2009 International Conference on Robotics and Automation (ICRA)
http://www.cds.caltech.edu/~murray/papers/hsdm09-icra.html

A Real-Time Helicopter Testbed for Insect-Inspired Visual Flight Control

Shuo Han, Andrew D. Straw, Michael H. Dickinson and Richard M. Murray

***Abstract*— The paper describes an indoor helicopter testbed that allows implementing and testing of bio-inspired control algorithms developed from scientific studies on insects. The helicopter receives and is controlled by simulated sensory inputs (e.g. visual stimuli) generated in a virtual 3D environment, where the connection between the physical world and the virtual world is provided by a video camera tracking system. The virtual environment is specified by a 3D computer model and is relatively simple to modify compared to realistic scenes. This enables rapid examinations of whether a certain control law is robust under various environments, an important feature of insect behavior. As a first attempt, flight stabilization and yaw rate control near hover are demonstrated, utilizing biologically realistic visual stimuli as in the fruit fly *Drosophila melanogaster*.**

I. Introduction

Insects were the first organisms to achieve flight, and are arguably the most sophisticated flying organisms. Although their brain consists of only 500,000 neurons, they are capable of controlling both high level behavior (e.g. food localization) and low level behavior (e.g. leg coordination) in apparently robust and successful ways. When compared to autonomous machines performing multi-level control, it becomes apparent that the tiny brains of insects must implement rather effective and efficient algorithms [1]. Among insects, flies in general and the fruit fly *Drosophila melanogaster* in particular, are especially well studied species. Various visuomotor responses, in addition to senses such as wind and rotation detection, have been studied to gain an understanding of how flies stabilize and guide flight [2]. The optics of the compound eye provide spatial resolution far inferior to modern electronic video cameras, with only 700 ommatidia (roughly, pixels) per side. The importance of vision, despite this limited spatial resolution, is suggested by the portion of the brain dedicated to visual processing—roughly two thirds of the fly brain is within the optic lobes, and this neglects parts of the central brain also involved with vision.

This paper introduces a vision-based helicopter control testbed (shown in Fig. 1), in which the control law takes visual input from a biologically realistic simulation of the visual system of *Drosophila*. Generation of the visual stimuli is accomplished through graphical rendering of a 3D virtual environment, rather than from onboard video cameras. The testbed serves two functions. First, this testbed enables certain experiments that are otherwise difficult to control or even not possible on real animals. Second, from an engineering perspective, it is intriguing to examine possible ways of adopting the mechanisms of insect flight control in practical systems and/or whether there would be any limitations.

The authors are with the Division of Engineering and Applied Science, California Institute of Technology, Pasadena, CA 91125, USA. {hanshuo, astraw, flyman}@caltech.edu, murray@cds.caltech.edu

Fig. 1. The helicopter used in this work. It is based on E-flite Blade CX2, a coaxial 4-channel indoor RC helicopter (gross weight: 224 g, rotor span: 345 mm). Modifications include: (1) removal of the original canopy and tail boom to increase payload; (2) adding a metal frame with 5 LEDs as feature points in video tracking.

II. Related work

The idea of exploiting visual information is not new in the field of aerial robotics and has been implemented in various unmanned aerial vehicle (UAV) projects [3], [4], [5]. However, only until recently researchers have started to study the possibility of developing control laws that make use of biologically meaningful visual inputs. Much work has been done at the simulation level. Early research usually ignores the detailed configuration of the compound eyes, and treats the two eyes as two single photodetectors [6], [7]. However, there are clues that insects use inputs from each ommatidium via lobular plate tangential cells (LPTC) that perform as "matched filters" [8]. Later work showed that simulated retinal velocities of each ommatidium can collectively provide a full estimate of the system states (e.g. forward speed, positions) under certain maneuver constraints using such algorithm [9], [10], [11]. The idea of matched filter has also been extended to insect flight control simulation with a more biologically realistic visual model and *Drosophila* body/wing dynamics [12], [13].

