Trajectory Generation for Underactuated Control of a Suspended Mass

Jarvis Schultz
Department of Mechanical Engineering
Northwestern University
Evanston, IL 60208
jschultz@u.northwestern.edu

Todd Murphey
Department of Mechanical Engineering
Northwestern University
Evanston, IL 60208
t-murphey@northwestern.edu

***Abstract*— The underactuated system under consideration is a magnetically-suspended, differential drive robot with a winch system articulating a suspended mass. A dynamic model of the system is first constructed, and then a nonlinear, infinite-dimensional optimization algorithm is presented. The Lagrangian mechanics based system model uses the principles of kinematic reduction to produce a mixed kinematic-dynamic model that isolates the modeling of the system actuators from the modeling of the rest of the system. In this framework, the inputs become generalized velocities instead of generalized forces facilitating real-world implementation in an embedded system. The optimization algorithm automatically deals with the complexities introduced by the nonlinear dynamics and underactuation to synthesize dynamically feasible system trajectories for a wide array of trajectory generation problems. Applying this algorithm to the mixed kinematic-dynamic model, several example problems are solved and the results are tested experimentally. The experimental results agree quite well with the theoretical showing promise in extending the capabilities of the system to utilize more advanced feedback techniques and to handle more complex, three-dimensional problems.**

I. INTRODUCTION

Applications involving cable-driven actuation of suspended payloads include material transportation in factories, control of inspection equipment in hangars, stroke patient rehabilitation, and robotic actuation of marionettes. The systems used in these applications are often extremely complex, and due to their nonlinear dynamics and underactuated nature, deriving effective control strategies is quite difficult. However, intelligently designed controls that utilize the inherent dynamics of the system allow the generation of extremely complex trajectories that span large workspaces. Automating the design of these controls will allow these systems to be used to their full potential. This paper develops a unique dynamic model that facilitates real-world implementation in conjunction with an infinite-dimensional optimization technique to automatically generate feasible controls for a wide array of trajectory generation problems; this algorithm is applied to two example problems, and the results are implemented experimentally.

In the field of robotics, many problems in cable actuated control have been studied. These problems include gantry and crane systems, cooperative control and multiple cable systems, and aerial conveyance of payloads [11], [4], [15]. The underactuated system studied in [2] and [3] consists of a single winch fixed in space that uses self-excitation to increase its available workspace. The authors developed an algorithm that was used to generate controls for several different trajectory generation problems. These problems include a point-to-point transfer of a load where only the initial and final states of the system matter, and a path tracing problem. The optimization technique utilized in this work is capable of generating admissible system trajectories for both of these situations; they will be used as example problems in the coming sections.

In this paper a nonlinear, projection based optimization technique capable of generating controls for a wide range of trajectories is applied to a robotic system utilizing stringed actuation of a suspended mass. This optimization algorithm relies on a dynamic model of the system. To fill this role, a mixed kinematic-dynamic model that facilitates real-world system implementation is developed. With the optimization utilizing this model, it is possible to automatically and robustly determine and utilize a set of controls for an arbitrary trajectory generation problem. As part of our collaboration with Disney Research, the robotic system that has been developed for experimental validation can also be used for performing automated marionette plays [10], [14]. The techniques posed in this paper will likely play a fundamental role in the execution of these plays in the future.

II. DYNAMIC MODELING

The present system consists of a differential drive mobile robot magnetically attached to an elevated planar surface such that it drives on the underside of the surface. The robot has an additional motor that acts as a winch for controlling the length of a string which has its other end attached to a mass. A schematic of this system is shown in Fig. 1. The robot utilizes a velocity-based motor controller motivating the mixed kinematic-dynamic model presented in this section. In this model, the kinematic inputs for the system are the same quantities utilized by the robot when controlling the motors. This eases experimental implementation of the results obtained by the optimization.

In our dynamic model, we assume that the inputs to the system are sufficiently powerful to be treated as kinematic inputs [7]; i.e., we assume that the inputs are capable of perfectly following an arbitrary prescribed trajectory regardless

This work is sponsored in part by the National Science Foundation and Disney Research.

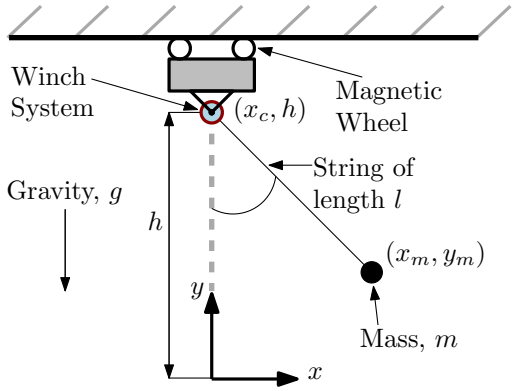


Fig. 1: Schematic of mechanical system including relevant geometric parameters.

of the magnitude of the forces and torques required to do so. Implicit in this modeling choice is the assumption that the frequency content of the actuator dynamics has little impact on the behavior of the rest of the system. By also assuming that the strings are massless, we can then use the principle of kinematic reduction to generate a mixed kinematic-dynamic model of the system [1]. The inputs to the system control the horizontal position of the robot x_c, and the length of the string l. Therefore we define the kinematic configuration variables as $q_K = [x_c \;\; l]^{\mathrm{T}}$. The dynamic configuration variables are then defined as $q_D = [x_m \;\; y_m]^{\mathrm{T}}$. When we assume that the mass at the end of the string is a point mass the Lagrangian for this system is then

$$L(q_D, \dot{q}_D) = \frac{1}{2} m \left(\dot{x}_m^2 + \dot{y}_m^2\right) - m g y_m \,. \tag{1}$$

