Autonomous Suspended Load Operations via Trajectory Optimization and Variational Integrators

Gerardo De La Torre[*], Evangelos Theodorou[†], and Eric N. Johnson[‡]
Georgia Institute of Technology, Atlanta, Georgia, 30332

Abstract

This paper present a real-time implementable trajectory optimization framework for autonomous suspended load operations in outdoor environments. The framework solves the posed optimal control problem with the iteration-based differential dynamic programming (DDP) algorithm. The algorithm utilizes a variational integrator to propagate the modeled system's state configuration and linearize the resulting discrete dynamics. It is shown that the variational integrator is an excellent candidate for real-time implementation since it remains accurate despite relatively large discretization time steps. Therefore, the computational effort of the DDP algorithm can be mitigated through the reduction of discrete time points. The state of the slung load is estimated via an augmentation to the existing navigation system that utilizes only vision-based measurements of the load. Simulation studies and a flight test are presented to demonstrate the effectiveness of the proposed framework.

Nomenclature

$DG(Y)$	derivative of $G(Y)$ with respect to Y
$D_iG(Y_1,\dots,Y_p)$	derivative of $G(Y_1, Y_2\dots, Y_p)$ with respect to its i^{th} argument, Y_i
$G_{Y_i}(Y_1,\dots,Y_p)$	derivative of $G(Y_1, Y_2\dots, Y_p)$ with respect to Y_i
$G_{Y_i,Y_j}(Y_1,\dots,Y_p)$	second order mixed derivative of $G(Y_1, Y_2\dots, Y_p)$ with respect to Y_i and Y_j

1 Introduction

Suspended load operations have long been studied for manned and unmanned systems. During the 1970s, assisting pilots by actively damping the carried load through specialized hardware [1–3] or feedback control [4–7] was the main focus of suspended load operations research. Recently, extensive piloted flight tests for a task-tailored control system with fuselage and cable angle/rate feedback has shown a significant reduction of load set-down time and a large improvement in average handling qualities [8, 9]. The development of agile indoor quadrotor systems has spurred the advancement of autonomous methods for suspended load operations. Geometric control methods [10, 11], a coupled adaptive feedback and dynamic programming approach [12–14], and reinforcement learning techniques [15, 16] have been proposed for indoor suspended load operations. However, it should be noted these methods have only been tested on highly agile, indoor quadrotors with the assistance of a precise position tracker, typically VICON, and, in some cases, extensive offboard computing. In addition, some of the research may not be applicable to larger platforms that are far less agile, such as the GTMax, or for use in outdoor and uncontrolled environments considered in this paper. For comparison, the GTMax, a modified Yamaha RMAX, weights 64 kg and is powered with a

[*]Graduate Research, School of Aerospace Engineering, 270 Ferst Drive Atlanta GA 30332-0150.
[†]Assistant Professor, School of Aerospace Engineering, 270 Ferst Drive Atlanta GA 30332-0150.
[‡]Associate Professor, School of Aerospace Engineering, 270 Ferst Drive Atlanta GA 30332-0150.

gasoline engine while the quadrotor used in Reference [15], a AscTec Hummingbird, weights 0.71 kg and uses electric motors.

There has been limited work on autonomous outdoor suspended load operations. Since precise measurements from a VICON system are not available, a GPS aided inertial navigation system is typically used to estimate the state of the vehicle. A variety of methods exist to estimate the state of the load: use of a magnetic encoder [17], attaching an IMU to the load [18], and image processing using a downward facing camera [19]. An input shaping method, previously implemented on cranes, has been implemented and successfully flight tested on the GTMax platform [20–23]. Delayed cable angle feedback has also been successfully demonstrated [24, 25]. These techniques have been used for load delivery onto a moving platform [24]. A simple torque compensator method has shown to allow for single vehicle and multi-vehicle load transportation [17]. Note that these methods generally try to reduce load oscillations through modification of the system's input and do not optimized an objective or cost. Lastly, the Kaman K-MAX/BURRO autonomous helicopter has completed successful military suspended load operations in Afghanistan [26–28].

The differential dynamic programming (DDP) algorithm is an iteration-based numerical method used to generate solutions to optimal control problems. By computing a quadratic approximation of the cost-to-go function and linearizing the system's dynamics around a nominal trajectory the DDP algorithm is able to obtain optimal open and closed loop control policies [29]. The algorithm is iterative in nature as it updates the optimized input sequentially such that each successive iteration further reduces the considered cost. The same basic principles were used to develop iterative linear quadratic regulators (iLQR) [30, 31]. The algorithm has been successfully implemented in simulation to enable robust bipedal robotic walking [32]. Extensions of the DDP algorithm have been developed in order to address state and control constraints [33, 34].

It should be expected that the manner in which the DDP algorithm propagates the modeled system's configuration and linearizes the resultant discrete dynamics has a large effect on its performance. In this paper, a variational integrator and its linearization are utilized by the optimization framework. Utilizing a variational integrator within the DDP algorithm has been shown to provide significant advantages when compared to utilizing Euler methods [35]. Specifically, the algorithm's performance is far less dependent on the size of the discretization time step. Therefore, the discretization time step can be made larger without sacrificing much performance and results in a reduction of computational effort. A complete and rigorous treatment of variational integrators is presented in References [36] and [37]. The variational integrator and its first-order linearization used in this paper were first reported in References [37] and [38], respectively. Variational integrators have been shown to be scalable and implementable for generic mechanical systems in generalized coordinates [39]. Furthermore, variational integrators were utilized to assist in a real-time implementation of an iterative projection-based optimization method [40].

This paper presents a real-time implementable trajectory optimization framework for autonomous suspended load operations. The computational effort required for the framework is mitigated by the use of variational integrators and a simplified, but representative, system model. The unmanned vehicle system's existing guidance, navigation, and control architecture is utilized and, therefore, results in minimal software reconfiguration. Furthermore, the state of the slung load is estimated via an augmentation to the existing navigation system and utilizes only vision-based measurements of the load. It is shown in simulation studies and a flight test the framework is real-time onboard implementable despite a relatively large time horizon of 12 seconds. The vehicle was able to maneuver the suspended load to track reference trajectories.

The remainder of the paper is organized as follows: Sections 2 and 3 review variational integrators and differential dynamic programming, respectively. The flight test vehicle and the simulation environment in which the proposed framework was implemented are outlined in Section 4. The vision-based load state estimation method is described in Section 5. Section 6 presents the proposed trajectory optimization framework for autonomous suspended load operations. Sections 7 and 8 present results obtained from simulation studies and the conducted flight test, respectively. Finally, Sections 9 and 10 give analysis of the results shown in Sections 7 and 8 and final conclusions, respectively.

2 A Variational Integrator

The variational integrator and its linearization utilized in the presented trajectory optimization framework is reviewed in this section. This integration scheme enables real-time implementation of the DDP algorithm due to its ability to accurately propagate a system's configuration despite relatively large discretization time steps. Furthermore, the reviewed variational integrator and its linearization have been shown to be better suited than Euler methods for algorithms that rely heavily on dynamical system propagation and linearization [35].

To begin our discussion, the formulation of the Euler-Lagrangian equations is reviewed by considering the Lagrangian of a dynamical system represented as

$$L(q(t), \dot{q}(t)) = T(q(t), \dot{q}(t)) - V(q(t)), \tag{1}$$

where q is the state configuration vector, $\dot{q}$ is its time derivative, $T(q(t), \dot{q}(t))$ describes the system's kinetic energy, and $V(q(t))$ describes the system's potential energy. The action, S, is defined as

$$S[q(t)] = \int_{t_0}^{t_\mathrm{f}} L(q(\tau), \dot{q}(\tau))\ \mathrm{d}\tau. \tag{2}$$

The Least Action principle is used to derive the variational relation $\delta S[q(t)] = \delta \int_{t_0}^{t_\mathrm{f}} L(q(\tau), \dot{q}(\tau))\ \mathrm{d}\tau = 0$, and results, when minimized, in the classical Euler-Lagrange equations [41]:

$$\frac{\partial}{\partial t}\frac{\partial L}{\partial \dot{q}}(q, \dot{q}) - \frac{\partial L}{\partial q}(q, \dot{q}) = 0. \tag{3}$$

Note equation (3) provides the fundamental characteristics of a dynamical system and describes how the system configuration vector propagates through time. However, it does not provide any method or algorithm that can be used to solve for the trajectory of the system. Simply using numerical integration schemes developed for general second order differential equations will result in numerical errors since the system's fundamental characteristics are ignored.

On the other hand, variational integrators approximate the continuous trajectory of mechanical systems with a sequence of discrete points while ensuring (or strongly enforcing) the conservation of fundamental quantities such as momentum and energy [37]. Specifically, for the variational integrator considered in this paper, a sequence of system configuration vectors $\{(t_0, q_0), (t_1, q_1), \ldots, (t_n, q_n)\}$ is found such that the continuous system trajectory is approximated as $q_m \approx q(t_m)$ where $\Delta t = t_{i+1} - t_i$ is the discretization time step. Derivations for the same variational integrator presented here are given in References [37, 42].

The derivation of the variational integrator begins with approximating the action integral over a small time interval with a generalized midpoint approximation as

$$L_\mathrm{d}(q_k, q_{k+1}) = L((1-\alpha)q_k + \alpha q_k, \frac{q_{k+1} - q_k}{\Delta t})\Delta t \approx \int_{t_k}^{t_{k+1}} L(q(\tau), \dot{q}(\tau))\ \mathrm{d}\tau, \tag{4}$$

where $\alpha \in [0, 1]$ defines the integration approximation and $L_\mathrm{d}(q_k, q_{k+1})$ is referred to as the *discrete Lagrangian.* The midpoint approximation defined with $\alpha = 1/2$ results in second order accuracy as discussed in Reference [43]. The action integral defined in equation (2) can be approximated as a sum of discrete Lagrangians:

$$S[q(t)] \approx \sum_{k=0}^{n-1} L_\mathrm{d}(q_k, q_{k+1}). \tag{5}$$

The Least Action principle is now used to derive the following variational relation

$$\delta S[q(t)] \approx \sum_{k=0}^{n-1} (D_1 L_\mathrm{d}(q_k, q_{k+1}) \cdot \delta q_k + D_2 L_\mathrm{d}(q_k, q_{k+1}) \cdot \delta q_{k+1}) = 0. \tag{6}$$

Equivalently,

$$\delta S[q(t)] \approx \sum_{k=1}^{n-1} (D_1 L_{\mathrm{d}}(q_k, q_{k+1}) + D_2 L_{\mathrm{d}}(q_{k-1}, q_k)) \cdot \delta q_k = 0, \tag{7}$$

since $\delta q_0 = \delta q_n = 0$. Note that the Least Action principle requires that the variations of the action sum be zero for any δq_k. As a result, the Discrete Euler-Lagrange (DEL) equation is derived as

$$D_1 L_{\mathrm{d}}(q_k, q_{k+1}) + D_2 L_{\mathrm{d}}(q_{k-1}, q_k) = 0. \tag{8}$$

Notice that the DEL equation is the discrete time equivalent to the classical Euler-Lagrange equation (3). However, the DEL equation provides a manner in which the system's discrete configuration can be propagated. Given two consecutive system configurations q_k and q_{k-1},

$$f(q_{k+1}) = D_1 L_{\mathrm{d}}(q_k, q_{k+1}) + D_2 L_{\mathrm{d}}(q_{k-1}, q_k) = 0, \tag{9}$$

can be solve implicitly for the next configuration q_{k+1}. Therefore, given q_0 and q_1 equation (9) can be solved iteratively to find $q_2, \ldots, q_n$. Note that propagating the system in this manner ensures that the variational relation described in equation (7) is satisfied. A simple root finder algorithm outline in Algorithm 1 can be used to solve equation (9). The required derivative $Df(\cdot)$ is given as

$$Df(q_{k+1}) = D_2 D_1 L_{\mathrm{d}}(q_k, q_{k+1}). \tag{10}$$

Algorithm 1 Simple Root Finder

```
while $|f(q_{k+1})| > \epsilon_{\mathrm{tol}}$ do
    $q_{k+1} \leftarrow q_{k+1} - Df^{-1}(q_{k+1}) \cdot f(q_{k+1})$
end while
```

The required derivatives in equations (9), (10), and those needed for the linearization of the integrator equations can be found using the chain rule and equation (4) [39]:

$$D_1 L_{\mathrm{d}}(q_k, q_{k+1}) = \frac{\partial}{\partial q} L(q, \dot{q})(1-\alpha)\Delta t - \frac{\partial}{\partial \dot{q}} L(q, \dot{q}), \tag{11}$$

$$D_2 L_{\mathrm{d}}(q_k, q_{k+1}) = \frac{\partial}{\partial q} L(q, \dot{q})\alpha\Delta t + \frac{\partial}{\partial \dot{q}} L(q, \dot{q}), \tag{12}$$

$$\begin{aligned} D_1 D_1 L_{\mathrm{d}}(q_k, q_{k+1}) = {} & \frac{\partial^2}{\partial q \partial q} L(q, \dot{q})(1-\alpha)^2 \Delta t - \frac{\partial^2}{\partial \dot{q} \partial q} L(q, \dot{q})(1-\alpha) \\ & - \frac{\partial^2}{\partial q \partial \dot{q}} L(q, \dot{q})(1-\alpha) + \frac{\partial^2}{\partial \dot{q} \partial \dot{q}} L(q, \dot{q}) \frac{1}{\Delta t}, \end{aligned} \tag{13}$$

$$\begin{aligned} D_2 D_1 L_{\mathrm{d}}(q_k, q_{k+1}) = {} & \frac{\partial^2}{\partial q \partial q} L(q, \dot{q})(1-\alpha)\alpha \Delta t + \frac{\partial^2}{\partial \dot{q} \partial q} L(q, \dot{q})(1-\alpha) \\ & - \frac{\partial^2}{\partial q \partial \dot{q}} L(q, \dot{q})\alpha - \frac{\partial^2}{\partial \dot{q} \partial \dot{q}} L(q, \dot{q}) \frac{1}{\Delta t}. \end{aligned} \tag{14}$$

2.1 The Forced Case

External forces can also be incorporated into the derivation of the variational integrator. When considering continuous trajectories, the Lagrange-d'Alembert principle is used to generalize the Euler-Lagrange equation by modifying the variation of the action, δS, to

$$\delta S[q(t)] = \delta \int_{t_0}^{t_{\mathrm{f}}} L(q(\tau), \dot{q}(\tau)) \, \mathrm{d}\tau + \int_{t_0}^{t_{\mathrm{f}}} F(q(\tau), \dot{q}(\tau), u(\tau)) \cdot \delta q \, \mathrm{d}\tau \tag{15}$$

where $F(q(\tau), \dot{q}(\tau), u(\tau))$ represents the total external forcing acting on the system and u is the system's control input (if any). Minimization of the variational relation leads to the forced Euler-Lagrange equation:

$$\frac{\partial}{\partial t}\frac{\partial L}{\partial \dot{q}}(q, \dot{q}) - \frac{\partial L}{\partial q}(q, \dot{q}) = F(q(t), \dot{q}(t), u(t). \tag{16}$$

