Output Maneuvering for Cartesian 3D Printer

Andreas Hanssen Moltumyr*, Mathias Hauan Arbo, Jan Tommy Gravdahl
Department of Engineering Cybernetics
Norwegian University of Science and Technology (NTNU)
Trondheim, Norway
*andreas.h.moltumyr@ntnu.no

***Abstract*— 3D printing, also known as additive manufacturing, is a production technique that can create highly customized parts and is therefore ideal for product prototyping and customized orders. An important aspect of 3D printer systems is the ability to accurately and precisely move the extruder along a planned path, ensuring the production of parts with low dimensional error. In this paper, output maneuvering is considered for the purpose of steering the extruder of a Cartesian 3D printer along a desired path. As slicing software provides waypoints with minimal change in angle between the line segments, a novel speed profile adjustment is introduced which prioritizes maintaining the current along-path speed when the angle between line segments is sufficiently low. Through a design example, a nonlinear maneuvering controller consisting of a geometric and a dynamic task is deduced. Positive and negative aspects of applying output maneuvering to additive manufacturing are discussed.**

***Index Terms*— Output Maneuvering, 3D printing, Additive Manufacturing, Non-linear control**

I. Introduction

In order for a 3D printer or additive manufacturing system to create highly customized parts with intended geometry, the extruder will have to be accurately and precisely moved around the print surface and the so-far-printed object. This must be done for new layers to properly bond to the object and to avoid collisions between the printer and the print. Failure to do so will lead to faulty parts, and waste of time and material. These failures may reduce the reliability of the manufacturing method overall, rendering it more costly than desired.

The most commonly available 3D printers use stepper motors to accurately track a reference trajectory that has been precomputed by a slicing software. This is a form of trajectory tracking where the dynamic model of the printer is not taken into account. Separating the path following and the geometric path deviation into a geometric and a dynamic task to be achieved by the motor controller allows for specific control of the desired speed profile between any waypoints, adhering to the physical limitations of the mechanical system and the additive manufacturing process.

Aguiar et al. [1], [2] have shown that the two-task path-following approach does not suffer from the same performance limitations as reference-tracking in the case of non-minimum phase systems, e.g. systems with flexible joint such as printers with flexible timing belts. Model predictive control has been successfully applied to path-following problem formulations in order to incorporate constraints on input and state signals [3] and [4]. Output maneuvering, as described by Skjetne et al. [5] is a form of path-following control where a specific speed profile is followed between waypoints, the method has been demonstrated with an adaptive control method for a ship model [6].

In this paper, the method of output maneuvering has been used to design a path-following controller for a simple Cartesian 3D printer head. Through a design example, a simple model of a Cartesian 3D printer is derived before an output maneuvering controller is designed for the system through Lyapunov analysis, a novel method is introduced to adjust the speed profile according to the angle between subsequent line segments of the model to only slow down for sharp corners. Lastly, a simulation of the Cartesian 3D printer with the derived controller is presented. The material on output maneuvering presented in this paper builds on [5].

II. Output Maneuvering

The goal of output maneuvering is for a time-dependent vector $\boldsymbol{y}(t)$, e.g. the position of the extruder head, to track some desired trajectory $\boldsymbol{y}_d(t)$, e.g. the desired print path, with high accuracy, while also satisfying some secondary objective like a timing law, speed law or acceleration law. This can be divided into two parts, a geometric task, and a dynamic task.

The objective of the geometric task is to ensure that the output converges to a path parametrized with θ. We need

$$\lim_{t\to\infty} \left\| \boldsymbol{y}(t) - \boldsymbol{y}_d(\theta(t)) \right\|_2 = 0. \tag{1}$$

The secondary objective, namely the dynamic task should make the output satisfy some dynamic behavior along the path. A much used dynamic task that we will be using in the following design example is speed assignment. With speed assignment, we have the dynamic task

$$\lim_{t\to\infty} \left| \dot{\theta}(t) - v_s(\theta(t), t) \right| = 0. \tag{2}$$

III. Theory

In this section, a design example of output maneuvering for a Cartesian 3D printer is presented. In III-A a simple model of a Cartesian manipulator is derived. In III-C an output maneuvering controller will be designed through Lyapunov analysis. Lastly, part IV presents the details of the simulation, while part IV-C presents the simulation results from tracing a multi-layered, triangle-shaped, trajectory.

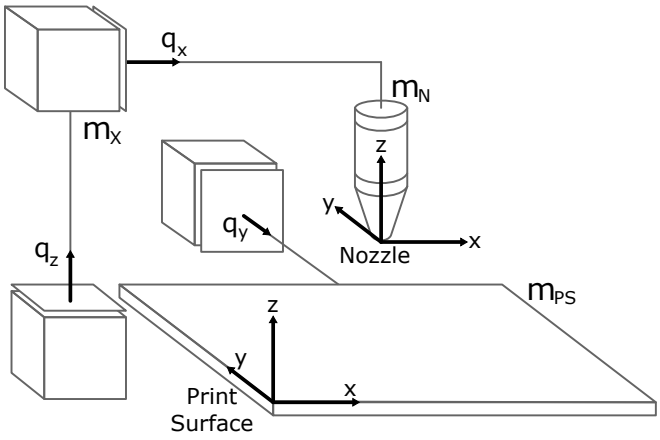


Fig. 1: Cartesian 3D printer model based on Prusa i3 [8].

A. 3D Printer Modelling

We use the Prusa i3 commercial 3D printer [8] as a basis for the model of a Cartesian 3D printer system. In the Prusa i3, the print surface is moved in the y direction relative to the world coordinate system. The nozzle is moved in the x and z direction relative to the world coordinate system. This gives the nozzle the ability to move with three degrees of freedom with respect to the print surface. In the Prusa i3, the linear motion in the x and y directions are facilitated by timing belts, each connected to one of four stepper motors. The motion in z direction is facilitated by leadscrews connected to the last two stepper motors, but for simplicity, we will also model the linear motion in z direction with a timing belt. In this study, we use regular DC motors as actuators and not stepper motors, since they are simpler to model with continuous transfer functions than stepper motors. It could also be mentioned that the high torques produced by stepper motors should not be necessary when positioning a 3D-printer nozzle since there are very little external forces acting on the printer head compared to the forces required for CNC applications like milling.