One of the early attempts on implementing insect flight control laws in aerial robots is a microflyer developed by Laboratory of Intelligent Systems at EPFL [14]. With an onboard lightweight video camera, optic flow can be calculated and used as a sensory input. The microflyer is in turn able to fly indoor while avoiding obstacles through

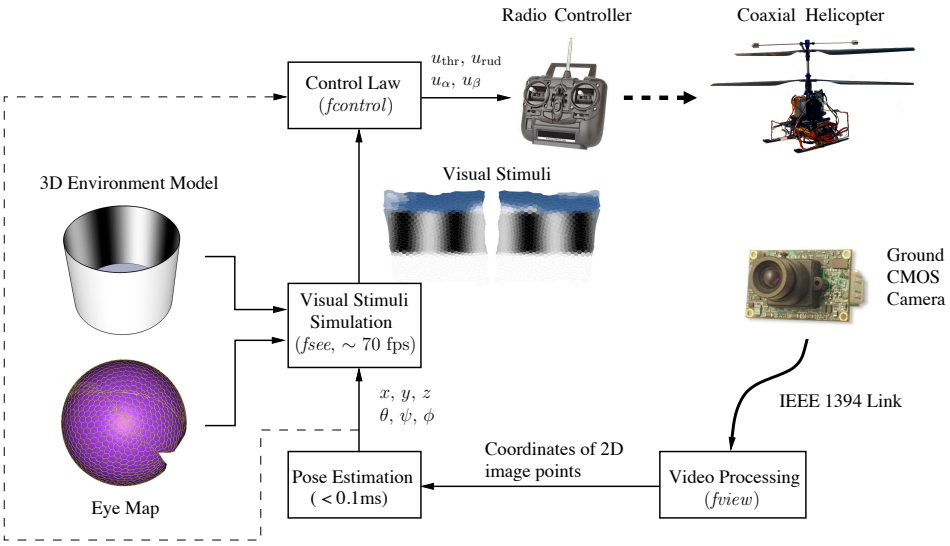

Fig. 2. A schematic of the indoor helicopter testbed incorporating biological control laws. The control laws currently only use simulated *Drosophila* (fruit fly) visual stimuli from *fsee*. The *fsee* program requires (1) a 3D environment model, (2) configuration of the compound eye optics (the eye map), and (3) position and orientation (known as pose) of the helicopter. The environment model and eye map are coded in software, whereas the pose is computed from 2D video images through a pose estimation algorithm. The designed control law, *fcontrol*, computes the control commands from the visual stimuli and feeds the commands to the helicopter via a 4-channel (u_{thr}, u_{rud}, u_α, u_β) radio controller.

intermittent saccades (turning by ~90 deg in yaw) triggered by certain changes in optic flow.

None of the work above, however, examines the possibility of incorporating a virtual environment in the control loop. This can be beneficial when a large number of different stimuli need to be tested because virtual environment allows batch generation. Building the entire testbed leads to two possible issues that need to be carefully addressed. First, simulating equivalent visual stimuli requires the current position and attitude of the helicopter, which can be obtained by pose estimation. However, pose estimation algorithms usually do not emphasize real-time performance, whereas aerial vehicle control often imposes stringent requirement on time delays ($\ll$ 20 ms for our helicopter). To tune the algorithm in order to meet our special need, our work combines two different pose estimation algorithms, POSIT and SoftPOSIT, together with a Kalman filter providing necessary initial pose guesses and noise reduction. This gives satisfactory performance (average delay $<$ 0.1 ms) even on an off-the-shelf PC, avoiding the use of much more expensive professional motion capture systems (e.g. VICON). Second, the aim of this paper is focused on stabilization and yaw control during hovering. Yet flight control during hovering is generally believed to be more challenging than forward flight because the vehicle is normally more susceptible to noise. It is uncertain that whether the noise will severely corrupt the state estimates and/or destabilize the helicopter. The paper shows that good yaw rate estimates can be obtained using a "matched filter" approach and discusses some associated limitations.

The paper is organized as follows. Sec. III gives an overview of the testbed, which consists of three major parts: (1) the helicopter itself, (2) the sensor subsystem, and (3) the control/actuation subsystem. Each part is introduced in sequence in Sec. IV–VI. Sec. VI also presents and discusses several preliminary results on helicopter control using bio-inspired algorithms. The paper concludes in Sec. VII with directions on future work.

