Visually Guided Robots for Ship Building[*]

Radu Horaud
GRAVIR and INRIA Rhône-Alpes
655 av. de l'Europe, 38330 Montbonnot, FRANCE

Bart Lamiroy
GRAVIR and INRIA Rhône-Alpes
655 av. de l'Europe, 38330 Montbonnot, FRANCE

Tom Drummond
Engineering Department, University of Cambridge
Trumpington Street, Cambridge CB2 1PZ, UK

Ole Knudsen-Neckelmann
Development Department, Odense Steel Shipyard, Ltd. (OSS)
P.O. Box 176, 5100 Odense, DENMARK

January 25, 2000

Abstract

One of the most common operations in ship building is the alignment of a tool with respect to an assembly part. If the tool is mounted onto a robot manipulator then the problem is to properly control the latter such that the desired alignment is accurately achieved.

Among various sensors to be used in conjunction with robot control, vision has a number of advantages: it is low-cost, fast, covers a wide field of view and can operate in various configurations such as cameras mounted onto the robot itself or cameras mounted onto independent fixtures.

We briefly review the main components of a visually-guided robot system, we compare the relative merits of hand-held cameras and independent-held cameras and we illustrate the alignment of a welding torque with respect to a mock-up.

1 Introduction

1.1 Background and Motivations

The task of aligning a tool with an assembly part is representative of a class of tasks, which are difficult to achieve with the current robot technology. State-of-the-art robotic systems combine off-line task planning with on-line task execution. Task planning examples are motion planning and grasp planning. These planners produce collision-free trajectories if both the surrounding 3D layout and the robot are properly modeled. Task execution is done, in general in open-loop mode, which means that the geometric configuration of the layout at runtime is the same as

[*]This work is done cooperatively between INRIA Rhône-Alpes, The University of Cambridge and Odense Steel Shipyard, Ltd. (OSS) as part of the Esprit-IV reactive LTR project, number 26247 VIGOR.

the geometric configuration available at planning time. If there is a discrepancy between these two configurations, the robot will not be able to properly perform the task.

A natural step is to combine robot control with calibrated cameras. A more ambitious and innovative goal is to use uncalibrated cameras. Therefore, the most prominent aspect of this paper is to demonstrate that a technological step can be skipped. A transition can be made from CAD-based robot motions to visually controlled robot motions with no prior knowledge about the precise location of the robot with respect to various workpieces. This work was conducted within the Esprit-IV reactive LTR project, number 26247 VIGOR.

We are applying these results to the shipbuilding industry. Today's shipbuilding robot technology is limited to certain parts of the ship's hull and the existence of the off-line robot programming procedure prevents the user from taking advantage of dynamic response from the shop-floor to the robot motion planning. One way to deal with this problem is to introduce sensors that can determine the exact location of the object and then before program execution transform the co-ordinates in the robot program accordingly. This method however leaves us with a classical problem: If the robot is too complex (more than 6 axes) you have no guarantee that the transformed program is collision free and if the robot is simple (5-6 axes) you can not use it in complex areas of the ship. And with today's automation level it is actually in the complex area of the ship that the big savings are to be found, because of the high number of man hours spent there. It can be said that the competition battle today lies in the complex area of the ship. The classical problem described above can be overcome in two different but overlapping ways (they share the possible extensive use of advanced vision):

- The programming tool can be moved into the controller thereby making it possible to generate the CAD-based program after the sensor input. With the right programming tool, this allows us to use complex robots in complex environments. This is how OSS is trying to solve the problem today, and although they have come a long way already there still is a long way to go.

- The robots are entirely controlled by on-line sensors in a closed loop such as described in this proposal. This is a most interesting concept with more spin-offs than just the ones mentioned in the objectives. For instance quality control and 3D measurements are other very important issues which this project might contribute significantly to.

It should also be mentioned that the VIGOR solution, described in this paper, is superior in small confined rooms on board the ship where today a huge number of man-hours is spent. It is impossible to build up a classical robot device in these environments and only feasible solution will be based on an autonomous moving robot. The reason for the possible success of VIGOR in this area is that collision is nearly impossible (straight rectangular rooms with few objects to recognize and avoid). The number of target types is very few while the number of targets are enormous and so is the economic potential.

1.2 Paper Organization

In this paper we are more particularly interested by the comparison of two different approaches of visually guided robot servoing. Either the robot end-effector is equipped with a camera, and thus the camera moves as the robot aligns its tool at the required position, either an independent, fixed, set of cameras observes the workspace of the robot, and determines the trajectory the robot is to follow to achieve its task. The former is more commonly referred to as *eye-in-hand*, and will be discussed in the following section. The latter is a *fixed camera* approach, and will be discussed in section § 3.

2 Eye-in-hand configuration

This section presents a system which employs the eye-in-hand configuration and explores the benefits and implications of the approach. This configuration is characterized by placement of the camera in a location that is rigid relative to the tool in the robot's end-effector. This configuration yields two immediate benefits. These are:

- It is only necessary to track a single structure (the workpiece) in order to obtain information about the relative positions of the tool and that workpiece, since the tool is rigid relative to the camera.
- As the robot approaches the target position, the camera also approaches the workpiece and this provides greater precision in measurement of the relative pose between the camera and the workpiece.

2.1 System overview

The visual servoing system is shown in Figure 1. The video signal from the camera is fed into a tracking system (which contains a CAD model of the workpiece). This system maintains an estimate of the pose of the workpiece relative to the camera updated at video frame rate (25Hz). This information is then fed into the robot control system which compares the current pose with a previously learned target pose in order to generate robot control commands.

