A bio-plausible design for visual pose stabilization

Shuo Han, Andrea Censi, Andrew D. Straw, and Richard M. Murray

Abstract—We consider the problem of purely visual pose stabilization (also known as servoing) of a second-order rigid-body system with six degrees of freedom: how to choose forces and torques, based on the current view and a memorized goal image, to steer the pose towards a desired one. Emphasis has been given to the bio-plausibility of the computation, in the sense that the control laws could be in principle implemented on the neural substrate of simple insects. We show that stabilizing laws can be realized by bilinear/quadratic operations on the visual input. This particular computational structure has several numerically favorable characteristics (sparse, local, and parallel), and thus permits an efficient engineering implementation. We show results favorable characteristics (sparse, local, and parallel), and thus permits an efficient engineering implementation. We show results of the control law tested on an indoor helicopter platform.

I. INTRODUCTION

The question of how the brain utilizes millions of neurons, presumably in a highly parallel fashion, to make timely decisions from information-rich sensory inputs still remains unanswered. This holds even for one of the simplest animals, the fruit fly, which has only about 300,000 neurons, orders of magnitude fewer than the human brain. Engineers marvel at the series of fast and robust behaviors implemented on the poor computational support of slow and noisy neurons. Particularly interesting is the processing of visual information (Fig. 1), which biologists believe is a dominant sense controlling flight in many species of insects. Examples of classical studied behaviors include turning towards features of interest [1] and altitude/velocity control [2]. In this paper, we are especially interested in the visual behaviors that allow insects to return to or to stay at a particular location in space (hovering and homing).

The hovering and homing behaviors are instances of what in robotics is called visual servoing, which we will refer to as pose stabilization. Our goal in this paper is to solve this problem while respecting the constraints of a “bio-plausible” computation. The motivation is twofold: on the one hand, this might provide clues for realizing a computational model of the insects behavior, useful to biologists; on the other hand, it might inspire useful engineering principles to realize similar robust systems. The question of what can be considered bio-plausible, is, of course, loosely defined. In brief, we consider something to be bio-plausible if it can be realized on neural circuits [3]. The computation should be implementable using parallel asynchronous units; in insects, a model of computation that is well accepted as bio-plausible is a local nonlinear operation followed by wide-field linear integration. An example of computation that could not be considered bio-plausible would be (point) feature-based visual servoing: not as much for implementing a feature-extraction algorithm, but because the logic of manipulating a “list of features” and matching them does not appear to be easily implementable on a neural substrate. An alternative is using the raw luminance, which, interestingly, is a new trend in robotics [4]–[7]. The feasibility of this scheme for insects has been considered, and an experimental assessment concluded that “view-based homing with panoramic images is in principle feasible in natural environments and does not require the identification of individual landmarks” [8].

Related work: Visual servoing is a classical problem in industrial robotics, with applications in controlling robot manipulators [9], [10] or visual navigation [11], [12]. The classical approach normally consists of extracting and tracking fiducial point/segments from the image [13]. However, in the last few years, several methods have been proposed that do not rely on features, but instead use the raw luminance directly [4]–[7]. These are formulated as computing the gradient of an image dissimilarity measure; the obvious but not unique choice is the squared error over the field of view. An issue that appears not
to be completely solved at the moment is formulating sufficient conditions for convergence, especially because the gradient directions depend on the distance to the objects, which is usually unknown and thus must be approximated.

**Contribution:** In this paper, we present proportional-derivative control laws that solve the visual servoing problem for a second-order system controlled in force/torque and demonstrate the pose stabilization results on an indoor helicopter testbed. We do not rely on velocity or inertial sensors; we only assume to know the current visual observations and a goal image taken at the desired pose (Fig. 2). The computation essentially consists of a tensor multiplication that involves the goal image and the current observations (Fig. 3). This paper is the ideal continuation of [14], where we restricted ourselves to purely rotational motion. The main improvement in this paper is that we now solve the problem for joint rotation and translation instead of pure rotation. This is not a trivial extension, because when performing translation, the visual observation will be affected by the distance to objects, which is not instantaneously observable.

With respect to related work in engineering on feature-free visual servoing, we offer the following contributions:

- We consider a second-order system: the control inputs are torque/force rather than velocities; this requires estimating velocity from vision to create suitable damping laws.
- We prove the stability of purely visual control laws independent of (or robust to changes in) the distance profile, given that the environment satisfies certain conditions; previous work did propose laws assuming constant distance, but did not prove convergence (e.g., [6]) or proved it assuming the availability of point features (e.g., [15]).

The paper is organized as follows. Section II introduces the notation used and preliminaries about the model. Section III–V show, in the order of complexity: 1) control laws for the first-order system, controlled in velocity, 2) damping control laws that stabilize the velocities to zero, and 3) proportional-derivative control of the second-order system. Section VI discusses the computational structure of the proportional-derivative control law and several of its favorable numerical features that allow an efficient implementation. Section VII shows numerical tests using simulated visual input of a fruit fly and discusses the bio-plausibility of the control law. Section VIII presents experiments on an indoor helicopter testbed.