Ignoring the electrical dynamics of the DC motors we have the following dynamic equations for the motors [7]:

$$J_m\ddot{\theta} + B\dot{\theta} = \frac{K_m}{R_a}V - \tau_{load}. \tag{3}$$

Here, $\dot{\theta}$ and $\ddot{\theta}$ are the angular velocity and acceleration of the motor shaft, τ_{load} is the load torque on the motor shaft from the timing belt, V is the motor input voltage. J_m is the moment of inertia of the motor, shaft, and pulley gear. B, K_m and R_a are, respectively, the motor friction, torque constant, and armature resistance.

Using that $\dot{\theta} = \frac{1}{r}\dot{x}$ and $\tau = rF$ where $\dot{x}$ is the linear speed of the timing belt and r is the radius of the pulley gear connecting the motor shaft to the timing belt and F is the linear force experienced by the timing belt, we find a linear motion variant of the differential equation (3).

$$\frac{J_m}{r^2}\ddot{x} + \frac{B}{r^2}\dot{x} = \frac{K_m}{rR_a}V - F_{load}. \tag{4}$$

Choosing the displacement of each of the three timing belts as generalized coordinates as shown in Fig. 1, the Lagrangian, total kinetic, and potential energy of the system can be expressed as

$$\mathcal{L} = \mathcal{K} - \mathcal{P}, \tag{5}$$

$$\mathcal{K} = \frac{1}{2}m_N\dot{q}_x^2 + \frac{1}{2}m_{PS}\dot{q}_y^2 + \frac{1}{2}(m_X + m_N)\dot{q}_z^2, \tag{6}$$

$$\mathcal{P} = g(m_X + m_N)q_z, \tag{7}$$

where m_X, m_N and m_{PS} are the mass of the x-axis motor, extruder system, and print surface, respectively. g is the gravitational constant.

Applying the Euler-Lagrange equations of motion, we get

$$m_N\ddot{q}_x = F_{x,load}, \tag{8}$$

$$m_{PS}\ddot{q}_y = F_{y,load}, \tag{9}$$

$$(m_N + m_X)\ddot{q}_z + (m_N + m_X)g = F_{z,load}. \tag{10}$$

Adding the actuator dynamics from (4) for each dimension and combine the terms, we get

$$\left(m_N + \frac{J_m}{r^2}\right)\ddot{q}_x + \frac{B}{r^2}\dot{q}_x = \frac{K_m}{rR_a}V_x, \tag{11}$$

$$\left(m_{PS} + \frac{J_m}{r^2}\right)\ddot{q}_y + \frac{B}{r^2}\dot{q}_y = \frac{K_m}{rR_a}V_y, \tag{12}$$

$$\left(m_N + m_X + \frac{J_m}{r^2}\right)\ddot{q}_z + \frac{B}{r^2}\dot{q}_z + (m_N + m_X)g = \frac{K_m}{rR_a}V_z'. \tag{13}$$

Setting

$$V_z' = V_z + \frac{rR_a}{K_m}(m_N + m_X)g, \tag{14}$$

we add a constant term to the control input in z-direction to remove the effect of gravity when considering motion control.

The dynamic equations for the Cartesian 3D printer can then be expressed in matrix form as

$$M\ddot{\boldsymbol{q}} + D\dot{\boldsymbol{q}} = \boldsymbol{u}, \tag{15}$$

where $\boldsymbol{q} = [q_x, q_y, q_z]^T$,

$$M = \begin{bmatrix} m_N + \frac{J_m}{r^2} & 0 & 0 \\ 0 & m_{PS} + \frac{J_m}{r^2} & 0 \\ 0 & 0 & m_N + m_X + \frac{J_m}{r^2} \end{bmatrix}, \tag{16}$$

$$D = \begin{bmatrix} \frac{B}{r^2} & 0 & 0 \\ 0 & \frac{B}{r^2} & 0 \\ 0 & 0 & \frac{B}{r^2}, \end{bmatrix}, \tag{17}$$

and

$$\boldsymbol{u} = k_1\boldsymbol{w} = \frac{K_m}{rR_a}\left[V_x, V_y, V_z\right]^T. \tag{18}$$

Setting $\boldsymbol{x}_1 = \boldsymbol{q}$ and $\boldsymbol{x}_2 = \dot{\boldsymbol{q}}$ we can express (15) in the following equivalent form

$$\dot{\boldsymbol{x}}_1 = \boldsymbol{x}_2, \tag{19}$$

$$\dot{\boldsymbol{x}}_2 = -M^{-1}D\boldsymbol{x}_2 + M^{-1}k_1\boldsymbol{w}. \tag{20}$$

B. Piecewise Linear Path

In this paper, we will only consider the problem of following piece-wise linear paths. This will in most cases be sufficient since regular slicer software works on triangle meshes and produces a sequence of points or a set of connecting polylines. One way of representing a piecewise linear path $\boldsymbol{y}_d(\theta)$ given a set of $N+1$ points $\{\boldsymbol{p}_0, \ldots, \boldsymbol{p}_N\}$ is

$$\boldsymbol{y}_d(\theta) = (\boldsymbol{p}_n - \boldsymbol{p}_{n-1})\frac{\theta - \theta_{n-1}}{\theta_n - \theta_{n-1}} + \boldsymbol{p}_{n-1}, \tag{21}$$

for $\theta \in [\theta_{n-1}, \theta_n)$, $n = 1, \ldots, N$. The first and second derivative with respect to θ is

$$\frac{\partial \boldsymbol{y}_d}{\partial \theta}(\theta) = \frac{\boldsymbol{p}_n - \boldsymbol{p}_{n-1}}{\theta_n - \theta_{n-1}}, \tag{22}$$

$$\frac{\partial^2 \boldsymbol{y}_d}{\partial \theta^2}(\theta) = \boldsymbol{0}, \tag{23}$$

for $\theta \in [\theta_{n-1}, \theta_n)$, $n = 1, \ldots, N$.

