2019 International Conference on Robotics and Automation (ICRA)
Palais des congres de Montreal, Montreal, Canada, May 20-24, 2019

Velocity Constrained Trajectory Generation for a Collinear Mecanum Wheeled Robot*

Matthew T. Watson[1], Daniel T. Gladwin[2], Tony J. Prescott[3], and Sebastian O. Conran[4]

***Abstract*— While much research has been conducted into the generation of smooth trajectories for underactuated unstable aerial vehicles such as quadrotors, less attention has been paid to the application of the same techniques to ground based omnidirectional dynamically balancing robots. These systems have more control authority over their linear accelerations than aerial vehicles, meaning trajectory smoothness is less of a critical design parameter. However, when operating in indoor environments these systems must often adhere to relatively low velocity constraints, resulting in very conservative trajectories when enforced using existing trajectory optimisation methods.**

This paper makes two contributions; this gap is bridged by the extension of these existing methods to create a fast velocity constrained trajectory planner, with trajectory timing characteristics derived from the optimal minimum-time solution of a simplified acceleration and velocity constrained model. Next, a differentially flat model of an omnidirectional balancing robot utilizing a collinear Mecanum drive is derived, which is used to allow an experimental prototype of this configuration to smoothly follow these velocity constrained trajectories.

I. Introduction

The ability for dynamically stable omnidirectional mobile robots to plan dynamically feasible trajectories through complex environments is a key precursor to their successful development and commercialisation. This is a more challenging task than the simple kinematic planning that is commonly used to generate trajectories for statically-stable holonomic wheeled vehicles, as the underactuation of balancing robots means they cannot be commanded to follow arbitrary trajectories through configuration space. The generated trajectory must therefore meet the desired navigation goals and constraints, whilst also remaining strictly within the set of dynamically feasible trajectories.

Optimal dynamically feasible constrained trajectories for underactuated nonlinear systems can be found by optimising a set of discrete input changes over a finite horizon, integrating an approximation of the nonlinear system dynamics to predict the system response to these inputs for the evaluation of a suitable objective function, solved as a nonlinear program. These techniques have been applied to complex dynamic systems such as that in this paper with solution times in the region of hundreds of milliseconds for simple trajectories, however, complexity and solve time rapidly increases for lengthier trajectories [1], limiting the suitability of these methods for real-time planning. This problem can also be approached by extension of the existing kinematic planners commonly used in ground vehicle trajectory planning to include the system dynamics, referred to as kinodynamic planning, with kinodynamic RRT* being a popular choice [2] [3]. This method builds a random tree of trajectories in the system's configuration space, rooted at the system's initial state. By integrating the nonlinear system dynamics between nodes all resulting trajectories through the tree are guaranteed to be dynamically feasible. However, this too is a computationally expensive method, taking seconds to minutes to plan simple trajectories.

A less computationally demanding alternative applies the concept of differential flatness to the planning problem, a model reduction technique that allows for state trajectories to be calculated algebraically from sufficiently continuously differentiable geometric trajectories in some possibly fictitious system outputs [4]. This allows the planning problem to be addressed in a top down manner by optimising trajectories in the system's outputs rather than its inputs, yielding less computationally demanding problem formulations. This research has mainly focused on quadrotors, which are naturally differentially flat [5] and therefore well suited to this type of planning, though some research has been undertaken into applying these techniques to dynamically stable omnidirectional ground robots of the ball balancing variety [6], [7], in which smooth trajectories between position waypoints are generated for a single planar direction of a ballbot. Another approach to trajectory generation for ball-balancing robots optimises trajectories in the lean angle state using trajectories comprised of piecewise summed parametrised hyperbolic secant and cubic spline functions [8]. However, this choice of basis function requires the solution of a complex constrained nonlinear program, again at high computational cost.

In this article we extend existing differential flatness based trajectory planning techniques to a vehicle utilizing a Collinear Mecanum Drive (CMD), shown in Figure 1. This wheel configuration operates in a similar manner to a two wheeled inverted pendulum, but instead utilizes three or more collinear Mecanum wheels to enable translation parallel to the wheel axis whilst simultaneously balancing, allowing for omnidirectional locomotion. By dynamically

*This work was supported by Consequential Robotics Ltd (CqR) and the Engineering and Physical Sciences Research Council [grant number EP/M508135/1]

[1]Matthew T. Watson is a PhD candidate at the Department of Electrical and Electronic Engineering, The University of Sheffield, Sheffield, UK m.t.watson@sheffield.ac.uk

[2]Daniel T. Gladwin is a Senior Lecturer at the Department of Electrical and Electronic Engineering, The University of Sheffield, Sheffield, UK d.gladwin@sheffield.ac.uk

[3]Tony J. Prescott is Professor of Cognitive Robotics at the Department of Computer Science, The University of Sheffield, Sheffield, UK, and co-founder of Consequential Robotics Ltd., London, UK t.j.prescott@sheffield.ac.uk

[4]Sebastian O. Conran is co-founder of Consequential Robotics Ltd., London, UK sconran@consequentialrobotics.com