2.2 Theoretical framework

The approach employed here for tracking known 3-dimensional structures is based upon maintaining an estimate of the camera projection matrix, P, in the co-ordinate system of the structure. This projection matrix is represented as the product of a matrix of internal camera parameters:

$$K = \begin{bmatrix} f_u & s & u_0 \\ 0 & f_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{1}$$

and a Euclidean projection matrix representing the position and orientation of the camera relative to the target structure:

$$E = \begin{bmatrix} R & t \end{bmatrix} \qquad \text{with } RR^T = I \text{ and } |R| = 1 \tag{2}$$

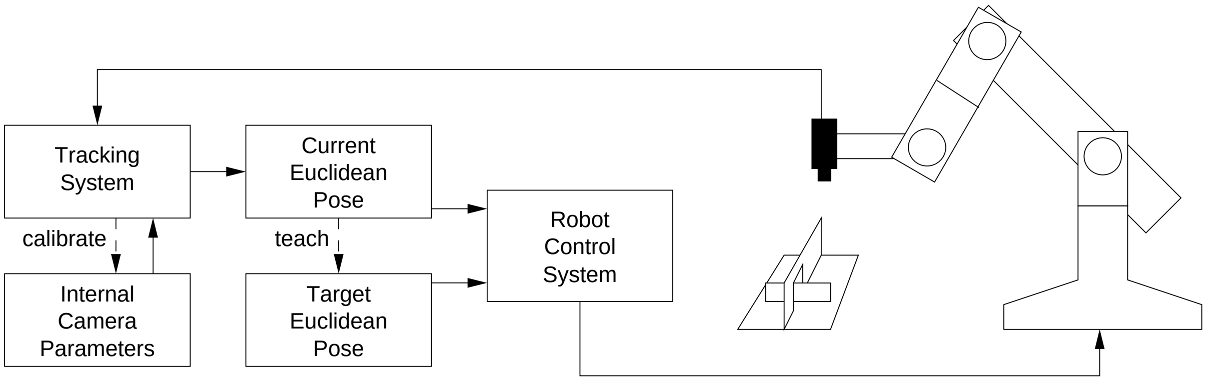

Figure 1: Visual servoing system

The projective co-ordinates of an image feature are then given by

$$\begin{pmatrix} u \\ v \\ w \end{pmatrix} = P \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \tag{3}$$

with the actual image co-ordinates given by

$$\begin{pmatrix} \tilde{u} \\ \tilde{v} \end{pmatrix} = \begin{pmatrix} u/w \\ v/w \end{pmatrix} \tag{4}$$

Rigid motions of the camera relative to the target structure between consecutive video frames can then be represented by right multiplication of the projection matrix by a Euclidean transformation of the form:

$$M = \begin{bmatrix} R & t \\ 0\;0\;0 & 1 \end{bmatrix} \tag{5}$$

These M, form a 4×4 matrix representation of the group SE(3), which is a 6-dimensional Lie Group. The generators of this group are typically taken to be translations in the x, y and z directions and rotations about the x, y and z axes, represented by the following matrices:

$$G_1 = \begin{bmatrix} 0&0&0&1 \\ 0&0&0&0 \\ 0&0&0&0 \\ 0&0&0&0 \end{bmatrix}, G_2 = \begin{bmatrix} 0&0&0&0 \\ 0&0&0&1 \\ 0&0&0&0 \\ 0&0&0&0 \end{bmatrix}, G_3 = \begin{bmatrix} 0&0&0&0 \\ 0&0&0&0 \\ 0&0&0&1 \\ 0&0&0&0 \end{bmatrix}, \tag{6}$$
$$G_4 = \begin{bmatrix} 0&0&0&0 \\ 0&0&1&0 \\ 0&-1&0&0 \\ 0&0&0&0 \end{bmatrix}, G_5 = \begin{bmatrix} 0&0&-1&0 \\ 0&0&0&0 \\ 1&0&0&0 \\ 0&0&0&0 \end{bmatrix}, G_6 = \begin{bmatrix} 0&1&0&0 \\ -1&0&0&0 \\ 0&0&0&0 \\ 0&0&0&0 \end{bmatrix}$$

These generators form a basis for the vector space (the Lie algebra) of derivatives of SE(3) at the identity. Consequently, the partial derivative of projective image co-ordinates under the ith generating motion can be computed as:

$$\begin{pmatrix} u' \\ v' \\ w' \end{pmatrix} = PG_i \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \tag{7}$$

with

$$L_i = \begin{pmatrix} \tilde{u}' \\ \tilde{v}' \end{pmatrix} = \begin{pmatrix} \frac{u'}{w} + \frac{uw'}{w^2} \\ \frac{v'}{w} + \frac{vw'}{w^2} \end{pmatrix} \tag{8}$$

giving the motion in true image co-ordinates. A least squares approach can then be used to fit the observed motion of image features between adjacent frames. This process is detailed in Section 2.3.3.

In a similar manner, the motion of features in the image due to the change of internal camera parameters can be computed and these motion fields incorporated into the least squares solution.

2.2.1 Tracking edges

An important aspect of the approach presented here is the decision to track the edges of the model (which appear as intensity discontinuities in the video feed). Edges are strong features that can be reliably found in the image because they have a significant spatial extent. Furthermore, this means that a number of measurements can be made along each edge, and thus they may be accurately localized within an image.