If we choose $\theta_n - \theta_{n-1} = 1$, the line segments have a length of one in the parameter θ, and we can simplify (21) and (22) to

$$\boldsymbol{y}_d(\theta) = (\boldsymbol{p}_n - \boldsymbol{p}_{n-1})(\theta - \theta_{n-1}) + \boldsymbol{p}_{n-1} \tag{24}$$

and

$$\frac{\partial \boldsymbol{y}_d}{\partial \theta}(\theta) = \boldsymbol{p}_n - \boldsymbol{p}_{n-1} \tag{25}$$

respectively.

C. Output Maneuvering Lyapunov Design

We want to control nozzle position $\boldsymbol{y} = \boldsymbol{x}_1$ and start by defining error variables for position and velocity

$$\boldsymbol{z}_1 = \boldsymbol{y} - \boldsymbol{y}_d(\theta) = \boldsymbol{x}_1 - \boldsymbol{y}_d(\theta), \tag{26}$$

$$\boldsymbol{z}_2 = \boldsymbol{x}_2 - \boldsymbol{\alpha}_1(\boldsymbol{x}_1, \theta), \tag{27}$$

where $\boldsymbol{\alpha}_1$ is a virtual control input to be specified later. We also define

$$\omega_s = v_s(\theta) - \dot{\theta}, \tag{28}$$

an internal state in the to-be-designed controller, representing the error between $v_s(\theta)$, the speed assignment along the path parameterized in θ, and the system's propagation speed along the path. Note that

$$\dot{\boldsymbol{z}}_1 = \dot{\boldsymbol{x}}_1 - \frac{\partial \boldsymbol{y}_d}{\partial \theta}\dot{\theta} = \boldsymbol{z}_2 + \boldsymbol{\alpha}_1 + \frac{\partial \boldsymbol{y}_d}{\partial \theta}(\omega_s - v_s). \tag{29}$$

We define the first Lyapunov control function

$$V_1 = \frac{1}{2}\boldsymbol{z}_1^T P_1 \boldsymbol{z}_1. \tag{30}$$

giving

$$\dot{V}_1 = \boldsymbol{z}_1^T P_1 \dot{\boldsymbol{z}}_1 \tag{31}$$

$$= \boldsymbol{z}_1^T P_1 (\boldsymbol{z}_2 + \boldsymbol{\alpha}_1 + \frac{\partial \boldsymbol{y}_d}{\partial \theta}\omega_s - \frac{\partial \boldsymbol{y}_d}{\partial \theta}v_s). \tag{32}$$

Choosing the virtual control input

$$\boldsymbol{\alpha}_1 = A_1 \boldsymbol{z}_1 + \frac{\partial \boldsymbol{y}_d}{\partial \theta}v_s, \tag{33}$$

where A_1 is a Hurwitz matrix satisfying $P_1 A_1 + A_1^T P_1 = -Q_1$ where P_1 and Q_1 are positive definite matrices. Defining a first tuning function $\tau_1 = \boldsymbol{z}_1^T P_1 \frac{\partial \boldsymbol{y}_d}{\partial \theta}$, we find that

$$\dot{V}_1 = \boldsymbol{z}_1^T P_1 \boldsymbol{z}_2 + \frac{1}{2}\boldsymbol{z}_1^T (P_1 A_1 + A_1^T P_1)\boldsymbol{z}_1 \tag{34}$$

$$+ \boldsymbol{z}_1^T P_1 \frac{\partial \boldsymbol{y}_d}{\partial \theta}\omega_s, \tag{35}$$

$$= -\frac{1}{2}\boldsymbol{z}_1^T Q_1 \boldsymbol{z}_1 + \boldsymbol{z}_1^T P_1 \boldsymbol{z}_2 + \tau_1 \omega_s. \tag{36}$$

Define the second Lyapunov control function

$$V_2 = V_1 + \frac{1}{2}\boldsymbol{z}_2^T P_2 \boldsymbol{z}_2. \tag{37}$$

Note that

$$\dot{\boldsymbol{\alpha}}_1 = A_1 \dot{\boldsymbol{z}}_1 + \left(\frac{\partial^2 \boldsymbol{y}_d}{\partial \theta^2}v_s + \frac{\partial \boldsymbol{y}_d}{\partial \theta}\frac{\partial v_s}{\partial \theta}\right)\dot{\theta} \tag{38}$$

$$= A_1 \boldsymbol{x}_2 + \left(-A_1 \frac{\partial \boldsymbol{y}_d}{\partial \theta} + \frac{\partial \boldsymbol{y}_d}{\partial \theta}\frac{\partial v_s}{\partial \theta}\right)\dot{\theta} \tag{39}$$

$$= A_1 \boldsymbol{x}_2 + \frac{\partial \boldsymbol{\alpha}_1}{\partial \theta}\dot{\theta}. \tag{40}$$

Also note that

$$\dot{\boldsymbol{z}}_2 = \dot{\boldsymbol{x}}_2 - \dot{\boldsymbol{\alpha}}_1 \tag{41}$$

$$= -M^{-1} D \boldsymbol{x}_2 + M^{-1} k_1 \boldsymbol{w} - A_1 \boldsymbol{x}_2 - \frac{\partial \boldsymbol{\alpha}_1}{\partial \theta}\dot{\theta}. \tag{42}$$

