An Integrated Algorithm for Path Planning and Flight Controller Scheduling for Autonomous Helicopters*

M. Egerstedt†
Optimization and Systems Theory
Royal Inst. of Technology
SE-100 44 Stockholm, Sweden

T.J. Koo‡
Department of EECS
University of California
CA 94720, USA

F. Hoffmann§
Department of EECS
University of California
CA 94720, USA

S. Sastry¶
Department of EECS
University of California
CA 94720, USA

Abstract

In this article we investigate how to generate flight trajectories for an autonomous helicopter. Given a set of nominal waypoints we generate trajectories that interpolate close to these points. This path generation is done for two different cases, corresponding to two controllers that either govern position or velocity of the helicopter. Based on a given cost functional, the planner selects the optimal one among these multiple paths. This approach thus provides a systematic way for generating not only the flight path, but also a suitable switching strategy, i.e. when to switch between the different controllers.

1 Introduction

For autonomous mobile robots in general and for aerial based ones in particular, the need to function in a dynamic, changing environment is a crucial and important feature in a successful design. If a robot detects an obstacle, or a helicopter flies too close to the ground, an immediate, appropriate action is required. In a purely reactive control system, this problem can be addressed by introducing an obstacle avoidance behavior based on, for instance, potential field methods (Arkin, 1998; Brooks 1986). However, a control system that is commanded to track reference trajectories, which is the case in this article, has to replan the trajectories on-line. Therefore any solution to the planning problem that relies too heavily on time consuming optimization techniques is likely to run into problems.

In this article we propose a solution to the trajectory planning problem that is based directly on the controller architecture itself. It is reasonable to identify different flight modes such as take-off, cruise, turn and landing, as well as different flight controllers, such as velocity and

*This work was supported by the Army Research Office under grant DAAH 04-96-1-0341, the ONR under grant N00014-97-1-0946, the Deutsche Forschungsgemeinschaft under grant Ho 1790/2-1 and the Swedish Foundation for Strategic Research through its Centre for Autonomous Systems at KTH, Sweden.

†magnuse@math.kth.se
‡koo@eecs.berkeley.edu
§fhoffman@eecs.berkeley.edu
¶sastry@eecs.berkeley.edu

position controllers, dedicated to tracking different reference signals (Shim *et al.*, 1998). Given a set of waypoints defined as states of a linear system (position, velocity and acceleration), the planning task is to generate a trajectory that interpolates among these points, subject to additional smoothness constraints on the path. The waypoints are used as soft constraints in the construction of the path, allowing for a trade off between accuracy and smoothness.

This path generation is done in two different cases, each corresponding to a different controller. A cost functional determines which of the two paths is optimal with respect to smoothness and interpolation accuracy. We thus provide a systematic way for generating not only a feasible flight path, but also a suitable switching strategy, i.e. when to switch between the different controllers. Our approach thus offers the major advantage that it does not only propose reference trajectories, but also advises the control system when to switch between the different controllers.

Our solution is based on techniques from linear optimal control theory. The main idea is to compose the flight path from motion primitives, normally referred to as flight modes, such as take-off and landing. The planner finds an optimal path and decides what controller provides the optimal solution for that path segment.

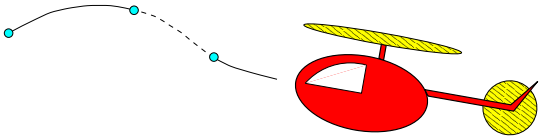


Figure 1: The helicopter tracks a reference path through given way points. Different controllers are active at different parts of the route.

The article is structured as follows: In Section 2 we introduce the helicopter model followed by, in Section 3, a discussion of the switched path planning algorithm. We then conclude with some simulation results, showing the numerical feasibility of our proposed method.