Similar to the discretization of the Lagrangian, the left, $F_{\mathrm{d}}^{-}(q_k, q_{k+1}, u_k)$, and right, $F_{\mathrm{d}}^{+}(q_k, q_{k+1}, u_k)$, discrete forces are introduce in order to obtain a discrete equivalent to equation (16). The variation of the continuous external force is approximated over a small time interval as

$$F_{\mathrm{d}}^{-}(q_k, q_{k+1}, u_k) \cdot \delta q_k + F_{\mathrm{d}}^{+}(q_k, q_{k+1}, u_k) \cdot \delta q_{k+1} \approx \int_{t_k}^{t_{k+1}} F(q(\tau), \dot{q}(\tau), u(\tau)) \cdot \delta q \, \mathrm{d}\tau \tag{17}$$

where a generalized midpoint approximation can be used to define the the left and right discrete forces as

$$F_{\mathrm{d}}^{\pm}(q_k, q_{k+1}, u_k) = \frac{1}{2}F((1-\alpha)q_k + \alpha q_{k+1}, \frac{q_{k+1} - q_k}{\Delta t}, u_k), \tag{18}$$

and $u_k = u(t_k)$. The variational relation given in equation (7) can then be modified and the resulting forced DEL equation is given as

$$D_1 L_{\mathrm{d}}(q_k, q_{k+1}) + F_{\mathrm{d}}^{-}(q_k, q_{k+1}, u_k) + D_2 L_{\mathrm{d}}(q_{k-1}, q_k) + F_{\mathrm{d}}^{+}(q_{k-1}, q_k, u_{k-1}) = 0. \tag{19}$$

or, equivalently, the forced DEL equation can be written in its *position-momentum* form

$$p_k + D_1 L_{\mathrm{d}}(q_k, q_{k+1}) + F_{\mathrm{d}}^{-}(q_k, q_{k+1}, u_k) = 0, \tag{20}$$

$$p_k = D_2 L_{\mathrm{d}}(q_{k-1}, q_k) + F_{\mathrm{d}}^{+}(q_{k-1}, q_k, u_{k-1}). \tag{21}$$

Note that p_k does not depend on q_{k+1} and in the unforced case p_k is the momentum quantity conserved by the integrator [39, 43]. Furthermore, the previously defined two-step mapping $(q_{k-1}, q_k) \rightarrow (q_{k+1})$ is now replaced with a one step mapping $(q_k, p_k) \rightarrow (q_{k+1}, p_{k+1})$. The integrator equation and its derivative are now defined as

$$f(q_{k+1}) = p_k + D_1 L_{\mathrm{d}}(q_k, q_{k+1}) + F_{\mathrm{d}}^{-}(q_k, q_{k+1}, u_k), \tag{22}$$

$$Df(q_{k+1}) = D_2 D_1 L_{\mathrm{d}}(q_k, q_{k+1}) + D_2 F_{\mathrm{d}}^{-}(q_k, q_{k+1}, u_k). \tag{23}$$

As before, given q_0, q_1, and the control input, $u(t)$, equation (22) can be solved iteratively to find $q_2, \ldots, q_n$.

2.2 The Constrained Case

Holonomic constraints can also be incorporated into the presented variational integrator. Specifically, the considered constraints are of the form

$$h(q) = [h_1(q), \ldots, h_m(q)]^{\mathrm{T}} \tag{24}$$

where the system configuration is said to be valid if $h(q) = 0$. Holonomic constraints restrict the set of possible system configurations to lie in a sub-manifold. Therefore, during propagation the computed system configurations should lie in the desired sub-manifold. The forced DEL equations can be modified to incorporate holonomic constraints [44]:

$$D_1 L_{\mathrm{d}}(q_k, q_{k+1}) + F_{\mathrm{d}}^{-}(q_k, q_{k+1}, u_k) + D_2 L_{\mathrm{d}}(q_{k-1}, q_k) + F_{\mathrm{d}}^{+}(q_{k-1}, q_k, u_{k-1}) = Dh^{\mathrm{T}}(q_k)\lambda_k, \tag{25}$$

$$h(q_{k+1}) = 0. \tag{26}$$

The term $Dh^{\mathrm{T}}(q_k)\lambda_k$ represents a force that imposes the constraint and λ_k is the discrete Lagrange multiplier that defines the magnitude of this force. Note that the inclusion of the equation $h(q_{k+1}) = 0$ ensures that each discrete system configuration, q_k, observes the defined holomonic constraints. The integrator equation and its derivative are now defined as

$$f(q_{k+1}, \lambda_k) = \begin{bmatrix} p_k + D_1 L_{\mathrm{d}}(q_k, q_{k+1}) + F_{\mathrm{d}}^{-}(q_k, q_{k+1}, u_k) - Dh^{\mathrm{T}}(q_k)\lambda_k \\ h(q_{k+1}) \end{bmatrix}, \tag{27}$$

$$Df(q_{k+1}, \lambda_k) = \begin{bmatrix} D_2 D_1 L_{\mathrm{d}}(q_k, q_{k+1}) + D_2 F_{\mathrm{d}}^{-}(q_k, q_{k+1}, u_k) & -Dh^{\mathrm{T}}(q_k) \\ Dh(q_{k+1}) & 0 \end{bmatrix}. \tag{28}$$

The system configuration, q_k, and the Lagrange multipliers, λ_k, are propagated. The simple root finder algorithm is modified such that the estimate of the discrete Lagrangian multipliers are also updated:

$$\begin{bmatrix} q_{k+1} \\ \lambda_k \end{bmatrix} \leftarrow \begin{bmatrix} q_{k+1} \\ \lambda_k \end{bmatrix} - Df^{-1}(q_{k+1}, \lambda_k) \cdot f(q_{k+1}, \lambda_k) \tag{29}$$

2.3 Propagation of a Mass-Spring System

In order to further elucidate the reviewed variational integrator, consider a simple linear mass-spring system with unity mass and spring stiffness constant of 5 N/m. The Lagrangian of the system is computed as

$$L(q, \dot{q}) = \frac{1}{2}\dot{q}^2 - \frac{5}{2}q^2. \tag{30}$$

If $\alpha = \frac{1}{2}$ then the resulting DEL equations in the *position-momentum* form are computed as

$$p_k - \frac{5(q_{k+1} + q_k)}{4}\Delta t - \frac{q_{k+1} - q_k}{\Delta t} = 0, \tag{31}$$

$$p_{k+1} = -\frac{5(q_{k+1} + q_k)}{4}\Delta t + \frac{q_{k+1} - q_k}{\Delta t}, \tag{32}$$

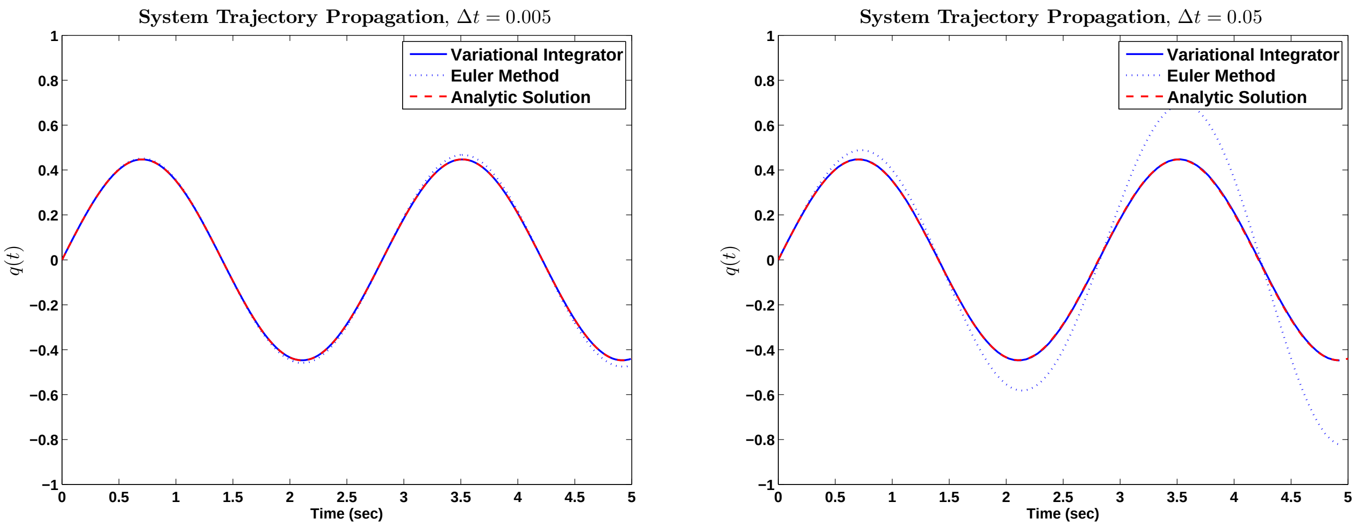
and the derivative of the integrator equation is $Df(q_{k+1}) = -\frac{5}{4}\Delta t - \frac{1}{\Delta t}$.

It is assumed that the initial conditions are given as $q(t_0) = 0$ and $\dot{q}(t_0) = 1$. In order to be consistent with the midpoint approximation used in this example ($\alpha = \frac{1}{2}$) initial conditions were set as $q_1 = \frac{1}{2}\dot{q}(t_0)\Delta t$ and $q_0 = -\frac{1}{2}\dot{q}(t_0)\Delta t$. Note that the selection of q_0 and q_1 is not unique. The quantity p_1 can be found through equation (32) and equation (31) can then be used to obtain q_2. This process can be repeated indefinitely.

Alternatively, the Euler method can be used to propagate the system configuration such that

$$\begin{bmatrix} q_{k+1} \\ \dot{q}_{k+1} \end{bmatrix} = \begin{bmatrix} 1 & \Delta t \\ -5q_k\Delta t & 1 \end{bmatrix} \begin{bmatrix} q_k \\ \dot{q}_k \end{bmatrix}. \tag{33}$$

Figure 1 shows the computed system trajectories along with the analytic solution ($q(t) = \frac{1}{\sqrt{5}}\sin(\sqrt{5}t)$) when $\Delta t = 0.005$ and $\Delta t = 0.05$. Notice that the variational integrator is able to maintain its accuracy despite the increase in the discretization time step while the Euler method deteriorates. As a result, when implementing algorithms that rely on discrete system dynamics less discretization points are needed if the presented variational integrator is utilized. For example, if a DDP algorithm was implemented with a time horizon of 10 seconds the number of discretization points can be reduced by 1800 if the time step if increased from $\Delta t = 0.005$ to $\Delta t = 0.05$. Therefore, utilizing a variational integrator enables the DDP algorithm to use less memory and leads to a reduction in computational effort.



(a) (b)

Figure 1: (a) and (b): Propagation of the mass position, $q(t)$, with initial condition $q(t_0) = 0$ and $\dot{q}(t_0) = 1$. Solid and dotted lines indicate that the propagation was obtained using a variational integrator and the Euler method, respectively, while the dashed lines display the analytic solution.

2.4 Structured Linearization of the Discrete Dynaimcs

As discussed in Reference [38] the one-step mapping (20) implicitly defines the function

$$x_{k+1} = g(x_k, u_k), \tag{34}$$

where $x_{k+1} = [p_{k+1}^{\mathrm{T}}, q_{k+1}^{\mathrm{T}}]^{\mathrm{T}}$. The linearization of the implicitly defined function $g(x_k, u_k)$ can be found explicitly (first shown in References [38] and [45]). Furthermore, the linearization is computed using only partial derivatives of the discrete Lagrangian and forces. Specifically, the derived forced DEL equations (20) and (21) are used to obtain a first-order linearization of the discrete dynamics of the form

$$\delta x_{k+1} = g_x(x_k, u_k)\delta x_k + g_u(x_k, u_k)\delta u_k, \tag{35}$$

or, equivalently,

$$\begin{bmatrix} \delta q_{k+1} \\ \delta p_{k+1} \end{bmatrix} = \begin{bmatrix} \frac{\partial q_{k+1}}{\partial q_k} & \frac{\partial q_{k+1}}{\partial p_k} \\ \frac{\partial p_{k+1}}{\partial q_k} & \frac{\partial p_{k+1}}{\partial p_k} \end{bmatrix} \begin{bmatrix} \delta q_k \\ \delta p_k \end{bmatrix} + \begin{bmatrix} \frac{\partial q_{k+1}}{\partial u_k} \\ \frac{\partial p_{k+1}}{\partial u_k} \end{bmatrix} \delta u_k. \tag{36}$$

To begin, equation (20) is implicitly differentiated with respect to q_k:

$$\frac{\partial}{\partial q_k}[p_k + D_1 L_{k+1} + F_{k+1}^- = 0],$$

$$0 + D_1 D_1 L_{k+1} + D_2 D_1 L_{k+1} \frac{\partial q_{k+1}}{\partial q_k} + D_1 F_{k+1}^- + D_2 F_{k+1}^- \frac{\partial q_{k+1}}{\partial q_k} = 0,$$

$$\frac{\partial q_{k+1}}{\partial q_k} = -M_{k+1}^{-1}[D_1 D_1 L_{k+1} + D_1 F_{k+1}^-], \tag{37}$$

where $M_{k+1} = D_2 D_1 L_{k+1} + D_2 F_{k+1}^-$ is assumed to be non-singular at q_k, p_k, and u_k. For ease of exposition, the discrete Lagrangian notation is condensed to $L_{k+1} = L_{\mathrm{d}}(q_k, q_{k+1})$ and that of the discrete forces to $F_{k+1}^{\pm} = F_{\mathrm{d}}^{\pm}(q_k, q_{k+1}, u_k)$. Repeating this procedure yields

$$\frac{\partial q_{k+1}}{\partial p_k} = -M_{k+1}^{-1}, \tag{38}$$

$$\frac{\partial q_{k+1}}{\partial u_k} = -M_{k+1}^{-1} D_3 F_{k+1}^-. \tag{39}$$

The remaining derivatives can be found by explicitly differentiating (21):

$$\frac{\partial p_{k+1}}{\partial q_k} = [D_2 D_2 L_{k+1} + D_2 F^{+}_{k+1}] \frac{\partial q_{k+1}}{\partial q_k} + D_1 D_2 L_{k+1} + D_1 F^{+}_{k+1}, \tag{40}$$

$$\frac{\partial p_{k+1}}{\partial p_k} = [D_2 D_2 L_{k+1} + D_2 F^{+}_{k+1}] \frac{\partial q_{k+1}}{\partial p_k}, \tag{41}$$

$$\frac{\partial p_{k+1}}{\partial u_k} = [D_2 D_2 L_{k+1} + D_2 F^{+}_{k+1}] \frac{\partial q_{k+1}}{\partial u_k} + D_3 F^{+}_{k+1}. \tag{42}$$

Note that q_{k+1} is needed in order to evaluate the derivatives and can be found by solving equation (20). A linearization for the constrained case and second-order linearizations (constrained and unconstrained) are formulated in References [38] and [45].

3 Differential Dynamic Programming

In this section the differential dynamic programming (DDP) algorithm is reviewed. Given a nominal input and associated system trajectory the algorithm finds the optimal control variation to minimize a given cost. The nominal input is then updated and the process is repeated. An abridged derivation of the algorithm is given here while References [29] and [46] provide a complete treatment of the topic.

