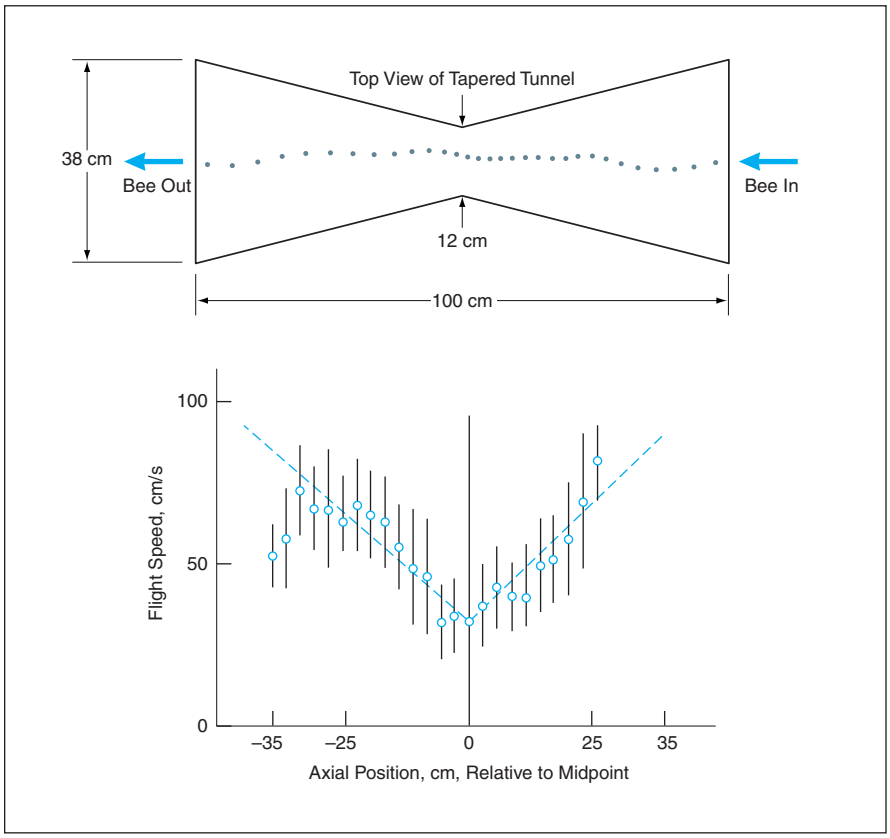
Insect-Inspired Flight Control for Unmanned Aerial Vehicles

Relatively simple sensory and computing systems would generate remarkably effective control in flight to allow close-up approach to hard terrain.

NASA's Jet Propulsion Laboratory, Pasadena, California

Flight-control and navigation systems inspired by the structure and function of the visual system and brain of insects have been proposed for a class of developmental miniature robotic aircraft called "biomorphic flyers" described earlier in "Development of Biomorphic Flyers" (NPO-30554), *NASA Tech Briefs*, Vol. 28, No. 11 (November 2004), page 54. These form a subset of biomorphic explorers, which, as reported in several articles in past issues of *NASA Tech Briefs* ["Biomorphic Explorers" (NPO-20142), Vol. 22, No. 9 (September 1998), page 71; "Bio-Inspired Engineering of Exploration Systems" (NPO-21142), Vol. 27, No. 5 (May 2003), page 54; and "Cooperative Lander-Surface/Aerial Microflyer Missions for Mars Exploration" (NPO-30286), Vol. 28, No. 5 (May 2004), page 36], are proposed small robots, equipped with microsensors and communication systems, that would incorporate crucial functions of mobility, adaptability, and even cooperative behavior. These functions are inherent to biological organisms but are challenging frontiers for technical systems. Biomorphic flyers could be used on Earth or remote planets to explore otherwise difficult or impossible to reach sites. An example of an exploratory task of search/surveillance functions currently being tested is to obtain high-resolution aerial imagery, using a variety of miniaturized electronic cameras.

The control functions to be implemented by the systems in development include holding altitude, avoiding hazards, following terrain, navigation by reference to recognizable terrain features, stabilization of flight, and smooth landing. Flying insects perform these and other functions remarkably well, even though insect brains contains fewer than 10^{-4} as many neurons as does the human brain. Although most insects have immobile, fixed-focus eyes and lack stereoscopy (and hence cannot perceive depth directly), they utilize a number of ingenious strategies for perceiving, and navigating in, three dimensions. Despite their lack of stereoscopy, insects infer distances to potential obstacles and other objects from image motion cues that result from their own motions in the environment. The concept of motion of texture in images as a source of motion cues is denoted generally as the concept of optic or optical flow. Computationally, a strategy based on optical flow is simpler than is stereoscopy for avoiding hazards and following terrain. Hence, this strategy offers the potential to design vision-based control computing subsystems that would be more compact, would weigh less, and would demand less power than would subsystems of equivalent capability based on a conventional stereoscopic approach.



Bees Flew Through a Tapered Tunnel in an experiment on how they use visual cues to control flight. The bees decelerated as the tunnel narrowed, then accelerated as it widened. The dashed line shows the flight-speed profile that one would obtain if the bees were to hold the angular velocity of the images of the walls constant at 320° per second. An obvious advantage of regulating flight speed to obtain constant image speed is that an insect would automatically decelerate to a safer speed to negotiate a narrow passage.