2 The Helicopter Model

In Koo and Sastry (1998), an approximate model of the helicopter dynamics is derived, based on the assumption that some of the cross-coupling terms can be neglected. Under this assumption, the model becomes

$$\begin{aligned} \ddot{P} &= R \begin{pmatrix} 0 \\ 0 \\ -T_M \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \\ \dot{\Theta} &= \Psi\omega \\ \dot{\omega} &= \mathcal{J}^{-1}(\tau - \omega \times \mathcal{J}\omega), \end{aligned} \tag{1}$$

where $P = [x, y, z]$ are the scaled Cartesian coordinates of the helicopter's center of mass and $R \in SO(3)$. Furthermore, Θ are the Euler angles (roll (ϕ), pitch (θ) and yaw (ψ)), ω the helicopter's angular velocity and τ describes the external torque that is applied to the helicopter's center of gravity. This nonlinear, dynamic system is driven by the inputs $[T_M, T_T, a_{1s}, b_{1s}]$, where T_M and T_T are the normalized main and tail rotor thrusts respectively while a_{1s} and b_{1s} are the longitudinal and lateral tilt of the main rotor.

Notice that the model (1) is highly nonlinear. Therefore control theoretical issues such as stability and tracking become much more complicated than for other mobile robots, for which

the dynamics are fairly simple. In the helicopter case, we can not, for instance, let the robot stop and "think" while reasoning about the next action. Since we can not rely on computationally expensive optimization algorithms it seems questionable whether optimal planning for the full dynamic helicopter model is a realistic option. This is something that we need to take into account when our planner is designed.

2.1 Controller Design

In Shim *et al.* (1998), different control designs are investigated, such as linear robust control, fuzzy control, and nonlinear feedback linearization. Due to the fact that the number of states is greater than the number of inputs, different controllers were used to govern specific outputs, such as position or velocity tracking[1]. Therefore the overall behavior of the helicopter mainly depends on the currently active set of controllers. Different flight modes, such as take-off, cruising or hovering, employ their own collection of controllers, and in this article, we show how to select these controllers in a systematic way.

There are a number of reasons for introducing these different flight modes. First of all, flight modes provide building blocks for composing more complex behaviors such as searching or investigating objects on the ground. In addition, behaviors can be decomposed and analyzed in terms of these motion primitives (Tomlin *et al.*, 1998), also used by human pilots. Furthermore, the flight mode approach allows us, at least partially, to decouple the state variables and to guarantee the stability of individual controllers for certain flight envelopes.

From the perspective of this article, on the other hand, we simplify the planning task by partitioning the overall path into smaller segments. This divide and conquer approach to the planning problem significantly reduces the computational complexity of the trajectory generation.

2.2 Differential Flatness

Before we describe the planning task, some comments about differential flatness need to be made. The model (1) is differentially flat, as shown in Koo and Sastry (1998), since it can be feedback linearized. In other words, there exists a diffeomorphism from the states of the nominal trajectory to the states in the helicopter model. If we assume that we fly in the so called *coordinated flight* mode, where we actively keep the side slip angle zero, then the heading of the helicopter can be reconstructed directly from the nominal trajectory. From this, all of the remaining states in the model (1) can be calculated, as shown in Koo and Sastry (1998).

Hence we can recover the states of the helicopter from the flight trajectory and its derivatives, and this is a desired property for two reasons. First of all we want to be able to, given a desired trajectory, use this for controlling all of the states, and thus the flatness property makes it possible for us to obtain the desired state trajectory as well as the nominal inputs of the helicopter. Secondly, we could use this property for imposing constraints on the trajectory, constraints that come from the fact that we only can apply limited inputs in order to avoid saturation.

So, what the flatness property can help us with is to transform the nonlinear, coupled system into a system with decoupled x, y, z-states. This makes it possible for us to view these states as separate when planning the paths, at the same time as we still design output trajectories that are compatible with the nonlinear helicopter dynamics (van Nieuwstadt and Murray, 1999), (under the assumption that we do not saturate the actuators.)

[1]This is due to practical considerations and not a theoretical consequence of the system dynamics.

The way this can be viewed is that the nominal trajectory provides the reference inputs for the actual tracking controller, designed on the full helicopter model. This controller can also be augmented by error feedback if necessary (Koo and Sastry, 1998).

3 Planning for the Switched Control System

Based on the discussion in the previous section, the coarse behavior of the helicopter dynamics can be simplified into a linear system governed by either the position or the velocity controller. Although this simplification can not be used for the controller design itself, it serves as a valid abstraction of the dynamics for planning purposes. The trajectories generated by the planner are then tracked using the controllers, designed with respect to the full, nonlinear helicopter model.