To begin, consider the optimal control problem given by a cost under minimization of the form

$$J(x, u, t) = h(x(t_{\mathrm{f}})) + \int_{t_0}^{t_{\mathrm{f}}} l(x(\tau), u(\tau), \tau)\ \mathrm{d}\tau, \tag{43}$$

subjected to the dynamics given by

$$\frac{\mathrm{d}x}{\mathrm{d}t} = f(x, u), \quad x_0 = x(t_0), \tag{44}$$

where $h(x(t_{\mathrm{f}}))$ is the terminal cost, $l(x(t), u(t), t)$ is the running cost, $x \in \mathbb{R}^n$ is the state vector, and $u \in \mathbb{R}^p$ is the control input. The DDP algorithm attempts to find the optimal sequence of discrete inputs to minimize the given cost such that the continuous input is then defined as $u(T_k) \in u_k,\ T_k = [t_0 + k\Delta t,\ t_0 + (k+1)\Delta t)$ where Δt is the discretization time step. As a result, the algorithm approximates continuous system trajectories by a sequence of state vectors such that $x_1 = x(t_0)$, $x_2 = x(t_0 + \Delta t)$, ..., $x_N = x(t_{\mathrm{f}})$. The appropriate selection of Δt depends on the given system dynamics and cost function. It should be noted that the DDP algorithm is iterative. That is, given a sequence of discrete inputs $\bar{U} = \{\bar{u}_1, \ldots, \bar{u}_{N-1}\}$ the algorithm finds the optimal control deviation, $\delta U^{\star}$, such that the control input is updated as

$$\bar{U}_{\mathrm{new}} = \bar{U} + \gamma \delta U^{\star}_i, \tag{45}$$

where γ is a user defined constant or is selected from an automated process (e.g. an Armijo line search [47]). Several iterations may be needed in order to arrive at a control input that is sufficiently close to the optimal solution.

To begin the derivation of the DDP algorithm it is assumed that a sequence of *nominal* discrete inputs $\bar{U} = \{\bar{u}_1, \ldots, \bar{u}_{N-1}\}$ is used to propagate the associated state trajectory $\bar{X} = \{\bar{x}_1, \ldots, \bar{x}_N\}$. Next, the first-order linearization of the discrete system dynamics along the nominal trajectory of the form

$$\delta x_{k+1} = A_k \delta x_k + B_k \delta u_k \tag{46}$$

is found. In this paper the linearization presented in equation (36) is used to define A_k and B_k. Other integration schemes and their associated linearizations can be used. For example, the Euler method can be utilized such that $A_k = \mathbf{I}_n + f_x(x,u)\Delta t$ and $B_k = f_u(x,u)\Delta t$. However, utilizing variational integrators within the DDP algorithm provides significant advantages when compared to utilizing Euler methods [35]. Specifically, the performance of the DDP algorithm is less dependent on the size of the discretization time step when variational integrators are used. Therefore, Δt can be made larger without sacrificing much performance and, as a result, the computational effort of implementing the DDP algorithm can be mitigated.

Utilizing the derived first-order linearization of the system dynamics a second-order expansion of the cost-to-go function,

$$V(x(t),t) = \min_{u} J(x,u,t), \tag{47}$$

around the nominal trajectory, $\bar{X}_i$, is obtained:

$$V(\bar{x}+\delta x) = V(\bar{x}) + V_x(\bar{x})^{\mathrm{T}}\delta x + \frac{1}{2}\delta x^{\mathrm{T}} V_{xx}(\bar{x})\delta x, \tag{48}$$

or, in its approximated discrete form,

$$\begin{aligned}
V(\bar{x}_{k+1}+\delta x_{k+1}) &= V(\bar{x}_{k+1}) + V_x(\bar{x}_{k+1})^{\mathrm{T}}\delta x_{k+1} + \delta x_{k+1}^{\mathrm{T}} V_{xx}(\bar{x}_{k+1})\delta x_{k+1}, \\
&= V(\bar{x}_{k+1}) + V_x(\bar{x}_{k+1})^{\mathrm{T}}(A_k\delta x_k + B_k\delta u_k) + \delta x_k^{\mathrm{T}} A_k^{\mathrm{T}} V_{xx}(\bar{x}_{k+1}) B_k \delta u_k \\
&\quad + \frac{1}{2}\delta x_k^{\mathrm{T}} A_k^{\mathrm{T}} V_{xx}(\bar{x}_{k+1}) A_k \delta x_k + \frac{1}{2}\delta u_k^{\mathrm{T}} B_k^{\mathrm{T}} V_{xx}(\bar{x}_{k+1}) B_k \delta u_k.
\end{aligned}$$

Now consider the discretized state action value function defined as

$$Q(x_k,u_k) = L(x_k,u_k) + V(x_{k+1}) \tag{49}$$

where $L(x_k,u_k) = l(x_k,u_k)\Delta t$. By invoking Bellman's Principle of Optimality it is known that the optimal control deviation, $\delta U^\star$, is found by minimizing the state action value. A local second-order expansion of the state action value function is derived as

$$\begin{aligned}
Q(\bar{x}_k+\delta x_k, \bar{u}_k+\delta u_k) &= Q(\bar{x}_k,\bar{u}_k) + \delta u_k^{\mathrm{T}} Q_u(\bar{x}_k,\bar{u}_k) + \delta x_k^{\mathrm{T}} Q_x(\bar{x}_k,\bar{u}_k) + \frac{1}{2}\delta u_k^{\mathrm{T}} Q_{uu}(\bar{x}_k,\bar{u}_k)\delta u_k \\
&\quad + \frac{1}{2}\delta x_k^{\mathrm{T}} Q_{xx}(\bar{x}_k,\bar{u}_k)\delta x_k + \delta u_k^{\mathrm{T}} Q_{ux}(\bar{x}_k,\bar{u}_k)\delta x_k,
\end{aligned} \tag{50}$$

where

$$\begin{aligned}
Q_x(\bar{x}_k,\bar{u}_k) &= L_x(\bar{x}_k,\bar{u}_k) + A_k^{\mathrm{T}} V_x(\bar{x}_{k+1}), \\
Q_u(\bar{x}_k,\bar{u}_k) &= L_u(\bar{x}_k,\bar{u}_k) + B_k^{\mathrm{T}} V_x(\bar{x}_{k+1}), \\
Q_{xx}(\bar{x}_k,\bar{u}_k) &= L_{xx}(\bar{x}_k,\bar{u}_k) + A_k^{\mathrm{T}} V_{xx}(\bar{x}_{k+1}) A_k, \\
Q_{uu}(\bar{x}_k,\bar{u}_k) &= L_{uu}(\bar{x}_k,\bar{u}_k) + B_k^{\mathrm{T}} V_{xx}(\bar{x}_{k+1}) B_k, \\
Q_{xu}(\bar{x}_k,\bar{u}_k) &= L_{xu}(\bar{x}_k,\bar{u}_k) + A_k^{\mathrm{T}} V_{xx}(\bar{x}_{k+1}) B_k.
\end{aligned} \tag{51}$$

The optimal control deviation is computed by minimizing the state action value as

$$\begin{aligned}
\delta u_k^\star &= \operatorname*{arg\,min}_{\delta u_k} Q(\bar{x}_k+\delta x_k, \bar{u}_k+\delta u_k), \\
&= -Q_{uu}(\bar{x}_k,\bar{u}_k)^{-1}(Q_u(\bar{x}_k,\bar{u}_k) + Q_{ux}(\bar{x}_k,\bar{u}_k)\delta x_k^\star).
\end{aligned} \tag{52}$$

where the optimal state deviation is propagated as

$$\delta x_k^\star = A_k \delta x_{k-1}^\star + B_k \delta u_{k-1}^\star, \quad \delta x_0^\star = 0. \tag{53}$$

Note that equation (52) contains a feedforward and a feedback component. Therefore, the term $Q_{uu}^{-1}Q_{ux}$ gives the optimal state feedback for a particular iteration. The feedback component attempts to keep the system's trajectory near the optimized trajectory, $\bar{X}$. Explicitly, supposing that $x(t_k)$ is the state of the system at time t_k then the input signal is given as

$$\bar{v}(t_k) = \bar{u}(t_k) - Q_{uu}^{-1}(\bar{x}_k, \bar{u}_k)^{-1}Q_{xu}(\bar{x}_k, \bar{u}_k)(x(t_k) - \bar{x}_k). \tag{54}$$

Plugging $\delta u_k^\star$ back into (50) yields a backward propagating approximation of the second-order expansion of the cost-to-go function:

$$V(\bar{x}_k) = V(\bar{x}_{k+1}) + L(\bar{x}_k, \bar{u}_k) - Q_u(\bar{x}_k, \bar{u}_k)Q_u(\bar{x}_k, \bar{u}_k)^{-1}Q_u(\bar{x}_k, \bar{u}_{k+1}), \tag{55}$$

$$V_x(\bar{x}_k) = Q_x(\bar{x}_k, \bar{u}_k) - Q_u(\bar{x}_k, \bar{u}_k)Q_u(\bar{x}_k, \bar{u}_k)^{-1}Q_{xu}^{\mathrm{T}}(\bar{x}_k, \bar{u}_k), \tag{56}$$

$$V_{xx}(\bar{x}_k) = Q_{xx}(\bar{x}_k, \bar{u}_k) - Q_{xu}(\bar{x}_k, \bar{u}_k)Q_u(\bar{x}_k, \bar{u}_k)^{-1}Q_{xu}^{\mathrm{T}}(\bar{x}_k, \bar{u}_k), \tag{57}$$

where the initial conditions are derived from the terminal cost, $h(\cdot)$, as $V(\bar{x}_N) = h(\bar{x}_N), V_x(\bar{x}_N) = h_x(\bar{x}_N), V_{xx}(\bar{x}_N) = h_{xx}(\bar{x}_N)$.

This completes the derivation of the DDP algorithm. The derived optimal control deviation, $\delta U^\star$, is used to update the nominal input as shown in equation (45). In the implementation presented in this paper an Armijo line search is used to automatically select an appropriate step size [47]. The process can then be repeated using the updated input as the new nominal input. The DDP algorithm is outlined in Algorithm 2.

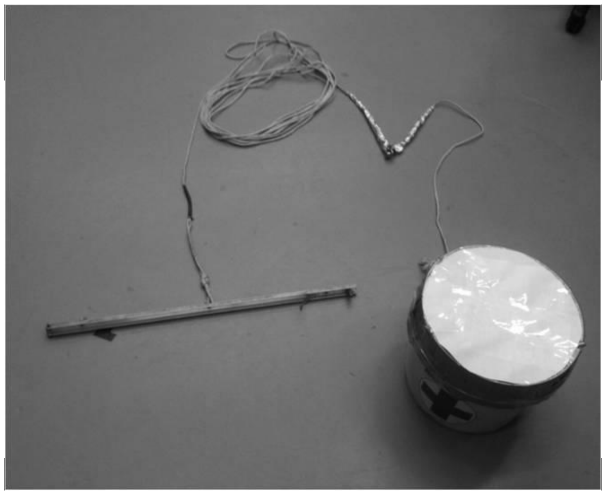
Algorithm 2 The DDP Algorithm with an Armijo Line Search

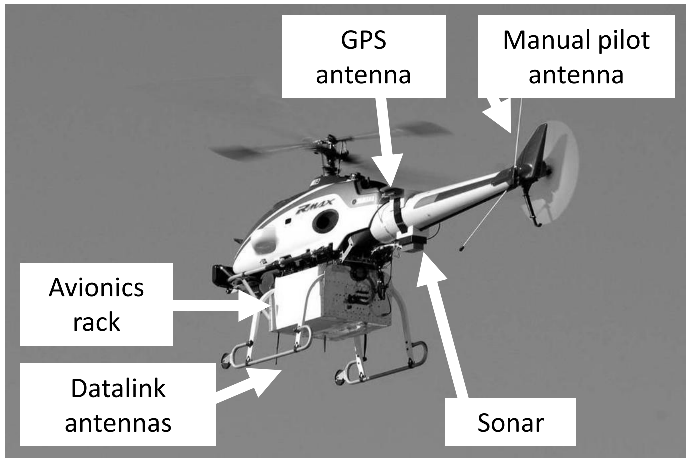
```
Require:
  Initial discrete control input $u(t)$, algorithm parameters $\kappa, \beta, \epsilon$
  cost function $J(x, u, t)$, system dynamics $\mathrm{d}x = f(x, u)\ \mathrm{d}t$
  while Cost updates results in more than $\epsilon$ in difference do
      Propagate the discretized trajectory using the nominal input $\bar{U}$
      Linearize the value function and system dynamics along the trajectory
      Approximate the value function through back-propagation
      Compute $\delta U^\star$ and $\delta X^\star$
      while $\mathrm{Cost}_\mathrm{p} > \mathrm{Cost} + \kappa\beta\Big(\sum_{K=1}^{N-1}\big(L_x^{\mathrm{T}}(x_k, u_k)\delta x_k^\star + L_u^{\mathrm{T}}(x_k, u_k)\delta u_k^\star\big) + h_x^{\mathrm{T}}(x_N, u_N)\delta x_N^\star\Big)$ do
          Find the proposed input $U_\mathrm{p} \leftarrow \bar{U} + \beta^j\delta U^\star$ and the corresponding trajectory, $X_\mathrm{p}$
          Find proposed cost $\mathrm{Cost}_\mathrm{p} \leftarrow J(X_\mathrm{p}, U_\mathrm{p}, t)$ and update $j \leftarrow j + 1$ if needed
      end while
      Update the control, trajectory and Cost: $\bar{U} \leftarrow U_\mathrm{p}, \bar{X} \leftarrow X_\mathrm{p}, \mathrm{Cost} \leftarrow \mathrm{Cost}_\mathrm{p}$
  end while
```

4 Flight Test Vehicle and Simulation Environment

Simulation studies and flight tests used in the validation of the proposed trajectory optimization framework were conducted with a modified Yamaha RMAX helicopter UAV, dubbed the GTMax [48]. Custom avionics and the flight control software needed for autonomous flight were designed at the Georgia Tech Unmanned Aerial Vehicle Research Facility (UAVRF). The GTMax vehicle is equipped with an extensive sensor suite including an Inertial Science IMU, short-range sonar, magnetometer, and differential GPS [48, 49]. In addition, the vehicle is instrumented with a Prosilica GC 1380 camera for vision-based operations [50–52]. Figure 2b shows the GTMax flight test platform.

Two separate onboard processes, primary and secondary, are used for all onboard computational requirements and are executed on a single onboard computer with an Intel Core i7 processor. The primary


(a)



(b)

Figure 2: (a): Suspended load used during the presented flight tests. (b): An annotated image of the GTMax vehicle test platform.

process performs the basic functionality needed for navigation, guidance, and control of the vehicle. An adaptive neural network model inversion flight controller, the vehicle's baseline controller, was used for all simulation studies and the presented flight test [53]. The secondary process is used to perform image capture and processing needed for the presented load state estimator. In addition, the proposed trajectory optimization framework was executed in the secondary process. Figure 7 shows an overview of the proposed computational architecture.

The suspended load weighs 8.4 pounds and is attached to the vehicle with a 46 foot braided nylon line. The bucket is sealed with a white cover to assist with tracking as discussed in Section 5. In addition, the load can be jettisoned by the safety pilot with a radio transmitter. Figure 2a shows the suspended load used in the flight test.

The Georgia Tech UAV Simulation Tool (GUST), developed at the UAVRF, is used to simulate flight tests in order to demonstrate the efficacy of the proposed trajectory optimization framework [54, 55]. GUST contains a high-fidelity vehicle and environment model, onboard flight control software, and ground station software and is used for rapid development and testing of software and hardware for all aspects of autonomous vehicle operation. The vehicle is represented with a rigid body six degree of freedom model with additional engine, fuel, and rotor dynamics. In addition, the vehicle model simulates sensor noise, communication delay, real world location, orientation, and actuator dynamics and saturation. Furthermore, a simulated camera capturing rendered graphics is used to predict the real-time performance of vision-based operations. External disturbances such as turbulence and wind are also simulated. The suspended load is simulated by GUST with a rigid body six degree of freedom model with flexible cable dynamics.

In addition, aerodynamic loading is represented with a reduced order model based on a quasi-steady aerodynamics, unsteady effects of body motion, and unsteady effects of vortex shedding [56]. This aerodynamic model significantly improves the fidelity of the suspended load simulation when compared to a simple model that only considers quasi-steady drag. Preliminary flight tests show a good correspondence between the model and the observed behavior of the load. Further details on the high fidelity aerodynamic loading model and the preliminary flight tests results can be found in Reference [56].

