Nonlinear Model Predictive Control With Neural Network Optimization for Autonomous Autorotation of Small Unmanned Helicopters

Konstantinos Dalamagkidis, *Member, IEEE*, Kimon P. Valavanis, *Senior Member, IEEE*, and Les A. Piegl

***Abstract*—This paper presents a solution to the autonomous vertical autorotation problem of unmanned helicopters using a novel nonlinear model predictive controller (NMPC) enhanced by a recurrent neural network (RNN) that handles the nonlinear optimization. The controller utilizes an internal, nonlinear autorotation model and is capable of handling input and output constraints that directly map to quality, efficiency and safety requirements. The RNN is employed to achieve real-time operation because it is capable of improving convergence performance, especially when hardware that supports task parallelization is available. This is of major importance when dealing with small unmanned helicopters with limited onboard computational power, where high update rates are required to successfully perform the autorotation maneuver. The proposed NMPC/RNN combination signifies the first use of a nonlinear model for online autorotation trajectory optimization, thus allowing for easier adaptation to other helicopter types without the need for retraining as in the case of machine learning techniques used by the state-of-the-art. An additional novelty of this research concerns the use of an objective function designed to eliminate the risk of fatalities to people on the ground. This is in contrast to previous works where the goal was to save the aircraft and is achieved by appropriately lowering the kinetic energy of the helicopter during the last phase of its descent. The paper discusses in detail the general design of the NMPC/RNN, before going into the specifics regarding the derivation, implementation and integration of the autonomous autorotation controller itself. The performance of the latter is validated using extensive simulation results.**

***Index Terms*—Helicopters, public safety, predictive control, recurrent neural networks (RNNs), unmanned systems.**

Nomenclature

A	Rotor disc area.
$\mathcal{C}$	Constraint function.
$C_{d;0}$	Average blade drag coefficient.
$C_{l;a}$	Lift curve slope.
C_Q	Torque coefficient.
C_T	Thrust coefficient.
E	Neural network epoch size.
f_e	Equivalent unit drag coefficient area.
f_g	Ground effect factor.
f_i	Inflow velocity factor.
f_r	Ratio of maximum to nominal rotor rpm.
g	Acceleration of gravity.
I_R	Rotor moment of inertia.
J	Jacobian matrix.
L	Cost function.
l_{TR}	Dinstance between main and tail rotor.
M	Helicopter mass.
N_c	Control horizon.
N_r	Number of constraints.
N_s	Prediction horizon.
P	Rotor power.
R	Rotor radius.
T	Thrust.
t_f	Final time.
t_k	Time constant.
t_s	Sampling period.
u	Action.
v	Velocity.
v_H	Helicopter sink rate.
w	Weight factor.
$\boldsymbol{\chi}$	Auxiliary vector.
x	State.
z	Helicopter altitude.
$\boldsymbol{z}$	Measurement vector.
z_0	Helicopter initial altitude.

Greek Letters

γ	Learning rate.
θ	Blade pitch at 3/4 of its length.
κ	Induced power correction factor.
ρ_α	Air density.

Manuscript received October 01, 2009; revised April 29, 2010; accepted June 03, 2010. Manuscript received in final form June 18, 2010. Date of publication July 26, 2010; date of current version June 17, 2011. Recommended by Associate Editor J. H. Lee. This work was supported by the NSF under Grant IIP-0856311.

K. Dalamagkidis is with the Faculty of Robotics and Embedded Systems, Department of Informatics, Technische Universität München, 85748 Garching bei München, Germany (e-mail: konstantinos.dalamagkidis@in.tum.de).

K. P. Valavanis is with the Electrical and Computer Engineering Department, University of Denver, Denver, CO 80210 USA (e-mail: kimon.valavanis@du.edu).

L. A. Piegl is with the Computer Science and Engineering Department, University of South Florida, Tampa, FL 33620 USA (e-mail: lap@cse.usf.edu).

Digital Object Identifier 10.1109/TCST.2010.2054092

σ	Rotor solidity factor.
Ω	Rotor speed of rotation.
Ω_0	Nominal rotor speed of rotation.

Subscripts

h	At hover.
i	Induced.
MR	Main rotor.
s	Steady-state.
tip	Rotor tip.
TR	Tail rotor.

I. Introduction

AUTOROTATION is a phenomenon that distinguishes helicopters from fixed-wing aircraft and allows them to maintain lift and control in the case of engine failure [1]. Autorotation takes advantage of the energy stored in the main rotor to allow limited control of the descent rate. It is mainly used as an emergency maneuver to bring a helicopter safely to the ground even when it has suffered an engine failure or is experiencing a range of severe problems that do not allow continued safe flight. This maneuver can be accomplished by all helicopters regardless of size and as a result it is of interest as an emergency flight termination system for unmanned helicopters as well.

Methods for the derivation of optimal autorotation trajectories have been proposed in the past and lately autorotation controllers based on machine learning techniques have also been presented. These approaches presented in detail in Section III, have one or more significant drawbacks that don't allow them to be incorporated as they are in current and future aircraft. Black-box approaches give results as good as the training data used and are difficult to reconfigure. On the other hand, the model-based approaches are implemented in an offline optimization fashion and are difficult to adapt to online, real-time operation. The latter is especially true for small unmanned helicopters where on-board processing power may be limited. Moreover all of the aforementioned approaches are designed to minimize damage to the aircraft itself, with no consideration regarding the safety of people on the ground.

The goal of this research is to develop a controller that is capable of performing the autorotation maneuver in small unmanned helicopters. One of the key differences from previous approaches is the controller objective. Whereas in current literature the controller is designed to ensure the survival of the aircraft by minimizing forward velocity and sink rate at zero altitude, in this case the objective is to maintain a kinetic energy below a safety threshold for a range of altitudes before touchdown. This threshold in turn ensures that the probability of fatalities or serious injuries in the case of impact with a person are appropriately minimized. This decision was taken because in the case of unmanned aircraft and especially those for civilian applications, the lack of passengers on-board, shifts the focus of safety to people on the ground. It can then be argued that it is possible and in certain cases preferable to allow an unmanned helicopter to crash, if that would avoid fatalities on the ground [2]. As a result the controller will be designed to minimize risk to people on the ground and secondarily, when possible, save the helicopter as well.

In parallel, to avoid the shortcomings of existing approaches, a novel nonlinear model predictive controller with neural network optimization (NMPC/RNN) is proposed. Such a controller allows online operation in real-time even in cases where computing power is severely limited as is the case with small unmanned helicopters. The use of the neural network for optimization allows a significant level of parallelization in hardware that is capable of supporting it. Moreover, due to the use of model predictive techniques, it is capable of enforcing performance and safety constraints and is robust to environmental disturbances and sensor errors. Being a model-based approach it is also easily reconfigurable for application to different helicopter types, in contrast to controllers based on machine learning techniques that would require retraining.

This paper is organized as follows. Section II describes the design of the controller starting from the prediction part, continuing with a presentation of the neural network design used for optimization and concluding with how these are integrated. Section III details the autorotation maneuver and the history of autonomous autorotation and is followed by a presentation of the vertical autorotation model used. The derivation of the vertical autorotation controller is the subject of Section V and includes the derivation of the internal model used by the controller, of the constraints and of an appropriate cost function. This is followed by Section VI that presents implementation details. Simulation results are given and analyzed in Section VII and this paper concludes with two sections summarizing important conclusions and proposed future improvements, respectively.

II. Controller Design

Key characteristics of the proposed approach include the use of an internal helicopter model, the capability of handling constraints and independent operation from the nominal control system. The use of a model-based approach allows easy reconfiguration for different helicopter types, while the independence from the nominal control system increases reliability. The constraint handling capability is important for operating within mechanical, structural and aerodynamic tolerances, as well as for achieving safety goals. To achieve the characteristics described above, a nonlinear model predictive control (NMPC) approach has been chosen. Furthermore, since on-board processing capacity of small unmanned helicopters is limited, the NMPC is augmented by a recurrent neural network (RNN) that is responsible for the nonlinear optimization. The latter allows meeting real-time performance requirements.

The idea behind model predictive control (MPC) in general is to start with a fixed prediction horizon (N_s), using the current state of the plant as the initial state. An optimal control sequence of length N_c $N_s \geq N_c$ is then obtained that minimizes a cost function over the prediction horizon, while at the same time satisfying posed constraints. After applying the first element of that sequence as an input to the plant, the new state is observed. The prediction horizon is then moved one step forward and the process is repeated.

The problem of obtaining the control sequence is a constrained optimization problem which has to be solved every time a new state is observed, something that can introduce significant computational burden. As a result early applications of MPC were limited to process control of plants with slow dynamics that allowed sampling times in the order of minutes. Nevertheless faster computers have allowed the use of MPC in other fields including aviation. MPC has been applied to the control of an F-16 [3], as well as of a Boeing 747 freighter aircraft under failures [4]. Additionally, Keviczky and Balas proposed an MPC for guidance control of a UAS [5], while Slegers *et al.* used MPC to control an unmanned parafoil and an autonomous glider [6]. Another application relating to unmanned aircraft where MPC has also been proposed, is trajectory planning under constraints and disturbances [7], [8].

The operation of the NMPC/RNN can be divided into two parts; the *prediction* where the future state sequence is calculated based on an initial control sequence and the *optimization* where the control sequence is updated to reduce the cost function $L(\boldsymbol{x}, \boldsymbol{u})$. This process is repeated using the updated control sequence until an optimal control sequence is obtained. The design of these two parts is presented in the following sections.

A. Prediction

A general single-input–multiple-output (SIMO), nonlinear, constrained, affine in the control problem with nonlinear cost function can be expressed in discrete time as

$$\boldsymbol{x}(t+1) = \boldsymbol{x}(t) + t_s \boldsymbol{f}(\boldsymbol{x}(t)) + t_s \boldsymbol{g}(\boldsymbol{x}(t)) u(t) \tag{1}$$

$$\min_{u(i)\in U} \sum_{i=0}^{t_f} L(\boldsymbol{x}(i), u(i)) \text{ s.t. } \mathcal{C}_j(\boldsymbol{u}, \boldsymbol{x}) \leq 0, \forall j \in [0, N_r] \tag{2}$$

where $x \in R^n$. The equations above imply discretization using Taylor series expansion with truncation in the first term. The latter was chosen to avoid increasing the computational complexity. The use of $\mathcal{C}(\boldsymbol{u}, \boldsymbol{x}) \leq 0$ does not impose any restriction on the form of the constraints, but is rather used to allow direct mapping to the way constraints are handled by the RNN.