We want the trajectories to have at least continuous second derivatives. Thus the paths fed into the position or velocity controller are produced by control systems on the form

$$\begin{array}{ll} x^{(3)} = k_p(u - x) & \text{position control} \\ x^{(3)} = k_v(u - \dot{x}) & \text{velocity control.} \end{array} \tag{2}$$

The reason for this construction is that if we minimize the L_2-norm of the control input (which will be the case in the next subsection) we get smooth signals. This means that in the position control case, x will stay close to the smooth controlled input. This results in small variations in x as well, which is a desired feature when the position controller is used. The same argument can then be applied in the velocity controller case.

Notice that these linear systems are used for trajectory planning only and not for control. We can thus neglect the stability issues since our control law is designed in a way that drives the linear system from waypoint to waypoint (Egerstedt and Martin, 1998).

By setting $\bar{x} = (x, \dot{x}, \ddot{x})^T$ and using a similar notation in the y- and z-direction, and letting

$$\begin{array}{ll} A_p = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -k_p & 0 & 0 \end{pmatrix}, & b_p = \begin{pmatrix} 0 \\ 0 \\ k_p \end{pmatrix} \\ A_v = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & -k_v & 0 \end{pmatrix}, & b_v = \begin{pmatrix} 0 \\ 0 \\ k_v \end{pmatrix} \end{array} \tag{3}$$

the overall system becomes

$$\begin{pmatrix} \dot{\bar{x}} \\ \dot{\bar{y}} \\ \dot{\bar{z}} \end{pmatrix} = \begin{pmatrix} A\bar{x} + bu_x \\ A\bar{y} + bu_y \\ A\bar{z} + bu_z \end{pmatrix} \quad \text{where} \quad \begin{pmatrix} A \\ b \end{pmatrix} = \begin{cases} \begin{pmatrix} A_p \\ b_p \end{pmatrix} & \text{position control} \\ \begin{pmatrix} A_v \\ b_v \end{pmatrix} & \text{velocity control} \end{cases} \tag{4}$$

This system decouples into three subsystems, which evolve independently in $\mathbb{R}^3$. Even though we are not obliged to switch between position and velocity control simultaneously in all the three subsystems, we prefer to consider them as one system, since the output (either position or velocity) is what the helicopter is asked to track.

3.1 Trajectory Planning

In this subsection, a general framework for generating trajectories for linear, single-input, multiple outputs control systems is presented based on linear optimal control theory. The main idea is to use soft constraints on the position, velocity and acceleration of the system at given, discrete times, $t_i, i = 1, \ldots, m$. The planning task becomes to find the control, u, which drives the individual subsystems of (4) close to the prespecified points in state space, $(\bar{x}_k, \bar{y}_k, \bar{z}_k, t_k)$. Notice that a general interpolation point can be formed by an arbitrary combination of constraints on position, velocity and acceleration along each dimension.

For example, the first subsystem of (4) becomes

$$\dot{\bar{x}} = A\bar{x} + bu, \quad \bar{x} \in \mathbb{R}^3, u \in \mathbb{R}. \tag{5}$$

In addition to the soft constraints, we want to minimize the L_2-norm of the control signal

$$\int_0^T \frac{1}{2} u^2(s) ds \tag{6}$$

because it smoothes the generated path. The flatness property, discussed earlier, implies that the actual variations in the internal states of the nonlinear helicopter model become smooth as well. There is much to gain in terms of performance and smoothness by using soft interpolation constraints rather than demanding exact interpolation.

Given a set of basis functions

$$g_i(t) = \begin{cases} e^{A(t_i - t)} b & t \leq t_i \\ 0 & t > t_i, \end{cases} \tag{7}$$

we obtain the (translated) states, $\hat{x}$, of (5) to be

$$\begin{aligned} \hat{x}(t_i) &= \bar{x}(t_i) - e^{At_i}\bar{x}_0 = \int_0^{t_i} e^{A(t_i - s)} b u(s) ds \\ &= \int_0^T g_i(s) u(s) ds, \quad i = 1, \ldots, m. \end{aligned} \tag{8}$$