We find

$$\dot{V}_2 = \dot{V}_1 + \boldsymbol{z}_2^T P_2 \dot{\boldsymbol{z}}_2 \tag{43}$$

$$= -\frac{1}{2}\boldsymbol{z}_1^T Q_1 \boldsymbol{z}_1 + \boldsymbol{z}_1^T P_1 \boldsymbol{z}_2 + \tau_1 \omega_s$$
$$+ \boldsymbol{z}_2^T P_2 \Big(M^{-1} k_1 \boldsymbol{w} - M^{-1} D \boldsymbol{x}_2 \tag{44}$$
$$- A_1 \boldsymbol{x}_2 + \frac{\partial \boldsymbol{\alpha}_1}{\partial \theta}(\omega_s - v_s)\Big).$$

Choosing control input $\boldsymbol{w}$ as

$$\boldsymbol{w} = \frac{1}{k_1} M \Big(A_2 \boldsymbol{z}_2 - P_2^{-1} P_1 \boldsymbol{z}_1$$
$$+ M^{-1} D \boldsymbol{x}_2 + A_1 \boldsymbol{x}_2 + \frac{\partial \boldsymbol{\alpha}_1}{\partial \theta} v_s \Big), \tag{45}$$

where A_2 is a Hurwitz matrix satisfying $P_2 A_2 + A_2^T P_2 = -Q_2$ where $P_2 = P_2^T$ and Q_2 are positive definite matrices, and define a second tuning function $\tau_2 = \boldsymbol{z}_2^T P_2 \frac{\partial \boldsymbol{\alpha}_1}{\partial \theta}$. We find

$$\dot{V}_2 = -\frac{1}{2}\boldsymbol{z}_1^T Q_1 \boldsymbol{z}_1 + \frac{1}{2}\boldsymbol{z}_2^T \big(P_2 A_2 + A_2^T P_2\big)\boldsymbol{z}_2$$
$$+ \tau_1 \omega_s + \tau_2 \omega_s \tag{46}$$

$$= -\frac{1}{2}\boldsymbol{z}_1^T Q_1 \boldsymbol{z}_1 - \frac{1}{2}\boldsymbol{z}_2^T Q_2 \boldsymbol{z}_2 + (\tau_1 + \tau_2)\omega_s. \tag{47}$$

Now define

$$\boldsymbol{z} = \left[\boldsymbol{z}_1^T, \boldsymbol{z}_2^T\right]^T, \qquad \boldsymbol{b} = \left[\frac{\partial \boldsymbol{y}_d^T}{\partial \theta}, \frac{\partial \boldsymbol{\alpha}_1^T}{\partial \theta}\right]^T, \tag{48}$$

$$Q = \operatorname{diag}(Q_1, Q_2), \qquad P = \operatorname{diag}(P_1, P_2), \tag{49}$$

and write (47) in compact form

$$\dot{V}_2 = -\frac{1}{2}\boldsymbol{z}^T Q \boldsymbol{z} + \boldsymbol{z}^T P \boldsymbol{b}\, \omega_s. \tag{50}$$

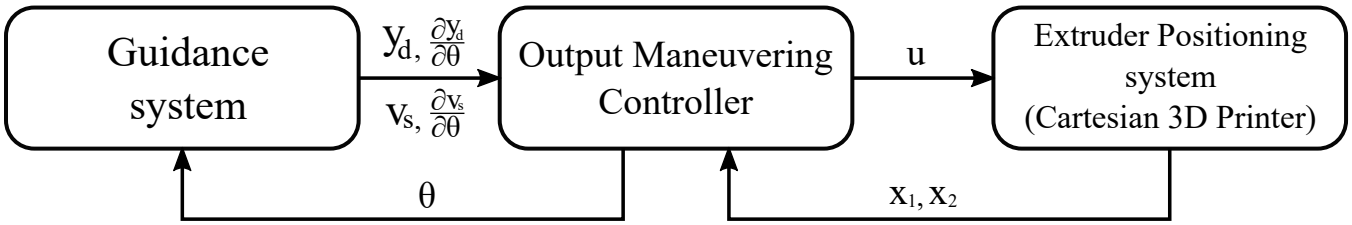


Fig. 2: Output maneuvering controller structure.

In order to make (50) negative definite and the output maneuvering controller asymptotically stable we have to close the loop with a speed assignment update law. Following the approach in [5], we apply the filtered-gradient update law. Define the third, and final, Lyapunov control function

$$V = V_2 + \frac{1}{2\lambda\mu_1}\omega_s^2, \tag{51}$$

and the speed law

$$\dot{\omega}_s = -\lambda(\omega_s + \mu_1 \boldsymbol{z}^T P \boldsymbol{b}). \tag{52}$$

Here, $\mu_1 > 0$ is the gain coefficient of the gradient update law, and $\lambda > 0$ is the cut-off frequency of a first order low-pass filter used for the reduction of measurement noise.

We find

$$\dot{V} = \dot{V}_2 + \frac{1}{\lambda\mu_1}\omega_s\dot{\omega}_s \tag{53}$$

$$= -\frac{1}{2}\boldsymbol{z}^T Q \boldsymbol{z} + \boldsymbol{z}^T P \boldsymbol{b}\omega_s - \frac{1}{\mu_1}\omega_s^2 - \boldsymbol{z}^T P \boldsymbol{b}\omega_s \tag{54}$$

$$= -\frac{1}{2}\boldsymbol{z}^T Q \boldsymbol{z} - \frac{1}{\mu_1}\omega_s^2, \tag{55}$$

which is negative definite, which renders the system input-to-state stable.

