


WS2: Workshop on Visual Servoing

IROS 2002 – EPFL – Switzerland

October 1st, 2002

Organizing Committee

Philippe Martinet Philippe.Martinet@lasmea.univ-bpclermont.fr
LASMEA, Blaise Pascal University, France

Enric Cervera ecervera@icc.uji.es
Robotic Intelligence Lab, Jaume-I University, Spain

Nicolas Andreff Nicolas.Andreff@ifma.fr
LaRAMA, Blaise Pascal University, France

Francois Chaumette chaumette@irisa.fr
IRISA, INRIA Rennes, France

Contents

Preface

Visual servoing is a mature robot control technique using vision in feedback control loops. Though the first systems date back to the late 1970s and early 1980s, it is not until the mid 1990s that there is a sharp increase in publications and working systems, due to the availability of fast and affordable vision processing systems.

Since the last workshops, tutorials, and invited sessions to focus on this topic in the most important robotics conferences throughout the world (ICRA 94, 96, 98; IROS'97; AAAI'00; ECCV'00; CIRA'01), significant advances have been achieved, driven by improvements of technology (sensor and computation) and the underlying theories of image segmentation, geometry, motion and control.

This workshop presents advanced theoretical and implementation issues in this field, with a selection of papers by the most prominent researchers in the community. Innovative control algorithms based on virtual fixtures, insect strategies, the human brain, or hybrid dynamical systems, as well as robust vision processing techniques are extensively discussed. Simulations and real experiments illustrate the presented approaches.

The article by Greg Hager outlines a theory of compliant virtual fixtures, and presents its application to vision-guided assistance, within the framework of human-machine cooperative manipulation. Visual servoing algorithms are converted into control algorithms that provide such visual fixtures. A preliminary implementation on the JHU Steady Hand Robot system is shown, in the domain of minimally invasive microsurgical tasks.

Though manipulator applications have historically lead the field, visual servoing for mobile robots is finding a niche with innovative developments. A visual homing control method for nonholonomic vehicles is presented in the article by Usher, Corke and Ridley. The method stresses the similarities between insect-based strategies and visual servoing. Experimental results using an outdoor mobile platform equipped with an omnidirectional camera illustrate the approach.

The lack of truly operational systems is characteristic in visual servoing. The need for a robust, flexible vision system hinders its application in domestic environments. This problem is discussed in the article by Kragic and Christensen, focusing on achieving robustness for manipulating a variety of objects in a living room setting. A different approach to increase stability is presented in the paper by Hutchinson and Gans. It relies on the use of hybrid dynamical systems, consisting of a set of continuous subsystems and a switching rule. A high-level decision maker selects from two low-level visual servo controllers, with promising simulation and experimental results.

New control architectures aim to solve the limitations of classical visual servoing approaches. The article by Hashimoto, Namiki and Ishikawa presents one such architecture based on the human brain motor control: it is a hierarchical parallel processing architecture, based on the hierarchical efferent / afferent interaction model for human visuo-motor control system. The task domain involves a dynamically moving object, grasped and handled by a manipulator equipped with a dexterous hand and a high speed vision system.

Finally, the article by Barreto, Martin and Horaud explores the benefits of enhancing

the field of view with central catadioptric cameras, resulting in a new visual robot control concept where tracking and control are embedded together. The catadioptric Jacobian matrix is studied, showing that no additional singularities are introduced with respect to the traditional pinhole camera model.

We would like to thank the authors for producing such high-quality contributions in quite a short time. They deserve the credits for this successful event and our gratitude for eagerly contributing to disseminate their most advanced research work.

Special thanks to Profs. Rüdiger Dillmann and Dario Floreano, IROS 2002 Workshop chairs, for their helpful support. And finally, we would like to thank IEEE, RSJ and EPFL for hosting this workshop in the framework of the IROS conference.

Nicolas Andreff
Enric Cervera
Franois Chaumette
Philippe Martinet
Lausanne, Switzerland
September 2002

Further information

- This workshop is organized in the area of EURON (`http://www.euron.org`). Since the beginning of 2002, there exists a European Scientific Interest Group on Visual Servoing (`http://www.robot.uji.es/EURON/visualservoing/`).
- In September 2002, a Summer School on Visual Servoing was organized in Benicàssim, Spain (`http://www.robot.uji.es/EURON/visualservoing/summerschool/`).

Human-Machine Cooperative Manipulation With Vision-Based Motion Constraints

Gregory D. Hager

Department of Computer Science
Johns Hopkins University
Baltimore, MD 21218

Abstract

This paper discusses a class of control algorithms that provide enhanced physical dexterity by imposing passive motion constraints. Such motion constraints are often referred to as virtual fixtures. It is shown that algorithms originally designed for vision-based control of manipulators can be easily converted into control algorithms that provide virtual fixtures. As a result it is possible to create advanced human-machine cooperative manipulation systems that take complete advantage of information provided by vision, yet permit the user to retain control of essential aspects of a given task.

1 Introduction

Much of "classical" robotics has focused on creating machines that are autonomous. However, such endeavors are fundamentally limited by our ability to create machines that can perceive, judge, and react to unforeseen (or sometimes foreseen!) circumstances in the world. To this day, there are still few situations, other than rote, open-loop manipulation, where robotics have even begun to compete with humans, in this regard.

At the other extreme, teleoperation tends to focus on providing a high-performance, high-fidelity operator interface to a machine. In many ways, this is a natural marriage, as the human now controls a machine that may be more accurate, reliable, or powerful than the human operator. However, while the machine is clearly amplifying some level of human skill, it does so at the cost of attenuating others. In particular, teleoperation systems do not have an underlying representation of the *intent* of the operator. As a result, it is not possible to adapt or modify the interface to enhance those skills that are most germain to the task at hand.

Our group in the Center for Computer Integrated Surgical Systems (CISST) has been working to create surgical systems that improve both the speed and precision of medical interventions. Our goal is to create mechanisms that are neither autonomous, nor purely passive. Rather, our intent is to create mechanisms that selectively provide cooperative assistance to a surgeon, while allowing the surgeon to retain ultimate control of the procedure.

In our recent work, we have focused on developing assistance methods for microsurgery. Here, the extreme challenges of physical scale accentuate the need for dexterity enhancement, but the unstructured nature of the tasks dictates that the human be directly "in the loop." For example, retinal vein cannulation [25] involves the insertion of a needle of approx. 20-50 microns in diameter into the lumen of a retinal vein (typically 100 microns in diameter or less)[1]. At these scales, tactile feedback is practically non-existent, and depth perception is limited to what can be seen through a stereo surgical microscope. In short, such a procedure is at the limit of what is humanly possible in conventional surgical practice.

Given the scale of operation, the most obvious need is to increase the precision of human motion, ideally without slowing or limiting the surgeon. In recent work [16, 17, 15], we have begun to develop assistant methods that are based on manipulating the apparent compliance of tools simultaneously held by both the surgeon and a robot. Intuitively, if a tool is extremely stiff, then it is easier to achieve high precision of motion, and to remove tremor. Conversely, a low stiffness makes it possible to perform large-scale "transport" motions.

Although they increase absolute precision, purely isotropic compliances cannot take advantage of natural task constraints to provide structured assistance. For example, when placing a needle into the lumen of a blood vessel, the natural mode of assistance would be to stabilize the needle in the lateral directions, but

[1] As a point of reference, a human hair is typically on the order of 80 microns in diameter.

permit relatively free, quasi-static positioning along the needle axis.

In this paper, we specifically focus on the use of *anisotropic* compliances as a means of assistance. In previous work we have related these anisotropic compliances to the notion of virtual fixtures [18, 22]. Virtual fixtures, like the real thing, provide a surface that confines and/or guides motion. We initially describe how virtual fixtures can be produced as a generalization of previous work in [1, 2, 20], and then turn to the problem of deriving vision-based virtual fixtures. Through example, we show how visual servoing algorithms for one camera [3] and two camera [7, 8] systems can be translated into virtual fixtures. As a result, much of the previous literature on visual servoing can be applied to the problem of human-machine cooperative manipulation.

2 Virtual Fixtures

Our work has been motivated by the JHU Steady Hand Robot, and, in particular, the assistance paradigm of direct manipulation it was designed for [15, 16, 24]. Briefly, the JHU Steady Hand robot is a 7 DOF robot equipped with a force sensing handle at the endpoint. Tools are mounted at the endpoint, and "manipulated" by an operator holding the force handle. The robot responds to the applied force, thus implementing a means of direct control for the operator. The robot has been designed to provide micron-scale accuracy, and to be ergonomically appropriate for minimally invasive microsurgical tasks [24].

In this section, we introduce the basic admittance control model used for the Steady Hand Robot, extend this control to anisotropic compliances, and finally relate anisotropic compliances to an underlying task geometry.

In the remainder of this paper, transpose is denoted by $'$, scalars are written lowercase in normal face; vectors are lowercase and boldface; and matrices are normal face uppercase.

2.1 Virtual Fixtures as a Control Law

In what follows, we model the robot as a purely kinematic Cartesian device with tool tip position $\mathbf{x} \in SE(3)$ and a control input that is endpoint velocity $\mathbf{v} = \dot{\mathbf{x}} \in \Re^6$, all expressed in the robot base frame. The robot is guided by applying forces and torques $\mathbf{f} \in \Re^6$ on the manipulator handle, likewise expressed in robot base coordinates.

In the Steady-Hand paradigm, the relationship between velocity and motion is derived by considering a "virtual contact" between the robot tool tip and the environment. In most cases, this contact is modeled by a linear viscous friction law

$$k\mathbf{v} = \mathbf{f} \tag{1}$$

or equivalently

$$\mathbf{v} = \frac{1}{k}\mathbf{f} \tag{2}$$

where $k > 0$ controls the stiffness of the contact. In what follows, it will be more convenient to talk in terms of a compliance $c \equiv 1/k$.

When using (2), the effect is that the manipulator is equally compliant in all directions. Suppose we now replace the single constant c with a diagonal matrix C. Making use of C in (2) gives us the freedom to change the compliance of the manipulator in the coordinate directions. For example, setting all but the first two diagonal entries to zero would create a system that permitted motion only in the x-y plane. It is this type of anisotropic compliance that we term a virtual fixture. In the case above, the fixture is "hard, " meaning it permits motion in a subspace of the workspace. If we instead set the first two entries to a large value, and the remaining entries to a small one, the fixture becomes "soft." Now, motion in all directions is allowed, but some directions are easier to move in than others. We refer to the motions with high compliance as *preferred* directions, and the remaining directions as *non-preferred* directions.

2.2 Virtual Fixtures as Geometric Constraints

While it is clearly possible to continue to extend the notion of virtual fixture purely in terms of compliances, we instead prefer to take a more geometric approach, as suggested in [1, 2]. We will develop this geometry by specifically identifying the preferred and non-preferred directions of motion at a given time point t. To this end, let us assume that we are given a $6 \times n$ time-varying matrix $D = D(t)$, $0 < n < 6$. Intuitively, D represents the instantaneous preferred directions of motion. For example, if n is 1, the preferred direction is along a curve in SE(3); if n is 2 the preferred directions span a surface; and so forth.

From D, we define two projection operators, the span and the kernel of the column space, as

$$\text{Span}(D) \equiv [D] = D(D'D)^{-1}D' \tag{3}$$

$$\text{Ker}(D) \equiv \langle D \rangle = I - [D] \tag{4}$$

This formulation assumes that D has full column rank. It will occasionally be useful to deal with cases where the rank of D is lower than the number of columns (in particular, the case when $D = 0$). For this reason, we will assume $[\cdot]$ has been implemented using the pseudo-inverse [23, pp. 142–144] and write

$$\text{Span}(D) \equiv [D] = D(D'D)^{+}D' \tag{5}$$

$$\text{Ker}(D) \quad \equiv \quad \langle D \rangle = I - [D] \tag{6}$$

The following properties hold for these operators [23]:

1. symmetry: $[D] = [D]'$
2. idempotence: $[D] = [D][D]$
3. scale invariance: $[D] = [kD]$
4. orthogonality: $\langle D \rangle' [D] = 0$
5. completeness: $\text{rank}(\alpha \langle D \rangle + \beta [D]) = n$ where D is $n \times m$ and $\alpha, \beta \neq 0$
6. equivalence of projection: $[\langle D \rangle f] f = \langle D \rangle f$

The above statements remain true if we exchange $\langle D \rangle$ and $[D]$. Finally, it is useful to note the following equivalences:

- $[[D]] = [D]$
- $\langle \langle D \rangle \rangle = [D]$
- $[\langle D \rangle] = \langle [D] \rangle = \langle D \rangle$

Returning to our original problem, consider now decomposing the input force vector, $\mathbf{f}$, into two components

$$\mathbf{f}_D \equiv [D]\mathbf{f} \quad \text{and} \quad \mathbf{f}_\tau \equiv \mathbf{f} - \mathbf{f}_D = \langle D \rangle \mathbf{f} \tag{7}$$

It follows directly from property 4 that $\mathbf{f}_D \cdot \mathbf{f}_\tau = 0$ and from property 5 that $\mathbf{f}_D + \mathbf{f}_\tau = \mathbf{f}$ Combining (7) and (2), we can now write

$$\mathbf{v} = c\mathbf{f} = c(\mathbf{f}_D + \mathbf{f}_\tau) \tag{8}$$

Let us now introduce a new compliance $c_\tau \in [0, 1]$ that attenuates the non-preferred component of the force input. With this we arrive at

$$\begin{aligned} \mathbf{v} &= c(\mathbf{f}_D + c_\tau \mathbf{f}_\tau) \\ &= c([D] + c_\tau \langle D \rangle)\mathbf{f} \end{aligned} \tag{9}$$

Thus, the final control law is in the general form of an admittance control with a time-varying gain matrix determined by $D(t)$. By choosing c, we control the overall compliance of the system. Choosing c_τ low imposes the additional constraint that the robot is stiffer in the non-preferred directions of motion. As noted above, we refer to the case of $c_\tau = 0$ as a *hard virtual fixture*, since it is not possible to move in any direction other than the preferred direction. All other cases will be referred to as *soft virtual fixtures*. In the case $c_\tau = 1$, we have an isotropic compliance as before.

It is also possible to choose $c_\tau > 1$ and create a virtual fixture where it is easier to move in non-preferred directions than preferred. In this case, the natural approach would be to switch the role of the preferred and non-preferred directions.

2.3 Choosing the Preferred Direction

The development to this point directly supports the following types of guidance:

- Motion in a subspace: suppose we are supplied with a time-varying, continuous function $D = D(t)$. Then applying (9) yields a motion constraint within that subspace.
- Motion to a target pose $\mathbf{x}_t \in SE(3)$: Suppose that we have a control law $\mathbf{u} = f(\mathbf{x}, \mathbf{x}_t)$ such that by setting $\mathbf{v} = \mathbf{u}$,

  $$\lim_{t \to \infty} \mathbf{x} = \mathbf{x}_t.$$

  Then by choosing $D = \mathbf{u}$ and applying (9), we create a virtual fixture that guides the user to the given target pose.

These two tasks are, in some sense, at the extremes of guidance. In one case, there is no specific objective to attain; we are merely constraining motion. In the second, the pose of the manipulator is completely constrained by the objective. What of tasks that fall between these two extremes?

To study this problem, let us take a simple, yet illustrative case: the case of maintaining the tool tip within a plane through the origin. For the moment, let us neglect manipulator orientation and consider the problem when controlling just the spatial position of the endpoint. We define the surface as $P(\mathbf{p}) = \mathbf{n} \cdot \mathbf{p} = 0$ where $\mathbf{n}$ is a unit vector expressed in robot base coordinates.

Based on our previous observations, if the goal was to allow motion *parallel* to this plane, then, noting that $\mathbf{n}$ is a non-preferred direction in this case, we would define $D = \langle \mathbf{n} \rangle$ and apply (9). However, if the tool tip is not in the plane, then it is necessary to adjust the preferred direction to move the tool tip toward it. Noting that $P(\mathbf{x})$ is the (signed) distance from the plane, we define a new preferred direction as follows:

$$D_c(\mathbf{x}) = [(1 - k_d)\langle \mathbf{n} \rangle \mathbf{f} / \|\mathbf{f}\| - k_d [\mathbf{n}]\mathbf{x}] \quad 0 < k_d < 1. \tag{10}$$

The geometry of (10) is as follows. The idea is first produce the projection of the applied force onto the nominal set of preferred directions, in this case $\langle \mathbf{n} \rangle$. At the same time, the location of the tool tip is projected onto the plane normal vector. The convex

combination of the two vectors yields a resultant vector that will return the tool tip to the plane. Choosing the constant k_d governs how quickly the tool is moved toward the plane. One minor issue here is that the division by $\|\mathbf{f}\|$ is undefined when no user force is present. Anticipating the use of projection operators, we make use of a scaled version of (10) that does not suffer this problem:

$$D_c(\mathbf{x}) = (1-k_d)\langle \mathbf{n} \rangle \mathbf{f} - k_d \|\mathbf{f}\| [\mathbf{n}]\mathbf{x} \quad 0 < k_d < 1. \quad (11)$$

We now apply (9) with $D = D_c$.

Noting that the second term could also be written

$$\|\mathbf{f}\| k_d P(\mathbf{x})\mathbf{n},$$

it is easy to see that, when the tool tip lies in the plane, the second term vanishes. In this case, it is not hard to show, using the properties of the projection operators, that combining (11) with (9) results in a law equivalent to a pure subspace motion constraint. One potential disadvantage of this law is that when user applied force is zero, there is no virtual fixture as there is no defined preferred direction. Thus, there is a discontinuity at the origin. However, in practice the resolution of any force sensing device is usually well below the numerical resolution of the underlying computational hardware, so the user will never experience this discontinuity.

With this example in place, it is not hard to see its generalization to a broader set of control laws. We first note that another way of expressing this example would be to posit a control law of the form:

$$\mathbf{u} = -(\mathbf{n} \cdot \mathbf{x})\mathbf{n} = -[\mathbf{n}]\mathbf{x} \quad (12)$$

and to note that assigning $\mathbf{v} = \mathbf{u}$ would drive the manipulator into the plane. This is, of course, exactly what appears in the second term of (11). If we now generalize this idea, we can state the following informal rule.

General Virtual Fixture Rule: Given:

1. A surface $S \subseteq SE(3)$ (the motion objective)
2. A control law $\mathbf{u} = f(\mathbf{x}, S)$ where by setting $\mathbf{v} = \mathbf{u}$,
$$\lim_{t \to \infty} \mathbf{x} \in S.$$
(the control law moves the tool tip into S)
3. A rule for computing preferred directions $D = D(t)$ relative to S where $\langle D \rangle \mathbf{u} = 0$ iff $\mathbf{u} = 0$ (the motion direction is consistent with the control law)

then applying the following choice of preferred direction:

$$D_g(\mathbf{x}) = (1 - k_d)[D]\mathbf{f} - k_d \|\mathbf{f}\| \langle D \rangle \mathbf{u} \quad 0 < k_d < 1. \quad (13)$$

yields a virtual fixture that controls the robot toward S and seeks to maintain user motion within that surface.

Note that a sufficient condition for condition 3 above to be true is that, for all pairs $\mathbf{u} = \mathbf{u}(t)$ and $D = D(t)$, $[D]\mathbf{u} = 0$. This follows directly from the properties of projection operators given previously.

To provide a concrete example, consider again the problem of moving the tool tip to a plane through the origin, but let us now add the constraint that the tool z axis should be oriented along the plane normal vector. In this case, $\mathbf{n}$ is a preferred direction of motion (it encodes rotations about the z axis which we don't care about). Let $\mathbf{z}$ denote the vector pointing along the tool z axis and define a control law that is

$$\mathbf{u} = \begin{bmatrix} -(\mathbf{x} \cdot \mathbf{n})\mathbf{n} \\ \mathbf{z} \times \mathbf{n} \end{bmatrix} \quad (14)$$

It is easy to see that this law moves the robot into the plane, and also simultaneously orients the end-effector z axis to be along the normal to the plane. Now, let

$$D = D(t) = \begin{bmatrix} \langle \mathbf{n} \rangle & 0 \\ 0 & \mathbf{n} \end{bmatrix}$$

It follows that $[D]$ is a basis for translation vectors that span the plane, together with rotations about the normal to the plane. Therefore $[D]\mathbf{u} = 0$ since (14) produces translations normal to the plane, and rotations about axes that lie in the plane. Thus, the general virtual fixturing rule can be applied.

3 Vision-Based Virtual Fixtures

Now, we turn to the problem of providing assistance, where the objective defining the virtual fixture is observed by one or more cameras. To simplify the presentation, in what follows we assume that we have calibrated the camera internal parameters and can therefore work in image normalized coordinates.

3.1 Controlling the Viewer: Pure Translation

Let us start with a well-studied problem. We have a camera fixed to the endpoint of the manipulator, and the camera observes a fixed, static environment. Our goal is to control the motion of the end-effector by defining a motion for the camera itself based on information extracted from the camera image.

To keep things simple, consider first the case of pure translation ($\mathbf{v} \in \Re^3$) where the camera is aligned

with the robot base frame. In this case, the relationship between the motion of the camera and the image motion of a fixed point in space is given by the well-known image Jacobian relationship [12]:

$$\dot{\mathbf{h}} = J\mathbf{v} \tag{15}$$

where $\mathbf{h} = (u, v)' \in \Re^2$ is the image location of a feature point, and J is 2×3.

It is again well-known [3, 12] that the rows of J span the (two-dimensional) space of motions that create feature motion in the image, and therefore $\langle J' \rangle$ is the (one-dimensional) space of motions that leave the point fixed in the image. Consider, thus, creating a virtual fixture by defining

$$D = J' \tag{16}$$

in (9). From the discussion above, it should be clear that this will create a virtual fixture that prefers motion in any direction except along the viewing direction. While it would seem we are done at this point, there is one minor issue: the image Jacobian depends on the depth of the estimated point. However, if we consider the form of the Jacobian in this case, we see it can be written thus:

$$J = \begin{bmatrix} \frac{1}{z} & 0 & \frac{-u}{z} \\ 0 & \frac{1}{z} & \frac{-v}{z} \end{bmatrix} = \frac{1}{z} \begin{bmatrix} 1 & 0 & -u \\ 0 & 1 & -v \end{bmatrix} \tag{17}$$

As such, we see that the term involving z is a scale factor and, as noted earlier, our projection operators are invariant over scaling of their argument. Thus, we have our first result:

Image plane translation: If we restrict $\mathbf{v}$ to be pure translation and choose an image location $\mathbf{h} = (u, v)'$, then implementing (9) using

1. $D = J'$ creates a virtual fixture that prefers motion in the plane normal to the viewing direction defined by $\mathbf{h}$ and the camera optical center
2. $D = \langle J' \rangle$ creates a virtual fixture that prefers motion along the viewing direction defined by $\mathbf{h}$ and the camera optical center.

As a special case, choosing $u = v = 0$ yields a virtual fixture parallel to the image plane, which, as is obvious from (17), is the camera x-y plane.

It is important to note that the image plane virtual fixtures defined above can be implemented both with and without feedback. That is, if we simply choose a fixed image location (e.g. the origin), then the camera will always move in a plane orthogonal to the line of sight through the chosen location. On the other hand, if we choose to track a feature over time, then the motion will always be orthogonal to the line of sight to that feature.

The other possibility, with a single feature, is to maintain the visual cue at a specific image location. For example, suppose the goal is to center an observed point $\mathbf{h}$ in the image. Since our objective is at the origin, we can define a control law of the form

$$\mathbf{u} = -J'\mathbf{h} \tag{18}$$

where J is evaluated at the origin. It is possible to show this law will converge for any feature starting and remaining in the image [14]. Furthermore, the preferred directions of motion in this case are $\langle J' \rangle$ and so it follows that $\langle \langle J' \rangle \rangle \mathbf{u} = -[J']J'\mathbf{h} = 0$ only when $\mathbf{h} = 0$. Since $\mathbf{u} \neq 0$ when $\mathbf{h} \neq 0$, we can apply the general virtual fixturing rule.

At this point, we can state a useful specialization for the rest of this paper:

General Vision-Based Virtual Fixtures Suppose we are supplied with an error term $\mathbf{e} = \mathbf{e}(\mathbf{x})$. Let $S = \{\mathbf{x} | \mathbf{e}(\mathbf{x}) = 0\}$, let $J = \partial \mathbf{e} / \partial \mathbf{x}$, and define $\mathbf{u} = WJ'\mathbf{e}$ where W is a symmetric, positive definite matrix of appropriate dimension (e.g. $W = (J'J)^+$). Then the general virtual fixture rule can be applied with preferred directions $\langle J' \rangle$ provided $\mathbf{u}$ so computed converges to S.

There is an interesting variation on this. Suppose we choose *no* preferred direction of motion (i.e. $D = 0$.) In this case, the first term of (13) disappears and the preferred direction in (9) is simply $\mathbf{u}$. Thus, the result is a virtual fixture that moves the robot to a target position (compare with the rules at the beginning of Section 2.3) and then becomes isotropic. Note, however, that by definition $\mathbf{u}$ is always orthogonal to the line of sight, so the camera prefers to maintain a constant distance to the point during motion.

To press home these points, consider a final problem: to place a specific image location on an observed line, and to facilitate motion along the line. Following the development in [8], suppose we observe a fixed line $\mathbf{l} \in \Re^3$, where the three components of $\mathbf{l}$ can be thought of as the normal vector to the line in the image, and the distance from the origin to the line. This vector is also parallel to the normal vector to the plane formed by the optical axis and the line as it appears in the image plane. We also furnish a distinguished image location $\hat{\mathbf{h}} \in \Re^3$, expressed in homogeneous coordinates. We can then define $\mathbf{e} = \hat{\mathbf{h}} \cdot \mathbf{l}$, to be the image-plane distance between the point and the line.

First, we note that the image Jacobian (relative to $\mathbf{e}$) is now simply

$$L = \mathbf{l}' \begin{bmatrix} J \\ 0 \end{bmatrix} \in \Re^3$$

(note that the z dependence in L is once again a non-issue). As we would now expect, L represents *non-preferred* directions of motion, as it spans the space of motions that change the distance from the point to the line. As a result, choosing preferred directions as $\langle L' \rangle$ in (9) would prefer camera motion within the plane encoded by $\mathbf{l}$.

In order to actually place the designated point onto the line, we note that the control law

$$\mathbf{u} = L'\mathbf{e} \tag{19}$$

will move the feature point $\hat{\mathbf{h}}$ to the observed line [8]. Hence, we apply (13) using $D = \langle L' \rangle$ and $\mathbf{u}$ as defined above — in short, another application of the general vision-based virtual fixture rule.

3.2 Controlling the Viewer: General Case

In moving from pure translation to general robot motion, almost nothing changes from the previous section other than increases in dimensionality. In the case of full motion in $SE(3)$, the image Jacobian becomes 2×6 and has the following general form:

$$\begin{bmatrix} \frac{1}{z} & 0 & \frac{-u}{z} & \frac{-uv}{1} & \frac{1^2+u^2}{1} & -v \\ 0 & \frac{1}{z} & \frac{-v}{z} & \frac{-1^2-v^2}{1} & \frac{uv}{1} & u \end{bmatrix} \tag{20}$$

As shown in [10], the kernel of the image Jacobian given in (20), is spanned by the four vectors

$$\begin{bmatrix} u \\ v \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad \begin{bmatrix} 0 \\ 0 \\ 0 \\ u \\ v \\ 1 \end{bmatrix} \quad \begin{bmatrix} uvz \\ -(u^2+1)z \\ vz \\ -1 \\ 0 \\ u \end{bmatrix} \quad \begin{bmatrix} dz \\ 0 \\ -udz \\ uv \\ -(u^2+1)z \\ u \end{bmatrix} \tag{21}$$

where $d = (u^2 + v^2 + 1)$. As such, we can see that the kernel spans motions that include: 1) motion along the line of sight; 2) rotation about the line of sight; 3) motion on a sphere linking the point with the camera; and, linear combinations thereof. Note that 3) spans two degrees of freedom.

If we reconsider all of the cases of the previous section, we see that by using the full Jacobian, we achieve the same virtual fixtures, albeit in a larger space of allowed motions. In particular, choosing $D = \langle J \rangle$ now prefers motions on a sphere about the observed point, together with rotation about, and translation along, the line of sight. It is, however, important to note that the distance from the camera to the observed point no longer "drops out" of the system as in the case of pure translation. Therefore, distance must be estimated, for example by using adaptive schemes as outlined in [21].

At this point, we redirect the reader to [3], where it is observed that regulating the motion of the camera through invariants defined on observed features or measures thereof creates a broad family of "virtual linkages" between the camera and the world. In effect applying the constructions as laid out above to these control laws, creates a corresponding family of virtual fixtures. Likewise, more recent "hybrid" approaches to control that seek to produce more reliable converge of visual servoing methods [19] and/or control other properties of the motion of features in the image plane [4, 5] can be applied, to the extent that the properties outlined in Section 2.3 are satisfied.

3.3 More General Camera Configurations

Until now, we have only considered a single end-effector mounted camera. However, it is important to note that everything we have said above can be applied to the case of a fixed camera observing an independently moving manipulator, with suitable adjustment of the form of the image Jacobian. Furthermore, we note that, for either configuration, observing both the end-effector and the external features defining the task creates an endpoint closed-loop control law which has well-known robustness against camera calibration error [7, 8, 11, 12]. Likewise, methods for estimating the image Jacobian online [13] can, in principle, be applied practically without change.

As a final generalization, we could also add a second observing camera. It is well known [8, 12] that the relationship between control velocities and changes in observed image errors are expressed by "stacking" the individual image Jacobians for each camera, now expressed in a common coordinate system. Furthermore, the estimate of z (depth) in the Jacobian becomes trivial using triangulation from the two cameras. If we return to our list of examples, the following comments apply:

Pure Translation In the case of feature points and pure translation, the stacked Jacobian matrix spans the entire space of robot motions (except for points along the baseline of the two-camera systems), and therefore it is not possible to define interesting

virtual fixtures other than point targeting.

If we consider the case of following a line, however, then when we stack the two Jacobians as before and apply the general vision-based virtual fixture rule, we arrive at a control law that effectively creates a prismatic joint that permits motion strictly along a line in space.

General Motion For general motion, we see that the "stack" of two Jacobians for feature-point servoing creates a spherical joint: the preferred degrees of freedom are motion on a sphere about the observed point while maintaining direction to the point (two degrees of freedom) and rotation about that line of sight (one degree of freedom). This may, at first, seem counter-intuitive since the stacked Jacobian has 4 rows. However, due to the epipolar constraints of the camera, one of these constraints is redundant and thus the Jacobian spans only three degrees of freedom. If we add a second point, we further reduce the degrees of freedom by two, with the remaining allowed motion being rotation about the line defined by the two observed points. A third observed point completely determines the pose of the observed (or observing) system, and so virtual fixturing once again reduces to motion to a target pose.

In the case of placing a point on a line, the image constraints now create a constraint on two degrees of positional freedom. It is, however, still possible to rotate in any direction (with the constraint that the rotation preserved distance to the observed point) and to move along the line. Placing a second point on the line reduces this to two degrees of freedom (rotation about the line and translation along it).

For a complete categorization of image-plane constructions for two-camera systems, we refer the reader to [8, 6].

4 Preliminary Tests

A preliminary version of the algorithms described above were implemented on the JHU Steady Hand Robot system (SHR) [16]. Here, we briefly describe the setup, the results, and its relationship to the more general framework given above. More details can be found in [1].

The robot was equipped with a vision sensor rigidly attached to the force-sensing handle on the end effector. We chose to execute two-dimensional tasks parallel to the image plane, which was in turn arranged to be parallel to two of the base stages of the robot. We performed experiments using a CCD camera at the macro scale and a grin lens endoscope at the micro scale. The vision sensor always viewed

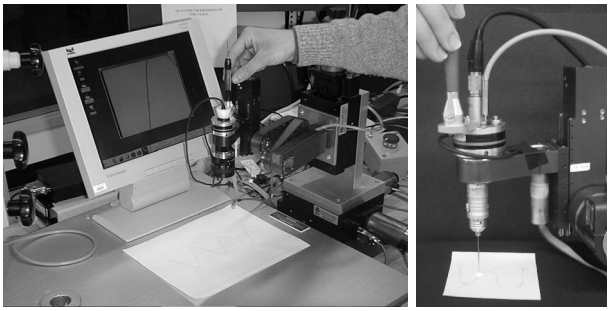


Figure 1: *The experimental setup of the steady hand robot using virtual fixtures to assist in path following and positioning tasks: left, macro scale; right, micro scale.*

the task plane, allowing reading of the motion references and real-time display of task execution (Figure 1). On-screen display of the task execution is useful for operators at the macro scale, and essential at the micro scale, as it would be impossible to complete the task using the naked eye.

The path was furnished to both the system and the user by printing a sine curve (35mm amplitude, 70mm wavelength, and 0.54mm width) on the task plane (in black on white paper). At micro scale, it was not possible to print a sufficiently smooth curve, so we instead embedded a wavy human hair (about 80 μm diameter) in glue on a yellow sheet of paper. In the macro case, the camera was positioned 200mm from the paper, yielding a pixel footprint of 0.066mm on the working surface. In the micro case, the endoscope was about 150 μm above the working surface, yielding a pixel footprint of about 1 μm (Figure 2).

The center of the image was graphically marked, and users were instructed to perform path following tasks relative to this mark. The sensor on the handle was used to record user commands. The force sensor resolution is $12.5mN$ and force values are expressed as multiples of this base unit.

Visual tracking (XVision system [9]) was used to measure (in real time) the local position and tangent direction, $\mathbf{d}$, to the path. Subpixel interpolation was used to increase the precision of these measurements. The vision and control subsystems executed on two different PCs, and the data exchange was realized over a local network. The control system operated at 100 Hz, using the most recent available data from the vision system and handle force sensor.