This approach also takes advantage of the aperture problem (that the component of motion of an edge, tangent to itself, is not observable locally). This problem actually yields an enormous benefit since the search for intensity discontinuities in the video image can be limited to a one dimensional path that lies along the edge normal, $\hat{n}$ (see Figure 2) and thus has linear complexity in the search range, rather than quadratic which makes it possible to track complex structures in real time on a standard workstation without additional hardware. The normal component of the motion fields, L_i are then also computed (as $L_i \cdot \hat{n}$).

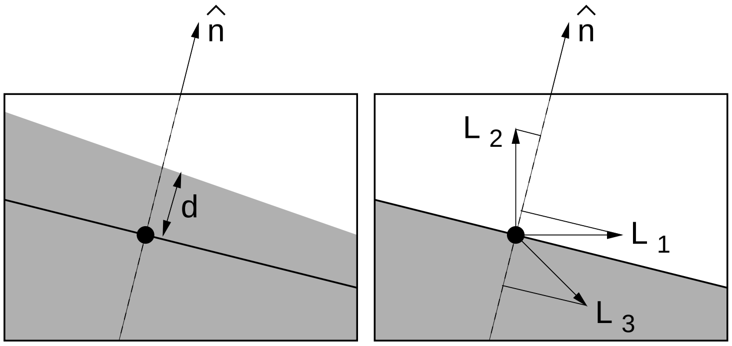

Figure 2: Computing the normal component of the motion

2.3 Tracking system

The three-dimensional tracking system makes use of constrained snake technology [1] to the follow edges of the workpiece that are visible in the video image. One novel aspect of this work is the use of a real-time hidden-line-removal rendering system (using binary space partition trees [2]) to dynamically determine the visible features of the model in real-time. This technique allows accurate frame rate tracking of complex structures such as the ship part shown in Figure 4.

Figure 3 shows system operation. At each cycle, the system renders the expected view of the object (a) using its current estimate of the projection matrix, P. The visible edges are identified and tracking nodes are assigned at regular intervals in image co-ordinates along these edges (b). The edge normal is then searched in the video feed for a nearby edge (c). Typically $m \approx 400$ nodes are assigned and measurements made in this way. The system then projects this m-dimensional measurement vector onto the 6-dimensional subspace corresponding to Euclidean transformations (d) using the least squares approach described in Section 2.3.3 to give the motion, M. The Euclidean part of the projection matrix, E is then updated by right multiplication with this transformation (e). Finally, the new projection matrix P is obtained by multiplying the camera parameters K with the updated Euclidean matrix to give a new current estimate of the local position (f). The system then loops back to step (a).

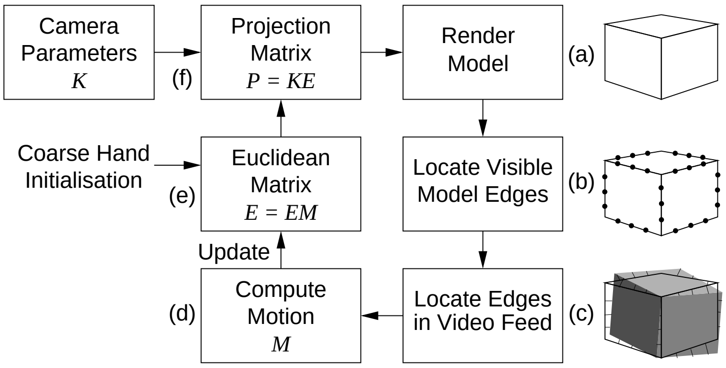

Figure 3: Tracking system operation

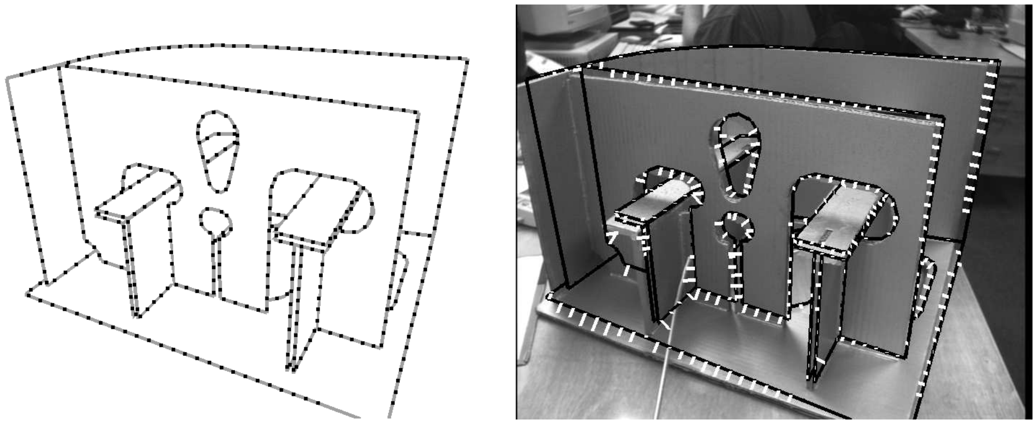

Figure 4: Model of ship part and image with tracking nodes assigned and distances measured

2.3.1 Rendering the model

In order to accurately render a CAD model of a complex structure such as the one shown in Figure 4 at frame rate, an advanced rendering technique such as the use of binary space partition trees is needed [2]. This approach represents the object as a tree, in which each node contains the equation of a plane in the model, together with a list of edges and convex polygons in that plane. Each plane partitions 3-dimensional space into the plane and the two open regions either side of the plane. The two branches of the tree represent those parts of the model that fall into these two volumes. Thus the tree recursively partitions space into small regions which, in the limit, contain no remaining model features. The rendering takes place by performing an *in-order* scan of the tree, where at each node, the viewpoint is tested to see if it lies in front, or behind the plane. When this is determined, those features lying closer to the camera are rendered first, then the plane itself, and finally, the more distant features. The use of a stencil buffer prevents over-writing of nearer features by more distant ones and also provides a layer map when the rendering is complete. The ship part contains 12 planes, but since 8 of these (corresponding to the T and L beams) are split into two parts by a vertical plane partition, there are 20 nodes in the tree.