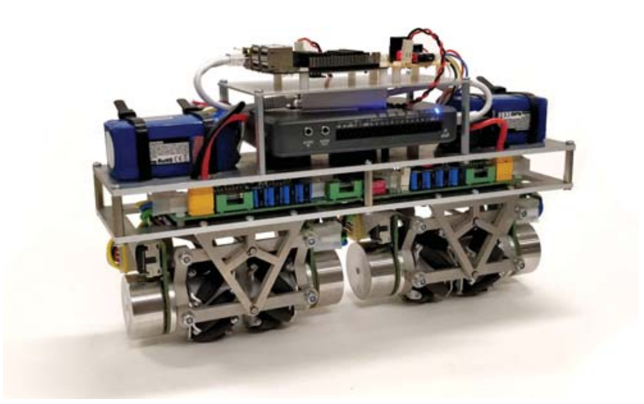

Fig. 1. Collinear Mecanum drive prototype platform [9]. This utilizes four torque controlled Mecanum wheels for locomotion, suspended in pairs to maintain traction. Sensing is provided by wheel encoders and triaxial gyroscopes and accelerometers.

balancing this drive mechanism can be used to create robots with the same height and ground footprint as a human, whilst possessing a minimum directly navigable gap that is only limited by its wheel diameter. This makes this drive mechanism an ideal candidate for robots that must physically interact with a standing human, whilst possessing a discrete ground footprint to maintain manoeuvrability and human-like dimensions.

II. Differentially Flat Model Derivation

The dynamic model of a collinear Mecanum drive [9] can be described in the inertial frame in the form

$$M(\zeta)\ddot{\zeta} + C(\zeta,\dot{\zeta})\dot{\zeta} + G(\zeta) + F(\zeta)\dot{\zeta} = H(\zeta)\tau \qquad (1)$$

with $\zeta = \begin{bmatrix} x & y & \phi & \theta_p \end{bmatrix}^T$, where x, y are the Cartesian positions of the platform base, ϕ is the rotation of the robot about the vertical, and θ_p the lean angle. $M(\zeta)$ represents the inertia matrix, $C(\zeta,\dot{\zeta})$ the Coriolis and centripetal matrix, $G(\zeta)$ the gravity matrix, $F(\zeta)$ the viscous friction matrix, $H(\zeta)$ the input matrix, and τ a vector of four wheel torque inputs.

Velocities in the body attached frame can be defined by a rotation of $\begin{bmatrix} \dot{x} & \dot{y} \end{bmatrix}^T$ by ϕ, giving the mapping

$$v = \begin{bmatrix} v_x \\ v_y \\ \dot{\phi} \\ \dot{\theta}_p \end{bmatrix} = \begin{bmatrix} R_\phi & 0 \\ 0 & I_{2\times 2} \end{bmatrix} \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\phi} \\ \dot{\theta}_p \end{bmatrix} = R_B(\zeta)\dot{\zeta} \qquad (2)$$

This can be substituted into (1) to give

$$M(\zeta)\left(\dot{R}_B^T(\zeta)v + R_B^T(\zeta)\dot{v}\right) + N(\zeta, v) = H(\zeta)\tau \qquad (3)$$

where $N(\zeta,v) = C\left(\zeta, R_B^T(\zeta)v\right)R_B^T(\zeta)v + G(\zeta) + F(\zeta)\dot{\zeta}$.

The dependency of $H(\zeta)$ on ζ can be removed by multiplication of the dynamics by $R_B(\zeta)^T$. Treating (3) as a set of four simultaneous equations, expression 2 of (3) can be multiplied through by r_w and summed with expression 4 to isolate the system's internal dynamics by elimination of τ, yielding an expression of the form $f\left(\theta_p, \dot{\theta}_p, \ddot{\theta}_p, v_x, \dot{v}_y, \dot{\phi}\right) = 0$. This is, however, still a differential equation that must be integrated with specified initial conditions to determine θ_p, and is therefore not differentially flat.

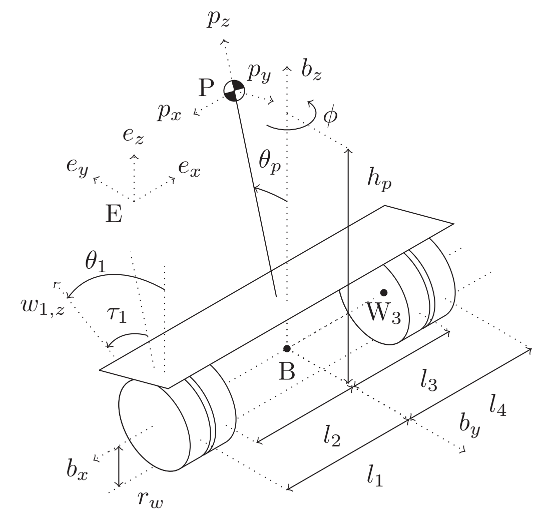

Fig. 2. Collinear Mecanum Drive coordinates and parameters

Shomin *et al.* [7] showed that a single planar axis of a ballbot can be differentially flattened by a combination of model simplification and a flat output of the form $S(t) = x(t) + \lambda\theta_p(t)$. This can be extended to the full 3D model and to include yaw rotation by defining the system's flat outputs as:

$$S_1(t) = x(t) - \sin(S_3(t))\lambda\theta_p(t) \qquad (4)$$

$$S_2(t) = y(t) + \cos(S_3(t))\lambda\theta_p(t) \qquad (5)$$

$$S_3(t) = \phi(t) \qquad (6)$$

The system states x, y, v_x, v_y, and all derivatives of ϕ can be trivially derived by differentiation and rotation of the flat outputs. Substituting these definitions into the internal dynamics and performing a small angles approximation of $\sin(\theta_p)$ and $\cos(\theta_p)$ about $\theta_p = 0$ yields the expression

$$0 = \ddot{\theta}_p(t)\left(b + c\lambda\right) + \theta_p(t)\left(d + e\dot{S}_3(t)^2 + c\lambda\dot{S}_3(t)^2\right) + a\theta_p(t)\dot{\theta}_p(t)^2 + c\sin(S_3(t))\ddot{S}_1(t) + c\cos(S_3(t))\ddot{S}_2(t) \qquad (7)$$

where a, b, c, d, and e are known constants. $\ddot{\theta}_p$ can be eliminated from this expression by selection of $\lambda = -{}^{b}/_{c}$, and it is a safe assumption that the centripetal force acting on θ_p due to $\dot{\theta}_p$ will always be small relative to other forces, so the term $a\theta_p\dot{\theta}_p^2$ can be omitted. This allows for solution of $\theta_p(t)$ as

$$\theta_p(t) = \frac{-c\left(\sin(S_3(t))\ddot{S}_1(t) + \cos(S_3(t))\ddot{S}_2(t)\right)}{d + e\dot{S}_3(t)^2 - b\dot{S}_3(t)^2} \qquad (8)$$

$\dot{\theta}_p$ and $\ddot{\theta}_p$ can then be determined by differentiation of θ_p, thus algebraically defining all system states in terms of the flat outputs S. Finally, the input τ can be derived by inversion of the system dynamics in (3) by multiplication with $H(\zeta)^+$

and substitution with system states in terms of S. This method of differential flattening introduces a singularity at $d + e\dot{S}_3(t)^2 - b\dot{S}_3(t)^2 = 0$, which for the prototype in this article occurs at $\dot{\phi} = \pm 8.78\,\mathrm{rad\,s^{-1}}$, meaning an angular velocity constraint must be observed on $\dot{\phi}$.

III. Trajectory Optimisation

With a differentially flat model, it is now possible to derive corresponding state and input trajectories for any continuous trajectory in the flat output that has a bounded 4th derivative, the snap of the output. Work applying differentially flat trajectory generation to quadrotors [5], [10] has shown that desirable smooth trajectories can be generated by minimising the nth derivative of a system of order $n-1$, so here the 5th derivative, or crackle, is to be minimised.

As in these prior works, S is defined by three univariate polynomials. In order to have a bounded fourth derivative a 5th degree polynomial is required. Additionally, to perform meaningful translations between positions it must be possible to constrain the start and end of the polynomial, requiring a further 4 degrees of freedom, so nonic polynomials are used. Each polynomial can only define a smooth path from one position waypoint to another, so n_s polynomial segments of durations $\Delta t \in \mathbb{R}^{n_s}_{>0}$ are chained together to define complex trajectories through $n_s + 1$ waypoints. This gives the cost function for a single flat output defined by a piecewise continuous chain of polynomials $p(t)$ as

$$J = \sum_{i=1}^{n_s} \int_0^{\Delta t_i} \left(\frac{d^5 p_i(t)}{dp_i(t)^5} \right)^2 dt \tag{9}$$

This can be arranged in the quadratic form $J = p^T H(\Delta t) p$, where p is a vector containing the concatenated coefficients of n_s chained polynomials of durations Δt.

Linear equality constraints are used to ensure continuity of the zeroth to fifth derivatives at the boundary between polynomials, as well as to enforce position constraints at waypoints $w_i \in \mathbb{R}$, $i = [0 \mathrel{..} n_s]$. The first to fifth derivatives are also constrained at the start and end of the entire trajectory, allowing a new trajectory to be generated that smoothly evolves from the system's current state, and ensuring the system can be made to come to rest at the end of the trajectory.

$$p_i^{(n)}(\Delta t_i) = p_{i+1}^{(n)}(0), \quad i = [1 \mathrel{..} n_s - 1],\ n = [0 \mathrel{..} 5] \tag{10}$$

$$p_i(0) = w_{i-1} \qquad i = [2 \mathrel{..} n_s] \tag{11}$$

$$p_1^{(n)}(0) = w_0^{(n)}, \quad p_{n_s}^{(n)}(\Delta t_{n_s}) = w_{n_s}^{(n)}, \quad n = [0 \mathrel{..} 5] \tag{12}$$

This defines the required polynomial optimisation with a quadratic cost and linear equality constraints, allowing the problem to be formulated as a quadratic program (QP). As only equality constraints are present this can be solved very efficiently using QR decomposition methods with sub-millisecond execution.