5 Vision-Based Estimation of Load

The presented vision-based suspended load state estimator is an extension to the suspended load state estimator previously presented in References [18] and [19]. The proposed framework provides estimates of the suspended load state (swing angle and rate) through augmentation of an existing vehicle navigation system. The suspended load is not instrumented with any sensors and only images captured from a downward facing camera provide sensing information. Furthermore, the estimated vehicle state is use to drive the suspended load's process model and an unscented Kalman filter produces estimation updates.

5.1 Process Model

The process and sensor model used in the presented framework are the same as those used in the estimator presented in References [18] and [19]. The system model is reviewed here and the cited references are referred to for further details.

A simple pendulum process model is used to described the suspended load system. Specifically, the suspended load is modeled as a point mass where the position of the load is given by generalized coordinates θ_w and ϕ_w (swing angles) which can be considered a 2-1 Euler angle rotation about the attachment point on the vehicle. Figure 3.a shows the representation of the system. The modeled system evolves as

$$\ddot{\theta}_w L = -\cos(\theta_w)\cos(\phi_w)\ddot{x}_{ha} + \sin(\theta_w)\cos(\phi_w)(\ddot{z}_{ha} - g), \tag{58}$$
$$\dot{\theta}_w(t_0) = \dot{\theta}_{w0}, \theta_w(t_0) = \theta_{w0},\ t \geq 0,$$
$$\ddot{\phi}_w L = \sin(\theta_w)\sin(\phi_w)\ddot{x}_{ha} + \cos(\theta_w)\sin(\phi_w)(\ddot{z}_{ha} - g) - \cos(\phi_w)\ddot{y}_{ha}, \tag{59}$$
$$\dot{\phi}_w(t_0) = \dot{\phi}_{w0}, \phi_w(t_0) = \phi_{w0},$$

where g is the gravitational constant, $\ddot{x}_{ha}$, $\ddot{y}_{ha}$, and $\ddot{z}_{ha}$ are the vehicle's translational accelerations at the attachment point in the inertial frame and L is the total length of the pendulum measured from the attachment point to the center of mass of the load. The position of the load is given as

$$\mathbf{R}_l^I = \mathbf{R}_h^I + \mathbf{R}_{ha}^I + \begin{bmatrix} \sin(\theta_w)\cos(\phi_w) \\ \sin(\phi_w) \\ \cos(\theta_w)\cos(\phi_w) \end{bmatrix} L,$$

where $\mathbf{R}_h^I$ is the position of the vehicle in the inertial frame, $\mathbf{R}_l^I$ is the position of the load in the inertial frame, and $\mathbf{R}_{ha}^I$ is the vector from the center of mass of the vehicle to the attachment point expressed in the inertial frame. Furthermore, the velocity of the load is computed as

$$\dot{\mathbf{R}}_l^I = \dot{\mathbf{R}}_h^I + \dot{\mathbf{R}}_{ha}^I + \begin{bmatrix} \cos(\theta_w)\cos(\phi_w) & -\sin(\theta_w)\sin(\phi_w) \\ 0 & \cos(\phi_w) \\ -\sin(\theta_w)\cos(\phi_w) & -\cos(\theta_w)\sin(\phi_w) \end{bmatrix} \begin{bmatrix} \dot{\theta}_w \\ \dot{\phi}_w \end{bmatrix} L.$$

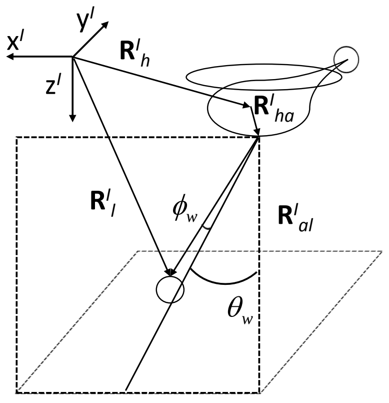
5.2 Sensor Model

The sensor information extracted by processing the captured images from a downward facing camera is the unit vector, $\mathbf{R}_{cl}^h$, from the camera to the white disk on top of the load expressed in the vehicle frame. The estimate of the load position relative to the attachment point of the helicopter is given by

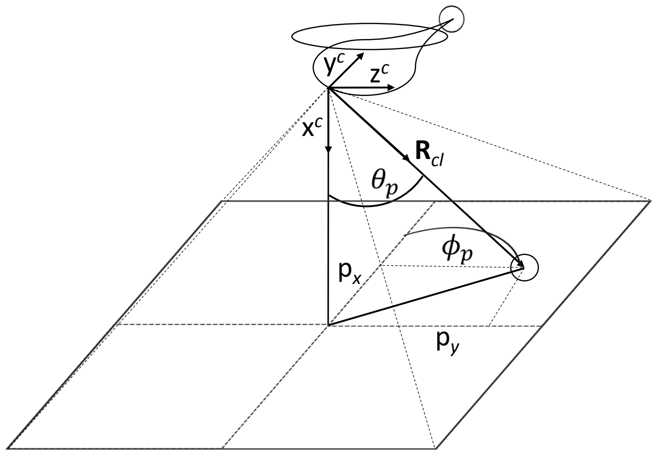
$$\hat{\mathbf{R}}_{al}^I = \begin{bmatrix} \sin(\hat{\theta}_w)\cos(\hat{\phi}_w) \\ \sin(\hat{\phi}_w) \\ \cos(\hat{\theta}_w)\cos(\hat{\phi}_w) \end{bmatrix} L,$$

where $\hat{\theta}_w$ and $\hat{\phi}_w$ are the estimated swing angles. The predicted measurement, $\hat{\mathbf{R}}_{cl}^h$, needed to implement an unscented Kalman filter is found as [57]

$$\hat{\mathbf{R}}_{cl}^h = \frac{\mathbf{T}_{hI}\hat{\mathbf{R}}_{al}^I + \mathbf{R}_{ha}^h - \mathbf{R}_{hc}^h}{|\mathbf{T}_{hI}\hat{\mathbf{R}}_{al}^I + \mathbf{R}_{ha}^h - \mathbf{R}_{hc}^h|}, \tag{60}$$



(a)



(b)

Figure 3: (a): Diagram of the process model used in the vision-based estimator. (b): Diagram of the camera coordinate frame. Figures recreated from those found in Reference [19].

.

where $\mathbf{R}^h_{hc}$ is the vector from the center of mass of the vehicle to the camera in the vehicle frame, $\mathbf{R}^h_{ha}$ is the vector from the center of mass of the vehicle to the helicopter attachment point in the vehicle frame, and $\mathbf{T}_{hI}$ is the transformation matrix from the inertial frame to the vehicle frame.

5.3 Image Capture and Processing

As previously mentioned, images from a downward facing camera are processed in order to obtain sensor information to aid in the estimation of the suspended load state. In order to assist in localization a white disc is placed on top of the load. A probability-based mapping is then computed on a gray-scaled image in order to identify the likelihood that pixels from the image are centers to a white disk of a specified radius. The pixels on the white disc are assumed to be sampled from a normal distribution parameterized with user-defined mean and variance. Using the cumulative distribution function of a normal distribution the mapping produces a score of how "white" the image around a pixel is:

$$S_{(r,s)} = \prod_{(i,j)\in\mathcal{R}} \frac{1}{2}\big(1 + \text{erf}\left(\frac{P_{(r+i,s+j)} - \mu}{\sigma\sqrt{2}}\right)\big) \tag{61}$$

where $S_{(r,s)} \in [0,1]$ is the pixel score, $\mathcal{R}$ is the set of pixel translations that approximates a disk of a desired radius, $P_{(r,s)} \in [0,255]$ is the gray-scale value of the (r,s) pixel where $P_{(r,s)} = 0$ if the pixel is black and $P_{(r,s)} = 255$ if the pixel is white, and erf$(\cdot)$ is the error function. As shown in Figure 4, set $\mathcal{R}$ is defined as

$$\mathcal{R} = \{(i,j) : i,j \in \mathbb{N},\ |i| + |j| \le r\} \tag{62}$$

where r is the radius of the disc. As an example, if a disc of unity radius is considered ($r = 1$) then $\mathcal{R} = \{(0,0),(1,0),(0,1),(-1,0),(0,-1)\}$. In order for a pixel to achieve a score close to unity all of its neighboring pixels and itself should be "whiter than most other pixels" given the specified mean and variance. If the pixel with the highest score meets user-defined minimum requirements a positive detection of the load occurs. Figure 5 gives an example of the mapping produced by (61) given $\mu = 220$, $\sigma = 5$ and $r = 2$ (pixel values scaled to produce a post-processed gray-scaled image). Notice the white pixels in the post-processed image correspond to the center of the load. The selection of r is determined by the length

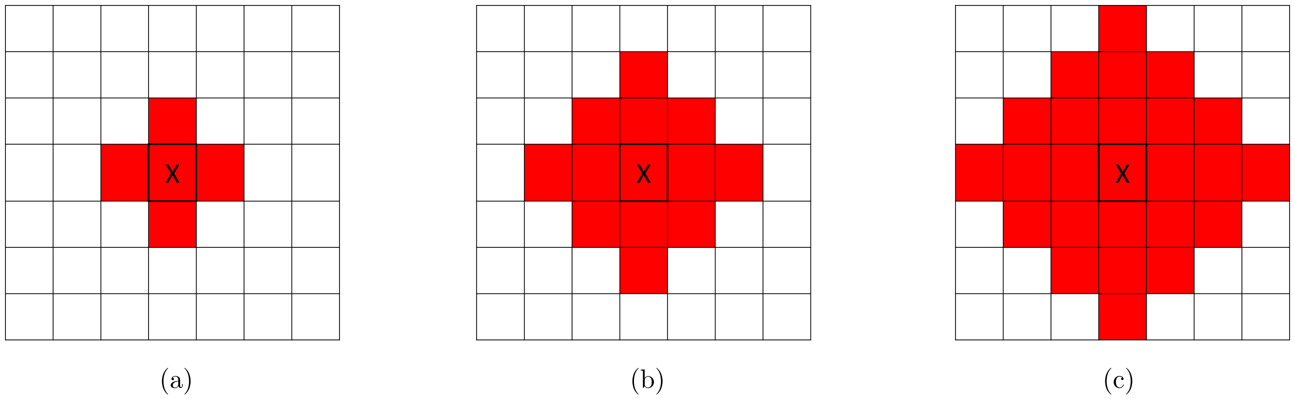


Figure 4: Depictions of the set $\mathcal{R}$ for a radius of 1 (a), 2 (b), and 3 (c).

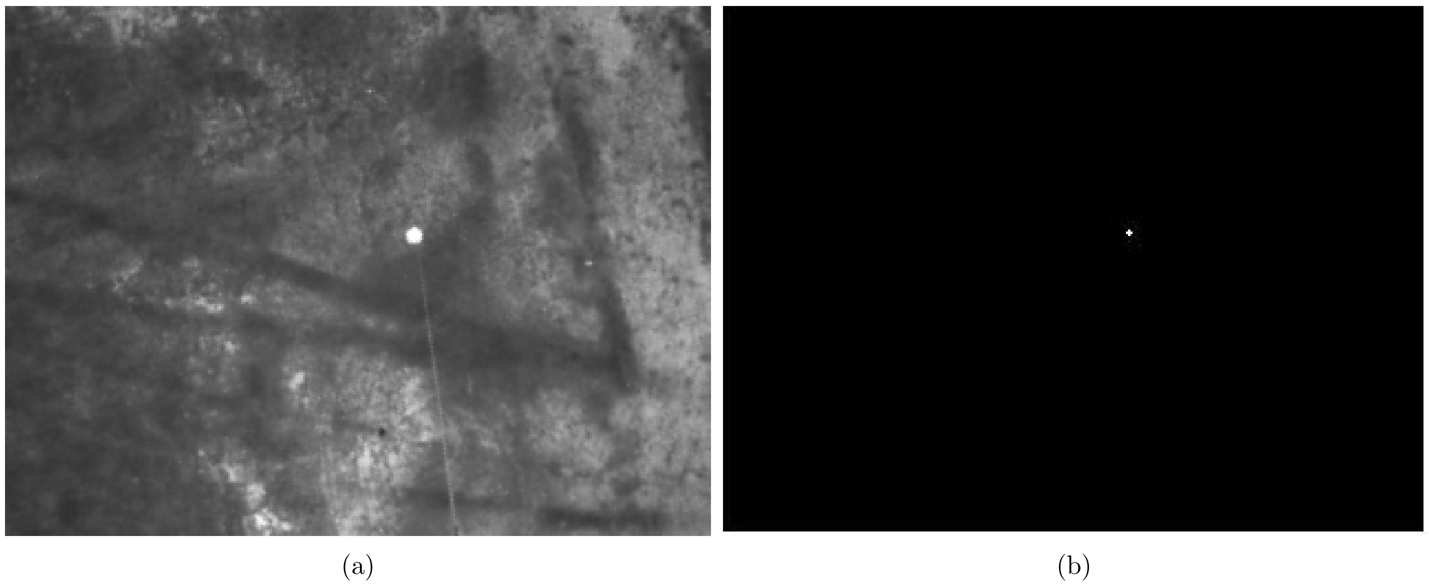


Figure 5: (a): Typical image from a flight test. (b): The post-processed image when $\mu = 220$, $\sigma = 5$ and $r = 2$.

of the cable and the camera resolution. Furthermore, if possible, σ and μ should be selected such that the environment does not produce scores above the user-defined minimum.

Note the use of the cumulative distribution function of a normal distribution in equation (61) is arbitrary. Other functions that map pixel values on to $[0, 1]$ can be used. However, in order for the function to be effective a pixel should receive a very low score if one of its neighbors is significantly different from the *ideal* pixel. Finally, it should be noted that the target disc can be made a different color depending on the application. For example, the disc can be red or orange in operations with thick snow/ice on the ground. In this case, a pixel will achieve a high score if its neighbors and itself are "more red than most other pixels". Appropriate changes to equation (61) can be made trivially.

Once a pixel is selected, the coordinates of the pixel (p_x, p_y), as shown in Figure 3.b, are used to compute the sensed unit vector from the camera to the load. First, the coordinate p_y is scaled based on the image dimensions as $\bar{p}_y = \frac{l_x}{l_y} p_y$ where l_x and l_y are the width and height of the image in pixels. Next, characteristic angles are computed as

$$\theta_p = \rho\sqrt{p_x^2 + \bar{p}_y^2}, \quad \text{and} \quad \phi_p = \operatorname{atan2}(\bar{p}_y, p_x) \tag{63}$$

where ρ is the ratio of the field of view in radians and the width in pixels of the image. The sensed unit

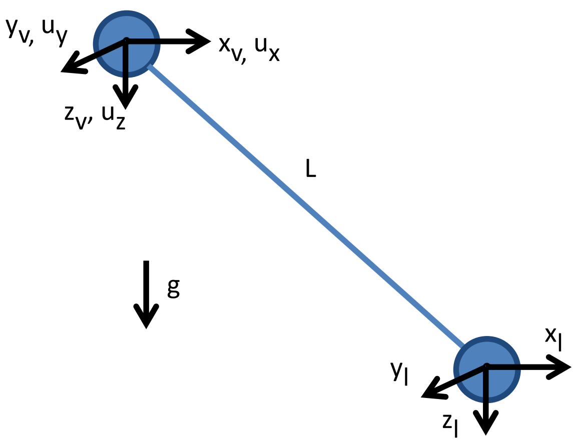


Figure 6: Diagram of the modeled suspended load system.

vector from the camera to the load can then be computed as

$$\mathbf{R}^{h}_{cl} = \mathbf{T}_{hc} \begin{bmatrix} \cos(\theta_p) \\ \sin(\theta_p)\cos(\phi_p) \\ \sin(\theta_p)\sin(\phi_p) \end{bmatrix},$$

where $\mathbf{T}_{hc}$ is the transformation matrix from the camera frame to the vehicle frame.