We can now summarize the equations of the controller:

$$\dot{\theta} = v_s(\theta) - \omega_s, \tag{56}$$

$$\dot{\omega}_s = -\lambda\omega_s - 2\lambda\mu_1 \left(\boldsymbol{z}_1^T P_1 \frac{\partial \boldsymbol{y}_d}{\partial \theta} + \boldsymbol{z}_2^T P_2 \frac{\partial \boldsymbol{\alpha}_1}{\partial \theta} \right), \tag{57}$$

$$\boldsymbol{u} = M\Big(P_2^{-1} P_1 \boldsymbol{z}_1 - K_d \boldsymbol{z}_2 + (M^{-1}D - K_p)\boldsymbol{x}_2 + \frac{\partial \boldsymbol{\alpha}_1}{\partial \theta} v_s \Big), \tag{58}$$

$$\boldsymbol{z}_1 = \boldsymbol{x}_1 - \boldsymbol{y}_d(\theta), \tag{59}$$

$$\boldsymbol{z}_2 = \boldsymbol{x}_2 - \boldsymbol{\alpha}_1(\theta), \tag{60}$$

$$\boldsymbol{\alpha}_1 = -K_p \boldsymbol{z}_1 + \frac{\partial \boldsymbol{y}_d}{\partial \theta} v_s, \tag{61}$$

$$\frac{\partial \boldsymbol{\alpha}_1}{\partial \theta} = K_p \frac{\partial \boldsymbol{y}_d}{\partial \theta} + \frac{\partial \boldsymbol{y}_d}{\partial \theta}\frac{\partial v_s}{\partial \theta}. \tag{62}$$

P_1 and P_2 can be found by choosing appropriate Q_1, Q_2, $A_1 = -K_p$ and $A_2 = -K_d$, and solve the Lyapunov equation $P_i A_i + A_i^T P_i = -Q_i$ with respect to the coefficients of the diagonal matrix P_i. The output maneuvering system can be divided into three parts as shown in Fig. 2. The first part is the guidance system which outputs information about the path and the speed profile for a given θ. The second part is the output maneuvering controller giving out control signals to the mechanical system forcing it to move along the path, while at the same time updating the guidance system on how far the system has moved along the path. The last part is the mechanical system, in this case, the extruder positioning system.

D. Speed Assignment with speed reduction at sharp corners

A speed assignment function $v_s(\theta)$, defining how fast the controller should move along the path, must be defined. A simple and effective speed assignment function enabling path following with a constant speed m_s is

$$v_{s,cs}(\theta) = \frac{m_s}{\|\boldsymbol{p}_n - \boldsymbol{p}_{n-1}\|_2}. \tag{63}$$

This choice of $v_{s,cs}(\theta)$ may, however, lead to large positional errors when the controller follows paths with sharp corners. To increase the positional accuracy, a good strategy is to reduce the speed. Here, we propose a speed assignment function that reduces the positional error by reducing the target speed only around sharp corners, while keeping a constant target speed m_s everywhere else.

A piece-wise linear path consists of several straight lines. The two lines stretching between $\boldsymbol{p}_{n-1}$, $\boldsymbol{p}_n$ and $\boldsymbol{p}_{n+1}$ create an angle

$$\alpha_n = \frac{180^\circ}{\pi} \cos^{-1}\left(\frac{(\boldsymbol{\Delta}_{n+1})^T \boldsymbol{\Delta}_n}{\|\boldsymbol{\Delta}_{n+1}\|_2 \, \|\boldsymbol{\Delta}_n\|_2} \right), \tag{64}$$

varying between 0° and 180°, where

$$\boldsymbol{\Delta}_n = \boldsymbol{p}_n - \boldsymbol{p}_{n-1}, \qquad n = 1, \ldots, N. \tag{65}$$

By dividing the area from 0° and 180° into three regions, by defining a lower and upper angle $\alpha_a, \alpha_b \in [0^\circ, 180^\circ]$, we can categorize the different corners into three classes:

Non-sharp: No need to slow down : $0^\circ \leq \alpha_n < \alpha_a$,
Sharp: Reduce speed with increasing angle : $\alpha_a \leq \alpha_n \leq \alpha_b$,
Very Sharp: Heavy speed reduction : $\alpha_b < \alpha_n \leq 180^\circ$.

Fig. 3 shows the division into regions based on α_a and α_b. Now, define the weighting/interpolation value k_n for $n = 1, \ldots, N-1$ according to

$$k_n = \begin{cases} 0, & 0^\circ \leq \alpha_n < \alpha_a, \\ \dfrac{\alpha_n - \alpha_a}{\alpha_b - \alpha_a}, & \alpha_a \leq \alpha_n \leq \alpha_b, \\ 1, & \alpha_b < \alpha_n \leq 180^\circ. \end{cases} \tag{66}$$

In addition, define $k_0 = 1$ and $k_N = 1$, so that the speed assignment begins and ends close to zero.

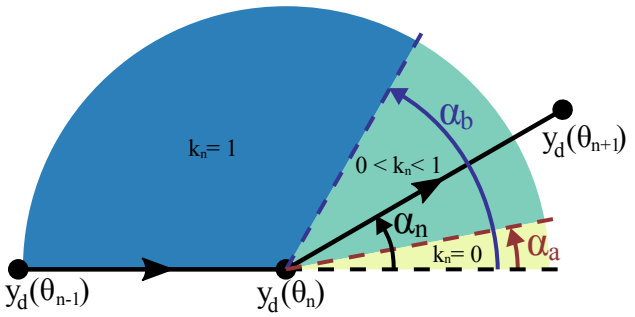


Fig. 3: Illustration showing the angle between two consecutive line segments, division into different regions based on α_a and α_b and the value of k_n for different α_n.