Figure 3 shows the optimised polynomial trajectory for a translation of $S_2 = [0, 1]$ over 1.5 s, along with the y and

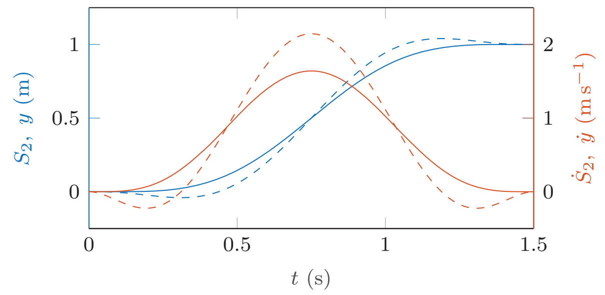

Fig. 3. Zeroth (blue) and first derivatives (red) of the flat output S_2 and $\dot{S}_2$ (solid) and corresponding state trajectories y and $\dot{y}$ (dashed) for a trajectory of 1.5 s duration with $\phi = 0$ between waypoints at $S_2 = [0, 1]$.

$\dot{y}$ state trajectories derived from the differentially flattened model. This shows how the differentially flattened model successfully captures the need for the robot to first move away from the objective in order to lean the pendulum towards it, despite the flat output being monotonic. Systems that move in this manner are referred to as *shape accelerated* systems.

In order to optimally select Δt a convex minimisation can be performed on the sum of the costs of the three flat outputs with coefficients optimised for a given Δt and a weighted sum of the total trajectory duration, using equality constraints to maintain coherence of position waypoints between flat outputs

$$\min_{\Delta t} \left\{ \sum_{i=1}^{n_s} \sum_{j=1}^{3} \int_0^{\Delta t_i} \left(\frac{d^5 p_{ij}(t)}{dp_{ij}(t)^5} \right)^2 dt + K_t \Delta t_i \right\} \tag{13}$$

The scaling factor K_t can be viewed as an analogue for the 'aggressiveness' of a trajectory, producing similarly aggressive trajectories for a given K_t regardless of waypoint number or position.

IV. Constraining Segment Velocity

In optimising a simple rest-to-rest trajectory for minimum crackle, the resulting velocity trajectory forms a bell-shaped profile, in which the average velocity is much smaller than the peak, as visible in Figure 3. In real-world scenarios a platform of this type will have to adhere to velocity constraints for safety, meaning the peak of this bell curve must lie within constraint bounds. Using the above method, K_t must be decreased to lengthen the duration of the trajectory until all peak velocities lie within constraint bounds, potentially decreasing the peak velocity of some segments much below the constraint in order to ensure constraint satisfaction of the segment with greatest peak velocity. Alternatively, these velocity constraints can be represented by smooth barrier penalty functions, allowing the optimal selection of segment durations so that all segments that were constraint violating have their peak velocity reduced to equal the constraint, at a cost of greatly increased solution difficulty. However, both of these methods yield far from time-optimal trajectories, as the resulting velocity profiles between waypoints will only equal

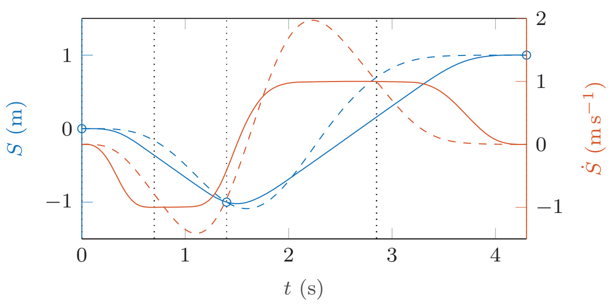

Fig. 4. Flat output zeroth (blue) and first derivatives (red) for position waypoints $S_2 = [0, -1, 1]$, with each segment split into three at its center, with middle segment durations of zero and velocity constraints of $\overline{v} = 1\,\mathrm{m\,s^{-1}}$ at segment boundaries. Dashed lines represent the standard QP solution, solid lines the SOS constrained solution, dotted lines segment boundaries, and blue circles position waypoints.

the constraint at at most a single point, with the majority of the trajectory being much slower than the constraints allow.

An ideal crackle-to-time optimal velocity constrained point to point trajectory would reach $\overline{v}$ as quickly as the crackle-to-time cost ratio allows, maintain $\overline{v}$ for an appropriate amount of time, and decelerate as quickly as the crackle-to-time cost ratio allows. This shape of trajectory can be incorporated by splitting all point-to-point polynomials into three separate polynomial segments, with no intermediary absolute position constraints. The first and last segments in each set of three are kept the same degree as the original polynomial, but by defining the middle segment using a cubic polynomial velocity constraint satisfaction of this segment can be trivially ensured for all t using just two inequalities at the segment start and end. The same continuity of derivatives between segments is enforced as in section III.