III. System Overview

The entire system (Fig. 2) consists of three major parts: the helicopter itself, a sensing subsystem that captures the status (position and attitude) of the helicopter and generates the corresponding biological visual stimuli, and a control/actuation subsystem that executes given control laws. There are mainly two types of radio-controlled (RC) helicopter commercially available, with different yaw control mechanisms. One type features a single main rotor, with its yaw torque controlled by a tail rotor; the other has a pair of counter-rotating coaxial main rotors where the yaw is controlled through the differential speed of the rotor pair. The latter is chosen in our case due to its similar yaw control mechanism with the fly, which achieves yaw motion by altering the beat frequencies of the wing pair. Besides electronics, the helicopter is also equipped with 5 IR light-emitting diodes (LED) with which the sensing subsystem can estimate the current pose of the helicopter.

The sensing subsystem is composed of two functions. A ground video camera first records the image of the IR LEDs, from which a pose estimation algorithm extracts out the 3D position (x, y, z) as well as the attitude (three Euler angles—θ, ψ, ϕ) of the helicopter with respect to the camera. To avoid possible ambiguity and error during pose estimation, a Kalman filter is also added in the estimation loop. With the complete 6-degree-of-freedom (DOF) information available, a program called *fsee* is able to simulate the instantaneous

biological visual input (see Fig. 2) for a given virtual environment, which shares the same origin with the real world. A control law will receive this input and compute the control commands.

The actuation is achieved by having a PC communicate with the radio controller shipped with the helicopter. To keep the original radio controller intact, the communication is done over the trainer port that accepts PPM signals. The signals can be generated from a certain peripheral circuit and several digital potentiometers controlled by the PC over an SPI interface. A video showing the helicopter in motion and the associated simulated *Drosophila* visual stimuli (a sinusoidal pattern is shown for better visual illustration) can be found in the supplementary materials [15].

IV. Helicopter configuration

Our testbed uses an E-flite Blade CX2 coaxial indoor helicopter. Its rotor blades are 345 mm in diameter and are driven by two separate DC brushed motors. Powered by a 2-cell 800-mAh lithium-polymer battery, the helicopter is able to fly for 10-15 minutes. The small size and excellent flight time make this model particularly ideal for indoor experiments. The pitch of the rotor blades is fixed. Therefore, lift control is achieved by changing the speeds of the upper and lower rotors collectively through the throttle command (u_{thr}). Yaw control is realized by tuning the differential speed between the two counter-rotating rotors via the rudder command (u_{rud}). Pitching and rolling are controlled by the cyclic pitch of the lower rotor blades, which is controlled by a 2-DOF servo-driven swashplate that takes inputs u_α and u_β; the upper rotor is passively controlled by gyroscopic forces generated from the attached stabilizer bar. All commands (u_{thr}, u_{rud}, u_α, u_β) are sent through a miniaturized 4-channel 2.4 GHz radio system. The rudder channel is also mixed with the output from the onboard gyroscope, which is however turned off in the following experiments for yaw control testing purposes. The helicopter has been retrofitted to meet several requirements. The outer plastic enclosure is removed to gain more payload. The helicopter also carries 5 IR LEDs (from NaturalPoint) that are arranged in a non-coplanar fashion, as required by the pose estimation algorithm. The algorithm requires that the positions of the LEDs should be measured accurately. For this purpose, the LEDs are soldered onto several designated spots located on a pre-designed frame, which is in turn attached to the helicopter.

One limitation of our helicopter system is that it is under-actuated, because it has 6 DOFs with yet only 4 command inputs. This is evident in that the helicopter must, for example, pitch forward in order to initiate forward flight. Due to this limitation, this paper tests only two scenarios that do not involve large lateral motion: hovering and yaw rotation. Yaw rotation is selected because this is relevant to saccades, which are rapid turns performed by flies intermittently. Yet this issue of underactuatedness needs to be carefully considered in the future if significant translational movement is present (e.g. forward flight regulation).

