Planar Spline Trajectory Following for an Autonomous Helicopter

Kale Harbick, James F. Montgomery and Gaurav S. Sukhatme

kale|monty|gaurav@robotics.usc.edu

Robotic Embedded Systems Laboratory
Center for Robotics and Embedded Systems
Department of Computer Science
University of Southern California
Los Angeles, CA 90089-0781

This paper proposes a technique for planar trajectory following for an autonomous aerial robot. A trajectory is modeled as a planar spline. A behavior-based control system which stabilizes the robot and enforces trajectory following, has been implemented and tested on an autonomous helicopter. Results from two flight experiments are presented. The trajectory tracking error is on the order of the size of the robot (1.8 m). Given the inherent error in GPS positioning, and environmental disturbances (wind), this is quite reasonable.

Keywords: Path planning, Robotic Helicopter, UAV, Trajectory following

1. Introduction

Autonomous aerial robots have tremendous appeal; one can imagine a large number of applications for them (defense, search-and-rescue, traffic monitoring, etc.). Helicopters are particularly attractive due to their Vertical Take Off and Landing (VTOL) capability. Due to cross-coupling effects and nonlinearities in the dynamics, the control of a helicopter for arbitrary flight profiles is a difficult problem and a general solution is unknown.

Several helicopter controllers have been designed for restricted flight profiles that include hover and low-speed, point-to-point navigation. Control approaches have been based upon a variety of sensors as the primary source of guidance and control, including GPS[5)], vision[1)], ultrasonics and vision[13)], and GPS/INS systems[12, 15)]. There are many applications for which more complex flight profiles are desirable. Fast-forward flight and curved trajectories are two examples of these. Possible applications for these flight profiles include testing vision software for Mars landers, minimizing fuel consumption (since following a curve uses less fuel than following a sequence of line segments), repeatable camera shots for movies, etc.

Controllers capable of following circular trajectories while simultaneously executing pirouette maneuvers have been implemented in simulation [9)]. Circular trajectory following has also been implemented on real robot platforms[4, 9)].

This paper focuses on a particular sub-problem of curved trajectory following. We implement, test and evaluate a technique for curved trajectory following in the plane. The reference trajectory is specified by a cubic B-spline. This spline format is desirable because it can be easily constructed with minimal user input (a few control points are graphically input via a mouse). Further, the tangent and curvature computation is minimal.

The paper is organized as follows. Section 2 describes the overall control system architecture of the helicopter. Section 3 describes the spline following behavior in detail. Experimental results are presented in Section 4. Conclusions and future work are discussed in Section 5.

2. Helicopter Platform

2.1. Hardware

The AVATAR (Autonomous Vehicle Aerial Tracking And Reconnaissance) [2)] is a radio-controlled helicopter (**Figure 1**) augmented with a PC/104 stack and several sensors. A Novatel RT20 GPS card provides position to an accuracy of 20 cm CEP (Circular Error Probable, i.e. the radius of a circle, centered at the true location of the receiver antenna, that contains 50 percent of the individual position measurements made using a particular navigation system). A three-axis accelerometer and three-axis gyro are contained in a Boeing CMIGITS-II INS unit.

Figure **1**: The AVATAR (Autonomous Vehicle Aerial Tracking And Reconnaissance) in flight.

The OCU (Operator Control Unit) is a laptop that is used to send high-level commands (such as the control points of the spline) to the helicopter. They communicate through wireless ethernet. The ground computer also sends differential GPS corrections to the heli.

2.2. Flight Control System

Autonomous flight is achieved using a *behavior-based* control architecture[14)]. Behavior-based systems are an extension of reactive architectures[3)]. They have been shown to store state and support representation [10)]. They have been used in navigation, mapping[6, 7)], distributed group foraging and collective coordinated pushing[11)] to name a few examples. However, behavior-based systems are not hybrid systems. Unlike hybrid systems, which essentially layer a planner on top of a reactive module, behavior-based systems rely on an interconnected structure of capabilities, each typically encapsulated as a process.

The helicopter control system is shown in the diagram of **Figure 2**. Each atomic capability (denoted by an oval) is a behavior. The low-level behaviors (roll, pitch, yaw and altitude stabilization) are implemented with proportional controllers. As an example, the roll control process reads in the current roll angle from the Inertial Navigation System and outputs a proportional servo command that is sent to the lateral cyclic servo of the helicopter which is the actuator responsible for directly controlling roll. This is shown in Equation 1, where τ is the servo command, θ is the roll angle, θ_d is the desired roll angle, and K_p is the proportional gain (typically 7.0 for the AVATAR). Our system is stable without derivative terms in the control loops because the craft is sluggish. As long as we maintain a hover or slow speed, the local stability region is large.

$$\tau = -K_p(\theta - \theta_d) \tag{1}$$

The short-term goal behaviors are intermediate between the navigation control behavior and the low-level behaviors. They are also implemented as proportional controllers. For example, the lateral velocity controller reads a desired forward/back velocity and a desired right/left velocity from the navigation control, then outputs desired roll and pitch angles to roll and pitch control, respectively.