The future state sequence can be obtained directly from (1). Nevertheless the value of $\mathrm{d}L/\mathrm{d}\boldsymbol{u}$ must also be calculated to be used for the optimization. Differentiating (1) with respect to a control action $u(i)$

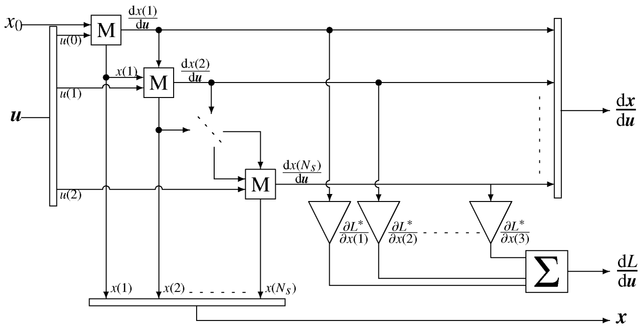
$$\begin{aligned}\frac{\mathrm{d}\boldsymbol{x}(k+1)}{\mathrm{d}u(i)} &= [I + t_s J_f(\boldsymbol{x}(k)) + t_s J_g(\boldsymbol{x}(k)) u(k)] \\ &\quad \times \frac{\mathrm{d}\boldsymbol{x}(k)}{\mathrm{d}u(i)} + t_s g(\boldsymbol{x}(k)) \frac{\mathrm{d}u(k)}{\mathrm{d}u(i).}\end{aligned} \tag{3}$$

Since actions can only affect future states

$$\frac{\mathrm{d}\boldsymbol{x}(k)}{\mathrm{d}u(i)} = 0 \quad \forall x(k), u(i) : i \geq k \tag{4}$$

and do not depend on past or future actions

$$\frac{\mathrm{d}u(k)}{\mathrm{d}u(i)} = \begin{cases} 1, & \text{if } i = k \\ 0, & \text{otherwise.} \end{cases} \tag{5}$$



x_0
$\boldsymbol{u}$
$u(0)$
$u(1)$
$u(2)$
M
M
M
$\frac{\mathrm{d}x(1)}{\mathrm{d}\boldsymbol{u}}$
$\frac{\mathrm{d}x(2)}{\mathrm{d}\boldsymbol{u}}$
$\frac{\mathrm{d}x(N_S)}{\mathrm{d}\boldsymbol{u}}$
$x(1)$
$\frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}\boldsymbol{u}}$
$\frac{\partial L^*}{\partial x(1)}$
$\frac{\partial L^*}{\partial x(2)}$
$\frac{\partial L^*}{\partial x(3)}$
Σ
$\frac{\mathrm{d}L}{\mathrm{d}\boldsymbol{u}}$
$x(1)$
$x(2)$
$x(N_S)$
$\boldsymbol{x}$

Fig. 1. Block diagram of the prediction module of the controller. On the left half a cascaded connection is used to calculate $\mathrm{d}x(i)/\mathrm{d}\boldsymbol{u}$ from $\mathrm{d}x(i-1)/\mathrm{d}\boldsymbol{u}$. The summation block is initialized to the value of $2w\boldsymbol{u}$.

Using (3) and taking into account (4) and (5), the following update rule for calculating the $\mathrm{d}\boldsymbol{x}(t)/\mathrm{d}u(i)$ from the previous prediction step is obtained

$$\frac{\mathrm{d}\boldsymbol{x}(k+1)}{\mathrm{d}u(i)} = \begin{cases} 0, & \text{if } k+1 \\ t_s g(\boldsymbol{x}(k)), & \text{if } k+1 = i \\ K(k) \frac{\mathrm{d}\boldsymbol{x}(k)}{\mathrm{d}\boldsymbol{u}(i)}, & \text{otherwise} \end{cases} \tag{6}$$

where $K(k)$ is given by

$$K(k) = I + t_s J_f(\boldsymbol{x}(k)) + t_s J_g(\boldsymbol{x}(k)) u(k). \tag{7}$$

A cost function that separates the contribution of the control sequence from that of the state of the helicopter is assumed

$$L = \sum_{j=1}^{N_s} L^*(\boldsymbol{x}(j)) + w \boldsymbol{u}^T \boldsymbol{u} \tag{8}$$

where L^* is a function of the state and w is a positive weight factor.

The derivative of the cost function is then given by

$$\frac{\mathrm{d}L}{\mathrm{d}\boldsymbol{u}} = \sum_{i=1}^{N_s} \left(\frac{\partial L^*}{\partial \boldsymbol{x}(i)} \frac{\mathrm{d}\boldsymbol{x}(i)}{\mathrm{d}\boldsymbol{u}} \right) + 2w\boldsymbol{u}. \tag{9}$$

Fig. 1 shows a block diagram of the prediction module of the controller. On the left the initial state and the control sequence are used to determine future states and their derivatives with respect to the control sequence in a cascade of model evaluations. Although these evaluations can be parallelized with respect to the control sequence, each step in the state prediction dimension requires the previous to be completed.

B. Nonlinear Optimization

As discussed in Section II-A, the NMPC optimization problem can be expressed by

$$\min_{u(i)\in U} \sum_{i=0}^{t_f} L(\boldsymbol{x}(i), u(i)) \quad \text{s.t. } \mathcal{C}(\boldsymbol{u}, \boldsymbol{x}) \leq 0 \tag{10}$$

where L is the cost function and $\mathcal{C}$ an appropriate expression of the constraints imposed. To solve problems such as the one above, Xia *et al.* have proposed a series of RNNs [9]–[13]. These networks are capable of solving convex, nonlinear optimization problems with linear [11] or nonlinear [10] constraints. The constraints themselves can be modeled as inequality [9], [10], [13], equality [11], or both [12]. Furthermore in the later iterations, the requirement for convexity was relaxed and convergence was guaranteed for a class of non-convex problems as well.

The idea behind their approach is to build a neural network that models an ODE whose equilibrium point is the optimal solution to the problem (1), (2). A significant advantage of this approach is that, it does not require calculation of the Hessian matrix of the Langragian. The latter can be computationally costly.

The state equation of the neural network is given by [13]

$$d\begin{pmatrix}\boldsymbol{u}\\ \boldsymbol{\chi}\end{pmatrix} = \gamma\begin{pmatrix}-\boldsymbol{u}+\left(\boldsymbol{u}-\frac{\mathrm{d}L}{\mathrm{d}\boldsymbol{u}}-\frac{\mathrm{d}\mathcal{C}}{\mathrm{d}\boldsymbol{u}}\boldsymbol{\chi}\right)^{+}\\ -\boldsymbol{\chi}+(\boldsymbol{\chi}-\mathcal{C}(\boldsymbol{u}))^{+}\end{pmatrix} \tag{11}$$

and the output is given by $\boldsymbol{u}$.

In (11), γ is a learning rate constant, $(\cdot)^{+}$ is an activation function and $\boldsymbol{\chi}$ is an auxiliary vector with size equal to the number of constraints. The auxiliary vector is responsible for capturing the violation of constraints and is subsequently used to move the output of the neural network accordingly.

In [13], Xia *et al.* proved that if the problem is convex, then the output of the network converges globally to the optimal solution of the nonlinear optimization problem.

Theorem 1: Let $h(\boldsymbol{u},\boldsymbol{\chi}) = \nabla^2 L(\boldsymbol{u}) + \sum_{i=1}^{N_c}\boldsymbol{\chi}_i\nabla^2\mathcal{C}_i(\boldsymbol{u})$ and a set $S_0 = \{\boldsymbol{\chi}_0 \in R^{N_c} \,|\, \boldsymbol{\chi}_0 = (\boldsymbol{\chi}(t) + e^{t_0-t}\boldsymbol{\chi}_0(t_0))^{+}, t \in [t_0,\infty)\}$. Assume that $h(\boldsymbol{u},\boldsymbol{\chi}^{\star})$ is positive semidefinite on $R_{+}^{N_s}$ and $h(\boldsymbol{u}^{\star},\boldsymbol{\chi}^{\star})$ is positive definite. For any initial point $(\boldsymbol{u}(t_0),\boldsymbol{\chi}(t_0))$, if $h(\boldsymbol{u},\boldsymbol{\chi})$ is positive semidefinite on $R_{+}^{N_s}\times S_0$, then the neural network of (11) is stable at a Karush–Kuhn–Tucker point and its output trajectory converges globally to a minimum solution of the nonlinear optimization problem and the convergence rate is proportional to the design constant γ. This theorem is taken from Theorems 1 and 2 of [13] and is not proven in this paper. A corollary is that convergence is also guarantied regardless of the initial point for convex problems as well. The importance of the Theorem above lies in the fact that it allows the neural network of (11) to be used in a certain class of non-convex problems as well, depending on the form of the constraint function $\mathcal{C}$.

C. Computational Complexity

In this section, the effect of the prediction and control horizons on execution time will be investigated and techniques to improve performance will be presented.

In the prediction step, the calculation of $g(\boldsymbol{x}(k))$ and $K(k)$ does not depend on either the prediction or the control horizons. Assuming that either calculation—including a multiplication with a constant—takes constant time, then the time required for the calculation of $\mathrm{d}\boldsymbol{x}(k)/\mathrm{d}\boldsymbol{u}, k \in [0, N_s)$ will have a computational complexity of $O(N_sN_c)$. The calculation of $\mathrm{d}L/\mathrm{d}\boldsymbol{u}$ is of the same complexity, giving a total execution time for the prediction step of $O(N_sN_c)$.

To calculate the time required for a loop of the RNN, the following assumptions will be made: the number of constraints is N_r and the activation function takes constant time to be evaluated. The execution time for $\mathcal{C}(\boldsymbol{x},\boldsymbol{u})$ and $\mathrm{d}\mathcal{C}/\mathrm{d}u$ is $O(N_r)$ and $O(N_rN_c)$, respectively. For the $\boldsymbol{u}$ calculation, and in addition to the cost of $\mathrm{d}\mathcal{C}/\mathrm{d}\boldsymbol{u}$ and that of the activation function, $N_cN_r + 4N_c$ additions and $N_cN_r + N_c$ multiplications are also required. Similarly for the calculation of $\boldsymbol{\chi}$ the additional cost corresponds to $3N_r$ additions and N_r multiplications. As a result the total execution time of the RNN will be $O(N_rN_c)$. The total execution time for each complete update will be $O(N_sN_c + N_rN_c)$.