We are now ready to define the speed assignment with speed reduction at sharp corners,

$$v_s(\theta) = \begin{cases} \frac{m_s}{\|\boldsymbol{\Delta}_n\|_2}\left(k_{n-1}h(\theta-\theta_{n-1})+(1-k_{n-1})h\left(\frac{1}{2}\right)\right), \\ \qquad \theta \in \left[\theta_{n-1}, \frac{\theta_n+\theta_{n-1}}{2}\right), \\ \frac{m_s}{\|\boldsymbol{\Delta}_n\|_2}\left(k_n h(\theta_n-\theta)+(1-k_n)h\left(\frac{1}{2}\right)\right), \\ \qquad \theta \in \left[\frac{\theta_n+\theta_{n-1}}{2}, \theta_n\right), \end{cases} \tag{67}$$

where

$$h(\theta) = \frac{1}{\pi}\tan^{-1}\left(\frac{\theta-a_1}{a_2}\right)+\frac{1}{2}, \tag{68}$$

is a sigmoid-shaped function that goes from 0 to 1 with increasing θ. The parameter a_1 moves the transition area in the θ direction, while a_2 changes the shape of, or scales, the transition area.

The partial derivative of the speed assignment with respect to the path variable then becomes

$$\frac{\partial v_s}{\partial \theta}(\theta) = \begin{cases} \frac{m_s}{\|\boldsymbol{\Delta}_n\|_2}\frac{k_{n-1}a_2}{a_2^2+(\theta-\theta_{n-1}-a_1)^2}, \\ \qquad \theta \in \left[\theta_{n-1}, \frac{\theta_n+\theta_{n-1}}{2}\right), \\ \frac{m_s}{\|\boldsymbol{\Delta}_n\|_2}\frac{-k_n a_2}{a_2^2+(\theta_n-\theta-a_1)^2}, \\ \qquad \theta \in \left[\frac{\theta_n+\theta_{n-1}}{2}, \theta_n\right). \end{cases} \tag{69}$$

IV. Simulation

A. Parameters

The parameters in Tab. I were used for the Cartesian 3D printer model (16, 17, 18, 20). The values does not represent a real 3D printer system, but are used as an example. Hence, some aspects of this model could be unrealistic. The parameters of the controller are given in Tab. II.

B. Scenarios

The 3D model used for simulation can be seen in Fig. 5. The model was specifically created with both sharp corners and rounded curves of varying eccentricity. The model given in the Standard tessellation language format (STL) was sliced using the open-source slicing software Slic3r [9], with waypoints extracted using libslic3r. For comparison of the behavior, three different controller scenarios were investigated: a constant value speed profile, the sigmoidal speed profile presented by Skjetne et al. [5], and the speed reduction at sharp corners presented in III-D.

TABLE I: Parameters of the Cartesian 3D printer model.

Parameter	Value	Unit
m_N	0.4	kg
m_X	0.2	kg
m_{PS}	0.8	kg
J_m	$5.0 \cdot 10^{-6}$	kgm^2
B	$1.1 \cdot 10^{-4}$	$\text{kgm}^2\text{s}^{-1}\text{rad}^{-1}$
R_a	$1 \cdot 10^{-3}$	Ω
K_m	$1 \cdot 10^{-4}$	Nm/A
r	0.01	m

TABLE II: Controller and speed profile parameters.

Parameter	Value	Parameter	Value
K_p	$500I$	μ	1000
K_d	$1667I$	α_a	5°
Q_1	I	α_b	20°
Q_2	I	a_1	0.05
λ	30	a_2	0.005

C. Simulation Result

Fig. 6 shows the result of path-following along ten 0.4 mm layers of the test model. Fig. 7, 8 and 9 shows the cross-track error, along-track error, speed profile and extruder speed of the first layer in Fig. 6 for each of the three speed profile scenarios, respectively. A comparison of max value and integral of absolute error (IAE) of the cross-track error is shown in Tab. III in addition to the total time spent to print one layer.

The along-track error s and cross-track error e, as seen in Fig. 4, are calculated as follows with φ the angle between the print surface and the path coordinate systems.

$$\begin{bmatrix} s \\ e \end{bmatrix} = \begin{bmatrix} \cos\varphi & \sin\varphi & 0 \\ -\sin\varphi & \cos\varphi & 0 \end{bmatrix} \boldsymbol{z}_1 \tag{70}$$

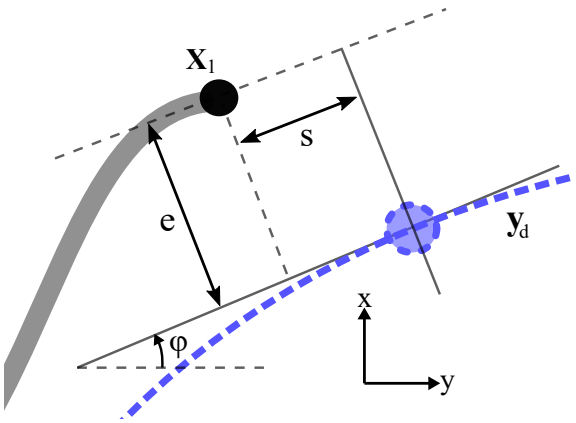
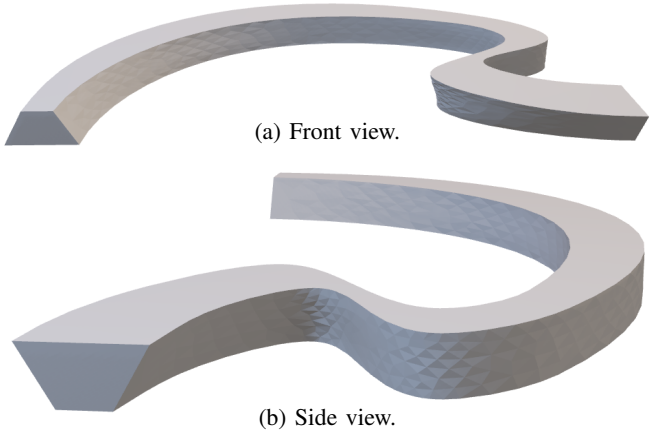


Fig. 4: Cross-track and along-track error.



(a) Front view.

(b) Side view.