The highest-level behavior, navigation control, is responsible for long-term goals, such as moving to a particular position. If the goal position error and heading error are both greater than allowed thresholds (currently 7.0 m and 45 degrees, respectively), then the navigation control behavior first directs heading control to point the helicopter toward the goal while commanding zero lateral velocities. When heading error becomes less than 45 degrees, navigation control gives desired lateral velocities (proportional to the distance from goal) to the lateral velocity controller. If the goal position is closer than the threshold distance of 7.0 m, navigation control computes lateral velocity components and passes these to the lateral velocity behavior. In other words, the helicopter re-orients and flies towards goals positions far from its present location, but flies sideways to goal positions that are nearby.

With the present instrumentation, the helicopter cannot take-off autonomously, so a human pilot flies the the craft up to a safe height and then switches control over to autonomous flight mode. Switching logic onboard the AVATAR allows servo commands to be routed from the radio receiver (human controlled flight) or the onboard CPU (autonomous flight).

2.3. User Interface

The flight control system onboard the AVATAR can receive high-level tasking in a variety of ways from the OCU. These are summarized in **Figure 3**. For example, the AVATAR can be commanded by the operator to follow objects (such as other robots[16)]) on the ground, using GPS information alone. The AVATAR can also follow objects using its vision system. This is based on a simple tracking algorithm[17)] that detects regions of high contrast. The vision system can also lock on to a pattern (e.g. a large white H) on the ground and align the helicopter with the pattern in preparation for landing. The user can drag and drop an icon of the helicopter to another location inside the perimeter (defined by the GPS locations of the corners of the flying area), and the helicopter will fly to that location. The user can use a joystick to command forward, back, left and right flight. The control system interprets these commands as desired velocities and produces the proper desired roll and pitch angles. The AVATAR can also be put into an autonomous tasking mode in which it may switch between object tracking and GPS-based point-to-point navigation without any human commands.

A recent addition to the command set is to follow a user-specified spline trajectory in the plane. This behavior is described in detail in the next section.

3. Spline Following Behavior

The Catmull-Rom[8)] method of spline representation was chosen for three reasons. First, for a closed path, each control point lies on the path. This allows the user to precisely specify a series of waypoints that the AVATAR must pass through. Second, moving a single control point provides local control of the curve. In other representations, moving one control point can have undesirable effects on the entire curve. Finally, the tangent vector at control point P_i is parallel to the line connecting control points P_{i-1} and P_{i+1}. This was important since we wanted the AVATAR to stay tangent to the trajectory, and this property makes it easy to compute desired headings.

Equation 2 shows the construction of Catmull-Rom splines, where M_{CR} represents the Catmull-Rom basis matrix, T is a vector of powers of t from 3 to 0, and G_{Bs_i} is the B-spline geometry vector for segment Q_i.

$$\begin{aligned} Q_i(t) &= T \cdot M_{CR} \cdot G_{Bs_i} \\ &= \frac{1}{2} \cdot T \cdot \begin{bmatrix} -1 & 3 & -3 & 1 \\ 2 & -5 & 4 & -1 \\ -1 & 0 & 1 & 0 \\ 0 & 2 & 0 & 0 \end{bmatrix} \begin{bmatrix} P_{i-3} \\ P_{i-2} \\ P_{i-1} \\ P_i \end{bmatrix} \end{aligned} \tag{2}$$

The spline trajectory is constructed after the user specifies a series of control points on the OCU. A

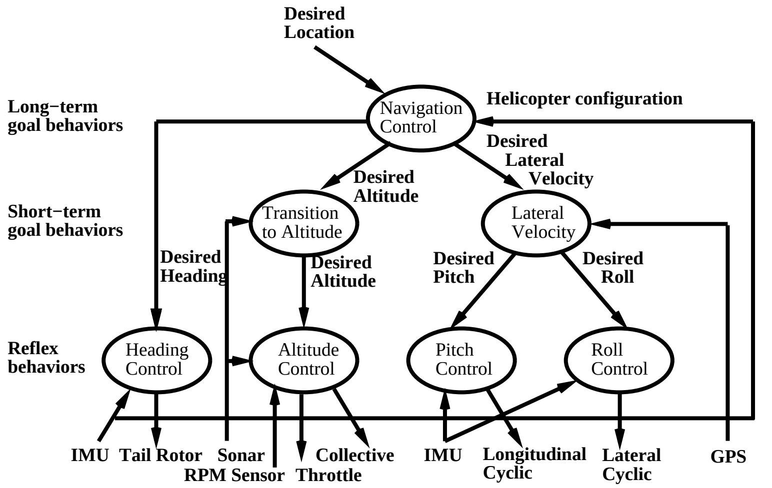


Figure **2**: Behavior-based flight control system architecture for the AVATAR.