V. Sensor subsystem

A. Video capture and processing

A video system provides the interface between the real world and the PC. Because the present work is focused on helicopter operated near hover, it suffices to use a single camera system, which does not require any synchronization as in the case of multiple cameras. Our platform uses a PointGrey Firefly MV CMOS camera with a frame rate of 60 fps (wide VGA, 752×480). The camera is equipped with a 6-mm microlens, providing a viewing angle of 42 deg and 27 deg in the length and width directions, respectively. Pointing upward, the camera can track a 76.6×48.9 cm^2 area 1 m above it, with a spatial resolution of $\sim$1 mm. This tracking range is enough for our present experiment where the helicopter hovers in place. For future experiments requiring a larger tracking range, the system can be upgraded to a multi-camera configuration [16]. An IR-pass filter (Schneider Optics, B+W 093, $\lambda_c = 830$ nm) is added in the front to reduce the influence from ambient stray light. The camera itself serves as the origin in the real world, which also coincides with the origin of the virtual environment (see Sec. V-C).

The camera is connected through IEEE 1394 to a PC, where the video stream is processed by *fview*, part of *motmot* [17], which is a collection of open source packages for real-time collection and analysis of uncompressed digital images. At each frame, a plugin in *fview* called *trackem* will detect the feature points (LEDs in this case) by thresholding and report their coordinates on the 2D image over a UDP port.

B. Pose estimation

Simulation of biological visual stimuli requires the current position and orientation (also known as pose) of the helicopter with respect to the camera. Given the 3D positions of the feature points (in cm) on the helicopter and their locations on the image (in pixels), the pose can be solved by finding the best rotational and translational match between the 3D coordinates and the 2D projections. Two types of pose estimation algorithms are used in our testbed: POSIT (Pose from Orthography and Scaling with ITerations) [18] and SoftPOSIT (POSIT + SoftAssign). POSIT is relatively simple and fast (< 0.1 ms@2.4 GHz Core2 Duo), but requires one-to-one mutual correspondence between the feature points and their projected images; SoftPOSIT, on the other hand, can handle unknown correspondence and missing points, yet is less robust and much slower (2–4 ms).

To fully exploit the video camera bandwidth, the delay caused by pose estimation is expected to be negligible compared to the camera (17 ms). However, POSIT itself is not suitable in our case because the image-object correspondence can be unknown during the flight. All the 5 LEDs appear identically on the 2D images and some of the LEDs might be blocked by the fuselage occasionally and will be missing in the camera image. To take advantage of both POSIT and SoftPOSIT, our pose estimation process combines the two through a Kalman filter:

1) Run SoftPOSIT to solve the initial correspondence;
2) Read the locations of feature points from the current camera image;
3) Run the time-update (prediction) stage of the Kalman filter to predict the current pose;
4) From the predicted pose, either (a) determine the current correspondences using nearest neighborhood and solve the current pose from POSIT or (b) when POSIT fails, solve the pose from SoftPOSIT directly. Using nearest neighborhood method can eliminate spurious feature points on the 2D image.
5) Run the measurement-update stage of the Kalman filter to smooth out the pose estimation result and reduce the effect of occasional false estimation.

The dynamical model of our Kalman filter is a simple one assuming constant velocities for all the 6 DOFs, yet performs surprisingly well. Occasional occlusion of one LED has been tested to have negligible influence on pose estimation results. It is worth noting that a Kalman filter incorporating the dynamical model of the helicopter is expected to gain more accuracy, although this was not tested in this work. There is also a possibility that the Kalman filter may filter out some realistic noises from the helicopter. This artificial smoothing issue can be a potential limiting factor when testing the high-frequency performance of control laws such as disturbance rejection. Future work will be conducted to characterize the bandwidth of the video tracking system.

C. Visual stimuli simulation

To acquire visual information, a virtual 3D environment model needs to be specified. In our study, the environment was set to be a round arena with certain image patterns painted on its inner wall. The model is built with Google Sketchup and exported in COLLADA (.dae) format, which can then be loaded and rendered by OpenSceneGraph, an open source, cross-platform 3D graphics toolkit based on OpenGL. At each sampling instant, according to the estimated pose of the helicopter, the 3D environment model is rendered as a cube map, which consists of 6 snapshots of the surroundings. A fast algorithm can then transform the cube map to the luminance profile, which is blurred by a Gaussian kernel to emulate the optics of the compound eye. The transformation uses a biologically realistic distribution of the ommatidia, which is obtained by Buchner for *Drosophila* [19]. The eye map includes 699 elements on each compound eye, with each element spanning a solid angle of 4.5 to 6 deg. Fig. 3 shows the actual eye map and a typical simulation of the visual stimuli. Details of the simulation are described elsewhere [13].