The coupling between the kinematic inputs and the dynamic mass is accomplished through the use of a constraint. Initially, possible loss of tension in the string is ignored, and the string is considered to be a rigid wire between the robot and the mass. The constraint that this wire imposes on the system is given by

$$\phi(q) = (x_m - x_c)^2 + (y_m - h)^2 - l^2 = 0 \tag{2}$$

where we have implicitly used the definition of q as $q = [q_D \;\; q_K]^{\mathrm{T}}$. To account for the fact that the terms in this equation are allowed to vary with time we differentiate (2) with respect to time and rewrite it as a Pfaffian constraint. We then rearrage the constraint to separate the kinematic and dynamic portions

$$A(q)\dot{q} = [A_D(q) \;\; A_k(q)] \begin{bmatrix} \dot{q}_D \\ \dot{q}_K \end{bmatrix} = 0 \tag{3}$$

where $A_D(q) = \partial\phi(q)/\partial q_D$ and $A_K(q) = \partial\phi(q)/\partial q_K$. To complete the system's dynamic model, we use the familiar Euler-Lagrange equations along with the Lagrange-d'Alhembert principle to arrive at

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_D}\right) - \frac{\partial L}{\partial q_D} = \lambda \, A_D^{\mathrm{T}}(q) \,. \tag{4}$$

This equation has introduced the Lagrange multiplier λ as a new unknown term leaving the system of equations underdefined. We differentiate Eq. (3) with respect to time to provide an additional equation

$$A_D \ddot{q}_D + \dot{A}_D \dot{q}_D + A_K \ddot{q}_K + \dot{A}_K \dot{q}_K = 0 \,. \tag{5}$$

Finally we define the inputs to the system as

$$u = \ddot{q}_K = \begin{bmatrix} \ddot{x}_c \\ \ddot{l} \end{bmatrix} . \tag{6}$$

Using Eqns. (4), (5), and (6), we can algebraically solve for the second derivatives of the dynamic configuration variables and the Lagrange multiplier. After this rearranging, we are left with a set of equations of the form

$$\ddot{q}_D = \begin{bmatrix} g_1(q_D, q_K, \dot{q}_D, \dot{q}_K, u) \\ g_2(q_D, q_K, \dot{q}_D, \dot{q}_K, u) \end{bmatrix} . \tag{7}$$

To facilitate numerical integration and our optimal control algorithm we would like the above to be in the form $\dot{x} = f(x,u)$. Thus, we define the state to be

$$x = \begin{bmatrix} q_D \\ q_K \\ \dot{q}_D \\ \dot{q}_K \end{bmatrix} = \begin{bmatrix} x_m & y_m & x_c & l & \dot{x}_m & \dot{y}_m & \dot{x}_c & \dot{l} \end{bmatrix}^{\mathrm{T}} . \tag{8}$$

Then Eq. (7) becomes

$$\dot{x} = f(x,u) = \begin{bmatrix} x_5 & x_6 & x_7 & x_8 & g_1 & g_2 & u_1 & u_2 \end{bmatrix}^{\mathrm{T}} . \tag{9}$$

While this final system has a higher dimensional state than it would if generalized forces were the system inputs, it provides a nice division between the dynamics of the system and the dynamics of the actuators. This is especially convenient from a practical implementation viewpoint. In many real engineering systems, such as those controlled by DC motors, it is often much more convenient to control velocities and accelerations than it is to control forces.

III. OPTIMIZATION ALGORITHM

For applications involving cable-actuated payloads, it is often the case that it is desirable to specify some trajectory for the payload to follow while being unconcerned with the trajectory that the controlling mechanism follows e.g. [2], [3], [4]. In this case, it is possible that the desired trajectory is not dynamically feasible. Additionally, due to the nonlinear dynamics and underactuated nature of the system, it is not obvious how to determine the inputs to generate a payload trajectory that follows the prescribed trajectory. In this section we briefly describe a trajectory optimization technique that automatically addresses both of these issues simultaneously (more detail can be found in [5], [6], [8]).

This optimization routine is particularly well suited to handling the present system for several reasons. First, it is fully capable of handling an underactuated or uncontrollable system. Additionally, the projection operator (to be discussed later in this Section) provides a stabilizing feedback controller about a feasible system trajectory even if the system

is locally or globally unstable. The mathematical formalism behind the algorithm poses the problem in an infinite-dimensional vector space. Thus in the statement of the problem the system is left in its natural, infinite-dimensional representation. Numerical approximations happen at the solution level rather that the problem statement level. This is advantageous when a system contains dynamics in a range of time scales. More traditional optimization techniques, such as the discrete forms of model-predictive control or dynamic programming, require an *a priori*, constant discretization of the time horizon. When the frequency of the dynamics vary over a large range, this discretization will either be too coarse for high-frequency dynamics (reducing accuracy), or too fine for low-frequency dynamics (reducing computational efficiency). The present optimization algorithm allows the use of powerful, adaptive time stepping integration algorithms to automatically deal with this situation.