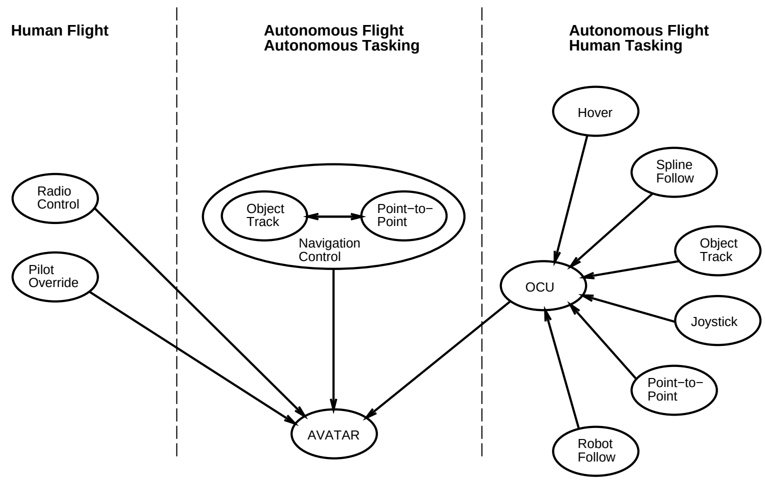


Figure **3**: The AVATAR can be flown by a human or fly autonomously. The human pilot can always take control back by toggling the pilot override switch on the RC transmitter. In autonomous flight, high-level tasks can come from the user or the helicopter can autonomously task itself.

screenshot of the OCU is shown in **Figure 4**. The sequence of points in the screenshot is an example of a user-specified trajectory. These points can also be loaded from a file. The OCU interpolates ten waypoints between each pair of control points. For example, the figure eight path used in the experiments described in the next section was specified with eight control points. The OCU interpolated these to produce a total of eighty waypoints. The desired heading at waypoint i is set to the the slope of the line connecting waypoint $i-1$ and waypoint $i+1$. The computed latitude and longitude coordinates and corresponding headings are transmitted to the AVATAR, so that the robot has complete knowledge of the path before it starts to follow the trajectory.

The pilot flies the AVATAR to a height of approximately 10 m and hovers, then switches to robot control. When the user sees that the helicopter is hovering autonomously, the user commands the follow-spline behavior on the OCU and the AVATAR flies to the first point on the trajectory. The flight control system generates desired lateral velocities using a proportional servo loop on the GPS position. A proportional servo loop on heading values from the INS is used to generate yaw commands. When the helicopter is within 1 m of the desired waypoint, the next waypoint in the sequence becomes the new desired waypoint. This continues until the last point is reached, and then begins again from the first point.

4. Experiments

For the experiments described here, the user specified a figure eight trajectory for the helicopter to follow. The figure eight measured approximately 12 m across at its widest. **Figure 5** shows two traversals of