Figure 4 shows the QP solution to a trajectory through position waypoints at $S_2 = [0, -1, 1]$, in which each position to position segment has been split into three, with velocity inequalities constraints of $|v| \leq 1$ enforced at all segment boundaries. The duration of the middle segment of each split polynomial is left at zero for demonstration; a systematic method of selecting this new variable is introduced later. Despite the QP solution satisfying constraints at segment boundaries, large violations occur in the middle of two of the segments, making the standard QP approach insufficient to guarantee velocity constraint satisfaction across the whole trajectory.

V. Enforcing Velocity Constraints on Entire Trajectories

Enforcing velocity constraints over the whole trajectory using the original QP problem formulation requires a discrete set of points along the segment to be chosen at which the constraint is enforced using linear inequalities. This offers no guarantee of constraint satisfaction for the continuous trajectory, as with insufficient sample points the constraint may still be violated between constraints. This can be tackled by increasing the density of sampling, but with an associated increase in problem complexity and therefore solve time. This can alternatively be performed in a recursive manner [11], using root finding to insert constraints at the peak violations of the previous QP solution, at a risk of a large increase in solve time if a large number of iterations are required to fully constrain a segment.

An alternative presented here utilizes sum-of-squares (SOS) programming to enforce constraints directly on the continuous time polynomials themselves, rather than at discrete sampled points. A polynomial $p(t)$ of degree $2d$ that is a sum-of-squares polynomial can be written in the Gram matrix form $z(t)^T H z(t)$, where $z(t)$ is a column vector containing the monomials of $p(t)$ up to degree d, and H is positive semidefinite. If a univariate polynomial $p(t)$ can be represented in this form, then $p(t) \geq 0 \;\forall\; t \in \mathbb{R}$ [12].

Furthermore, the constraint $p(t) \geq 0$ can be enforced on just the interval $t = [a, b]$ if $p(t)$ can be written in the form

$$p(t) = \begin{cases} s(t) + (t-a)(b-t)q(t), \; \textit{if} \; \deg(p) \; \textit{is even} \\ (t-a)s(t) + (b-t)q(t), \; \textit{if} \; \deg(p) \; \textit{is odd} \end{cases} \tag{14}$$

where $s(t)$ and $q(t)$ are SOS. For even $\deg(p)$, $\deg(p) = 2d$, $\deg(s) \leq 2d$, and $\deg(q) \leq 2d-2$. For odd $\deg(p)$, $\deg(p) = 2d+1$, $\deg(s) \leq 2d$, $\deg(q) \leq 2d$ [13].

These constraints cannot be used to directly enforce $-\overline{v} \leq v(t) \leq \overline{v}$, as doing so would require both $\overline{v} + s(t)$ and $\overline{v} - s(t)$ to be SOS, an infeasible problem. This cannot be circumvented by using only an appropriately signed one-sided inequality, as complex trajectories can result in violations of both $v(t) \leq \overline{v}$ and $-\overline{v} \leq v(t)$ within the same segment.

We therefore opt to instead enforce velocity monotonicity of each segment, forcing the velocity extrema to occur at the segment boundaries, where they can be constrained by fixed linear inequalities. This is achieved by forcing the second derivative of the flat output in each nonic segment to be of constant sign for the duration of the segment, by defining the second derivatives of all nonic segments in the form $ts(t) + (\Delta t_i - t)q(t)$, with the first and zeroth derivatives obtained by integration and introduction of initial velocity and position coefficients. The original QP is then reformulated as a semidefinite program (SDP), allowing positive semidefinite constraints $H \succeq 0$ and therefore $\ddot{p}(t) \geq 0$ or $-\ddot{p}(t) \geq 0$ to be explicitly included as constraints for all nonic polynomials, thus enforcing the monotonicity of $\dot{p}(t)$. The desired sign for the SOS constraint must be determined *a priori*. This reformulated problem can then be efficiently solved by existing SDP solvers. This allows the constraining of the velocity of the entire trajectory for all t using four SOS constraints and one linear inequality per nonic segment, and one linear inequality per cubic segment.

Figure 4 shows a comparison between the original constrained QP and new SDP solutions. In contrast to the QP solution, the new trajectory now demonstrates exact constraint satisfaction. This problem formulation also generates a more desirable trajectory profile, exhibiting a more direct path with zero overshoot. Solving the SDP for this example using MOSEK V8.1 [14] with preprocessing by YALMIP [15] takes 60 ms using an Intel i7-4720HQ processor, sufficiently fast for use in online real-time planning.

VI. Optimisation of Velocity Constrained Segment Durations

Any method of including velocity constraints prevents a fast analytical solution using QR decomposition as is performed with standard QP approaches. This in turn prevents the usual selection of optimal segment durations by gradient descent as in (13), as the number of function evaluations and therefore QP solutions required to compute numerical gradients for three flat outputs at each iteration - whilst also maintaining time coherence between position waypoints - quickly makes this problem intractable in real-time. Also, with the addition of these velocity constraints the optimisation is now only feasible for a reduced set of segment durations, resulting in loss of convexity. An alternative method of selecting segment durations is therefore required. Shomin [7] uses a fast heuristic method to specify point-to-point polynomial segment durations based on a velocity constrained trapezoidal profile with fixed known start and end velocities, but doesn't provide a systematic method for choosing these velocities. This prevents the use of a similar heuristic in this application, unless the robot is expected to come to rest at every waypoint. We build on this concept, instead opting to select segment durations by optimising three acceleration and velocity constrained trapezoidal profiles for minimum time, from which the optimal durations can be used to define the full dynamically feasible SOS constrained optimisation. This also provides a convenient method for defining the sign of the SOS constraints, determined by examining the sign of the resulting optimal acceleration. This new problem is formulated by simplifying all nonic polynomial segments to second order polynomials and all cubic segments to linear polynomials, optimising the convex nonlinear objective function