A. Problem Setting

We begin by defining the desired trajectory as $\xi_d = (x_d(\cdot), u_d(\cdot))$ over the time horizon $[t_0, t_f]$ where the pair $(x_d(\cdot), u_d(\cdot))$ may or may not satisfy the system dynamics. We are interested in solving the following constrained optimization problem:

$$\begin{aligned} &\text{minimize} \qquad J(\xi) = \int_{t_0}^{t_f} l\,(\tau, x(\tau), u(\tau))\, d\tau + m(x(t_f)) \\ &\text{subject to} \qquad \dot{x}(t) = f(x(t), u(t)), \quad x(0) = x_0 \end{aligned} \tag{10}$$

with $\xi = (x(\cdot), u(\cdot))$. The integrand and terminal cost that we utilize are given by

$$\begin{aligned} l(\tau, x(\tau), u(\tau)) &= (x(\tau) - x_d(\tau))^{\mathrm{T}} Q\,(x(\tau) - x_d(\tau)) \\ &\quad + (u(\tau) - u_d(\tau))^{\mathrm{T}} R\,(u(\tau) - u_d(\tau)) \end{aligned} \tag{11a}$$

$$m(x(t_f)) = \big(x(t_f) - x_d(t_f)\big)^{\mathrm{T}} P_1 \big(x(t_f) - x_d(t_f)\big) \tag{11b}$$

where Q, R, and P_1 are positive definite weighting matrices.

We would like to solve the optimization problem in Eq. (10) using a standard iterative descent technique such as gradient descent or Newton's method [9]. However, if we found a descent direction and added it to the current trajectory, the resultant pair of curves would no longer satisfy the system dynamics. Thus, we introduce a projection operator that maps arbitrary trajectories to trajectories satisfying the system dynamics. Using this operator, we can effectively define an equivalent unconstrained optimization problem.

B. Projection Operator

To formally discuss the projection operator, we begin by defining the trajectory manifold for the dynamic system as $\mathcal{T}$. If the pair $\eta = (x(\cdot), u(\cdot))$ obey $\dot{x} = f(x,u)$, they lie on the trajectory manifold i.e. $\eta \in \mathcal{T}$. Given an arbitrary pair $\xi = (\alpha(t), \mu(t))$, we define the projection operator $\mathcal{P}$ as the mapping

$$\mathcal{P}: \;\; \xi = (\alpha, \mu) \;\; \mapsto \;\; \eta = (x, u)\,. \tag{12}$$

Presuming that this operator exists we can convert the original constrained optimization problem in Eq. (10) to an equivalent unconstrained optimization problem i.e.

$$\min_{\xi \in \mathcal{T}} J(\xi) = \min_{\xi} J(\mathcal{P}(\xi))\,. \tag{13}$$

This optimization problem can be solved using standard descent techniques. Following [5] and [8], the projection operator that we choose is a linear feedback law given by

$$\begin{aligned} x(t_0) &= \alpha(t_0) \\ u(t) &= \mu(t) + K(t)(\alpha(t) - x(t)) \\ \dot{x} &= f(x(t), u(t)) \end{aligned} \tag{14}$$

where $K(t)$ is a time-varying feedback gain matrix that stabilizes the system about the current trajectory. To find $K(t)$, we solve a finite horizon linear quadratic regulator (LQR) problem [12] on a linearized representation of the system, found by linearizing the system about its current trajectory. The projection operator is one of the primary features of this optimization technique; its innate stabilizing properties make this a particularly robust algorithm even when dealing with unstable systems.

C. Descent Direction

Due to its infinite-dimensional structure, finding a descent direction for this optimization problem is slightly different than a traditional finite-dimensional problem. As is usually done, a quadratic model function is used to locally approximate the cost function around the current iteration point. Then we solve the following problem to find the descent direction ζ_i:

$$\zeta_i = \arg\min_{\zeta} \; DJ(\mathcal{P}(\xi_i)) \circ \zeta + \frac{1}{2} q \circ (\zeta, \zeta)\,. \tag{15}$$

The bilinear operator [17] $q \circ (\cdot, \cdot)$ can be chosen to give different descent algorithms. If q is identity the descent direction corresponds to a gradient descent algorithm, and if $q = D^2 J(\mathcal{P}(\xi_i))$ the solution to (15) corresponds to a Newton's method descent direction.

To solve the problem in Eq. (15), we convert the unconstrained optimization problem into a constrained optimization problem by restricting $\zeta \in \mathrm{T}_{\xi_i}\mathcal{T}$ i.e. ζ must lie in the set of trajectories of the linearization of $\dot{x} = f(x,u)$ about $\xi_i \in \mathcal{T}$. This set $\mathrm{T}_{\xi_i}\mathcal{T}$ actually forms a Banach space and therefore this constrained optimization can be thought of as a linear optimal control problem that can be solved without the use of iterative search methods. In fact the solution to this problem is found by using a Ricatti transformation to produce a set of two initial value problems (IVP) that can be numerically integrated [5].