In terms of our previous formulation, the marked image location at the image center means that $\mathbf{x} = 0$ Further, the workspace is a plane, ($\mathbf{x} \in \Re^2$). The preferred direction is given by the tangent measurements from the tracking algorithm. Implicitly, the

control law used to position the manipulator on the line is

$$\mathbf{u} = \mathbf{s} - \mathbf{x} = \mathbf{s} \tag{22}$$

where $\mathbf{s} \in \Re^2$ is the current location of the visual tracker in the image. Further, $\mathbf{s}$ was constrained to lie along the line through the marked image location, normal to $\mathbf{d}$. Choosing $\mathbf{d}$ as the preferred direction of motion, we see the conditions of the general virtual fixturing rule are satisfied.

This class of virtual fixtures has been tested at both macro and micro scales. Results for a specific user and a wide class of compliances and situations can be found in [1, 2]. Tests for a larger class of users can be found in [20].

5 Conclusion

In this paper, we have outlined a broad theory of compliant virtual fixtures, and have applied that theory to the specific case of vision-guided assistance. Our earlier work suggests that such virtual fixtures can be a useful aid to dextrous manipulation.

In many ways, this paper is intended to point toward interesting further directions to be explored. First and foremost, we have begun to develop a general means for translating control algorithms into virtual fixtures. However, the treatment to this point has not been sufficiently formal to determine when such a translation is possible. In particular, we have not described how to exhibit a set of preferred directions that are consistent with a control input. Further, we have not offered a formal definition of guidance with virtual fixtures that would permit a general theoretical statement of an equivalence between active control and passive virtual fixturing. These remain interesting open problems.

On the practical side all of our experiments with vision-guided virtual fixtures have been within a very specific setup. Numerous issues must be solved before a robust, general implementation of vision-guided virtual fixtures can be achieved. For example, in our previous work, gain shaping was essential to maintain stability. Similarly, there needs to be careful gain shaping to accommodate the differing scales of forces and torques. More importantly, the ergonomics of this wider class of guidance modes remains to be explored.

Finally, it is important to point out that most virtual fixtures apply in a very limited task context. Thus, it is important to consider how to combine guidance modes in parallel (e.g. a force-based guidance mode along a needle axis combined with a vision-based virtual fixture to position the needle and a position-based alignment fixture), and to sequence them. Initial work on the latter problem will be reported in a forthcoming paper.

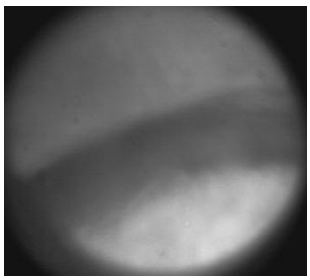


Figure 2: *Endoscope image of the 80 μm-diameter human hair used as the path in micro scale experiments.*

Acknowledgments

We would like to acknowledge the contributions of Pannada Marayong and Allison Okamura to this work. This material is based upon work supported by the National Science Foundation under Grant Nos. IIS-0099770 and EEC-9731478. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the National Science Foundation.

References

[1] A. Bettini, S. Lang, A. Okamura, and G. Hager. Vision assisted control for manipulation using virtual fixtures. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1171–1176, 2001.

[2] A. Bettini, S. Lang, A. Okamura, and G. Hager. Vision assisted control for manipulation using virtual fixtures: Experiments at macro and micro scales. In *Proc. IEEE International Conference on Robot. Automat.*, pages 3354–3361, 2002.

[3] F. Chaumette, P. Rives, and B. Espiau. Classification and realization of the different vision-based tasks. In K. Hashimoto, editor, *Visual Servoing*, pages 199–228. World Scientific, 1994.

[4] P. Corke and S. Hutchinson. A new partitioned approach to image-based visual servo control. *IEEE Trans. Robot. Autom.*, 17(4):507–515, 2001.

[5] N.J. Cowan, J. Weingarten, and D.E. Koditschek. Vision-based control via navigation functions. *IEEE Trans. Robot. Autom.*, 2002. To Appear.

[6] Z. Dodds. *Task Specification Languages for Uncalibrated Visual Servoing.* PhD thesis, Yale University, 2000.

[7] Z. Dodds, G. Hager, A.S. Morse, and J.P. Hespanha. Task specification and monitoring for uncalibrated hand/eye coordination. In *Proc. IEEE Int. Conference Rob. Automat.*, pages 1607–1613, May 1999.

[8] G. D. Hager. A modular system for robust hand-eye coordination. *IEEE Trans. Robot. Automat*, 13(4):582–595, 1997.

[9] G. D. Hager and K. Toyama. The "XVision" system: A general purpose substrate for real-time vision applications. *Comp. Vision, Image Understanding.*, 69(1):23–27, January 1998.

[10] R. M. Haralick and L. G. Shapiro. *Computer and Robot Vision*. Addison Wesley, 1993.

[11] J. Hespanha, Z. Dodds, G. D. Hager, and A. S. Morse. What tasks can be performed with an uncalibrated stereo vision system? *International Journal of Computer Vision*, 35(1):65–85, 1999.

[12] S. Hutchinson, G. D. Hager, and P. Corke. A tutorial introduction to visual servo control. *IEEE Trans. Robot. Automat*, 12(5):651–670, 1996.

[13] M. Jägersand. *On-line Estimation of Visual-Motor Models for Robot Control and Visual Simulation*. PhD thesis, University of Rochester, 1997.

[14] D. Kim, A. Rizzi, G. D. Hager, and D. Koditschek. A "robust" convergent visual servoing system. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume I, pages 348–353. IEEE Computer Society Press, 1995.

[15] R. Kumar, T. M. Goradia, A. Barnes, P. Jensen, L. M. Auer L. L. Whitcomb, D. Stoianovici, and R. H. Taylor. Performance of robotic augmentation in microsurgery-scale motions. In *Proceedings of Medical Image Computing and Computer Assisted Intervention*, volume 1679 of *Lecture Notes in Computer Science*, pages 1108–1115. Springer-Verlag, 1999.

[16] R. Kumar, G.D. Hager, P. Jensen, and R.H.Taylor. An augmentation system for fine manipulation. In *Proc. Medical Image Computing and Computer Assisted Intervention*, pages 956–965. Springer-Verlag, 2000.

[17] R. Kumar, P. Jensen, and R. H. Taylor. Experiments with a steady hand robot in constrained compliant motion and path following. *IEEE RO-MAN*, pages 92–97, 1999.

[18] F. Lai and R.D. Howe. Evaluating control modes for constrained robotic surgery. In *IEEE International Conference on Robotics and Automation*, pages 603–609, 2000.

[19] E. Malis, F. Chaumette, and S. Boudet. 2 1/2 D visual servoing. *IEEE Trans. Robot. Autom.*, 15(2):238–250, 1999.

[20] P. Marayong, A. Bettini, and A.M. Okamura. Effect of virtual fixture compliance on human-machine cooperative manipulation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, page To Appear, 2001.

[21] N. P. Papanikolopoulos and P. K. Khosla. Adaptive Robot Visual Tracking: Theory and Experiments. *IEEE Trans. on Automatic Control*, 38(3):429–445, 1993.

[22] L. Rosenberg. *Virtual Fixtures*. PhD thesis, Dept. of Mechanical Engineering, Stanford University, 1994.

[23] G. Strang, editor. *Linear Algebra and its Applications*. Academic Press, New York, 1980.

[24] R. Taylor, P. Jensen, L. Whitcomb, A. Barnes, R. Kumar, D. Stoianovici, P. Gupta, Z. X. Wang, E. deJuan, and L. Kavoussi. Steady-hand robotic system for microsurgical augmentation. *The International Journal of Robotics Research*, 18(12):1201–1210, 1999.

[25] J.N. Weiss. Injection of tissue plasminogen activator into a branch retinal vein in eyes with central retinal vein occlusion. *Opthalmology*, 108(12):2249–2257, 2001.

Home Alone: Mobile robot visual servoing

Kane Usher[a,b] Peter Corke[a] and Peter Ridley[b]

[a] CSIRO Manufacturing Science and Technology
P.O. Box 883
Kenmore 4069, Queensland, Australia
[b] School of Mechanical, Manufacturing and Medical Engineering
Queensland University of Technology, George Street
Brisbane 4001, Queensland, Australia
Email: kane.usher@csiro.au

Abstract

In this paper, we extend the planar visual servoing method of Corke [8] to the case of a nonholonomic, car-like vehicle. The control method presented is based on a Lyapunov-like function, such as that presented by Aicardi et al. [12] and also incorporates a hypothesis on the navigation behaviour of the desert ant cataglyphis bicolor. *The method allows positioning to a learnt location based on feature bearing angle and range discrepancies between the robot's current view of the environment and that at the learnt location. We present simulations and experimental results, the latter obtained using our outdoor mobile platform which is fitted with, among other sensors, an omnidirectional camera.*

1 Introduction

The problem we address here is that of stabilising a nonholonomic mobile robot to a specified pose using purely sensor-based strategies. To this end we have drawn inspiration from the more traditional closed-loop control-based approaches, and those taken by researchers interested in the navigational behaviour of insects. In fact, there are vast similarities between these two approaches; here we exploit the advantages of each.

Insects in general display amazing navigation abilities, traversing distances far surpassing the best of our mobile robots on a relative scale. To do this, evolution has provided insects with many 'shortcuts' enabling the achievement of relatively complex tasks with a minimum of resources in terms of processing power and sensors [27]. In particular, the high ground temperatures encountered by the desert ant, cataglyphis bicolor, eliminate pheremones as a potential navigation aid, as is used by ants in cooler climates [19]. The desert ant navigates using a combination of path integration and visual homing.

Visual homing, also known as visual piloting, is the process of matching an agent's current view of a location in a distinctive locale to a (pre-stored) view at some target position, using any discrepancies between the two views to generate a command that drives the agent closer to the target position. The process enables the agent to 'find' positions in distinctive locales. These distinctive locales can then be linked to generate paths through an environment [14, 18], eliminating the need for complex map-like representations, instead embedding this knowledge in terms of what the agent's sensors can 'see' [17].

There are two classes of visual homing, the image-based and the landmark-based techniques. Image-based homing uses differences in the raw images taken at the current and target locations to derive a homing vector. Landmark-based homing uses salient features extracted from the current and target views to derive a homing vector [29]. Here, we are interested in the landmark-based techniques, and in particular those that do not require uniquely identified landmarks — landmarks are characterised by their range and bearing with respect to a compass direction.

Interesting strides have been made in servoing to a position through the application of hypotheses on insect navigation to the mobile robot case [19, 29, 30]. These strategies utilise the discrepancies between the agent's (robot's) current view of the workspace and a snapshot of the target location. For visual piloting, insects rely on the relative bearing angles between landmarks, and knowledge of an absolute reference direction, such as that provided by the polarisation pattern of sunlight [19, 28]. Insects cannot measure range directly due to their fixed focus optics and immobile eyes. However, estimates of range can be achieved through optic flow techniques and by estimating changes in the apparent size of an object [26].

In the control community, the problem of stabilising a mobile robot to a specific pose has generally been approached from two directions; the open loop strategies and the feedback control strategies. Open loop strategies seek to find a bounded sequence of control inputs, driving the vehicle from an initial position to some arbitrary position, usually working in conjunction with a motion planner (e.g. [20, 23]). The reactive, feedback control systems use the environment itself for navigation. However, due to the well-known limitations presented by Brockett, there is no smooth, continuous control law which can locally stabilise closed loop nonholonomic systems to a point [4]. These limitations can be overcome by either relaxing the constraints on desired pose (i.e. stabilising to a point without a guarantee on orientation, see e.g. [19, 25, 29, 30]), using discontinuous control techniques (see e.g. [2, 3, 7]), or by using time-varying control (see e.g. [25]).

For the task at hand, we argue the case for the feedback control methods, with vision used as the primary sensor.

These feedback control methods correspond with the insect inspired strategies discussed earlier. Feedback control systems are generally more robust to uncertainty and disturbances as opposed to their open-loop counterparts. All real mobile robots and sensors are subject to noise and uncertainty — feedback control would thus seem essential. However, it is clear that a measure of open loop planning is also usually required in order to prevent deadlock situations from occurring. Here, we are concerned only with the feedback component for the task of servoing to a position and orientation.

Similarities between the insect based strategies and those of the more rigorous control based visual servoing algorithms are abundant. However, when applied to mobile robots, the constraints of Brockett's theorem prevent the insect-based strategies from completely resolving the point stabilisation problem; they can servo to a position but cannot guarantee a particular orientation. In contrast, the control-based strategies are still in their infancy with regard to effective use of vision as a sensor. Both the closed loop control and insect based strategies bear striking similarity to the attractive fields found in artificial potential fields.

The remainder of this paper is arranged as follows: Section 2 details our control method and how we combine a non-linear control method with the insect-based literature; Section 3 describes our experimental system and briefly outlines some preliminary results; Section 4 concludes the paper and presents some directions of future interest.

2 Control Strategy

In this section we develop a control strategy which combines a Lyapunov-like formulation with a derivative of the insect inspired Average Landmark Vector model of navigation presented by Lambrinos et al. [19]. First we adapt the Lyapunov-based controller presented by Aicardi et al. [1,2] to a car-like vehicle. We then outline a technique which allows servoing to a pose, based on measurements of the relative distance and orientation to one or more landmarks in a workspace. The technique used has its basis in a model of ant navigation but incorporates some improvements available to our sensor's higher resolution view of the world.

2.1 Kinematics

This particular application of visual servoing seeks to relate the bearing angle changes of selected landmarks to vehicle motion through three state equations derived from transforming the Cartesian representation of a car-like vehicle's kinematics into a polar representation. The polar representation is a more 'natural' representation for a vehicle, and corresponds to the space used by the vehicle's sensors. This transformation also allows the limitations of Brockett's theorem to be overcome [12]. Referring to Figure 1, the kinematics in Cartesian space of our experimental vehicle are cart-like and given by:

$$\begin{aligned} \dot{x} &= v\cos\theta \\ \dot{y} &= v\sin\theta \\ \dot{\theta} &= v\frac{\tan\phi}{L} \end{aligned} \tag{1}$$

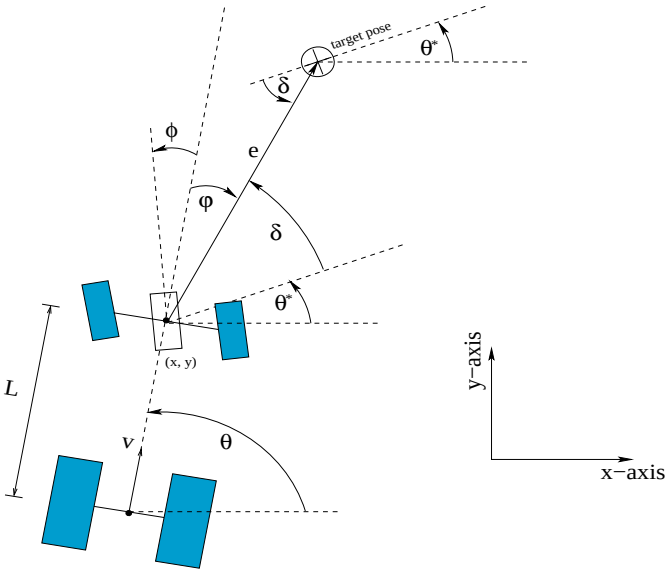


Figure 1: *The coordinate system and conventions used. All angles are measured counter-clockwise positive.*

where v is the vehicle's forward velocity (measured at the centre axle of the rear wheels), L is the vehicle's length, ϕ is the steering angle, and the point (x,y) refers to the centre of the front axle. In a polar representation, when referring to some target pose as marked in Figure 1, we define several quantities:

- δ — the desired heading angle to the goal position relative to the goal orientation, θ^*.
- φ — the desired heading angle to the goal position relative to the vehicle's current heading, θ.
- e — the distance to the goal position.

The angles' δ and φ are related by: $\delta = (\theta - \theta^*) + \varphi$. Hence, in polar coordinates, the kinematic equations become:

$$\begin{aligned} \dot{e} &= -v\cos\varphi \\ \dot{\varphi} &= \frac{v\sin(\varphi)}{e} - \frac{v\tan\phi}{L} \\ \dot{\delta} &= \frac{v\sin\varphi}{e} \end{aligned} \tag{2}$$

as θ^* is a constant.

The first of the above equations relates the rate of change of the distance e to the target position, to the vehicle's velocity and the relative orientation of the target position. The second relates the rate of change of the bearing to the target location, relative to the desired orientation θ^*, and the vehicle's control inputs (v and ϕ). The third equation is the basic optic flow equation, relating the rate of change of the target positions absolute orientation to the vehicle's translational velocity.

2.2 Lyapunov-like controller

Aicardi et al. [1, 2] presented a method based on a Lyapunov-like formulation enabling a unicycle-like vehicle to stabilise to a pose from anywhere in the workspace.

Our technique builds on this work, adapting it to a car-like vehicle, allowing servoing to a goal position and orientation.

The Lyapunov-like formulation seeks to minimise the 'energy' of the system described by the system's Lyapunov function — this function is minimised at the target pose. The system is driven towards the minimum energy state by ensuring that the derivative of the Lyapunov function is always non-positive. In fact, this has many similarities with the attractive fields used to drive robots towards a goal pose using artificial potential fields (see e.g. [16, 20]). This similarity could be further exploited by also including analogous repulsive components in the Lyapunov function, leading to combined homing and obstacle avoidance.

In order to stabilise the system described by equation 2 to a particular pose, we consider the simplest choice for a candidate Lyapunov function; the positive definite form:

$$\begin{aligned} E &= E_1 + E_2 \\ &= \frac{1}{2}\lambda e^2 + \frac{1}{2}\left(\varphi^2 + k_3\delta^2\right) \quad \lambda, k_3 > 0 \end{aligned}$$

Taking the derivative of this function gives:

$$\begin{aligned} \dot{E} &= \dot{E}_1 + \dot{E}_2 \\ &= \lambda \dot{e} e + \left(\dot{\varphi}\varphi + k_3\dot{\delta}\delta\right) \\ &= -\lambda e v \cos\varphi + \left[\varphi\left(\frac{v\sin(\varphi)}{e} - \frac{v\tan\phi}{L}\right) + \delta\frac{v\sin\varphi}{e}\right] \end{aligned}$$

The first term can be made non-positive with the following choice for the vehicle's velocity:

$$v = k_1 e\cos\varphi \qquad k_1 > 0 \tag{3}$$

leading to: $\dot{E}_1 = -\lambda k_1 e^2\cos^2\varphi$.

The second term can be made non-positive by choosing the arctangent of the vehicle's steering angle to be:

$$\arctan\phi = \frac{L}{e}\left[\frac{k_2\varphi}{k_1\cos\varphi} + \frac{\sin\varphi}{\varphi}\left(\varphi + k_3\delta\right)\right] \quad k_2 > 0 \tag{4}$$

where the earlier choice for velocity has been substituted. This leads to $\dot{E}_2 = -k_2\varphi^2$.

The overall derivative of the Lyapunov function is then:

$$\dot{E} = -\lambda k_1 e^2\cos^2\varphi - k_2\varphi^2$$

which is always non-positive. By invoking LaSalle's invariant set theory, it can be shown that the above system is globally asymptotically stable and converges to $(\varphi, e, \delta) = (0,0,0)$.

The control technique derived above assumes that the distance and angles to the target location can be measured. In the next section, we detail a landmark-based technique which yields this information without explicit knowledge of the robot's pose; sensing is provided by an omnidirectional camera and a compass.

2.3 Ant navigation

The desert ant, cataglyphis bicolor, is unable to use pheremones to navigate due to the high ground temperatures found in its habitat. Thus, it relies on a combination of visual piloting and path integration, enabling it to find nest openings of less than a few millimetres after foraging journeys of several hundred metres [19, 31]. Figure 2 shows a particular foraging path such an ant [32]. On this journey, the ant has travelled a round trip distance of over two hundred metres, returning to a nest with an opening of less than 5 mm, equating to a drift rate of less than 0.0025% in its navigation system. Comparing this to high-end commercial inertial navigation units for land based navigation, which have drift rates of the order 0.1% of distance travelled [11], demonstrates the effectiveness of this insects navigation system.

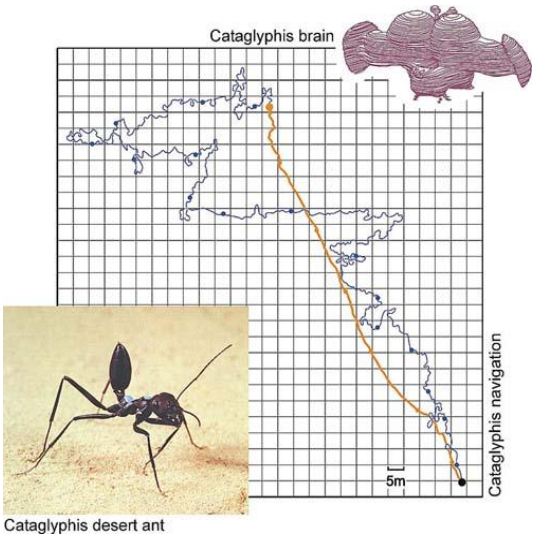


Figure 2: *An example of the amazing navigation feats of the humble ant. Diagram courtesy www.neuroscience.unizh.ch/e/groups/wehner00.htm, Rudiger Wehner.*

Here however, we are interested in the visual piloting component of the ants navigation system. When visual piloting, ants take a rather unprocessed view of the target location and match it with a current view, using the discrepancies to derive a direction of movement. Matching characteristics used are the differences in landmark bearings, apparent size and apparent height [19].

An elegant homing method developed from hypotheses on how these ants might use visual piloting is the Average Landmark Vector model. An ALV for any particular position in the workspace is found by summing unit vectors towards all currently visible landmarks, and dividing by the number of landmarks. By matching the current ALV with a pre-stored ALV of the target location, a homing vector can be formed which drives the agent (robot) towards the target location [19]. In order to consistently add the vectors in the ALV model, an absolute reference direction is required, and, unless apparent size information is incorporated, a minimum of three landmarks is needed.

In mathematical terms, a single landmark vector is defined as:

$$L_i(x) = \frac{x - x_i}{||x - x_i||}$$

where x is the current location and x_i is the landmark location. Note that $L_i(x)$ gives the landmark's orientation relative to the current position x; all that is required to measure L_i is the absolute bearing of the landmark. The Average Landmark Vector at location x is then given by:

$$ALV(x) = \frac{1}{n}\sum_{i=1}^{n} L_i(x)$$

where n is the number of landmarks. The ALV at the target location is recorded and designated ALV_t. Given ALV_t and the current Average Landmark Vector, ALV_c, the homing vector is given by:

$$H(x) = ALV_c - ALV_t$$

The ALV method does not require any knowledge of the robot's location. All that is needed is the target ALV vector, which represents home, and the bearings of the landmarks that can be seen at the current location. Thus, it is a sensor-based strategy. Refer to Figure 3 (a) for an illustration of the ALV method. Figure 3 (b) shows the attractive nature of the ALV method for a workspace with several landmarks, illustrating its similarity to the e and φ measures used in the Lyapunov controller.

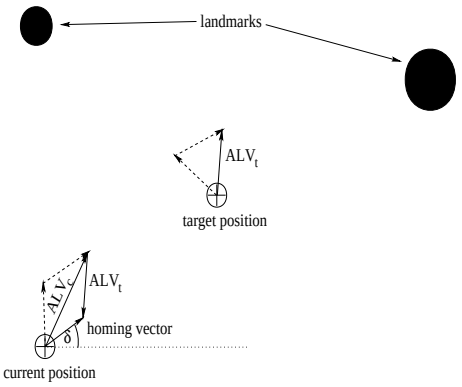
Consider a workspace of $10\,\text{m} \times 10\,\text{m}$, with the origin defined as the home position, and landmarks placed at $(8,8), (-5,5), (-3,-6), (4,-7)$ and $(3,9)$. We can discretise the workspace and calculate the magnitude and direction of the homing vector at discrete positions in the environment as shown in Figure 4. As a comparison, the actual angle and distance toward home for each position are also shown. Note that although the ALV method gives a fairly close approximation of the angle and (scaled) distance to the goal, there are slight differences in the angle measure. Furthermore, the magnitude of the ALV homing vector is *not* a monotonic function of the distance to the goal. Figure 5 shows the angle differences between the angles generated by the ALV strategy and the actual angles toward home.

We have found that if we use the homing vector as generated by the ALV strategy to estimate δ and a scaled version of e as the inputs to the Lyapunov controller of equations 3 and 4, these slight differences lead to the agent obtaining the home position, but with a steady-state error in orientation, even in the highly idealised case of the simulation. The severity of this steady-state error is increased as the landmark arrangement becomes more asymmetric.

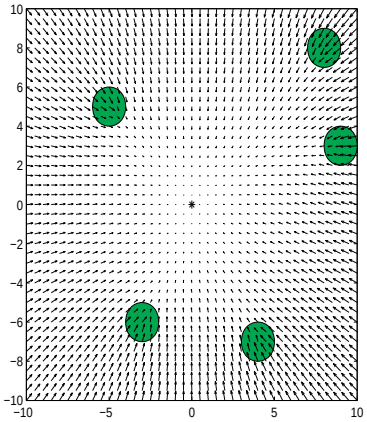
2.4 Improved ALV

These slight errors led us to investigate improvements to the ALV method. In its original incarnation, the ALV method required the bearings to landmarks only. Range information could be incorporated by including landmark apparent size, and slight improvements to its behaviour could be made. However, we have found that by including range information directly, a significant improvement is made and in fact, the distance and angle to the goal are yielded directly.

Mathematically, the improved ALV, or IALV[1] is derived as follows: the individual landmark vectors are given by their range and direction:



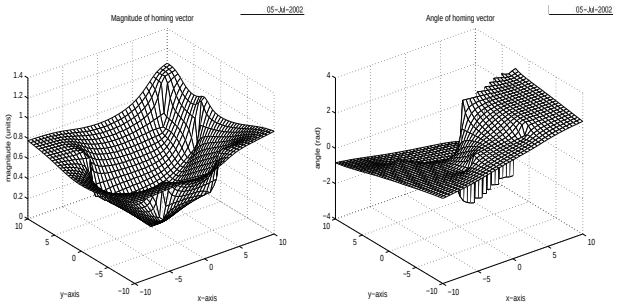
(a) How the homing vector is calculated.



(b) Homing vector calculated for a particular landmark set at each point in the environment.

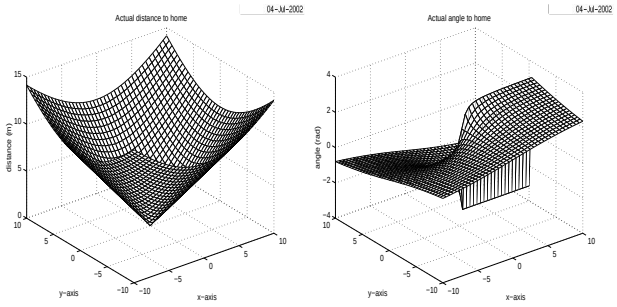
Figure 3: *The average landmark vector model of insect navigation is shown in (a). Unit vectors towards salient features in the environment are combined to form the average landmark vector at a particular location (ALV_c in the diagram). ALV_c is compared to the average landmark vector at the target location, ALV_t, to form the homing vector V_H. Figure (b) shows the attractive nature of the vector for a home position defined at the origin. The homing vector has been calculated at every point in the environment.*

[1] Perhaps augmented ALV would be more appropriate here as the original incarnation of the ALV method avoided the use of range information directly.



(a) Homing vector magnitude.

(b) Homing vector direction.



(c) Actual range to home.

(d) Actual angle towards home.

Figure 4: *Illustration of the homing vectors generated by the ALV method applied at every point in a discretised workspace. Also shown are the actual distances and angles toward home. Note the similarity in the measures.*

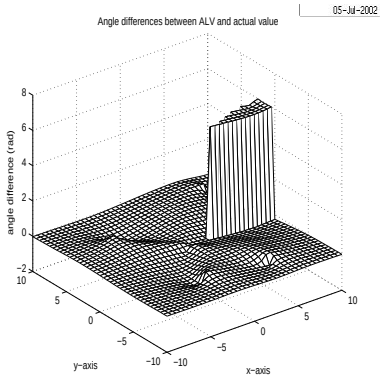


Figure 5: *Angle differences between angles generated by the ALV method and the actual angles toward home. The differences along $y = 0$ for positive x are significant, as are the ripples around the home position.*

$$IL_i(x) = x - x_i$$

where x is the current location and x_i is the landmark location. All that is required to measure IL_i is the absolute bearing of the landmark and its range. The IALV at location x is then given by:

$$IALV(x) = \frac{1}{n}\sum_{i=1}^{n} IL_i(x)$$

where n is the number of landmarks. The IALV at the target location is recorded and designated $IALV_t$, and as before, given $IALV_t$ and the current Improved Average Landmark Vector, $IALV_c$, the homing vector is given by:

$$H(x) = IALV_c - IALV_t$$

An example of the IALV method is shown in Figure 6. In essence, the IALV method is equivalent to finding a position relative to the centroid of the landmarks in the workspace.

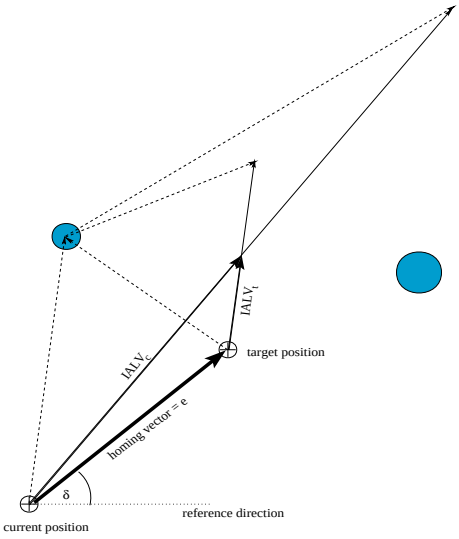
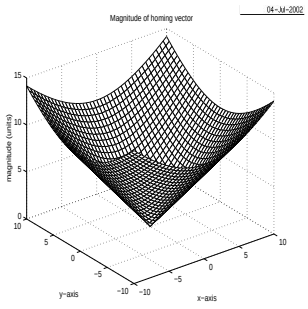


Figure 6: *Illustration of the the IALV method for two landmarks in a workspace. The IALV's are found by adding the vectors to the individual landmarks, and dividing the resulting vector by the number of landmarks. The home vector is then calculated by subtracting the target IALV from the currently IALV.*

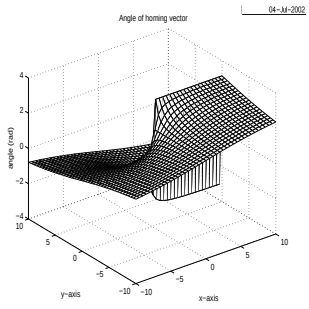
Figure 7 shows the magnitude and direction of the IALV homing vector calculated over the discretised workspace, with the accompanying figures indicating the differences between actual values to the goal and those generated by the IALV homing vector.

As with the ALV method, the IALV method is purely sensor-based. Landmark bearings are readily ascertained with an omnidirectional camera. If a flat-earth assumption is made, range information can also be derived from an omnidirectional camera image through the geometry of the camera/mirror optics. Alternatively, optic flow techniques could be used to determine landmark range, as in [5].

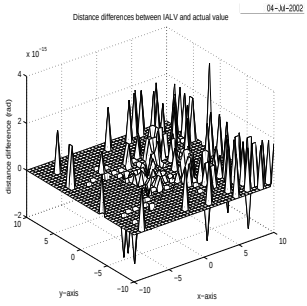
Implementations of the ALV method to date have used an omnidirectional camera, restricting the analysed image to a horizontal slice of the environment. Here, we wish to use the whole image and we have found that unwarping of the image is unnecessary.



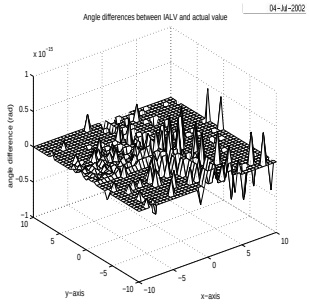
(a) Magnitude of the IALV homing vector.



(b) Direction of the IALV homing vector.



(c) Error in distance calculation.



(d) Error in angle calculation.

Figure 7: *The improved ALV method applied to the same workspace is Figure 4. Note that this method yields the distance and orientations to the goal location directly.*

One of the advantages of the ALV, and hence the IALV method, is that knowledge of a target location is contained within a single quantity. This reduces the need for complex map-like representations of the environment and is well suited for a topological navigation method, (see e.g. [10, 13, 18, 21, 22, 24]).

Importantly, with this method landmarks need not be unique, and the need for landmark correspondence is also bypassed. Many of the other homing algorithms require that the landmarks in the current image by 'paired' or matched with those at the target location, usually by minimisation of the sum of the bearing differences (see e.g. [30]). If landmarks are occluded or missing, these methods strike trouble. In fact we have found that these methods are normally restricted to a workspace enclosed by the polygon formed by the landmarks (which in a loose sense guarantees that landmarks will not be occluded). Of course, like all sensor-based techniques, this method has a finite catchment area, limited by the omnidirectional sensor's range and in addition has the potential to suffer from perceptual aliasing, or in a similar sense, the local minima problem.

The homing vector provided by the IALV method provides a means of driving the agent towards home but does not provide a means of guaranteeing an orientation on reaching it. However, the distance and orientation information provided by the IALV method could be fed into the Lyapunov based controller presented in equations 3 and 4. The next section presents simulations of this very point.

2.5 Simulations

In these simulations, landmarks are modelled as points, consistent with the requirements of the IALV model which merely requires the range and bearing to one or more non-unique landmarks in the workspace. An IALV is taken at a target location, nominally the origin, with this target IALV being matched with the agent's current IALV throughout the vehicle's journey.