2.3.2 Locating edges

Once rendering is complete, the layer map is used to locate the visible parts of each edge by comparing the assigned layer of the plane for each edge in the model with the layer in the stencil buffer at a series of points along that edge. Where the depths agree, trackers are assigned to search for the nearest edge in the video feed along the edge normal (see Figure 4).

The result of this process is a set of trackers with known position in the model co-ordinate system, with computed edge normals and the distance along those normals to the nearest image edge. Grouping these distances together provides an m-dimensional measurement vector.

2.3.3 Computing the motion

Step (d) in the process involves the projection of the measurement vector onto the subspace defined by the Euclidean transformation group. The action of each of the generators of SE(3) on the tracking nodes in image co-ordinates can be found by computing PG_i and applying this to the homogeneous co-ordinates of the node in 3-space. This can be projected to give a vector, L_i^ξ describing the image motion of the ξth node for the ith generator of Euclidean motion of the object. $L_i^\xi \cdot \hat{n}^\xi$ then describes the magnitude of the edge normal motion that would be

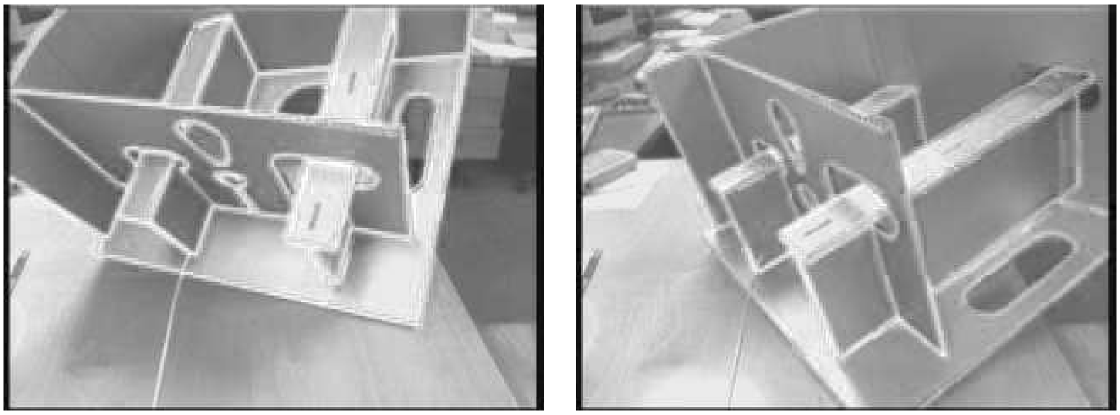

Figure 5: Frames from tracking sequence

observed in the image at each node for each group generator. These can be considered as a set of m-dimensional vectors which describe the motion in the image for each mode of Euclidean transformation. The system then projects the m-vector corresponding to the measured distances to the observed edges onto the subspace spanned by the transformation vectors. This provides a solution to finding the geometric transformation of the part which best fits the observed edge positions, minimizing the square error between the transformed edge position and the actual edge position (in pixels). This process is performed as follows:

$$O_i = \sum_{\xi} d^{\xi}(L_i^{\xi} \cdot \hat{n}^{\xi}) \tag{9}$$

$$C_{ij} = \sum_{\xi} (L\xi_i \cdot \hat{n}^{\xi})(L_j^{\xi} \cdot \hat{n}^{\xi}) \tag{10}$$

$$\alpha_i = C_{ij}^{-1} O_j \tag{11}$$

(with Einstein summation convention over Latin indices). The α_i then contain the quantity of each mode of Euclidean motion that has been observed. The final step is to compute the actual motion of the model and apply it to the matrix E in (2). This is done by using the exponential map connecting α_i with SE(3).

$$E_{t+1} = E_t \exp(\textstyle\sum_i \alpha_i G_i) \tag{12}$$

2.4 Robot control system

The robot control system takes the Euclidean matrix, E as output from the tracking system and uses this within a non-linear control law to provide feedback to servo the robot to a stored target position. These are learned using teaching-by-showing where the target Euclidean matrix is acquired when the robot is placed in the target position by the supervisor. The inverse of this target matrix, E_t^{-1}, is easily computed and the product of this with the current position matrix yields the transformation from the target position to the current position.

$$T = EE_t^{-1} \tag{13}$$

The translation and rotation vectors that must be applied to the robot are then easily extracted from this representation. (here i, j, k = 1,2,3):

$$t_i = T_{i4} \tag{14}$$

$$r'_i = \frac{1}{2} \sum_{jk} \epsilon_{ijk} T_{jk}$$

$$r_i = \frac{r'_i \sin^{-1}(|r'|)}{|r'|} \tag{15}$$

The vectors t and r are then multiplied by a gain factor and sent to the robot as end effector translation and rotation velocities. The gain is dependent on the magnitudes of t and r so that small velocities are damped to obtain higher precision, while large errors in position may be responded to quickly. A maximum velocity clamp is also applied for safety reasons and to prevent possible instabilities due to latency.