D. Optimization Algorithm

The implementation of this optimization is programmed in `Mathematica`. An initial guess is made for the set of inputs $u(t)$ that will follow the desired trajectory. This guess is typically either trivial $u(t) = 0$ or inverse kinematics

based. These inputs are used to numerically integrate Eq. (9) to obtain a feasible trajectory to seed the optimization. If the space is locally non-convex we will be unable to determine a Newton- descent direction, and the algorithm reverts to a gradient descent algorithm utilizing and Armijo line search [9]. Typically several steps of the gradient descent algorithm are required at the beginning of the optimization to get the initial guess near enough to the optimum to begin using the Newton-descent direction. Inability to determine the Newton-descent direction is manifested by the lack of a solution to the aforementioned set of IVPs generated by the Ricatti transform [5]. It was shown in [5] that this technique maintains quadratic when utilizing Newton's method.

Once the optimization has converged to a final answer, we check that the desired solutions inputs are within the actuator limits. Also, we look at the Lagrange multiplier associated with the constraint in Eq. (4) to see if it indicates the string being placed in compression by the ball. Since the string cannot physically apply this type of force, this situation is indicative of the string going slack and the ball entering a free-flight phase. Even though this is physically possible, we avoid this situation so that our optimization algorithm does not need to handle the discontinuous velocities induced by the string regaining tension.

If the solution exceeds actuator limits or the string goes slack we simply adjust the weighting terms in the cost function and use the original solution as the first guess for the adjusted weights. This process repeats until we have a converged solution that is physically realizable. It is important to note that in the face of very strict actuator limits it may be impossible to find a solution that satisfies the actuator limits. Additionally if the desired reference trajectory is highly dynamic, the algorithm may need a reasonable initial guess to make any progress and in the worst case, the desired trajectory may be so far from the system's reachable space that the algorithm will be unable to find a feasible trajectory.

IV. EXPERIMENTAL SETUP

A. Robotic System

The robotic system consists of a number of magnetically-suspended differential drive mobile robots developed as part of our collaboration with Disney Research. Fig. 2 highlights the important components of the system. In this experiment we restrict ourselves to a single robot driving in a straight line so that our problem is two-dimensional. A large, plastic membrane is used for the surface that the robots drive on. To provide rigidity to the thin, flexible plastic, it is held in tension in all directions. A rectangular steel frame is placed around the edges of the plastic membrane and a system of clamps and turnbuckles provide the tension. The entire assembly is then mounted on a support frame such that the plane formed by the plastic sheet is parallel with the ground at a height of approximately eight feet.

Each robot has two 24V DC motors powering large magnetic drive wheels. There are also two passive, magnetic caster wheels providing stability. A completely passive

