,” *Math. Problems Eng.*, vol. 2013, Aug. 2013, Art. no. 659457.

[18] F. Edelvik, O. Tiedje, J. Jonuscheit, and J. S. Carlson, “SelfPaint–A self-programming paint booth,” *Proc. CIRP*, vol. 72, pp. 474–479, Feb. 2018.

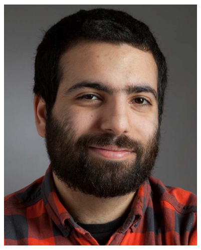
[19] N. Güttler *et al.*, “A self-programming painting cell ‘SelfPaint’: Simulation-based path generation with automized quality control for painting in small lot sizes,” in *Proc. Adv. Automot. Prod. Technol.-Theory Appl., Stuttgart Conf. Automot. Prod. (SCAP)*, 2021, pp. 302–310.

[20] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Math. Program.*, vol. 106, no. 1, pp. 25–57, 2006.

[21] D. Gleeson, C. Larsen, J. S. Carlson, and B. Lennartson, "Implementation of a rapidly executing robot controller," in *Proc. IEEE 15th Int. Conf. Autom. Sci. Eng. (CASE)*, Aug. 2019, pp. 1341–1346.

Daniel Gleeson was born in Säter, Sweden, in 1988. He received the M.Sc. degree in engineering physics from the Chalmers University of Technology, Gothenburg, Sweden, in 2012, where he is currently pursuing the Ph.D. degree with the Electrical Engineering Department, with a part time employment. After his M.Sc. degree, he has been employed by the Fraunhofer-Chalmers Research Centre for Industrial Mathematics, Gothenburg, where he has worked in robotics within the Department of Geometry and Motion Planning. His research focus is on robot trajectory optimization and robot controller emulation.

Stefan Jakobsson was born in Uppsala in 1970. He received the Ph.D. degree in mathematics from Lund University in 2000. Between 2001 and 2006, he worked at the Swedish Defence Research Agency, FOI, and at the Fraunhofer Chalmers Centre between 2006 and 2017. Since 2018, he has been working with GE Additive with process modelling and simulations for additive manufacturing.



Raad Salman moved from Luleå in 2009 and started his studies with the Mathematics Department, Gothenburg University, in 2010. He received the Licentiate degree in applied mathematics in 2017, after studying algorithms for variants of the travelling salesperson problem. He currently works as a Research Engineer at the Fraunhofer-Chalmers Centre, Gothenburg, mainly working with robotics for surface treatment applications and optimization problems in production systems.

Fredrik Ekstedt received the M.S. and Ph.Lic. degrees in applied mathematics from Gothenburg University, Gothenburg, Sweden, in 1994 and 1997, respectively. In 1999, he joined Ericsson Microwave Systems, Gothenburg, developing algorithms for radar target tracking. In 2002, he joined the Fraunhofer-Chalmers Centre for Industrial Mathematics, Gothenburg, as an Applied Researcher, where his fields of research has ranged from image and signal processing to computational geometry and combinatorial optimization.

Niklas Sandgren received the M.Sc. degree in engineering physics and complex adaptive systems from the Chalmers University of Technology in 2013. His passion is the cross-section between science and new technologies where he has been developing tools and algorithms for GPU computations, cloud computations, and VR applications. He has started a company developing a cloud native workflow for engineering calculations. For many years, he has been active in research of algorithms, methodologies, and tools for virtual manufacturing in general and paint shop processes in particular. He is currently enabling virtual product realization for manufacturing companies at Industrial Path Solutions and the Fraunhofer-Chalmers Research Centre for Industrial Mathematics.

Fredrik Edelvik was born in Örebro, Sweden, in 1972. He received the M.S. degree in engineering physics and the Ph.D. degree in scientific computing from Uppsala University, Uppsala, Sweden, in 1997 and 2002, respectively. From 2003 to 2004, he was a Post-Doctoral Research Associate at the Institut für Theorie Elektromagnetischer Felder (TEMF), Technische Universität Darmstadt, Darmstadt, Germany. In 2006, he was appointed as an Associate Professor in Scientific Computing at Uppsala University. Since 2005, he has been with the Fraunhofer-Chalmers Research Centre for Industrial Mathematics, Gothenburg, Sweden, where he is the Vice Director and the Head of the Computational Engineering Department. He has authored and coauthored more than 130 publications in the computational engineering area. He has co-founded two software companies that develop simulation tools to support virtual product and process development.

Johan S. Carlson was born in 1972. He received the Ph.D. degree in mathematical statistics on how to reduce geometrical variation in assembled products from the Chalmers University of Technology in 2000. He is the Director of the Fraunhofer-Chalmers Research Centre for Industrial Mathematics, FCC, and is heading the Department of Geometry and Motion Planning. He has over 20 years of experience with industrial development and implementation related to mathematics as a leading edge in virtual product realization and most of the results have been transferred into commercial software products and working procedures. His research interests include methods, algorithms, and tools for virtual product realization and in particular geometry simulation and assurance.

Bengt Lennartson (Fellow, IEEE) was born in Gnosjö, Sweden, in 1956. He received the Ph.D. degree from the Chalmers University of Technology, Gothenburg, Sweden, in 1986.

Since 1999, he has been a Professor of the Chair of Automation, Department of Electrical Engineering. From 2004 to 2007, he was the Dean of Education at the Chalmers University of Technology, and since 2005, he has been a part-time Professor with University West, Trollhättan. He is a (co)author of two books and over 300 peer-reviewed papers in international journals and conferences. His main areas of interest include discrete event and hybrid systems, AI planning and learning, as well as robust feedback control. He is a fellow of IEEE for his contributions to hybrid and discrete event systems for automation and sustainable production. He was the General Chair of the 11th IEEE Conference on Automation Science and Engineering (CASE) 2015 and the 9th International Workshop on Discrete Event Systems (WODES'08), and an Associate Editor of *Automatica* from 2002 to 2005 and IEEE TRANSACTIONS ON AUTOMATION SCIENCE AND ENGINEERING from 2012 to 2015.