VI. Control/Actuation Subsystem

A. Hardware setup

A hardware interface is required for the PC to send control commands to the helicopter, or essentially, the radio controller. Like most advanced radio controllers, our radio controller comes with a trainer port that receives pulse position modulation (PPM) signals from a buddy box. We use the circuit taken from a second radio controller of the same model to generate the PPM signal. All the 4 onboard potentiometers that control the 4 channels are replaced by digital ones. The PC can then communicate with the digital potentiometers over an SPI bus. A GUI frontend called *fcontrol*, written in Python, provides user interaction (e.g. data recording) and data visualization. The *fcontrol* interface allows the user to switch between manual and automatic control mode, where the former mode takes joystick inputs through *pygame* module and the latter one sends control commands given by the control law detailed below.

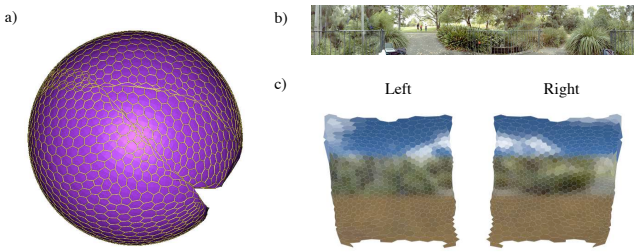

Fig. 3. a) Configuration of *Drosophila* compound eye optics (eye map) represented in 3D, according to Buchner's data [19]. Each cell on the spherical surface indicates the solid angle spanned by a single ommatidium. There is a missing wedge located in the aft portion, meaning the fly is not able to observe immediate visual cues in the back; b) Naturalistic scene used in the 3D arena model; c) Simulated visual stimuli from both two compound eyes (shown in Mercator projection) when the helicopter is at the center of the arena. The cloud and sky are from the background of the 3D world where the arena is placed, not the arena itself.

B. Control law and results

The first goal in helicopter flight control is to stabilize the vehicle in hover. The hovering condition requires that 6 velocities (3 translational, 3 angular) be zero: $\dot{x} = \dot{y} = \dot{z} = 0$, $\dot{\theta} = \dot{\psi} = \dot{\phi} = 0$. However, controlling solely the lateral velocities ($\dot{x}$ and $\dot{y}$) will eventually lead to cumulative errors in the lateral positions and cause the helicopter to move beyond the visible range of the camera system. The hovering condition is then restricted to be "hover in place" ($x = 0$, $y = 0$) in order to circumvent this issue, forcing the helicopter to stay near the origin. For safety purposes, the altitude (z position) is controlled manually through the throttle; it is used to initiate takeoff/landing and is simply held constant once the helicopter reaches the desired height. Pitch and roll motions are observed to be passively stable and do not require additional control.

Our control law controls the rest 3 DOFs—x, y, and yaw rate—via the collective pitch commands (u_α and u_β) and the rudder command (u_{rud}), respectively. For simplicity, couplings between the three channels are ignored and three independent control laws are applied. The yaw rate, which is approximately $\dot{\phi}$ near hover, is controlled by a proportional-integral (PI) controller, whereas x and y by proportional-derivative (PD) controllers:

$$u_{\mathrm{rud}} = k^P_{\phi\_\mathrm{dot}}(\dot{\phi}_{\mathrm{ref}} - \dot{\phi}) + k^I_{\phi\_\mathrm{dot}} \int (\dot{\phi}_{\mathrm{ref}} - \dot{\phi})\, dt, \qquad (1)$$

$$u_\alpha = k_x^P(x_{\text{ref}} - x) + k_x^D \frac{d}{dt}(x_{\text{ref}} - x), \quad (2)$$

$$u_\beta = k_y^P(y_{\text{ref}} - y) + k_y^D \frac{d}{dt}(y_{\text{ref}} - y). \quad (3)$$