This can be contrasted with the computational complexity of gradient descent methods that would feature cubic or even quartic complexity, as shown next. Traditional SQP methods require repeatedly calculating the Hessian matrix of the Langragian and then solving a quadratic program. The Hessian matrix in this case is given by

$$\begin{aligned}\nabla^2\mathcal{L}(\boldsymbol{x},\boldsymbol{u},\lambda) &= \nabla_{\boldsymbol{uu}}\left\{L(\boldsymbol{u},\boldsymbol{x}) - \lambda^T\mathcal{C}(\boldsymbol{u},\boldsymbol{x})\right\}\\ &= \sum_{i=1}^{N_s}\left[\left(\frac{\mathrm{d}\boldsymbol{x}(i)}{\mathrm{d}\boldsymbol{u}}\right)^T(\nabla_{\boldsymbol{xx}}L^*)\left(\frac{\mathrm{d}\boldsymbol{x}(i)}{\mathrm{d}\boldsymbol{u}}\right)\right]\\ &\quad+\sum_{i=1}^{N_s}\left[\sum_{j=1}^{n}\left(\frac{\partial L^*}{\partial x_j}\nabla_{\boldsymbol{uu}}x_j(i)\right)\right]\\ &\quad+\sum_{i=1}^{N_r}\lambda_i\nabla_{\boldsymbol{uu}}\mathcal{C}_i(\boldsymbol{u},\boldsymbol{x})\end{aligned} \tag{12}$$

where λ is the Langragian multiplier. Assuming that $\nabla_{\boldsymbol{xx}}L^*$ can be calculated in constant time, the first term requires $2N_s$ matrix multiplications that can be completed in $O({N_s}^4)$ time. Similarly, assuming constant time for calculation of the Hessian matrix of $\boldsymbol{x}$, with respect to $\boldsymbol{u}$, the second term is calculated in $O(N_s)$ time. Finally, given that $\nabla_{\boldsymbol{uu}}\mathcal{C}_i(\boldsymbol{u},\boldsymbol{x})$ is calculated in $O({N_c}^2)$ time, the last term requires $O(N_r{N_c}^2)$. After the aforementioned matrix has been calculated, a quadratic program of size N_c with N_r inequality constraints needs to be solved, requiring $O((N_c+N_r)^3)$ time. Even when quasi-Newton approximations are used instead of the actual Hessian, the complexity is only reduced to cubic. As a result, the neural network has a distinct advantage over traditional techniques in terms of execution time and offers at the same time guarantied convergence in the case of convex problems.

It is noteworthy that, since the calculation of $\mathrm{d}\boldsymbol{x}(k)/\mathrm{d}u(i)$ is independent with respect to i, if the hardware supports parallelization the execution time can be reduced to $O(N_s)$. The same holds for the calculation of the terms of the $\mathrm{d}L/\mathrm{d}\boldsymbol{u}$, reducing the prediction step to $O(N_s)$ time. Similar improvements can be accomplished in the RNN, by calculating the elements of the $\boldsymbol{u}$ and $\boldsymbol{\chi}$ vectors in parallel, resulting in an overall execution time complexity of only $O(N_s + N_r)$. This is important due to the limited time available for convergence and the limited computational power.

Other possible optimizations include the offline precomputation of certain terms and skipping certain operations by observing from (6) that $\mathrm{d}\boldsymbol{x}/\mathrm{d}\boldsymbol{u}$ is a triangular matrix. These approaches allow reduction of the associated time constant and improved execution speed.

D. Integration

Putting together the recurrent neural network described above with the traditional nonlinear model predictive controller, we arrive at the NMPC/RNN that was used in this paper. The block

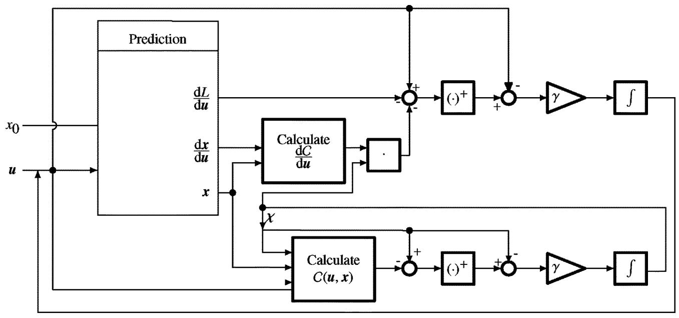


Fig. 2. Block diagram of the NMPC/RNN. The left part concerns the model-based prediction, while on the right a recurrent neural network is used for nonlinear optimization. With the exception of the prediction block, the other operations can be run in parallel.

diagram of the entire NMPC/RNN is provided in Fig. 2. The right part of the NMPC/RNN implements the recurrent neural network used for nonlinear optimization. The top path updates the control sequence while the lower is used to update the auxiliary vector $\boldsymbol{\chi}$ mapping the constraints.

The NMPC/RNN requires the calculation of the $\mathrm{d}L/\mathrm{d}\boldsymbol{u}$, $\mathcal{C}(\boldsymbol{u})$, and $\mathrm{d}\mathcal{C}/\mathrm{d}\boldsymbol{u}$ quantities, that happens in the left half of the NMPC/RNN. This involves the prediction block and the two constraint handling blocks. It should be noted that this distinction is made only for presentation purposes since it is possible to handle the constraints in parallel with the prediction, as values become available.

The algorithm summarizing the controller is given in Fig. 3. The inner loop corresponds to the NMPC/RNN calculations to determine the optimal control sequence and executes E times. On the other hand, the outer loop executes when a new state is observed and until the helicopter reaches the ground. Inside the inner loop, a smaller prediction loop is executed that calculates incrementally the $\boldsymbol{x}$, $\mathrm{d}\boldsymbol{x}/\mathrm{d}\boldsymbol{u}$, $\mathrm{d}L/\mathrm{d}\boldsymbol{u}$, $\mathcal{C}(\boldsymbol{u})$, and $\mathrm{d}\mathcal{C}/\mathrm{d}\boldsymbol{u}$ quantities. This approach was chosen to lower the amount of memory required to execute the calculations. Specifically only the most current value of $\boldsymbol{x}$, $\mathrm{d}\boldsymbol{x}/\mathrm{d}\boldsymbol{u}$, and $\mathrm{d}L/\mathrm{d}\boldsymbol{u}$ is kept.

It should also be noted that the number of repetitions of the inner loop need not necessarily be constant. An alternative design may be used that continuously updates the control sequence until a new state is observed or the neural network has converged.

As discussed in the previous section, several of the calculations required by the NMPC/RNN can be executed in parallel thus allowing higher update rates. Fig. 4 provides an overview of how the algorithm of Fig. 3 can be executed on hardware that supports parallelization or a special-purpose chip.

III. Autorotation

Due to the aerodynamics of the main rotor, even when no power is supplied to it, it is still possible to maintain a steady rate of descent. This is accomplished by using the air flowing through the rotor disk to rotate the main rotor—the reverse process from normal flight. In this case the main rotor acts as a parachute, breaking the helicopter sink rate, while maintaining near-nominal rpm. This is known as autorotative flight or simply autorotation. Before landing the helicopter pilot will raise collective and exchanging rotor rpm for a reduction in

```
foreach New sensor data (z) received do Outer loop
    Filter ($\boldsymbol{z}$) to obtain an estimate on current state $\boldsymbol{x}(0)$
    for $k \leftarrow 0$ to $E-1$ do Inner loop
        $\frac{\mathrm{d}L}{\mathrm{d}\boldsymbol{u}} \leftarrow w\boldsymbol{u}^T\boldsymbol{u}$
        for $i \leftarrow 0$ to $N_s$ do
            for $j \leftarrow 0$ to $N_c - 1$ do
                if $i = j$ then
                    $\frac{\mathrm{d}\boldsymbol{x}(i)}{\mathrm{d}\boldsymbol{u}(j)} \leftarrow t_s g(\boldsymbol{x})$
                end
                else if $i > j$ then
                    $\frac{\mathrm{d}\boldsymbol{x}(i)}{\mathrm{d}\boldsymbol{u}(j)} \leftarrow K(i)\frac{\mathrm{d}\boldsymbol{x}(i-1)}{\mathrm{d}\boldsymbol{u}(j)}$
                end
            end
            $\boldsymbol{x} \leftarrow \boldsymbol{x} + t_s[f(\boldsymbol{x}) + g(\boldsymbol{x})\boldsymbol{u}]$
            $\frac{\mathrm{d}L}{\mathrm{d}\boldsymbol{u}} \leftarrow \frac{\mathrm{d}L}{\mathrm{d}\boldsymbol{u}} + \frac{\mathrm{d}L}{\mathrm{d}\boldsymbol{x}}\frac{\mathrm{d}\boldsymbol{x}(i)}{\mathrm{d}\boldsymbol{u}}$
            Calculate $\mathcal{C}$
            for $j \leftarrow 1$ to $N_c - 1$ do
                Calculate $\frac{\mathrm{d}\mathcal{C}}{\mathrm{d}\boldsymbol{u}(j)}$
            end
        end
        $\boldsymbol{\chi} \leftarrow \boldsymbol{\chi} + \gamma\left[-\boldsymbol{\chi} + (\boldsymbol{\chi} - \mathcal{C}(\boldsymbol{u}))^+\right]$
        $\boldsymbol{u} \leftarrow \boldsymbol{u} + \gamma\left[-\boldsymbol{u} + \left(\boldsymbol{u} - \frac{\mathrm{d}L}{\mathrm{d}\boldsymbol{u}} - \frac{\mathrm{d}\mathcal{C}}{\mathrm{d}\boldsymbol{u}}\boldsymbol{\chi}\right)^+\right]$
    end
    Send control command $\boldsymbol{u}(0)$
    $\boldsymbol{u} \leftarrow [\boldsymbol{u}(1)\ \boldsymbol{u}(2)\ \ldots\ \boldsymbol{u}(N_c-1)\ \boldsymbol{u}(N_c-1)]$
end
```

Fig. 3. NMPC/RNN algorithm. On line 3 the updating may be allowed to continue until new sensor data are available or the network has converged.

sink rate, a procedure known as flare. The flare enables the helicopter to land safely, at sink rates that are within mechanical and human tolerances.

During autorotation the helicopter sink rate and rotor rpm will reach a steady-state value. At this state the energy added to the rotor from the air flowing through the rotor disk exactly balances the energy lost to overcome blade drag. If the autorotative sink rate increases beyond the value corresponding to the current rotor rpm, the rotor rpm will also increase and vice versa. It should be noted that the autorotative sink rate can be quite high, thus necessitating the use of the flare.

The concept of autorotation was known for years before the first successful demonstrations carried out in the late 1930's [1], [14], [15]. Nevertheless, a systematic study to optimize the performance of the maneuver was not done until much later. In 1977, Johnson derived a nonlinear autorotation model that includes vertical as well as longitudinal movement [16]. The optimal control was derived using a cost function that minimized horizontal and vertical speed at touchdown. The derivation of the control law was based on iterative numerical integration forwards and then backwards between the two boundary points and updating using the steepest descent method. The results were then compared with the performance of a modified Bell OH-58A carrying a High Energy Rotor System (HERS) during flight tests. These tests were performed in 1976 as part of a Bell Helicopter Company research project sponsored by the U.S. Army [17].

A few years later in Stanford, Allan Yeow-Nam Lee improved on Johnson's work by introducing state inequality constraints that were converted to equality using slack variables [18]. The controller was derived by numerical parameter optimization using the Sequential Gradient Restoration technique (an iterative method).