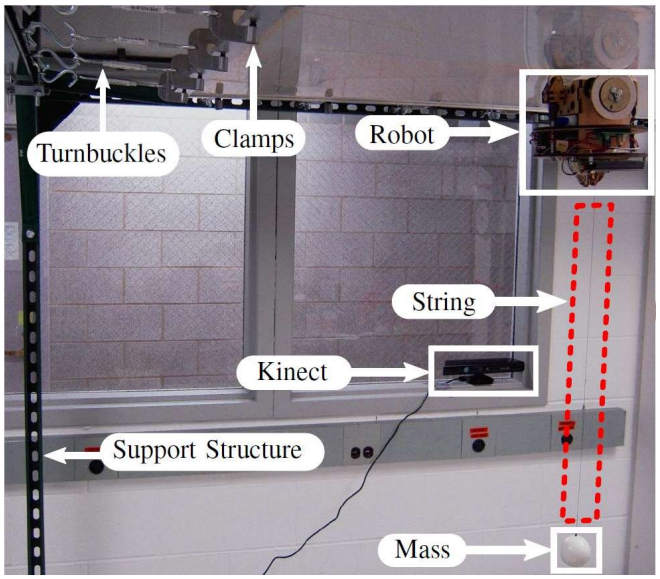


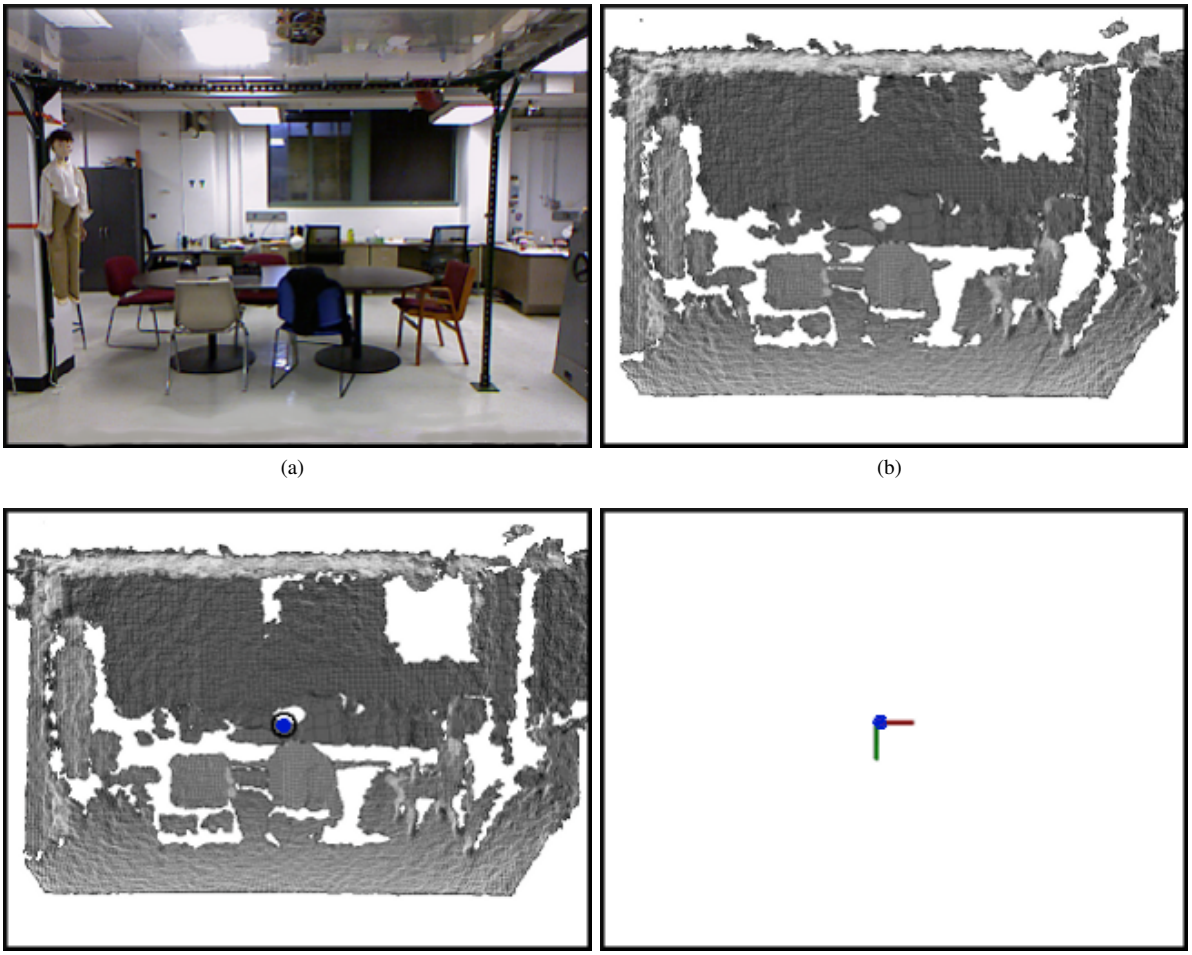
Fig. 2: Image showing the main components of the experimental apparatus.

"idler" mechanism provides a magnetic surface for the robots to cling to. The idler has the same geometric footprint as a robot, but its magnetic fields have opposite polarity. This idler is placed on the top surface of the plastic tarp, and the robots then hang on its underside. The force generated by the magnetic attraction between the robot and the idler is quite high; the robot can support its own weight plus an additional 25 lbs of static load. The robots also have an additional motor and associated string management system acting as a winch for articulating suspended payloads.

Each robot is powered by two 12V lithium iron phosphate batteries. Processing, communication and motor control for each robot is handled by a Microchip PIC® 32-bit microcontroller. Wireless communication is performed using Digi XBee® modules. A central PC performs the optimizations, and then the desired inputs to the system are communicated wirelessly to the robot. The robot uses simple PID controllers to track the desired inputs to the system; optical encoders act as the feedback sensors. A tracking system provides real-time information on the location of the mass (described in the next section), however at this time the information is used only for experimental validation and not for providing feedback to the robot.

B. Motion Tracking

Motion tracking of the mass is handled using a Microsoft Kinect® sensor. The Kinect uses an infra-red structured light array to build a three-dimensional point cloud representation of its view of the world. Every element in the point cloud contains the real-world location of a single infra-red point with respect to the Kinect's local coordinate system. Interface to the Kinect is provided by the NITE™ open source Kinect drivers released by PrimeSense™ [13]. The Robot Operating System (ROS) is used for collecting, recording,



(a) (b) (c) (d)

Fig. 3: Illustration of Kinect filtering algorithm. Image 3a shows a picture of the experimental space taken from the point-of-view of the Kinect. Image 3b shows the point cloud representation of this three dimensional space. After the filtering algorithm is implemented, the Kinect has identified the points located on the mass; these points are shown in blue and a circle has been drawn around them in image 3c. When the mass is identified, it is segmented, and a centroidal value is calculated. The coordinate system in 3d lies on this centroidal value.

and processing the data coming in from the Kinect [16] at approximately 30 Hz.

When the tracking software begins running, the point cloud is immediately segmented to remove the floor, ceiling and all data beyond the far plane of the experimental setup. In the absence of noise this smaller, segmented point cloud will contain only points lying on the suspended mass. However in a real, noisy environment there is a possibility of random, errant data points existing within this volume. Additionally through testing it was discovered that when the ball is in motion, it is possible for the Kinect to detect the string connecting the ball and the robot.

To eliminate the errors introduced by this noise a small "volume of interest" is defined around the centroid of the mass. Using the mass location at the previous two samples and a linearized model of the mass' dynamics a prediction for the current location of the mass is used to determine where the small volume should be placed. If the number of points detected in the volume drops below a threshold value, the volume is expanded, and the mass location process is restarted. When this small volume is properly tracking the mass, we simply find the location of the centroid of all points contained within the volume to determine where the mass is located. This process is illustrated in Fig. 3.

