Visual Servo
...through the Pages of the Transactions on Robotics
(... and Automation)

Seth Hutchinson — University of Illinois
Visual Servo Control — The Basic Idea

The aim of vision-based control schemes is to minimize an error $e(t)$ which is typically defined by

$$e(t) = s(m(t), a) - s^*$$

- The vector $m(t)$ is a set of image measurements (e.g., the image coordinates of interest points, or the parameters of a set of image segments).
- The image measurements are used to compute a vector of $k$ visual features, $s(m(t), a)$.
- The vector $a$ is a set of parameters that represent potential additional knowledge about the system (e.g., coarse camera intrinsic parameters or 3D model of objects).
- The vector $s^*$ contains the desired values of the features.

Typically, one merely writes: $e(t) = s(t) - s^*$
An Example

\[ s(t) = \text{coordinates of image points} \]
The Basic Problem

There are numerous considerations when designing a visual servo system, but the basic, prototypical problem includes the following basic assumptions:

• Eye-in-hand systems — the camera is mounted on the end effector of a robot and treated as a free-flying object with configuration space $\mathcal{Q} = SE(3)$.

• Static (i.e., motionless) targets.

• Purely kinematic systems — we do not consider the dynamics of camera motion, but assume that the camera can execute accurately the applied velocity control.

• Perspective projection — the imaging geometry can be modelled as a pinhole camera.

Some or all of these may be relaxed as one progresses to more advanced topics.
Given \( s \), control design can be quite simple.

A typical approach is to design a velocity controller, which requires the relationship between the time variation of \( s \) and the camera velocity.

- Let the spatial velocity of the camera be denoted by \( \xi = (v, \omega) \),
  - \( v \) is the instantaneous linear velocity of the origin of the camera frame and
  - \( \omega \) is the instantaneous angular velocity of the camera frame.
- The relationship between \( \dot{s} \) and \( \xi \) is given by
  \[
  \dot{s} = L_s \xi
  \]
  in which \( L_s \in \mathbb{R}^{k \times 6} \) has been called by many names, including feature sensitivity matrix, interaction matrix, and image Jacobian.

The key to visual servo — choosing \( s \) and the control law.
For Image-Based Visual Servo (IBVS)

- Features $s(t)$ are extracted from computer vision data.
- Camera pose is not explicitly computed.
- The error is defined in the image feature space, $e(t) = s(t) - s^*$.
- The control signal $\xi = (V, \Omega)$ is again a camera body velocity specified w.r.t. the camera frame, but for IBVS it is computed directly using $s(t)$.

For example, if the feature is a single image point with image plane coordinates $u$ and $v$, we have $s(t) = (u(t), v(t))$. 
A First Visual Servo System... in simulation


\[ \delta f_{\text{ref}} = J_{\text{feat}} \delta X_{\text{rel}} \]

- Simulation studies (maybe not the first visual servo system)
- Used MRAC, thus avoiding explicit computation (and derivation) of the feature sensitivity matrix, \( J_{\text{feat}} \)
- Observed that the ideal feature sensitivity matrix would be constant, diagonal matrix
- Introduced notion of Image-Based vs. Position-Based Visual Servo
A First Visual Servo System


\[ \dot{f} = f J_c(c_x) c_x \dot{x} \quad \text{and} \quad f J_c(c_x) = \frac{\partial I}{\partial c_x} \]

- Actual experiments using PUMA robot arm (binary planar patterns)
- Loads of special-purpose vision hardware
- Trajectory planning (i.e., set points) in the image
- Controlled position and orientation about optical axis (four dof)
- Closed form for \( f J_c(c_x) \)
What about the control law...

Control design relies on the relationship $\dot{e} = L_e \xi$, which relates the change in error the commanded camera velocity.

Using

$$e(t) = s(t) - s^* \quad and \quad \dot{s} = L_s \xi$$

we can easily obtain the relationship between the camera velocity and the rate of change of the error, $\dot{e} = L_e \xi$ as

$$\dot{e}(t) = \dot{s}(t) = L_s \xi$$

assuming that $s^*$ is constant.

In this case, we have $L_e = L_s \rightarrow$ the relationship between $\xi$ and $\dot{s}$ is the same as between $\xi$ and $\dot{e}$.