## II. Preliminaries

We consider the general 6-degree-of-freedom (6-DOF) pose stabilization problem using the output from a generalized vision sensor. In the section, we will introduce the model of the system dynamics, describe the model of the vision sensor, and formalize the visual control problem.

### A. Dynamic model

We consider the fully actuated rigid body motion on the special Euclidean group $\text{SE}(3)$. A pose in $\text{SE}(3)$ can be described by the body position $p \in \mathbb{R}^3$ and the body attitude $r$, represented as a $3 \times 3$ rotation matrix, or, more formally, an element in the special orthogonal group $\text{SO}(3)$.

For many traditional visual servoing applications where the velocities can be controlled directly, it suffices to consider the system kinematics (also referred to as the first-order system):

$$\begin{align*}
\dot{r} &= r (\omega)^\wedge, \\
\dot{p} &= rv,
\end{align*}$$

where $\omega \in \mathbb{R}^3$ is the angular velocity and $v \in \mathbb{R}^3$ is the linear velocity, both expressed in the body frame. The “hat map” $(\cdot)^\wedge$ maps a vector $x \in \mathbb{R}^3$ to a skew-symmetric matrix $x^\wedge \in \mathbb{R}^{3 \times 3}$ such that $(x^\wedge)y = x \times y$, with “$\times$” being the ordinary cross product in $\mathbb{R}^3$ [16]. Control in velocity will be discussed primarily in Section III only.

In other applications, such as stabilizing aerial vehicles, where the control inputs are torques and forces, the full dynamics (also referred to as the second-order system) need to be considered:

$$\begin{align*}
\dot{r} &= r (\omega)^\wedge, \\
\dot{\omega} &= (\llbracket \omega \rrbracket) \times \omega + \tau, \\
\dot{p} &= rv, \\
v \dot{v} &= mv \times \omega + f,
\end{align*}$$

where $\mathbb{I} \in \mathbb{R}^{3 \times 3}$ is the moment of inertia; $m \in \mathbb{R}$ is the body mass; $\tau \in \mathbb{R}^3$ is the control torque; $f \in \mathbb{R}^3$ is the control force. Most of this paper will address the issue of controlling such systems. Although this model does not fully capture the dynamics of some platforms such as a helicopter (underactuated, as will be discussed in Section VIII), it provides a good approximation at least near the equilibrium (i.e., at hovering).

### B. Sensor model

We assume to have available only the output of a vision sensor. With good generality, we model the vision sensor as a device that at each time $t$ returns a series of values $y(s_i, t)$, each corresponding to the observed luminance from a pixel $i$ in the direction $s_i \in \mathbb{S}^2$. Here $\mathbb{S}^2 \subset \mathbb{R}^3$ denotes the unit sphere. There are two important aspects of such a model:

- **Sensor output:** The model gives raw luminance as the output. As a comparison, traditional visual servoing literature often assumes to know the positions of fiducial points on the image plane (except in, e.g., [4]–[7]). The formulation also works if an instantaneous point-wise filter, such as contrast normalization, is applied to the raw luminance.
- **Sensor geometry:** Using the directions $s_i \in \mathbb{S}^2$ to model the disposition of “pixels” on the sensor makes the model equally apt for a normal perspective camera, a catadioptric camera, or the compound eye of a fruit fly (see Fig. 1).

In deriving the theory, we make a number of simplifying assumptions: 1) We are able to sample the entire visual field from “a continuum of pixels”, i.e., we know $y(s, t)$
for all $s \in \mathbb{S}^2$; 2) We can observe both $y(t)$ and $\dot{y}(t)$; 3) We ignore the effect of occlusions. Note that, however, in the simulations we do incorporate non-ideal factors such as blurring, sensor dynamics, and occlusions, some of which appear in the experiments as well. We will also discuss how to approximate/discretize the control law when the visual field appears in the experiments as well. We will also discuss how blurring, sensor dynamics, and occlusions, some of which in the simulations we do incorporate non-ideal factors such need the relation between the distance to the object in direction $s$ in the way, on the “nearness” of the space of images:

$$
\dot{y}(s,t) = \mu(s)\nabla_s y(s,t)^* v + (s \times \nabla_s y(s,t))^* \omega.
$$

or more compactly, dropping s and t, and defining the operator $S$ as: $S y \triangleq s \times \nabla_s y$,

$$
\dot{y} = \mu(\nabla y)^* v + (Sy)^* \omega.
$$

The two terms in $\dot{y}$ describes the contributions to change in luminance by translational velocity $v$ and rotational velocity $\omega$, respectively. This is not a new result and has appeared, often in a disguised form with different notation, in many other papers. Here we have adopted the convention that $\nabla y$, when evaluated at a specific $s \in \mathbb{S}^2$, is an element of $\mathbb{R}^3$; i.e., we think of it as an “arrow attached to the sphere $\mathbb{S}^2$.”