During hovering, all the reference set points, $\dot{\phi}_{\text{ref}}$, x_{ref}, and y_{ref}, are set to zero. The above control laws require estimation of x, y, and $\dot{\phi}$. $\dot{\phi}$ can be obtained from retinal velocity, defined as the movement rate of the visual world projected onto the retina. The retinal velocity represents the fly's visual perception of the surrounding scenery. Retinal velocity near each ommatidium can be obtained from two adjacent ommatidia through a nonlinear temporospatial correlation. This correlation model, proposed by Hassenstein and Reichardt from experiments on beetles, is often known as the elementary motion detector (EMD) [20]. Fig. 4a shows the original EMD configuration, which computes the difference between two balanced branches that each performs correlation:

$$\text{EMD}_i(t) = I_A(t-\tau)I_B(t) - I_A(t)I_B(t-\tau). \quad (4)$$

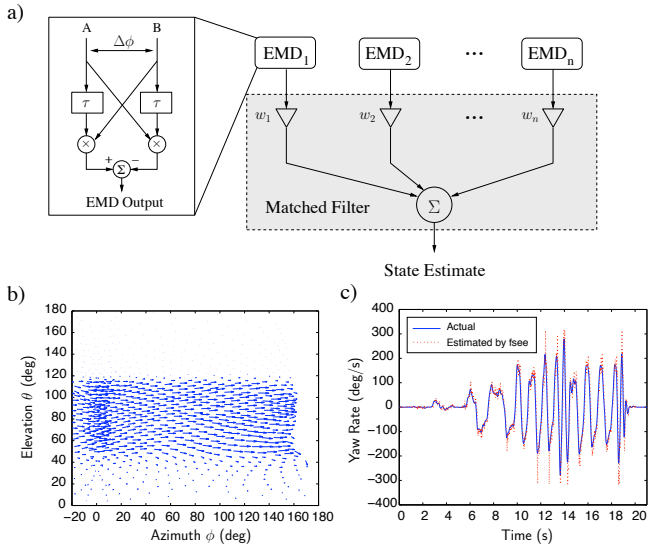

Fig. 4. a) Hassenstein-Reichardt EMD model and matched filter used for state estimation. The EMD computes the temporospatial correlation between inputs from two adjacent ommatidia A and B. $\Delta\phi$ is the angle spacing between the two ommatidia. τ is the time delay appeared in correlation. All the EMD outputs are fed into a matched filter with weights w_j ($j = 1, 2, \ldots, n$) to obtain the corresponding state estimate. b) Matched filter for yaw rate estimation plotted against elevation θ and azimuth ϕ. Each point on the 2D quiver plot corresponds to a point on the eye map ($\phi = 0$: front, $\theta = 0$: top), whereas the length and direction of arrows represent the weights w_j (2D vectors); c) Comparison of yaw rate estimate vs. actual measurement. The estimated yaw rate is obtained from the matched filter shown in b), whereas the actual one is from numerical differentiation of the pose data.

Response of an EMD to a certain moving pattern is nonlinear, nonmonotonic, and sensitive to brightness/contrast [21]. Although the EMD response can be obtained analytically for sinusoidal patterns, no closed form is available for naturalistic patterns. Therefore, the nonlinearity and brightness/contrast dependency have been addressed by building an empirical look-up table from simulation results. In addition, the yaw rate is kept within the monotonic region of EMD response, so that the retinal velocities from each EMD pair can be determined without ambiguity. The retinal velocities are known to give an estimate of the yaw rate through linear weighted summation, using the method proposed in [10]. The weights used in summation correspond to certain specific motion patterns, hence are also called matched filters. A number of different patterns have been identified by electrophysiological recordings on the LPTCs in insects' brain [8]. As opposed to finding the suitable matched filter analytically, in this experiment the weights are obtained from the outputs of the EMDs under given stereotyped motions—yaw rotation in our case. Fig. 4b shows the matched filter pattern, consisting mainly of yaw motion as expected. Pose data from a realistic helicopter trajectory are used to compare the estimation of yaw rate using numerical differentiation vs. a matched filter. Because our helicopter is not equipped with an IMU, numerical differentiation is considered as the ground truth. From the comparison in Fig. 4c, it can be seen that the matched filter approach gives a fairly accurate estimate of the yaw rate.