	1	2	3	4	5	6	7	8	9
1	$\boldsymbol{x}(1)$								
2	$\boldsymbol{x}(2)$	$\frac{\mathrm{d}\boldsymbol{x}(1)}{\mathrm{d}\boldsymbol{u}}$							
3	$\boldsymbol{x}(3)$	$\frac{\mathrm{d}\boldsymbol{x}(2)}{\mathrm{d}\boldsymbol{u}}$	$\frac{\partial L^*}{\partial \boldsymbol{x}(1)} \frac{\mathrm{d}\boldsymbol{x}(1)}{\mathrm{d}\boldsymbol{u}}$	$\mathcal{C}(0)$	$\mathcal{C}(N_c)$	$\frac{\mathrm{d}\mathcal{C}(0)}{\mathrm{d}\boldsymbol{u}}$	$\frac{\mathrm{d}\mathcal{C}(N_c)}{\mathrm{d}\boldsymbol{u}}$		
$\vdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$		
N_c+1	$\boldsymbol{x}(N_c+1)$	$\frac{\mathrm{d}\boldsymbol{x}(N_c)}{\mathrm{d}\boldsymbol{u}}$	$\frac{\partial L^*}{\partial \boldsymbol{x}(N_c-1)} \frac{\mathrm{d}\boldsymbol{x}(N_c-1)}{\mathrm{d}\boldsymbol{u}}$	$\mathcal{C}(N_c-1)$	$\mathcal{C}(2N_c-1)$	$\frac{\mathrm{d}\mathcal{C}(N_c-1)}{\mathrm{d}\boldsymbol{u}}$	$\frac{\mathrm{d}\mathcal{C}(2N_c-1)}{\mathrm{d}\boldsymbol{u}}$		
N_c+2	$\boldsymbol{x}(N_c+2)$	$\frac{\mathrm{d}\boldsymbol{x}(N_c+1)}{\mathrm{d}\boldsymbol{u}}$	$\frac{\partial L^*}{\partial \boldsymbol{x}(N_c)} \frac{\mathrm{d}\boldsymbol{x}(N_c)}{\mathrm{d}\boldsymbol{u}}$						
$\vdots$	$\cdots$	$\cdots$	$\cdots$						
N_s	$\boldsymbol{x}(N_s)$	$\frac{\mathrm{d}\boldsymbol{x}(N_s-1)}{\mathrm{d}\boldsymbol{u}}$	$\frac{\partial L^*}{\partial \boldsymbol{x}(N_s-2)} \frac{\mathrm{d}\boldsymbol{x}(N_s-2)}{\mathrm{d}\boldsymbol{u}}$						
N_s+1		$\frac{\mathrm{d}\boldsymbol{x}(N_s)}{\mathrm{d}\boldsymbol{u}}$	$\frac{\partial L^*}{\partial \boldsymbol{x}(N_s-1)} \frac{\mathrm{d}\boldsymbol{x}(N_s-1)}{\mathrm{d}\boldsymbol{u}}$						
N_s+2			$\frac{\partial L^*}{\partial \boldsymbol{x}(N_s)} \frac{\mathrm{d}\boldsymbol{x}(N_s)}{\mathrm{d}\boldsymbol{u}}$						
N_s+3								$\boldsymbol{u}$	χ

Fig. 4. Overview of the calculations in the NMPC/RNN loop that can run in parallel. Each box can be executed in $O(1)$ time. Columns signify computations running in parallel, specifically nine parallel computation streams. To execute the computations of each row, the results of the previous row must first be available. This means that the entire loop requires $O(N_s)$ time.

Although Johnson and Lee derived optimal autorotation trajectories, the issue of autonomous autorotation was not addressed until almost two decades later, specifically in 2004. In Japan, Hazawa *et al.* [19] derived two autorotation models, one nonlinear using first principles, and one linear. A PI controller was then used to land a small unmanned helicopter in simulation.

In a continuation of the work of Johnson and Lee, Aponso *et al.*, presented their own method to optimize the trajectory and control inputs for a full-size helicopter during an autorotation landing [20]. The model used was the same to that Johnson derived and the optimization problem was solved using sequential quadratic programming. The goal of their work was to provide autorotation guidance to ensure the survival of sensitive sensors and data stored on-board the helicopter in the case of non-catastrophic failures. Because of the mismatch between model and simulation, a flare law was necessary, that forces the flare to occur at 30 ft. Their work was evaluated against a high fidelity Bell 206 simulator.

During 2008, two groups published research results on autonomous autorotation, both using machine learning techniques. In the first approach [21], the controller was trained using prerecorded pilot reference autorotations that provided a model of the aircraft and the "ideal" trajectory. The landing itself was achieved by forcing the helicopter to hover at 0.5 m. The performance of the controller was demonstrated using a small unmanned helicopter (XCell Tempest).

The second approach [22] was a straightforward application of reinforcement learning to train a controller using the Johnson model, cost function and experimental data. The final state-action space has 10 dimensions and was covered using radial basis functions whose parameters were updated using backpropagation. After 9000 epochs the number of radial basis functions was about 19 000 and the success rate around 80%.

IV. Vertical Autorotation Model

It is common during autorotation of manned helicopters to maintain some longitudinal velocity because it allows better view of the landing location and provides the pilot with some additional reaction time. Nevertheless it can be argued that in the case of unmanned helicopters, a better policy would be to minimize the volume of airspace the helicopter has to travel through and in effect minimize the risk for collision with other aircraft in the area. As a result the controller will attempt a purely vertical autorotation. The vertical autorotation model is based on the derivations in [16], but modified to include the induced velocity dynamics as modeled in [1] and a non-unitary ground effect factor. The resulting model equations are given by

$$\dot{v_H} = g - \frac{\rho_\alpha A R^2 \Omega^2}{2M} f_g \sigma C_{l,a} \left[\frac{\theta}{3} - \frac{v_i - v_H}{2\Omega R}\right] - \frac{\rho_\alpha f_e}{2M} {v_H}^2 \tag{13}$$

$$\dot{\mathsf{z}} = -v_H \tag{14}$$

$$\dot{\Omega} = -\frac{\rho_\alpha A R^3 \Omega^2}{I_R} \frac{v_i - v_H}{R\Omega} f_g \sigma C_{l,a} \left[\frac{\theta}{3} - \frac{v_i - v_H}{2\Omega R}\right] - \frac{\rho_\alpha A R^3 \Omega^2}{8 I_R} \sigma C_{d,0} \tag{15}$$

$$\dot{v_i} = -\frac{2.356}{R}\left({v_i}^2 - {v_{i,s}}^2\right). \tag{16}$$

The controlled variable above is the main rotor blade pitch θ, also known as collective. The term $v_{i,s}$ refers to the steady-state induced velocity which is calculated as the product of the induced velocity at hover $(v_{i,h})$ and an induced velocity factor (f_i). The latter is given by [1]

$$f_i = \begin{cases} \kappa\frac{v_H}{2v_{i,h}} - \kappa\sqrt{\left(\frac{v_H}{2v_{i,h}}\right)^2 - 1}, & 2 \le \frac{v_H}{v_{i,h}} \\ \kappa + k_1\frac{v_H}{v_{i,h}} + k_2\left(\frac{v_H}{v_{i,h}}\right)^2 & \\ \quad +k_3\left(\frac{v_H}{v_{i,h}}\right)^3 + k_4\left(\frac{v_H}{v_{i,h}}\right)^4, & \text{otherwise} \end{cases} \tag{17}$$

where $k_1 = 1.125$, $k_2 = -1.372$, $k_3 = 1.718$, $k_4 = -0.655$.

The first case refers to the windmilling state where the air flow through the rotor is smooth, while the second term concerns an empirical relationship for the induced velocity in the turbulent wake and vortex ring states. Although alternate empirical relationship do exist, this one was chosen because it provides continuous derivatives.

Because the rotor wake meets the ground the pressure below the rotor rises, a phenomenon known as ground effect [23]. This increase in pressure results in a higher thrust generation for the same power, cushioning the touchdown especially in unpowered landings [24] and is taken into account in the model by the f_g parameter. An empirical model for ground effect and the one used in this paper is given by

$$\left[\frac{T}{T_\infty}\right]_{P=\text{const}} = \frac{1}{1-\left(\frac{R}{\mathsf{z}}\right)^2}. \tag{18}$$

Finally, the term $v_{i,h}$ is calculated as [1]

$$v_{i,h} = \sqrt{\frac{Mg}{2\rho_\alpha A}}. \tag{19}$$

V. Autorotation Controller Derivation

The first step in the design of the model predictive controller is the derivation of an appropriate internal model. The simulation model presented in Section IV is not suitable since it assumes knowledge of states that are not observable. The following section describes the simplified internal model used by the NMPC/RNN. This is followed by the derivation of the constraints and of an appropriate cost function.

A. Internal Vertical Autorotation Model

The use of the autorotation model derived in Section IV is not possible without certain modifications. Since the inflow dynamics are generally not measurable during flight, the model used internally by the controller will be simplified to include only the observable states, namely v_H, z and Ω. As a result, the controller assumes a steady-state inflow velocity, that is

$$v_i = v_{i,s} = v_{i,h} f_i. \tag{20}$$

In addition to that and in order to keep the model equations relatively simple and the computational complexity low, the ground effect is considered negligible, i.e., $f_g = 1$.

To avoid numerical issues, scaling is also necessary. Specifically the control input is scaled to the $[0,1]$ range, while the model equations are nondimensionalized by dividing lengths by the radius R of the rotor and velocities by the tip velocity $(v_{\text{tip}} = \Omega R)$ and then scaling

$$\tau = \frac{\Omega_0 t}{100} \Rightarrow \frac{\mathrm{d}}{\mathrm{d}t} = \frac{\Omega_0}{100}\frac{\mathrm{d}}{\mathrm{d}\tau} \tag{21}$$

$$x_1 = \frac{100 v_H}{\Omega_0 R} \Rightarrow v_H = \frac{\Omega_0 R}{100} x_1 \tag{22}$$

$$x_2 = \frac{\mathsf{z}}{10R} \Rightarrow \mathsf{z} = 10R x_2 \tag{23}$$

$$x_3 = \frac{\Omega}{\Omega_0} \Rightarrow \Omega = \Omega_0 x_3 \tag{24}$$

$$x_4 = \frac{100 v_{i,s}}{\Omega_0 R} \Rightarrow v_{i,s} = \frac{\Omega_0 R}{100} x_4 \tag{25}$$

$$u = \frac{\theta + \theta_{\min}}{\theta_{\max} - \theta_{\min}} \Rightarrow \theta = u(\theta_{\max} - \theta_{\min}) + \theta_{\min}. \tag{26}$$

The final controller model equations are

$$\begin{aligned} \dot{x}_1 = {} & \frac{10^4}{{\Omega_0}^2 R} g - \frac{\rho_\alpha f_e R}{2M} x_1^2 + \frac{25\rho_\alpha AR\sigma C_{l,a}}{M} x_3(x_4 - x_1) \\ & - \frac{29.1\rho_\alpha AR\sigma C_{l,a}\theta_{\min}}{M} x_3^2 \\ & - \frac{29.1\rho_\alpha AR\sigma C_{l,a}(\theta_{\max} - \theta_{\min})}{M} x_3^2 u \end{aligned} \tag{27}$$

$$\dot{x}_2 = -\frac{1}{10} x_1 \tag{28}$$

$$\begin{aligned} \dot{x}_3 = {} & -\frac{100\rho_\alpha AR^3\sigma C_{d,0}}{8I_R} x_3^2 + \frac{\rho_\alpha AR^3\sigma C_{l,a}}{400 I_R}(x_4 - x_1)^2 \\ & - \frac{2.91\rho_\alpha AR^3\sigma C_{l,a}\theta_{\min}}{10^3 I_R}(x_4 - x_1)x_3 \\ & - \frac{2.91\rho_\alpha AR^3\sigma C_{l,a}(\theta_{\max} - \theta_{\min})}{10^3 I_R}(x_4 - x_1)x_3 u. \end{aligned} \tag{29}$$

The vertical autorotation model described by (27)–(29) can be discretized to be used in the NMPC/RNN.

