2007 IEEE International Conference on
Robotics and Automation
Roma, Italy, 10-14 April 2007

Calibration Free Image Point Path Planning Simultaneously Ensuring Visibility and Controlling Camera Path

Florian Schramm[♮], Franck Geffard[⋆], Guillaume Morel[♮] and Alain Micaelli[⋆]

Université Pierre et Marie Curie, Paris6
[♮]Laboratoire de Robotique de Paris, FRE 2507
Fontenay-Aux-Roses, France
firstname.surname@robot.jussieu.fr

Commissariat à l'Energie Atomique (CEA)
[⋆]LIST – DTSI
Fontenay-Aux-Roses, France
firstname.surname@cea.fr

***Abstract*— This paper proposes a novel planing algorithm for image based visual servoing (IBVS). IBVS is conceptually simple but prone to fail in case of large displacements. Hence, the objective is to provide pathes for reference points in the image plane, such that the camera is steered from initial position to desired position. The novel scheme guarantees that the camera is always oriented such that the object of interest is in the field of view and that any reference point set corresponds indeed to a feasible camera pose. No camera calibration is necessary. As a novelty, this work is planning a camera motion, hence yielding compact pathes in robot work space.**
The proposed planner is compared to others by experiments.

I. Introduction

Eye-in-hand image-based visual servoing (IBVS) [1] consists in controlling robot motion by a visual sensor mounted on the end-effector. With this approach, a desired image is first recorded when the camera is placed at a desired location with respect to the observed object. Then, from any initial location where the object can be seen by the camera, the controller shall provide convergence towards the desired image. A number of theoretical and experimental studies have evaluated its properties : (I) In the case of large control errors [2], IBVS produces unnecessary robot motion, since control is addressed in image space. This may lead to a task failure. (II) IBVS is not globally stable and can be proven to be stable only in a (yet unknown) neighbourhood of the desired location [3], [4]. Such shortcomings have motivated interpolating the initial image and the final desired image. This reference path can then be fed to the real time controller, which is left to control small errors, thus increasing robustness.

A major difficulty is that the path planned for n image features must correspond to a camera path in SE(3), where usually $n > 6$. However, computing a path in SE(3) would require a 3D reconstruction and hence an accurate calibration and an accurate objet model. Rather, we will focus on path planners that do not rely on any precise information, while guaranteeing that the planned path corresponds to a (although unknown) Euclidean path.

Moreover, during any robot motion, the target object must be *visible* for the camera. Otherwise the servoing will fail.

In the sequel we focus on 6dof manipulation tasks where the end-effector is possibly operating nearby the target.

Related Work

A classical method for integrating constraints in path planning are potential functions. [5] assigns repulsive forces to image borders in image space and joint limits in workspace. Other approaches switches directly the control law [6], [7], [8], [9], mainly between IBVS and position based visual servoing (PBVS). IBVS is known to be helpful in order to keep features visible whereas there is no control of associated camera path, and conversely for PBVS. However, all these controllers yield unpredictable overall pathes, getting eventually trapped at a local minimum of the planning criterium.

A natural planning scheme is given in [10], [11]: rotation is planned to decay exponentially on a geodesic. Translational control is set up such that the centre of either the object either the camera follows a straight line in the image space. The first allows for pulling the camera further back, thus ensuring visibility, whereas the later reorients the camera, thus not ensuring visibility. However, both approaches require an accurate calibrated camera.

The need for an object model can be circumvented by reconstructing a partial pose, called the Homography matrix, computed from the initial and the desired view, subject to an accurate calibration. In this way, [12] interpolates N discrete intermediary views where rotational interpolation is performed by means of the matrix exponential map, and the resulting path is guaranteed to be feasible.

Based on the partial pose reconstruction (hence model-free), more recent work [13], [14], [15] addresses also the visibility issue, essentially by proposing some circular camera path, centred at the object. Yet all these controllers need some further manual adjustment to determine the curvature of the path.

Indeed the model-freeness makes it appealing to use the exponential map for interpolating the rotation. Yet the angle and the axes of rotation as elements of the euclidean space are recovered by an estimate of camera calibration. It is well known that reconstructed position (epipole, rotation axes) is sensitive to input variations, due to their strong intern structure. Our approach is conceptually different: we interpolate the uncalibrated Homography (the collineation matrix); no reconstruction of any sensible pose parameter is necessary and the planning is *invariant* to camera calibration.

1-4244-0602-1/07/$20.00 ©2007 IEEE.