Now our problem is merely to find the control input $\xi = u(t)$ that gives the desired error performance.
Designing the Control Law (cont)

• In many cases, we would like to ensure an exponential decoupled decrease of the error

\[ \dot{e} = -\lambda e \rightarrow L_s \xi = -\lambda e \]

• To this end, we may choose the control law as

\[ u(t) = \xi = -\lambda L_e^+ e \] (1)

where \( L_e^+ \in \mathbb{R}^{6 \times k} \) is chosen as the Moore-Penrose pseudo-inverse of \( L_e \),

\[ L_e^+ = (L_e^T L_e)^{-1} L_e^T \]

when \( L_e \) is of full rank 6.

• This is a *left* pseudoinverse, since

\[ L_e^+ L_e = \left[ (L_e^T L_e)^{-1} L_e^T \right] L_e = I \]
Designing the Control Law (cont)

- The choice of $\xi = -\lambda L_e^+ e$ is the usual least squares solution, and gives
  \[
  \dot{e} = L_e \xi = -\lambda L_e^+ L_e e
  \]

- This is also the solution that locally minimizes $||\xi||$.

- When $k = 6$, if $\det L_e \neq 0$ it is possible to invert $L_e$. In this case we may use the control
  \[
  \xi = -\lambda L_e^{-1} e
  \]
  which gives exactly the desired behavior of $\dot{e} = -\lambda e$. 
Feddema returns - with two more dof’s


- Real experiments (same set-up as before)
- Full six-dof control
- Derivation of now-famous interaction matrix for points
- Added bonus of feature selection (using, e.g., condition number of $J$)

Nearly every paper written on the subject of visual servo control includes at least a tip of the hat to this derivation (if not a re-derivation, or restatement of the result).
Consider a point $P$ with coordinates $(x, y, z)$ w.r.t. the camera frame. Using perspective projection, $P$’s image plane coordinates are given by

$$u = \lambda \frac{x}{z}, \quad v = \lambda \frac{y}{z}$$

in which $\lambda$ is the camera focal length.
The Interaction Matrix (for a point feature)

As an example, consider the interaction matrix for a single point with coordinates $x, y, z$.

To determine the interaction matrix for a point,

1. Compute the time derivatives for $u$ and $v$.
2. Express these in terms of $u, v, \dot{x}, \dot{y},$ and $\dot{z}$ and $z$.
3. Find expressions for $\dot{x}, \dot{y},$ and $\dot{z}$ in terms of $\xi$ and $x, y, z$.
4. Combine equations and grind through the algebra.
The Interaction Matrix (for a point feature)

Step 1:
Using the quotient rule

\[
\dot{u} = \lambda \frac{z \dot{x} - x \dot{z}}{z^2}, \quad \dot{v} = \lambda \frac{z \dot{y} - y \dot{z}}{z^2}
\]

Step 2:
The perspective projection equations can be rewritten to give expressions for \( x \) and \( y \) as

\[
x = \frac{uz}{\lambda}, \quad y = \frac{vz}{\lambda}
\]

Substitute these into the equations above for \( \dot{u} \) and \( \dot{v} \).

\[
\dot{u} = \lambda \frac{\dot{x}}{z} - \frac{uz \dot{z}}{z^2}, \quad \dot{v} = \lambda \frac{\dot{y}}{z} - \frac{vz \dot{z}}{z^2}
\]
Step 3:
The velocity of (the fixed point) $P$ relative to the camera frame is given by

$$\dot{P} = -\Omega \times P - V$$

which gives equations for each of $\dot{x}$, $\dot{y}$, and $\dot{z}$.

Expanding $\dot{P} = -\Omega \times P - V$ we obtain

$$\dot{x} = -v_x - \omega_y z + \omega_z y$$

$$\dot{y} = -v_y - \omega_z x + \omega_x z$$

$$\dot{z} = -v_z - \omega_x y + \omega_y x$$

Now it’s just algebra...
Step 4:

Combining equations we obtain

\[
\begin{align*}
\dot{u} &= -\frac{\lambda}{z} v_x + \frac{u}{z} v_z + \frac{uv}{\lambda} \omega_x - \frac{(\lambda^2 + u^2)}{\lambda} \omega_y + v \omega_z \\
\dot{v} &= -\frac{\lambda}{z} v_y + \frac{v}{z} v_z + \frac{(\lambda^2 + v^2)}{\lambda} \omega_x - \frac{uv}{\lambda} \omega_y - u \omega_z
\end{align*}
\]

These equations can be nicely written in matrix form.
In matrix form we obtain

\[
\begin{pmatrix}
\dot{u} \\
\dot{v}
\end{pmatrix}
= \begin{bmatrix}
-\frac{\lambda}{z} & 0 & \frac{u}{z} & \frac{uv}{\lambda} & -\frac{\lambda^2 + u^2}{\lambda} & v \\
0 & -\frac{\lambda}{z} & \frac{v}{z} & \frac{\lambda^2 + v^2}{\lambda} & -\frac{uv}{\lambda} & -u
\end{bmatrix} \xi
\]

which can be written more compactly as

\[
\dot{s} = L(s, z)\xi.
\]

Feddema, et al. simply called this the \textit{Jacobian}.