B. Constraints

There are three types of constraints imposed on the controller. The first concerns the physical limits of the actuator

$$\theta_{\min} \le \theta \le \theta_{\max} \Rightarrow 0 \le \boldsymbol{u} \le 1. \tag{30}$$

This type of constraint is easily handled by appropriate design of the neural network activation function. In this case the activation function is a saturation function that limits the input to the range $[0,1]$, so that the control sequence is always within the actuator limits

$$(\cdot)^+ = \min(1, \max(\cdot, 0)). \tag{31}$$

The second constraint is incorporated to avoid blade stall. Assuming that the thrust required to keep the helicopter in the area is constant, as the rpm drops the blade pitch needs to be adjusted to compensate. Nevertheless there is a limit after which the air flow separates and lift is lost. This is typically modeled using the blade loading coefficient (C_T/σ) which can take values up to 0.12–0.14 without stalling [1]. In this case a conservative limit of 0.125 was chosen

$$C_T \le \frac{\sigma}{8} \Rightarrow \frac{1}{6}\sigma C_{l,a}\left[u(i)[\theta_{\max}-\theta_{\min}]+\theta_{\min} - \frac{3}{200}\frac{x_4(i)-x_1(i)}{x_3(i)}\right] \le \frac{\sigma}{8}, \quad \forall i: 1 \le i \le N_s. \tag{32}$$

The last constraint is designed to protect the main rotor from mechanical stress. This is because for very low blade pitch the rotor angular velocity may increase above nominal, possibly damaging the rotor assembly. The constraint is typically expressed as a function of the nominal rotor rpm

$$\Omega \le f_r\Omega_0 \Rightarrow x_3 \le f_r \tag{33}$$

where from literature f_r typically takes values between 1.05 and 1.25 [20], [25].

A problem occurs when using the Ω constraint in the case where the helicopter's sink rate exceeds the autorotative sink rate, that is the sink rate where no energy is added or subtracted from the main rotor. At this state if collective is suddenly increased the rotor rpm will show an increase instead of decreasing. The reason is that the inflow ratio will be negative and the collective dependent term in the torque balance will be positive and higher than the torque losses. This results in a $\mathrm{d}\mathcal{C}/\mathrm{d}\boldsymbol{u}$ that has elements with positive sign. As the main rotor approaches the rpm limit, the controller will accurately predict the violation of the constraint, but the commanded action will be to decrease collective instead of increasing it.

To accommodate this issue an inequality constraint on the rpm rate of change is introduced. Specifically when the rpm approaches the rpm upper limit, the rate of change is restricted to zero or negative values. As a consequence the controller needs to raise the collective and reduce the helicopter sink rate to the autorotative value.

The constraints in vector form are given by

$$\mathcal{C}(\boldsymbol{u}) = \begin{bmatrix} \left[C_T(i) - \frac{\sigma}{8}\right]_{1\le i\le N_c} \\ \left[\dot{\Omega}(j)\right]_{1\le j\le N_c} \end{bmatrix}. \tag{34}$$

In (34), C_T and $\dot{\Omega}$, can be substituted using (32) and (15), respectively. Finally the $\mathrm{d}\mathcal{C}/\mathrm{d}\boldsymbol{u}$ needs to be calculated by differentiating (34) with respect to the control sequence.

C. Cost Function

In manned aircraft the objective is to minimize the risk to the platform itself, which implies minimization of the risk to people onboard. Nevertheless the goal of an autorotation controller for unmanned helicopters is to minimize the risk to people on the ground. This is because these helicopters are unoccupied and their survival is considered secondary.

The risk to people on the ground can be quantified using the probability of fatality after a helicopter ground impact. This probability is generally difficult to calculate because of the large number of factors that need to be taken into consideration and

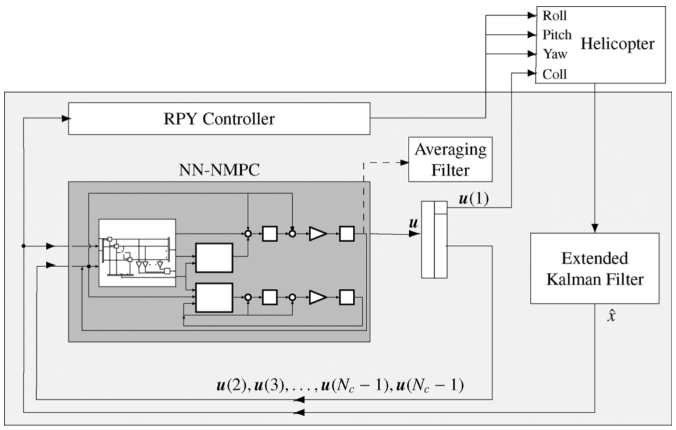


Fig. 5. Block diagram of the vertical autorotation controller. The controller is comprised of the NMPC/RNN which is responsible for collective, the RPY-Controller for roll, pitch and yaw as well as an EKF to filter the information sent by the sensors.

because of the scarcity of actual experimental data. Nevertheless it has been repeatedly suggested that it can be expressed as a function of the kinetic energy on impact [2], [26]–[28], although other parameters such as the presence of obstacles and the geometry of the aircraft may also play a role.

Unfortunately, there is no consensus in the literature on how this relationship/function is best defined. According to study results presented in RCC323 [27], a 1 lb object with kinetic energy of 50 J has a probability of causing a fatality of 10%, while for more than 200 J that probability rises to above 90%. According to study results presented in RCC321 [26], the corresponding kinetic energy estimates for an impact of a 1000 lb object to the torso are approximately 1.2 and 3.5 kJ, respectively, a difference of at least an order of magnitude from the previous model. These differences can be attributed to the fact that kinetic energy does not correlate well with fatality probabilities estimated from accident data [26]. As a result, impact of objects of different mass can have different effects, even if the kinetic energy imparted at impact is the same.

The strictest limit was proposed in [26] where it was argued that direct impact of debris with kinetic energy equal to or less than 15 J will not result in fatalities. As a result, the goal of the autonomous autorotation controller will be set to maintain the kinetic energy of the helicopter below this limit, at least during the last phase of the descent where the risk of impact with people is high.

Based on the aforementioned arguments, the cost function chosen is given by

$$L^* = \begin{cases} 0.1(v_H - 1.25\mathrm{z} - 0.1)^2, & v_H - 1.25\mathrm{z} \ge 0.1 \\ 0, & \text{otherwise.} \end{cases} \tag{35}$$

This function is designed to introduce large penalties for high sink rates during the last phase of the descent. Specifically the penalty is introduced when the numeric value of the sink rate in ms^{-1} is higher than that of the altitude in m when the latter is increased by 25%. In effect this should reduce the sink rate to about 3 ms^{-1} at an altitude of 2.5 m and to 0.1 ms^{-1} during touchdown. Since the helicopter weighs approximately 3 kg,

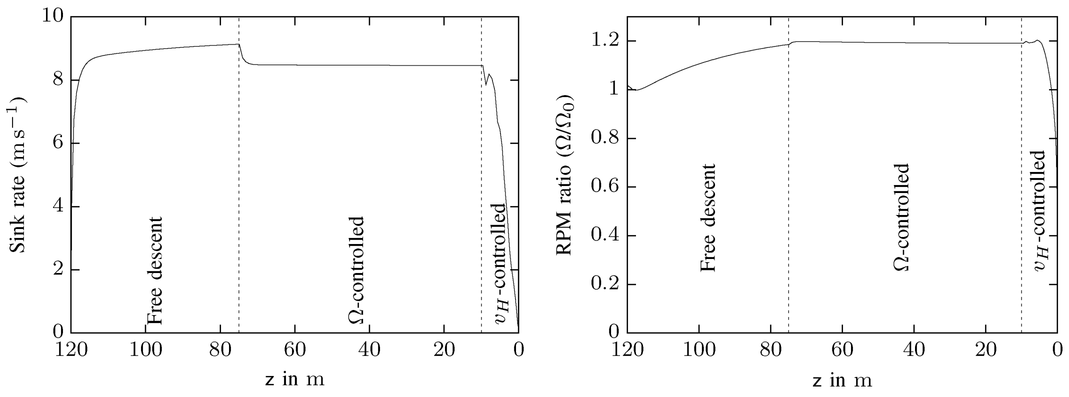


Fig. 6. During the autorotative descent the helicopter traverses three distinct regions of operation; free, Ω-controlled and v_H-controlled descent.

this ensures that the kinetic energy remains below the 15 J during the last 2.5 m of the descent, the latter chosen to include the height of most of the population. The scaling constant was added after trial and error to reduce the magnitude of the cost and constraint derivatives.

VI. Implementation Details

The block diagram of the vertical autorotation controller is given in Fig. 5. This diagram includes two additional modules; the RPY controller and an EKF filter. Since the NMPC/RNN was designed to provide only the collective pitch, an additional component required is the RPY controller that determines the cyclic and tail rotor pitch commands necessary to maintain the aircraft level, with constant heading and to keep it from drifting. This controller can either be the nominal flight controller of the helicopter or an independent module implemented in the vertical autorotation controller. In the first case, the tail-rotor pitch must be calculated separately. This is because the input to the tail rotor is normally used to counter the moment generated by the main rotor on the helicopter fuselage and keep the latter from spinning. When the helicopter is in autorotation this moment changes and a new tail-rotor thrust coefficient needs to be calculated using

$$C_{T\,\mathrm{TR}} = \frac{\Omega_{\mathrm{MR}}^2 R_{\mathrm{MR}}^5}{\Omega_{\mathrm{TR}}^2 R_{\mathrm{TR}}^4} \frac{C_{Q\,\mathrm{MR}}}{l_{\mathrm{TR}}}. \tag{36}$$

The tail-rotor thrust coefficient can then be easily converted to the tail-rotor pitch and the command sent to the helicopter.

The information from the sensors is filtered using the second of the aforementioned modules, an extended Kalman filter (EKF) that utilizes the internal, nonlinear model of vertical autorotation. This allows the removal of sensor noise, while reusing the calculations carried out in the prediction step of the NMPC/RNN. As a result of the latter the contribution of the EKF to the time required for a complete loop is minimal.

There are two distinct loops in the controller. The first is internal to the NMPC/RNN and is used to optimize the control sequence for the current state. The update rate used is determined by the epoch size parameter (E) which corresponds to the number of iterations of the neural network. In the simulations detailed in Section VII, it is executed with rates up to 3 kHz.