### C. Control problem

We assume that the agent knows a “goal” image $g(s)$ taken at the goal pose $q_g = (r_g, p_g)$. The problem we wish to solve can be stated as follows:

**Problem 1 (Visual pose stabilization):** Given the goal image $g$, design a stabilizing control law for $\tau$ and $f$, depending only on $y$, $\dot{y}$, $g$, such that $q \rightarrow q_g$ and $(\omega, v) \rightarrow 0$.

Without loss of generality, we will assume that $(r_g, p_g) = (\text{Id}, 0)$, the identity element of $\text{SE}(3)$.

### III. VISUAL CONTROL IN VELOCITY

We first consider the problem of controlling the first-order system (1), assuming that we can impose $\omega$ and $v$ directly. Motivated by a control law constructed from gradient descent, we obtain a stabilizing control law that does not rely on knowledge of the structure of the environment (i.e., the nearness $\mu(s)$). Not only will the result be useful for controlling the first-order system (1) per se, it will also be important in constructing a stabilizing control law for the full second-order system (2), as will be discussed in Section V.

#### A. Control with the knowledge of distance

A natural way to define a stabilizing control law is to use the gradient of the quadratic cost function $J$ defined in the space of images:

$$
J(q) = \frac{1}{2} \int_{s \in \mathbb{S}^2} (y(s) - g(s))^2 dS.
$$

This integration is taken over the entire visual sphere $\mathbb{S}^2$, with $S$ as the unique rotation-invariant measure on $\mathbb{S}^2$. We will use a shorthand notation to denote this integration:

$$
\langle f \rangle \triangleq \int_{s \in \mathbb{S}^2} f(s) dS.
$$

Using this notation, equation (5) can be written compactly as

$$
J(q) = \frac{1}{2} \langle (y - g)^2 \rangle.
$$

Here, the cost function $J$ is expressed as a function of $q$ because $y$ varies with $q$. Again, note that $J$ is computed directly from the luminance, not from the error in the positions of point features.

However, simply applying the (negative) gradient as the control input may fail to drive the pose $q$ to $q_g$; as an extreme case, an environment with uniform luminance ($y(s) = \text{const}$) will make $q$ unobservable from the visual input $y$ alone. Therefore, some additional condition on the environment is expected for the gradient control law to work, which is stated formally in the following proposition.

**Proposition 1 (Exact gradient):** The gradient control law

$$
\begin{align*}
\dot{\omega} &= \langle (Sy)(g - y) \rangle, \\
\dot{v} &= \langle \mu(\nabla y)(g - y) \rangle,
\end{align*}
$$

locally stabilizes the first-order system (1) asymptotically if the $6 \times 6$ “contrast matrix”

$$
C(q) = \begin{bmatrix}
\langle SyS^*y \rangle & \langle \mu Sy \nabla y^* \rangle \\
\langle \mu yS^*y \rangle & \langle \mu^2 \nabla y\nabla y^* \rangle
\end{bmatrix}
$$

is strictly positive definite at $q = q_g$. Note again we have used the convention that both $Sy$ and $\nabla y$ (evaluated at a specific $s$) are vectors in $\mathbb{R}^3$.

**Proof:** (Sketch only, see [17] for detail) We first note that we have to deal with quantities living on manifolds. Therefore there is some additional difficulty with respect to Euclidean spaces. For example, the gradient is not “a vector in $\mathbb{R}^n$,” and the Hessian is not “a matrix in $\mathbb{R}^{n \times n}$.” Our first step is a change of coordinates for the attitude. There are many choices of Euclidean coordinates for $\text{SO}(3)$, including Euler angles, quaternions, etc. Here we choose the exponential coordinates $\varphi \in \mathbb{R}^3$ due to its compact written form:

$$
\varphi = \text{Log}(r)^*, \quad r = \text{Exp}(\varphi),
$$

where Log and Exp are the matrix logarithm and exponential, and $(\cdot)^*$ is the inverse of the hat map $(\cdot)^\wedge$. Note in particular that $(r = \text{Id}) \iff (\varphi = 0)$. Under such a representation, the kinematics (1) can be rewritten as

$$
\begin{align*}
\dot{\varphi} &= \omega, \\
\dot{p} &= \text{Exp}(\varphi^\wedge)v.
\end{align*}
$$

Note that both $\omega$ and $v$ depend on $y$, which is a function of $\varphi$ and $p$. To prove local stability, it suffices to investigate the linearization of the system (8) around the equilibrium $(\varphi, p) = (0, 0)$:

$$
\dot{\xi} = -C(q_g)\xi, \quad \xi = \begin{bmatrix} \varphi \\ p \end{bmatrix},
$$

where $C(q_g)$ is the $6 \times 6$ “contrast matrix” at $q = q_g$.
where $C$ is the “contrast matrix”. Therefore, the system is locally asymptotically stable if $C(q_g) > 0$.