The vehicle model used here consists of the Cartesian version of the vehicle kinematic equations (as given in equation 1), with the loop closed through calculation of the homing vector and the subsequent calculation of the control inputs via the Lyapunov-based controller presented earlier. Figure 8 shows the path generated, pose, and control inputs for a starting pose of $(x,y,\theta) = (-10,-10,0)$ and a goal pose of $(0,0,0)$, while Figure 9 shows the same information for a starting and goal pose of $\left(-10,10,\frac{\pi}{2}\right)$ and $\left(0,0,-\frac{pi}{4}\right)$. Gains were set to $(k_1,k_2,k_3) = (0.5,1.2,1.5)$ for these simulations. The method works for all starting and goal poses.

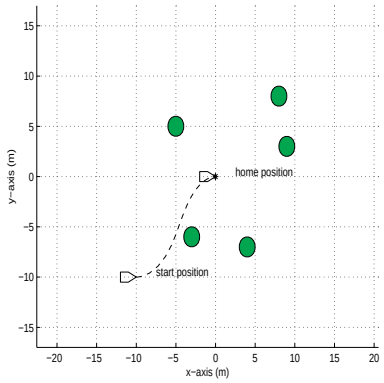
2.6 Discussion

Using the Lyapunov controller, servoing to a pose based on the relative position of a set of landmarks is a relatively straight forward task. The method is simple and leads to intuitive paths for the vehicle. In fact, the Lyapunov energy function is analogous to the attractive component of an artificial potential field. Clearly this method could be adapted to include obstacle avoidance by adding a complementary repulsive field to obstacles sensed in the environment.

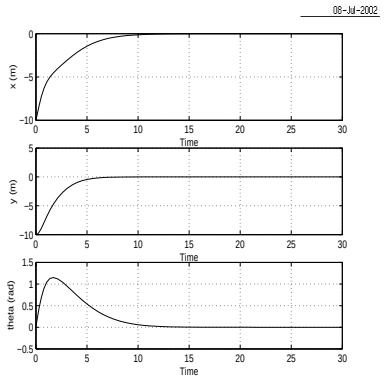
The main advantage of the homing method presented is the avoidance of explicit localisation; it is achieved through a rather simple mechanism which yields the inputs to the Lyapunov controller directly. Further to this, no landmark correspondence is required, circumventing the need for landmark association. This is in opposition to many of the other homing methods which require landmarks in the current and target views to be 'paired up'. We have found that it is this association process that most limits the catchment areas of such methods.

3 Experiments

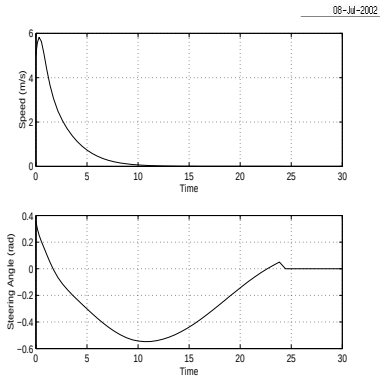
In this section we describe the experiments testing the validity of the IALV/Lyapunov control technique. To this end, we test the theory on our outdoor mobile robot; the Little Red Tractor. Our workspace is a flat, concreted, outdoor environment in which we have placed a set of bright orange traffic cones which act as our landmarks. The LRT is equipped with an omnidirectional camera which allows us to track the landmarks through colour segmentation, while a magnetometer provides us with a reference direction.



(a) Vehicle path.

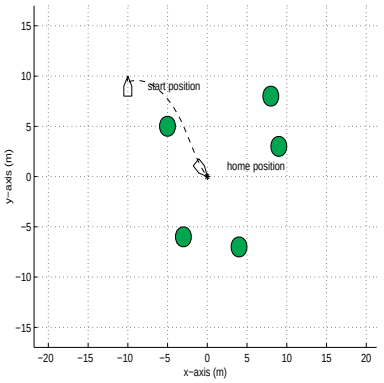


(b) Robot pose throughout journey.

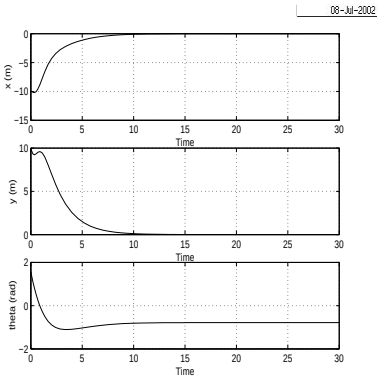


(c) Control demands.

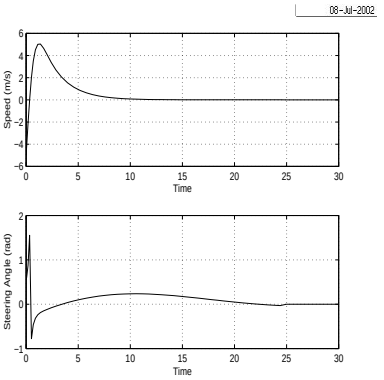
Figure 8: *Simulation results for a starting pose of* $(x,y,\theta) = (-10,-10,0)$ *and a goal pose of* $(0,0,0)$*. Note that no obstacle avoidance has yet been included.*



(a) Vehicle path.



(b) Vehicle pose throughout journey.



(c) Control demands.

Figure 9: *Simulation results for a starting pose of* $(x,y,\theta) = \left(-10,10,\frac{\pi}{2}\right)$ *and a goal pose of* $\left(0,0,-\frac{pi}{4}\right)$.

First we introduce our robotic testbed and briefly describe our sensing and image processing. We then detail the experiments, which at the time of publication were incomplete. We finish with a discussion of the results.

3.1 Robotic testbed

The experimental platform is a Toro ride-on mower which has been retro-fitted with actuators, a control system, and a computer, enabling control over the vehicle's operations. All control and computing occurs on-board. The vehicle is fitted with an array of sensors including odometry, GPS, a magnetometer, a laser range-finder and an omni-

directional camera (see Figure 10 for a photograph of the vehicle). For the experiments cited here, the primary sensor used is the omnidirectional camera with the magnetometer providing an absolute reference direction.

Figure 10: *The experimental platform. Note the omnidirectional camera mounted over the front wheels and the box at the rear which houses the control and computer system.*

3.2 Image processing

The distinguishing feature of the landmarks used in this experiment is their colour. Hence, we use colour segmentation to track them; not a trivial task in an outdoor environment with no control over lighting conditions. Our framegrabber provides us with a YCrCb signal, and, to reduce processing time, we work directly in this colour space.

Essentially, we use a bivariant histogram on the Cr and Cb of the objects we wish to track, creating a two-dimensional lookup table on colour. As each image is acquired, pixels that fall within the histogram are flagged as belonging to a landmark. We then use several morphological processes to eliminate isolated pixels and to 'close' any holes in grouped pixels. Blob extraction is then performed, the objects sorted by size, and very small and very large objects eliminated. The coordinates of the centroid of each blob are then converted to a polar form, giving the radial distance and bearing of each blob with respect to the centre of the image. At no stage do we attempt to 'unwrap' the image; we believe this to be a waste of valuable processing time and instead work directly with the radial image.

Relative landmark bearings are yielded directly from the above process. These bearings can then be combined with the robot's orientation obtained from the magnetometer to give bearings with respect to the reference direction. Landmark range however is not directly available. The next section describes how we use the optics of our camera/mirror system to determine landmark range.

Range from image radius. Using a flat-earth assumption, an estimate of range can be determined in a similar manner to Horswil's range from height in image method [9,15] if the geometry of the camera/optic system is known. An alternative to using the geometry of the system would be to determine an empirical relationship between ground-plane range and radial image distance.

The mirror in our omnidirectional camera/mirror assembly has equiangular optics meaning that for a given angle of incidence into the mirror, the reflected ray is elevated by a particular gain depending on the camera/mirror separation distance [6]; for an illustration of this point refer to Figure 11. The equation describing the surface of such mirrors [6] is :

$$\left(\frac{r}{r_o}\right)^{-\frac{1+\alpha}{2}} = \cos\left[\frac{\theta(1+\alpha)}{2}\right] \tag{5}$$

where the parameters are defined with reference to figure 11. Mapping distance in the ground plane to radial pixel

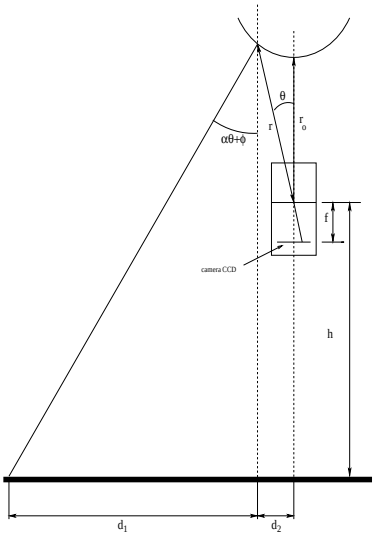


Figure 11: *The optics of the camera-mirror system related to the ground plane. Diagram adapted from [6]*

distance can be achieved through simple geometry. Referring to Figure 11, the distance d_1 can be calculated by:

$$d_1 = (h + r\cos\theta)\tan(\alpha\theta + \phi) \tag{6}$$

and the distance d_2 is given by:

$$d_2 = r\sin\theta \tag{7}$$

Adding these together gives the theoretically determined ground plane distance, R_t. Combining equations 5, 6 and 7, the full equation for R_t is:

$$\begin{aligned} R_t &= r_o\left(\cos\left(\tfrac{\theta(1+\alpha)}{2}\right)\right)^{-\frac{2}{1+\alpha}}\sin\theta \\ &+ \left[h + r_o\left(\cos\left(\tfrac{\theta(1+\alpha)}{2}\right)\right)^{-\frac{2}{1+\alpha}}\right]\tan(\alpha\theta+\phi) \end{aligned} \tag{8}$$

Now to map a ground plane distance into the image plane we need a relationship between θ and radial pixel distance p. A relationship between r_o and f can be found through knowledge of a known radial image distance (measured in pixels) and an actual distance in an image. Here, we use the diameter of the mirror (which is known to be 7.36 cm), and the radius of its edge as it appears in the image. Referring to Figure 12, which is applicable to our system,

$$\tan\theta = \frac{u}{f} = \frac{36.8\times10^{-3}\,\mathrm{m}}{r_o}$$

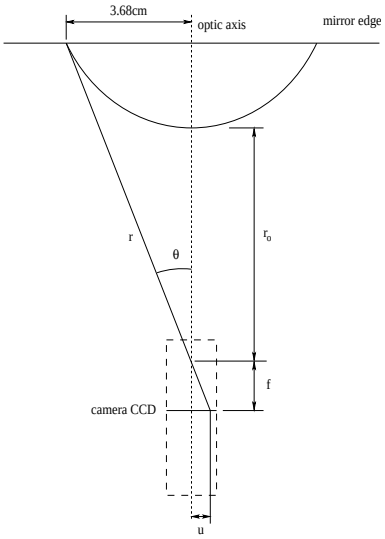


Figure 12: *Relationship between f and u.*

For the case of a $288\,\text{lines} \times 384\,\text{pixels}$ image, the edge of our mirror appears in the image at a radial distance of $u = 178\,\text{pixels}$; the CCD used here is 1/3" across its diagonal, and, using the number of lines at this image resolution (assuming the pixels are square), we arrive at $u = 178 \times \frac{5.987\times10^{-3}\,\text{m}}{288\,\text{lines}} = 3.7 \times 10^{-3}\,\text{m}$. Hence,

$$\frac{r_o}{f} = 9.9454$$

A relationship for θ at any pixel radius p is then given by:

$$\theta = \arctan\left[\frac{p \times \frac{5.987\times10^{-3}\,\text{m}}{288\,\text{lines}} \times 9.9454}{r_o}\right] \qquad (9)$$

where r_o is given in metres. With knowledge of the four parameters $r_o, \phi\, h$, and α (which we determine experimentally), and the radial image distance of a landmark, its ground-plane range can be estimated.

Figure 13 shows a plot of estimated range as determined from the process outlined above, and the actual ground plane range. The match is excellent for ranges of up to about 10 m, with the error between experimental data and estimated range being less than 0.2 m for ranges less than 7.5 m. Inaccuracies at greater ranges can be attributed to measuring errors and to degradation of the constant α assumption at high angular elevations [6], i.e. high radial distances.

3.3 Experiments

Hardware difficulties prevented complete experimentation at the time of publication. However, some preliminary results will be presented. The first experimental run placed the robot in a workspace containing a single landmark. At the target pose, the landmark was located at approximately 2.85 m directly in front of the robot; all angles and distances are defined with reference to this pose. Figure 14 (a) shows

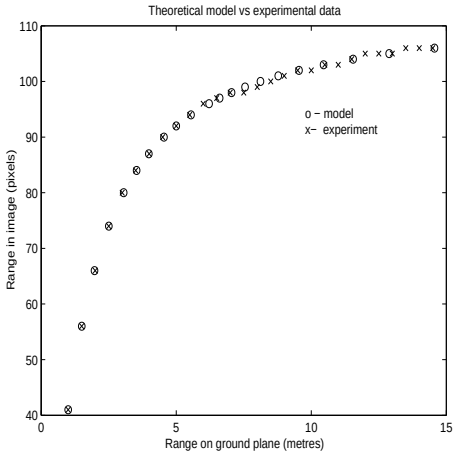


Figure 13: *Comparison of estimated and actual range data. The estimated data is plotted as an 'o', while the actual ground plane range is plotted as a '*'.*

the robot's initial and final pose; clearly this is a rather simplistic task but it is the first step in demonstrating the effectiveness of the method. Figure 14 (b) shows the control demands as calculated online by the IALV/Lyapunov controller while Figure 14 (c) shows the response of the vehicle. Figure 14 (d) shows the values of the inputs to the Lyapunov controller; in this figure e is a measure of the distance to the goal position, while θ measures the orientation error. If the distance error dropped below 0.1 m and the orientation error was less than $\pm 0.1\,\text{rad} \approx \pm 6^\circ$ the controller was switched off and the vehicle halted.

3.4 Discussion

Although rather simplistic, this experiment demonstrates that the method indeed works. The gains in the controller of equations 3 and 4 were set at $(k_1, k_2, k_3) = (0.1, 0.5, 1)$, which are significantly different to those used in the simulation. The simulation gains were found by linearising the closed loop state equations about $(\varphi, \delta) = (0, 0)$, which ensures the vehicle approaches the target pose along the rectilinear path aligned with the target pose [1]. The analysis for the selection of the gains in the simulation was based on the kinematics of the system alone. Clearly the dynamics of the vehicle's velocity and steering loops, along with the significant delays caused by the image processing, will have a marked effect on the performance of the system and a much deeper analysis incorporating these effects is required in order to correctly select the controller gains for the real system. Along with extensive experimental verification of the method, this analysis will be the focus of our work in the immediate future.

4 Conclusion

We have described a method of stabilising a car-like vehicle to a target pose based on the discrepancies between a target view of the landmarks in a workspace and the robot's current view. The landmarks need not be unique; all that is required is their bearing and range. Our method combines

results from non-linear control theory with research derived from hypotheses on insect navigation.

The robot's view of the workspace is summarised by a single quantity, the Improved Average Landmark Vector, which augments the original formulation of Lambrinos et al. [19] with range information. At each instant, the robot compares the current IALV with that at the target location which yields directly the distance and orientation to the target position. At no stage is there a requirement for landmark correspondence; this represents a key advantage over many other homing methods. This information is then fed into a non-linear controller derived from a Lyapunov-like formulation, enabling the robot to servo to the target pose using omnidirectional vision and a compass. Furthermore, the method uses polar representations of the landmarks, and thus, the need for processor intensive image unwarping is circumvented.

We have presented simulations and a brief experiment demonstrating the validity of the technique. Further to this, we have presented a method of deriving range information from a single omnidirectional image, based on a flat-earth assumption. Our future work includes extensive experimental validation of the method, along with a deeper understanding of the impact of dynamic elements in the control loops. Finally, it is clear that this technique is ideally suited to topological navigation, and this too represents a direction for further research.

Acknowledgements

The technical support of the CMST Tractor Team — Peter Hynes, Stuart Wolfe, Stephen Brosnan, Graeme Winstanley, Pavan Sikka, Elliot Duff, Les Overs, Craig Worthington and Steven Hogan — is gratefully acknowledged, while Jonathon O'Brien at UNSW is gratefully thanked for the loan of the Toro ride-on mower.

The first author gratefully acknowledges the funding and technical support provided by CMST, the funding provided by an APA grant through QUT and would also like to thank Peter Corke and Peter Ridley for their guidance and support throughout this course of research.

References

[1] Michele Aicardi, Giuseppe Casalino, Aldo Balestrino, and Antonio Bichi. Closed loop smooth steering of unicycle-like vehicles. In *Proceedings of the 33rd Conference on Decision and Control*, pages 2455–2458, Lake Buena Vista, Florida, USA, December 1994.

[2] Michele Aicardi, Giuseppe Casalino, Antonio Bichi, and Aldo Balestrino. Closed loop steering of unicycle-like vehicles via Lyapunov techniques. *IEEE Robotics and Automation Magazine*, pages 27–35, March 1995.

[3] A. Astolfi. Exponential stabilization of a car-like vehicle. In *Proceedings of the International Conference on Robotics and Automation*, pages 1391–1396, Nagoya, Japan, 1995. IEEE.

[4] R.W. Brockett. Asymptotic stability and feedback stabilization. In R. W. Brockett, R. S. Millman, and H. J. Sussman, editors, *Differential Geometric Control Theory*, pages 181–191. Birkhauser, Boston, USA, 1983.

[5] J. S. Chahl and M. V. Srinivasan. Range estimation with a panoramic visual sensor. *Journal of the optical society of America*, 14(9), September 1997.

[6] J. S. Chahl and M. V. Srinivasan. Reflective surfaces for panoramic imaging. *Applied Optics*, 36(31):8275–8285, November 1997.

[7] F. Conticelli, D. Prattechizzo, F. Guidi, and A. Bichi. Vision-based dynamic estimation and set-point stabilization of nonholonomic vehicles. In *International Conference on Robotics and Automation*, pages 2771–2776, San Francisco, California, USA, 2000. IEEE.

[8] Peter Corke. Mobile robot navigation as a planar visual servoing problem. In *10th International Symposium of Robotics Research*, pages 217–223. IFRR, 2001.

[9] A.K. Das, R. Fierro, V. Kumar, B. Southall, J. Spletzer, and C.J. Taylor. Real-time vision based control of a nonholonomic mobile robot. In *International Conference on Robotics and Automation*, pages 1714–1719, Seoul, Korea, May 2001. IEEE.

[10] Tom Duckett and Ulrich Nehmzow. Exploration of unknown environments using a compass, topological map and neural network. Proceedings of International Symposium on Computational Intelligence in Robotics and Automation, 199. pp 312-317.

[11] H. R. Everett. *Robot Motion Planning - Sensors for Mobile Robots - Theory and Application*. A.K. Peters Ltd, 1995.

[12] Johan Forsberg, Ulf Larsson, and Ake Wernersson. Mobile robot navigation using the range-weighted Hough transform. *IEEE Robotics and Automation Magazine*, pages 19–26, March 1995.

[13] Jose Gaspar, Niall Winters, and Jose Santos-Victor. Vision-based navigation and environmental representations with an omnidirectional camera. *IEEE Transactions on Robotics and Automation*, 16(6):890–898, December 2000.

[14] Jiawei Hong, Xiaonan Tan, Brian Pinette, Richard Weiss, and Edward M. Riseman. Image-based homing. In *International Conference on Robotics and Automation*, pages 620–625, Sacramento, California, USA, April 1991. IEEE.

[15] I. Horswill. Polly: A vision-based arificial agent. In *Proceedings of the eleventh national conference on artificial intelligence (AAAI'93)*, Washington DC, USA, July 1993. MIT Press.

[16] Oussama Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *International Journal of Robotics Research*, 5(1):90–98, 1986.

[17] B. J. A. Krose and R. Bunschoten. Probabilistic localization by appearance models and active vision. In *International Conference on Robotics and Automation*, pages 2255–2260, Detroit, Michigan, 1999. IEEE.

[18] Benjamin Kuipers and Yung-Tai Byun. A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations. *Robotics and Autonomous Systems*, 8:47–63, 1991.

[19] Dimitrios Lambrinos, Ralf Moller, Thomas Labhart, Rolf Pfiefer, and Rudiger Wehner. A mobile robot employing insect strategies for navigation. *Robotics and Autonomous Systems*, 30:39–64, 2000.

[20] Jean-Claude Latombe. *Robot Motion Planning*. Kluwer Academic, 1991.

[21] Tod S. Levitt and Daryl T. Lawnton. Qualitative navigation for mobile robots. *Artificial Intelligence*, 44:305–360, 1990.

[22] Maja J. Mataric. Integration of representation into goal-driven behaviour-based robots. *IEEE Transactions on Robotics and Automation*, 8(3):304–312, June 1992.

[23] Richard M. Murray and S. Shankar Sastry. Nonholonomic motion planning: Steering using sinusoids. *IEEE Transactions on Automatic Control*, 38(5):700–716, May 1993.

[24] Ulrich Nehmzow. Animal and robot navigation. *Robotics and Autonomous Systems*, 25:71–81, 1995.

[25] C. Samson and K. Ait-Abderrahim. Feedback control of a non-holonomic wheeled cart in cartesian space. In *International Conference on Robotics and Automation*, pages 1136–1141, Sacramento, California, USA, April 1991. IEEE.

[26] M. V. Srinivasan. How insects infer range from visual motion. In F. A. Miles and J. Wallman, editors, *Visual Motion and its Role in the Stabilization of Gaze*. Elsivier Science Publishers, New York, 1993.

[27] M. V. Srinivasan, J. S. Chahl, K. Weber, S. Venkatesh, M. G. Nagle, and S. W. Zhang. Robot navigation inspired by principles of insect vision. *Robotics and Autonomous Systems*, 26:203–216, 1999.

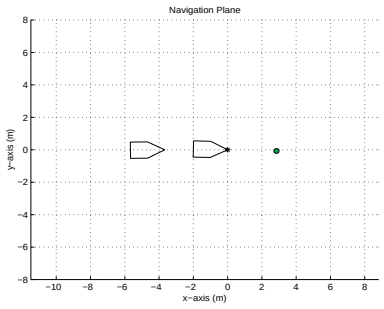
[28] Kane Usher, Peter Corke, and Peter Ridley. A camera as a polarised light compass: Preliminary experiments. In *Proceedings of the 2001 Australian Conference on Robotics and Automation*, Sydney, Australia, November 2001. published via CDROM.

[29] Keven Weber, Svetha Venkatesh, and M. V. Srinivasan. An insect based approach to robotic homing. In *International Conference on Pattern Recognition*, pages 297–299, Brisbane, Australia, 1998. IEEE Computer Society Press.

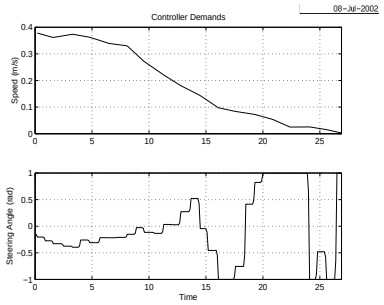
[30] Keven Weber, Svetha Venkatesh, and Mandyam Srinivasan. Insect-inspired robotic homing. *Adaptive Behavior*, 7(1):65–97, 1999.

[31] R. Wehner, B. michel, and P. Antonsen. Visual navigation in insects: Coupling og egocentric and geocentric information. *Journal of Experimental Biology*, 199:129–140, 1996.

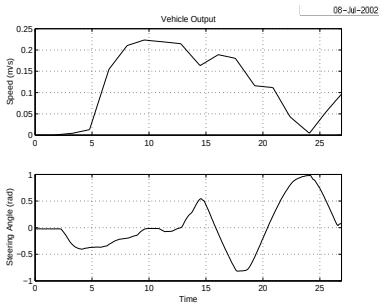
[32] R. Wehner and S. Wehner. Insect navigation: Use of maps orAriadne's thread. In *Ethology, Ecology and Evolution 2*, pages 27–48, 1990.



(a) Initial and final (goal) positions.



(b) Demanded control signals.



(c) Vehicle speed and steer angle.

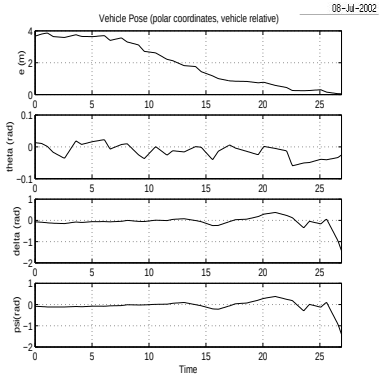


Figure 14: *Results from the first experiment. At the target pose, (shown as* $(x,y,\theta) = (0,0,0)$*, the single landmark is located at approximately* $2.85\,\mathrm{m}$ *directly in front of the robot. The robot was then driven to the starting pose of* $(x,y,\theta) = (-4,0,0)$ *at which time the IALV/Lyapunov controller was activated. The robot then 'homed' back to the starting pose with minimal residual error.*

Visual Servoing Meets the Real World

Danica Kragic and Henrik I. Christensen
Centre for Autonomous Systems
Royal Institute of Technology
{danik,hic}@nada.kth.se

1 Introduction

Visual servoing is by now a mature research field in the sense that the basic paradigms largely have been defined as for example seen in the tutorial by [1]. Over the last few years a variety of visual servoing refinements have been published. It is, however, characteristic that relatively few truly operational systems have been reported. Given that the basic methods are available, why is it that so few *systems* are reported in the literature. Some examples have been reported. Good examples include the work on vision for highway driving by [2] and the work on vision based assembly by [3] and [4], and the work on vision guided welding by [5]. It is characteristic for these applications that they are implemented in setting where the scenario and its constituent objects are well defined and can be model at great detail prior to the execution of the task.

1.1 Issues in servoing

For use of visual servoing in a general setting such as a regular domestic environment it is not only necessary to perform the basic servoing based on a set of well defined features, the full process involves:

1. Recognition of the object to interact with
2. Detection of the features to be used for servoing
3. Initial alignment of model to features (pose estimation or initial data association)
4. Servoing using the defined features

In addition it must be considered that most visual servoing techniques have a limited domain of operation, i.e. some techniques are well suited for point based estimation, while others are well suited for detailed alignment at close range. For many realistic tasks there is consequently a need to decompose the task into several visual servoing stages to allow robust operation. I.e. something like

1. Point based visual servoing to achieve initial alignment
2. Approximate alignment using 2.5D or homography based servoing
3. Model based servoing to achieve high quality alignment

Depending on the task at hand these tasks might be executed in parallel (i.e. in car driving the coarse alignment is used for longer term planning and model estimation, while the model based techniques is used for short term correction – model based correction), or in sequence (initial alignment, that can be used for initialization of the intermediate model, which in term provides an estimate for model initialization). In this paper we will in particular study the manipulation of objects in a domestic setting and consequently the sequential model will be used.

1.2 Outline

In the next section (2) the basic processes involved in end-to-end visual servoing are outlined. To achieve robustness it is necessary to utilize a mixture of object models (object models, appearance models, geometric models) and fusion of multiple cues to compensate for variations in illumination, viewpoint, and surface texture, this is discussed in section 3. The use of the presented methodology is illustrated for the task of manipulating a variety of objects in a living room setting as reported in section 4. Finally a few observations and issues for future research are discussed in section 5.

2 Processes in Vision based Grasping

2.1 Detection/Recognition

The object to be manipulated is first recognized using the view-based SVM (support vector machine) system presented in [6]. For execution of a task such as "pickup coke can on the dinner table" the robot will navigate to the living room, and position itself with the camera directed towards the dinner table. The recognition step will then deliver regions in the image that may represent the object of interest. The recognition step delivers the image position and approximate size of the image region occupied by the object. This information is required by the tracking system to allow initialization through use of a *window of attention*. Initially the recognition step is carried out at a distance of 2-3 meters to allow use of a field of view that is likely to be large enough include the object of interest. Once a valid image region has been identified the position and scale information is delivered to the image based servoing step.

2.2 Object Approach

While approaching the object (getting within reach), we want to keep it in the field of view or even centered of the image. This implies that we have to estimate the position/velocity of the object in the image and use this information to control the mobile platform.

Our tracking algorithm employs the four step *detect–match–update–predict loop*, Fig. 1. The objective here is to track a part of an image (a region) between frames. The image position of its center is denoted with $\mathbf{p} = [x\,y]^T$. Hence, the state is $\mathbf{x} = [\,x\,y\,\dot{x}\,\dot{y}\,]^T$ where a piecewise constant white acceleration model is used [7]:

$$\begin{aligned} \mathbf{x}_{k+1} &= \mathbf{F}\mathbf{x}_k + \mathbf{G}\mathbf{v}_k \\ \mathbf{z}_k &= \mathbf{H}\mathbf{x}_k + \mathbf{w}_k \end{aligned} \tag{1}$$

where $\mathbf{v}_k$ is a zero–mean white acceleration sequence, $\mathbf{w}_k$ is measurement noise and

$$\mathbf{F} = \begin{bmatrix} 1 & 0 & \Delta T & 0 \\ 0 & 1 & 0 & \Delta T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},\ \mathbf{G} = \begin{bmatrix} \frac{\Delta T^2}{2} & 0 \\ 0 & \frac{\Delta T^2}{2} \\ \Delta T & 0 \\ 0 & \Delta T \end{bmatrix},\ \mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \tag{2}$$

For prediction and estimation, the $\alpha - \beta$ filter is used, [7]:

$$\begin{aligned} \hat{\mathbf{x}}_{k+1|k} &= \mathbf{F}_k \hat{\mathbf{x}}_k \\ \hat{\mathbf{z}}_{k+1|k} &= \mathbf{H}\hat{\mathbf{x}}_{k+1|k} \\ \hat{\mathbf{x}}_{k+1|k+1} &= \hat{\mathbf{x}}_{k+1|k} + \mathbf{W}[\mathbf{z}_{k+1} - \hat{\mathbf{z}}_{k+1|k}] \end{aligned} \tag{3}$$

with

$$\mathbf{W} = \begin{bmatrix} \alpha & 0 & \frac{\beta}{\Delta T} & 0 \\ 0 & \alpha & 0 & \frac{\beta}{\Delta T} \end{bmatrix}^T$$

where α and β are determined using steady–state analysis The tracking of the object is used to direct the arm so

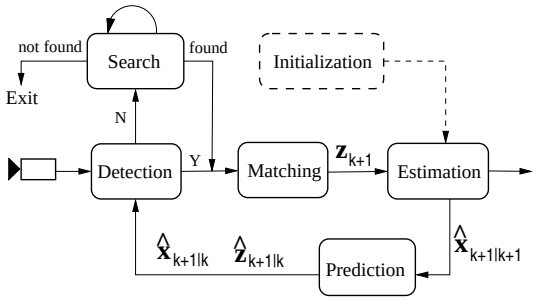


Figure 1: A schematic overview of the tracking system.

as to ensure that the TPC is pointing at the object while it is being approached. The standard image Jacobian is being used for specification of the control of the arm. Once the object occupies more than, an object, specified portion of the image the servoing is terminated and control is handed over to the coarse alignment control for pre-grasping and gripper - object alignment.

2.3 Coarse alignment

The basic idea is the use of a reference position for the camera. It is assumed to be known how the object can be grasped from the reference position (see Fig. 2).

Using a stored image taken from the reference position, the manipulator should be moved in such a way that the current camera view is gradually changed to match the stored reference view. Accomplishing this for general scenes is difficult, but a robust system can be made under the assumption that the objects are piecewise planar [8], i.e. a homography models the transformation between the current and the reference view. The scheme implemented requires three major components – initialization, servoing and tracking. Each component exploits the planarity of the tracked object. For initialization, a wide baseline matching algorithm [8] is employed to establish point correspondences between the

Figure 2: The left image shows the gripper in a reference position while the right image shows the camera in another position and situation. Servoing is then used to move the camera to the reference position relative to the object.

current and the reference image. The point correspondences enable the computation of the homography H relating the two views. Assuming known internal camera parameters, the homography matrix is used in the "2.5D" servoing algorithm of Malis et al [9]. Finally, as initialization is computationally expensive, matches are established in consecutive images using tracking. This is accomplished by making a prediction of the new homography H' relating the current and the reference images, given the arm odometry between frames and the homography H relating the reference image and the previous image [8]. H' is then used in a guided search for new correspondences between the current image and the reference image [10].

2.4 Grasp alignment

Although suitable for a number of tasks, previous approaches lacks the ability to estimate position and orientation (pose) of the object. In terms of manipulation, it is usually required to accurately estimate the pose of the object to, for example, allow the alignment of the robot arm with the object or to generate a feasible grasp and execute a grasp of the object. Using prior knowledge about the object, a special representation can further increase the robustness of the tracking system. Along with commonly used CAD models (wire–frame models), view– and appearance–based representations may be employed [11].

A recent study of human visually guided grasps in situations similar to that typically used in visual servoing control, [12] has shown that the human visuomotor system takes into account the three dimensional geometric features rather than the two dimensional projected image of the target objects to plan and control the required movements. These computations are more complex than those typically carried out in visual servoing systems and permit humans to operate in large range of environments.

We have therefore decided to integrate both appearance based and geometrical models to solve different steps of a manipulation task. Many similar systems use manual pose initialization where the correspondence between the model and object features is given by the user, (see [13] and [5]). Although there are systems where this step is performed automatically [14], [15] proposed approaches are time consuming and not appealing for real–time applications. One additional problem, in our case, is that the objects to be manipulated by the robot are highly textured (see Fig. 3) and therefore not suited for matching approaches based on, for example, line features [16], [17], [18].

After the object has been recognized and its position in the image is known, an appearance based method is employed to estimate its initial pose. The method we have implemented has been initially proposed in [19] where just three pose parameters have been estimated and used to move a robotic arm to a predefined pose with respect to the object. Compared to our approach, where the pose is expressed relative to the camera coordinate system, they express the pose relative to the current arm configuration, making the approach unsuitable for robots with different number of degrees of freedom.

Compared to the system proposed in [20], where the network has been entirely trained on simulated images, we use real images for training where no particular background was considered. As pointed out in [20], the illumination conditions (as well as the background) strongly affect the performance of their system and these can not be easily obtained with simulated images. In addition, the idea of projecting just the wire–frame model to obtain training images can not be employed in our case due to the objects' texture. The system proposed in [16] also employs a feature based approach where lines, corners and circles are used to provide the initial pose estimate. However, this initialization approach is not applicable in our case since, due to the geometry and textural properties, these features are not easy to find with high certainty.