$$\min_{\Delta t} \sum_{i=1}^{3} \sum_{j=1}^{n_s} \left(\Delta t_{i,j} + K_a \int_0^{\Delta t_{i,j}} \ddot{p}_{i,j}^2 dt \right) \tag{15}$$
$$s.t. \quad |\ddot{p}_{i,j}| \leq \overline{a}, \quad |\dot{p}_{i,j}(0)| \leq \overline{v}, \quad 0 \leq \Delta t_{i,j}$$

where K_a is a sufficiently small constant such that acceleration is minimised without any meaningful increase in the optimal total duration in order to ensure a unique solution. Equality constraints are enforced between flat outputs for all position waypoints in order to maintain spatial coherence between trajectories, along with the same equalities as in (10) to (12). Solution of this NLP is performed using MATLAB's `fmincon` function, though superior solvers exist that would be expected to deliver reduced solution time. Figure 5 shows the optimal trapezoidal profile generated for a trajectory through 11 random waypoints in S_2, along with the resulting SOS constrained fully dynamically feasible trajectory. Often a number of segments will be optimised to zero duration, allowing them to be removed to simplify the SDP. Figure 6 shows solver cold-start execution time for both the trapezoidal and SDP optimisations with increasing numbers of

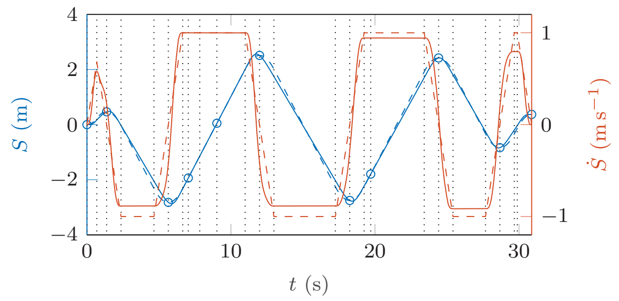

Fig. 5. The optimised minimum-time trapezoidal profile (dashed) through 11 random position waypoints (blue markers), with the corresponding fully dynamically feasible SDP optimised velocity constrained trajectory (solid).

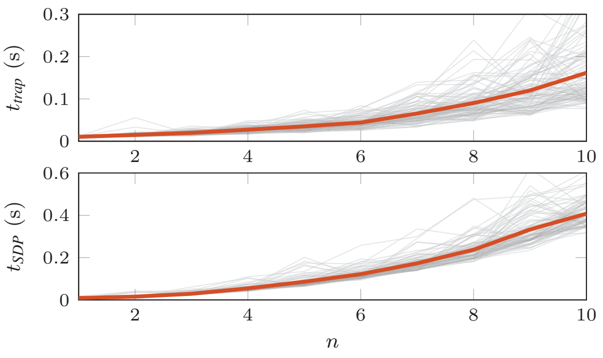

Fig. 6. Trapezoidal and SDP optimisation solver cold-start execution time for 100 random trajectories of increasing position waypoint number n, with mean execution time in red.

position waypoints, demonstrating suitability of this method for online replanning. Also, in a replanning scenario there is likely to be minimal difference between consecutive iterations of the planner, meaning each optimisation can be initialised with the solution to the previous, further reducing solve time.

VII. Experimental Trajectory Tracking

Asymptotic trajectory tracking is achieved in the presence of disturbance, unmodelled dynamics, and parameter uncertainty using a full state feedback time varying LQR. This is derived from a linearisation of the model dynamics about the stationary upright position and time varying ϕ, with $Q = \mathtt{diag}\left(\begin{bmatrix}3 & 3 & 1 & 1 & 0.1 & 0.1 & 0.01 & 0\end{bmatrix}\right)$ and $R = 0.5 I_{4\times4}$. Localisation in the inertial frame is achieved by dead reckoning, using an extended Kalman filter to fuse odometry and inertial data. This provides suitable accuracy and negligible drift for the experiments demonstrated here, but would require an absolute position reference for longer distance navigation tasks. Three example trajectories are demonstrated. Figure 7 shows the robot performing a 2 m translation from a starting pose of $\phi = 0$ to a terminal pose of $\phi = \pi$, with a 0.4 m section placed in the middle of the path in which a yaw angle of $\phi = \pi/2$ is enforced on entry and exit in order for the robot to navigate a gap that is too narrow to be navigated without rotating. Velocity constraints of $\overline{v} = 1\,\mathrm{m\,s^{-1}}$ and $\overline{\dot{\phi}} = 6\,\mathrm{rad\,s^{-1}}$ are enforced, yielding an