The convex cost functional that we want to minimize, with respect to u, becomes

$$J(u) = \int_0^T \frac{1}{2} \rho u(s)^2 ds + \sum_{k=1}^{m} \frac{1}{2} (\hat{x}(t_k) - \alpha_k)^T \tau_k (\hat{x}(t_k) - \alpha_k), \tag{9}$$

where α_k is the waypoint that we want the system to drive closely to at time t_k, and

$$\tau_k = \begin{pmatrix} \tau_{k_1} & 0 & 0 \\ 0 & \tau_{k_2} & 0 \\ 0 & 0 & \tau_{k_3} \end{pmatrix}. \tag{10}$$

Here, τ_{k_j} indicates how important it is that $\hat{x}$'s jth component is close to the desired point at the time t_k. If no constraint on that component is imposed at this time, we simply let $\tau_{k_j} = 0$.

Taking the Fréchet derivative of this functional (Luenberger, 1969), with respect to u and setting it equal to 0, based on the guess that

$$u(t) = \xi^T g(t), \tag{11}$$

where

$$g(t) = \left(g_1(t)^T, \ldots, g_m(t)^T\right)^T, \tag{12}$$

gives us

$$\xi = (\rho\mathcal{I} + \mathcal{T}\mathcal{G})^{-1}\mathcal{T}\alpha. \tag{13}$$

In (13), the terms $\mathcal{T}$ and $\mathcal{G}$ are defined as

$$\mathcal{T} = \begin{pmatrix} \tau_1 & & 0 \\ & \ddots & \\ 0 & & \tau_m \end{pmatrix} \tag{14}$$

and

$$\mathcal{G} = \int_0^T g(s)g(s)^T ds. \tag{15}$$

It should be pointed out that in Egerstedt and Martin (1998), it was proved that this problem is a convex optimization problem. Thus our necessary optimality conditions are in fact sufficient ones as well.

One advantage of this approach is that even though we only control a single input, we are able to impose constraints on all of the states as shown in Figure 2.

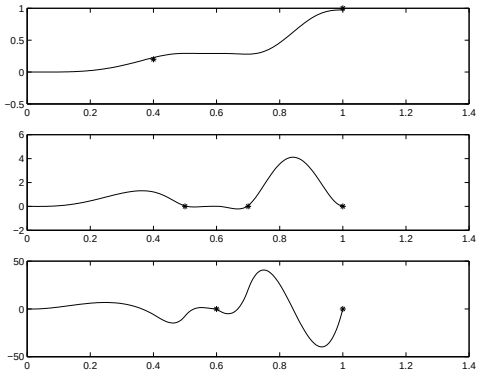


Figure 2: Interpolation through points specified simultaneously for the position (upper graph), velocity (middle graph) and acceleration (lower graph). The reason why the trajectory seems to interpolate through and not just close to the desired points is due to the fact that we, in this case, chose to let the τ_k:s be large, giving a higher priority to interpolation rather than smoothing.

3.2 Motion Primitives

The planning task becomes to generate a feasible trajectory assuming a specific motion primitive for the current segment. In the *cruising* case, we ask the helicopter to maintain a given altitude, h, while moving horizontally at a constant velocity, v_c. Assuming, without loss of generality,

that we want to fly in x-direction, the constraints become

$$\begin{pmatrix} x \\ \dot{x} \\ \ddot{x} \\ y \\ \dot{y} \\ \ddot{y} \\ z \\ \dot{z} \\ \ddot{z} \end{pmatrix}(t_0) = \begin{pmatrix} \star \\ v_c \\ \star \\ \star \\ 0 \\ \star \\ h \\ 0 \\ \star \end{pmatrix} \rightarrow \begin{pmatrix} x \\ \dot{x} \\ \ddot{x} \\ y \\ \dot{y} \\ \ddot{y} \\ z \\ \dot{z} \\ \ddot{z} \end{pmatrix}(t_1) = \begin{pmatrix} \star \\ v_c \\ \star \\ \star \\ 0 \\ \star \\ h \\ 0 \\ \star \end{pmatrix}, \tag{16}$$

where the $\star$ indicates that no constraint is imposed on that state at that time.

The same type of way points can be identified for other motion primitives. The planning for each of the segments can be done separately, using the final state configuration from one motion as the initial value for the next. Although this divide and conquer approach does not guarantee a global optimum, it provides a good solution requiring a minimal computational effort. For the linear optimal control problem in itself, this may not be such a big benefit, but we already in the introduction talked about the switched planning. This refers to the case when we switch between different A:s and b:s and, as we will see in the next subsection, this leads to a combinatorial optimization problem which complexity increases with the number of waypoints.