2.5 Results

The tracking system and visual servoing system have been tested in a number of experiments to assess their performance both quantitatively and qualitatively. These experiments were conducted with an SGI O2 workstation (225 MHz) controlling a Mitsubishi RV-E2 robot.

2.5.1 Stability of the tracker with respect to image noise

The stability of the tracker with a stationary structure was measured to assess the effect of image noise on the tracker. The standard deviation of position and rotation as measured from the Euclidean matrix were measured over a run of 100 frames. From a viewing distance of 30cm, the apparent r.m.s. translational motion was found to be 0.03mm with the r.m.s. rotation being 0.015 degrees.

2.5.2 Accuracy of positioning

The accuracy of positioning the robot was measured with two experiments. Firstly, the ship part was held fixed and the robot asked to home to a given position from a number of different starting points. When the robot had ceased to move, the program was terminated and the robots position queried. The standard deviation of these positions was computed and the r.m.s. translational motion was 0.08mm with the r.m.s. rotation being 0.06 degrees.

The second accuracy experiment was performed by positioning the ship part on an accurate turntable. The part was turned through fifteen degrees in one degree rotations and the robot asked to return to the target position each time. Again, the position of the robot was queried and a circle was fitted to the data. The residual error was computed and found to give an r.m.s. positional error of 0.12mm per measurement (allowing for the three degrees of freedom absorbed into fitting the circle).

2.6 Calibration issues

There are a number of calibration issues that must be addressed in the eye-in-hand configuration. First, since only a single camera is used, the internal parameters of the camera must be known in order to obtain stable, high precision performance. Second, the transformation between the camera coordinate frame and that of the robot end-effector must be known in order to derive robot control commands from visual data. Finally, the transformation between the camera and the tool must be modeled in some way.

The system presented in this Section includes a facility for on-line calibration of the internal camera parameters (focal length, aspect ratio and principal point). The camera to tool transformation is modeled implicitly using the teaching-by-showing approach, while the camera to end-effector transformation must be supplied as a parameter.

3 Fixed Camera Configuration

This section presents a fixed camera configuration. One or more cameras are placed in such a way as to cover the workspace of an observed robot. A known 3D model of some visible features of the end-effector (in our case four highly reflective control patches) allow for an accurate localization of the robot with respect to the observing cameras. The knowledge of a required goal position of these features allow us to guide to robot to that location with a closed-loop visual guidance algorithm.

With respect to the *hand-in-eye* approach we identify the following advantages:

1. The tracking and identification problem is greatly alleviated since the cameras observe the whole scene, and only need to identify the robot effector (which can be easily marked, compared to the whole environment used in the previous approach).
2. There is no need in constantly computing the 3D relationship between the cameras and the whole environment, since that relationship remains fixed. The only needed relationship that is easily computed is the pose of the robot effector with respect to the cameras.
3. There is no need for a precise modeling of the work piece or the environment, once the work piece has been identified in the images only the robot needs to be tracked.
4. The cameras are not mounted near the effector tool, so there is less risk in collision or deterioration of the vision hardware.

This goes however at the expense of a certain loss in precision, since the cameras are further off. On the other hand, the method can cope with a wider range of amplitude for the robot movement.

3.1 Hardware setup

In our setup, we're controlling a Stäubli RX90 6DOF Puma-like robot (6 revolute joints) through a Sinters 5DOF Cartesian robot (3 prismatic joints, 2 revolute joints) carrying vision hardware (grey level stereo rig).

Both robots are controlled by a dedicated VME rack running VXWorks each. End-user control is usually done through socket connections from a workstation. A DBV44 card on the Cartesian robot rack allows real-time image processing at approx. 10 Hz. Network latency and low image acquisition rates result in a 2Hz high level control loop.

This setup is currently being ported on one single PC running Linux and containing a 25Hz image acquisition card, which should drive the high level control loop up to more than 10Hz.

3.2 Visual Servoing: a Theoretical Framework

The visual servoing problem we're addressing here was developed in [3] for the monocular case, and recently extended [4] for the case where a stereo rig is used for visual control. Since the stereo case is a quite straightforward extension of the monocular case (once all theoretical verifications have been made), we're assuming, for simplicity, that only one camera is used.

The problem that has to be resolved is: *given a current observed 2D position* $\mathbf{s}_t$ *of a well known robot, and given a known 2D goal position* $\mathbf{s}^\star$*, control the robot* $\mathcal{R}$ *to a 3D position so that its observed 2D position coincides with* $\mathbf{s}^\star$.

We suppose the following parameters are known:

1. The inverse kinematics of the robot $\mathcal{R}$. More precisely, let $\mathbf{q} \in I\!R^n$ be the vector of the articular position of the robot, and let $\mathbf{r} \in I\!R^6$ be the Cartesian position of the end effector with respect to the local coordinate frame. Then the inverse kinematics or $6 \times n$ Jacobian matrix of the robot $\mathbf{J}_\mathcal{R}$ is defined as $\frac{\partial \mathbf{r}}{\partial \mathbf{q}}$. In our case, the RX90 robot has 6 DOF. Therefore, $n = 6$.
2. The 3D position of four control points in the reference frame of the robot end effector.
3. The intrinsic camera parameters of a control camera $\mathcal{C}$. The relationship between the image coordinate system and the local Cartesian robot reference frame is given by

$$\begin{pmatrix} \sigma u \\ \sigma v \\ \sigma \end{pmatrix}_\mathcal{I} = \begin{pmatrix} \alpha_u & 0 & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{pmatrix} (\mathbf{R}\ \mathbf{t}) \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}_\mathcal{R}$$

α_u, α_v, u_0 and v_0 are referred to as the intrinsic camera parameters, while $\mathbf{R}$ and $\mathbf{t}$ are referred to as the external camera parameters, or pose.