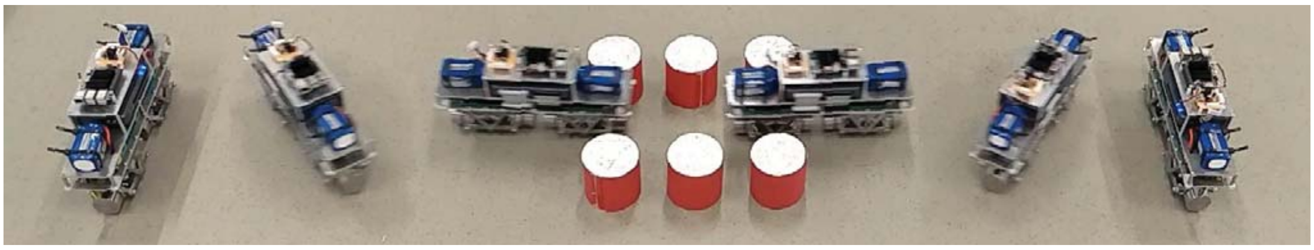

Fig. 7. A $2\,\mathrm{m}$ trajectory through a narrow gap in $2.9\,\mathrm{s}$ with $\overline{v} = 1\,\mathrm{m\,s^{-1}}$, demonstrating the real-world manoeuvrability of this drive configuration.

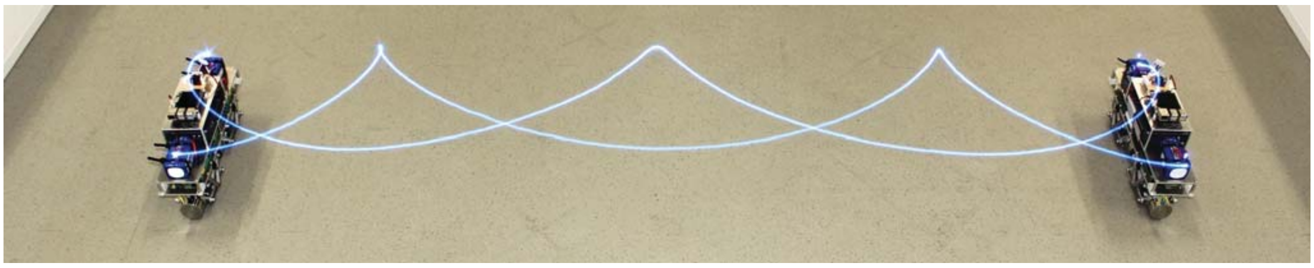

Fig. 8. A $2\,\mathrm{m}$ translation with two full rotations in $3.2\,\mathrm{s}$, demonstrating the smoothness of transition between shape accelerated and lateral motion.

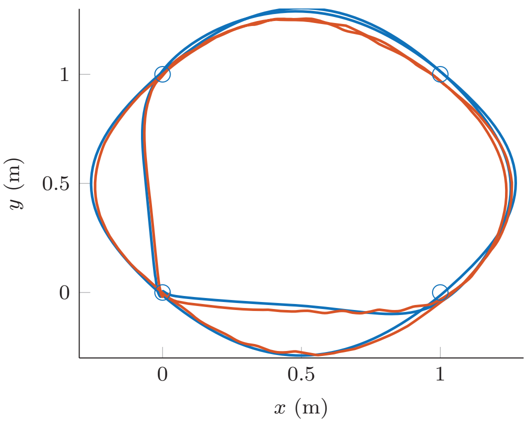

Fig. 9. A Cartesian state trajectory (blue) planned to pass through the four points of a unit square twice, whilst also completing four full rotations about the yaw axis in $10\,\mathrm{s}$, along with the experimental tracked trajectory (red).

optimal trajectory of $2.9\,\mathrm{s}$ duration in a combined solve time of $190\,\mathrm{ms}$. This demonstrates how this approach to trajectory planning and tracking yields smooth transitions between the robot's two modes of locomotion, allowing it to exploit its narrow width to improve environment accessibility. Figure 8 aims to further demonstrate the smoothness of transition between shape-accelerated and lateral movement. To do so a trajectory is again planned for a translation of $2\,\mathrm{m}$, however now a full two revolutions are simultaneously performed, for a terminal pose of $\phi = 4\pi$. For the same constraints as above this yields a trajectory of $3.2\,\mathrm{s}$ duration. Two LED markers are attached to each end of the robot, and a long exposure shot is used to capture the motion of the robot through time. This demonstrates the smoothness of the generated trajectories, and the accuracy of tracking that results from directly deriving dynamically feasible trajectories.

Finally, Figure 9 shows a Cartesian state trajectory (blue) of $10\,\mathrm{s}$ duration passing through the four points of a unit square twice, along with the actual tracked trajectory (red). The optimised trajectory follows an intuitive path, with each intermediary waypoint passed at the maximal allowed velocity, and with smooth changes in direction between waypoints. The trajectory is tracked well, with RMS errors of $0.010\,\mathrm{m}$ and $0.031\,\mathrm{m}$, sampled at $100\,\mathrm{Hz}$.