V. RESULTS

A. *Point to Point Trajectory Generation*

In this first example, we analyze a simplified version of the system that is very similar to the one studied in [2] and [3]. We remove the mobile robot's translational degree of freedom so we are essentially left with a stationary winch. The desired trajectory is simply a point in state space at the end of the time horizon. This scenario is representative of the situation where a suspended payload is initally at rest on a support structure. The goal is then to use the underactuated winch system to relocate the payload to some new support

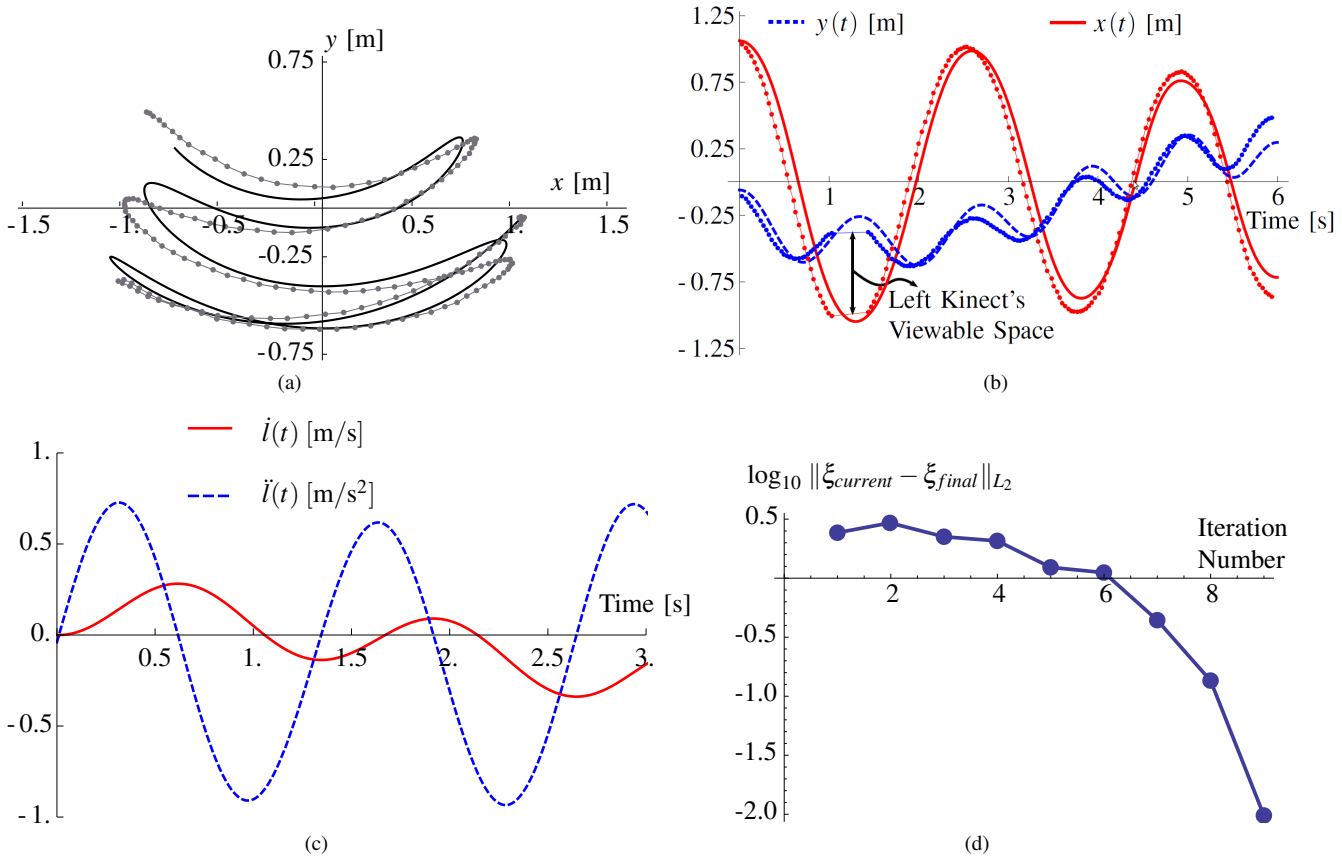


Fig. 4: Sample solution to the point to point problem. 4a shows the optimized trajectory in black and the corresponding set of experimental data points in dotted gray. 4b shows the simulated and experimental dynamic configuration variables throughout the time horizon. 4d shows the quadratic convergence achieved by the optimization algorithm.

structure where the intermediate trajectory followed by the payload is unimportant.

To accomplish this task, we slightly modify the dynamic model presented in Section II by removing the robot's translational degree of freedom. We then set the desired trajectory as the goal state for all time in the arbitrarily chosen time horizon. The weights in the cost function are chosen such that very little weight is placed on the values of the state throughout the time horizon i.e. the terms in Q are small. Several orders of magnitude higher weights are placed on the inputs R, and even greater weights are placed on the final values for the state using the terms in P_1.

Fig. 4 shows an example solution to the optimization and the corresponding experimental data. The mass is 0.124 kg with a starting string length of 1.5 m and an initial angle of 45° from the vertical and zero velocity. In the global coordinate system the robot is fixed at $(x_c, h) = (1, 0)$ m. The final desired location is a string length of 1 m with an angle of −45° from the vertical and zero velocity. The optimization produces a dynamically feasible system trajectory that achieves the desired final state, and the experimental system does a reasonable job of following the simulated trajectory. Note that Fig. 4b points out a zone where the mass left the viewable space of the Kinect and the mass was temporarily lost; the results of this are also evident at the end of the first swinging cycle in the experimental data set of Fig. 4a.