Fig. 5: Visualization of the STL model used in the simulation.

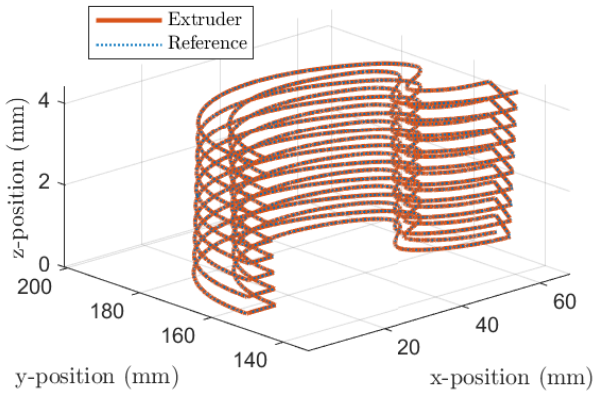
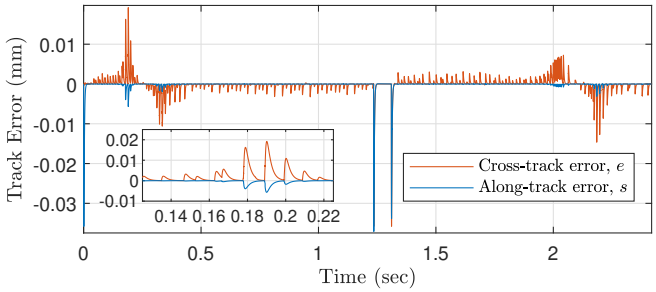


Fig. 6: Simulation of path-following along ten 0.4 mm layers of the test model shown in Fig. 5.

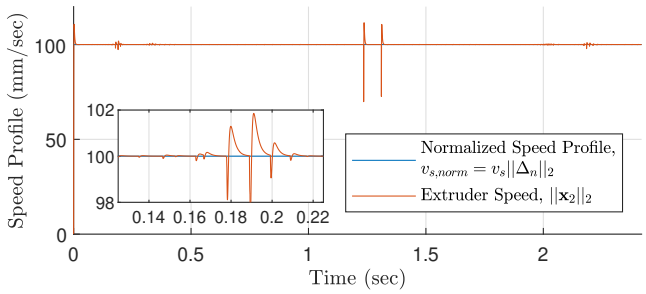
V. Discussion and Further Work

Comparing the cross-track and along-track error for each of the three speed profile scenarios, we see that the error in position can be reduced by using a speed assignment that slows down at the transition from one line segment to the next. In Fig. 7 we see that the controller with constant speed assignment manages to keep a constant speed m_s and quickly drives the positional error to zero after switching from one line segment to another. However, the greater the angle between the line segments, the greater the positional error will be when transitioning from one line segment to the next. This positional error will also increase with increased speed. Looking at the spikes in the middle of Fig. 7b, we see the negative effect of the two 90 degree turns at the left end of the model in Fig. 5a on the positional error. By reducing the speed locally around the line segment transitions as is done in the last two speed profile scenarios, the error in position can be reduced.

Looking at Fig. 8 we see the simulation results of the controller with the speed assignment from [5] that reduces the speed at the start and end of each line segment. Looking at Tab. III or comparing Fig. 8a and Fig. 7a, we see that the speed profile from [5] reduces the overall magnitude of the cross-track error significantly compared to the one



(a) Error



(b) Speed

Fig. 7: Constant speed scenario with $m_s = 0.1$ m/s.

TABLE III: Comparison of cross-track error for the three different speed profile scenarios considered.

Speed profile scenario	$\|\cdot\|_\infty$	IAE	Total time [s]
Constant speed	$1.92 \cdot 10^{-2}$	$1.82 \cdot 10^{-3}$	2.42
Reduction every transition	$5.80 \cdot 10^{-4}$	$5.76 \cdot 10^{-5}$	6.11
Reduction sharp corners	$9.54 \cdot 10^{-3}$	$1.26 \cdot 10^{-3}$	2.57

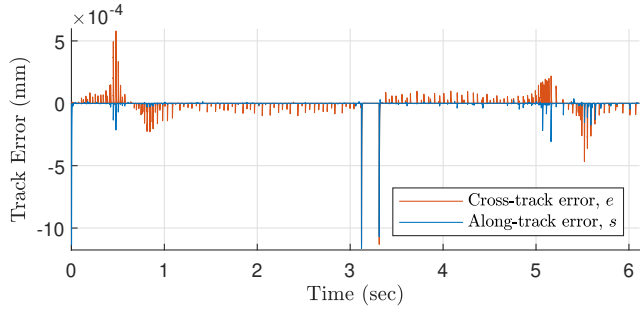
with constant speed. However, this start and stop scheme is significantly slower, increasing the time it takes to complete the path from 2.42 seconds to 6.11 seconds.

Finally, Fig. 9 shows the simulation results of using the controller with the speed assignment presented in Section III-D, where the speed at the line segment transitions is only reduced if the angle is greater than some α_a, in this case $\alpha_a = 5°$. Compared to Fig. 7a we see an overall reduction in the cross-track and along-track error and a huge reduction in the positional error at the two 90 degree turns. Also, note that the time it takes to complete the path has not increased substantially when compared to the time it took with a constant speed assignment.

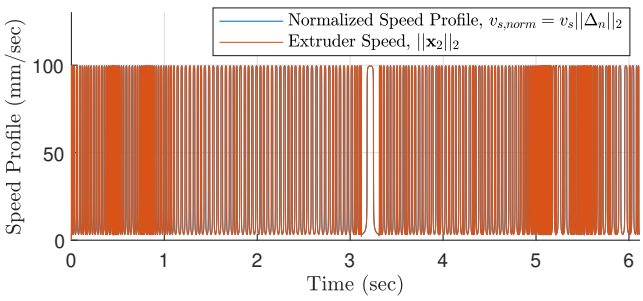
This shows, that the proposed speed assignment has the potential to make a good trade-off between positional accuracy when following the path and the total time it takes to complete the path. Making it an interesting technique for positioning control of the extruder in 3D printing.