These control loops for stabilizing attitude and/or holding altitude would include optoelectronic ocelli and would be based partly on dragonfly ocelli-simple eyes that exist in addition to the better-known compound eyes of insects. In many insects the ocelli only detect changes in light intensity and have minimal observable effect on flight. In dragonflies, the ocelli play an important role in stabilizing attitude with respect to dorsal light levels. The control loops to be implemented would incorporate elements of both dragonfly ocellar functions and optical flow computation as derived from principles observed in honeybee flight.

On Earth, bees use sky polarization patterns in the ultraviolet part of the spectrum as a direction reference relative to the position of the Sun. A robotic direction-finding technique based on this concept is more robust in comparison with a simple Sun compass because the ultraviolet polarization pattern is distributed across the entire sky on Earth and is redundant and hence can be extrapolated from a small region of clear sky in an elsewhere cloudy sky that hides the Sun.

A bee tends to adjust its flight speed to maintain a constant optical flow (that is,

a constant angular velocity of the image of the environment) over its compound eye (see figure). Consistent with this strategy, a bee utilizes the following simple control laws when approaching a landing site on a flat surface:

1. The optical flow of the surface is held constant throughout the descent.
2. Forward speed is held proportional to vertical speed throughout the descent.

This simple combination of control laws enables a smooth landing with minimal computation. The forward speed and rate of descent are reduced together, and are both close to zero at touchdown. No knowledge or measurement of instantaneous speed or height above the ground are necessary. This combination of control laws can readily be modified for a biomorphic flyer, which has a nonzero stalling speed.

This work was done by Sarita Thakoor of Caltech for **NASA's Jet Propulsion Laboratory** *and by G. Stange, M. Srinivasan, and Javaan Chahl of Australian National University and Butler Hine and Steven Zornetzer of Ames Research Center for the NASA Intelligent Systems Program. Further information is contained in a TSP (see page 1).*
NPO-30545

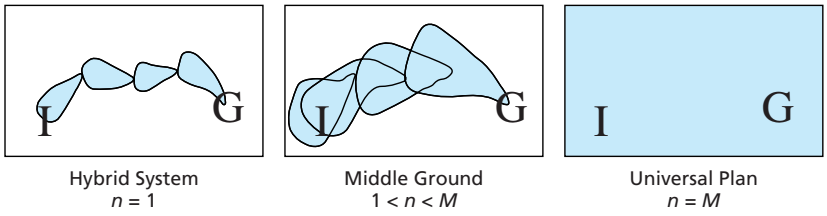
Domain Compilation for Embedded Real-Time Planning

Robustness is increased at the price of a moderate increase in complexity.

NASA's Jet Propulsion Laboratory, Pasadena, California

A recently conceived approach to automated real-time control of the actions of a robotic system enables an embedded real-time planning algorithm to develop plans that are more robust than they would otherwise be, without imposing an excessive computational burden. This approach occupies a middle ground between two prior approaches known in the art as the universal-plan and hybrid approaches.

Ever since discovering the performance limitations of taking a sense-plan-act approach to controlling robots, the robotics community has endeavored to follow a behavior-based approach in which a behavior includes a rapid feedback loop between state estimation and motor control. Heretofore, system architectures following this approach have been based, variously, on algorithms that implement universal plans or algorithms that function as hybrids of planners and executives. In a typical universal-plan case, a set of behaviors is merged into the plan, but the system must be restricted to relatively small problem domains to avoid having to reason about too many states and represent them in the plan. In the hybrid approach, one implements actions as small sets of behaviors, each applicable to a limited set of circumstances. Each action is intended to bring the system to a subgoal state. A planning algorithm is used to string these actions together into a sequence to traverse the state space from an initial or current state to a goal state. The hybrid approach works well in a static environment, but it is inherently brittle in a dynamic environment because a failure can occur when the environment strays beyond the region of applicability of the current activity.



Note: Each rectangle symbolizes state space, and colored areas represent portions of the state space covered by a universal(*n*) plan for target goals from the initial state I to the goal state G.

Coverage of State Space by an *n*-level plan increases with *n*.

In the present approach, a system can vary from the hybrid approach to the universal-plan approach, depending on a single integer parameter, denoted n, which can range from 1 to a maximum domain-dependent value of M. As illustrated in the figure, $n = 1$ represents the hybrid approach, in which each linked action covers a small part of the state space of the system. As n increases, the portion of state space associated with each action and its subgoal grows. When n reaches M, coverage extends over the full state space, so that the system contains a universal plan.

Through incorporation of an embedded real-time planning algorithm that follows this middle-ground approach, a hybrid system can be made much more robust in a dynamic environment. In such a system, the planning algorithm passes the current subgoals (instead of activities) to an executive algorithm. The executive algorithm then uses the real-time planning algorithm to determine when to perform which action until it determines either that the current subgoals have been reached or that they cannot be reached within n steps. If the current subgoals have been reached, the planning algorithm gives new subgoals to the executive algorithm. If it has been determined that the current subgoals cannot be reached, the planning algorithm must alter the sequence of actions.

A structure for finding the next step on an n or fewer step path to a subgoal is called a universal(n) plan. Because complexity increases sharply with n, it is necessary to choose n small enough to avoid an excessive computational burden but large enough that it is possible to make a universal(n) plan that makes the system robust in the sense that it can reach each given subgoal state from any state in a large region of the state space.

This approach involves utilizing knowledge compilation research by implementing an off-line compiler that generates a universal(n) plan from a system description. In the first step of a two-step process, the system description is converted into a logical expression, in what is known as a conjunctive nor-