It has since been called the \textit{Interaction Matrix} [Espiau, et al., 1992], or the \textit{image Jacobian} [Hutchinson, et al., 1996].
The Null Space of the Interaction Matrix

The null space of this image Jacobian matrix is spanned by

\[
\begin{bmatrix}
  u \\
v \\
\lambda \\
0 \\
0 \\
\lambda
\end{bmatrix}
\begin{bmatrix}
  0 \\
  0 \\
  0 \\
u \\
v \\
\lambda
\end{bmatrix}
\begin{bmatrix}
  uvz \\
-u^2 + \lambda^2 z \\
\lambda vz \\
-\lambda^2 \\
0 \\
u\lambda
\end{bmatrix}
\begin{bmatrix}
  \lambda(u^2 + v^2 + \lambda^2)z \\
  0 \\
-u(u^2 + v^2 + \lambda^2)z \\
  uv\lambda \\
-(u^2 + \lambda^2)z \\
u\lambda^2
\end{bmatrix}
\]
Intuitively, this basis of the null space corresponds to

- translation along a projection ray
- rotation about a projection ray
- rotation about the camera y-axis, keeping the camera pointed in the right direction using the linear motion
- translation in the camera y-direction, keeping the camera pointed in the right direction using the rotational motion

These are point motions that cannot be “seen” by the camera.

*It seems like one should be able to exploit these “invisible” degrees of freedom...*
A New Approach...


- A rigorous mathematical framework for visual servo control (*c’est toujours comme ça, chez les françaises*)
- Task function approach for incorporating multiple tasks (essentially projection of secondary task onto the null space of the primary task’s interaction matrix)
- Derivation of interaction matrices for several primitives (including lines, circles, spheres, etc.)
- Real experiments with 6-dof robot
- Birth of “the French school” of visual servo contro
The few other pre-1996 VS papers

Some Highlights

• A very cool ping-pong playing robot - Andersson, 1989... even though it might not really be visual servo

• Vision-based control of a nonholonomic mobile robot - Skaar, et al., 1992

• Application of state-space control methods to derive image-plane control laws (optimal, adaptive) - Papanikolopoulos, et al., 1993, 1995

• A partitioned position-vision controller and a path planner for the system - Castano, Fox, Hutchinson, 1994, 1995

• Uncalibrated visual servo by exploiting rotational invariance - Yoshimi, Allen, 1995
A Special Issue (well... section...) - Vol.12, no. 5, 1996

- Corke, P.I.; Good, M.C. "Dynamic effects in visual closed-loop systems," pp.671-683
- Wilson, W.J.; Williams Hulls, C.C.; Bell, G.S. "Relative end-effector control using Cartesian position based visual servoing," pp.684-696
- Nelson, B.J.; Khosla, P.K. "Force and vision resolvability for assimilating disparate sensory feedback," pp.714-731
- Grosso, E.; Metta, G.; Oddera, A.; Sandini, G. "Robust visual servoing in 3-D reaching tasks," pp.732-742
- Kelly, R. "Robust asymptotically stable visual servoing of planar robots," pp.759-766
Wilson, W.J.; Williams Hulls, C.C.; Bell, G.S. "Relative end-effector control using Cartesian position based visual servoing," *T-RA*, vol.12, no.5, pp.684-696, Oct 1996

- Computer vision data are used to compute the pose of the camera $d, R$ relative to the world frame.
- The error $e(t)$ is defined in the pose space, $d \in \mathbb{R}^3, R \in SO(3)$.
- The control signal $\xi = (v, \omega)$ is a camera body velocity.

If the goal pose is given by $d = 0, R = I$, the role of the computer vision system is to provide, in real time, a measurement of the pose error.
Visual servo control is now a fairly popular research area within the robotics community. Topics of interest include (at least) the following:

- the search for better features
- switched control systems
- the use of novel imaging geometries
- consideration of dynamics
- investigation of novel sensors

All of these topics can be found in the pages of the Transactions.