The second loop runs at a significantly lower rate (10–20 Hz) and includes the helicopter. In each iteration of the outer loop it first samples the state of the neural network which corresponds to the (near-)optimal control sequence. The first element of that sequence is sent to the helicopter, while the rest is fed back to be used as the initial neural network state for the next optimization epoch. At the same time the new state of the helicopter is observed, filtered using the EKF and fed back to the prediction block of the NMPC/RNN.

VII. Simulation Results

The proposed controller is tested in simulation using a model of the Thundertiger Raptor 30v2, a popular, gas-powered, remote-controlled helicopter. It has a payload capacity of more than 1 kg and run-time of about 10–20 min depending on payload and flight type (hovering, way-point navigation or aggressive maneuvering). This helicopter and others of similar size and characteristics are often used for robotic research and development. Despite its small size it can still possess a significant amount of kinetic energy on impact to pose a risk to people especially if the sink rate is not kept in check. The values of the Thundertiger Raptor 30v2 model parameters used for the simulations are summarized in Table I.

Before going into the simulation results and to facilitate later discussion, a typical trajectory is shown in Fig. 6. It is evident from that figure that the helicopter traverses three different regions of operation. In the first region it is descending freely gaining speed. In the second region, the Ω-controlled, the helicopter has reached the rpm limit and the sink rate is adjusted so that no further increase in rpm occurs. Finally in the last region or v_H-controlled, the sink rate is the controlled variable. This distinction is important because the controller behaves differently from region to region and especially in the transition between regions.

The baseline scenario involves a descent from an initial altitude of 120 m using perfect sensor information. The simulation is performed at an update rate of 1 kHz while the controller is run at 10 Hz. The neural network parameters are $E = 150$ and $\gamma = 0.05$. The resulting trajectory for $N_s = 4$ and $N_c = 3$ is presented in Fig. 7.

As is evident in Fig. 7, in the beginning the helicopter free falls, but quickly reaches the rpm limit. Then the controller reduces the sink rate to the autorotative value for that rpm, or about 6.9 ms^{-1}. At an altitude of about 9 m, collective is increased and the sink rate starts to decrease. The helicopter reaches the 3

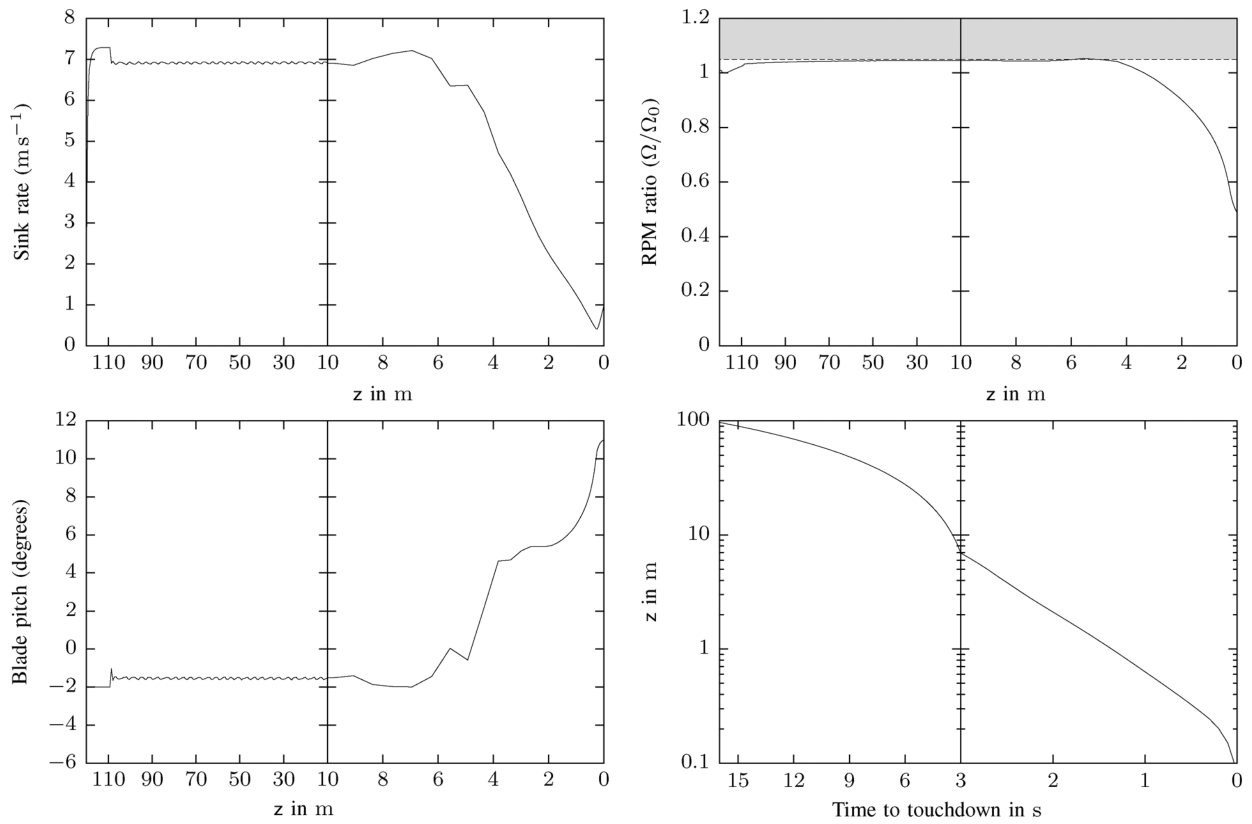


Fig. 7. During the autorotative descent the helicopter traverses three distinct regions of operation; free, Ω-controlled, and v_H-controlled descent.

TABLE I
MODEL PARAMETERS FOR THE THUNDERTIGER RAPTOR 30V2 R/C MODEL HELICOPTER. THE PARAMETERS WERE DERIVED USING DIRECT MEASUREMENT, MANUFACTURER DATA, AND EXPERT ESTIMATES

Helicopter mass (M)	$3\,\mathrm{kg}$
Main rotor moment of inertia (I_R)	$0.03\,\mathrm{kg\,m^2}$
Solidity factor (σ)	0.0455
Main rotor disc radius (R)	$0.62\,\mathrm{m}$
Main rotor mean drag coefficient ($C_{d,0}$)	0.0085
Main rotor lift coefficient ($C_{l,a}$)	$5.84\,\mathrm{rad^{-1}}$
Equivalent unit drag coefficient area (f_e)	$0.03\,\mathrm{m^2}$
Induced power correction factor (κ)	1.15
Nominal main rotor speed (Ω_0)	1,800 rpm
Air density (ρ_α)	$1.225\,\mathrm{kg\,m^{-3}}$
Minimum main rotor pitch	$-6°$
Maximum main rotor pitch	$12°$
Maximum main rotor rpm	$1.05\Omega_0$

ms^{-1} sink rate limit at an altitude of approximately 2.5 m and 2 s later touches down with a sink rate of 0.8 ms^{-1}. None of the constraints are violated. During the last 0.5 m the helicopter increases its sink rate from about 0.3 to 0.8 ms^{-1} because the thrust coefficient limit has been reached. Increasing the E parameter was found to have no observable effect on the resulting trajectory.

A. Real-Time Performance

A very important metric in this problem is the execution speed of the controller. A test was carried out using a single-core Athlon XP 3200+ (model of 2005) CPU, throttled to a frequency of 1 GHz running a 32-bit version of Debian Linux. No parallelization or offline optimization was used to improve the execution time. The execution time measured is 190 μs per iteration. Based on this and at a controller update rate of 10 Hz, more than 500 iterations are possible. Similarly if $E = 150$, a maximum update rate of 35 Hz can be achieved. The cost of the EKF was also calculated under the same conditions and was found to be approximately 128 μs or roughly equivalent to a single iteration of the NMPC/RNN.

B. Nonlinear Optimization Problem Convexity

The nonlinear optimization problem is not guarantied to be convex. In fact, high prediction horizons may lead to non-convex problems. Additionally, the same issue appears near regions where the helicopter transitions operating modes, e.g., when abruptly reducing its descent rate. On the other hand, the latter occurs for a small number of iterations, where suboptimal control does not significantly affect the helicopter trajectory. Since it is possible to enter regions where the problem is non-convex, random sampling was used to determine the maximum allowable prediction horizon. Specifically 500 samples where taken over the entire action-space and for different combinations of t_s and N_c, totaling 28 000 samples. For each sample the highest value of N_s (up to 100) where the $\mathrm{d}^2L^*/\mathrm{d}\boldsymbol{u}^2$ remains positive definite was recorded. This N_s represents a conservative limit on the longest prediction horizon available. This is because it is possible that even after momentarily entering a non-convex region, at the next epoch the problem will be convex again. Furthermore as discussed in Section II-B, the neural network is capable of global convergence for a class of non-convex problems as well.

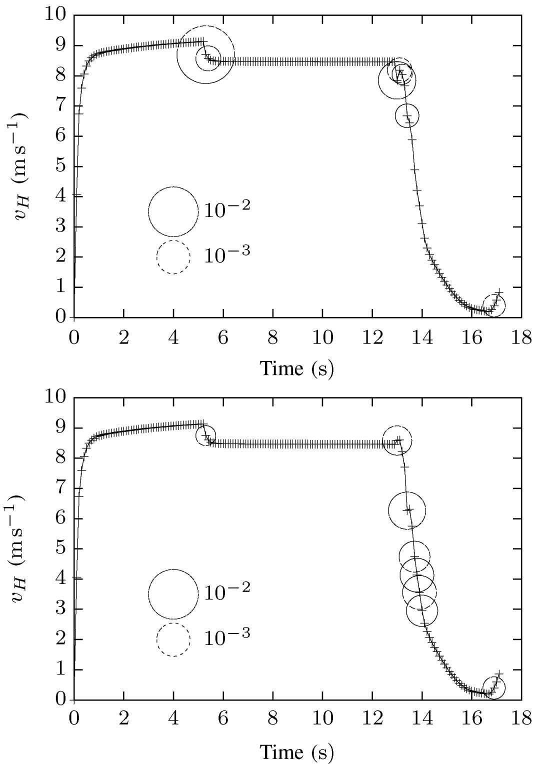


Fig. 8. Variance of the neural network output during the last 20% of the epoch as a function of sink rate and time. Larger circles mean larger variance while variances smaller than 10^{-5} are not shown. The top graph shows the results for $\gamma = 0.05$ while the bottom shows how variance increases for $\gamma = 0.11$.