VIII. Conclusion

This paper has made two distinct contributions. First, it is shown that fully omnidirectional motion of a Collinear Mecanum Drive can be be described by three differentially flat outputs with bounded fourth derivatives, allowing the generation of smooth dynamically feasible trajectories between arbitrary sets of waypoints. Second, a novel approach to the generation of velocity constrained polynomial trajectories has been demonstrated, using sum-of-squares programming to guarantee constraint satisfaction for the entire continuous time trajectory. The combination of these two contributions allows for omnidirectional balancing robots to generate and follow trajectories through an arbitrary set of waypoints with velocity constraints in much closer to minimum time than existing polynomial trajectory generation methods, making this method well suited to the planning of fast but safe indoor trajectories.

Future work will aim to incorporate full localisation into the platform to allow for the navigation of a complex map by autonomous selection of suitable waypoints. Recent advances in SOS programming will also be explored for use in this planning problem, notably sparse SOS [16] and diagonal/scaled-diagonal SOS programming [17]. These are SOS representations that allow the simplification of the SDP into a SOCP or QP, greatly reducing problem complexity.

References

[1] D. Pardo, L. Moller, M. Neunert, A. W. Winkler, and J. Buchli, "Evaluating Direct Transcription and Nonlinear Optimization Methods for Robot Motion Planning," *IEEE Robotics and Automation Letters*, vol. 1, no. 2, pp. 946–953, jul 2016.

[2] S. Karaman and E. Frazzoli, "Optimal kinodynamic motion planning using incremental sampling-based methods," in *Proceedings of the IEEE Conference on Decision and Control*. IEEE, dec 2010, pp. 7681–7687.

[3] D. J. Webb and J. Van Den Berg, "Kinodynamic RRT*: Asymptotically optimal motion planning for robots with linear dynamics," in *Proceedings - IEEE International Conference on Robotics and Automation*. IEEE, may 2013, pp. 5054–5061.

[4] M. Fliess, J. Levine, P. Martin, and P. Rouchon, "Flatness and defect of non-linear systems: Introductory theory and examples," *International Journal of Control*, vol. 61, no. 6, pp. 1327–1361, jun 1995.

[5] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *Proceedings - IEEE International Conference on Robotics and Automation*. IEEE, may 2011, pp. 2520–2525.

[6] M. Shomin and R. Hollis, "Differentially flat trajectory generation for a dynamically stable mobile robot," in *Proceedings - IEEE International Conference on Robotics and Automation*. IEEE, may 2013, pp. 4467–4472.

[7] M. Shomin and R. Hollis, "Fast, dynamic trajectory planning for a dynamically stable mobile robot," in *IEEE International Conference on Intelligent Robots and Systems*, 2014, pp. 3636–3641.

[8] U. Nagarajan, G. Kantor, and R. L. Hollis, "Trajectory planning and control of an underactuated dynamically stable single spherical wheeled mobile robot," in *Proceedings - IEEE International Conference on Robotics and Automation*, 2009, pp. 3743–3748.

[9] M. T. Watson, D. T. Gladwin, T. J. Prescott, and S. O. Conran, "Design and control of a novel omnidirectional dynamically balancing platform for remote inspection of confined and cluttered environments," in *2018 IEEE International Conference on Industrial Electronics for Sustainable Energy Systems (IESES)*. IEEE, jan 2018, pp. 473–478.

[10] C. Richter, A. Bry, and N. Roy, "Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments," in *Springer Tracts in Advanced Robotics*, vol. 114, 2016, pp. 649–666.

[11] J. Chen, T. Liu, and S. Shen, "Online generation of collision-free trajectories for quadrotor flight in unknown cluttered environments," in *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2016-June. IEEE, may 2016, pp. 1476–1483.

[12] S. Prajna, A. Papachristodoulou, P. Seiler, and P. Parrilo, "New developments in sum of squares optimization and SOSTOOLS," in *Proceedings of the 2004 American Control Conference*. IEEE, 2004, pp. 5606–5611 vol.6.

[13] G. Polya and G. Szego, *Problems and theorems in analysis*. Springer, 1998.

[14] M. ApS, *The MOSEK optimization toolbox for MATLAB manual. Version 8.1.*, 2017.

[15] J. Lofberg, "YALMIP : a toolbox for modeling and optimization in MATLAB," in *2004 IEEE International Conference on Robotics and Automation (IEEE Cat. No.04CH37508)*. IEEE, 2004, pp. 284–289.

[16] Y. Zheng, G. Fantuzzi, and A. Papachristodoulou, "Sparse sum-of-squares (SOS) optimization: A bridge between DSOS/SDSOS and SOS optimization for sparse polynomials," *ArXiv e-prints*, Jul. 2018.

[17] A. A. Ahmadi and A. Majumdar, "DSOS and SDSOS optimization: LP and SOCP-based alternatives to sum of squares optimization," in *2014 48th Annual Conference on Information Sciences and Systems, CISS 2014*. IEEE, mar 2014, pp. 1–5.