6 Trajectory Optimization Framework

The proposed trajectory optimization framework is presented in this section. The framework utilizes the DDP algorithm to obtain optimized vehicle trajectories in a receding horizon fashion. A simplified system model that captures the fundamental pendulum-like dynamics of the system is used. The simplified representation causes the optimization procedure not to be overly complex while ensuring that the outputted solution can be used to guide the coupled rotorcraft and suspended load system. Furthermore, the presented variational integrator allows for reliable propagation of the system configuration and results in a faithful representation of the linearization of the discrete system dynamics.

6.1 System Model and Cost Function

Since guidance (not control) was the goal of the optimization framework a simplified model was used to represent the complex rotorcraft and suspended load system. The model consist of two point masses under a gravitational field with a holonomic constraint restricting the distance between them to equal a defined length. Furthermore, three control forces can accelerate the mass representing the vehicle. Figure 6 shows a depiction of the simplified model. Variational integrators were used to propagate and linearize the system in the DDP algorithm. As discussed in Section 2.2, the variational integration ensures that each discrete system configuration will observe the defined constraint. The Lagrangian of the system is given as

$$L(q,\dot{q}) = \frac{1}{2}M(\dot{x}_{\mathrm{v}}^2 + \dot{y}_{\mathrm{v}}^2 + \dot{z}_{\mathrm{v}}^2) + \frac{1}{2}m(\dot{x}_{\mathrm{l}}^2 + \dot{y}_{\mathrm{l}}^2 + \dot{z}_{\mathrm{l}}^2) + mgz_{\mathrm{l}} + mgz_{\mathrm{v}}, \quad (64)$$

where $q = [x_\mathrm{v}, y_\mathrm{v}, z_\mathrm{v}, x_\mathrm{l}, y_\mathrm{l}, z_\mathrm{l}]^\mathrm{T}$, M is the mass of the vehicle, and m is the mass of the load. The holonomic constraint is defined as

$$h(q) = (x_\mathrm{v} - x_\mathrm{l})^2 + (y_\mathrm{v} - y_\mathrm{l})^2 + (z_\mathrm{v} - z_\mathrm{l})^2 - L^2, \tag{65}$$

where L is the cable length (slacked lines not considered). Finally, the left and right discrete forces needed to evaluate the constrained-forced DEL equations (25) are defined as

$$F_\mathrm{d}^\pm(q_k, q_{k+1}, u_k) = \left[\begin{array}{cccccc} \frac{1}{2}u_x\Delta t & \frac{1}{2}u_y\Delta t & \frac{1}{2}u_z\Delta t & 0 & 0 & 0 \end{array}\right]^\mathrm{T} \tag{66}$$

where Δt is the discretization time step. A variational integrator can be defined for the simplified model by using equations (27) and (28).

The reference trajectory based cost function used in the optimization process is given as

$$\begin{aligned} J(q, u, t) = \int_{t_0}^{t_\mathrm{f}} &\left(w_\mathrm{q}(q(\tau) - q_\mathrm{ref}(\tau))^2 + w_\mathrm{v}(\dot{q}(\tau) - \dot{q}_\mathrm{ref}(t))^2 + w_u(u(\tau) - u_\mathrm{trim}(t))^2\right) \,\mathrm{d}\tau \\ &+ w_\mathrm{f}(q(t_\mathrm{f}) - q_\mathrm{ref}(t_\mathrm{f}))^2. \end{aligned} \tag{67}$$

where $(q_\mathrm{ref}(t), \dot{q}_\mathrm{ref}(t))$ is a reference system trajectory, $w_\mathrm{q}, w_\mathrm{v} \geq 0$ are the state tracking error cost matrices, $w_\mathrm{f} \geq 0$ is the final state tracking error cost matrix, and $w_u > 0$ is the control cost matrix. Though the reference system trajectory should be reasonable (i.e. limited vehicle velocity, acceleration and jerk), it does not have to be dynamically feasible. For example, a reference trajectory with no load swing ($x_\mathrm{v}(t) = x_\mathrm{l}(t)$ and $y_\mathrm{v}(t) = y_\mathrm{l}(t)$) is valid.

The DDP algorithm outlined in Section 3 was implemented to solve the presented trajectory optimization problem in real-time. The initial nominal discrete inputs and its associated state trajectory were the output of the previous optimization cycle shifted to the current initial time t_0. Furthermore, during each optimization cycle a maximum of 5 iterations were performed before a control and state trajectory were outputted. The time horizon $(t_\mathrm{f} - t_0)$ was 12 seconds and $\Delta t = 0.1$. In addition, the cost parameters were defined as $w_\mathrm{q} = \mathrm{diag}([0.5, 0.5, 0.5, 3.0, 3.0, 3.0])$, $w_\mathrm{v} = \mathrm{diag}([0.5, 0.5, 0.5, 3.0, 3.0, 3.0])$, $w_u = \mathrm{diag}([1.0, 1.0, 1.0])$, and $w_\mathrm{f} = \mathrm{diag}([0.5, 0.5, 0.5, 2.0, 2.0, 2.0])$.

The Armijo search parameters were set as $\kappa = 1 \times 10^{-5}$ and $\beta = 0.7$ and the maximum number of possible Armijo iterations was set to 15. If the Armijo search failed (i.e. the Armijo cost criteria was not met after 15 iterations) the current optimization cycle was terminated and the current nominal input and state trajectory were outputted. Furthermore, if $\sum_{K=1}^{N-1}\left(L_x^\mathrm{T}(x_k, u_k)\delta x_k^\star + L_u^\mathrm{T}(x_k, u_k)\delta u_k^\star\right) + h_x^\mathrm{T}(x_N, u_N)\delta x_N^\star < 1 \times 10^{-6}$ the optimization cycle was terminated and the current nominal input and state trajectory were outputted.

6.2 Integration

Figures 7 and 8 give an overview of how the proposed optimization framework can be integrated into an existing onboard computer. It is assumed the existing guidance, navigation, and control system is able to follow reasonable reference trajectories relatively accurately. A second thread in the secondary process is used exclusively to solve the DDP algorithm. As a result, other procedures executed in the secondary process (i.e. image-based sensing, communication to primary process, etc.) are not affected by the computationally heavy optimization procedure. Simulation studies and flight tests have shown that the stored optimization solution is updated at an average rate faster the 10 Hertz. Note that the computational time is much smaller than the optimization horizon of 12 seconds. Therefore, as confirm in simulation and flight tests, real-time trajectory optimization is achieved by the proposed framework.

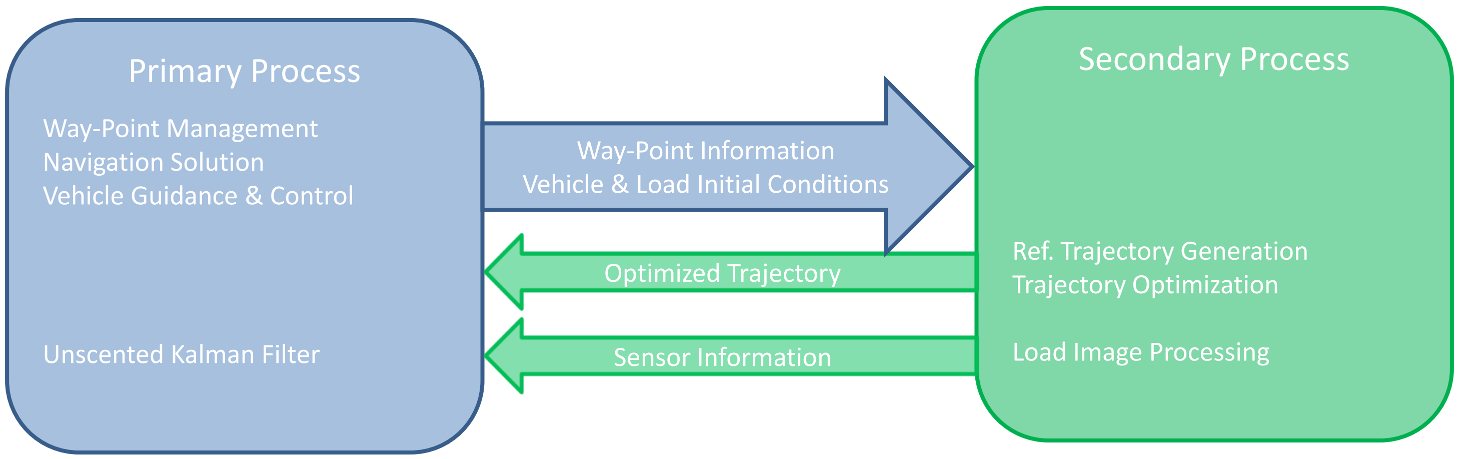


Figure 7: Overview of the implemented communication and computation architecture.

7 Simulation Implementation Results

In order to test the effectiveness of the proposed framework a set of 3 basic maneuvers were performed. First, it was ensured that the vehicle could maintain the load at a fixed location (Maneuver 1: Hover). In order to test for robustness in simulation, the response to a instant change in load velocity was also investigated. The second reference trajectory travels in-between two way-points with a given prescribed maximum velocity and acceleration (Maneuver 2: Way-point). Maneuver 2 reference trajectory was generated using a maximum velocity and acceleration of 10 feet/sec and 2 feet/sec^2, respectively, and traveled about 46 feet and 115 feet along the x-axis and y-axis, respectively. The third reference trajectory was constructed to inscribe a circle with a 50 foot radius every 75 seconds such that its tangential velocity is 4.19 feet/sec (Maneuver 3: Circle). The reference trajectories for the load and the vehicle were identical except for a offset in the altitude. Therefore, the generated reference trajectories are not dynamically feasible. However, note that the consider cost function, presented in equation (67), penalizes deviations associated with the load more than vehicle deviations. As a result, the optimization process favors adjusting the vehicle's trajectory in order to maintain the load closer to the reference trajectory.

In addition to simulating the proposed framework, the response of the system when the vehicle's trajectory was not optimized was simulated. The optimized and unoptimized responses are compared in order to assess the benefits of using the proposed optimization framework. Furthermore, simulation studies were conducted in which the proposed framework was utilized when the navigation solution was error free. That is, the navigation output matched the simulated system states. The effects of the navigation's solution estimation error are evaluated.

7.1 Unoptimized Response

This section presents results obtained from simulations studies where no optimized trajectory was computed and the reference trajectory was used in its place. Therefore, the vehicle simply followed the generated reference trajectory without any consideration of the response of the load or a cost function. Figure 9 shows the trajectory history of the load when Maneuver 1 was performed. At $t = 157.80$ the velocity of the load along the x-axis was instantaneously change to 10 feet/sec. Note that the velocity of the load slowly decays and the load retains a 12 foot oscillation at about 30 seconds after the induce velocity disturbance. Figures 10 and 11 show the trajectory history of the load when Maneuver 3 was performed. As seen in Maneuver 1, large load oscillations are found in the trajectory history. Note the large oscillations along the y-axis at the beginning of the maneuver and those along the x-axis at the end of the maneuver.

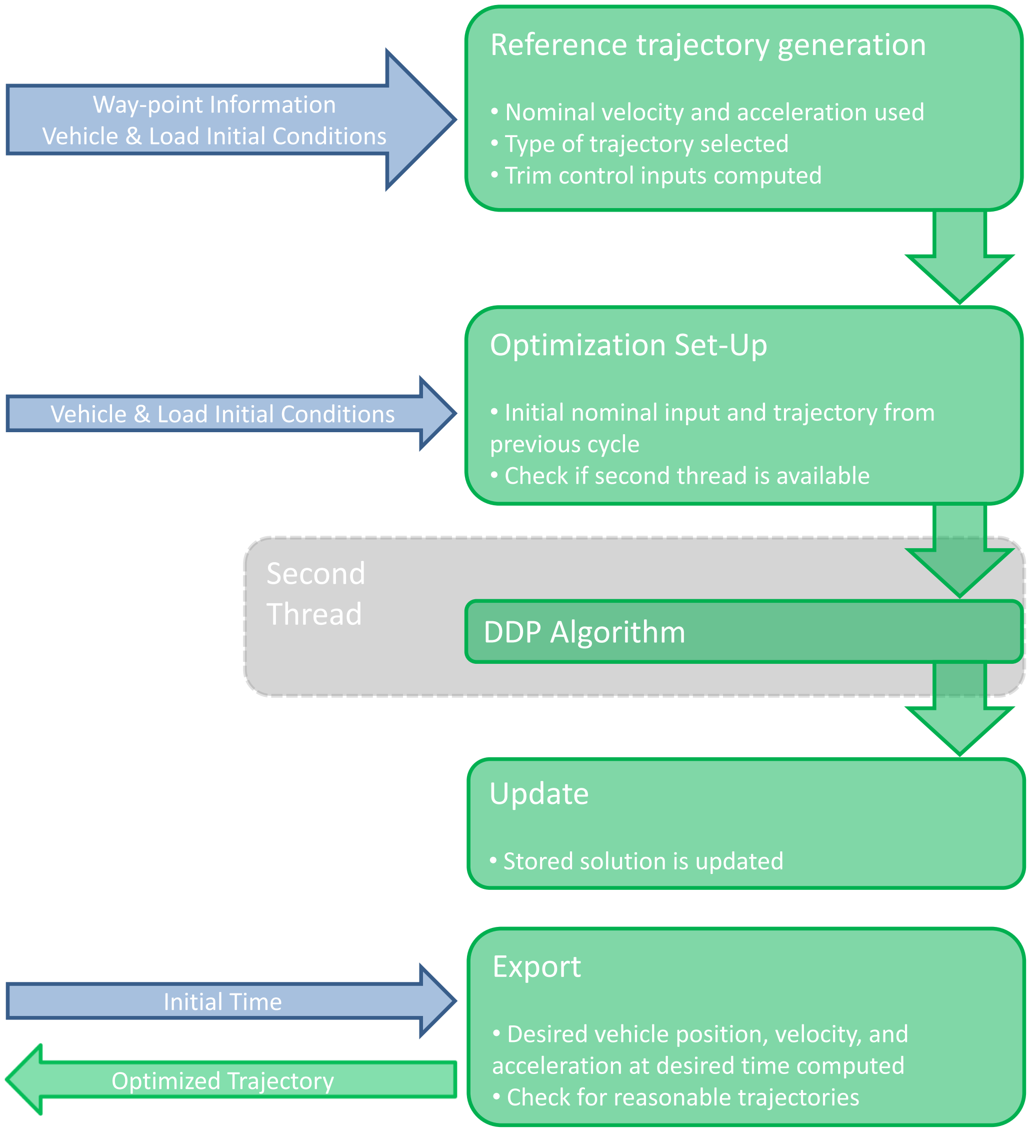

Way-point Information
Vehicle & Load Initial Conditions
Reference trajectory generation
• Nominal velocity and acceleration used
• Type of trajectory selected
• Trim control inputs computed
Vehicle & Load Initial Conditions
Optimization Set-Up
• Initial nominal input and trajectory from previous cycle
• Check if second thread is available
Second Thread
DDP Algorithm
Update
• Stored solution is updated
Initial Time
Optimized Trajectory
Export
• Desired vehicle position, velocity, and acceleration at desired time computed
• Check for reasonable trajectories

Figure 8: Overview of the optimization trajectory optimization process.