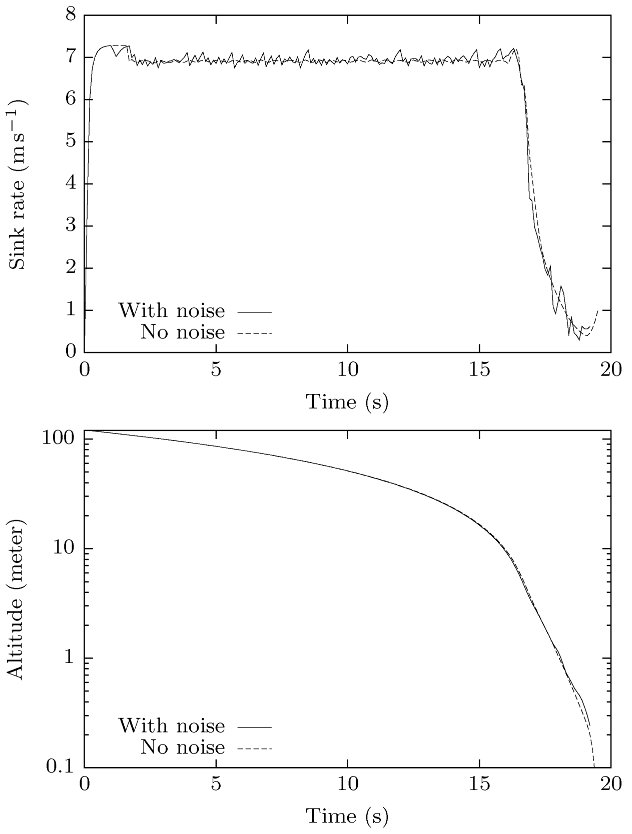


Fig. 9. Trajectory of the helicopter with and without noise. Although the controller is forced to make more corrections because of the noise, the resulting trajectory remains very close to the noise-free scenario.

The results showed that when the sampling period, t_s is lower than 50 ms, the maximum prediction horizon is not very sensitive with respect to the control horizon selected. Furthermore, the worst prediction horizons where found for high sink rates. This is exacerbated by correspondingly high action values that would normally violate the thrust coefficient constraint. The worst-case N_s was found to be 5, which is higher than the prediction horizon used. If the sampling period is lowered to $t_s =$ 10 ms the worst case N_s was found to be closer to 10 and most of the samples did not reach the non-convex region even for $N_s = 100$.

C. Convergence

Examining the evolution of the neural network output at each epoch, it was found that in most cases the collective pitch converged to 10^{-4} of a degree within the first 10 to 20 iterations. The exceptions to this rule concerned cases where the helicopter was in transition between the operating regimes described in Fig. 6. This resulted because of the significant change in output required to satisfy constraints and minimize the cost function but affected at most three successive epochs. This is evident in Fig. 8 where the variance of the network output during the last 20% of each epoch is presented. Although in most cases the variance is below 10^{-5} indicating convergence, when the collective is raised first to comply with the Ω-constraint and then to reduce the value of the cost function, some variance remains which indicates that the RNN may not have converged. The neural network also didn't fully converge during one of the last few epochs before touchdown and this is because in that case the blade stall limit was reached and the thrust coefficient constraint was enforced. Fig. 8 also shows results for higher learning rate γ, where convergence improved in the transition between free descent and Ω-controlled descent, but deteriorated in the v_H-controlled region. As a result learning rate scheduling may be needed to optimize performance over the entire autorotation maneuver.

D. Noise

The effect of sensor noise on controller performance is investigated using zero-mean, Gaussian noise with varying standard deviation. Specifically, the noise of the nondimensionalized sink rate, altitude and rpm measurements was chosen to be 0.2, 0.25, and 0.01 respectively. This corresponds to a standard deviation of 0.24, 1.55, and 18 in the respective actual measurements.

Fig. 9 shows the simulation results with and without noise. The final trajectory is almost the same. Nevertheless, a lot of controller correction affecting the sink rate is evident in the region where the helicopter is trying to maintain the rpm below the constraint. This occurs because of the design of the controller that tries to maintain constant rpm. Additionally at the last phase of the descent, the v_H and z errors affect the cost function and it is derivative, resulting in large corrections.

E. Initial Altitude

Fig. 10 presents the resulting trajectories using different initial altitudes. For initial altitudes exceeding 30 m the helicopter will have time to reach the limit of the rotor rpm allowable and

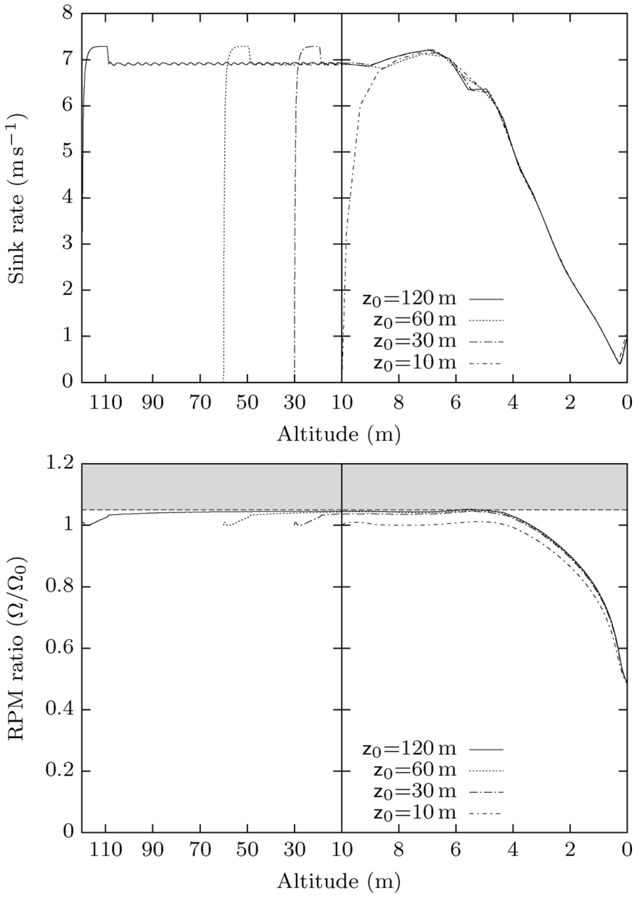


Fig. 10. Sink rate and rotor rpm of the Thundertiger Raptor 30v2 for a descent from an initial altitude of 120 m ($N_s = 4, N_c = 3$). The shaded region in the Ω graph represents the posed constraint. The scale on the right side of the graphs has been altered to present the last stage of the descent in higher detail.

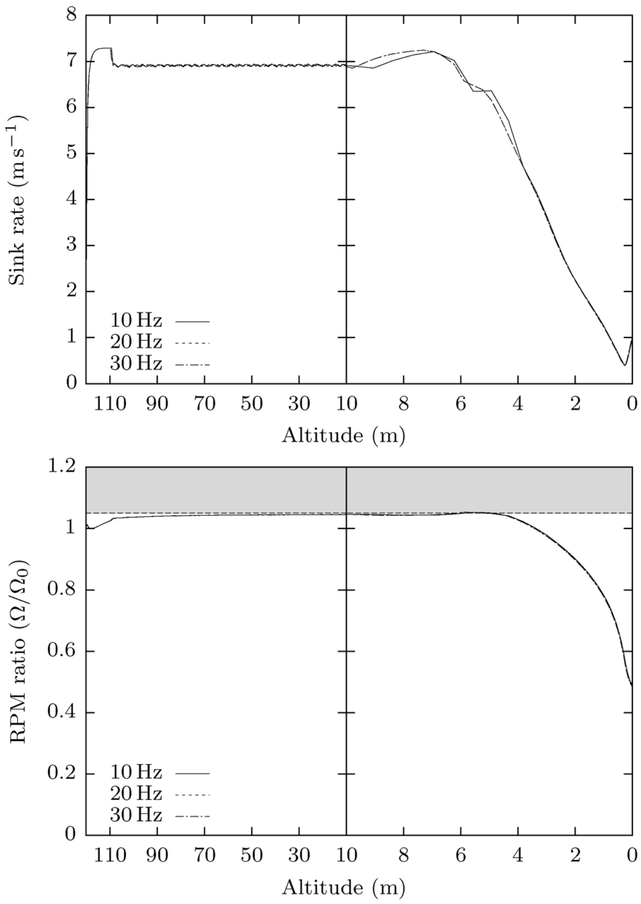


Fig. 11. Controller performance as a function of sampling rate.

as a result the trajectory after that point will be the same. Conversely in situations were the initial altitude is lower than 30 m the helicopter will have to halt the descent rate at lower rpm. Nevertheless, this does not significantly affect the resulting trajectory. In the case where the helicopter was initially at 10 m, the trajectory of the sink rate from 8 m until touchdown is very close to that encountered in cases where the helicopter started from higher altitudes. The only difference is that because of the lower rpm available on the rotor, the stall limit is reached earlier and the sink rate is close to 1 ms^{-1} at touchdown.

F. Sampling Rate

The controller was also tested to determine the performance improvement of higher update rates. Specifically the baseline 10 Hz is compared to the trajectories obtained using 20 and 30 Hz. It is obvious from Fig. 11 that the increase in update rate has minimal effect and only in the transition from the Ω-controlled to the v_H-controlled region. This is because the higher rate allows more frequent sensor measurements in a region of operation where the dynamics are fast. Nevertheless, higher update rates are to be preferred when possible since they can reduce the effect of noise. Although higher update rates reduce the amount of time available to the network to converge, in this application and with under the prediction and control horizons tested it is not a significant problem since epochs of size 150 can be achieved even under 30 Hz.

G. Reaction Time

The delay between failure onset and detection can significantly impact the rotor energy available during the flare. Although pilot reaction time to such events is typically in the range of 1–2 s, health monitoring systems can reduce this interval to sub-second duration. The effect of this delay was tested using up to 3 s delays. During the period between failure onset and detection, the controller maintains the blade pitch required for hovering, resulting in rapid loss of rpm. The results show that the delay time has a similar effect with starting from a lower initial altitude. The rotor speed reaches maximum allowed rpm at lower altitudes depending on the delay. Nevertheless the trajectories after that and until touchdown are almost identical.

VIII. Conclusion

This paper presented a novel control design for performing autonomous autorotation with small unmanned helicopters. The controller presented, features unique characteristics such as configurability, real-time operation and constraint handling using an aerodynamics-based internal model of the helicopter. Computational complexity is kept low and execution times can be further improved through parallelization and offline optimizations. Simulation results demonstrate that autonomous autorotation can be used to safely land an unmanned helicopter even in the presence of noise and regardless of the initial state of the helicopter.

A feature that sets this particular controller apart is the use of an objective function aimed to primarily reduce the risk of people on the ground and secondarily minimize the damage to the helicopter itself. In fact it was possible to maintain a kinetic energy below 15 J during the last phase of the descent, which is a conservative limit on the energy required to cause a fatality. This design choice has far-reaching implications since it significantly increases the safety performance of small unmanned helicopters, which in turn allows operations with less restrictions and in the future will simplify integration into the National Airspace System.

IX. Future Work

Although the performance of the controller was satisfactory, in the future possible improvements will be investigated. As discussed earlier the Ω-constraint is enforced by putting limits on the values taken by $\dot{\Omega}$ which may negatively affect performance especially in the presence of significant noise. Alternatively the use of v_H or a combination of Ω and v_H may be used to enforce this constraint. Another improvement may result from the introduction of constraints on the rate of change of the control sequence. Although this will affect the computational complexity, it will also allow the use of higher learning rates and improve convergence time. Other possible improvements include the use of different types of prediction models, learning rate scheduling, cost function changes and improved constraint handling in an effort to improve performance both in terms of accuracy as well as speed.