Compared to the system proposed in [20], where the network has been entirely trained on simulated images, we use real images for training where no particular background was considered. As pointed out in [20], the illumination conditions (as well as the back-

Figure 3: Some of the objects we want robot to manipulate.

ground) strongly affect the performance of their system and these can not be easily obtained with simulated images. In addition, the idea of projecting just the wire–frame model to obtain training images can not be employed in our case due to the objects' texture. The system proposed in [16] also employs a feature based approach where lines, corners and circles are used to provide the initial pose estimate. However, this initialization approach is not applicable in our case since, due to the geometry and textural properties, these features are not easy to find with high certainty.

Our model based tracking system is presented in Fig. 4. Here, a wire-frame model of the object is used to estimate the pose and the velocity of the object in each frame. The model is used during the *initialization* step where the initial pose the object relative to the camera coordinate system is estimated. The main loop starts with a *prediction* step where the state of the object is predicted using the current pose (velocity, acceleration) estimate and a motion model. The visible parts of the object are then projected into the image (*projection and rendering* step). After the *detection* step, where a number of features are extracted in the vicinity of the projected ones, these new features are *matched* to the projected ones and used to estimate the new pose of the object. Finally, the calculated pose is input to the *update* step. The systems has the ability to cope with partial occlusions of the object, and to successfully track the object even in the case of significant rotational motion.

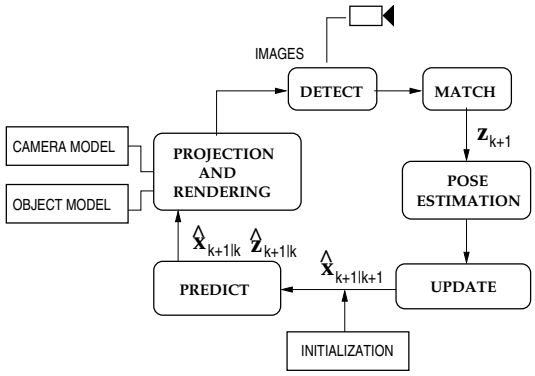


Figure 4: Block diagram of the model based tracking system.

2.4.1 Prediction and Update

The system state vector consists of three parameters describing translation of the target, another three for orientation and an additional six for the velocities:

$$\mathbf{x} = \left[X, Y, Z, \phi, \psi, \gamma, \dot{X}, \dot{Y}, \dot{Z}, \dot{\phi}, \dot{\psi}, \dot{\gamma}\right] \tag{4}$$

where ϕ, ψ and γ represent roll, pitch and yaw angles [21]. The following piecewise constant white acceleration model is considered [7]:

$$\begin{aligned} \mathbf{x}_{k+1} &= \mathbf{F}\mathbf{x}_k + \mathbf{G}\mathbf{v}_k \\ \mathbf{z}_k &= \mathbf{H}\mathbf{x}_k + \mathbf{w}_k \end{aligned} \tag{5}$$

where $\mathbf{v}_k$ is a zero–mean white acceleration sequence, $\mathbf{w}_k$ is the measurement noise and

$$\mathbf{F} = \begin{bmatrix} \mathbf{I}_{6\times6} & \Delta T\mathbf{I}_{6\times6} \\ \mathbf{0} & \mathbf{I}_{6\times6} \end{bmatrix}, \quad \mathbf{G} = \begin{bmatrix} \frac{\Delta T^2}{2}\mathbf{I}_{6\times6} \\ \Delta T\mathbf{I}_{6\times6} \end{bmatrix}, \quad \mathbf{H} = \begin{bmatrix} \mathbf{I}_{6\times6} \mid \mathbf{0} \end{bmatrix} \tag{6}$$

For prediction and update, the $\alpha - \beta$ filter is used:

$$\begin{aligned} \hat{\mathbf{x}}_{k+1|k} &= \mathbf{F}_k\hat{\mathbf{x}}_k \\ \hat{\mathbf{z}}_{k+1|k} &= \mathbf{H}\hat{\mathbf{x}}_{k+1|k} \\ \hat{\mathbf{x}}_{k+1|k+1} &= \hat{\mathbf{x}}_{k+1|k} + \mathbf{W}[\mathbf{z}_{k+1} - \hat{\mathbf{z}}_{k+1|k}] \end{aligned} \tag{7}$$

Here, the pose of the target is used as measurement rather than image features, as commonly used in the

literature (see, for example, [22], [14]). An approach similar to the one presented here is taken in [18]. This approach simplifies the structure of the filter which facilitates a computationally more efficient implementation. In particular, the dimension of the matrix **H** does not depend on the number of matched features in each frame but it remains constant during the tracking sequence.

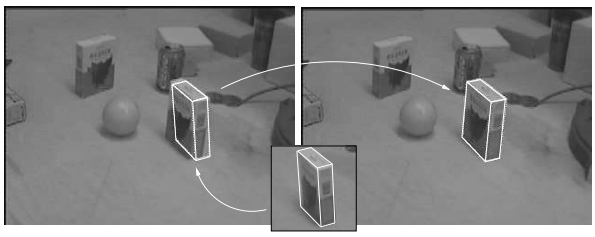


Figure 5: On the left is the initial pose estimated using PCA approach. On the right is the pose obtained by local refinement method.

2.4.2 Initial Pose Estimation

Initialization step uses the ideas proposed in [19]. During training, each image is projected as a point to the eigen-space and the corresponding pose of the object is stored with each point. For each object, we have used 96 training images (8 rotations for each angle on 4 different depths). One of the reasons for choosing this low number of training images is the workspace of the PUMA560 robot used. Namely, the workspace of the robot is quite limited and for our applications this discretization was satisfactory. To enhance the robustness with respect to variations in intensity, all images are normalized. At this stage, the size of the training samples is 100×100 pixels color images. The training procedure takes about 3 minutes on a Pentium III 550 running Linux.

Given an input image, it is first projected to the eigenspace. The corresponding parameters are found as the closest point on the pose manifold. Now, the wire-frame model of the object can be easily overlaid on the image. Since a low number of images is used in the training process, pose parameters will not accurately correspond to the input image. Therefore, a local refinement method is used for the final fitting, see Fig. 5. The details are given in the next section.

During the training step, it is assumed that the object is approximately centered in the image. During task execution, the object can occupy an arbitrary part of the image. Since the recognition step delivers the image position of the object, it is easy to estimate the offset of the object from the image center and compensate for it. This way, the pose of the object relative to the camera frame can also be arbitrary.

An example of the pose initialization is presented in Fig. 6. Here, the pose of the object in the training image (far left) was: *X=-69.3, Y=97.0, Z=838.9,* ϕ*=21.0,* ψ*=8.3 and* γ*=-3.3*. After the fitting step the pose was: *X=55.9, Y=97.3, Z=899.0,* ϕ*=6.3,* ψ*=14.0 and* γ*=1.7* (far right), showing the ability of the system to cope with significant differences in pose parameters. In addition,

2.4.3 Detection and matching

When the estimate of the object's pose is available, the visibility of each edge feature is determined and internal camera parameters are used to project the model of the object onto the image plane. For each visible edge, a number of image points is generated along the edge. So called tracking nodes are assigned at regular intervals in image coordinates along the edge direction. The discretization is performed using the Bresenham algorithm [23]. After that, a search is performed for the maximum discontinuity (nearby edge) in the intensity gradient along the normal direction to the edge. The edge normal is approximated with four directions: $\{-45, 0, 45, 90\}$ degrees, see Fig. 7.

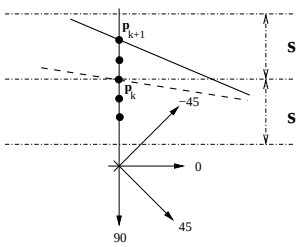


Figure 7: Determining normal displacements for points on a contour in consecutive frames.

In each point $\mathbf{p}_i$ along a visible edge, the perpendicu- $_i$ to the nearby edge is determined using a one–dimensional search. The search region is denoted by $\{S_i^j, j \in [-s, s]\}$. The search starts at the projected model point $\mathbf{p}_i$ and the traversal continues simultaneously in opposite search directions until the first local maximum is found. The size of the search region s

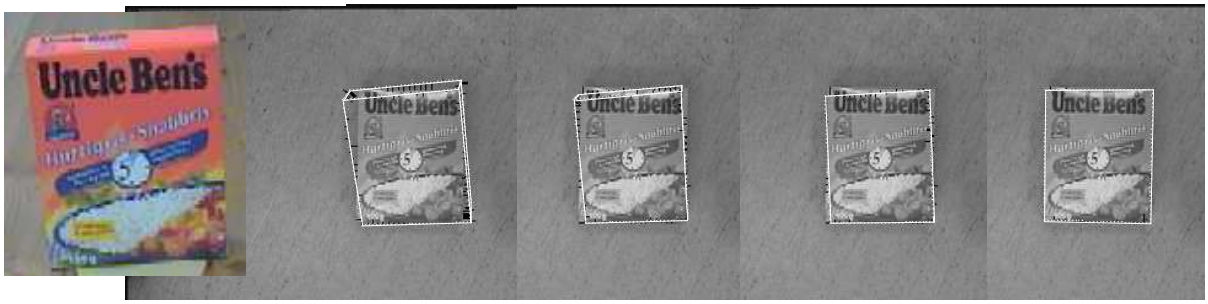


Figure 6: Training image used to estimate the initial pose (far left) followed by the intermediate images of the fitting step.

is adaptive and depends on the distance of the objects from the camera.

After the normal displacements are available, the method proposed in [5] is used. Lie group and Lie algebra formalism are used as the basis for representing the motion of a rigid body and pose estimation. Implementation details can be found in [24].

2.5 Grasping

The ability of a system to generate both a *feasible* and a *stable* grasp adds to its robustness. By a *feasible* grasp, a kinematically feasible grasp is considered, that is, given the pose of the object the estimated configuration of the manipulator and the end-effector are kinematically valid. By a *stable* grasp, a grasp for which the object will not twist or slip relative to the end-effector.

We have integrated our model–based vision system with a model-based grasp planning and visualization system called GraspIt! (see [25] and [26] for more details). To plan a grasp, grasp planners need to know the pose of objects in a workspace. For that purpose a vision system may be employed which, in addition, may be used to determine how a robot hand should approach, grasp, and move an object to a new configuration. A synergistic integration of a grasp simulator and a model based visual system is described. These work in concert to i) find an object's pose, ii) generate grasps and movement trajectories, and iii) visually monitor the execution of a servoing task [27].

An example grasp is presented in Fig. 8. It demonstrates a successfully planned and executed stable grasp of an L-shaped object. After the first image of the scene is acquired, the estimated pose of the object is sent to GraspIt, which aids the user in selecting an appropriate grasp. After a satisfactory grasp is planned, it is executed by controlling both the robot arm and the robot hand. As can be seen in the figures, the grasping is performed in two steps. The first step sends the arm to the vicinity of the object, i.e., to the pose form which the grasp will be performed. After the arm is positioned, the hand may be closed.

3 Robustness

To achieve robustness in realistic situations it is necessary to deploy multiple features and integrate these into a joint representation. To fusion of multiple cues here deploys a voting strategy to enable use of weak model for the integration of cues.

3.1 Multiple Cues

3.1.1 Voting

Voting, in general, may be viewed as a method to deal with n input data objects, c_i, having associated votes/weights w_i (n input data–vote pairs (c_i, w_i)) and producing the output data–vote pair (y, v) where y may be one of the c_i's or some mixed item. Hence, voting combines information from a number of sources and produces outputs which reflect the consensus of the information.

The reliability of the results depends on the information carried by the inputs and, as we will see, their number. Although there are many voting schemes proposed in the literature, mean, majority and plurality voting are the most common ones. In terms of voting, a visual cue may be motion, color, disparity. Mathematically, a cue is formalized as a mapping from an action space, **A**, to the interval [0,1]:

$$c \; : \; \mathbf{A} \rightarrow [0,1] \tag{8}$$

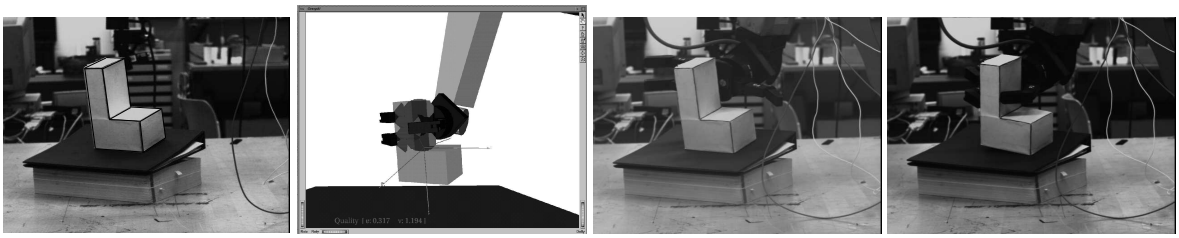


Figure 8: Grasping of the L-shaped object: The pose of the object is estimated in the camera coordinate frame (the estimated pose is overlaid in black). The pose is transformed to the manipulator coordinate system using the off–line estimated camera/robot transformation. The estimated pose is propagated to GraspIt (first row, right) where a stable grasp is planed (the measures of the quality of the grasp are shown in the lower left corner of the image: e=0.317 and v=1.194). The planned grasp is used to position the manipulator and the hand.

This mapping assigns a *vote* or a preference to each action $a \in \mathbf{A}$, which may, in the context of tracking, be considered as the position of the target. These votes are used by a *voter* or a *fusion center*, $\delta(\mathbf{A})$. Based on the ideas proposed in [28], [29], we define the following voting scheme:

Definition 3. 1 - ***Weighted Plurality Approval Voting*** *For a group of homogeneous cues, $\boldsymbol{C} = \{c_1, \ldots, c_n\}$, where n is the number of cues and O_{c_i} is the output of a cue i, a weighted plurality approval scheme is defined as:*

$$\delta(a) = \sum_{i=1}^{n} w_i \, O_{c_i}(a) \tag{9}$$

where the most appropriate action is selected according to:

$$a' = \text{argmax}\{\delta(a) | a \in \boldsymbol{A}\} \tag{10}$$

3.1.2 Visual Cues

The cues considered in the integration process are:

Correlation - The standard sum of squared differences (SSD) similarity metric is used and the position of the target is found as the one giving the lowest dissimilarity score:

$$SSD(u,v) = \sum_{n}\sum_{m}[I(u+m, v+n) - T(m,n)]^2 \tag{11}$$

where $I(u,v)$ and $T(u,v)$ represent the grey level values of the image and the template, respectively.

Color - It has been shown in [30] that efficient and robust results can be achieved using the Chromatic Color space. Chromatic colors, known as "pure" colors without brightness, are obtained by normalizing each of the components by the total sum. Color is represented by r and g component since blue component is both the noisiest channel and it is redundant after the normalization.

Motion - Motion detection is based on computation of the temporal derivative and image is segmented into regions of motions and regions of inactivity. This is estimated using image differencing:

$$M[(u,v),k] = \mathcal{H}[|I[(u,v),k] - I[(u,v),k-1]| - \Gamma] \tag{12}$$

where Γ is a fixed threshold and $\mathcal{H}$ is defined as:

$$\mathcal{H}(x) = \begin{cases} 0 & : \quad x \leq 0 \\ x & : \quad x > 0 \end{cases} \tag{13}$$

Intensity Variation - In each frame, the following is estimated for all $m \times m$ (details about m are given in Section 3.1.4) regions inside the tracked window:

$$\sigma^2 = \frac{1}{m^2}\sum_{u}\sum_{v}[I(u,v) - \bar{I}(u,v)]^2 \tag{14}$$

where $\bar{I}(u,v)$ is the mean intensity value estimated for the window. For example, for a mainly uniform region, low variation is expected during tracking. On the other hand, if the region was rich in texture large variation is expected. The level of texture is evaluated as proposed in [31].

3.1.3 Weighting

In (Eq. 9) it is defined that the outputs from individual cues should be weighted by w_i. Consequently, the

reliability of a cue should be estimated and its weight determined based on its ability to track the target. The reliability can be determined either i) *a–priori* and kept constant during tracking, or ii) estimated during tracking based on cue's success to estimate the final result or based on how much it is in agreement with other cues. In our previous work the following methods were evaluated [24]:

1. Uniform weights - Outputs of all cues are weighted equally: $w_i = 1/n$, where n is the number of cues.

2. Texture based weighting - Weights are preset and depend on the spatial content of the region. For a highly textured region, we use: color (0.25), image differencing (0.3), correlation (0.25), intensity variation (0.2). For uniform regions, the weights are: color (0.45), image differencing (0.2), correlation (0.15), intensity variation (0.2). The weights were determined experimentally.

3. One-step distance weighting - Weighting factor, w_i, of a cue, c_i, at time step k depends on the distance from the predicted image position, $\hat{\mathbf{z}}_{k|k-1}$. Initially, the distance is estimated as

$$d_i = ||\mathbf{z}_k^i - \hat{\mathbf{z}}_{k|k-1}|| \tag{15}$$

and errors are estimated as

$$e_i = \frac{d_i}{\sum_{i=1}^{n} d_i}. \tag{16}$$

Weights are than inversely proportional to the error with $\sum_{i=1}^{n} w_i = 1$.

4. History-based distance weighting - Weighting factor of a cue depends on its overall performance during the tracking sequence. The performance is evaluated by observing how many times the cue was in an agreement with the rest of the cues. The following strategy is used:

a) For each cue, c_i, examine if $||\mathbf{z}_k^i - \mathbf{z}_k^j|| < d_T$ where $i, j = 1, \ldots, n$ and $i \neq j$. If this is true, a_{ij}=1, otherwise a_{ij}=0. Here, a_{ij}=1 means that there is an agreement between the outputs of cues i and j at that voting cycle and d_T represents a distance threshold which is set in advance.

b) Build the $(n-1)$ value set for each cue: $c_i : \{a_{ij} | j = 1, \ldots, n \text{ and } i \neq j\}$ and estimate sum $s_i = \sum_{j=1}^{n} a_{ij}$.

c) The accumulated values during N tracking cycles, $S_i = \sum_{k=1}^{N} s_i^k$, indicate how many times a cue, c_i, was in the agreement with other cues. Weights are then simply proportional to this value:

$$w_i = \frac{S_i}{\sum_{i=1}^{n} S_i} \quad \text{with} \quad \sum_{i=1}^{n} w_i = 1 \tag{17}$$

3.1.4 Implementation

We have investigated two approaches where voting is used for: i) *response fusion*, and ii) *action fusion*. The first approach makes the use of "raw" responses from the employed visual cues in the image which also represents the action space, **A**. Here, the response is represented either by a binary function (yes/no) answer, or in the interval [0,1] (these values are scaled between [0,255] to allow visual monitoring). The second approach uses a different action space represented by a *direction* and a *speed*, see Fig. 9. Compared to the first approach, where the position of the tracked region is estimated, this approach can be viewed as estimating its velocity. Again, each cue votes for different actions from the action space, **A**, which is now the velocity space.

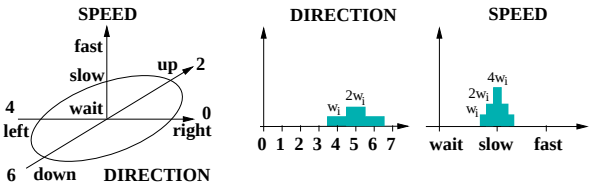


Figure 9: A schematic overview of the *action fusion* approach: the desired direction is (*down and left)* with a (*slow*) speed. The use of bins represents a *neighborhood voting scheme* which ensures that slight differences between different cues do not result in an unstable classifi cation.

Initialization

According to Fig. 1, a tracking sequence should be initiated by *detecting* the target object. If a recognition module is not available, another strategy can be used. In [32] it is proposed that *selectors* should be employed which are defined as heuristics that selects regions possibly occupied by the target. When the system does not have definite state information about the target it should actively search the state space to find it. Based on this ideas, color and image differences (or foreground motion) may be used to detect the target in the first image. Again, if a recognition module is not available, these two cues may also be used in cases where the target has

either i) left the field of view, or ii) it was occluded for a few frames. Our system searches the whole image for the target and once the target enters the image, tracking is regained.

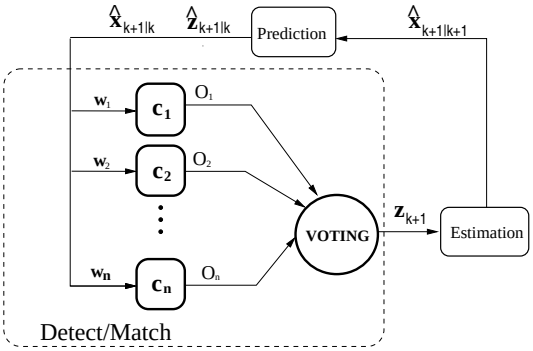


Figure 10: A schematic overview of the *response fusion* approach.

Response Fusion Approach

After the target is located, a template is initialized which is used by the correlation cue. In each frame, a color image of the scene is acquired. Inside the window of attention the response of each cue, denoted O_i, is evaluated, see Fig. 10. Here, $\mathbf{x}$ represents a position:

Color - During tracking, all pixels whose color falls in the pre–trained color cluster are given value between [0, 255] depending on the distance from the center of the cluster:

$$\begin{aligned} &0 \leq O_{color}(\mathbf{x},k) \leq 255 \quad \text{with} \\ &\mathbf{x} \in [\hat{\mathbf{z}}_{k|k-1} - 0.5\mathbf{x}_w,\ \hat{\mathbf{z}}_{k|k-1} + 0.5\mathbf{x}_w] \end{aligned} \tag{18}$$

where $\mathbf{x}_w$ is the size of the window of attention.

Motion- Using (Eq. 12) and (Eq. 13) with $\Gamma = 10$, image is segmented into regions of motion and inactivity:

$$\begin{aligned} &0 \leq O_{motion}(\mathbf{x},k) \leq 255 - \Gamma \quad \text{with} \\ &\mathbf{x} \in [\hat{\mathbf{z}}_{k|k-1} - 0.5\mathbf{x}_w,\ \hat{\mathbf{z}}_{k|k-1} + 0.5\mathbf{x}_w] \end{aligned} \tag{19}$$

Correlation - Since the correlation cue produces a single position estimate, the output is given by:

$$\begin{aligned} &O_{SSD}(\mathbf{x},k) = 255e^{\left(-\frac{\bar{\mathbf{x}}^2}{2\sigma^2}\right)} \quad \text{with} \quad \sigma = 5 \\ &\mathbf{x} \in [\mathbf{z}_{SSD} - 0.5\mathbf{x}_w,\ \mathbf{z}_{SSD} + 0.5\mathbf{x}_w], \\ &\bar{\mathbf{x}} \in [-0.5\mathbf{x}_w,\ 0.5\mathbf{x}_w] \end{aligned} \tag{20}$$

where the maximum of the Gaussian is centered at the peak of the SSD surface. The size of the search area depends on the estimated velocity of the region.

Intensity variation - The response of this cue is estimated according to (Eq. 14). If a low variation is expected, all pixels inside an $m \times m$ region are given values (255-σ). If a large variation is expected, pixels are assigned σ value directly. The size $m \times m$ of the subregions which are assigned same value depends on the size of the window of attention with $n = 0.2\mathbf{x}_w$. Hence, for a 30 $\times$ 30 pixels window of attention, m=6. The result is presented as follows:

$$\begin{aligned} &0 \leq O_{var}(\mathbf{x},k) \leq 255 \quad \text{with} \\ &\mathbf{x} \in [\hat{\mathbf{z}}_{k|k-1} - 0.5\mathbf{x}_w,\ \hat{\mathbf{z}}_{k|k-1} + 0.5\mathbf{x}_w] \end{aligned} \tag{21}$$

Response Fusion

The estimated responses are integrated using (Eq. 9):

$$\delta(\mathbf{x},k) = \sum_i^n w_i O_i(\mathbf{x},k) \tag{22}$$

However, (Eq. 10) can not be directly used for selection, as there might be several pixels with same number of votes. Therefore, this equation is slightly modified:

$$\delta'(\mathbf{x},k) = \begin{cases} 1: & \text{if } \delta(\mathbf{x},k) \text{ is } \operatorname{argmax}\{\delta(\mathbf{x}',k)|\mathbf{x}' \\ & \in [\hat{\mathbf{z}}_{k|k-1} - 0.5\mathbf{x}_w, \hat{\mathbf{z}}_{k|k-1} + 0.5\mathbf{x}_w]\} \\ 0: & \text{otherwise} \end{cases} \tag{23}$$

Finally, the new measurement $\mathbf{z}_k$ is given by the mean value (first moment) of $\delta'(\mathbf{x},k)$, i.e., $\mathbf{z}_k = \bar{\delta'}(\mathbf{x},k)$.

Action Fusion Approach

Here, the action space is defined by a direction d and speed s, see Fig. 9. Both the direction and the speed are represented by histograms of discrete values where the direction is represented by eight values, see Fig. 11:

$$\begin{aligned} &\text{LD}\begin{bmatrix}-1\\1\end{bmatrix}, \text{L}\begin{bmatrix}-1\\0\end{bmatrix}, \text{LU}\begin{bmatrix}-1\\-1\end{bmatrix}, \text{U}\begin{bmatrix}0\\-1\end{bmatrix}, \\ &\text{RU}\begin{bmatrix}1\\-1\end{bmatrix}, \text{R}\begin{bmatrix}1\\0\end{bmatrix}, \text{RD}\begin{bmatrix}1\\1\end{bmatrix}, \text{D}\begin{bmatrix}0\\1\end{bmatrix} \end{aligned} \tag{24}$$

with L-left, R-right, D-down and U-up. Speed is represented by 20 values with 0.5 pixel interval which means that the maximum allowed displacement between successive frames is 10 pixels (this is easily made adaptive

based on the estimated velocity). There are two reasons for choosing just eight values for the direction: i) if the update rate is high or the inter–frame motion is slow, this approach will still give a reasonable accuracy and hence, a smooth performance, and ii) by keeping the voting space rather small there is a higher chance that the cues will vote for the same action. Accordingly, each cue will vote for a desired direction and a desired speed. As presented in Fig. 9 a *neighborhood voting scheme* is used to ensure that slight differences between different cues do not result in an unstable classification. (Eq. 3) is modified so that:

$$\mathbf{H} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \text{ and } \mathbf{W} = \begin{bmatrix} \alpha\Delta T & 0 & \beta & 0 \\ 0 & \alpha\Delta T & 0 & \beta \end{bmatrix}^T \quad (25)$$

In each frame, the following is estimated for each cue:
Color - The response of the color cue is first estimated according to (Eq. 18) followed by:

$$\mathbf{a}_{color}(k) = \frac{\sum_{\mathbf{x}} O_{color}(\mathbf{x},k)\mathbf{x}(k)}{\sum_{\mathbf{x}} \mathbf{x}(k)} - \hat{\mathbf{p}}_{k|k-1} \quad (26)$$
$$\text{with} \quad \mathbf{x} \in [\hat{\mathbf{p}}_{k|k-1} - 0.5\mathbf{x}_w,\ \hat{\mathbf{p}}_{k|k-1} + 0.5\mathbf{x}_w]$$

where $\mathbf{a}_{color}(k)$ represents the desired action and $\hat{\mathbf{p}}_{k|k-1}$ is the predicted position of the tracked region. Same approach is used to obtain $\mathbf{a}_{motion}(k)$ and $\mathbf{a}_{var}(k)$.
Correlation - The minimum of the SSD surface is used as:

$$\mathbf{a}_{SSD}(k) = \underset{\mathbf{x}}{\text{argmin}}(SSD(\mathbf{x},k)) - \hat{\mathbf{p}}_{k|k-1} \quad (27)$$

where the size of the search area depends on the estimated velocity of the tracked region.

Action Fusion After the desired action, $\mathbf{a}_i(k)$, for a cue is estimated, the cue produces the votes as follows:

$$\begin{aligned} &\text{direction } d_i = \mathcal{P}(\text{sgn}(\mathbf{a}_i)), \\ &\text{speed } s_i = ||\mathbf{a}_i|| \end{aligned} \quad (28)$$

where $\mathcal{P} : \mathbf{x} \rightarrow \{0,1,\ldots,7\}$ is a scalar function that maps the two–dimensional direction vectors (see (Eq. 24)) to one–dimensional values representing the bins of the direction histogram. Now, the estimated direction, d_i, and the speed, s_i, of a cue, c_i, with a weight, w_i, are used to update the direction and speed of the histograms according to Fig. 9 and (Eq.9). The new measurement is then estimated by multiplying the actions from each histogram which received the maximum number of votes according to (Eq. 10):

$$\mathbf{z}_k = \mathcal{S}(\underset{d}{\text{argmax}} HD(d)) \underset{s}{\text{argmax}} HS(s) \quad (29)$$

where $\mathcal{S} : x \rightarrow \{\begin{bmatrix} -1 \\ 0 \end{bmatrix}, \ldots, \begin{bmatrix} -1 \\ 1 \end{bmatrix}\}$. The update and prediction steps are then performed using (Eq. 25) and (Eq. 3). The reason for choosing this particular representation instead of simply using a weighted sum of first moments of the responses of all cues is, as it has been pointed out in [29], that arbitration via vector addition can result in commands which are not satisfactory to any of the contributing cues.

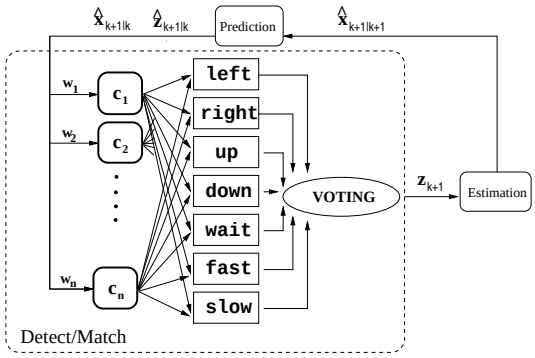


Figure 11: A schematic overview of the *action fusion* approach.

4 Experiments

Here, we consider the problem of real manipulation in a realistic environment - a living room. Similarly to the previous example, we assume that a number of predefined grasps is given and suitable grasp is generated depending on the current pose of the object. The experiment shows a XR4000 platform with a hand-mounted camera. The task is to approach the dinner table including several objects. The robot is instructed to pick up a package of rice having an arbitrary placement. Here, Distributed Control Architecture [33] is used for integration of the different methods into a fully operational system. The system includes i) recognition, ii) image based tracking, iii) initial pose estimation, iv) hand alignment for grasping, v) grasp execution and vi) delivery of object. The details of the system implementation are unfortunately beyond this paper. The results of a mission with the integrated system are outlined below.

The sequence in Fig. 12 shows the robot starting at the far end of the living room, moving towards a point

where a good view of the dinner table can be obtained. After the robot is instructed to pick up the rice package, it recognizes it and locates in the scene. After that, the robot moves closer to the table keeping the rice package centered in the image. Finally, the gripper is aligned with the object and grasping is performed. The details about the alignment can be found in [34].

5 Summary

A key problem of deployment of manipulation in realistic setting is integration of a multitude of methods into a unified framework that manages the different phases of grasping from i) identification, ii) approach, iii) pregrasping, and iv) grasp execution. In this paper we have presented a system in which all of these aspects are integrated into an operational system. Each of the participating processes have been designed to accommodate handling of a large variety of situations. No single visual servoing technique is well suited for operation of a large range variations, consequently techniques with varying operation characteristics have been combined to provide an end-to-end system that can operate in situations with realistic complexity.

At present the system is tailored to the task in terms of sequential execution of the different steps in the servoing process. To achieve smooth transition between the different phases it is naturally of interest to provide a control framework for integrated modelling of the control, which will be a key problem to be studies as part of future work. In addition generality across a range of different application domains will be studies.

Acknowledgment

This research has been carried out by the Intelligent Service Robot team at the Centre for Autonomous Systems. The work has involved P. Jensfelt, A. Oreback, D. Tell, M. Strandberg, A. Miller, D. Roobaert, L. Petersson and M. Zillich. Their contribution to the integration of the overall system and numerous discussions is gratefully acknowledged.

The research has been sponsored by the Swedish Foundation for Strategic Research, this support is gratefully acknowledged.

References

[1] S. Hutchinson, G. Hager, and P. Corke, "A tutorial on visual servo control," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 5, pp. 651–670, 1996.

[2] E. Dickmanns, "Vehicles capable of dynamic vision: a new breed of technical beings?," *Artificial Intelligence*, vol. 103, pp. 49–76, August 1998.

[3] H. Kollnig and H. Nagel, "3D pose estimation by directly matching polyhedral models to gray value gradients," *International Journal of Computer Vision*, vol. 23, no. 3, pp. 282–302, 1997.

[4] G. Hirzinger, M. Fischer, B. Brunner, R. Koeppe, M. Otter, M. Gerbenstein, and I. Schäfer, "Advanced in robotics: The DLR experience," *International Journal of Robotics Research*, vol. 18, pp. 1064–1087, November 1999.

[5] T. Drummond and R. Cipolla, "Real-time tracking of multiple articulated structures in multiple views," in *Proceedings of the 6th European Conference on Computer Vision, ECCV'00*, vol. 2, pp. 20–36, 2000.

[6] D. Roobaert, *Pedagogical Support Vector Learning: A Pure Learning Appr oach to Object Recognition*. PhD thesis, Computatioal Vision and Active Perception Laboratory (CVAP), Royal Institute of Technology, Stockholm, Sweden, May 2001.

[7] Y. Bar-Shalom and Y. Li, *Estimation and Tracking:Principles, techniques and software*. Artech House, 1993.

[8] D. Tell, *Wide baseline matching with applications to visual servoing*. PhD thesis, Computational Vision and Active Perception Laboratory (CVAP), Royal Institute of Technology, Stockholm, Sweden, 2002.