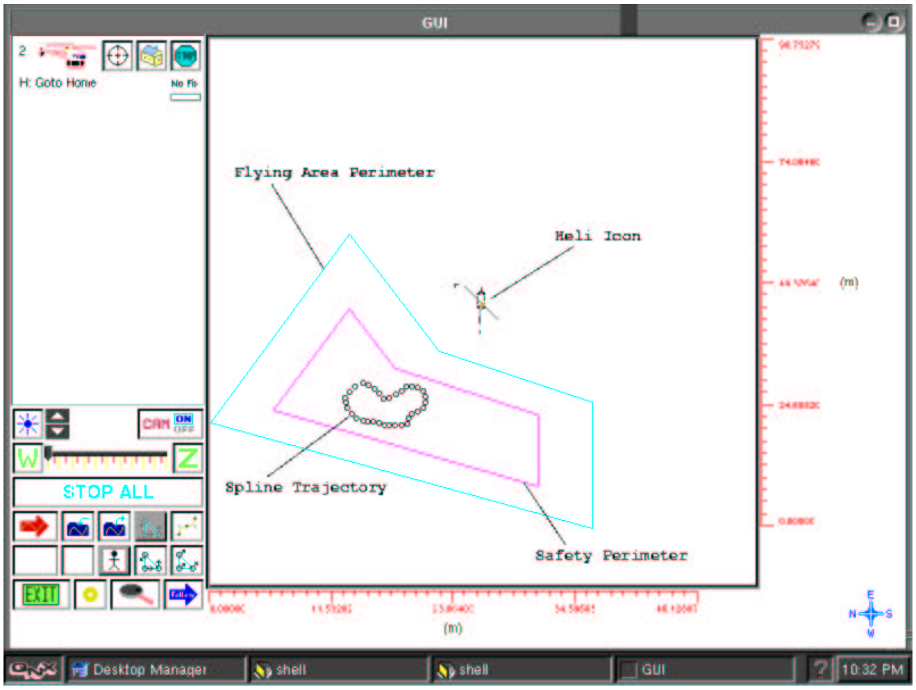


Figure **4**: A screenshot of the OCU after the user has specified a desired spline trajectory.

		Avg. Error (m)	**Std. Dev. (m)**
Trial 1	**Actual Path**	1.15	0.48
	w/o Anomaly	0.94	0.25
Trial 2	**Actual Path**	1.70	0.68
	w/o Anomaly	1.55	0.62

Table 1: GPS position errors and standard deviations.

the trajectory.

The circles represent the waypoints of the path, and the solid line is the actual path taken by the AVATAR. The largest errors result from the anomaly present in the upper right of both figures. We believe this anomaly is caused by the strong breeze from the southwest that was present on the day of our experiments. As the helicopter was traveling northeast on the far right of the path, the wind increased the speed of the AVATAR so that it wasn't able to make the sharp turn quickly enough to stay on the path. Table 1 shows the average position error in meters along with standard deviations. The right column shows the average position error if the readings from the anomaly are ignored. The average standard deviation of the GPS measurements themselves was approximately 25 cm.

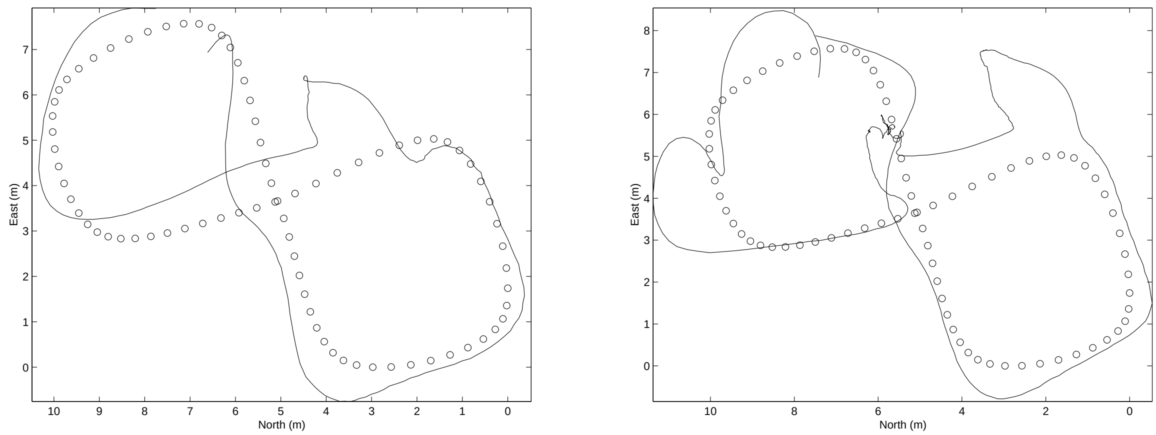
5. Conclusions and Future Work

We have a presented a planar, spline-based, trajectory following algorithm for an autonomous robotic helicopter. This capability allows a user to command the helicopter by simply providing a set of via points for the trajectory. Experiments show that that the average tracking error of our system is approximately 1.5 m.

The span of the helicopter's main rotor is approximately 1.8 meters, so the average position errors were approximately the size of the craft. It may be noted that we are forced to specify a relatively small path because of flying area size limitations. Further, we are constrained by the accuracy of the GPS which is a source of some of the positioning error.

Given a larger flying area, and hence longer flight paths, we would expect to see no change in the position errors. In fact, if the curves in the path were not as tight (i.e. of lower curvature) as those in the figure eight we used for the experiments reported here, we hypothesize that the average tracking error would decrease. This however, remains to be tested.