Other planned future research includes the derivation of necessary or at least sufficient conditions for guarantied convergence. These conditions will invariably be a function of both the helicopter itself, as well as MPC parameters such as the prediction and control horizon (N_s, N_c). An investigation of how this research can benefit manned helicopters is also underway. In the latter case the presence of the pilot must also be taken into account and the system should be modified to provide assistance rather than take over control, which should be reserved for only extreme cases.

The longterm goal is to integrate this controller into a complete fault-tolerant control architecture that will enable helicopters to fail gracefully even when continued flight is no longer possible. This will require tighter coupling with sensor information processing as well as the mission and path planning subsystems. Specifically the proposed system will feature fault detection and identification (FDI), high level Contingency Planning (CP) as well as basic way-point navigation modules as shown in Fig. 12. The FDI module is responsible for isolating failures that require an emergency landing and is capable of both independent operation as well as communicating with other health monitoring services on-board the helicopter. The CP module on the other hand is responsible for determining the remaining functionality of the helicopter and selecting an appropriate plan of action. Several alternatives are available in the case of emergencies; immediate vertical autorotation, delayed autorotation after finding a suitable landing location, controlled crash or returning to base. The latter alternatives will need to make use of the basic way-point navigation facility, especially if the one on the nominal flight controller is not available. The proposed system is designed to have its independent CPU, memory and power supply. This will eliminate the risk of failures originating in the nominal flight control architecture influencing the capability of the helicopter to handle emergencies.

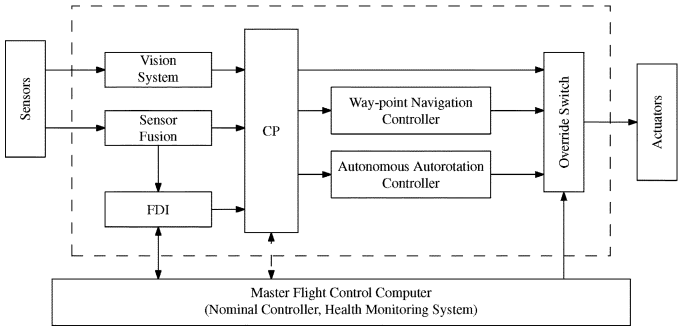

Sensors
Vision System
Sensor Fusion
FDI
CP
Way-point Navigation Controller
Autonomous Autorotation Controller
Override Switch
Actuators
Master Flight Control Computer
(Nominal Controller, Health Monitoring System)

Fig. 12. Block diagram of the proposed emergency landing system. The CP module is central in integrating the information from the vision system, other sensors and the FDI module as well as defining way-points and allocating control to the emergency controllers.

References

[1] J. G. Leishman, *Principles of Helicopter Aerodynamics*, ser. Cambridge Aerospace Series, 2nd ed. Cambridge, MA: Cambridge Univ. Press, 2006.

[2] K. Dalamagkidis, K. Valavanis, and L. Piegl, *On Integrating Unmanned Aircraft Systems Into the National Airspace System: Issues, Challenges, Operational Restrictions, Certification, and Recommendations*, ser. Intelligent Systems, Control and Automation: Science and Engineering. New York: Springer-Verlag, 2009, vol. 36.

[3] R. Bhattacharya, G. J. Balas, M. A. Kaya, and A. Packard, "Nonlinear receding horizon control of an F-16 aircraft," *J. Guid. Control Dyn.*, vol. 25, no. 5, pp. 924–931, 2002.

[4] D. A. Joosten and T. J. J. van den Boom, "Computationally efficient use of MPC and dynamic inversion for reconfigurable flight control," presented at the AIAA Guid., Nav., Control Conf. Exhibit, Honolulu, HI, 2008.

[5] T. Keviczky and G. J. Balas, "Software-enabled receding horizon control for autonomous unmanned aerial vehicle guidance," *J. Guid. Control Dyn.*, vol. 29, no. 3, pp. 680–694, 2006.

[6] N. Slegers, J. Kyle, and M. Costello, "Nonlinear model predictive control technique for unmanned air vehicles," *J. Guid. Control Dyn.*, vol. 29, no. 5, pp. 1179–1188, 2006.

[7] A. Richards and J. How, "Decentralized model predictive control of cooperating uavs," in *Proc. 43rd IEEE Conf. Decision Control (CDC'04)*, 2004, pp. 4286–4291.

[8] Y. Kuwata, T. Schouwenaars, A. Richards, and J. How, "Robust constrained receding horizon control for trajectory planning," in *Proc. AIAA Guid., Nav. Control Conf. Exhibit*, 2005, pp. 4482–4487.

[9] Y. Xia, H. Leung, and J. Wang, "A projection neural network and its application to constrained optimization problems," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 49, no. 4, pp. 447–458, Apr. 2002.

[10] Y. Xia and J. Wang, "A recurrent neural network for nonlinear convex optimization subject to nonlinear inequality constraints," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 51, no. 7, pp. 1385–1394, Jul. 2004.

[11] Y. Xia and J. Wang, "A recurrent neural network for solving nonlinear convex programs subject to linear constraints," *IEEE Trans. Neural Netw.*, vol. 16, no. 2, pp. 379–386, Mar. 2005.

[12] Y. Xia, G. Feng, and M. Kamel, "Development and analysis of a neural dynamical approach to nonlinear programming problems," *IEEE Trans. Autom. Control*, vol. 52, no. 11, pp. 2154–2159, Nov. 2007.

[13] Y. Xia, G. Feng, and J. Wang, "A novel recurrent neural network for solving nonlinear optimization problems with inequality constraints," *IEEE Trans. Neural Netw.*, vol. 19, no. 8, pp. 1340–1353, Aug. 2008.

[14] S. S. McGowen, *Helicopters: An Illustrated History of Their Impact*. Santa Barbara, CA: ABC-CLIO, 2005.

[15] K. Munson*, German War Birds: From World War 1 to NATO Ally.*. New York: Sterling Pub Co Inc., 1986.

[16] W. Johnson, "Helicopter optimal descent and landing after power loss," Ames Research Center, National Aeronautics and Space Administration, Moffett Field, CA, NASA TM 73244, May 1977.

[17] L. W. Dooley and R. D. Yeary, "Flight test evaluation of the high inertia rotor system," Bell Helicopter Textron, Fort Eustis, VA, USARTL-TR-79-9, Jul. 1979.

[18] A. Y.-N. Lee, "Optimal landing of a helicopter in autorotation," Ph.D. dissertation, Dept. Aeronaut. Astronaut., Stanford Univ., Stanford, CA, Jul. 1985.

[19] K. Hazawa, J. Shin, D. Fujiwara, K. Igarashi, D. Fernando, and K. Nonami, "Autonomous autorotation landing of small unmanned helicopter," *Trans. Japan Soc. Mech. Eng. C*, vol. 70, no. 698, pp. 2862–2869, 2004.

[20] B. Aponso, E. Bachelder, and D. Lee, "Automated autorotation for unmanned rotorcraft recovery," presented at the AHS Int. Specialists' Meet. Unmanned Rotorcraft, Chandler, AZ, 2005.

[21] P. Abbeel, A. Coates, T. Hunter, and A. Y. Ng, "Autonomous autorotation of an RC helicopter," presented at the 11th Int. Symp. Experimental Robot., Athens, Greece, Jul. 2008.

[22] D. J. Lee, H. Bang, and K. Baek, "Autorotation of an unmanned helicopter by a reinforcement learning algorithm," presented at the AIAA Guid., Nav., Control Conf. Exhibit, Honolulu, HI, Aug. 2008.

[23] J. Seddon*, Basic Helicopter Aerodynamics*, ser. AIAA education series. Oxford, U.K.: Blackwell Scientific Publications, 1990.

[24] A. Gessow and G. C. Myers*, Aerodynamics of the Helicopter*. New York: F. Ungar Pub. Co., 1967.

[25] A. Y.-N. Lee, "Optimal autorotational descent of a helicopter with control and state inequality constraints," *J. Guid. Control Dyn.*, vol. 13, no. 5, pp. 922–924, Sep. 1990.

[26] Range Safety Group, Range Commanders Council, White Sands Missile Range, NM, "Common risk criteria standards for national test ranges: Supplement," Supplement to document 321-07, Jun. 2007.

[27] Range Safety Group, Range Commanders Council, White Sands Missile Range, NM, "Range safety criteria for unmanned air vehicles—Rationale and methodology supplement," Supplement to document 323-99, Dec. 1999.

[28] R. E. Weibel, "Safety considerations for operation of different classes of unmanned aerial vehicles in the national airspace system," Master's thesis, Dept. Aeronaut. Astronaut., Massachusetts Inst. Technol., Cambridge, Jun. 2005.

Konstantinos Dalamagkidis (M'06) received the Ph.D. degree in computer science and engineering from the University of South Florida, Tampa, in 2009.

Currently he is working as a Research Associate with the Faculty of Robotics and Embedded Systems, Department of Informatics, Technische Universität München, Munich, Germany. His research focuses on unmanned systems modeling, simulation, and control. He is particularly interested in topics concerning high level planning, operational safety and human/robot interaction. With respect to UAS safety and certification requirements, he is the coauthor of the first book in the area, titled *On Integrating Unmanned Aircraft Systems into the National Airspace System: Issues, Challenges, Operational Restrictions, Certification, and Recommendations* (Springer-Verlag, 2009).

Kimon P. Valavanis (SM'81) received the Ph.D. degree in computer and systems engineering from Rensselaer Polytechnic Institute, Troy, NY, in 1986.

He is Professor and Chair with the Electrical and Computer Engineering Department, and Acting Chair of the Computer Science Department, University of Denver, Denver, CO. His interests are focused in the areas of unmanned systems, distributed intelligence systems, robotics and automation. He has published over 300 book chapters, technical journal/transaction and referred conference papers, and he is the author/coauthor of ten books.

Dr. Valavanis was Editor-in-chief of the Robotics and Automation Magazine, 1996–2005, and currently of the Journal of Intelligent and Robotic Systems. He is also the General Chair of the 2011 IEEE Multi-Conference on Systems and Control. He was a Distinguished Speaker in the Robotics and Automation Society (2003) and a Fellow of the AAAS. He is also a Fulbright Scholar.

Les A. Piegl is a Professor with the Department of Computer Science, Engineering, and Information Systems, University of South Florida, Tampa, where he directs USF's CAD lab. His research centers around geometric computing, software engineering, CAD/CAM, bio-engineering applications, geometric and software aspects of robotics, e-learning and virtual enterprise modeling.

Dr. Piegl is an Honorary Editor of *Computer-Aided Design*, the Founding Editor-in-Chief of *Computer-Aided Design and Applications*, and Subject Editor of the *Journal of Intelligent and Robotic Systems*. He founded the annual CAD conferences and is now Federation Chair of the annual CAD and UAV conferences.