Fig. 5 and 6 show the estimated pose (x, y, z, θ, ψ, ϕ) of the helicopter near hover and step changes in yaw, respectively. Yaw rate control is realized by simply changing the set point, $\dot{\phi}_{\text{ref}}$, from 0 deg/s to 20 deg/s. The current limitation of state estimation using matched filters, though, is that it cannot provide satisfactory estimate of lateral positions, which is not claimed in previous studies on planar vehicles [22]. This can be expected to happen in aerial vehicle control due to more DOFs involved. The estimated lateral positions, which are given by integrating the translational velocities ($\dot{x}$ and $\dot{y}$) from matched filter outputs, suffer from the cumulative errors of $\dot{x}$ and $\dot{y}$ and will eventually move the helicopter out of the tracking range. Instead, estimation of x and y positions are given directly by pose estimation in the current implementation.

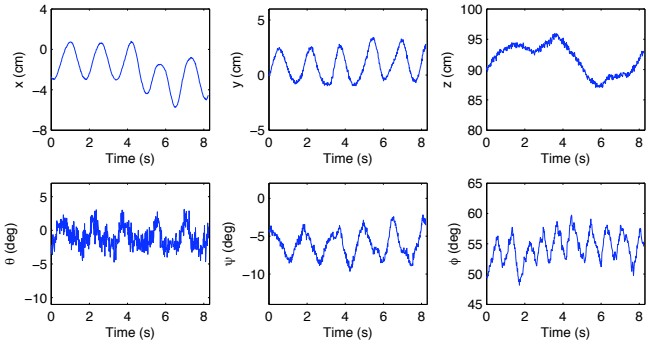

Fig. 5. Position and attitude measurements when the helicopter is in stabilized hover. The oscillations around the equilibrium point are due to air disturbances and/or communication errors of the radio system, and are controlled within a satisfactory range. The non-zero offsets in pitch and roll are caused by inaccuracy in mounting the LED frame.

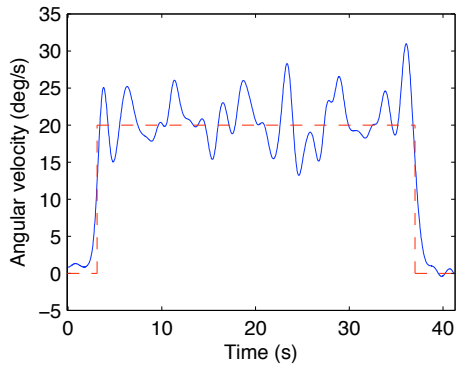

Fig. 6. Yaw rate control. The helicopter is switched from hover to a yaw rate of $\dot{\phi}_{\text{ref}} = 20$ deg/s and back to hover at $t \approx 4$ s and $t \approx 37$ s, respectively, as shown by the red dashed line. The yaw rate computed from pose estimation results is shown in blue.

VII. Conclusions and Future Work

The paper describes a real-time helicopter platform that allows implementation and testing of control algorithms—especially bio-inspired ones, operating in different environments. As opposed to laying out realistic objects and scenes, the environment is specified in a virtual world using a 3D computer model, allowing rapid examination of a large number of environments. The connection between the real world and the virtual environment is provided by a home-made camera system that tracks the pose of the helicopter. As a first step, we have demonstrated flight stabilization near hover and yaw control. These promising results can motivate further investment on this testbed.

The present work, nevertheless, only incorporates certain preliminary features of a fly's sensory system, in particular, visual feedback system. It is not surprising that flies fuse information from multiple sensors to guide their flight maneuvers. For example, halteres, the fly's equivalent of gyroscope, can potentially assist translational velocity estimation by providing accurate yaw rate measurement [13]. Furthermore, this paper does not address the biological control law implemented by insects, which are more interesting from an engineering point of view. Yet the helicopter testbed "immersed" in a virtual environment presented in this paper can be useful in studying more elaborate control algorithms and shed light on the underlying mechanisms in the future.

VIII. Acknowledgments

The authors would like to thank Sawyer Fuller and Francisco Zabala for helpful discussions on helicopter control and hardware implementations. This work is supported in part through the Institute for Collaborative Biotechnology (ICB), the Boeing Corporation, and the California Institute of Technology.

References

[1] M. A. Frye and M. H. Dickinson, "Fly flight: A model for the neural control of complex behavior," *Neuron*, vol. 32, pp. 385–388, 2001.

[2] M. B. Reiser, J. S. Humbert, M. J. Dunlop, D. Del Vecchio, R. M. Murray, and M. H. Dickinson, "Vision as a compensatory mechanism for disturbance rejection in upwind flight," in *American Control Conference (ACC)*, 2004.