B. Trajectory Tracking

For this problem a desired trajectory for the mass to follow is specified, and the optimization attempts to find an admissible system trajectory $\eta \in \mathcal{T}$ where the mass follows the desired trajectory as closely as possible. This problem is representative of the situation where a cable-suspended payload must follow a specific path and the path that the robot follows is unimportant; e.g., if the payload was a piece of inspection equipment in an automobile manufacturing facility that must be swept over a manufactured surface in search of defects. For the problems analyzed here, the full robotic system described in Section II is utilized.

Fig. 5 shows several example solutions to this problem. The desired trajectories are arbitrary curves chosen simply for illustration purposes. Fig. 5c shows an example of the optimization automatically dealing with the situation where the system's initial conditions do not lie on the desired trajectory. In both plots the experimental results track the optimal admissible trajectories fairly well — especially considering there is no state feedback for the mass being provided to the robotic system. Note that the system inputs in Fig. 5b and 5d

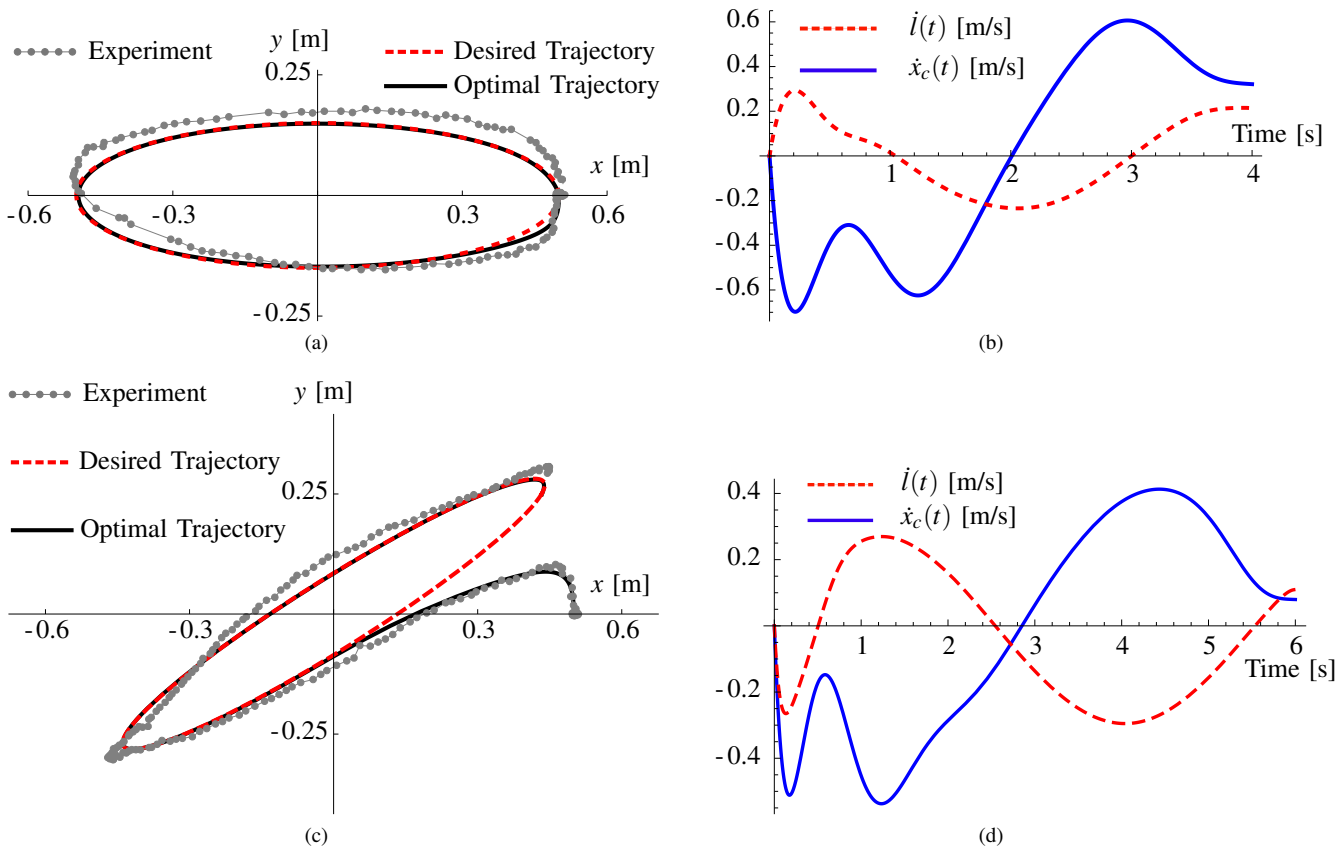


Fig. 5: Sample solutions and experimental trials for the trajectory tracking problem. 5a and 5c show the optimized trajectory in black, the reference trajectory in dashed red, and the experimental results in dotted gray. 5b and 5d show the optimal inputs to the system for the reference trajectories of 5a and 5c respectively.

are highly dynamic even though the desired trajectories are quite smooth and controlled. This nicely illustrates why it is difficult to manually devise control strategies for this underactuated system. Videos of these trajectories being tested experimentally can be seen at the Northwestern University LIMS Laboratory Vimeo page *http://vimeo.com/lims*.