The above condition on the contrast matrix imposes requirements on the visual appearance (reflected in the visual input $y$) and structure (reflected in the nearness profile $\mu$) of the environment. If the condition fails to hold (i.e., the matrix $C$ loses rank), there exists a certain “direction” in SE(3) such that the visual observation will not change when moving along this direction infinitesimally near the goal $q_g$. In other words, the environment possesses some kind of spatial invariance that makes Problem 1 impossible to solve with visual input.

Alternatively, we note that the contrast matrix $C$ is in fact the Hessian of $J$, expressed in the exponential coordinates $\varphi$ and $p$. The Hessian of $J$ being strictly positive definite at $q_g$ ensures that $q_g$ is an isolated minimum of $J(q)$. This means that $q = q_g$ is the only (local) solution of the equation $y(s) = g(s)$, which is equivalent to $q_g$ being observable from the visual input $y$.

B. Control without the knowledge of distance

One drawback of the above control law (7) is that $v$ depends on the unknown nearness profile $\mu(s)$. A possible solution would be to estimate $\mu$ by solving a structure from motion problem (e.g., [18–20]). Our approach, instead, will show that it is possible to use a control law that does not depend on $\mu$, if we are only concerned with local stability near $q_g$. Since this control law does not contain $\mu$, it is only an approximation of the exact gradient direction, and requires a different condition other than that in Proposition 1 to hold in order to guarantee local stability:

**Proposition 2 (Approximate gradient):** The approximate gradient control law

\[
\begin{align*}
\omega &= \langle (Sy)(g-y) \rangle, \\
\dot{v} &= \alpha \langle (\nabla y)(g-y) \rangle
\end{align*}
\]

locally stabilizes the first-order system asymptotically, if there exists $\alpha > 0$ such that the $6 \times 6$ “modified contrast matrix”

\[
\tilde{C}(q, \alpha) \equiv \begin{bmatrix}
\langle SyS^* \rangle & \alpha \langle \nabla yS^* \rangle \\
\alpha \langle \nabla yS^* \rangle & \alpha \langle \mu \nabla y S^* \rangle
\end{bmatrix}
\]

is strictly positive definite at $q = q_g$. Note that $\alpha$ is a constant scalar and does not depend on the directions $s$ as opposed to $\mu(s)$ in (7).

**Proof:** (Sketch only, see [17] for detail) The proof of stability is similar to the previous one. The linearization of the system (8) is: $\dot{\xi} = -F\xi$, where

\[
F = \begin{bmatrix}
\langle SyS^* \rangle & \alpha \langle \nabla yS^* \rangle \\
\alpha \langle \nabla yS^* \rangle & \alpha \langle \mu \nabla y S^* \rangle
\end{bmatrix}.
\]

Note that, at $q = q_g$, the modified contrast matrix $\tilde{C}$ is the symmetric part of $F$: $\tilde{C}(q_g, \alpha) = \frac{1}{2} (F + F^*)$. If $\tilde{C} > 0$, stability can be immediately concluded from the Lyapunov function $V = |\xi|^2$.

**Remark 1:** Under at least some special cases, the positive definiteness of $\tilde{C}(q, \alpha)$ is equivalent to the positive definiteness of $C(q)$. For example, when the environment is spherical ($\mu(s) = \pi > 0$), if we let $\alpha = \pi$ in $\tilde{C}(q, \alpha)$, then $\tilde{C} = C$. This is expected because knowledge of the nearness $\mu$ becomes unimportant for a spherical environment.

IV. VISUAL VELOCITY DAMPING

Our ultimate goal is to construct a proportional-derivative controller, which is necessary to achieve asymptotic stability of the second-order system (2). The result from the previous section can be viewed as the proportional part (i.e., “proportional” to the error in pose). In this section, we switch the focus to designing the derivative part (i.e. the time derivative of the pose), in order to introduce artificial damping to the rigid body dynamics. Equivalently, this means one needs to choose force and torque such that the velocities $\omega$, $v$ are driven to 0. Traditionally, the derivative part is constructed to be proportional to the velocities estimated from sensory input, here being $y$ and $\dot{y}$. We will first examine the least-squares estimates, but will soon switch to a much simpler solution using bilinear forms. Despite the simplicity of the latter solution, the resulting control law is effective in that it will stabilize the velocities to 0.