[9] E. Malis, F. Chaumette, and S. Boudet, "Positioning a coarse-calibrated camera with respect to an unknown object by 2-1/2-d visual servoing," in *ICRA*, pp. 1352–1359, 1998.

[10] P. Pritchett and A. Zisserman, "Wide baseline stereo matching," in *ICCV*, pp. 863–869, 1998.

[11] L. Bretzner, *Multi-scale feature tracking and motion estimation*. PhD thesis, CVAP, NADA, Royal Institute of Technology, KTH, 1999.

[12] Y. Hu, R. Eagleson, and M. Goodale, "Human visual servoing for reaching and grasping: The role of 3D geometric features," in *Proceedings of the IEEE International Conference on Robotics and Automation, ICRA'99*, vol. 3, pp. 3209–3216, 1999.

Figure 12: An experiment where the robot moves up to the table, recognizes the rice box, approaches it, picks it up and hands it over to the user.

[13] N. Giordana, P. Bouthemy, F. Chaumette, and F. Spindler, "Two-dimensional model-based tracking of complex shapes for visual servoing tasks," in *Robust vision for vision-based control of motion* (M. Vincze and G. Hager, eds.), pp. 67–77, IEEE Press, 2000.

[14] V. Gengenbach, H.-H. Nagel, M. Tonko, and K. Schäfer, "Automatic dismantling integrating optical flow into a machine-vision controlled robot system," in *Proceedings of the IEEE International Conference on Robotics and Automation, ICRA'96*, vol. 2, pp. 1320–1325, 1996.

[15] D. Lowe, *Perceptual Organisation and Visual Recognition*. Robotics: Vision, Manipulation and Sensors, Dordrecht, NL: Kluwer Academic Publishers, 1985. ISBN 0-89838-172-X.

[16] M. Vincze, M. Ayromlou, and W. Kubinger, "An integrating framework for robust real-time 3D object tracking," in *Proceedings of the First International Conference on Computer Vision Systems, ICVS'99*, pp. 135–150, 1999.

[17] D. Koller, K. Daniilidis, and H. Nagel, "Model-based object tracking in monocular image sequences of road traffi c scenes," *International Journal of Computer Vision*, vol. 10, no. 3, pp. 257–281, 1993.

[18] P. Wunsch and G. Hirzinger, "Real-time visual tracking of 3D objects with dynamic handling of occlusion," in *Proceedings of the IEEE International Conference on Robotics and Automation, ICRA'97*, vol. 2, pp. 2868–2873, 1997.

[19] S. Nayar, S. Nene, and H. Murase, "Subspace methods for robot vision," *IEEE Transactions on Robotics and Automation*, vol. 12, pp. 750–758, October 1996.

[20] P. Wunsch, S. Winkler, and G. Hirzinger, "Real-Time pose estimation of 3D objects from camera images using neural networks," in *Proceedings of the IEEE International Conference on Robotics and Automation, ICRA'97*, vol. 3, (Albuquerque, New Mexico), pp. 3232–3237, April 1997.

[21] J. Craig, *Introduction to Robotics: Mechanics and Control*. Addison Wesley Publishing Company, 1989.

[22] E. Dickmanns and V. Graefe, "Dynamic monocular machine vision," *Machine Vision nad Applications*, vol. 1, pp. 223–240, 1988.

[23] J. Foley, A. van Dam, S. Feiner, and J. Hughes, eds., *Computer graphics - principles and practice*. Addison-Wesley Publishing Company, 1990.

[24] D. Kragic, *Visual Servoing for Manipulation: Robustness and Integration Issues*. PhD thesis, Computational Vision and Active Perception Laboratory (CVAP), Royal Institute of Technology, Stockholm, Sweden, 2001.

[25] A. Miller and P. Allen, "Examples of 3D grasp quality computations," in *Proceedings of the of the 1999 IEEE International Conference on Robotics and Automation*, pp. 1240–1246, 1999.

[26] A. T. Miller and P. Allen, "Graspit!: A versatile simulator for grasping analysis," in *To appear in Proceedings of the of the 2000 ASME International Mechanical Engineering Congress and Exposition*, 2000.

[27] D. Kragić, A. Miller, and P. Allen, "Realtime tracking meets online grasp planning," in *IEEE International Conference on Robotics and Automation, ICRA'01*, 2001.

[28] D. Blough and G. Sullivan, "Voting using predispositions," *IEEE Transactions on reliability*, vol. 43, no. 4, pp. 604–616, 1994.

[29] J. Rosenblatt and C. Thorpe, "Combining multiple goals in a behavior-based architecture," in *IEEE International Conference on Intelligent Robots and Systems, IROS'95*, vol. 1, pp. 136–141, 1995.

[30] H. Christensen, D. Kragic, and F. Sandberg, "Vision for interaction," in *Intelligent Robotic Systems* (G. Hager and H. Christensen, eds.), Lecture notes in Computer Science, Springer, 2001. to be published.

[31] J. Shi and C. Tomasi, "Good features to track," in *Proceedings of the IEEE Computer Vision and Pattern Recognition, CVPR'94*, pp. 593–600, 1994.

[32] K. Toyama and G. Hager, "Incremental focus of attention for robust visual tracking," in *Proceedings of the Computer Society Conference on Computer Vision and Pattern Recognition, CVPR'96*, pp. 189 –195, 1996.

[33] L. Petersson, D. Austin, and H. Christensen, "DCA: A Distributed Control Architecture for Robotics," in *Proc. IEEE International Conference on Intelligent Robots and Systems IROS'2001*, vol. 4, (Maui, Hawaii), pp. 2361–2368, October 2001.

[34] D. Tell, *Wide baseline matching with applications to visual servoing*. Phd thesis, Computational Vision and Active Perception Laboratory (CVAP), Royal Institute of Technology, Stockholm, Sweden, march 2002.

Switching Approaches to Visual Servo Control

Seth Hutchinson, Nicholas Gans

ngans@uiuc.edu, seth@uiuc.edu
Dept. of Electrical and Computer Engineering
The Beckman Institute for Advanced Science and Technology
University of Illinois at Urbana-Champaign
Urbana, IL, USA

Abstract

There exist numerous methods for visual servo control of a robot system, each with particular strengths and weaknesses. We investigate the use of dynamic switching approaches to visual servo control, in which a high-level decision maker selects from two low-level visual servo controllers. We will discuss two such switching controllers, one of which uses two PBVS systems and one of which utilizes an IBVS and a PBVS system. We present experimental results and will introduce a discussion of stability for each system.

1 Introduction

Visual servoing has proven to be a highly effective means to control a robot manipulator through the use of visual data. It provides a high degree of accuracy using even simple camera systems, robustness in the face of signal error and uncertainty of system parameters.

Visual servo methods have classically been divided into two camps, Position Based Visual Servoing (PBVS) and Image Based Visual Servoing (IBVS). PBVS uses camera data to estimate the robot's current pose; the error signal is typically the difference between the current pose and a desired pose stored in memory, and the control state is expressed in 3D Cartesian space.

IBVS came about in the mid-eighties, with the introduction of the image Jacobian [1, 2]. The error signal is measured as the difference between the current image and a desired image stored in memory. The state exists in the two dimensional image, and the image Jacobian maps the error signal to 3D Cartesian velocities for the end effector.

IBVS proved extremely popular, and for the next several years, researchers proposed a variety of image-based control systems that relied either explicitly or implicitly on the linearization embodied by the image Jacobian (e.g., [3, 4]). In the late nineties, Chaummette outlined a number of problems that cannot be solved using the traditional local linearized approaches to visual servo control [5]. This resulted in a variety of visual servo systems which used the image Jacobian linearization of IBVS for specific degrees of freedom, and 3D techniques exemplified in PBVS for the remainder [6, 7, 8, 9].

Rather than combining systems, another approach is the use of hybrid dynamical systems, i.e., systems that comprise a set of continuous subsystems along with a rule that switches between them [10, 11]. Dynamic switching systems can potentially increase stability, or move systems to their region of attraction that might otherwise fail, though often at the cost of increased complexity and computation. We have begun investigating the feasibility of using dynamic switching systems for visual servoing.

In Section 2, we introduce the fundamentals behind IBVS and PBVS systems and present concepts methods utilized in these systems. Section 3 will provide an introduction to the theory behind switched systems, including a discussion of stability. Finally, in Sections 4 and 5 we will present dynamic switching systems we have developed, along with mathematical investigations into stability and experimental results.

2 Visual Servoing

2.1 IBVS

In IBVS systems, the control exists in the image space. In the common case of a camera mounted on the robot end effector, the motion of a two-dimensional feature point $\mathbf{f} = [u, v]^T$ in the image is related to the velocity screw of the end effector $\dot{\mathbf{r}} = [T_x, T_y, T_z, \omega_x, \omega_y, \omega_z]^T$ by the relation

$$\dot{\mathbf{f}} = J_{im}(u, v, z)\dot{\mathbf{r}}, \tag{1}$$

where $\mathbf{J}_{im}$ is the image Jacobian defined as

$$\mathbf{J}_{im} = \begin{bmatrix} \frac{\lambda}{z} & 0 & -\frac{u}{z} & -\frac{uv}{\lambda} & \frac{\lambda^2+u^2}{\lambda} & -v \\ 0 & \frac{\lambda}{z} & -\frac{v}{z} & \frac{-\lambda^2-v^2}{\lambda} & \frac{uv}{\lambda} & u \end{bmatrix} \tag{2}$$

in which λ is the focal length of the camera. Derivations of this can be found in a number of references including

[12, 13, 14]. The simplest approach to IBVS is to merely use (1) to construct the control law

$$\mathbf{u} = \mathbf{K} J_{im}^{-1}(\mathbf{r}) \dot{\mathbf{f}} \tag{3}$$

in which $\dot{\mathbf{f}}$ is the desired feature motion on the image plane, $\mathbf{K}$ is a (typically diagonal) gain matrix, and $\mathbf{u} = \dot{\mathbf{r}}$ is the control input, an end-effector velocity. Since images are acquired at discrete time instants, one typically defines $\dot{\mathbf{f}} = \mathbf{f}^* - \mathbf{f}$, in which the superscript * denotes the goal value for a quantity. Thus, given a goal image (either previously captured or calculated), IBVS systems zero the error in the image space, bringing the features of the current image coincident with those the goal image.

This approach assumes that the image Jacobian is square and nonsingular, and when this is not the case, a generalized inverse, $\mathbf{J}_{im}^{+}$, is used [12].

Since (3) essentially represents a gradient descent on the feature error, when this control law is used, feature points tend to move in straight lines to their goal positions (though the values of the gain matrix will have a strong effect on the feature point velocities). This provides desirable performance in the image space, but as first reported by Chaumette[5] it can lead to extraneous motions of the end effector in 3D Cartesian space. Taken to the extreme, these motions can lead to singular positions for the robot or singularities in the image Jacobian, leading to task failure.

For faster visual servo systems, equation (3) can be expanded to detail the robot dynamics, which can affect the visual feedback loop. Define a vector of the robot joint positions by $\mathbf{q}$ and the respective joint velocities by $\dot{\mathbf{q}}$. Define $\mathbf{M}$ as the symmetric positive definite manipulator inertia matrix, $\mathbf{C}$ as a matrix describing the effects of centripetal and Corialis torques, and $\tilde{\mathbf{J}} = \mathbf{J}_{im}\mathbf{J}_{rob}$ as the product of the image Jacobian and robot manipulator Jacobian. We now define two positive definite gain matrices $\mathbf{K}_p$ and $\mathbf{K}_v$. The control loop can now be described by:

$$\mathbf{K}_v \dot{\mathbf{q}} + \mathbf{M}\ddot{\mathbf{q}} + \mathbf{C}\dot{\mathbf{q}} = \mathbf{K}_p \tilde{\mathbf{J}}^{-1} \dot{\mathbf{f}} \tag{4}$$

Clearly then, $[\mathbf{q}, \dot{\mathbf{q}}]^{\mathrm{T}} = [\mathbf{q_d}, \mathbf{0}]$, where $\mathbf{q_d}$ is the desired joint configuration, is an equilibrium point for this system.

Indeed, IBVS has been proven asymptotically stable under certain conditions related to accuracy of the camera calibration and the lack of singularities in the robot [3]. Of particular note is a proof by Kelly, et al., [15], using Lyapunov's direct method. Kelly, et al, basically develop a total energy energy function of the form

$$V = \frac{1}{2}\dot{\mathbf{q}}^T \mathbf{M} \dot{\mathbf{q}} + \frac{1}{2}\dot{\mathbf{f}}^T \mathbf{K}_p \dot{\mathbf{f}} \tag{5}$$

Which is the sum of the kinetic and potential energies.

Taking the time derivative and making some substitutions (see [15] for specifics) , Kelly, et al., arrive at

$$\dot{V} = -\dot{\mathbf{q}}^T \mathbf{K}_v \dot{\mathbf{q}}. \tag{6}$$

Since $\mathbf{K}_v$ is defined to be positive definite, the system is shown to be asymptotically stable.

2.2 PBVS

In PBVS systems, the control task remains in 3D Cartesian space. Numerous techniques exist for accurately estimating the position and orientation of end effector from a set of image features using iterative [16, 17] or Kalman filter techniques [18]. The general technique in PBVS is to capture the image of a known set of features, and estimate the pose of the end effector in relation to the features. This estimated pose is compared against a desired pose, and a trajectory then generated to eliminate the difference between the two poses.

In contrast to IBVS, PBVS systems generate a gradient descent in 3D space. The end effector will typically follow the shortest path to the goal position. This, however, leads to large motions of the features in the image space. This can cause the feature points to leave the field of view, resulting in system failure.

PBVS has also been proven to be asymptotically stable if certain conditions are met. Recently, Deng, at al, proved asymptotic stability through Lyapunov techniques based on the proof of Kelly, et al, for IBVS [19]. They describe the following control function,

$$\mathbf{K_v}\dot{\mathbf{q}} + \mathbf{M}\ddot{\mathbf{q}} + \mathbf{C}\dot{\mathbf{q}} = \mathbf{K_p}\mathbf{J}_{rob}^{-1}\dot{\mathbf{r}} \tag{7}$$

where $\mathbf{J}_{rob}$ is the robot manipulator Jacobian and $\dot{\mathbf{r}}$ is the distance between the current and goal poses. This leads to the candidate Lyapunov function

$$V = \frac{1}{2}\dot{\mathbf{q}}^T \mathbf{M} \dot{\mathbf{q}} + \frac{1}{2}\dot{\mathbf{r}}^T \mathbf{K}_p \dot{\mathbf{r}} \tag{8}$$

which has precisely the same time derivative as Kelly, et al, arrived at for IBVS,

$$\dot{V} = -\dot{\mathbf{q}}^T \mathbf{K}_v \dot{\mathbf{q}}. \tag{9}$$

3 Switching Control

The theory of hybrid dynamical systems, i.e., systems that comprise a number of continuous subsystems and a high-level decision maker that switches between them, has received considerable attention in the control theory community. In this paper, we are concerned with the specific topic of switched control systems. In particular, we present a controller that uses a high level decision maker to select from two visual servo controllers.

In general, a hybrid dynamical system can be represented by the differential equation

$$\dot{x} = f_\sigma(x) : \sigma \in \{1..n\} \tag{10}$$

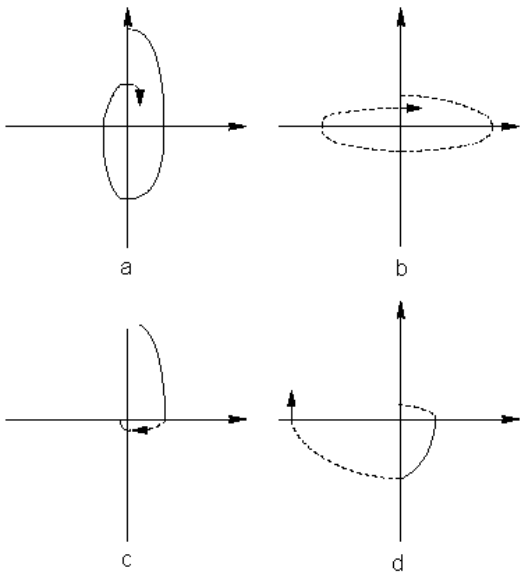


Figure 1: *trajectories of switched systems*

where f_σ is a collection of n distinct functions. For our purposes, it is convenient to explicitly note that the switching behavior directly affects the choice of the control input u

$$\dot{x} = f(x, u_\sigma) : \sigma \in \{1, 2\}. \tag{11}$$

In our system, each visual servo controller provides a velocity screw, $u = [T_x, T_y, T_z, \omega_x, \omega_y, \omega_z]^T$, and a switching rule determines which is used as the actual control input at each control cycle.

The stability of a switched system is not insured by the stability of the individual controllers. Indeed, a collection of stable systems can become unstable when inappropriately switched. As an illustration, Figure 1 (from [20]) show trajectories for two asymptotically stable subsystems in (a) and (b). A set of switches resulting in a stable system is shown in (c), while a series of switches resulting in an unstable system are shown in (d). To insure stability, it must be proved that all switches from one system to another cannot result in an unstable output, regardless of the time of the switch or the state of the system. Stability of switched systems can be proven using Lyapunov's direct method under certain conditions [20, 21]. Generally this requires establishing a common Lyapunov function which works for all subsystems. Alternately, one can establish a family of Lyapunov functions for the systems such that at each switch, the value of the function at the end of that interval is less than the value of the function of the interval that proceeded it, as illustrated for a one dimensional family of two functions in Figure 2. In our initial investigations reported here, we have not yet formally proven the stability of our system.

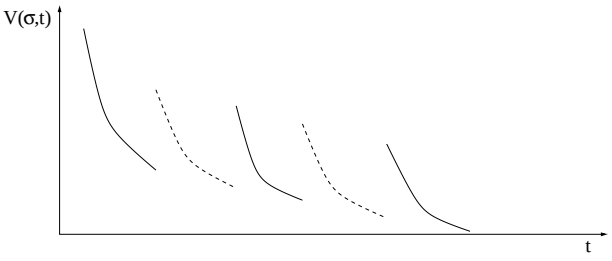


Figure 2: *stable family of Lyapunov function*

We have, however, performed extensive empirical evaluations that demonstrate the efficacy of our approach, and some of these are presented below.

4 Homography/Affine Switched System Controller

The first dynamic switched system presented here was introduced in [22]. That paper focused primarily on initial simulated results, while here we will present early experimental results and a discussion of stability issues.

The system is composed of two subsystems which, given two views of the same set of features, estimate the translation and change in orientation of the camera. In the sense that these methods use a goal and current image, they certainly fall into the realm of IBVS systems. However, a complete translation and rotation is estimated, thus delivering a trajectory relating the current pose and the goal pose, which is the output of a PBVS system. In addition, these systems tend to perform similarly to PBVS systems in regards to optimal motion of the end effector at the expense of sub-optimal feature trajectories. Additionally, knowledge of the geometry of the features can be exploited to make the systems much more accurate. Thus we will typically consider these methods to be PBVS methods.

4.1 Homography Based Controller

The homography method exploits the epipolar constraints between two images of planar feature points. The homography matrix has been used previously for visual servoing in [6, 8] to control a restricted set of degrees of freedom. We however, use it to generate a control for all degrees of freedom.

Define $\tilde{\mathbf{f}}^*$, $\tilde{\mathbf{f}}$, as the homogeneous coordinates in two images of a set of 3D points lying on a plane π. These are related by

$$\tilde{\mathbf{f}}^* = \mathbf{H}\tilde{\mathbf{f}} \tag{12}$$

where $\mathbf{H}$ is the calibrated homography matrix. As shown in [23, 24], $\mathbf{H}$ can be decomposed as

$$\mathbf{H} = \mathbf{R}(\mathbf{I}_3 - \frac{\mathbf{t}\mathbf{n}^T}{d}) \tag{13}$$

where $\mathbf{I}_3$ is a 3×3 identity matrix and $\mathbf{R}$ and $\mathbf{t}$ are the rotation matrix and translation vector, respectively, relating the two camera views. The parameter $\mathbf{n}$ is the the normal of the plane π and describes the orientation of π with respect to the current camera view; d is the distance from the current camera origin to the plane π. We calculate the vector $\mathbf{T} = [T_x, T_y, T_z]^T = \hat{d}\,\mathbf{t}$, where $\hat{d}$ is an estimate of d. Given knowledge of the geometry of the feature point locations it is possible to accurately estimate $\hat{d}$ and so determine $\mathbf{t}$ to the proper scale. From the rotation matrix $\mathbf{R}$, we extract the roll, pitch and yaw angles, which we use as $\omega_x, \omega_y, \omega_z$ to obtain the velocity screw $u = k[T_x, T_y, T_z, \omega_x, \omega_y, \omega_z]$ in which k is a scalar gain constant, or a 6×6 gain matrix..

Of the numerous methods to calculate $\mathbf{H}$, we have used a linear solution since visual servoing, in general, requires quicker calculations than iterative methods may provide. Decomposing the homography as in (13) is not a trivial exercise and generally cannot be solved to a unique solution. Additional information or views are required to select form multiple solutions. In visual servoing, an extra view is necessary at the first iteration, as the results of previous iterations can be used for all later calculations.

4.2 Affine-Approximation Controller

For camera motions that do not involve rotation about the camera x- or y- axes, the initial and goal images will be related by an affine transformation. While this is a constrained set of motions, it is common in many situations, such as aligning camera with a component on a conveyer belt.

Define $\mathbf{f}^*$, $\mathbf{f}$, as the calibrated pixel coordinates of two points in the image plane. Then these points are related by the affine transformation

$$\begin{aligned} \mathbf{f}^* &= \mathbf{A}\mathbf{f} + \mathbf{b} \\ &= \begin{bmatrix} 1 & a \\ 0 & 1 \end{bmatrix} \begin{bmatrix} s_1 & 0 \\ 0 & s_2 \end{bmatrix} \begin{bmatrix} C_\theta & -S_\theta \\ S_\theta & C_\theta \end{bmatrix} \begin{bmatrix} f_x \\ f_y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix} \end{aligned} \quad (14)$$

in which C_θ and S_θ denote respectively $\cos\theta$ and $\sin\theta$; f_x and f_y are image point coordinates; a, s_i, and θ describe the skew, scale, and rotation respectively; and $\mathbf{b}$ is the translation. Both $\mathbf{A}$ and $\mathbf{b}$ can be obtained by solving a linear system of equations, and QR decomposition can then be used to determine a, s_i, and θ.

$$\mathbf{f}^* = \mathbf{R}\mathbf{Q}\mathbf{f} + \mathbf{b} = \begin{bmatrix} r_{11} & r_{12} \\ 0 & r_{22} \end{bmatrix} \begin{bmatrix} q_{21} & q_{12} \\ q_{11} & q_{22} \end{bmatrix} \begin{bmatrix} f_x \\ f_y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix} \quad (15)$$

The $\mathbf{Q}$ matrix is a permutation of the rotation matrix in (14), and rotation θ about the camera z axis equals arcsine(q_{21}). During an affine transformation, the rotations about the $x-$ and y axes are, by definition, zero. The r_{11} and r_{22} of (15) respectively equal the s_1 and s_2 parameters of (14) and provide z translation to scale. Translation along the $x-$ and $y-$ axes are defined to scale in the vector $\mathbf{b}$. Multiplying the scaled translations by a depth estimate will provide true values. Again, knowledge of the feature point geometry will allow for the depth estimate to be accurately derived.

Given a, s_i, θ, t_x, t_y we again have the position and orientation relating the initial and camera goal positions. This controller provides the velocity screw $u = k[t_x, t_y, s_2, 0, 0, \theta]$ where k is a gain constant or matrix. Note that if there is no rotation about the $x-$ or $y-$axes, we will have $s_1 = s_2$.

In general, two images are unlikely to be perfectly related by an affine transformation. However, in the absence of signal noise the approximation is often fairly accurate. In our scheme, it is left to the switching rule to determine when it is appropriate to use this controller.

4.3 Methodology of System Switching

To date, we have explored three switching rules. The first is deterministic, and is based on our evaluation of the performance of each controller under various conditions. The other two switching rules include a nondeterministic element.

Comparison of the Two Controllers. The major strength of the homography-based controller in our system is that it is the only controller capable of handling general motions which include rotation about the x and $y-$axes. If the camera motion does not involve such rotations, the two approaches have similar performance. However, in the presence of noise, the affine method is much more accurate. We conducted a series of Monte Carlo tests in which both systems performed an identical affine motion under the effects of increasing white noise. The homography-based method typically had an error in the pose estimation that was fifty times greater than the affine approach, and error in the total rotation was almost fifteen times greater.

With regard to speed, the affine approximation controller requires far fewer calculations per iteration. However, both systems are capable of operating at above the typical camera frame rate of 30Hz.

Discussion of Switching Rules. In this section we describe the three switching rules.

Deterministic Switching. The degree to which a camera motion can be approximated by an affine transformation can be determined by examining the rotation matrix returned by solving the homography. In particular, if the motion can be approximated by an affine solution, then the rotation about the camera $x-$ or $y-$ axes is necessarily very small. We take the RMS value of the rotations about $x-$ and $y-$ and compare them to a threshold value, if the amount of rotation is less than this value, we select the affine solution.

The threshold value is arbitrary, a lower value will force

the system to choose the affine method for larger rotations about the $x-$ and $y-$axes. We have found that, the affine method can zero the error for any motion with a rotation about y or x less then 0.2°. Thus the rotation matrix is calculated at each iteration, and if RMS(ω_x, ω_y) is greater than that amount, the homographic method is chosen to reduce that rotation.

Random Switching. The previous switching rule captures the full abilities of the homographic method, but fails to fully utilize the benefits of the affine system in the case of noise. Our second switching rule is to randomly select, with equal likelihood, one controller at each iteration. Random switching has been used in network control systems to handle task routing [25]. A random choice is the simplest decision mechanism, and in a stable hybrid system will result in an averaging of the performance of all systems. However, this may be a dangerous choice for a potentially unstable system.

Biased Random Switching. A compromise can be struck between the two switching methods. At each iteration the affine approximation controller is selected with a probability that is a monotonic function of RMS(ω_x, ω_y), since, as noted earlier, a larger value for RMS(ω_x, ω_y) indicates that the homographic method is more suitable. However, it still allows for the affine method to occasionally occur. Likewise, the affine method is much more likely to be chosen should the rotation about the $x-$ and $y-$ axes be small, but the homographic method can still occasionally be selected to reduce minor rotations about the $x-$ and $y-$ axes.

4.4 Experimental Results

Our experiments were performed using a camera mounted on the end effector of a PUMA 560 robot. The camera is a Sony VFW-V500, which has 640×480 color pixel display. The lens focal length was 14.4mm. The feature points consisted of four color dots on a black sheet. The image was thresholded in RBG space to locate the center points of each dot. This provided 4 co-planar feature points.

Experiments presented here consisted of a goal image wherein the camera optical axis was parallel to the normal of the feature point plane, and approximately 0.75 meters away. The initial error image was an oblique view, much closer to the feature point plane with heavy rotation about the camera $y-$ axis, moderate rotation about camera the z-axis, and translation along all axes. The two images can be seen in Figure 3.

We first explored the deterministic switching method. Since there is a great deal of rotation about that the camera $y-$ axis, we expect that it will use the homographic method for the majority of iterations, and switch to using the affine method when the y axis rotation has become very small.

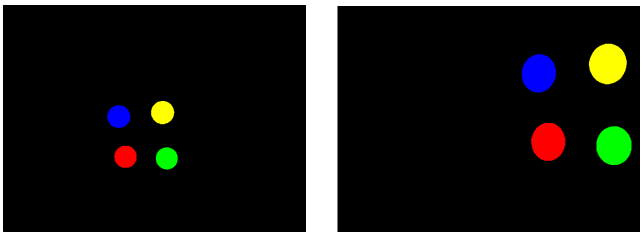


Figure 3: *Goal and Initial Image*

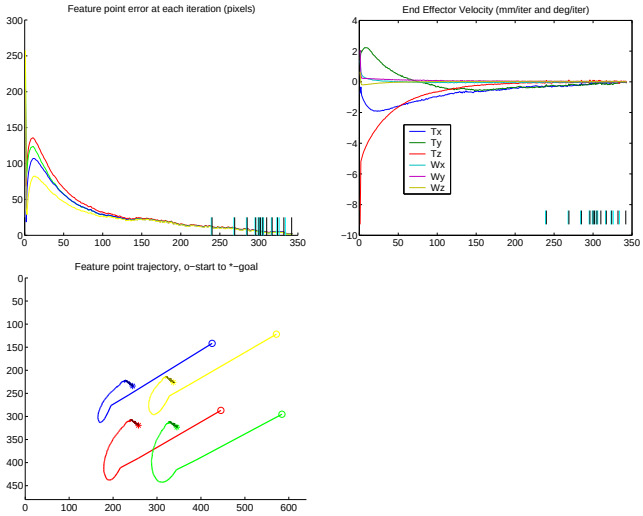


Figure 4: *Deterministic Switching*

Figure 4 shows the feature point error, the velocity screw of the vector and the recorded feature point position for each iteration. In the first two graphs, small lines on the bottom of the graph indicate a switch has taken place; a black line indicates the homography method is being used for the following iteration, while a cyan line indicated the affine method will be used. We do see the homographic method used for almost two thirds of the iterations, at which point it switches between the affine solution and the homographic method as the amount of $y-$ axis rotation becomes negligible. The third graphs shows the trajectory the point followed in the image plane. Portions of the lines with a black shadow indicate when the affine method is being used.

Figure 5 shows the results of random switching. All the measured values are much more chaotic. The feature point error tends to be greater, as does the magnitude of the of the velocity screw variables. However, the error is still zeroed in approximately the same amount of time and we also avoid the extremely large initial motion which the homographic method introduced in the deterministic case.

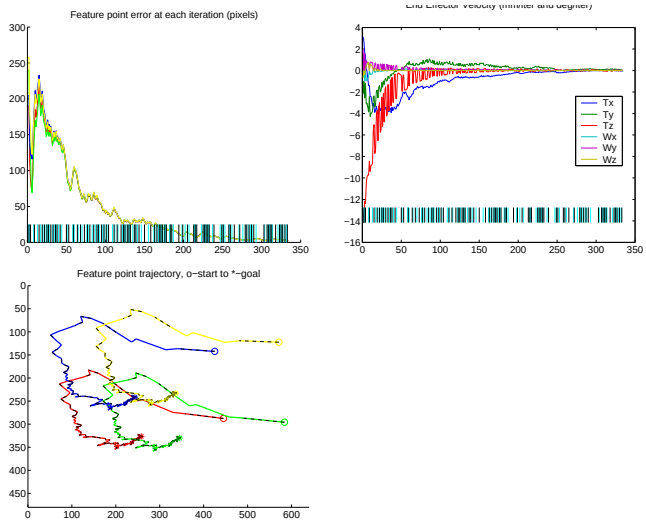


Figure 5: *Random Switching*

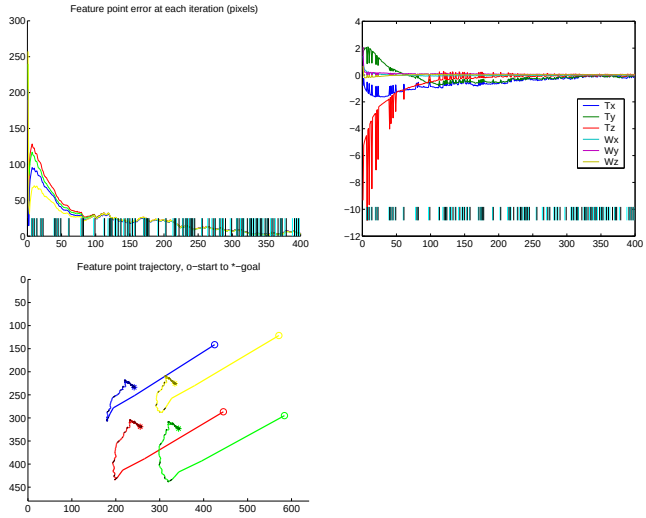


Figure 6: *Probabilistic Switching*

Finally, Figure 6 shows the probabilistic system. The feature point error is similar to the deterministic method, though it tends to be slightly smaller. Likewise the velocity screw tends to have similar shape and size to the deterministic method. The system is, however, not able to zero the error as quickly as the other switching methods.

4.5 Discussion of Stability

While we have not yet established a proof of stability, we will present a brief discussion of what we feel are the main topics. Both systems are PBVS systems, and thus both can use the common Lyapunov function (8), reproduced below.

$$V = \frac{1}{2}\dot{\mathbf{q}}^T\mathbf{M}\dot{\mathbf{q}} + \frac{1}{2}\dot{\mathbf{r}}^T\mathbf{K}_p\dot{\mathbf{r}} \tag{16}$$

However, for the affine system there is not guaranteed to be a solution which takes the potential energy function to zero, rather it will take it to a local minimum.

5 IBVS/PBVS Switched System Controller

The second switched system presented here is a combination of IBVS and PBVS methods. This system was designed in hopes of using the strengths of each system to counterbalance the weaknesses of the other.

5.1 IBVS and PBVS systems

The IBVS system was discussed in section 2.1. At the time of publication, we have not successfully created an iterative or Kalman filter based PBVS system. Thus for experimental results we have elected to use the homography based method described in Section 4.1. As noted there, the the homography based system has performance characteristic of a PBVS system, and should be indicative of how other PBVS systems should perform in a dynamic switching system.

5.2 Methodology of System Switching

We have explored the the same types of switching rules for this system as for the Homography/Affine system. Namely a deterministic rule, a random choice of systems, and a probabilistic rule.

Comparison of the Two Controllers. As discussed in Sections 2.1 and 2.2, IBVS systems exhibit optimal performance in the image space. Feature points will tend along straight lines or least distance trajectories towards the goal configuration. However, this can result in non ideal performance in the 3D Cartesian space as unnecessary motions are performed, failure can result from the robot reaching a singular position or reaching the end of the task space.

In contrast, PBVS systems perform optimally in the Cartesian space, moving the camera straight to the goal position. However feature points can be lost during the process, resulting in system failure.