7.2 Differential Dynamic Programming Implementation

This section presents results obtained from simulations studies where the proposed trajectory optimization framework was used. Figure 12 shows the trajectory history of the load when Maneuver 1 was performed. At $t = 112.72$ and $t = 150.88$ the velocity of the load along the x-axis was instantaneously change to 5 feet/sec and 10 feet/sec, respectively. Note that the vehicle is able to maintain the load close to the desire position and at "steady state" the load tracking error is always less than 2.5 feet. Furthermore, the system is able to decrease the velocity of the load after the induced disturbance. Figure 13 shows the load position estimation errors (or residuals). Note that there is a bias in both directions. These biases are caused by errors in the vehicle's navigation solution and, in particular, by the estimation error in attitude and accelerometer bias terms. Any error in the estimate of the vehicle's state affects the sensor model used to update the estimate of the load's state. As shown in Figure 12, these biases are manifested in the trajectory history of the load. Figures 14 and 15 shows the trajectory history of the load when Maneuver 2 was performed. Though experiencing large tracking errors during transitional portions of the reference trajectory the suspended load is able to track the reference trajectory generally well. As shown in Figure 15 some of the tracking error is caused by a delayed response. Figures 16 and 17 shows the trajectory history of the load when Maneuver 3 was performed. Note that following an initial transient phase the

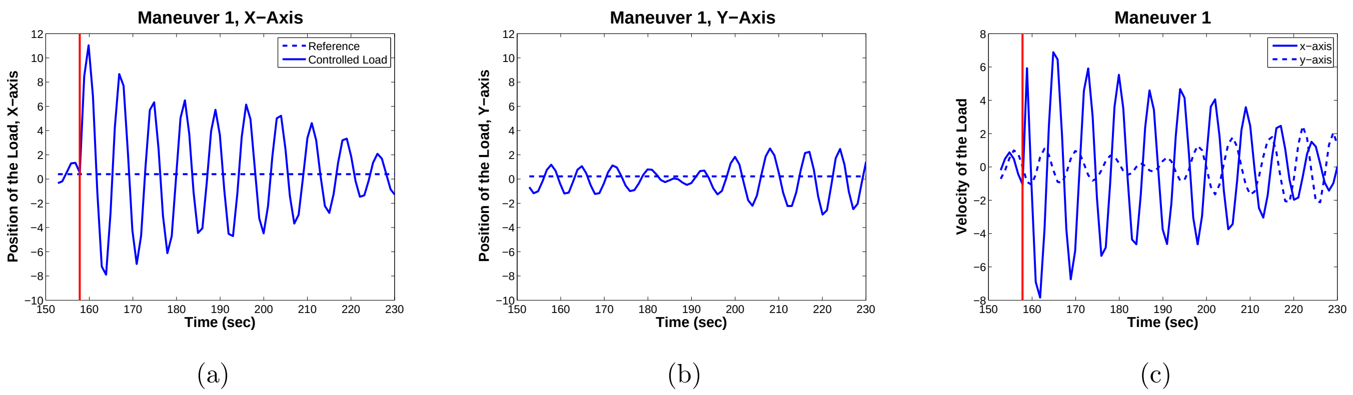


Figure 9: Simulation-Unoptimized Maneuver 1 (a) and (b): Load position as a function of time along the x-axis and y-axis, respectively. Solid and dotted lines represent the load state and reference trajectory, respectively. (c): Load velocity as a function of time. Solid and dotted lines represent the velocity along the x-axis and y-axis, respectively.

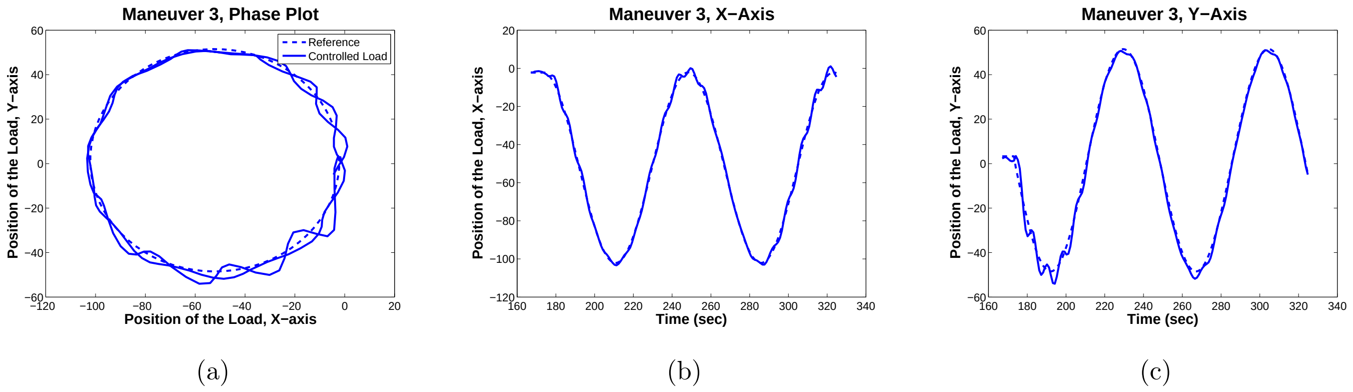


Figure 10: Simulation-Unoptimized Maneuver 3 (a): Phase plot of the maneuver. (b) and (c): Load position as a function of time along the x-axis and y-axis, respectively. Solid and dotted lines represent the load state and reference trajectory, respectively.

load is able to track the reference trajectory with a relatively small error.

7.3 Effect of State Estimator Error

This section presents results obtained from simulations studies where the proposed trajectory optimization framework was used and, in addition, the system's navigation solution was error free (estimated state equaled simulated state). Therefore, the performance of the framework can be judged without considering the detrimental affects of the navigation solution. Figure 18 shows the trajectory history of the load when Maneuver 1 was performed. At $t = 107.87$ and $t = 151.04$ the velocity of the load along the x-axis was instantaneously change to 5 feet/sec and 10 feet/sec, respectively. Note that the vehicle is able to maintain the load close to the desire position and at "steady state" the load tracking error is negligible. Furthermore, the system is able to decrease the velocity of the load at a faster rate when compared to the trajectory shown in Figure 12. Therefore, the performance of the framework can be improved if the navigation solution of the vehicle and the load is improved.

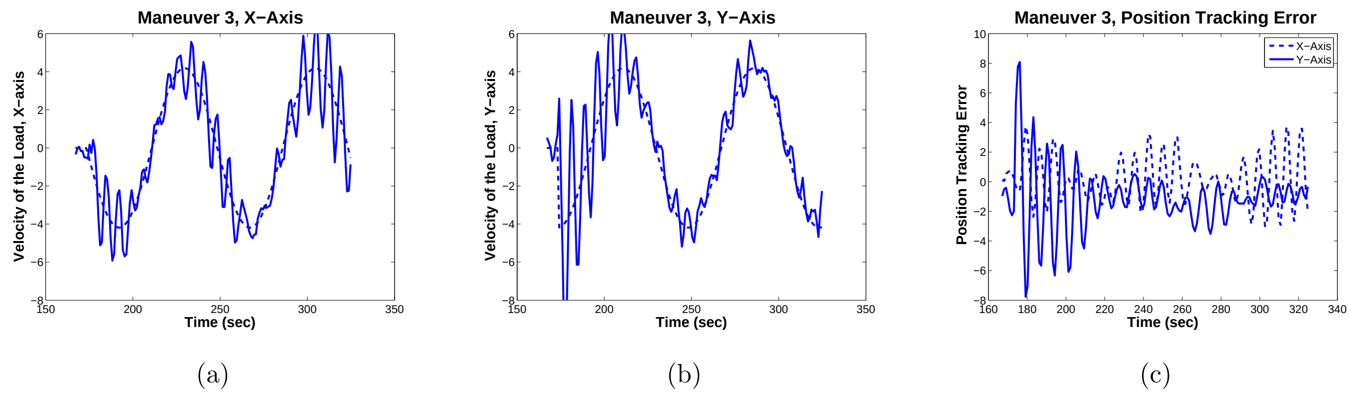


Figure 11: Simulation-Unoptimized Maneuver 3 (a) and (b): Load velocity as a function of time. Solid and dotted lines represent the load state and reference trajectory, respectively. (c): Position tracking error as functions of time. Solid and dotted lines represent the tracking error along the x-axis and y-axis, respectively.

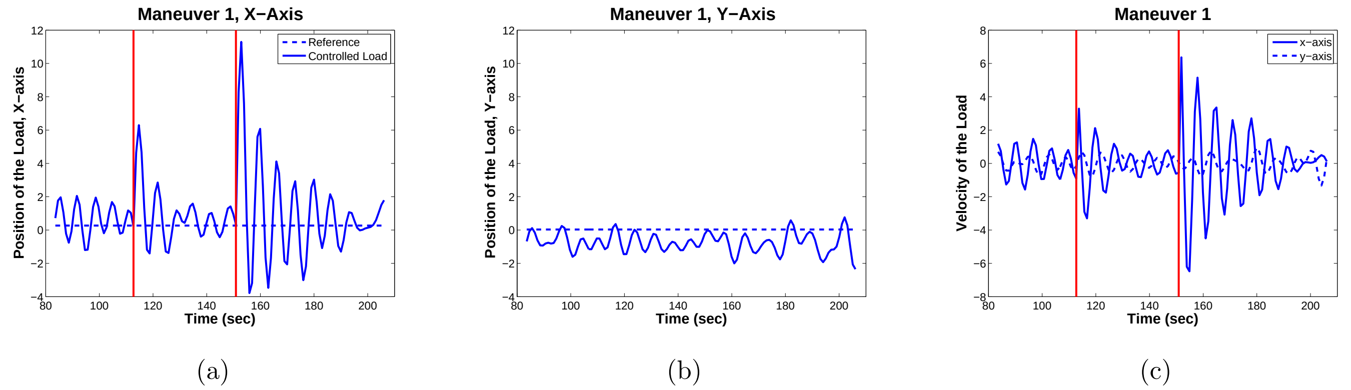


Figure 12: Simulation-DDP Maneuver 1 (a) and (b): Load position as a function of time along the x-axis and y-axis, respectively. Solid and dotted lines represent the load state and reference trajectory, respectively. (c): Load velocity as a function of time. Solid and dotted lines represent the velocity along the x-axis and y-axis, respectively.

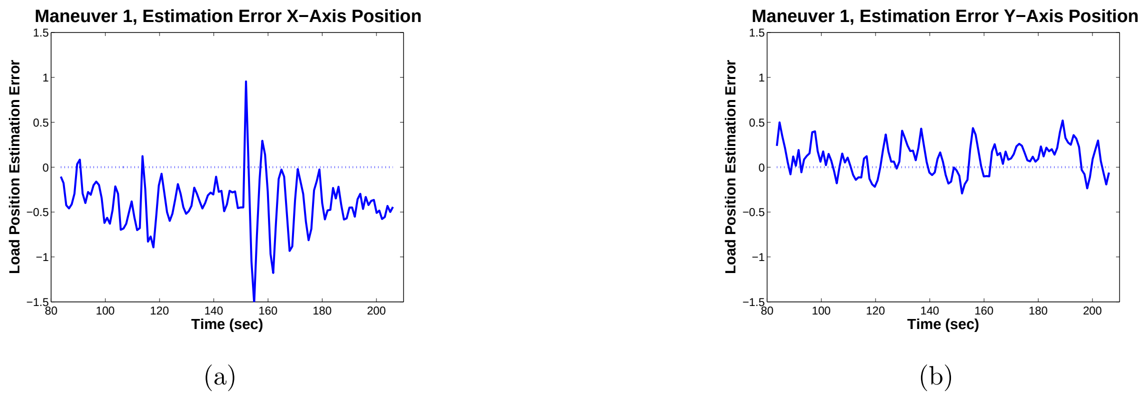
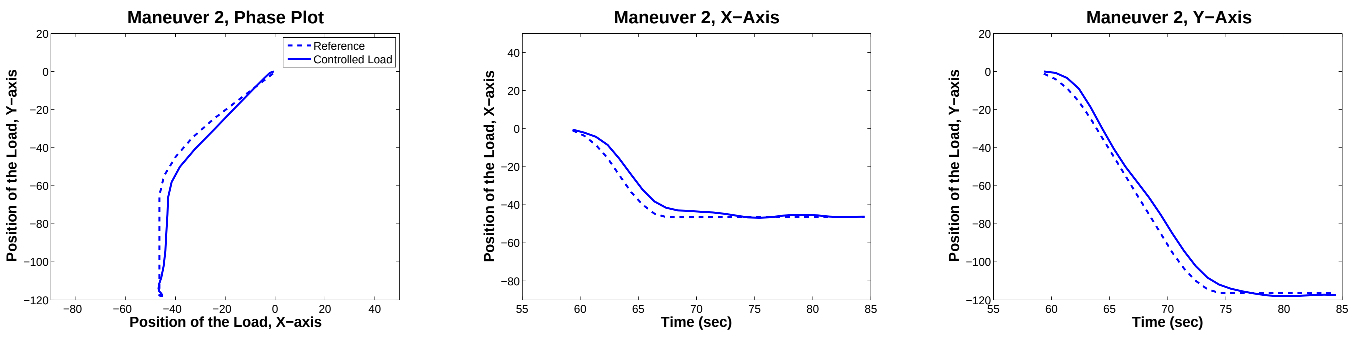


Figure 13: Simulation-DDP Maneuver 1 (a) and (b): The estimation error of the load's position along the x-axis and y-axis, respectively.

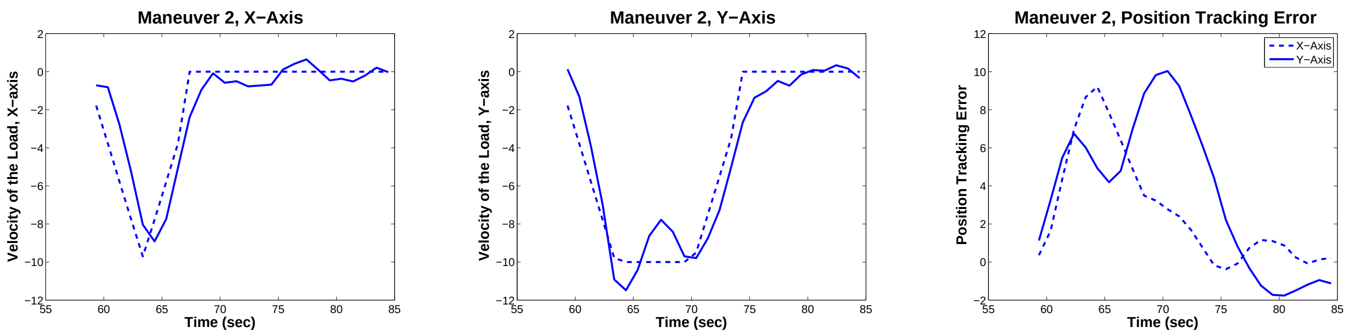
8 Flight Test Results

This section presents results obtained from a flight test where the proposed trajectory optimization framework was utilized. The flight test was perform on March 18, 2015 at around 10am. An air temperature of



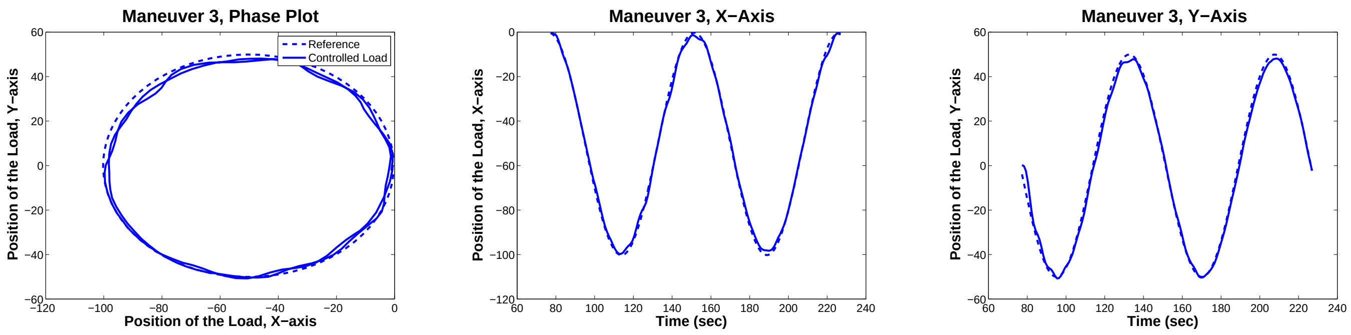
(a) (b) (c)

Figure 14: Simulation-DDP Maneuver 2 (a): Phase plot of the maneuver. (b) and (c): Load position as a function of time along the x-axis and y-axis, respectively. Solid and dotted lines represent the load state and reference trajectory, respectively.