Furthermore, we know that the speed of the 2D projection $(u\ v)^\mathsf{T}$ in the image of a 3D point $\boldsymbol{P} = (X\ YZ)^\mathsf{T}$ expressed in the camera reference frame is given by the following formula:

$$\begin{pmatrix} \dot{u} \\ \dot{v} \end{pmatrix} = \begin{pmatrix} \alpha_u & 0 \\ 0 & \alpha_v \end{pmatrix} \begin{pmatrix} \frac{1}{Z} & 0 & -\frac{X}{Z^2} & -\frac{XY}{Z^2} & 1 + \frac{X^2}{Z^2} & -\frac{Y}{Z} \\ 0 & \frac{1}{Z} & -\frac{Y}{Z^2} & -\left(1 + \frac{Y^2}{Z^2}\right) & \frac{XY}{Z^2} & \frac{X}{Z} \end{pmatrix} \begin{bmatrix} \boldsymbol{V}_\mathcal{C} \\ \Omega_\mathcal{C} \end{bmatrix}$$

Where $[\boldsymbol{V}_\mathcal{C}\ \Omega_\mathcal{C}]^\mathsf{T}$ is the kinematic screw, applied to the point in the reference frame of the camera. We know that this kinematic screw can be expressed in the robot reference frame using the following transformation:

$$\begin{bmatrix} \boldsymbol{V}_\mathcal{C} \\ \Omega_\mathcal{C} \end{bmatrix} = \begin{pmatrix} \mathbf{R} & -\mathbf{R}S\left(-\mathbf{R}^\mathsf{T}\mathbf{t}\right) \\ 0 & \mathbf{R} \end{pmatrix} \begin{bmatrix} \boldsymbol{V}_\mathcal{R} \\ \Omega_\mathcal{R} \end{bmatrix}$$

Which finally gives

$$\begin{pmatrix} \dot{u} \\ \dot{v} \end{pmatrix} = \mathbf{J}\left(\boldsymbol{P}\right) \begin{bmatrix} \boldsymbol{V}_\mathcal{R} \\ \Omega_\mathcal{R} \end{bmatrix}$$

Where $\mathbf{J}\left(\boldsymbol{P}\right)$ is the image Jacobian at the point $\boldsymbol{P}$.

Now, returning to out initial problem, our aim is to make the currently observed position of the robot $\mathbf{s}$ coincide with the goal position $\mathbf{s}^\star$. The most straightforward way for obtaining this, and guaranteeing an exponential convergence, is to have $\mathbf{s}$ move in the direction of $\mathbf{s}^\star$. In other words, have $\dot{\mathbf{s}} = g\left(\mathbf{s}^\star - \mathbf{s}\right)$. Applying the previous equations this can be rewritten as

$$\mathbf{J} \begin{bmatrix} \boldsymbol{V}_\mathcal{R} \\ \Omega_\mathcal{R} \end{bmatrix} = g\left(\mathbf{s}^\star - \mathbf{s}\right)$$

$\mathbf{J}$ being composed of the image Jacobian matrices formed by the observed 3D points through $\mathbf{s}$. In order to obtain the kinematic screw that has to be applied to the robot end effector, we take the pseudo-inverse of $\mathbf{J}$, and thus obtain

$$\begin{bmatrix} \boldsymbol{V}_{\mathcal{R}} \\ \Omega_{\mathcal{R}} \end{bmatrix} = \left(\mathbf{J}^{\top}\mathbf{J}\right)^{-1} \mathbf{J}^{\top} g \left(\mathbf{s}^{\star} - \mathbf{s}\right)$$

Using this information we can infer a closed servo loop by constantly observing the set of control points $\mathbf{s}(t)$, and by adequately updating the kinematic screw controlling the robot, until the observed position coincides with the required goal position $\mathbf{s}^{\star}$.

3.3 Computing the Goal Position

The remaining difficulty resides in the fact that the goal position that needs to be attained by the robot-effector is defined with respect to the considered work-piece (*e.g.* bolt position, welding joint, *etc.*) and that the actual position of this work-piece is unknown due to unprecise positioning of the object itself or due to discrepancies between the CAD model and the real object.

The proposed solution is depicted in Figure 6, and consists of three main phases. We suppose we dispose of a 3D representation of the required goal position in some canonical frame. Either this is CAD given or it consists of a previously observed real situation (Obtaining a projective 3D reconstruction from a set of points, observed through a stereo rig is a classical problem in computer vision [5]).

Phase 1: We establish a relationship between the reference 3D representation and the currently observed projective 3D scene reconstruction. We know that this relationship is a 4×4 homography $\mathbf{H}$.

Phase 2: Since we know the required 3D position of the robot-effector in the first representation we can use the obtained homography $\mathbf{H}$ to compute the goal position in the current 3D representation, and back project this goal position in the images.

Phase 3: This back-projection actually defines $\mathbf{s}^{\star}$ and allows us to apply the previously described visual servoing algorithm.

3.4 Computing H

Recent work by R. Horaud *et al.* [6], offers a robust method for computing the homography relating two projective reconstructions obtained after a rigid motion of either the scene, or, in a dual way, the stereo rig.