[3] M. Ichikawa, H. Yamada, and J. Takeuchi, "A flying robot controlled by a biologically inspired vision system," in *International Conference on Neural Information Processing (ICONIP)*, 2001.

[4] T. Kanade, O. Amidi, and Q. Ke, "Real-time and 3D vision for autonomous small and micro air vehicles," in *IEEE Conference on Decision and Control (CDC)*, 2004.

[5] L. O. Mejias, S. Saripalli, P. Cervera, and G. S. Sukhatme, "Visual servoing for tracking features in urban areas using an autonomous helicopter," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2006.

[6] L. Schenato, X. Deng, W. Wu, and S. Sastry, "Virtual insect flight simulator (VIFS): a software testbed for insect flight," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2001.

[7] F. Ruffier and N. Franceschini, "Visually guided micro-aerial vehicle: automatic take off, terrain following, landing and wind reaction," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2004.

[8] H. G. Krapp and R. Hengstenberg, "Estimation of self-motion by optic flow processing in single visual interneurons," *Nature*, vol. 384, pp. 463–466, 1996.

[9] M. B. Reiser and M. H. Dickinson, "A test bed for insect-inspired robotic control," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 361, pp. 2267–2285, 2003.

[10] J. S. Humbert, R. M. Murray, and M. H. Dickinson, "A control-oriented analysis of bio-inspired visuomotor convergence," in *IEEE Conference on Decision and Control and European Control Conference (CDC-ECC)*, 2005.

[11] J. S. Humbert and M. A. Frye, "Extracting behaviorally relevant retinal image motion cues via wide-field integration," in *American Control Conference (ACC)*, 2006.

[12] M. Epstein, S. Waydo, S. B. Fuller, W. B. Dickson, A. D. Straw, M. H. Dickinson, and R. M. Murray, "Biologically inspired feedback design for *Drosophila* flight," in *American Control Conference (ACC)*, 2007.

[13] W. B. Dickson, A. D. Straw, and M. H. Dickinson, "Integrative model of *Drosophila* flight," *AIAA Journal*, vol. 46, pp. 2150–2164, 2008.

[14] A. Beyeler, J.-C. Zufferey, and D. Floreano, "3D vision-based navigation for indoor microflyers," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2007.

[15] Also available at: http://purl.org/hanshuo/2008/yaw_control_with_fsee (last visited on Feb 8, 2009).

[16] Y. Yoshihata, K. Watanabe, Y. Iwatani, and K. Hashimoto, "Multi-camera visual servoing of a micro helicopter under occlusions," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2007.

[17] Available at: http://code.astraw.com/projects/motmot (last visited on Feb 8, 2009).

[18] D. F. Dementhon and L. S. Davis, "Model-based object pose in 25 lines of code," *International Journal of Computer Vision*, vol. 15, pp. 123–141, 1995.

[19] E. Buchner, "Dunkelanregung des stationaeren flugs der fruchtfliege *Drosophila*. Dipl. thesis," Ph.D. dissertation, Univ. Tuebingen, Tuebingen, Germany, 1971.

[20] B. Hassenstein and W. Reichardt, "Systemtheoretische analyse der zeit-, reihenfolgen-, und vorzeichenauswertung bei der bewegungsperzeption des rüsselkäfers *Chlorophanus*," *Zeitschrift Fur Naturforschung*, vol. 11b, pp. 513–524, 1956, ("System theoretical analysis of the time-, order-, and sign-evaluation in movement perception by the weevil *Chlorophanus*," *Journal of Natural Research*) (English translation).

[21] R. O. Dror, D. C. O'Carroll, and S. B. Laughlin, "Accuracy of velocity estimation by Reichardt correlators," *J. Opt. Soc. Am. A*, vol. 18, pp. 241–252, 2001.

[22] S. B. Fuller, A. D. Straw, M. Epstein, S. Waydo, W. B. Dickson, M. H. Dickinson, and R. M. Murray, "Geometric analysis of Hassenstein-Reichardt elementary motion detectors and application to control in a fruit fly simulator and a robot," in *International Symposium on Flying Insect Robotics*, Ascona, Switzerland, 2007.