(a) (b) (c)

Figure 15: Simulation-DDP Maneuver 2 (a) and (b): Load velocity as a function of time. Solid and dotted lines represent the load state and reference trajectory, respectively. (c): Position tracking error as functions of time. Solid and dotted lines represent the tracking error along the x-axis and y-axis, respectively.



(a) (b) (c)

Figure 16: Simulation-DDP Maneuver 3 (a): Phase plot of the maneuver. (b) and (c): Load position as a function of time along the x-axis and y-axis, respectively. Solid and dotted lines represent the load state and reference trajectory, respectively.

63.3 degrees Fahrenheit and a wind speed of 2.3 miles per hour in varied directions was reported for area code 31805 by Weather Undergound.

Figures 19 and 20 shows the *estimated* trajectory history of the load when Maneuver 2 was performed. Note that the response of the system is very similar to the response seen in simulation (Figures 14 and 15). As expected, the tracking errors in the flight test are larger than the ones seen in simulation. However,

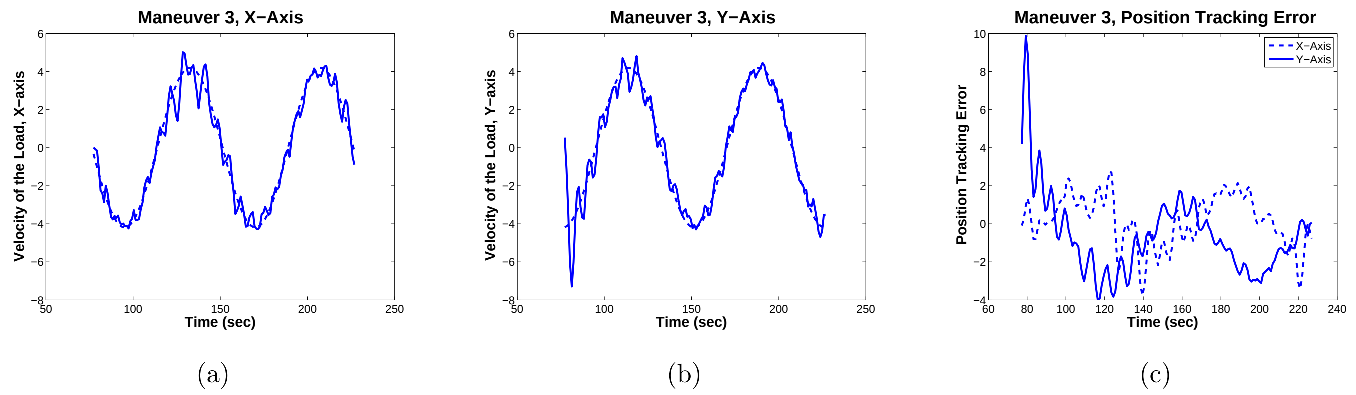


Figure 17: Simulation-DDP Maneuver 3 (a) and (b): Load velocity as a function of time. Solid and dotted lines represent the load state and reference trajectory, respectively. (c): Position tracking error as functions of time. Solid and dotted lines represent the tracking error along the x-axis and y-axis, respectively.

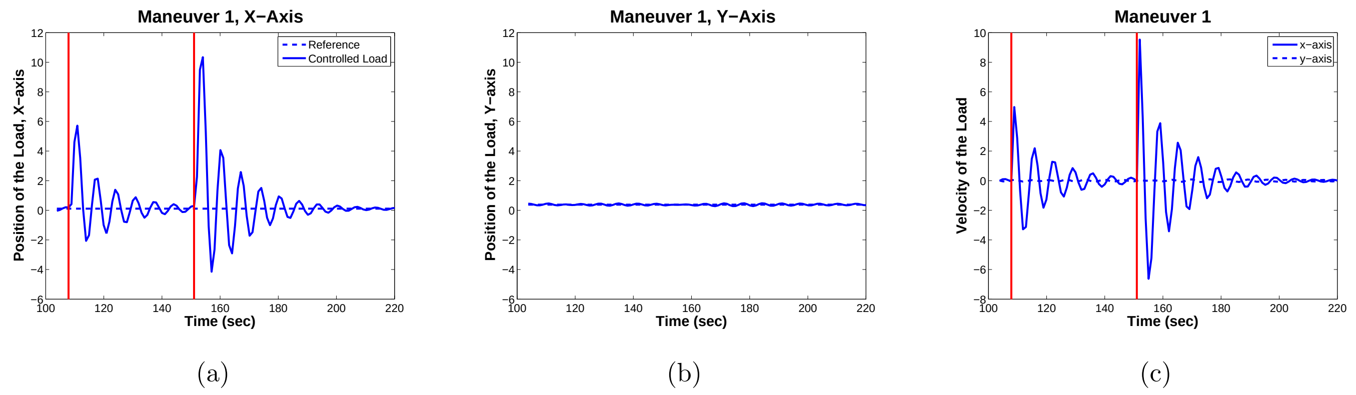
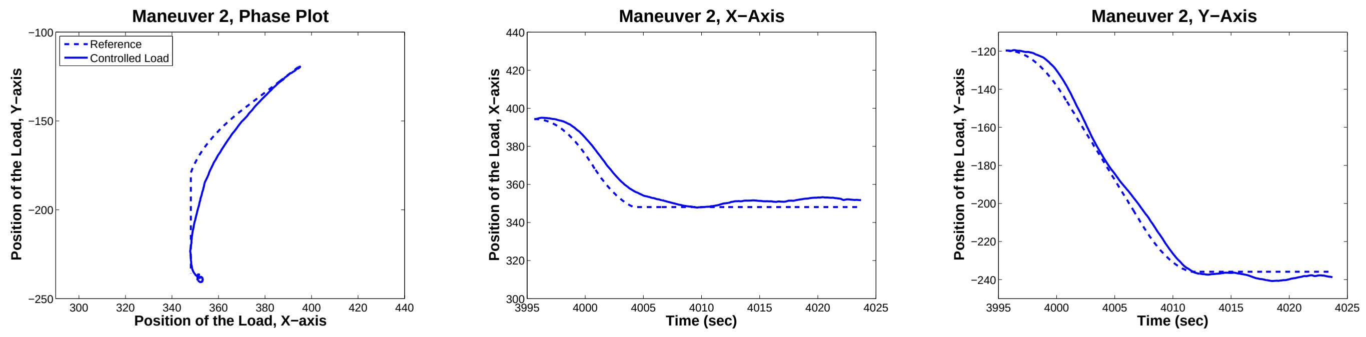


Figure 18: Simulation-DDP (with error free navigation) Maneuver 1 (a) and (b): Load position as a function of time along the x-axis and y-axis, respectively. Solid and dotted lines represent the load state and reference trajectory, respectively. (c): Load velocity as a function of time. Solid and dotted lines represent the velocity along the x-axis and y-axis, respectively.

given that the length of the line is 46 feet and the system is subject to environmental factors, like wind, a tracking error of about 6 feet is reasonable. Furthermore, the estimation bias shown in simulation may also have contributed to the tracking errors. Figures 21 and 22 show the trajectory history of the load when Maneuver 3 was performed. As seen in the previous maneuver, the results obtained through simulation studies are very similar to those obtained in the flight test. Figure 23 shows the histograms of the computational times related to image processing and trajectory optimization when Maneuver 3 was performed. In the recorded data set the computational time required to perform an optimization cycle (maximum of 5 iterations) had a mean of 0.0965 seconds and a standard deviation of 0.0167. This relatively small computational time allowed for real-time trajectory optimization and is possible due to the large discretization time step facilitated by the use of a variational integrator. Furthermore, the computational time required to process an image required for load state estimation had a mean of 0.1329 seconds and a standard deviation of 0.005. When compared to other sensors used onboard for vehicle navigation this "sensor" has a relatively slow update rate.

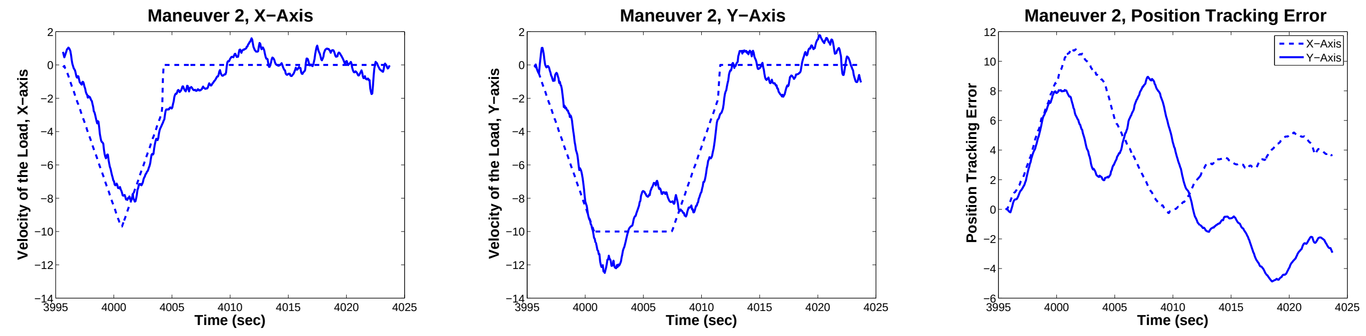
9 Analysis of Results

By comparing Figures 9 and 10 with Figures 12 and 16 it can be seen that the load tracking error is significantly reduced when the vehicle trajectory is optimized. Furthermore, the trajectory of the load



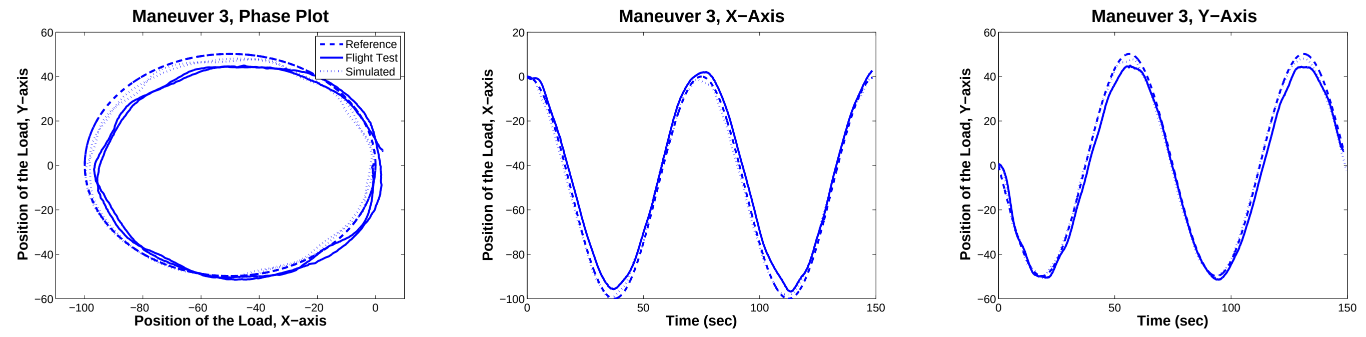
(a) (b) (c)

Figure 19: Flight Test-DDP Maneuver 2 (a): Phase plot of the maneuver. (b) and (c): Load position as a function of time along the x-axis and y-axis, respectively. Solid and dotted lines represent the estimated load state and reference trajectory, respectively.



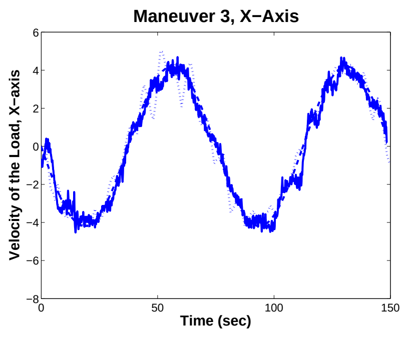
(a) (b) (c)

Figure 20: Flight Test-DDP Maneuver 2 (a) and (b): Load velocity as a function of time. Solid and dotted lines represent the estimated load state and reference trajectory, respectively. (c): Position tracking error as functions of time. Solid and dotted lines represent the estimated tracking error along the x-axis and y-axis, respectively.

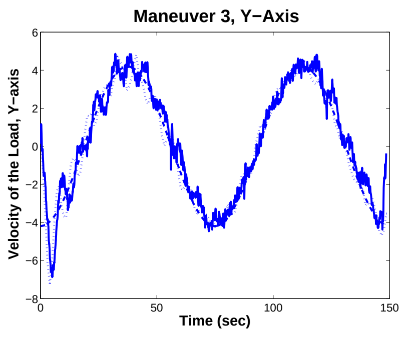


(a) (b) (c)

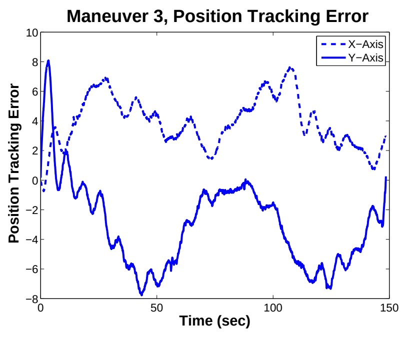
Figure 21: Flight Test-DDP Maneuver 3 (a): Phase plot of the maneuver. (b) and (c): Load position as a function of time along the x-axis and y-axis, respectively. Solid and dotted lines represent the estimated load state and reference trajectory, respectively.



(a)

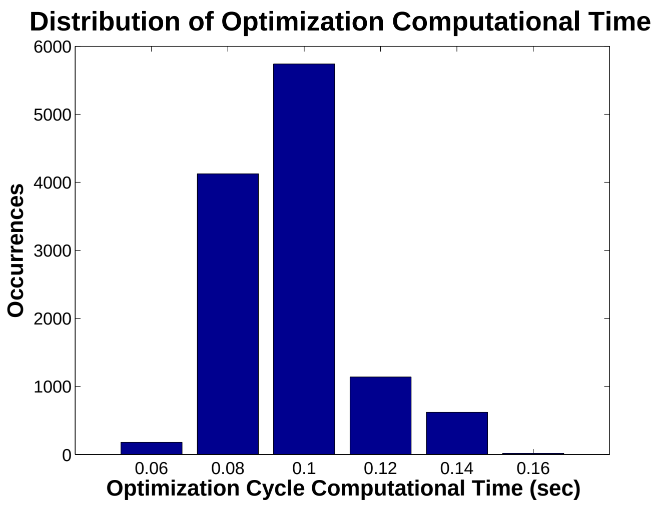


(b)

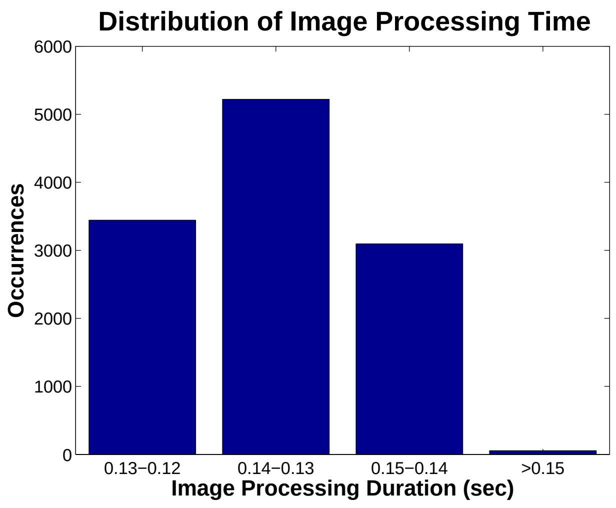


(c)

Figure 22: Flight Test-DDP Maneuver 3 (a) and (b): Load velocity as a function of time. Solid and dotted lines represent the estimated load state and reference trajectory, respectively. (c): Position tracking error as functions of time. Solid and dotted lines represent the estimated tracking error along the x-axis and y-axis, respectively.