Let $\mathcal{S} = \{\boldsymbol{M}_1, \dots, \boldsymbol{M}_n\}$ be the set of n 3D points composing a scene. We dispose of two stereo pairs of this scene $\mathcal{S}$, one, the reference pair, taken at time t_0, another the observed pair at the time of execution t_{ex}. Between t_0 and t_{ex} $\mathcal{S}$ has undergone an unknown rigid motion (the rig may have undergone a non rigid motion in-between).

Let $\mathcal{S}_1 = \left\{\boldsymbol{M}_1^1, \dots, \boldsymbol{M}_n^1\right\}$ and $\mathcal{S}_2 = \left\{\boldsymbol{M}_1^2, \dots, \boldsymbol{M}_n^2\right\}$ be the 3D projective reconstructions resulting from both stereo pairs, respectively expressed in two different bases $\mathcal{B}_1$ and $\mathcal{B}_2$ of $I\!P^3$. We know there exists a homography $\mathbf{H}$ mapping $\mathcal{S}_1$ on $\mathcal{S}_2$ such that

$$\mu_i \boldsymbol{M}_i^2 = \mathbf{H}\boldsymbol{M}_i^1 \qquad \text{with} \qquad \mu_i \neq 0, \forall i$$

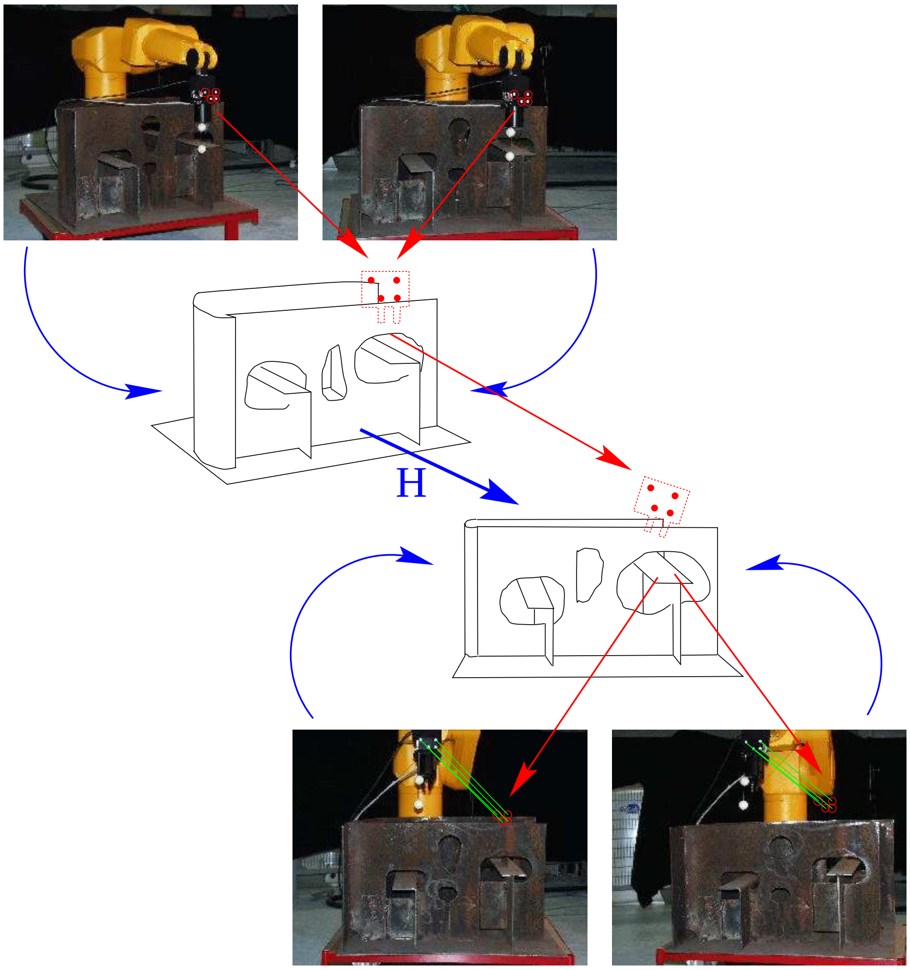

Figure 6: The INRIA VIGOR Demonstrator, Three Main Phases.

First Phase (in blue): two cameras observe a manufactured object (in our case, the OSS mock-up), and allow the computation of a 3D projective reconstruction of it (top). The object is moved with respect to the cameras, and another 3D reconstruction is computed (bottom). Both reconstructions are in different reference frames, but it is possible to obtain the homography projecting the first frame onto the other (**H**). We assume that the first reconstruction contains the reference position of the RX90 end effector, while in the second one the robot is in a random "*off-line*" position.

Second Phase (in red): the reference position of the effector is extracted from the first stereo pair and reconstructed in the first reference frame. Then **H** is applied to its 3D projective position to obtain the reference position in the second reference frame. Back projection into the second stereo pair gives the wanted 2D position for the new configuration (Transfer and back-projection are fairly straightforward and computation is robust as long as the normalizations, described in [5], have been applied).

Third Phase (in green): the RX90 is visually guided from its current "*off-line*" position to the thus computed reference position.

$\mathbf{H}$ has 15 DOF. In order to determine $\mathbf{H}$, we need at least five point correspondences between $\mathcal{S}_1$ and $\mathcal{S}_2$, since each correspondence $\left(\boldsymbol{M}_i^1, \boldsymbol{M}_i^2\right)$ gives us 3 constraints after elimination of μ_i.