A. Some caveats

A problem with the proposed speed profile and the way the path have been parameterized with $\theta_i - \theta_{i-1} = 1, i = 0, \ldots, N+1$ is that the length of the transition area between line segments changes with the length of the line segment. This results in short line segments having fast transients while long line segments having slow transients. This can

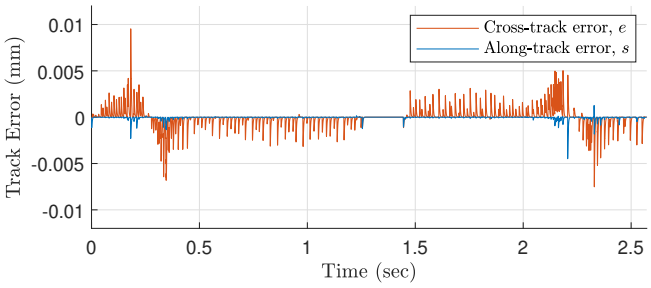


(a) Error

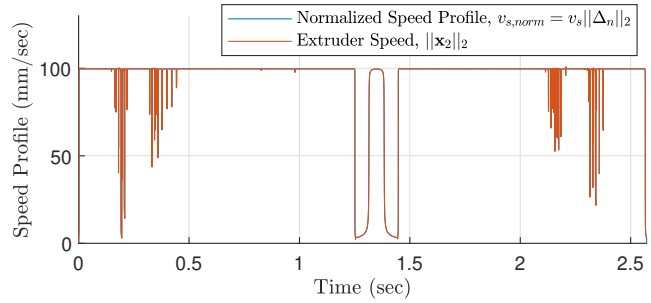


(b) Speed

Fig. 8: Scenario with speed reduction at every line segment transition.



(a) Error



(b) Speed

Fig. 9: Results of scenario with speed reduction at sharp corners. $\alpha_a = 5°, \alpha_b = 20°$.

be seen in Fig. 9b, looking at the speed reduction in the middle of the diagram.

Another problem with this parameterization is that if we were to discretize with a fixed step method we must ensure that the step length is small enough relative to the shortest line segment divided by the speed. If not, the steps in θ conducted by the controller may exceed one, and thereby, possibly skipping one or more line segments in the path, resulting in the wrong path being followed. A possible strategy to avoid these problems and make the overall method more robust is to parameterize the path based on the length of the path instead of having $\theta_i - \theta_{i-1} = 1, i = 0, \ldots, N+1$ for each segment. Another solution is to preprocess the path, removing short lines and/or making the line segments somewhat the same length. Path parameterization based on path length will be an area for further investigation.

One drawback of the output maneuvering controller presented by Skjetne et al. [5] is that the errors $\boldsymbol{z} = z_1, z_2, z_3$ to be reduced is along the x, y and z axes of the print surface and not relative to the movement along the path. In other words, in the Lyapunov design, we do not look at the cross-track and along-track error directly. This makes it impossible to tune the cross-track error and along-track error independently. Lyapunov design with a focus on a path relative frame will be a topic for further investigation.

VI. Conclusion

This article demonstrates the output maneuvering controller for path following in a Cartesian 3D printer. A novel speed assignment based on the relative angle between subsequent line segments of the print was presented. The speed adjustment maintains a constant speed when there is little difference in the current to the new angle, but can slow down completely when a sharp 90-degree corner occurs. The method demonstrated a smaller cross-track error than when a constant speed profile was used and not as much of a slow down as when a stop-and-go profile was used.

VII. Acknowledgements

The work reported in this paper was based on activities within centre for research based innovation SFI Manufacturing in Norway, and is partially funded by the Research Council of Norway under contract number 237900.

References

[1] A. P. Aguiar, J. P. Hespanha, and P. V. Kokotović, "Path-following for nonminimum phase systems removes performance limitations", *IEEE Trans. Automat. Contr.*, vol. 50, no. 2, pp. 234-239, Feb. 2005.

[2] A. P. Aguiar, J. P. Hespanha, and P. V. Kokotović, "Performance limitations in reference tracking and path following for nonlinear systems", *Automatica*, vol. 44, no. 3, pp. 589-610, Mar. 2008.

[3] T. Faulwasser, B. Kern, and R. Findeisen, "Model predictive path-following for constrained nonlinear systems", in *48th IEEE Conference on Decision and Control (CDC)*, Dec. 2009, pp. 8642–8647.

[4] T. Faulwasser and R. Findeisen, "Nonlinear Model Predictive Control for Constrained Output Path Following", *IEEE Transactions on Automatic Control*, vol. 61, no. 4, pp. 1026–1039, Apr. 2016.

[5] R. Skjetne, T. I. Fossen, and P. V. Kokotović, "Robust output maneuvering for a class of nonlinear system", *Automatica*, vol. 40, no. 3, pp. 373–383, Mar. 2004.

[6] R. Skjetne, T. I. Fossen, and P. V. Kokotović, "Adaptive maneuvering, with experiments, for a model ship in a marine control laboratory", *Automatica*, vol. 41, no. 2, pp. 289-298, 2005.

[7] M. W. Spong and S. Hutchinson and M. Vidyasagar, *Robot Modeling and Control*, Wiley, Hoboken, NJ, 2006.

[8] Prusa Research by Josef Prusa, "The Original Prusa I3 MK3S 3D Printer", Accessed on: Nov. 20, 2020. [Online]. Available: https://www.prusa3d.com/original-prusa-i3-mk3/

[9] Alessandro Ranellucci, "Slic3r", Accessed on: Nov. 20, 2020. [Online]. Available: https://github.com/slic3r