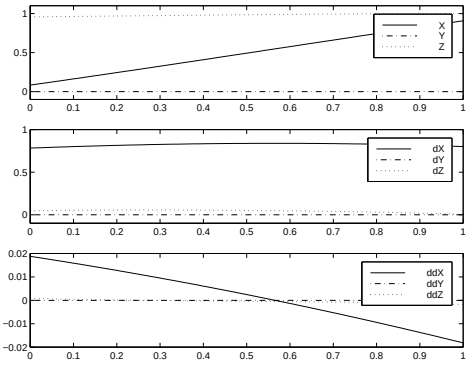


Figure 3: Here a cruise motion in the x-direction is planned and the upper graph shows the position, the middle the velocity while the lower shows the acceleration of the planned path. In the figures, x is solid, y is dotted, and z is dash-dotted. The reason why $\ddot{x} \neq 0$ is that in this case our initial condition was not $\ddot{x}(t_0) = 0$.

3.3 Controller Scheduling

In order to make the optimization problem numerically tractable, we assume that switchings are only allowed to occur at the waypoints. In the future we plan to employ reinforcement learning to learn an optimal control switching policy. Whenever the reinforcement learner inserts new switching points into the coarse flight path, the planner replans the trajectory ON-line using these new waypoints.

For a motion primitive composed of M waypoints, the number of possible switching policies is $3 \cdot 2^M$. Since we keep the number of waypoints small within each motion primitive, this does not lead to a too large number of possible policies from a computational point of view.

We define a functional for evaluating the performance for each of these different $3 \cdot 2^M$ solutions, as

$$\begin{aligned} J_j(u) &= \frac{1}{2}\rho\int_0^T (u_{jx}(t)^2 + u_{jy}(t)^2 + u_{jz}(t)^2)dt \\ &+ \frac{1}{2}\sum_{k=1}^{m} \Big\{ (\hat{x}_j(t_k) - \alpha_{xk})^T \tau_{xk} (\hat{x}_j(t_k) - \alpha_{xk}) \\ &+ (\hat{y}_j(t_k) - \alpha_{yk})^T \tau_{yk} (\hat{y}_j(t_k) - \alpha_{yk}) \\ &+ (\hat{z}_j(t_k) - \alpha_{zk})^T \tau_{zk} (\hat{z}_j(t_k) - \alpha_{zk}) \Big\}, \end{aligned} \tag{17}$$

where u_{jx} is the control signal corresponding to the j:th switching strategy for the x-subsystem. In the same way, $\hat{x}_j$ are the states for this subsystem driven by u_{jx}.

It should be noted that we do not need to calculate the entire functional for each j since old results can be reused in order to reduce the numerical complexity.

The planner generates the final trajectory using the path number $j^\star \in [1, 3 \cdot 2^M]$ with the lowest value on the functional. This corresponds to the path that is optimal with respect to a weighted sum of a waypoint-fitting and a smoothness criterion.

Figure (5) shows an entire flight path.

4 Conclusions

This article investigates the problem of generating optimal flight trajectories for an autonomous helicopter. We propose a planning strategy that partitions the optimization problem into isolated segments. The planner can employ different controllers in order to generate a smooth trajectory that minimizes the deviation from the given, constraining waypoints.

Our approach constitutes a systematic way for not only generating the flight path, but also provides a suitable switching strategy, i.e. when to switch between the different controllers.

Our approach also offers an analytical solution to the planning problem that does not require any computationally expensive numerical optimization. Therefore the planner is able to generate trajectories ON-line under the real time constraints given by the operation of the helicopter.