Deterministic Switching. We simply attempt to avoid the weaknesses of both systems by switching when the current system is approaching a problematic state. We determine a threshold level for how far the feature points will be allowed to stray from the center of the image, as well as a threshold on the distance we will allow the camera to move from the feature points. At each iteration, we compare each switching paramater to its treshold. If the feature points are closer to their threshold we select IBVS to bring them towards goal; if the camera distance

is closer to it's threshold we select PBVS to bring the camera towards goal.

Random Switching. Again, a simple binary random variable is used to select between the two systems with equal probability.

Biased Random Switching. The deterministic levels discussed above are now used as part of probabilistic function used to determine the next system used. The farther the camera is from the image plane, the more likely the system is to choose the PBVS method. Likewise, the farther the feature points are from the image center, the more likely a the system is to choose to use IBVS. The probability will be unity at either threshold level defined for the deterministic switching.

Like the determinstic system, we see how close each switching paramater is to it's threshhold. We focus on the more pressing paramter, and divide it by its threshold. We then create a random number between 0 and 1. If the random number is smaller than the the divided parameter, ue we choose the method we would have chosen under deterministic switching. If the random umber is larger, we choose the alternate method.

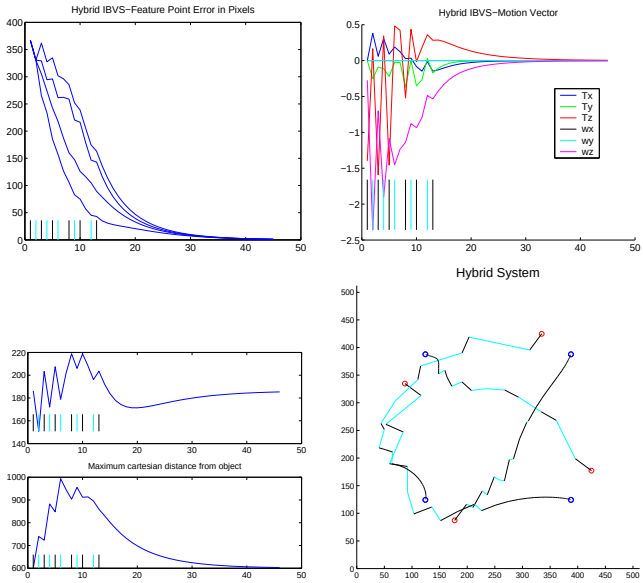


Figure 7: *Deterministic Switching*

5.3 Experimental Results

Due to errors in precisely measuring the offset of the focal point of our mounted camera to the end effector, the IBVS system did not perform ideally in our experiments. The system did not always reduce the feature point error, though it did perform expected motions such as camera retreat during rotation. For this reason we first present a series of simulation to show performance under ideal conditions. We will then present the results of our experiments. We feel that the fact that the systems worked well, even when the subsystems did not perform properly, is a testiment to the strength of the switching method.

For the simulations we have a goal image where the feature points are close to the image border, and an error image where the camera is rotated by 160°about the optical axis. This is an extremely difficult task for the indivisual subsystems. In our simulations, using only the PBVS method would result in a loss of the feature points, and using only the IBVS method induced a camera retreat of 10 meters. Either of these would likely cause failure in a physical system. All three methods of switching systems were successfully able to zero the error.

The first simulation shows the deterministic switching system. Figure 7 Shows the feature point errors, the velocity screw, the value of our switching parameters and the feature point positions at each iteration. Tick marks at the bottom of the graphs show the system currently being used: black for IBVS, cyan for PBVS. The color of the lines regarding position follow the same color scheme regarding which system is determining the motion.

The feature points begin far from the center of the image, so we begin in IBVS mode to bring the points towards the goal. Indeed, the maximum error decreases, along with a sharp increase in the distance of the camera from the feature points. We enforced a threshold of 1 meter for the camera distance, so the camera performs a number of switches, keeping both distance and feature error below their threshold. when the majority of rotation is performed and camera retreat no longer occurs, the system settles into IBVS to complete the motion.

Figure 8 shows results for the random switching method. The feature points are kept within the image, but the camera retreats past the threshold we set for the deterministic method. Our final simulation result is for probabilistic switching, shown in Figure 9. The feature point excursion is kept extremely low, and the camera distance is generally a bit lower than that experienced under the deterministic method.

We conducted experiments for a very similar situation, feature points close to the image edge and a large amount of rotation. We used the same camera, robot and feature points described in Section 4.4; examples of the goal image and initial offset image are shown in Figure 10. Both subsystems failed for this task. PBVS lost the feature points, and IBVS, experienced a great deal of camera retreat before ultimately losing the feature points when making rotations.

The figures show the same data we presented in the simulated results, with some subtle changes. The graphs of both the feature point error and feature point trajectories have the color of the dot they correspond to, and trajectories with a black shadow indicate that PBVS was used to

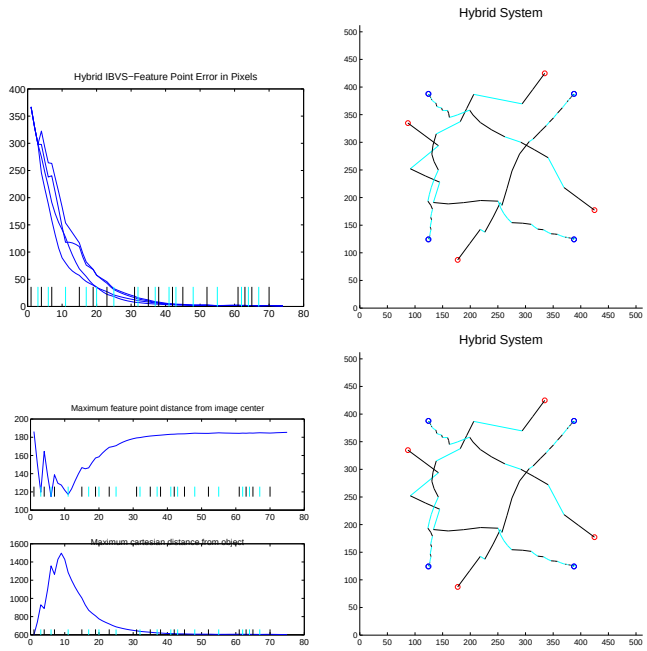


Figure 8: *Random Switching*

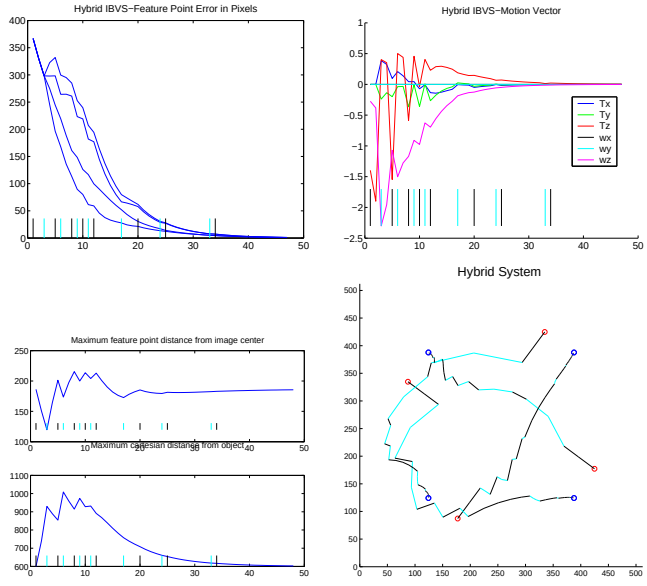


Figure 9: *Probabilistic Switching*

calculate that motion.

Figure 11 shows the results for the deterministic switching method. Since the feature points are close to the image edge we are in IBVS mode initially. The distance of the feature points from the image center is reduced with a corresponding increase in camera distance. As we reported, the IBVS system does not perform perfectly, there

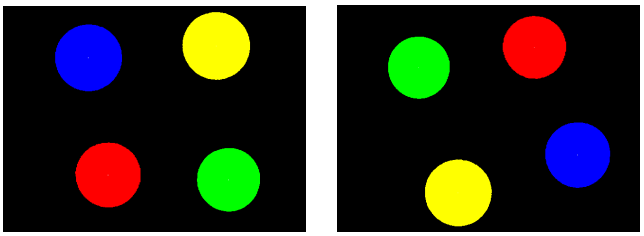


Figure 10: *Goal and Initial Image*

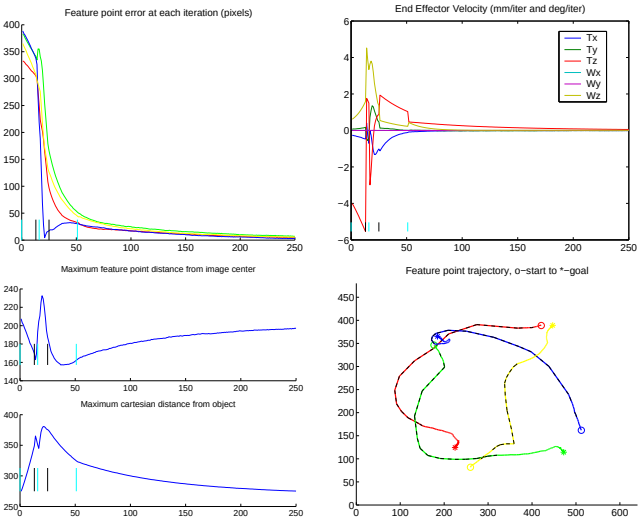


Figure 11: *Deterministic Switching*

is a brief period where both camera distance and increase, then IBVS is able to begin reducing the feature point distance again while increasing the camera distance. Finally PBVS takes over and is able to reduce both. The system is unable to completely zero the feature point error, after 250 iterations when visual servoing was halted. The only remaining motion was translation along the optical axis.

Results for random selection are seen in Figure 12. The feature distance from the image center is higher than seen in the deterministic method, but the camera retreat is kept much lower. Due to the lower camera retreat, the system is able to zero the error faster than either the deterministic or probabilistic methods. However, the maximum feature point distance is 250 pixels; clearly, this must be along the horizontal image axis, or the feature point would have left the field of view and the system would have failed. This indicates a potential for system failure using the random method, though the system never failed during our experiments.

The probabilistic method is presented in Figure 13 The method choices are identical to the deterministic method for the first thirty iterations or so, thus the graphs are ex-

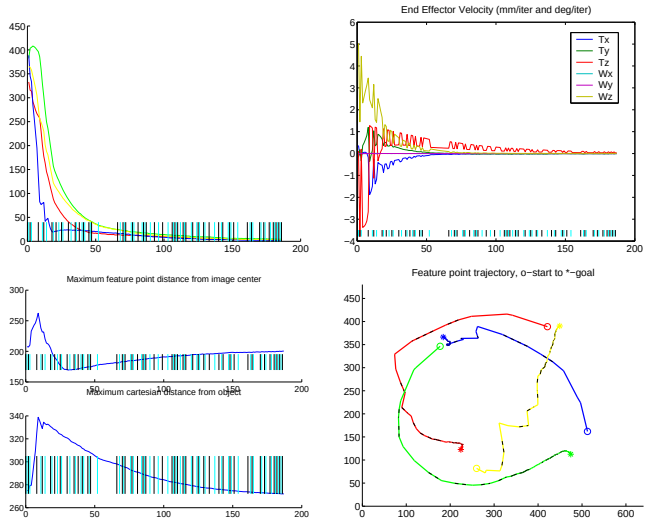


Figure 12: *Random Switching*

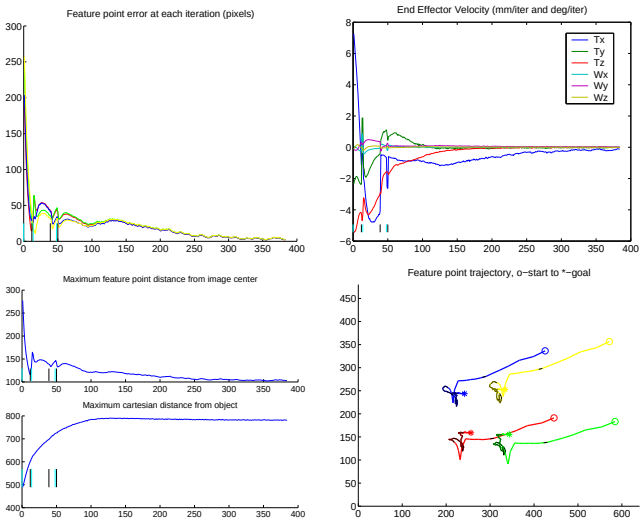


Figure 14: *Deterministic Switching*

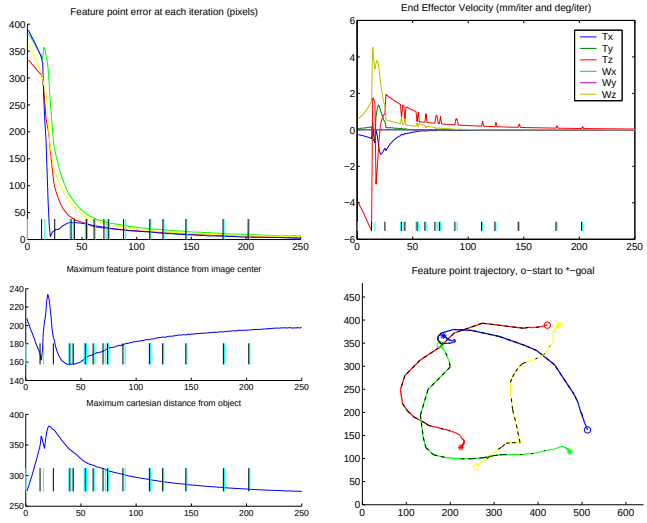


Figure 13: *Probabilistic Switching*

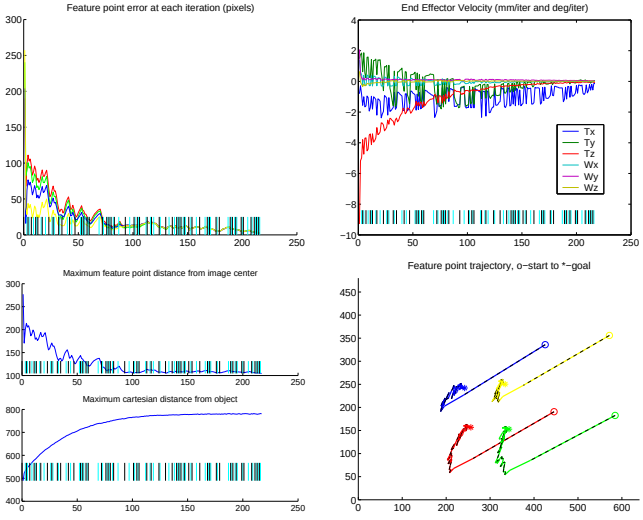


Figure 15: *Random Switching*

tremely similar. After this point the switching does become fairly random, though the velocity screw is the only graph where there is much evidence of change. This system is also unable to completely zero the error after 250 iterations.

In figures 14, 15 and 16 we repeated the experiments of section 4.4 for an oblique view. For this task, both subsystems are capable of zeroing the feature point error, and camera retreat is not a dangerous factor here. The deterministic and probabilistic methods again perform very similarly. The random system is able to zero the error more quickly than the other two methods, but, in general, experiences a larger feature point error.

5.4 Discussion of Stability

As mentioned in section 3, there are two potential ways to establish stability of this switched system that seem fruitful. The two Lyapunov functions, (5) and (8), for the IBVS and PBVS systems, respectively, are reproduced below with an added notation to distinguish the potential energy matrices $\mathbf{K}_p$.

$$V = \frac{1}{2}\dot{\mathbf{q}}^T\mathbf{M}\dot{\mathbf{q}} + \frac{1}{2}\dot{\mathbf{f}}^T\mathbf{K}_{pIB}\dot{\mathbf{f}} \tag{17}$$

$$V = \frac{1}{2}\dot{\mathbf{q}}^T\mathbf{M}\dot{\mathbf{q}} + \frac{1}{2}\dot{\mathbf{r}}^T\mathbf{K}_{p}\dot{\mathbf{r}} \tag{18}$$

These functions are extremely similar, and according to [19]both have the same time derivative, (6). Therefore it

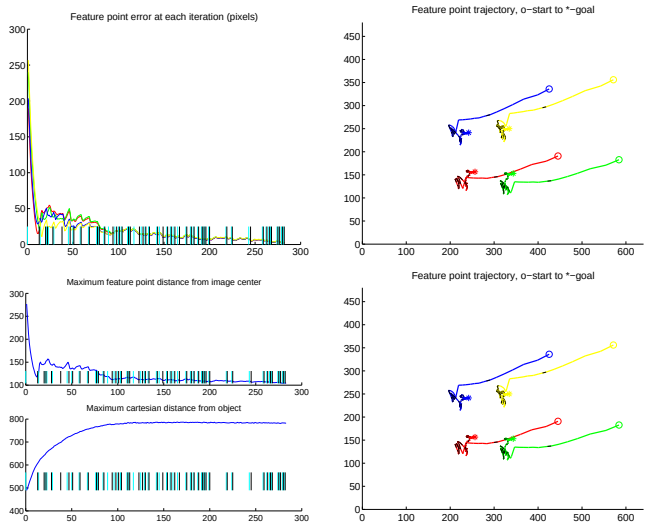


Figure 16: *Probabilistic Switching*

seems promising that we can find a common Lyapunov function for both systems. Indeed, using (1) we can give (5) the same form as 8.

$$V = \frac{1}{2}\dot{\mathbf{q}}^T\mathbf{M}\dot{\mathbf{q}} + \frac{1}{2}\dot{\mathbf{r}}^T\hat{\mathbf{K}}_{pPB}\dot{\mathbf{r}} \qquad (19)$$

where $\hat{\mathbf{K}}_{pPB} = {\mathbf{J_{im}}}^T\mathbf{K}_{pIB}\mathbf{J}_{im}$. We now have a common Lyapunov function, however the matrix $\hat{\mathbf{K}}_{pPB}$ is time varying and dependent upon $\tilde{\mathbf{W}}$. The effects this will have on the proof of stability have not yet been explored.

Alternately we can leave the Lyapunov functions independent, but focus on their similarities. The magnitude of both functions differ only in the potential energy portion. Due to the performance characteristics of the systems, it is possible that a decrease in the potential energy of one function will be accompanied by an increase in the other, e.g. camera retreat reduces the feature point error and the IBVS energy function, but increases camera distance and the PBVS energy function. If it can be established or enforced that the system only switches when the current energy function has been brought below the last value of the energy function before the previous switch (Figure 2), then stability can be assumed.

6 Conclusion

We have presented two dynamic switching visual servo systems. Each system is composed of two very different subsystems, each a visual servo system in its own right, and a decision maker that chooses which system to use at each iteration depending on the current state of the camera or image features. Simulation and experimental results are extremely promising, in some cases allowing for successfully completing a task when either subsystem would fail alone. We investigated several basic switching rules to see how this affected performance. While there remains room for future experimentation and mathematical establishment of stability, the results seen thus far are compelling.

References

[1] L. E. Weiss, A. C. Sanderson, and C. P. Neuman, "Dynamic sensor-based control of robots with visual feedback," *IEEE Journal of Robotics and Automation*, vol. RA-3, pp. 404–417, Oct. 1987.

[2] J. Feddema and O. Mitchell, "Vision-guided servoing with feature-based trajectory generation," *IEEE Trans. on Robotics and Automation*, vol. 5, pp. 691–700, Oct. 1989.

[3] B. Espiau, F. Chaumette, and P. Rives, "A new approach to visual servoing in robotics," *IEEE Trans. on Robotics and Automation*, vol. 8, pp. 313–326, June 1992.

[4] N. P. Papanikolopoulos, P. K. Khosla, and T. Kanade, "Visual tracking of a moving target by a camera mounted on a robot: A combination of vision and control," *IEEE Trans. on Robotics and Automation*, vol. 9, no. 1, pp. 14–35, 1993.

[5] F. Chaumette, "Potential problems of stability and convergence in image-based and position-based visual servoing," in *The confluence of vision and control* (D. Kriegman, G. Hager, and S. Morse, eds.), vol. 237 of *Lecture Notes in Cont. and Info. Sci.*, pp. 66–78, Springer-Verlag, 1998.

[6] E. Malis, F. Chaumette, and S. Boudet, "2-1/2d visual servoing," *IEEE Trans. on Robotics and Automation*, vol. 15, pp. 238–250, Apr. 1999.

[7] G. Morel, T. Liebezeit, J. Szewczyk, S. Boudet, and J. Pot, "Explicit incorporation of 2d constraints in vision based control of robot manipulators," in *Experimental Robotics VI* (P. Corke and J. Trevelyan, eds.), vol. 250 of *Lecture Notes in Cont. and Info. Sci.*, pp. 99–108, Springer-Verlag, 2000.

[8] K. Deguchi, "Optimal motion control for image-based visual servoing by decoupling translation and rotation," in *Proc. Int. Conf. Intelligent Robots and Systems*, pp. 705–711, Oct. 1998.

[9] P. I. Corke and S. A. Hutchinson, "A new partitioned approach to image-based visual servo control," in *Proc. on 31st Int'l Symposium on Robotics and Automation*, 1999.

[10] M. Branicky, V. Borkar, and S. Mitter, "A unified framework for hybrid control," in *Proc, of the 33rd IEEE Conf. on Decision and Control*, 1994.

[11] R. W. Brockett, *Hybrid models for motion control systems*. 1993. H. L. Trentelman and J. C. Willems, Eds.

[12] S. Hutchinson, G. Hager, and P. Corke, "A tutorial on visual servo control," *IEEE Trans. on Robotics and Automation*, vol. 12, pp. 651–670, Oct. 1996.

[13] J. Aloimonos and D. P. Tsakiris, "On the mathematics of visual tracking," *Image and Vision Computing*, vol. 9, pp. 235–251, Aug. 1991.

[14] R. M. Haralick and L. G. Shapiro, *Computer and Robot Vision*. Addison Wesley, 1993.

[15] R. Kelly, R. Carelli, O. Nasisi, B. Kuchen, and F. Reyes, "Stable visual servoing of camera-in-hand robotic systems," in *IEEE Trans. on Mechnatronics*, 2000.

[16] R. Haralick, C. Lee, and K. Ottenberg, "Analysis and solutions of the three point perspective pose estimation problem," in *Proc. IEEE Conf. Computer Vision and Patter Recognition*, 1991.

[17] D. DeMenthon and L. S. Davis, "Model-based object pose in 25 lines of code," in *European Conference on Computer Vision*, pp. 335–343, 1992.

[18] W. Wilson, "Visual servo control of robots using Kalman filter estimates of relative pose," in *Proc. IFAC 12th World Congress*, (Sydney), pp. 9–399 to 9–404, 1993.

[19] L. Deng, F. Janabi-Sharifi, and W. Wilson, "Stability and robustness of visual aervoing methods," in *Proc. of 2002 IEEE Conf. of Robotics and Aut.*, 2002.

[20] D. Liberzon and A. Morse, "Basic problems in stability and design of switched systems," *IEEE Control Systems Magazine 19*, 1999.

[21] M. Branicky, "Multiple lyapunov functions and other analysis tools for switched and hybrid systems," in *IEEE Trans. Automat. Contr.*, 1998.

[22] N. R. Gans and S. A. Hutchinson, "A switching approach to visual servo control," in *to appear in Proc. IEEE Int'l Symp. on Intelligent and Control*, 2002.

[23] O. Faugeras and F. Lustman, "Motion and structure from motion in a piecewise planar environment," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 2, no. 3, pp. 485–508, 1988.

[24] Z. Zhang and A. Hanson, "3d reconstruction based on homography mapping," in *ARPA Image Understanding workshop, Palm Springs, CA*, 1996.

[25] R. Boel and J. van Schuppen, "Distributed routing for load balancing," in *Proceedings of the IEEE, vol.77, Iss.1, 1989*, pp. 210–221, jan 1989.

A Visuo-Motor Control Architecture for High Speed Visual Servoing

Koichi Hashimoto, Akio Namiki and Masatoshi Ishikawa

Department of Information Physics and Computing, University of Tokyo
http://www.k2.t.u-tokyo.ac.jp

Abstract

A control architecture for visual servoing is discussed. A basic comparison is given between position-based and feature-based schemes. Also an architecture based on the human brain motor control is presented. The model has five hierarchical modules: motoneurons, premotor interneurons, pattern generator, parameter selection and action planning. In this architecture, the effectors including biophysical properties receive the commands from motoneurons; the premotor interneurons and motoneurons are implemented as a servo module; the pattern generator corresponds to the motion planner; and the parameter selection is realized by adaptation module. The afferent information is the feedback signal and the efferent information corresponds motion command and parameter adaptation. This architecture is implemented on a DSP network system with a high-speed vision, a dextrous hand and a 7 DOF manipulator. An example of task, grasping/handling of a dynamically moving object, is realized. The results show responsive and flexible actions that exhibit the effectiveness of the proposed hierarchical modular structure.

1 Introduction

Human beings have high ability to achieve dynamical tasks based on visual information feedback. Ball games such as baseball, football and tennis are good examples that require vision-based dynamical tasks. Hitting a cockroach with folded newspaper is a little more difficult because the dynamics of the object motion is unknown. To realize such ability it is essential to measure the dynamically changing environment and the moving object in real-time. The object motion model learned by training may be an important help but real-time visual feedback is necessary since model construction should be done through visual measurement. Thus use of real-time sensory feedback of vision as well as other sensors is essential [1, 2].

Visual servoing schemes proposed so far can be classified into two categories: position-based and feature-based. Discussion on the difference have been done in many references but discussion based on human architecture has not been found. Since human architecture provides us important suggestions, a review is given below.

Limiting the task to grasping, several applications of real-time sensory (mainly visual) feedback have been studied. Allen et al. realized grasping of a toy train based on the optic flow field [3]. Hong et al. realized grasping of a flying ball using stereo visual feedback [4]. Jägersand et al. realized grasping using visual servo with model estimation [5]. Namiki et al. developed a high-speed grasping system using visual feedback at the cycle time of 1 ms [6]. For the grasping tasks realized so far, the visual feedback is used only for position control of the gripper.

However, the visuo-motor loop in human brain controls not only hand position. At least, the shape of the hand and the directions of both eyes are simultaneously controlled. Moreover, the sensory outputs from other sensors (afferent information) and the information of arm motion command (efferent copy) interact in the visuo-motor loop. A model of this motor control architecture is shown in Figure 1 [7]. It illustrates the many structural/functional levels on which sensory and motor pathways interact.

In this paper, a control architecture for visual servoing is discussed. Some comparison on position-based and feature-based architecture is given. Also an architecture based on the human brain motor control is presented. The hierarchical parallel processing architecture is proposed based on the hierarchical efferent/afferent interaction model for human visuo-motor control system. The hierarchical structure of the proposed architecture is appropriate for multiple heterogeneous tasks. Grasping is a good example of a coordination of heterogeneous tasks, such as tracking control of active vision, position control of the arm and preshaping of the hand. Thus we developed

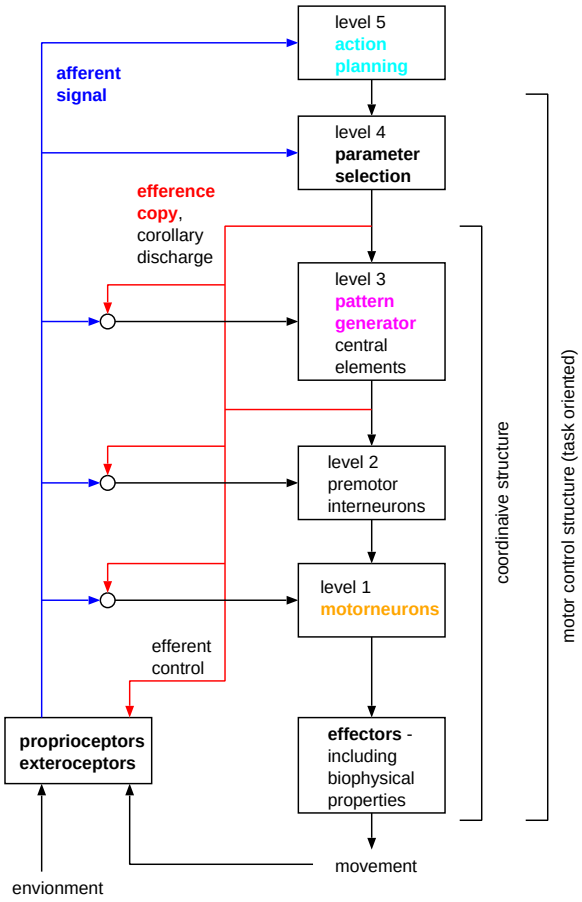


Figure 1: *Efferent/afferent interaction model*

a real-time system that realizes very quick grasping tasks. Experimental results of high-speed tracking, grasping, manipulation and collision avoidance are presented to demonstrate the effectiveness of the proposed hierarchical parallel processing architecture.

2 Basic Discussion: position or feature?

To clarify the difference between position-based and feature-based visual servoing, we consider a simple example. Suppose a 2 link robot and a pair of cameras shown in Figure 2. The robot is controlled on the basis of the image obtained by the stereo cameras. The robot can move in a horizontal plane and the cameras are placed in parallel on the plane. Let the optical axes of the cameras be in parallel to the z axis. The image plane is perpendicular to the z axis and the base line length is b.

In left camera, let the hand image be l_h and the goal image be l_g. In right camera, the hand and goal images are r_h and r_g, respectively. For the robot, let the joint angle be $\theta = (\theta_1, \theta_2)$, link length be

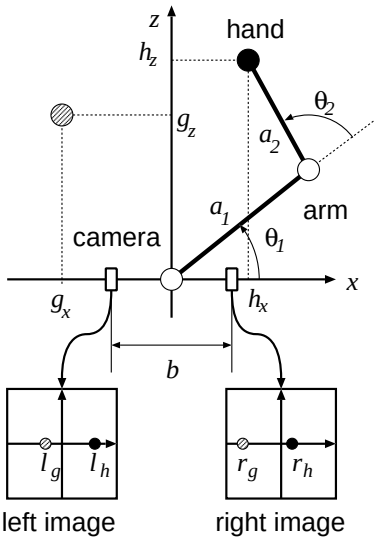


Figure 2: *2 Link Arm with Stereo Setup*

$a = (a_1, a_2)$, hand position be $h = (h_x, h_z)$. The goal position is $g = (g_x, g_z)$.

2.1 Position-based Scheme

Imaging Model. We denote the imaging model by $C_g : g \to \xi_g$, where $\xi_g = (l_g, r_g)$ is the goal image in both cameras. Simple calculation yields

$$l_g = f\frac{g_x + b/2}{g_z}, \quad r_g = f\frac{g_x - b/2}{g_z} \tag{1}$$

where f is the focal length.

Stereo Model. The stereo model $E_g : \xi_g \to \hat{g}$ is the inverse of (1), i.e.,

$$\hat{g}_x = \hat{f}\frac{\hat{b}(l_g + r_g)}{d_g}, \quad \hat{g}_z = \hat{f}\frac{\hat{b}}{d_g} \tag{2}$$

where $d_g = l_g - r_g$ is called disparity. It is well known that disparity is important in 3D reconstruction of human. Error in the estimate of goal position $\hat{g}$ is due to the error of camera's internal and external parameters (in this example, only $\hat{f}$ and $\hat{b}$ are used).

Kinematic Model. If one estimates the hand position from the joint angle measurements θ, kinematic model $\hat{K} : \theta \to \hat{h}$ where

$$\begin{aligned} \hat{h}_x &= \hat{l}_1 \cos\theta_1 + \hat{l}_2 \cos(\theta_1 + \theta_2), \\ \hat{h}_z &= \hat{l}_1 \sin\theta_1 + \hat{l}_2 \sin(\theta_1 + \theta_2) \end{aligned} \tag{3}$$

are necessary. The kinematic parameters $(\hat{l}_1, \hat{l}_2$ in this example) affects the hand position accuracy.

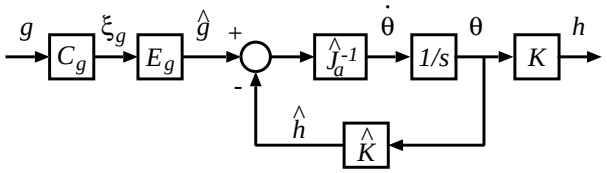


Figure 3: *Position-based Stereo*

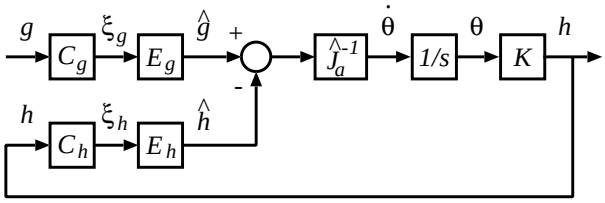


Figure 4: *Position-based Stereo II*

Control Law. Assume that the robot has perfect velocity controller (the command velocity is assumed to be perfectly realized). Then a very simple resolved motion velocity control law can be used. Since the relationship between the joint and hand velocities are given by

$$J_a = \frac{\partial h}{\partial \theta} \tag{4}$$
$$= \begin{bmatrix} -l_1 \sin\theta_1 - l_2 \sin(\theta_1+\theta_2) & -l_2 \sin(\theta_1+\theta_2) \\ l_1 \cos\theta_1 + l_2 \cos(\theta_1+\theta_2) & l_2 \cos(\theta_1+\theta_2) \end{bmatrix},$$

the control law becomes as follows:

$$\dot{\theta} = \lambda \hat{J}_a^{-1}(\hat{g} - \hat{h}) \tag{5}$$

The elements of the matrix J_a includes the kinematic parameters, the error in kinematic parameters can affect the closed loop stability. The block diagram of this control scheme is shown in Figure 3.

Robustness. If there are errors in the model E or $\hat{K}$, the feedback, which tries to $\hat{h} \to \hat{g}$, will not work to eliminate the model errors; and thus h will not converge to g. This fact can be easily understood from the block diagram (Figure 3).