A. Velocity estimation using least squares

The sensory input that encodes information on the velocities is the temporal change of luminance $\dot{y}$. Recall the relationship between $\dot{y}$ and $\omega$, $v$ in (4) and note that $\dot{y}$ is linear in $\omega$, $v$, respectively. In the pure rotation ($v = 0$) or pure translation ($\omega = 0$) case, a straightforward solution is to estimate the velocities directly using least squares:

**Lemma 1:** The least-squares estimates of the velocities are

\[
\hat{\omega}_{LS} = \langle SyS^* \rangle^{-1} \langle Sy \rangle \hat{y}, \quad v_{LS} = \langle \mu \nabla y \rangle \langle \mu \nabla y^* \rangle^{-1} \langle \mu \nabla y \rangle \hat{y}\ (v = 0),
\]

Although the above only holds for separate motions and will not be our final solution, they provide intuition for the bilinear velocity estimation introduced in the next.

B. Velocity estimation using bilinear forms

For control purposes, it is likely that we do not need to estimate the velocities perfectly, since the feedback loop can usually tolerate some uncertainties. We now attempt to simplify the above velocity estimates, hoping to make the computation more bio-plausible. First, we will remove the matrix inverse from the least-squares estimates. Second, motivated by the previous section, we will drop the (unknown) nearness $\mu(s)$. This gives the following velocity estimates that are bilinear (denoted as “BL”) forms of $y$ and $\dot{y}$, namely, linear in $y$ and $\dot{y}$ separately:

\[
\omega_{BL} \triangleq \langle (Sy) \dot{y} \rangle, \quad v_{BL} \triangleq \langle (\nabla y) \dot{y} \rangle.
\]

Surprisingly, as we will see, despite such significant simplification, they do not prevent us from obtaining a stabilizing
damping control law. One insight is that the resulting approxi-
mate velocity estimates are “good enough” because they will 
remain “close enough” (or more precisely, within 90 deg) from 
the true velocities (in the case of separate motions):

**Proposition 3**: In the case of separate motions, the bilinear 
velocity estimates are related to the true velocities as

\[
\omega^* \omega_{BL} \geq 0, \quad v^* v_{BL} \geq 0.
\]

**Proof**: For pure rotation, \( \dot{y} = (Sy)^* \omega \) and \( \omega^* (Sy)^* \omega \geq 0 \).

Similarly, for pure translation, \( \dot{y} = \mu (\nabla y)^* v \) and \( v^* v_{BL} = v^* (\mu \nabla y \nabla y^*) v \geq 0. \)

Omitting the matrix inverse makes the velocity estimator
arbitrarily inaccurate in scale, without more constraints on
the environment. In fact, suppose that \( \omega_{BL} = \omega \) in a certain
environment \( y \). For another environment that is twice as bright:
\( y' = 2y \), the resulting \( \omega_{BL} \) will be off by a factor of 4.

Nevertheless, we will show that \( \omega_{BL} \) and \( v_{BL} \) are useful for
control purposes, since the unknown gain factor plays less role
in system stability.

C. Velocity damping using bilinear velocity estimates

We will now prove the main result of this section: the con-
tral law constructed from the bilinear estimates will regulate
the velocities to 0. This is actually not immediate, because
the we have been treating rotational and translational motions
separately, whereas the dynamics of \( \omega \) and \( v \) interact in
practice. The condition for this to hold is again related to the
modified contrast matrix. This is not too surprising because
we have dropped the nearness \( \mu(s) \) from the least-squares
estimates like in the previous section.

**Proposition 4 (Visual damping)**: Assume that \( \tilde{C}(q, \alpha) > 0 \).
Then the control law

\[
\begin{cases}
\tau = -k_d \omega_{BL} = -k_d (Sy)\dot{y}, \\
f = -\alpha k_d v_{BL} = -\alpha k_d (\nabla y)\dot{y},
\end{cases}
\]

with \( k_d > 0 \), globally stabilizes \( \omega, v \) to 0.

**Proof**: (Sketch only, see [17] for detail) The proof follows
from a standard Lyapunov argument by using the kinetic
energy as the Lyapunov function.

Note that the velocity control law in the previous section
also shares the same particularly simple computational form,
being bilinear in \( y \) and \( (g - y) \). We will further discuss its
consequences in later sections.

V. VISUAL CONTROL IN FORCE/TORQUE

By combining the results from the previous two sections, we
now state the main theoretical result of this paper, namely the
visual proportional-derivative (PD) control of the full rigid-
body dynamics, with force \( f \) and torque \( \tau \) as the control
inputs. Aside from the condition on the modified contrast
matrix \( \tilde{C} \), the only additional requirement to ensure stability
is that the damping factor (i.e., \( k_d \)) is “large enough”.

**Proposition 5 (Proportional-derivative control)**: Assume
that the modified contrast matrix \( \tilde{C}(q, \alpha) \) is positive definite
at \( q = q_o \). Then the control law

\[
\begin{cases}
\tau = I (Sy)(g - y) - k_d (\nabla y)\dot{y}, \\
f = \alpha m (\nabla y)(g - y) - \alpha m k_d (\nabla y)\dot{y},
\end{cases}
\]

guarantees local asymptotic stability of the second-order system
(2) near \( q = q_o \) for large enough \( k_d \).