References

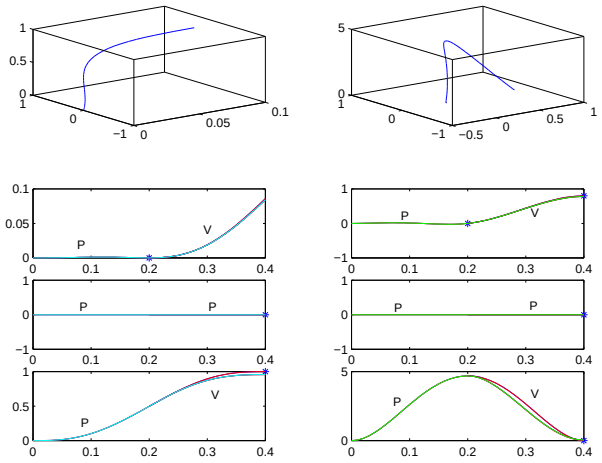
[1] Arkin, R.C. (1998). *Behavior-Based Robotics*, The MIT Press, Cambridge, Massachusetts.

[2] Branicky, M.S., V.S. Borkar, and S.K. Mitter (1998). "A Unified Framework for Hybrid Control: Model and Optimal Control Theory," *IEEE Transactions on Automatic Control*, vol. 43, no. 1.

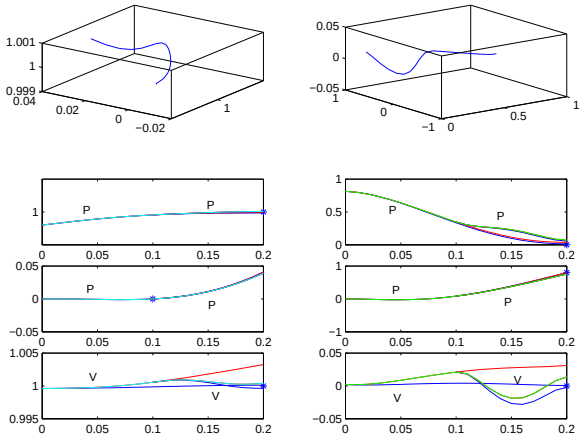
[3] Brooks, R. (1986). "A Robust Layered Control System for a Mobile Robot," *IEEE Journal of Robotics and Automation*, Vol. RA-2, No. 1, pp. 14-23.

[4] Egerstedt, M. and C.F. Martin (1998). "Trajectory Planning for Linear Control Systems with Generalized Splines," proceedings of the *Mathematical Theory of Networks and Systems* in Padova, Italy.

[5] Koo, T.J. and S. Sastry (1998). "Output Tracking Control Design of a Helicopter Model Based on Approximate Linearization," proceedings of the *37th IEEE CDC*, Tampa, Florida.



(a) Take-off motion.



(b) Turn motion.

Figure 4: Simulation of different motion primitives.The upper figures shows 3D-plots of the position and the velocity respectively in the motion primitive. The lower left plots show x, y and z while the right ones show $\dot{x}, \dot{y}$ and $\dot{z}$. The P:s and the V:s indicate what controller is active at each part of the path.

[6] Luenberger, D.G. (1969). *Optimization by Vector Space Methods*, John Wiley and Sons, Inc., New York.

[7] van Nieuwstadt, M.J. and R.M. Murray (1999). "Real Time Trajectory Generation for Differentially Flat Systems," to appear in *International Journal of Robust and Nonlinear Control*.

[8] Shim, H., T.J. Koo, F. Hoffman, and S. Sastry (1998). "A Comprehensive Study on Control

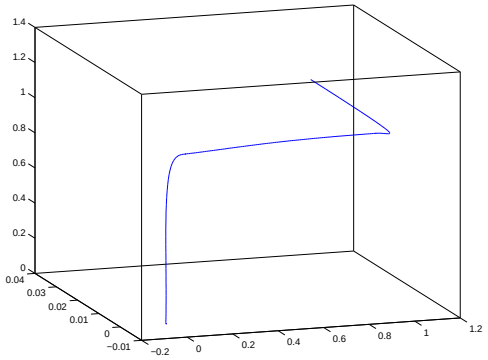


Figure 5: Here, a simulation of a planned route, built up from different motion primitives (take-off, cruise and turn) and controllers is displayed.

Design of Autonomous Helicopter," proceedings of the *37th IEEE CDC*, Tampa, Florida.

[9] Tomlin, C.J., G.J. Papas, J. Košecká, J. Lygeros, and S. Sastry (1998). "Advanced Air Traffic Automation: A Case Study in Distributed Decentralized Control," *Control Problems in Robotics*, Lecture Notes in Control and Information Sciences 230, Springer-Verlag, London.