(a)



(b)

Figure 23: Maneuver 1 (a): Histogram of the trajectroy optimization cycle computational times. The recorded data set had a mean of 0.0965 seconds and a standard deviation of 0.0167. (b): Histogram of image processing computational times. The recorded data set had a mean of 0.1329 seconds and a standard deviation of 0.005.

is more oscillatory in the unoptimized case. Therefore, the proposed optimization framework provides a significant improvement in the performance of the rotorcraft and suspended load system. Figure 23 demonstrates that the proposed framework is real-time implementable due to its relatively small computation time. It should be noted that any required computation, primary and second processes, is done onboard. Furthermore, note that the presented simulation studies predicted data collected during the flight test. The discrepancies between the simulation studies and the flight test were expected due to environmental and other unmodeled factors. However, it can be expected that further simulation studies where different or more aggressive reference trajectories are implemented will reasonably predict the performance of the framework during a flight test.

The results presented in Section 7.3 suggest that a limiting factor of the presented framework is the accuracy of the vehicle's and suspended load's navigation solution. However, it should be noted that the vehicle's navigation solution is far more accurate than the load's. Therefore, a significant improvement in the framework's performance is expected if a more accurate load sensor is used (e.g. an attached GPS, attachment point encoder, etc.), though additional hardware will be needed. The reference trajectories, in particular Maneuver 2, can also be redefined to be more realistic. Currently, the reference trajectories do not have any imposed constraints on jerk and snap and, therefore, the acceleration of reference trajectories changes instantaneously. The large initial tracking errors seen in Figures 15.C and 20.C should be reduced if the reference trajectories were constrained by defined jerk and snap limits. These limits can be define very naturally by the operational limits of the vehicle.

10 Conclusion

A trajectory optimization framework for autonomous suspended load operations is proposed. The iteration-based differential dynamic programming algorithm was used to provide guidance to an existing unmanned vehicle system. Using a simplified, but representative, model ensured that the optimization procedure was not overly complex but still provided effective guidance to the vehicle. Variational integrators were used to propagate and linearize the system model in the DDP algorithm. Previous studies have shown that DDP algorithms utilizing variational integrators are far less affected, when compared to utilizing Euler's methods, by changes to the discretization time step. As a result, the utilized variational integrator enabled the use of a relatively large time step and, therefore, reduced the computational burden of the implemented DDP algorithm. The main contribution of this paper is the demonstration, both in simulation and in flight test, of an on-board and real-time implementable trajectory optimization framework. Simulation studies and a flight test demonstrated the effectiveness of the proposed framework. Unlike previously proposed feedback and input shaping methods, the proposed trajectory optimization framework provides a foundation in which more complex objectives, including state and input constraints, obstacle avoidance, flight time optimization, and aggressive maneuvering, can be studied and implemented.

References

[1] Asseo, S. J. and Whitbeck, R. F., "Control Requirements for Sling-Load Stabilization in Heavy Lift Helicopters," *Journal of the American Helicopter Society*, Vol. 18, No. 3, 1973, pp. 23–31.

[2] Gera, J. and Farmer Jr., S. W., "A Method of Automatically Stabilizing Helicopter Sling Loads," Tech. Rep. TN D-7593, NASA, 1974.

[3] Rosen, A., Ronen, T., and Raz, R., "Active aerodynamic stabilization of a helicopter/sling-load system," *Journal of Aircraft*, Vol. 26, No. 9, 1989, pp. 822–828.

[4] Liu, D. T., "In-Flight Stabilization of Exernally Slung Helicopter Loads," Tech. Rep. NORT-72-39, Northrop Corp, 1973.

[5] Dukes, T. A., "Maneuvering heavy sling loads near hover part I: Damping the pendulous motion," *Journal of the American Helicopter Society*, Vol. 18, No. 2, 1973, pp. 2–11.

[6] Gupta, N. K. and Ryson, A. E., "Near-hover control of a helicopter with a hanging load," *Journal of Aircraft*, Vol. 13, No. 3, 1976, pp. 217–222.

[7] Hutto, A. J., "Flight-Test Report on the Heavy-Lift Helicopter Flight-Control System," *Journal of the American Helicopter Society*, Vol. 21, No. 1, 1976, pp. 32–40.

[8] Ivler, C., *Design and Flight Test of a Cable Angle Feedback Control System for Improving Helicopter Slung Load Operations at Low Speed*, Ph.D. thesis, Stanford University, 2012.

[9] Ivler, C. M., Tischler, M. B., and Powell, J. D., "Cable angle feedback control systems to improve handling qualities for helicopters with slung loads," *AIAA Guidance, Navigation, and Control Conference*, 2011.

[10] Sreenath, K., Lee, T., and Kumar, V., "Geometric Control and Differential Flatness of a Quadrotor UAV with a Cable-Suspended Load," *IEEE Conference on Decision and Control*, 2013, pp. 2269–2274.

[11] Sreenath, K., Michael, N., and Kumar, V., "Trajectory generation and control of a quadrotor with a cable-suspended load - A differentially-flat hybrid system," *IEEE International Conference on Robotics and Automation*, 2013, pp. 4888 – 4895.

[12] Palunko, I., Fierro, R., and Cruz, P., "Trajectory generation for swing-free maneuvers of a quadrotor with suspended payload: A dynamic programming approach," *IEEE International Conference on Robotics and Automation*, 2012, pp. 2691 – 2697.

[13] Palunko, I., Cruz, P., and Fierro, R., "Agile Load Transportation : Safe and Efficient Load Manipulation with Aerial Robots," *IEEE Robotics Automation Magazine*, Vol. 19, No. 3, 2012, pp. 69–79.

[14] Palunko, I. and Fierro, R., "Adaptive control of a quadrotor with dynamic changes in the center of gravity," *IFAC World Congress*, 2011, pp. 2626–2631.

[15] Faust, A., Palunko, I., Cruz, P., Fierro, R., and Tapia, L., "Automated aerial suspended cargo delivery through reinforcement learning," *Artificial Intelligence*, in press, available online.

[16] Faust, A., Malone, N., and Tapia, L., "Preference-balancing Motion Planning under Stochastic Disturbances," *IEEE International Conference on Robotics and Automation*, 2015, pp. 3555 – 3562.

[17] Bernard, M., Kondak, K., and Hommel, G., "A Slung Load Transportation System Based on Small Size Helicopters," *Autonomous Systems - Self-Organization, Management, and Control*, edited by B. Mahr and S. Huanye, 2008, pp. 49–61.

[18] Bisgaard, M., la Cour-Harbo, A., and Bendtsen, J. D., "Full State Estimation for Helicopter Slung Load System," *AIAA Guidance, Navigation and Control Conference*, 2007.

[19] Bisgaard, M., la Cour-Harbo, A., Johnson, E. N., and Bendtsen, J. D., "Vision aided state estimator for helicopter slung load system," *IFAC Symposium on Automatic Control in Aerospace*, 2007, pp. 425–430.

[20] Singhose, W., *Command Generation for Flexible Systems*, Ph.D. thesis, Massachusetts Institute of Technology, 1997.

[21] Singer, N., Singhose, W., and Kriikku, E., "An Input Shaping Controller Enabling Cranes to Move Without Sway," *ANS Topical Meeting on Robotics and Remote Systems*, 1997.

[22] Khalid, A., Huey, J., Singhose, W., Lawrence, J., and Frakes, D., "Human Operator Performance Testing Using an Input-Shaped Bridge Crane," *Journal of Dynamic Systems, Measurement, and Control*, Vol. 128, No. 4, 2006, pp. 835–841.

[23] Bisgaard, M., la Cour-Harbo, A., and Bendtsen, J. D., "Input shaping for helicopter slung load swing reduction," *AIAA Guidance, Navigation and Control Conference*, 2008.

[24] Ottander, J. A. and Johnson, E. N., "Precision slung cargo delivery onto a moving platform," *AIAA Modeling and Simulation Technologies Conference*, 2010.

[25] Bisgaard, M., la Cour-Harbo, A., and Bendtsen, J. D., "Swing damping for helicopter slung load systems using delayed feedback," *AIAA Guidance, Navigation and Control Conference*, 2009.

[26] McGonagle, J. G., "The design, test, and development challenges of converting the K-MAX helicopter to a heavy lift rotary wing UAV," *American Helicopter Society Forum*, 2001.

[27] Colucci, F., "Evolving Autonomy Unmanned K-MAX Points the Way," *Vertiflite*, 2013.

[28] Head, E., "Production Potential," *Vertical Magazine*, 2014.

[29] Jacobson, D. H. and Mayne, D. Q., *Differential Dynamic Programming*, Elsevier, 1970.

[30] Todorov, E. and Li, W., "A generalized iterative LQG method for locally-optimal feedback control of constrained nonlinear stochastic systems," *American Control Conference*, 2005, pp. 300–306.

[31] Li, W. and Todorov, E., "Iterative linear quadratic regulator design for nonlinear biological movement systems," *International Conference on Informatics in Control, Automation, and Robotics*, 2004, pp. 222–229.

[32] Morimoto, J. and Atkeson, C. G., "Minimax Differential Dynamic Programming: An Application to Robust Biped Walking," *International Conference on Intelligent Robots and Systems*, 2003, pp. 1927–1932.

[33] Lantoine, G. and Russell, R. P., "A hybrid differential dynamic programming algorithm for robust low-thrust optimization," *AAS/AIAA Astrodynamics Specialist Conference and Exhibit*, 2008.

[34] Yakowitz, S. J., "The stagewise Kuhn-Tucker condition and differential dynamic programming," *IEEE Transactions on Automatic Control*, Vol. 31, No. 1, 1986, pp. 25–30.

[35] De La Torre, G. and Theodorou, E., "Stochastic Variational Integrators for System Propagation and Linearization," *IMA Conference on Mathematics of Robotics*, 2015.

[36] Hairer, E., Lubich, C., and Wanner, G., *Geometric numerical integration: structure-preserving algorithms for ordinary differential equations*, Springer, 2006.

[37] Marsden, J. E. and West, M., "Discrete Mechanics and Variational Integrators." *Acta Numerica*, Vol. 10, 2001, pp. 357–514.

[38] Johnson, E. R., Schultz, J., and Murphey, T. D., "Structured Linearization of Discrete Mechanical Systems for Analysis and Optimal Control," *IEEE Transactions on Automation Science and Engineering*, Vol. 12, No. 1, 2015, pp. 140–152.

[39] Johnson, E. R. and Murphey, T. D., "Scalable Variational Integrators for Constrained Mechanical Systems in Generalized Coordinates," *IEEE Transactions on Robotics*, Vol. 25, No. 6, 2009, pp. 1249–1261.

[40] Schultz, J. and Murphey, T. D., "Trajectory generation for underactuated control of a suspended mass," *IEEE International Conference on Robotics and Automation*, 2012, pp. 123 –129.

[41] Lurie, A. I., *Analytical mechanics*, Springer, 2002.

[42] Ober-Blöbaum, S., Junge, O., and Marsden, J. E., "Discrete mechanics and optimal control: an analysis," *ESAIM: Control, Optimisation and Calculus of Variations*, Vol. 17, No. 2, 2011, pp. 322–352.

[43] West, M., *Variational integrators*, Ph.D. thesis, California Institute of Technology, 2004.

[44] Marsden, J. E. and Ratiu, T. S., *Introduction to mechanics and symmetry: a basic exposition of classical mechanical systems*, Springer, 1999.

[45] Johnson, E., *Trajectory optimization and regulation for constrained discrete mechanical systems*, Ph.D. thesis, Northwestern University, 2012.

[46] Tassa, Y., Erez, T., and Smart, W. D., "Receding horizon differential dynamic programming," *Advances in neural information processing systems*, 2008, pp. 1465–1472.

[47] Kelley, C. T., *Iterative methods for optimization*, SIAM, 1999.

[48] Johnson, E. N. and Schrage, D. P., "The Georgia Tech Unmanned Aerial Research Vehicle: GTMax," *AIAA Guidance, Navigation, and Control Conference*, 2003.

[49] Dittrich, J. S. and Johnson, E. N., "Multi-sensor Navigation System for an Autonomous Helicopter," *Digital Avionics Systems Conference*, 2002, pp. 1–19.

[50] Chowdhary, G., Johnson, E. N., Magree, D., Wu, A., and Shein, A., "GPS-denied Indoor and Outdoor Monocular Vision Aided Navigation and Control of Unmanned Aircraft," *Journal of Field Robotics*, Vol. 30, No. 3, 2013, pp. 415–438.

[51] Wu, A. D., Johnson, E. N., and Proctor, A. A., "Vision-Aided Inertial Navigation for Flight Control," *Journal of Aerospace Computing, Information, and Communication*, Vol. 13, No. 3, 2006, pp. 63–71.

[52] Ludington, B., Johnson, E., and Vachtsevanos, G., "Vision based navigation and target tracking for unmanned aerial vehicles," *Advances in Unmanned Aerial Vehicles*, Springer, 2007, pp. 245–266.

[53] Johnson, E. N. and Kannan, S. K., "Adaptive trajectory control for autonomous helicopters," *Journal of Guidance, Control, and Dynamics*, Vol. 28, No. 3, 2005, pp. 524–538.

[54] Johnson, E. N. and Schrage, D. P., "System integration and operation of a research unmanned aerial vehicle," *Journal of Aerospace Computing, Information, and Communication*, Vol. 1, No. 1, 2004, pp. 5–18.

[55] Johnson, E. N. and Mishra, S., "Flight Simulation for the Development of an Experimental UAV," *AIAA Modeling and Simulation Technologies Conference*, 2002.

[56] Prosser, D. T. and Smith, M. J., "A Novel, High Fidelity 6-DoF Simulation Model for Tethered Load Dynamics," *American Helicopter Society Forum*, 2013.

[57] Simon, D., *Optimal state estimation: Kalman, $H_{infinity}$, and nonlinear approaches*, John Wiley & Sons, 2006.

11 Acknowledgments

This study is funded by the U. S. Army under the Vertical Lift Research Center of Excellence (VLRCOE) program managed by the National Rotorcraft Technology Center, Aviation and Missile Research, Development and Engineering Center under Cooperative Agreement W911 W6-11-2-0010 between the Georgia Institute of Technology and the U. S. Army Aviation Applied Technology Directorate. The authors would like to acknowledge that this research and development was accomplished with the support and guidance of the NRTC. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Aviation and Missile Research, Development and Engineering Center or the U.S. Government.