Five points $\{\boldsymbol{P}_i\}_{i=1\ldots5}$ generally define a projective basis of $\mathbb{P}^3$, and the homography mapping the standard projective basis to this set of points is given by the matrix $\mathbf{T}$ such that

$$\mathbf{T} = (\lambda_1 \boldsymbol{P}_1 \;\; \lambda_2 \boldsymbol{P}_2 \;\; \lambda_3 \boldsymbol{P}_3 \;\; \lambda_4 \boldsymbol{P}_4)$$
$$(\lambda_1 \; \lambda_2 \; \lambda_3 \; \lambda_4)^{\mathsf{T}} = (\boldsymbol{P}_1 \; \boldsymbol{P}_2 \; \boldsymbol{P}_3 \; \boldsymbol{P}_4)^{-1} \boldsymbol{P}_5$$

We therefor easily conclude that

$$\mathbf{H} = \mathbf{T}_2 {\mathbf{T}_1}^{-1}$$

where $\mathbf{T}_1$ (resp. $\mathbf{T}_2$) be the transformations taking $\mathcal{B}_1$ (resp. $\mathcal{B}_2$) to the canonical projective basis of $\mathbb{P}^3$.

4 Results

In order to measure the quality of the final visual servoing, we use the duality principle between a moving camera and a moving scene. *I.e.*, instead of using a fixed stereo rig, and moving the OSS mock-up, we move the stereo rig. Visually servoing the RX90 to its required position is strictly equivalent than in the case where the mock-up moves. The sole difference is that we can register the RX90's position in its joint space, and measure the error that has been made after it has been servoed home.

4.1 Repeatability

Measured 3D repeatability of visually homing the robot to a registered position in the image (*i.e.* without introduction of numerical instabilities due to the computation of the goal position) shows that the RX90 is capable of returning to its position with a precision of less than 0.1 degrees per joint, resulting in a less than 1mm positioning error in 3D Cartesian space. Image precision is 0.1 pixels per control point, seen from 2m distance.

Note: the currently implemented controller is quite basic, and does not yet take into account the dynamics of the RX90 due to its weight *etc.*.

4.2 3D Convergence

Measured 3D convergence is a direct function of the quality of $\mathbf{H}$. The disposition of the points used for computing $\mathbf{H}$ greatly influence the quality of the reprojected goal points. Observed 3D convergence varies therefore between 1.0 and 0.2 degrees per joint, resulting in a Cartesian positioning error between 10 and 1.5 mm for an image precision of 0.1 pixels per control point, seen from 2m distance.

4.3 Control

Figure 7 and Figure 8 show the evolution of the different control parameters of a typical servo task. Figure 7 shows the convergence towards the goal position in 6D joint space or 3D Cartesian space (ignoring rotation). Figure 8 shows the convergence in the image of each of the control points.

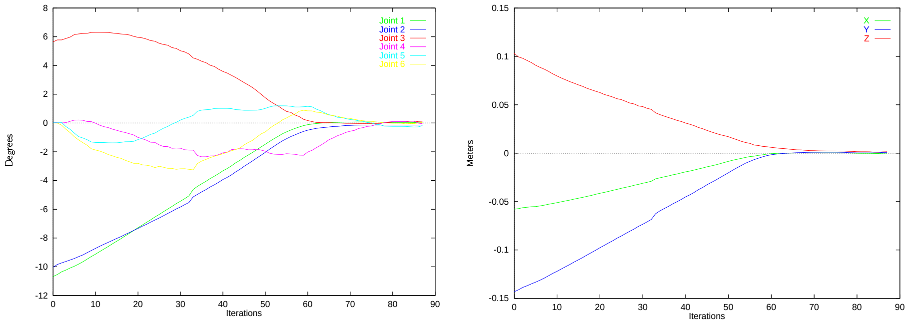

Figure 7: 3D convergence in joint space (*left)* and in Cartesian space (*right*).

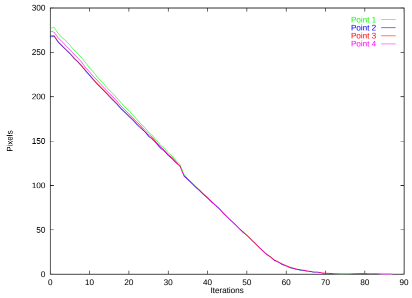

Figure 8: 2D convergence for each of the image points.

Conclusions

We have described a series of tools allowing efficient and robust visual control of robots. We have shown that there are a great number of practical and cost-saving applications without loss of precision or robustness and with a substantial gain in time. We are currently applying them in the ship industry, through an important technological transfer with Odense Steel Shipyard, Ltd. (OSS) within the VIGOR project. The use of visual guidance introduces a higher degree of flexibility and lowers the number of constraints of an industrial production line.

Bibliography

[1] R. Cipolla and A. Blake. Image divergence and deformation from closed curves. *International Journal of Robotics Research*, 16(1):77–96, 1997.

[2] M. Paterson and F. Yao. Efficient binary space partitions for hidden surface removal and solid modeling. *Discrete and Computational Geometry*, 5(5):485–503, 1990.

[3] R. Horaud, F. Dornaika, and B. Espiau. Visually guided object grasping. IEEE *Transactions on Robotics and Automation*, 14(4), August 1998.

[4] B. Lamiroy, B. Espiau, N. Andreff, and R. Horaud. Controlling robots with two cameras: How to do it properly. In *Proc. IEEE Int. Conf. on Robotics and Automation, San Fransisco, California, USA, April 24–28*, 2000.

[5] R. Hartley and P. Sturm. Triangulation. *Computer Vision and Image Understanding*, 68(2):146–157, 1997.

[6] G. Csurka and R. Horaud. Finding the collineation between two projective reconstructions. Technical Report 3468, INRIA, August 1998.