**Proof**: (Sketch only, see [17] for detail) Linearize the
system (2) near \( q = q_o \). The stability can be shown by a
standard Lyapunov argument using the following Lyapunov function:

\[ V = \xi^* \tilde{C} \xi + \frac{1}{2} z^* z + k_d^{-1} \xi^* z, \]

where \( \xi = [\varphi^* p^*]^* \) and \( z = [\omega^* v^*]^* \). Aside from various
extra gain factors, the above control law is a direct combination
of the control laws in velocity (“proportional” part) and the
velocity damping laws (“derivative” part) from Proposition 2
and 4, respectively.

VI. COMPUTATIONAL STRUCTURE

So far we have been assuming a spatially continuous visual
sensor to derive the theoretical results. However, since physical
visual sensors (cameras, fly’s compound eye) have only a finite
number of photodetectors, the PD control law needs to be dis-
cretized before being implemented. This will also give insights
on the computational structure of the control law, especially its
localness (which implies sparsity) in computation.

Recall that the directions of pixels are denoted as \( s_i \in S^2 \),
and define the discretized visual input as \( g_i = g(s_i) \) and \( y_i = y(s_i) \).
Now \( g \) and \( y \) are vectors in \( \mathbb{R}^n \), where \( n \) is the total
number of pixels. We also need to discretize the differential
operators \( S \) and \( \nabla \) to approximate the result of differentiation
at each pixel locations for an arbitrary input image \( y \). In an
informal notation, we look for tensors \( A \) and \( B \) that act on
discretized images and satisfy:

\[ \text{Discretize } [Sy] \approx Ay, \quad \text{Discretize } [\nabla y] \approx By. \]

Arithmetically, both \( A \) and \( B \) can be thought as \( n \times n \times 3 \) arrays:
- in the above equations, they act on vectors in \( \mathbb{R}^n \) and
- return \( n \times 3 \) matrices. We will use the notation \( A_{ijk} \) to denote
the entries of a tensor \( A \), and a missing subscript to imply the
collection of all the entries in the corresponding dimension
(like the operation “,” on array indices in MATLAB/Octave).
For example, \( A_{ijk} \) represents a vector in \( \mathbb{R}^3 \).

One choice of approximation is to apply spatial smoothing
before differentiation, akin to the Sobel operator used in
computer vision for edge detection. The difference is that here
the approximated differentiation does not assume a uniformly
sampled visual field, and can be applied to an arbitrary
disposition of pixels. Under this approximation, the tensors
\( A \) and \( B \) can be obtained as (up to normalization factors,
see [17] for detail):

\[
\begin{align*}
A_{ij} &= -q'(\alpha_{ij}) \cdot (s_i \times s_j) / \sin(\alpha_{ij}), \\
B_{ij} &= -q'(\alpha_{ij}) \cdot (I - s_i s_i^*) s_j / \sin(\alpha_{ij}),
\end{align*}
\]

(14)
where \( I \) is the \( 3 \times 3 \) identity matrix, \( q \) is a smooth kernel function (e.g., Gaussian), \( q' \) denotes its derivative, and \( \alpha_{ij} = \cos^{-1}(s_i \cdot s_j) \) is the angle formed by \( s_i \) and \( s_j \). We adopt the convention that \( A_{si} = B_{si} = 0 \). Note that, if chosen appropriately, the kernel \( q \) can be localized and endows a sparse approximation for \( A \) and \( B \). This important feature makes the method numerically favorable during implementation, despite that the number of entries in \( A \) and \( B \) could be large for a practical sensor. Moreover, the localness of the kernel also implies that this method could have an efficient implementation on existing parallel computational architectures (e.g., GPU), where access to non-local data is expensive. More discussions on the bio-plausibility of such computation will be given later in Section VII.

Under the new (discrete) notation, the previous PD control law (13) becomes:

\[
\begin{align*}
\mathbf{f}_k &= \sum_{i,j} y_i A_{ijk} [(g_j - y_j) - k_d y_j], \\
\mathbf{τ}_k &= \sum_{i,j} y_i B_{ijk} [(g_j - y_j) - k_d y_j].
\end{align*}
\] (15)

For conciseness, we have ignored some constant gain factors such as \( I \) and \( n \) and collected the two parts in (13). The index \( k \) spans the three components of force and torque; the indices \( i \) and \( j \) range from 1 to \( n \): \( 1 \leq i,j \leq n \). More concretely, in MATLAB/Octave, synthesis of the discretized control law (15) will look like the following:

```matlab
% n: total number of pixels
% For perspective cameras, n = width * height
q = reshape(goal_image, n, 1);
y = reshape(current_image, n, 1);
for k = 1:3
    torque(k) = y' * A(:,:,k) * (g - y - k_d * y_dot);
    force(k) = y' * B(:,:,k) * (g - y - k_d * y_dot);
end
```

This shows that the computation is simple and robust, compared with traditional point-feature-based methods that often involve feature extraction, matching, and outlier removal, etc.