If E and $\hat{K}$ are accurate enough, then the condition for convergence of h to g depends on $\hat{J}_a$. By considering a Lyapunov function candidate as

$$V = (h-g)^T(h-g) \tag{6}$$

then it is easy to verify that the condition is: $J_a \hat{J}_a^{-1} > 0$. This is due to

$$\dot{V} = -2\lambda(h-g)^T J_a \hat{J}_a^{-1}(h-g) \tag{7}$$

and this condition is much milder than the assumption "E and $\hat{K}$ are accurate enough."

2.2 Position-based Scheme II

If one can observe the hand by cameras, the hand position can be estimated using vision. Let the imaging model for hand be C_h and stereo model for hand position estimation be E_h. Then the block diagram of this control scheme becomes Figure 4.

In this case we do not need $\hat{K}$. It is easy to confirm that, if E_g and E_h use the same camera parameters, the parameter error do not affect the task accuracy. For example if b and f are modeled as b' and f', respectively, then the effects of the estimation error on $\hat{g}$ and $\hat{h}$ become the same ($\frac{b'f'}{bf}$). Thus this control scheme guarantees the task accuracy even under parameter uncertainty. The closed loop robustness condition is the same as the previous position-based scheme.

2.3 Feature-based Scheme

In feature-based scheme the hand image is directly controlled. The relationship between the joint and hand image $\xi_h = (l_h, r_h)$ velocities is given by

$$J_i = \frac{\partial \xi_h}{\partial \theta} = \frac{\partial \xi_h}{\partial h}\frac{\partial h}{\partial \theta}. \tag{8}$$

It is straightforward to compute

$$\frac{\partial \xi_h}{\partial h} = \begin{bmatrix} f\frac{b}{2h_x} & -f\frac{h_x + b/2}{h_z^2} \\ -f\frac{b}{2h_x} & -f\frac{h_x - b/2}{h_z^2} \end{bmatrix} = \begin{bmatrix} d_h/2 & -l_h d_h/fb \\ -d_h/2 & -r_h d_h/fb \end{bmatrix}. \tag{9}$$

where $d_h = l_h - r_h$ the disparity. The matrix $\frac{\partial h}{\partial \theta}$ is given in (4). A resolved motion rate control for hand image becomes as follows:

$$\dot{\theta} = \lambda \hat{J}_i^{-1}(\xi_g - \xi_h). \tag{10}$$

The block diagram of this control scheme is depicted in Figure 5.

This control scheme do not need the kinematic model nor the stereo model, thus the task accuracy is not affected by the uncertainties in these parameters. Though, the matrices $\frac{\partial \xi_h}{\partial h}$ and $\frac{\partial h}{\partial \theta}$ includes these parameters, the estimation error in J_i is not very significant because the closed loop stability is guaranteed when $J_i \hat{J}_i^{-1} > 0$. However, one should note that the stability of this control scheme is local and the task definition is not very flexible (suppose a task to keep constant the distance between hand and moving object: how do you define the reference image?).

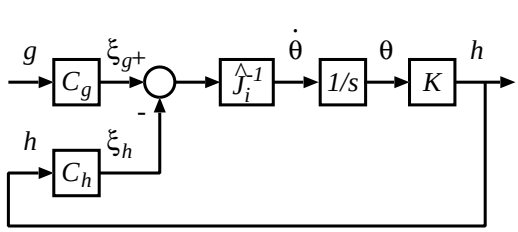


Figure 5: *Feature-based Stereo*

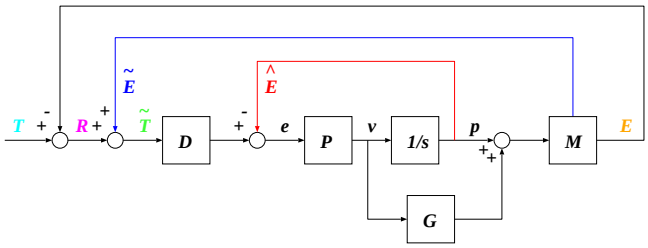


Figure 6: *Model of Saccade*

3 Efferent/afferent Interaction Model

The different functions incorporated in motor control structure define a hierarchy of control levels. Generation of activity in specific motor units lies at the bottom and task function is at the top. The signal from sense organs (afferent information) and the copy of the commands (efferent copy) for the pattern generator (and premotor interneurons) act on each levels [7].

The lowest structural level denotes the reflex in which activity in sensory fibers (proprioceptors and exteroceptors) is passed directly onto motoneurons. The function of these connections ranges from selecting a response to modifying details of motor activity. These direct connections provide fast responses but offer limited complexity. Direct connections from sense organs to motoneurons which trigger actions are common in withdrawal reflexes where a fast response is advantageous for avoiding injury.

The premotor interneurons are introduced as an intermediate structural level between the oscillator or pattern generator and the motor neurons. This level is not present in all systems but it is useful for illustrating some kinds of temporal and spatial integration. Premotor interneurons can participate in temporal integration by prolonging a response triggered by brief sensory inputs and by transforming sensory input in various ways. The introduction of interneurons is a simple way to increase the number of response combinations.

The pattern generator may incorporate complicated central networks as well as sensory influences. In many behaviors, sensory inputs can affect a rhythm and the sensory information on a pattern generator is used to determine the successful completion of a movement.

At higher levels of motor organization, sensory information is used to select appropriate motor control structures and to determine their parameters. This function depends on the extraction of appropriate sensory information about the environment. It requires sensory processing of variable complexity and may even involve internal maps or representation of the animal and its relation to its environment.

Note that the hierarchical representation of the control functions does not necessarily correspond to separate structural levels in the nervous system.

4 A Model for Saccade

When the eyes track a moving stimulus, there is a slow phase of eye movement to keep the gaze focused on the object (smooth pursuit); when the eyes move to focus on a different point, the eyes snap to the new position in a rapid movement (saccade) [8]. In this section the model of saccade control is considered. Two hypotheses can be made on the model of saccade: one is the image-based and the other is the position-based. The image-based scheme assumes that the motion is generated to compensate the error between the target image in retina and the center of retina. In the position-based hypothesis, the brain constructs the object position in 3D space to control the position estimation.

A model of saccade proposed by Zee [9] is shown in Figure 6. The block M is the eye muscle and the signal E stands for the eye direction. The target position is T and the retina image is R. The block D means a delay, P and G are the gain block. The signals v and p are the velocity and position parts in the motion control command. The brain estimates the target direction $\tilde{T}$ from the visual information of the retina R and the direction of eye with respect to the head $\tilde{E}$. After a delay of about 0.2 second, saccade motion is started. At the beginning of the motion the brain compares the directions of eye and target, and until the difference becomes sufficiently small the brain commands the eye to keep moving.

This model has two important engineering interpretation. One is that the brain continuously monitors the directional error between the eye and the target to generate the motion command. The other is that the brain integrates the temporal information (di-

rection of motion of the eye) and spatial information (retinal image).

From biological experiments it is proved that the eye direction signal is composed of two different signals $\tilde{E}$ and $\hat{E}$. First is the afferent signal from proprioceptor (sensor signal) and the other is the copy of efference signal (copy of motion command). The sensor signal includes about 10ms delay because it fires after eye completes its motion. On the other hand, the efference copy do not include the delay. However the efference copy requires learning based on proprioceptor signal to keep the accuracy.

The efference copy is used in the inner feedback loop to servo the eye direction. Since the proprioceptor signal include large delay it is not suitable for stabilization. On the other hand, the proprioceptor signal is used in conjunction with the retina image to reconstruct the object direction. This outer loop feedback composes a position-based visual servo system.

5 Usage of Visual Feedback

5.1 Feedback for servo control

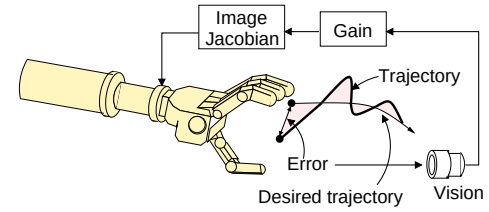
Part (a) in Figure 7 shows an example of servo control in which a hand is controlled by observing the positional error through vision. When the vision sensor is slow compared to the arm dynamics, the vision feedback is used as an outer loop of the joint servo loop and an inner servo loop based on efference copy becomes necessary [2, 10]. However, as shown in this example, if the vision system is fast enough to construct direct servo loop, a stable and highly responsive control can be expected.

5.2 Desired trajectory generation

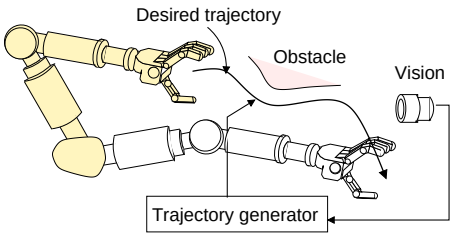
Sensory feedback can be used for generating desired hand trajectory. Part (b) in Figure 7 is an example in which the optimal arm trajectory against a moving obstacle is computed by using visual feedback. The servo control system becomes a sublayer of this module.

5.3 Task switching

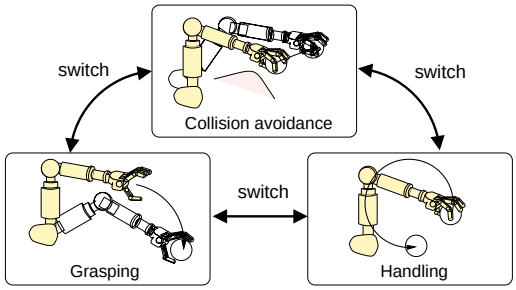
Sensory feedback can be used for switching tasks according to the state of environment. Part (c) in Figure 7 gives an example of task switching in which three subtasks (grasping, object handling, and collision avoidance) are switched according to the state of the environment. To select an optimal subtask in a dynamical changing environment, high-speed feedback is needed. Each subtask requires the trajectory generation module.



(a) Servo control



(b) Trajectory generation



(c) Task switching

Figure 7: *Use of real-time sensory feedback*

6 Hierarchical Sensory Feedback Model

To simplify the problem, it is assumed that the state of the system is uniquely described by a state vector $\boldsymbol{z} \in \boldsymbol{R}^{m_z}$. A part of the state can be controlled, e.g. joint angle of the arm and direction of the eye. We assume the existence of a function $\boldsymbol{f}$ that selects the controllable state from the whole state $\boldsymbol{\theta} = \boldsymbol{f}(\boldsymbol{z})$. The controllable state $\boldsymbol{\theta} \in \boldsymbol{R}^{m_\theta}$ is called the motion parameter. We also introduce a state of sensor signal $\boldsymbol{s} \in \boldsymbol{R}^{m_s}$. It is also assumed that the sensor signal is rich enough to estimate the whole system state.

6.1 Control layer

Corresponding to the usage of vision sensor shown in Figure 7 (a), servo control layer is implemented as the levels 1 and 2 of the efferent/afferent interaction model.

Let the number of subtasks that need servo control be M_z and let the state concerning each tasks be $\boldsymbol{z}_i(\boldsymbol{s}) \in \boldsymbol{R}^{m_z}$ for $(i = 1, \ldots, M_z)$. For example, let

the first subtask $i = 1$ corresponds to the pan motion of the eye and the pan angle q_p is the third element of the system state $\boldsymbol{z}$. Then $\boldsymbol{z}_1(s)$ selects q_p from the sensor output $\boldsymbol{s}$ and maps it to the third element of the state, i.e., $\boldsymbol{z}_1 = [0, 0, q_p, 0, \ldots, 0]^T$. If the pan angle control needs other sensor output, $\boldsymbol{z}_1$ also picks up the sensor signal, combines the signal with the pan angle and maps the combined signal into the third element.

Then we define the actual state that is estimated from current sensor output as a linear combination of all subtasks

$$\boldsymbol{z} = \sum_{i=1}^{M_z} S_i(\boldsymbol{s})\boldsymbol{z}_i(\boldsymbol{s}). \tag{11}$$

The coefficient matrices $S_i(\boldsymbol{s}) \in \boldsymbol{R}^{m_z \times m_z}$ for $i = 1, \ldots, M_z$ are functions of $\boldsymbol{s}$, sensor output, and are used for adaptation. The adaptation law is described in 6.3. Note that the coefficients are semi positive definite and satisfy $\sum_{i=1}^{M_z} S_i(\boldsymbol{s}) = I_{m_z}$ where I_{m_z} is the $m_z \times m_z$ identity matrix.

We assume that all actuators are velocity-control type and we adopt a simple proportional control law

$$\dot{\boldsymbol{\theta}} = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{z}} (\boldsymbol{z}_d - \boldsymbol{z}), \tag{12}$$

where $\boldsymbol{z}_d$ is the reference state. Since the purpose of this paper is to present the architecture, we do not discuss the control law. One may use any kind of control law to improve the accuracy. The important point is the estimation of system state (11) and generation of the reference $\boldsymbol{z}_d$, which is discussed below.

6.2 Planning layer (pattern generation)

The reference trajectory is generated in this layer on the basis of the sensor usage described in Figure 7 (b). This layer corresponds to the level 3 in the efferent/afferent interaction model. To deal with various subtasks, the desired trajectories (for eye direction, arm motion, finger shape and so on) are given as combinations of trajectories for all subtasks.

Let the number of subtasks that affects the control parameters of the system be M_{dz} (which is not necessarily equal to M_z) and let the reference state for each subtask be $\boldsymbol{z}_{di}(\boldsymbol{s}) \in \boldsymbol{R}^{m_z}$ for $(i = 1, \ldots, M_{dz})$. Then the actual reference state (motion pattern) is generated as a linear combination of each reference state

$$\boldsymbol{z}_d = \sum_{i=1}^{M_{dz}} U_i(\boldsymbol{s})\boldsymbol{z}_{di}(\boldsymbol{s}). \tag{13}$$

The coefficients $U_i(\boldsymbol{s}) \in \boldsymbol{R}^{m_z \times m_z}$ for $i = 1, \ldots, M_{dz}$ are used for adaptation and they are semi positive definite matrices that satisfy $\sum_{i=1}^{M_{dz}} U_i(\boldsymbol{s}) = I_{m_z}$.

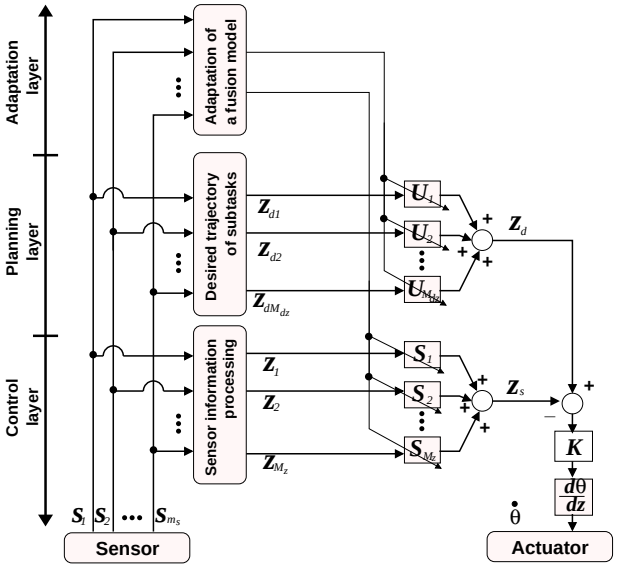


Figure 8: *Hierarchical sensory feedback model*

6.3 Adaptation layer (task selection)

The third layer is the adaptation. This corresponds to the sensor usage of Figure 7 (c). This layer is mainly for the selection of main task among various subtasks. The selection is realized by changing the coefficient matrices $S_i(\boldsymbol{s})$ and $U_i(\boldsymbol{s})$.

On the basis of these control strategies, a hierarchical parallel processing model is constructed as depicted in Figure 8. Since all feedback loops (including vision loop) have very high sampling rate (1 kHz), adaptation and task selection corresponding to the dynamic changes of the environment can be achieved. Moreover, this computation model has an advantage that the structural change of the control system, e.g., addition or deletion of sensory feedback, is relatively easy because the dependence between tasks and sensors are described explicitly in the definition of the state.

7 High-speed Grasping Algorithm

We assume that the system is composed by a 6-axis manipulator with a multi-finger hand and a monocular vision. Each joint of the finger has a force sensor. The contact force between the fingers and the object can be measured by using them. The task is a series of general manipulation processes, i.e., grasp the object, handle the object, and avoid other objects.

7.1 Control algorithm

Suppose that the joint angle vector is $\boldsymbol{\theta}^a \in \boldsymbol{R}^6$, the position and the orientation of the hand is $\boldsymbol{x}^a \in \boldsymbol{R}^6$,

the position and orientation of the object is $\boldsymbol{x}^o \in \boldsymbol{R}^6$, the force and moment observed by the force/torque sensor is $\boldsymbol{F}^a \in \boldsymbol{R}^6$. The object position is measured by vision and the force is measured by force/torque sensor. Then, on the basis of a typical force control scheme, the desired joint angle velocity $\boldsymbol{v}_d^a \in \boldsymbol{R}^6$ is computed as follows:

$$\boldsymbol{v}_d^a = J^{a^{-1}} K^a (\boldsymbol{x}_d - \boldsymbol{x}_s) - K^{av} \dot{\boldsymbol{\theta}}^a + J^{a^T} K^{af} \boldsymbol{F}^a, \quad (14)$$

where $\boldsymbol{x}_d$ and $\boldsymbol{x}_s$ correspond to the desired and the actual hand positions, respectively. These vectors are defined in the following subsection. The matrix $J^a \equiv \frac{\partial \boldsymbol{x}^a}{\partial \boldsymbol{\theta}^a}$ is the arm Jacobian and the matrices K^a, K^{av} and K^{af} are positive definite diagonal gain. In this equation the first term denotes the position control based on vision sensor ($\boldsymbol{x}_d$ and $\boldsymbol{x}_s$ are generated from the object position and the hand position observed by vision sensor), the second term denotes the velocity feedback, and the third term is the force feedback using the force/torque sensors.

7.2 Task encoding

The variables appeared in the first term of (14) are defined as follows:

$$\begin{aligned} \boldsymbol{x}_d \equiv\ & (I_6 - G^a)(I_6 - G^m)(I_6 - G^s)\, \boldsymbol{x}^o \\ & + (I_6 - G^a)(I_6 - G^m)\, G^s\, \boldsymbol{x}_d^a, \\ & + (I_6 - G^a)\, G^m \boldsymbol{x}_d^o + G^a\, \boldsymbol{x}_d^c, \quad (15) \\ \boldsymbol{x}_s \equiv\ & (I_6 - C^m)\, \boldsymbol{x}^a + C^m\, \boldsymbol{x}^o, \quad (16) \end{aligned}$$

where $\boldsymbol{x}_d$ is the reference trajectory generated by integrating desired trajectories of the subtasks and $\boldsymbol{x}_s$ is the actual controlled position. The reference is a function of the object position $\boldsymbol{x}^o$, the desired handling trajectory $\boldsymbol{x}_d^o$, the desired avoidance trajectory $\boldsymbol{x}_d^c$ and the desired reaching trajectory $\boldsymbol{x}_d^a$. The matrices G^a and G^m select the subtasks and the matrix G^s divide the workspace into tracking and reaching subspaces. The vector $\boldsymbol{x}_s$ denotes the task dependent state of the system generated by combining the visual information $\boldsymbol{x}^o$ with the internal sensor information $\boldsymbol{x}^a$ (the fingertip position computed from the joint angle measurement $\boldsymbol{\theta}^a$). The matrix C^m switches the controlled point (If the hand is holding the object the controlled point becomes the object. Otherwise, the controlled point is the fingertip).

7.3 Selection matrices

The coefficient matrices appeared in (15) and (16) are defined as follows:

$$\begin{aligned} G^s &= \mathrm{diag}\,(1,\, 0,\, 0,\, 0,\, 1,\, 1)\,, \\ G^m &= \mathrm{diag}\,(\, g_i^m\,)\,, \quad i = 1, 2, \cdots, 6, \quad (17) \end{aligned}$$

$$g_i^m = \begin{cases} 1 & (1 < \bar{g}_i^m) \\ \bar{g}_i^m & (0 \le \bar{g}_i^m \le 1) \\ 0 & (\bar{g}_i^m < 0) \end{cases}, \quad (18)$$

$$\bar{g}_i^m = \int_{t'=0}^{t} \gamma_i^m \mathrm{sgn}(\tau^h(t') - \tau_o^h)\, dt', \quad (19)$$

$$\tau^h = \sqrt{\sum_{i=1}^{4} \boldsymbol{\tau}_i^{h^T} H_i \boldsymbol{\tau}_i^h},$$

$$G^a = \mathrm{diag}\,(\, g_i^a\,)\,, \quad i = 1, 2, \cdots, 6, \quad (20)$$

$$g_i^a = \frac{1}{1 + \exp(\gamma_i^a (l - l_o))},$$

$$C^m = C^\alpha\ G^m, \quad (21)$$

where τ^h shows the size of the weighted average of the joint torque of the hand $\boldsymbol{\tau}_i^h \in \boldsymbol{R}^3$, $\tau_o^h \in \boldsymbol{R}$ is a threshold, $l \in \boldsymbol{R}$ shows the distance between the manipulated object and the obstacle, and $l_o \in \boldsymbol{R}$ is a threshold for adjusting the distance in which avoidance motion starts. The matrices C^α, H_i and the scalers γ_i^m, γ_i^a are appropriate constants.

7.4 Encoded arm subtasks

The arm control law (14), (15) and (16) are switched according to the subtasks, i.e., tracking and reaching; handling and collision avoidance. During these subtasks, the control law can be simplified as follows:

Tracking and Reaching. In these subtasks, since the fingers do not touch the object, the finger joint torque is zero ($\tau^h = 0$) and thus the selection matrices satisfy $G^m = O_6$ and $G^a = O_6$. Therefore the control law (14) becomes

$$\begin{aligned} \boldsymbol{v}_d^a = J^{a^{-1}} K^a \{(I_6 - G^s)\,(\boldsymbol{x}^o - \boldsymbol{x}^a) + G^s\,(\boldsymbol{x}_d^a - \boldsymbol{x}^a)\} \\ - K^{av} \dot{\boldsymbol{\theta}}^a + J^{a^T} K^{af} \boldsymbol{F}^a, \quad (22) \end{aligned}$$

where the tracking motion $(I_6 - G^s)\,(\boldsymbol{x}^o - \boldsymbol{x}^a)$ is orthogonal to reaching motion $G^s\,(\boldsymbol{x}_d^a - \boldsymbol{x}^a)$, there is no interference between them.

Handling. After grasping the object, the subtask is switched to handling mode. When the grasp is stiff, the switching function g_i^m becomes unity. Thus the selection matrices are given by $G^m = I_6$ and $G^a = O_6$. Then the control law (14) is described by

$$\boldsymbol{v}_d^a = J^{a^{-1}} K^a\,(\boldsymbol{x}_d^o - \boldsymbol{x}_s^o) - K^{av} \dot{\boldsymbol{\theta}}^a + J^{a^T} K^{af} \boldsymbol{F}^a, \quad (23)$$

where $\boldsymbol{x}_s^o \equiv (I_6 - C^\alpha)\boldsymbol{x}^a + C^\alpha \boldsymbol{x}^o$ is the controlled position generated by combining the visual information with the internal sensor information.

Collision avoidance. After grasping, if other objects appear in the viewing area then the subtask is switched to collision avoidance mode. In this subtask

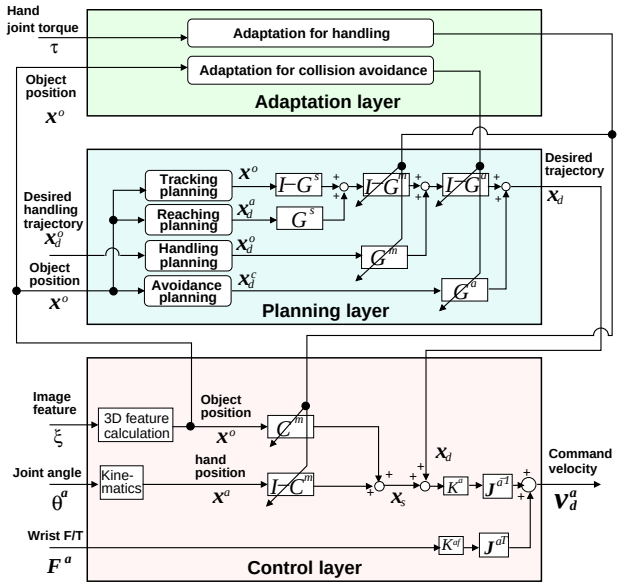


Figure 9: *Block diagram*

the selection matrices become $G^m = I_6$ and $G^a = I_6$. Thus the control law (14) is given by

$$\boldsymbol{v}_d^a = J^{a^{-1}} K^a \left(\boldsymbol{x}_d^c - \boldsymbol{x}_s^o\right) - K^{av}\dot{\boldsymbol{\theta}}^a + J^{a^T} K^{af} \boldsymbol{F}^a \quad (24)$$

Note that the avoidance force is generated by the term $(\boldsymbol{x}_d^c - \boldsymbol{x}_s^o)$ which is obtained by combining the visual information with the internal sensor information. The desired trajectory for collision avoidance is defined by

$$\boldsymbol{x}_d^c \equiv \begin{cases} \boldsymbol{x}_s^o + \begin{bmatrix} \boldsymbol{n}^l \left(l_o - l\right) \\ \boldsymbol{0} \end{bmatrix} & \text{if } l_o - l > 0, \\ \boldsymbol{x}_s^o & \text{otherwise,} \end{cases} \quad (25)$$

where $\boldsymbol{n}^l \in \boldsymbol{R}^3$ is the unit vector in the direction from the obstacle to the manipulated object.

8 Experiment

8.1 Experimental system

The experimental system is the 1 ms sensory-motor fusion system [11]. This system consists of a hierarchical parallel processing system, an active vision, a 7-axis axis arm, and a 4-fingered hand. The hierarchical parallel processing system is a multiprocessor system with eleven general-purpose floating-point DSPs. The active vision is a 2-DOF actuator (tilt and pan) and a high-speed vision chip system SPE-256 (resolution: 16×16 pixel) [12] is mounted on it. The hand is a 4-fingered 14-DOF system that has force sensors in each joint. The cycle time of whole system control is 1 ms.

8.2 Experimental result

We assume that the handling subtask is to keep the object at a specified position. Part (a) of Figure 10, shows the time response of object and hand trajectories. After grasping, obstacle trajectory is also shown. It illustrates that tracking, grasping, handling and collision avoidance subtasks are carried out. To show the subtask switching mechanism, the time responses of coefficients for adaptation G^a and G^m is given in part (b) and the time response of the grasping torque τ^h is given in part (c). Task switching from the reaching subtask to the object handling subtask is realized by the change of the coefficient matrix G^m according to the grasping torque τ^h. The task switching from the object handling subtask to the collision avoidance subtask is realized by the change of the coefficient matrix G^a due to the approach of the obstacle. In Figure 11 the total manipulation process is shown as a continuous sequence of pictures every 0.3s.

9 Conclusion

In this paper, a hierarchical parallel processing model based on high-speed sensory feedback is proposed. On the basis of this model, grasping and manipulation tasks are realized. It is shown that the system is highly responsive and flexible to the dynamic changes of an environment. The proposed algorithm for multiple heterogeneous subtasks is well suited for hierarchical efferent/afferent interaction model. The experimental results demonstrate that the manipulation is adaptable to dynamic changes of an environment and the adaptation can be realized by using the real-time sensory feedback.

References

[1] L. Weiss, A. Sanderson, and C.P. Neuman. Dynamic sensor-based control of robots with visual feedback. *IEEE J. of Robotics and Automation*, RA-3(5), 1987.

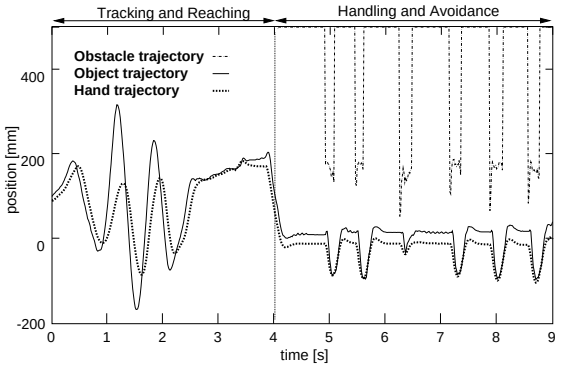
[2] K. Hashimoto, editor. *Visual Servoing —Real-Time Control of Robot Manipulators Based on Visual Sensory Feedback*. World Scientific, Singapore, 1993.

[3] P.K. Allen, B. Yoshimi, and A. Timucenko. Real-time visual servoing. *Proc. IEEE Int. Conf. Robotics and Automation*, pages 2376–2384, 1991.

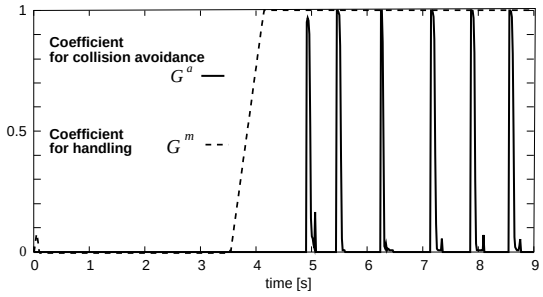
[4] W. Hong and J.E. Slotine. Experiments in hand-eye coordination using active vision. *Proc. 4th Int. Symp. on Experimental Robotics (ISER'95)*, 1995.

[5] M. Jägersand, O. Fuentles, and R. Nelson. Acquiring visual-motor models for precision manipulation with robot hands. *Proc. of 4th European Conf. on Computer Vision*, pages 603–612, 1996.

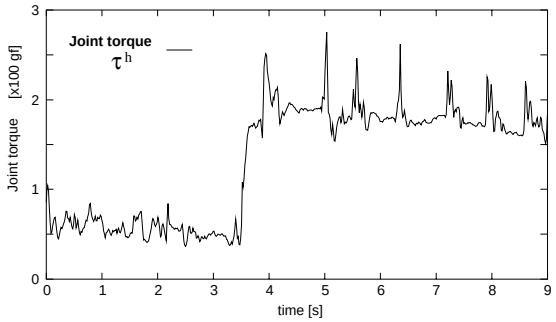
[6] A. Namiki, Y. Nakabo, I. Ishii, and M. Ishikawa. High speed grasping using visual and force feedback. *Proc. IEEE Int. Conf. on Robotics and Automation*, 1999.



(a) Trajectory



(b) Coefficient



(c) Joint torque

Figure 10: *Time response: task switching*

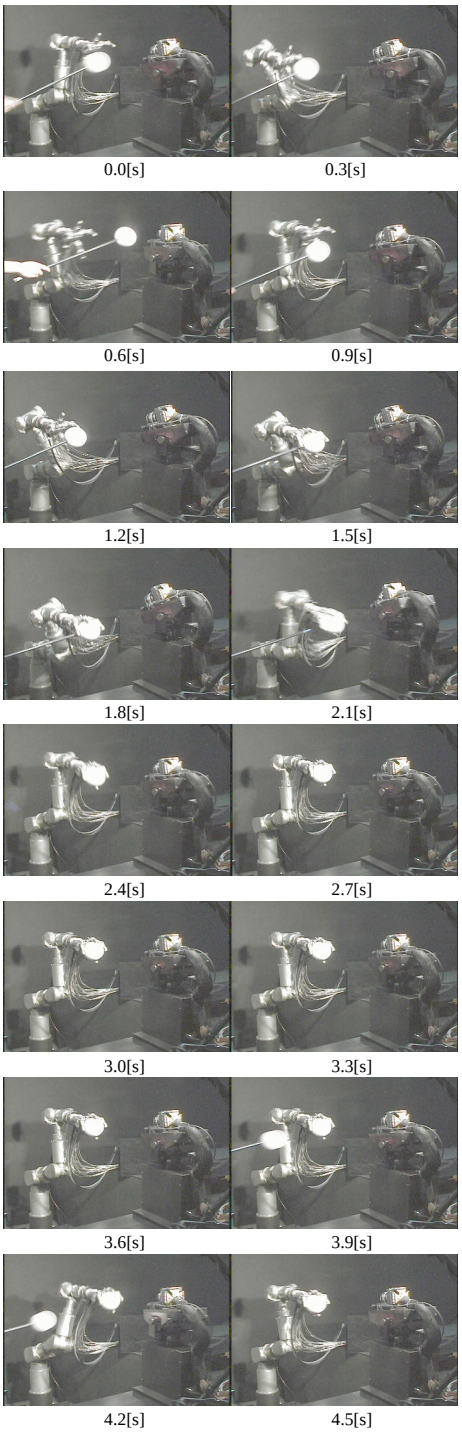


Figure 11: *Collision avoidance*

[7] J. Dean. The neuroethology of perception and action. In *Relationships Between Perception and Action, Edited by O. Neuman and W. Printz, Springer-Verlag*, pages 81–131, Berlin Heidelberg, 1990.

[8] O. Steward. *Functional Neuroscience*. Springer, New York, 2000.

[9] D. S. Zee, J. D. Cook L. M. Optican, D. A. Robinson, and W. K. Engel. Slow saccades in spinocerebellar degeneration. *Arch. Neurology*, 33:243–251, 1976.

[10] S. Hutchinson, G. D. Hager, and P. I. Corke. A tutorial on visual servo control. *Trans. on Robotics and Automation*, 12(5):651–670, 1996.

[11] A. Namiki, Y. Nakabo, I. Ishii, and M. Ishikawa. 1ms sensory-motor fusion system. *IEEE/ASME Trans. on Mechatronics*, 5(3):244–252, 2000.

[12] Y. Nakabo, I. Ishii, and M. Ishikawa. High speed target tracking using 1ms visual feedback system. In *Video Proceedings of IEEE Int. Conf. Robotics and Automation*, Minneapolis, 1996.