VI. CONCLUSIONS AND FUTURE WORKS

The dynamic model presented in Section II uses kinematic reducibility to facilitate real-world implementations of the system. In an attempt to control this system, we have presented a trajectory generation optimization routine that robustly and automatically solves a variety of problems. This routine was used to solve several example problems and their solutions were implemented on a novel robotic system for verification. In the problems analyzed, both the dynamic model, and the optimization routine behaved very well. The system model produced realistic simulation behavior and the optimization routine reliably achieved quadratic convergence even with very poor initial guesses. It is suspected that the errors between experimental and simulation results are caused by several factors. The simulation does not take into account any losses that occur due to the elasticity of the string, and it assumes that the mass is a point mass when in reality the mass has a finite volume and corresponding mass moment of inertia. It is feasible that improving these aspects of the simulation could increase its accuracy.

The other source of error, likely the far greater one, is the experimental setup itself. The system is essentially being run open-loop in that the controls from the optimization are being sent to the robot regardless of any errors in the system's state that may have accumulated. Additionally it is possible that the tracking mechanism itself is imperfect and could be providing skewed results. The experimental setup is in the early stages of development, and as the work progresses these issues will be addressed.

The system is currently capable of executing complex three-dimensional trajectories, and in future work we intend to expand the capabilities of the software to investigate these types of problems. Based on the open-loop experimental results presented in the previous section, it is expected that if the system were further developed to utilize real-time state feedback, the experimental results would improve significantly. Conveniently, the projection operator from the final optimization iteration provides a state feedback law that stabilizes the system about the optimal admissible trajectory, which can be used for closing the loop. To facilitate the analysis of these more complex problems, we also hope to further develop the optimization technique to increase its

computational efficiency.

VII. ACKNOWLEDGMENTS

This work was sponsored in part by two grants from the National Science Foundation, IIS-0917837 and CCF-0907869. Additional support was provided by Disney Research, and Walt Disney Imagineering.

We would additionally like to thank Lanny Smoot of Disney Research for his contributions to the development of the robotic platform, and John Ware of Northwestern University for his support in the software development.

REFERENCES

[1] F. Bullo, *Geometric Control of Mechanical Systems*. New York: Springer-Verlag, 2004.

[2] D. Cunningham and H. H. Asada, "The winch-bot: A cable-suspended, under-actuated robot utilizing parametric self-excitation," in *IEEE International Conference on Robotics and Automation*, Kobe, Japan, May12–17 2009, pp. 1844–1850.

[3] ——, "Continuous path tracing by a cable-suspended, under-actuated robot: The winch-bot," in *IEEE International Conference on Robotics and Automation*, Anchorage, AK, May3–8 2010, pp. 1255–1260.

[4] J. Fink, N. Michael, S. Kim, and V. Kumar, "Planning and control for cooperative manipulation and transportation with aerial robots," *The International Journal of Robotics Research*, vol. 30, pp. 324–334, Mar. 2011.

[5] J. Hauser, "A projection operator approach to the optimization of trajectory functionals," in *IFAC World Congress*, Barcelona, Spain, July21–26 2002.

[6] J. Hauser and D. G. Meyer, "The trajectory manifold of a nonlinear control system," in *IEEE Conference on Decision and Control*, Tampa, FL, Dec.16–18 1998, pp. 1034–1039.

[7] E. Johnson and T. D. Murphey, "Dynamic modeling and motion planning for marionettes: Rigid bodies articulated by massless strings," in *International Conference on Robotics and Automation*, Roma, Italy, Apr.10–14 2007, pp. 330–335.

[8] E. R. Johnson and T. D. Murphey, "Automated trajectory synthesis from animation data using trajectory optimization," in *IEEE Conference on Automation Science and Engineering*, Bangalore, India, Aug.22-25 2009, pp. 274–279.

[9] C. T. Kelley, *Iterative Methods for Optimization*. Philadelphia, PA: Society for Industrial and Applied Mathematics (SIAM), 1999.

[10] T. D. Murphey and M. Egerstedt, "Choreography for marionettes: Imitation, planning and control," in *IEEE Conference on Intelligent and Robotic Systems: Workshop on Art and Robotics*, San Diego, CA, Nov. 2007.

[11] R. M. Murray, "Trajectory generation for a towed cable system using differential flatness," in *IFAC World Congress*, San Francisco, jul 1996.

[12] D. S. Naidu, *Optimal Control Systems*. Boca Raton, FL: CRC Press, 2003.

[13] (2010) PrimeSense™ NITE Middleware. PrimeSense. [Online]. Available: http://www.primesense.com/?p=515

[14] J. Schultz, T. Caldwell, T. Murphey, and L. Smoot, "Magnetically suspended ambulatory robots supporting automated marionette shows," in *IEEE International Conference on Robotics and Automation*, Saint Paul, MN, May14–18 2012, submitted for publication.

[15] W. Shiang, "Dynamic analysis of the cable array robotic crane," in *IEEE International Conference on Robotics and Automation*, Detroit, MI, 1999, pp. 2495–2500.

[16] (2011) Robot Operating System. Willow Garage. [Online]. Available: http://www.ros.org

[17] E. Zeidler, *Applied Functional Analysis: Applications to Mathematical Physics*. New York: Springer-Verlag, 1995.