VII. BIO-PLAUSIBILITY OF THE COMPUTATION

The control law introduced in Section V can be considered bio-plausible because of its simple computational structure. It is a multilinear form of \( y, g, \dot{y} \), combined in a simple and feedforward way, thus can be potentially realized on a neural substrate [3]. In synthesis, the computation is a wide-field integration of local nonlinear operations. In [14], we showed that part of the computation is equivalent to a wide-field integration of elementary motion detectors, which is an accepted model for capturing many aspects of the visual computation in the fruit fly brain [21]. Moreover, generally, the visual inputs \( y \) and \( g \) do not need be the raw luminescence as observed directly. As an example, it has been widely accepted that flies perform contrast normalization on the perceived luminescence during its early stage of visual processing [22]. Such operation is considered bio-plausible as long as they still return some form of “dense” visual input. To corroborate the bio-plausibility, we will show simulation results of the control law for flight stabilization of a fruit fly.

We have used fsee [23] to simulate the visual input \( y \) of the fruit fly Drosophila. It takes a 3D CAD world model (Fig. 4a) and renders the visual perception as per the compound-eye optics of the fruit fly. The simulation incorporates the spatial disposition of the 1,398 “pixels” (ommatidia) that form the compound eye, spatial blurring, and temporal filtering of the underlying photoreceptors, all based on current knowledge in visual physiology. Fig. 4b shows a typical simulation of the visual stimulus in Mercator projection. We have ignored the detailed dynamics (wing, multi-body, etc.) and made a simplified assumption that the dynamics of the fruit fly can be treated as a rigid-body. During simulation, all the physical quantities are scaled based on experimental data from fruit flies: for example, the typical translational velocity of a fruit fly is around 0.2 m/s.

A typical convergent example using the proportional-derivative control law (13) is shown in Fig. 4c (rotation component) and Fig. 4d (translation component). The simulation starts at a joint rotational error of 30 deg and translational error of 3 m. Eventually, the rotation error settles down to about 1 deg, which is reasonable given the spatial discretization of the sensor: each “pixel” spans about 5 steradians. This nonzero rotation error induces a corresponding small nonzero translation error. The nonzero, in fact oscillating, final velocities are probably due to numerical discretization in the simulation. Since the control law is only guaranteed to be locally stable, we also used the Monte Carlo method to perform numerical tests of the region of attraction by sampling from a large...
number of initial conditions. The percentage of convergent cases is shown in Fig. 4e for increasing, randomly sampled initial (joint) rotation and translation error. Despite the fact that the environment has few distinctive features, the control law provides a reasonable region of attraction, since it does not rely on feature extraction.

Figure 5. (a) Helicopter used in the experiments, custom-built from E-flite Blade CX2. One wireless camera in encased in the tail boom. (b) Illustration of the control inputs: throttle \( u_{\text{thr}} \), rudder \( u_{\text{rud}} \), aileron \( u_{\text{ail}} \), and elevation \( u_{\text{ele}} \).

(a) Without decay factor \( \gamma \)  (b) With decay factor \( \gamma \)

(a) The goal image (before and after) (b) Image at time \( t \approx 26 \) s (before and after processing).

(c) Evolution in time of the cost function \( J \). The blue line indicates the time when the image in (b) was taken.

To demonstrate the engineering value of the bilinear control law, we have also implemented the control law on an indoor helicopter testbed. The helicopter is customized from the E-flite Blade CX2 coaxial indoor helicopter, by adding wireless camera(s) onboard as the vision sensor (Fig. 5). Due to limited payload, the helicopter is only able to carry 1-2 cameras, which do not give an omnidirectional view of the environment. All the control commands computed by the PC are sent to the radio controller via its trainer port after being converted to PPM format by a customized microcontroller. The wireless camera placed on the helicopter is able to provide 640×480 images at a maximum frame rate of 30 fps. Due to computational constraints, we choose to subsample the relayed image at 220×166. To compensate for lighting variations, the images are passed through a Sobel edge-detection filter followed by thresholding and Gaussian smoothing \( (\sigma = 15 \) px) before being used to compute the control law. The control laws are computed using the discrete models \( (14) \). The majority of the computation is performed on an Intel Core2 Duo 2.40 GHz machine. The total delay of the entire control procedure is estimated to be 78 ms.