Visual Servoing/Tracking Using Central Catadioptric Images

João P. Barreto† Frédérick Martin‡ Radu Horaud‡

† Institute of Systems and Robotics
DEEC - University of Coimbra
3030 Coimbra, Portugal *
‡ INRIA Rhône-Alpes
655 Avenue de l'Europe
38330 Montbonnot St. Martin, France

Abstract

Visual control of robot motion may benefit from enhanced camera field of view. With traditional cameras the available fields of view are only enough to view a region around the observed object (for eye-in-hand systems) or around the end-effector (for independent-eye systems). Central catadioptric systems have larger fields of view thus allowing the entire robot AND the surrounding objects to be imaged with a unique camera. Therefore, the whole robot's articulated mechanism can be observed and its joints can be tracked and controlled simultenously. This results in a new visual robot control concept where tracking and control are embedded together. Key to the understanding of both servoing and tracking is the central catadioptric Jacobian matrix linking the robot's joint velocities to image observations. In spite of a more complex projection matrix associated with catadioptric sensors, we study the catadioptric Jacobian matrix and we show that it does not introduce any additional singularity with respect to the traditional pinhole camera model. Experiments showing a rigid body being tracked with a catadioptric camera are described.

1 Introduction

Machine vision provides noncontact measurements of the world, extending the robot ability to operate in circumstances and environments which can not be accurately controlled. The approach of controlling motion using visual information is referred in the literature as visual servoing. Visual control of motion has been the object of intensive research in the last years. Several applications have been described for pose estimation [3], robot navigation [13] and positioning tasks of robotic manipulators [1, 2].

Visual servoing applications can benefit from sensors providing large fields of view. The advantages of omnidirectional imaging in egomotion recovery from video were first discussed in [5]. Ambiguities and confusion between translation and rotation may arise whenever the translation direction lies outside the camera field of view. Panoramic sensors overcome this problem making the uncertainty of egomotion estimation independent of the direction of motion. More recently Aloimonos et al. proposed a spherical eye built with six cameras specifically designed for egomotion recovery [10]. Enhanced fields of view can also be advantageous for positioning tasks of robotic manipulators. The approaches to this problem are traditionally classified in two groups: position based and image based visual servoing [4]. In the former the control input is defined in the 3D task space. The pose of the target is estimated from image features based on the knowledge of a geometric model of the object and the camera calibration [1]. With only one camera there are ambiguities and singularities in pose estimation and the target can get out of the

field of view during the tracking. In [3] a multiple camera approach is used to cope with these difficulties. Panoramic imaging can also overcome the refered problems avoiding multiple view geometry and calibration of several cameras.

One effective way to enhance the field of view of a camera is to use mirrors. The general approach of combining mirrors with conventional imaging systems is referred to as catadioptric image formation. In [6], Baker and Nayar derive the entire class of catadioptric systems with an unique viewpoint. Central catadioptric systems can be highly advantageous for many applications because they combine two important features: a single projection center and a wide field of view. Applications of these sensors in visual servoing, mainly for robot navigation purposes, appear in the literature [10, 13]. However a global theory for visual control of motion using central catadioptric images has never been proposed.

This work introduces the Jacobian matrix $\mathbf{J}$ for a generic central catadioptric system. Matrix $\mathbf{J}$ is derived from the central catadioptric mapping function presented in [8]. According to this unifying theory, central catadioptric imaging can be modeled by a generic function $\mathbf{f_i}$, with the type of sensor and shape of the mirror described by a parameter ξ. For the particular case of a conventional perspective camera the parameter ξ is null. Thus, by assuming $\xi = 0$, the general Jacobian matrix $\mathbf{J_g}$ becomes the well known interaction matrix $\mathbf{J_p}$ introduced the first time in [2]. Moreover it is shown that the derived Jacobian matrix can be decomposed in the product of two matrices $\mathbf{J_c}$ and $\mathbf{J_p}$ ($\mathbf{J_g} = \mathbf{J_c}.\mathbf{J_p}$). $\mathbf{J_c}$ is a 2×2 matrix that is always invertible which proves that the general catadioptric Jacobian $\mathbf{J_g}$ has exactly the same singularities as the standard perspective Jacobian $\mathbf{J_p}$ [11, 12].

Experiments on iterative pose estimation from points in the catadioptric image are performed. The singularities of $\mathbf{J_g}$ and the stability and convergence of image based visual servoing from catadioptric images are discussed. Point-to-contour tracking [3] on omnidireccional images is used to estimate the rigid displacement of objects. The application of the derived framework to control the position of a robotic arm is also discussed.

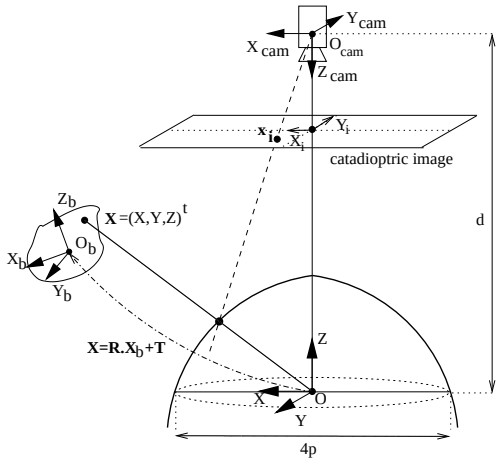


Figure 1: Central catadioptric projection of a rigid body

2 Modelling Central Catadioptric Image Formation

A catadioptric realization of omnidirectional vision combines reflective surfaces and lenses. In [5], Baker et al. derive the entire class of catadioptric systems verifying the fixed viewpoint constraint. The fixed viewpoint constraint is a requirement ensuring that the visual sensor only measures the intensity of light passing through a single point in 3D space. An unique projection center is a necessary condition for the generation of geometrically correct perspective images [5], and for the existance of epipolar geometry inherent to the moving sensor and independent of the scene structure [7]. A central catadioptric system can be built by combining a parabolic mirror with an orthographic camera or an hyperbolic, elliptical or planar mirror with a perspective camera.

Fig.1 is a scheme of the catadioptric system combining an hyperbolic reflective surface with a perspective camera. Consider the coordinate systems $\Re$ and $\Re_{\mathbf{cam}}$ associated respectively with the mirror and the perspective camera. The hyperbola axis is coincident with the Z-axis of $\Re$, and its foci are coincident with $\mathbf{O}$ and $\mathbf{O_{cam}}$ (the origins of $\Re$ and $\Re_{\mathbf{cam}}$). The latus rectum of the hyperbolic surface is $4p$ and the distance between the foci is d. Light rays incident with $\mathbf{O}$ (the inner focal point) are

reflected into rays incident with $\mathbf{O_{cam}}$ (the outer focal point). If the projection center of the perspective camera is coincident with $\mathbf{O_{cam}}$ the the captured light rays go originally through the inner focus of the hyperbolic surface. The effective viewpoint of the grabbed image is $\mathbf{O}$ and is unique. Elliptical catadioptric images are obtained combining an elliptical mirror with a perspective camera in a similar way. In the parabolic situation a parabolic mirror is placed such that its axis is the Z-axis, and its unique finite real focus is coincident with $\mathbf{O}$. Light rays incident with $\mathbf{O}$ are reflected into rays parallel with the Z-axis which are captured by an orthographic camera with image plane perpendicular to the Z-axis. The effective viewpoint is in $\mathbf{O}$ and is unique. A catadioptric system made up of a perspective camera steering a planar mirror also verifies the fixed viewpoint constraint. The effective projection center is behind the mirror in the perpendicular line passing through camera center. Its distance to the camera center is twice the distance between the planar mirror and the camera. Tab. 1 shows the equations of the different reflective surfaces.

In [8] Geyer and Daniilidis introduce an unifying theory for central catadioptric systems. Assume that a point with 3D coordinates $\mathbf{X} = (X, Y, Z)^t$ is projected in $\mathbf{x_i} = (x_i, y_i)^t$ in the catadioptric image plane (Fig. 1). It can be shown that central panoramic projection is isomorphic to a projective mapping from a sphere to a plane. Consider the scheme of Fig. 2 with the unitary sphere centered in the effective viewpoint $\mathbf{O}$, the point $\mathbf{O_c}$ with coordinates $(0, 0, -\xi)^t$ and the plane $Z = \psi - 2\xi$ orthogonal to the Z axis. Both ξ and ψ are function of mirror parameters d and p (Tab. 1). The projective ray $\mathbf{x}$ going through $\mathbf{X}$ intersects the spherical surface in $\mathbf{Q} = (\frac{X}{\rho}, \frac{Y}{\rho}, \frac{Z}{\rho})^t$ with $\rho = \sqrt{X^2 + Y^2 + Z^2}$. A second projective ray $\mathbf{x_c}$ can be defined by joining the intersection point $\mathbf{Q}$ with $\mathbf{O_c}$. The intersection $\mathbf{x_i}$ of the projective ray $\mathbf{x_c}$ with the plane $Z = \psi - 2\xi$ is the catadioptric image of the original point $\mathbf{X}$. Central catadioptric imaging can be modeled by projecting the scene in the sphere surface and then re-projecting these points in the image plane from a novel projection center $\mathbf{O_c}$. If the reflective surface is parabolic then $\xi = 1$ and the re-projection is a stereographic projection. For the hyperbolic and elliptical mirror the re-projection center $\mathbf{O_c}$ is inside the sphere in the negative Z axis. The planar mirror is a degenerate case of central catadioptric imaging with $\xi = 0$ and $\mathbf{O_c}$ coincident with the effective viewpoint $\mathbf{O}$. Notice that a catadioptric sensor with a planar mirror is equivalent to a conventional perspective camera with a sign inversion in the Y axis.

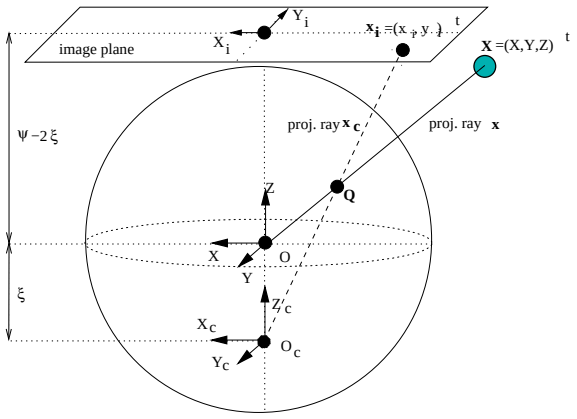


Figure 2: Modelling central catadioptric image formation

	Mirror Surface	ξ	ψ
Parabolic	$\sqrt{X^2+Y^2+Z^2} = 2p - Z$	1	$1+2p$
Hyperbolic	$\frac{(Z-\frac{d}{2})^2}{(\frac{1}{2}(\sqrt{d^2+4p^2}-2p))^2} - \frac{X^2+Y^2}{p(\sqrt{d^2+4p^2}-2p)} = 1$	$\frac{d}{\sqrt{d^2+4p^2}}$	$\frac{d+2p}{\sqrt{d^2+4p^2}}$
Elliptical	$\frac{(Z-\frac{d}{2})^2}{(\frac{1}{2}(\sqrt{d^2+4p^2}+2p))^2} + \frac{X^2+Y^2}{p(\sqrt{d^2+4p^2}+2p)} = 1$	$\frac{d}{\sqrt{d^2+4p^2}}$	$\frac{d-2p}{\sqrt{d^2+4p^2}}$
Planar	$Z = \frac{d}{2}$	0	1

Table 1: Column 1: Reflective surfaces for the different cases of central panoramic imaging. Column 2 and 3: Parameters ξ and ψ of the general central catadioptric model

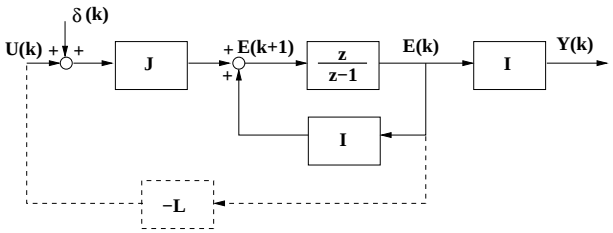


Figure 3: Iterative pose estimation is a regulation control problem. $\mathbf{I}$ is the $2n \times 2n$ identity matrix. The dashed line corresponds to the feedback loop.

$$\boldsymbol{f_i(X)} = \begin{pmatrix} \frac{f_x(\psi-\xi)X}{Z+\xi\sqrt{X^2+Y^2+Z^2}} - c_x \\ -\frac{f_y(\psi-\xi)Y}{Z+\xi\sqrt{X^2+Y^2+Z^2}} - c_y \end{pmatrix} \quad (1)$$

The catadioptric image is acquired by a camera steering the reflective surface. Assume that that the X and Y camera focal lengths are respectively f_x and f_y and $\mathbf{C} = (c_x, c_y)^t$ is the principal point. Equation 1 provides function $\mathbf{f_i}$ which maps points in the scene in the catadioptric image plane ($\mathbf{x_i} = \mathbf{f_i(X)}$). Any central catadioptric system with $\xi \neq 0$ can be easily calibrated from the image of three lines [9]. If the sensor calibration is known, function $\mathbf{f_i}$ can be simplified by making $f_x(\psi - \xi) = f_y(\psi - \xi) = 1$ and $c_x = c_y = 0$. We will assume without loss of generality that the mapping function is given by equation 2:

$$\boldsymbol{f_i(X)} = \begin{pmatrix} \frac{X}{Z+\xi\sqrt{X^2+Y^2+Z^2}} \\ -\frac{Y}{Z+\xi\sqrt{X^2+Y^2+Z^2}} \end{pmatrix} \quad (2)$$

3 Tracking and control

Fig. 1 depicts a moving rigid object observed by a central catadioptric sensor. The referential frame $\Re_\mathbf{b}$ is attached to the moving body, $\mathbf{R}$ is the rotation matrix between $\Re_\mathbf{b}$ and $\Re$ and $\mathbf{T}$ is the position of $\mathbf{O_b}$ in sensor coordinates. Our goal is to estimate the pose of the rigid body knowing the coordinates $\{\mathbf{X}_b^1, \mathbf{X}_b^2, \ldots, \mathbf{X}_b^n\}$ of a set of 'n' object points.

Let $\mathbf{X_b}$ be a generic point of the object model. If the pose $(\mathbf{R}, \mathbf{T})$ is known then the point 3D position in sensor coordinates is $\mathbf{X} = \mathbf{R}\mathbf{X_b} + \mathbf{T}$. From equation 2 it comes that point $\mathbf{X}$ is projected in $\mathbf{x_i} = \mathbf{f_i(X)}$ in the catadioptric image plane. Object rigid motion implies a change in pose that can be described by a kinematic screw $\delta = (\omega, \mathbf{v})^\mathbf{t}$. Consider the 3×6 matrix $\mathbf{J_m} = [\tilde{\mathbf{X}}|\mathbf{I}]$ where $\tilde{\mathbf{X}}$ is the skew-symetric matrix of $\mathbf{X}$ and $\mathbf{I}$ is the 3×3 identity matrix. The 3D velocity of point $\mathbf{X}$ due to object rigid motion is $\dot{\mathbf{X}} = \mathbf{J_m}\delta$. Moreover if $\mathbf{J_i}$ is the Jacobian matrix of function $\mathbf{f_i}$ (equation 2) and $\mathbf{J_g} = \mathbf{J_i}\mathbf{J_m}$ then the corresponding velocity in the catadioptric image plane is $\dot{\mathbf{x}}_\mathbf{i} = \mathbf{J_g}\delta$.

$$\boldsymbol{E} = \begin{bmatrix} \boldsymbol{x_i^1} - \boldsymbol{\hat{x}_i^1} \\ \boldsymbol{x_i^2} - \boldsymbol{\hat{x}_i^2} \\ \vdots \\ \boldsymbol{x_i^n} - \boldsymbol{\hat{x}_i^n} \end{bmatrix} \approx \begin{bmatrix} \boldsymbol{J_g^1} \\ \boldsymbol{J_g^2} \\ \vdots \\ \boldsymbol{J_g^n} \end{bmatrix} \hat{\boldsymbol{\delta}} = \boldsymbol{J}\hat{\boldsymbol{\delta}} \quad (3)$$

Let $\hat{\mathbf{s}} = \{\hat{\mathbf{x}}_i^1, \hat{\mathbf{x}}_i^2, \ldots, \hat{\mathbf{x}}_i^n\}$ be the set of model points projected in the image accordingly to a certain pose estimation $(\hat{\mathbf{R}}, \hat{\mathbf{T}})$, and $\mathbf{s} = \{\mathbf{x}_i^1, \mathbf{x}_i^2, \ldots, \mathbf{x}_i^n\}$ the real positions of those points. Vector $\mathbf{E}$ is defined as $\mathbf{E} = \mathbf{s} - \hat{\mathbf{s}}$ and depends on the pose estimation error described by the kinematic screw $\hat{\delta}$. From the above discussion it comes that $(\mathbf{x_i^j} - \hat{\mathbf{x}}_\mathbf{i}^\mathbf{j}) \approx \mathbf{J_g^j}\hat{\delta}$ with $j = 1, 2, \ldots, n$ and $\mathbf{J_g^j}$ the Jacobian matrix $\mathbf{J_g}$ evaluated on the j^{th} model point. Equation 3 establishes the relationship between the measured image error $\mathbf{E}$ and the error $\hat{\delta}$ on the pose estimation of the rigid body. $\mathbf{J}$ is a $2n \times 6$ matrix comprised by the Jacobian matrix $\mathbf{J_g}$ evaluated in the 'n' points of the object model. The objective is to update the pose estimation such that the image of the model becomes coincident with the object image and the measured error vector $\mathbf{E}$ converges to zero.

$$\begin{cases} \boldsymbol{E(k+1)} = \boldsymbol{I}\boldsymbol{E(k)} + \boldsymbol{J}\boldsymbol{\delta(k)} + \boldsymbol{J}\boldsymbol{U(k)} \\ \boldsymbol{Y(k)} = \boldsymbol{E(k)} \end{cases} \quad (4)$$

The problem stated in the previous paragraph can be formulated as a regulation control problem. Consider the system whose block diagram is depicted in Fig. 3. The state vector is the error $\mathbf{E(k)}$ measured in the catadioptric image, the system input matrix is $\mathbf{J}$ (equation 3), and the system output is $\mathbf{Y(k)}$ which must be zero. Accordingly to the system state-space equation 4 the pose change δ acts as a perturbation disturbing the output $\mathbf{Y(k)}$. The purpose is to find a state feedback controller $\mathbf{L}$ such that if $\mathbf{U(k)} = -\mathbf{LE(k)}$ then the disturbance is rejected and the system state converges to zero.

$$\boldsymbol{L} = (\boldsymbol{J^t J})^{-1}\boldsymbol{J^t} \tag{5}$$

$$\begin{cases} \boldsymbol{E(k+1)} = (\boldsymbol{I} - \boldsymbol{J}(\boldsymbol{J^t J})^{-1}\boldsymbol{J^t})\boldsymbol{E(k)} + \boldsymbol{J\delta(k)} \\ \boldsymbol{Y(k)} = \boldsymbol{E(k)} \end{cases} \tag{6}$$

The least squares solution of equation 3 is $\hat{\delta}(\mathbf{k}) = (\mathbf{J^t J})^{-1}\mathbf{J^t E(k)}$. $\hat{\delta}(\mathbf{k})$ is an estimate of the pose error associated with the measured image error $\mathbf{E(k)}$. System regulation can be achieved by making $\mathbf{U(k)} = \hat{\delta}(\mathbf{k})$ (equation 5). Equation 6 provides the state space model of the final closed loop system. System stability and transient response depend on the eigenvalues of the matrix $(\mathbf{I} - \mathbf{J}(\mathbf{J^t J})^{-1}\mathbf{J^t})$. However it is important to remind that the state transition matrix is a function of rigid body position which changes along time. Moreover the controller of equation 5 is only realizable when $(\mathbf{J^t J})$ is non singular. Whenever matrix $(\mathbf{J^t J})$ is not invertible we are in presence of a singularity.

4 The Jacobian Matrix for General Central Catadioptric Projection

To design the controller of equation 5 we need to obtain matrix $\mathbf{J}$ depending on the Jacobian matrix $\mathbf{J_g}$ which is evaluated on the 'n' points of the object model (equation 3).

Consider the central catadioptric mapping function $\mathbf{f_i}$ which maps 3D point coordinates $\mathbf{X}$ in image coordinates $\mathbf{x_i}$. The corresponding Jacobian matrix $\mathbf{J_i}$ is derived by diferentiating the function of equation 2. The achieved result is presented on equation 7 where $\rho = \sqrt{X^2 + Y^2 + Z^2}$.

Assume $\mathbf{J_i}$ provided by equation 7 and $\mathbf{J_m} = [\tilde{\mathbf{X}}|\mathbf{I}]$ with $\tilde{X}$ the skew symetric matrix associated with point coordinates $\mathbf{X}$. It was already shown that the Jacobian matrix $\mathbf{J_g}$ can be computed as $\mathbf{J_g} = \mathbf{J_i J_m}$. Equation 8 presents the general central catadioptric Jacobian matrix as a function of image position $\mathbf{x_i}$, point depth ρ and sensor ξ parameter ($\Upsilon = \sqrt{1 + (x_i^2 + y_i^2)(1 - \xi^2)}$). Notice that if $\xi = 0$ then matrix $\mathbf{J_g}$ becomes the well known Jacobian matrix $\mathbf{J_p}$ introduced in [2] for conventional perspective cameras.

The Jacobian $\mathbf{J_i}$ of the mapping function $\mathbf{f_i}$ can be decomposed in the matrix product of equation 9. $\mathbf{J_c}$ is the 2×2 matrix depending on point coordinates $\mathbf{X}$ and on the mirror parameter ξ. If $\xi = 0$ then $\mathbf{J_c}$ is the identity matrix. The second matrix has dimension 2×3 and it is the Jacobian matrix of the perspective mapping function $\mathbf{f_i} = (X/Z, -Y/Z)^t$ obtained making ξ equal to zero in equation 2. Thus the general catadioptric matrix $\mathbf{J_g}$ can be written as $\mathbf{J_g} = \mathbf{J_c J_p}$ with $\mathbf{J_p}$ the 2×6 Jacobian for the perspective camera situation. Moreover for $Z > 0$ the square matrix $\mathbf{J_c}$ is positive definite with eigenvalues $\{Z/(Z + \rho\xi); (Z^2(\rho + \xi Z))/(\rho(Z + \xi\rho)^2)\}$.

$$\boldsymbol{J} = \underbrace{\begin{bmatrix} \boldsymbol{J_c^1} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \boldsymbol{J_c^2} & \dots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \dots & \boldsymbol{J_c^n} \end{bmatrix}}_{\mathbf{C}} \underbrace{\begin{bmatrix} \boldsymbol{J_p^1} \\ \boldsymbol{J_p^2} \\ \vdots \\ \boldsymbol{J_p^n} \end{bmatrix}}_{\mathbf{P}} \tag{10}$$

The controller of equation 5 is realizable if and only if $\mathbf{J}$ is a full rank matrix. $\mathbf{J}$ has dimension

$$\boldsymbol{J_i} = \frac{1}{\rho(Z + \xi\rho)^2} \begin{bmatrix} \rho Z + \xi(Y^2 + Z^2) & -\xi XY & -X(\rho + \xi Z) \\ \xi XY & -(\rho Z + \xi(X^2 + Z^2)) & Y(\rho + \xi Z) \end{bmatrix} \tag{7}$$

$$\boldsymbol{J_g} = \begin{bmatrix} x_i y_i & \frac{(1+x_i^2)\Upsilon - y_i^2\xi}{\Upsilon + \xi} & y_i & \frac{1+x_i^2(1-\xi(\Upsilon+\xi))+y_i^2}{\rho(\Upsilon+\xi)} & \frac{x_i y_i \xi}{\rho} & -\frac{x_i \Upsilon}{\rho} \\ \frac{(1+y_i^2)\Upsilon - x_i^2\xi}{\Upsilon+\xi} & x_i y_i & -x_i & -\frac{x_i y_i \xi}{\rho} & \frac{1+x_i^2+y_i^2(1-\xi(\Upsilon+\xi))}{\rho(\Upsilon+\xi)} & -\frac{y_i \Upsilon}{\rho} \end{bmatrix} \tag{8}$$

$$\boldsymbol{J_i} = \underbrace{\begin{bmatrix} \frac{Z(\rho Z + \xi(Y^2+Z^2))}{\rho(Z+\xi\rho)^2} & \frac{\xi XYZ}{\rho(Z+\xi\rho)^2} \\ \frac{\xi XYZ}{\rho(Z+\xi\rho)^2} & \frac{Z(\rho Z + \xi(X^2+Z^2))}{\rho(Z+\xi\rho)^2} \end{bmatrix}}_{\mathbf{J_c}} \begin{bmatrix} \frac{1}{Z} & 0 & -\frac{X}{Z^2} \\ 0 & -\frac{1}{Z} & \frac{Y}{Z^2} \end{bmatrix} \tag{9}$$

$2n \times 6$ where 'n' is the number of considered model points. Clearly the full rank constraint can not be verified with less than three points. Equation 10 is derived from equation 3 knowing that $\mathbf{J_g^j} = \mathbf{J_c^j J_p^j}$. Matrix $\mathbf{J}$ is the product of a $2n \times 2n$ square matrix $\mathbf{C}$ with a matrix $\mathbf{P}$ with dimension $2n \times 6$. It was shown that $\mathbf{J_c^j}$ is positive definite for $j = 1, 2, \ldots, n$ and matrix $\mathbf{C}$ is always full rank. This poofs that $\mathbf{J}$ is rank deficient only when $\mathbf{P}$ is also rank deficient. The general central catadioptric situation does not present more singularities than the perspective case. These singularities were studied in [12, 3].

5 Tracking experiments and Conclusions

Based on the tracking method described above we implemented an object tracker. Since with catadioptric cameras straight lines map onto the image plane as quadrics, we devised a contour-to-point tracker along the lines described in [3]. The figures below show a rectangular object moving towards the camera and in a direction perpendicular to the camera. The advantage of this method is that only points along contours are to be found in the image thus avoiding the tedious and unreliable process of fitting a quadric to a set of points.

The model based tracking of a rigid object can be exploited in many ways for visual servoing applications. The proposed approach is being used in robot navigation and cooperation [13]. The experimental setup consists in two mobile plataforms both equipped with central catadioptric cameras. A visual landmark, similar to the one depicted in the figures, is positioned in the room ceil. One robot is the leader with independent motion and the other is the slave. The objective is to control slave motion such that the relative position between the two plataforms is kept constant. To achieve this goal both robots use the omnidirectional vision to estimate their pose from the model based tracking of the landamark. A method to control the position of a robotic arm using a static catadioptric system is also being develoed. Typically, in visual servoing using a conventional perspective camera, the available field of view is only enough to image the region around the end-effector. The pose

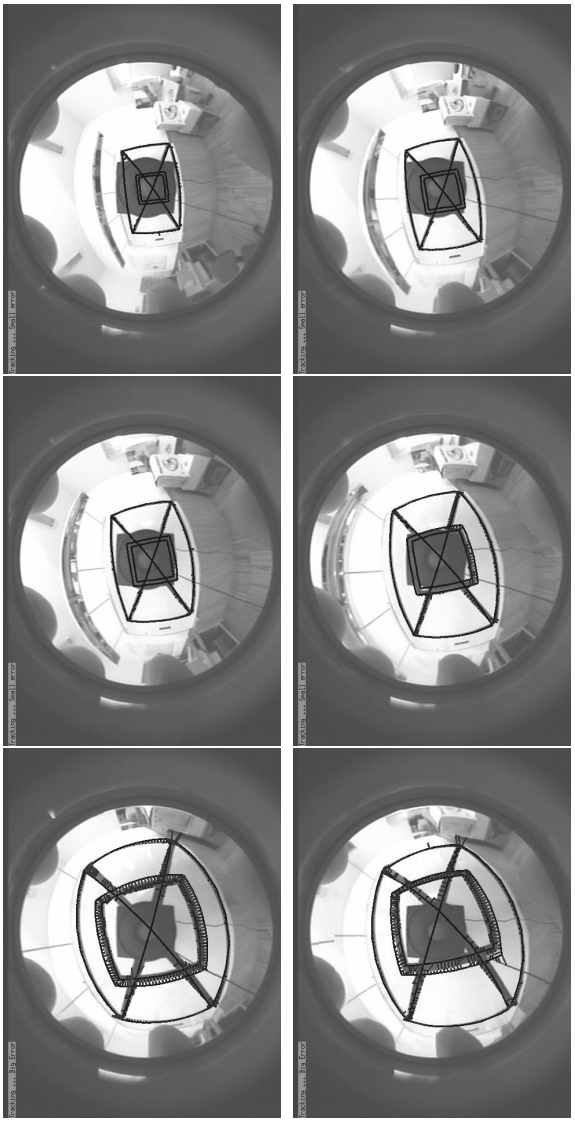


Figure 4: A tracking sequence. The object translates along axis of camera

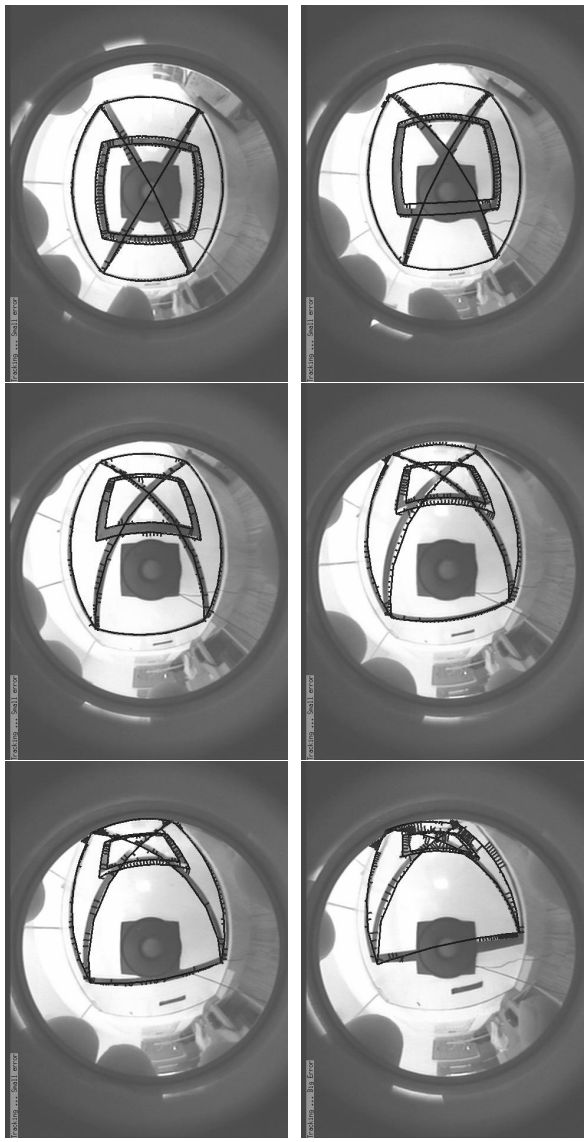


Figure 5: A tracking sequence. The object translates in front of camera..

of the end-effector is estimated by visual feedback, and motion control is achieved using the manipulator jacobian known "'a priori". The success of this approach is highly dependent on the arm calibration. We use the wide field of view provided by the omnidirectional sensor to image the entire arm. The different manipulator links are tracked in the catadioptric image and the motion of each joint is estimated. This approach increases the robustness and accuracy of the visual servoing.

References

[1] W. Wilson, C. Hulls, and G. Belles,"Relative end effector control using cartesian position-based visual servoing," *IEEE Trans. on Robotics and Automation*, vol. 12, no. 5, pp. 684–696, October 1996.

[2] B. Espiau, F. Chaumette, and P. Rives,"A new approach to visual servoing in robotics," *IEEE Trans. on Robotics and Automation*, vol. 8, no. 3, pp. 313–326, June 1992.

[3] F. Martin and R. Horaud, "Multiple Camera Tracking of Rigid Objects", *Research report 4268 INRIA*, Montbonnot, France, September 2001.

[4] S. Hutchinson, G. Hager, and P. I. Corke, "A tutorial on visual servo control," *IEEE Trans. on Robotics and Automation*, vol. 12, no. 5, pp. 651–670, October 1996.

[5] J. Gluckman and S. Nayar, " Egomotion and Omnidirectional Cameras," *ICCV98 - Proc. IEEE International Conference on Computer Vision*, pp. 999-1005, Bombay 1998.

[6] S. Baker and S. Nayar, "A Theory of Catadioptric Image Formation," *ICCV98 - Proc. IEEE International Conference on Computer Vision*, pp. 35-42, Bombay 1998.

[7] T. Svoboda, T. Pajdla and V. Hlavac, "Motion Estimation Using Central Panoramic Cameras," *Proc. IEEE Conference on Intelligent Vehicles, Stugart Germany 1998.*

[8] C. Geyer and K. Daniilidis, "A Unifying Theory for Central Panoramic Systems and Pratical Implications," *ECCV2000-Proc. European*

Conference on Computer Vision, pp. 445-461, Dublin 2000.

[9] Joao P. Barreto and H. Araujo,"Geometric Properties of Central catadioptric Line Images," *in Proc. of the European Conference on Computer Vision*, Copenhag, Denmark, May 2002.

[10] P. Baker, C. Fermuller, Y. Aloimonos and R. Pless,"A Spherical Eye From Multiple Cameras (Makes Better Models of the World)," *in Proc. of the IEEE Int. Conf. on Computer Vision and Pattern Recognition*, Kauai, Haway, USA, December 2001.

[11] Francois Chaumette, "Potential Problems of Stability and Convergence in Image Based and Position Based Visual Servoing", *The Confluence of Vision and Control*, Lecture Notes in Control and Information Systems, Vol. 237, pp 66-78, Springer-Verlag, 1998.

[12] H. Michel and P. Rives, "Singularities in the determination of the situation of a robot effector from the perspective view of 3 points", *Research report 1850 INRIA*, Sophia-Antipolis, France, February 1993.

[13] A. Paulino and H. Araujo, "Multiple Robots in Geometric Formation: Control Structure and Sensing", *in Int. Symposyum on Intelligent Robotic Systems*, Reading, UK, 2000.