Further, we hypothesize that changing the controller logic to allow for changes in desired speeds depending on the local curvature of the path would result in the largest reduction of position errors. The experiments described in this paper used planar trajectories, with the helicopter maintaining a fixed altitude. Following a general three-dimensional spline trajectory should be possible with our existing flight control system, but our OCU needs to be enhanced in order for the user to be able to specify a 3D path. Although we specified paths with desired headings that



(a) (b)

Figure **5**: (a) Trial 1 and (b) Trial 2 of the spline following experiment. Circles indicate desired trajectory waypoints, solid line indicates actual path taken by the helicopter.

were tangent to the curve, any desired yaw orientation should be possible, since a helicopter can fly in any direction. These capabilities are the subject of present and future research.

Acknowledgments

The authors would like to thank our helicopter pilot Doug Wilson, Francisco J. Mesa-Martinez and Andy Ramakrishna for hardware support, and Sanjeev Koppal for his work on the user interface. This work was supported by contracts F04701-97-C-0021 and DAAE07-98-C-L028 from DARPA under the Tactical Mobile Robotics (TMR) program, and by DARPA grant DABT63-99-1-0015 under the Mobile Autonomous Robotic Software (MARS) Program.

References

1) O. Amidi, T. Kanade, and R. Miller. Vision-based autonomous helicopter research at cmu. In Proceedings of Heli Japan 98, Gifu, Japan, 1998.

2) USC Autonomous Flying Vehicle Project website. http://www-robotics.usc.edu/~avatar.

3) R. A. Brooks. A robust layered control system for a mobile robot. *IEEE Journal Robotics and Automation*, 2(1):14–23, March 1986.

4) Marco La Civita. *Integrated Modeling and Robust Control for Full-Envelope Flight of Robotic Helicopters*. PhD thesis, Carnegie Mellon University, December 2002.

5) Andrew R. Conway. *Autonomous Control of an Unstable Helicopter Using Carrier Phase GPS Only*. PhD thesis, Stanford University, March 1995.

6) Goksel Dedeoglu, Maja J. Matarić, and Gaurav S. Sukhatme. Incremental, on-line topological map building with a mobile robot. In Proceedings of Mobile Robots XIV - SPIE 99 Boston, MA, pages 129–139, 1999.

7) Goksel Dedeoglu and Gaurav S. Sukhatme. Landmark-based matching algorithm for cooperative mapping by autonomous robots. In Distributed Autonomous Robotic Systems (DARS) 2000 Knoxville, Tennessee, pages 251–260, 2000.

8) James D. Foley, Andries van Dam, Steven K. Feiner, and John F. Hughes. *Computer Graphics: Principles and Practice*. Addison-Wesley, Reading, MA, second edition, 1997.

9) Suresh K. Kannan and Eric N. Johnson. Adaptive trajectory-based flight control for autonomous helicopters. In *Proceedings of the 21st Digital Avionics Systems Conference*, number 358, 2002.

10) Maja J. Matarić. Integration of representation into goal-driven behavior-based robots. *IEEE Transactions on Robotics and Automation*, 8(3):304–312, June 1992.

11) Maja J. Matarić. Behavior-based control: Examples from navigation, learning, and group behavior. *Journal of Experimental and Theoretical Artificial Intelligence*, 9(2–3):323–336, 1997.

12) J. F. Montgomery. *Learning Helicopter Control Through 'Teaching by Showing'*. PhD thesis, University of Southern California, May 1999.

13) J. F. Montgomery, A. H. Fagg, and G. A. Bekey. The USC AFV-I: A behavior-based entry in the 1994 International Aerial Robotics Competition. *IEEE Expert*, 10(2):16–22, April 1995.

14) James F. Montgomery. The USC Autonomous Flying Vehicle (AFV) Project: Year 2000 Status. Technical Report IRIS-00-390, Institute for Robotics and Intelligent Systems, University of Southern California, 2000.

15) H. Shim, T. J. Koo, F. Hoffmann, and S. Sastry. A comprehensive study on control design of autonomous helicopter. In Proceedings of IEEE Conference on Decision and Control, Florida, pages 3653–3658, 1998.

16) Gaurav S. Sukhatme, James F. Montgomery, and Maja J. Mataric. Design and implementation of a mechanically heterogeneous robot group. In Proceedings of SPIE: Sensor Fusion and Decentralized Control in Robotic Systems II Vol. 3839, pages 122–133, 1999.

17) R. T. Vaughan, G. S. Sukhatme, F. J. Mesa-Martinez, and J. F. Montgomery. Fly spy: lightweight localization and target tracking for cooperating ground and air robots. In *Proceedings of the International Symposium on Distributed Autonomous Robot Systems*, pages 315–324, 2000.