The helicopter has 4 control inputs: throttle, rudder, aileron, and elevation. Since the pitch of the rotor blades is fixed, the throttle command \( u_{\text{thr}} \) controls the thrust by changing the speeds of the upper and lower rotors collectively. Yaw control is realized by tuning the differential speed between the two counter-rotating rotors via the rudder command \( u_{\text{rud}} \). Lateral movements are controlled by the aileron and elevation commands \( u_{\text{ail}} \) and \( u_{\text{ele}} \), which determine the cyclic pitch of the lower rotor blades by driving a 2-DOF swashplate. One limitation of our helicopter system is that it is underactuated, because it has 6 DOFs with only 4 control inputs. This is evident in that the helicopter must, for example, pitch forward in order to initiate forward flight. In principle, a different model rather than (2) needs to be adopted to take into account the underactuated dynamics. However, when the helicopter is near hover, where the uncontrolled pitching and rolling are small, the helicopter dynamics can be well approximated by a simplified fully actuated model with 4 control inputs. By using Euler angles \( (\text{pitch } \theta, \text{roll } \psi, \text{and yaw } \phi) \) as the local coordinates for rotation near \( r_d = Id \), the linearized simplified model becomes:

\[
\mathbf{I} \ddot{\mathbf{\phi}} = \mathbf{u}_{\text{rud}} + \mathbf{b}_{\text{rud}},
\]

\[
\mathbf{m} \ddot{\mathbf{p}} = \begin{bmatrix}
    u_{\text{ail}} \\
    u_{\text{ele}} \\
    u_{\text{thr}}
\end{bmatrix} + \begin{bmatrix}
    0 \\
    0 \\
    b_{\text{thr}} + f_{\text{gravity}}
\end{bmatrix}.
\]

We have also included the hardware trim/bias in throttle and rudder, namely \( b_{\text{thr}} \) and \( b_{\text{rud}} \). In practice, these can often depend on various unknown factors such the remaining battery capacity, battery placement, and motor temperature. Moreover, the change in throttle trim will also affect other channels (rudder, aileron, elevation) because the lateral thrust is provided via the main rotors.
Compared to simulations, the testbed now introduces several technical challenges due to various non-idealities: 1) limited field of view, 2) underactuated dynamics, 3) delay in the control loop, 4) non-Gaussian sensor noise due to wireless interference, and 5) time-varying hardware uncertainties.

**Automatic bias compensation:** The first problem in operating such a helicopter is compensating the biases that make the helicopter drift away when one applies nominal inputs. Bias compensation is usually a long trial-and-error manual operation that must be often repeated. However, we discovered that the bilinear velocity estimates $\dot{\omega}_{BL,z}$ defined in (12) were so reliable that they provided an opportunity to perform simple and effective automatic bias compensation. For example, if $\dot{\omega}_{BL,z} > 0$, we know, even if the magnitude is unknown, that the helicopter is rotating to the left, and we can compensate by decreasing the rudder command. In formulas, we set

$$u_{thr}(t) \propto \int_0^t \eta(\tau) \dot{\omega}_{BL,z}(\tau) \, d\tau, \quad u_{rudder}(t) \propto \int_0^t \eta(\tau) \dot{\omega}_{BL,z}(\tau) \, d\tau. \quad (13)$$

In these expressions, $\eta(\tau) = e^{-\gamma \tau}$ is a decay factor used to suppress undesirable oscillations due to noise. As can be seen from Fig. 6b, the oscillations in rudder compensation (shown in Fig. 6a) disappear after a decay factor is used.

**Visual pose stabilization:** An attached video shows a demo of visual pose stabilization. Currently, we keep the throttle manually controlled for safety purpose, but leave the remaining three channels automatically controlled. Fig. 7a and 7b show the goal image and a typical current image; both the unprocessed one as seen from the camera and the processed one after Sobel edge detection and Gaussian smoothing are shown. Because we do not know the ground truth of the current pose, the only useful indicator is the cost function $J$ defined in (6), whose change over time is plotted in Fig. 7c. The oscillations in $J$ and the fact that $J$ is close to 0 during the first $\sim 30$ s of the trial indicate that the helicopter is being stabilized around the goal.

**IX. CONCLUSIONS**

We showed that the task of visual servoing can be solved (at least locally) through purely visual control laws. The non-obvious result is that, a locally stable control law can be obtained without knowing the distance to the objects, given that certain condition on the environment holds. These control laws respect the constraints of bio-plausible computation by using a bilinear/quadratic operation on the raw luminance: in a traditional implementation, the operation is only a tensor-vector multiplication with the raw pixel array. We are in the process of extending our work in two directions. From the engineering viewpoint, we want to ascertain whether the approach can give solid advantages over traditional methods (feature-based and not). This would imply looking at issues such as change of lighting condition, occlusions, locality, nonholonomic constraints, which we did not consider yet. At a more abstract level, we want to investigate whether the same bio-plausible design concept is applicable for more complex tasks, such as structure from motion and obstacle avoidance.

**Acknowledgements.** This work is supported in part by the US Army Institute for Collaborative Biotechnology (ICB) and the Boeing Corporation.

**REFERENCES**