Respective elementary relations are recalled in sec. II. Previously presented *feasible* and *calibration-free* path planners are given in sec. III-A. Then, in sec. III-B, the main contribution of this paper is presented. It computes an additional camera translation such that planned image points are guaranteed to be *visible* where camera displacement is *controlled* by being basically a straight line interpolation. Experimental results are given in sec. IV and some concluding remarks in sec. V.

II. Modelling

A. Constraint for two views with a Pinhole Camera

Consider a camera and its frame $\mathcal{F}_c$, centered at the optical center O_c, where $\vec{e}_z$ coincides with the optical axis. The camera performs a perspective projection, mapping a 3D point $\mathcal{P}_i$, which coordinates in $\mathcal{F}_c$ are $\mathbf{x}_i^c = [X_i^c \; Y_i^c \; Z_i^c]^{\mathrm{T}}$, into homogeneous image coordinates $\mathbf{p}_i = [u_i \; v_i \; 1]^T$,

$$Z_i^c \mathbf{p}_i = \mathbf{K}\mathbf{x}_i^c, \qquad \mathbf{K} \in \mathbb{R}^{3\times 3}, \tag{1}$$

where the regular matrix $\mathbf{K}$ groups the intrinsic camera parameters [16].

Let $\mathcal{F}_0$ and $\mathcal{F}_1$ be coordinate frames attached to the camera in its initial and final locations, respectively. A rigid target object of unknown shape is characterized by m points $\mathcal{P}_i$ on its surface. The m points $\mathcal{P}_i$ are visible in both the initial and final images. Each point $\mathcal{P}_i$ has coordinates $\mathbf{x}_i^0$, $\mathbf{x}_i^1$ in $\mathcal{F}_0$ and $\mathcal{F}_1$ respectively, related by

$$\mathbf{x}_i^0 = \mathbf{R}_{0\to 1}\mathbf{x}_i^1 + \mathbf{t}_{0\to 1}, \tag{2}$$

where $\mathbf{R}_{0\to 1}$ is the rotation matrix from $\mathcal{F}_0$ to $\mathcal{F}_1$ and $\mathbf{t}_{0\to 1}$ is the corresponding translation vector. Image coordinates $\mathbf{p}_i^1$ and $\mathbf{p}_i^0$ can be related by combining (1) and (2):

$$Z_i^0 \mathbf{p}_i^0 = \mathbf{G}_1 \; Z_i^1 \mathbf{p}_i^1 + \mathbf{g}_1 \;, \tag{3}$$

with affine motion parameters $\mathbf{G}_{0\to 1} = \mathbf{K}\mathbf{R}_{0\to 1}\mathbf{K}^{-1}$ and $\mathbf{g}_{0\to 1} = \mathbf{K}\mathbf{t}_{0\to 1}$. Eq. (3) can be seen as generalizing the rigid body constraint to an affine (non-orthogonal) 3D space.

The key idea of this work is to provide trajectories for $(\mathbf{p}_i, Z_i)$ in this 3D space, hence invariant to camera calibration, in such a way that the associated Euclidean motion exists. To do so, one decomposes the parameters $\mathbf{G}_{0\to 1}$ and $\mathbf{g}_{0\to 1}$ into a canonical form, which is easy to reparameterize.

B. Retrieving Point Depth

Assumption 1: (see also [17],[18]) From the image coordinates of m points $\mathcal{P}_i$, matched in the initial, $\mathbf{p}_i^0$, and the final image, $\mathbf{p}_i^1$, the matrix $\mathbf{G}_{0\to 1}$ can be computed, while the depth fields Z_i^1, Z_i^0 and the vector $\mathbf{g}_{0\to 1}$ can be estimated up to an unknown constant scale factor ζ, without requiring any model of the camera, nor any model of the object. ■

Several algorithms in the literature allow for such an identification without requiring any calibration information. A key issue in this matter is to distinguish between the rotation and translation displacements in order to identify the corresponding entities separately. For example, in [17], matched points at the infinity, only affected by the rotational displacement, are used to reconstruct $\mathbf{G}_{0\to 1}$ from $\mathbf{p}_i^0$ and $\mathbf{p}_i^1$. Alternatively, standard matched points extracted from three images can be used, two of them being separated by a pure translation [18].

C. Interpolation of the collineation matrix $\mathbf{G}_{0\to 1}$

Matrix $\mathbf{G}_{0\to 1}$ (cf. eq. 3) depends on matrices $\mathbf{K}$ and $\mathbf{R}_{0\to 1}$. Since camera calibration $\mathbf{K}$ is constant, interpolation of $\mathbf{G}_{0\to 1}$ comes down to an interpolation of $\mathbf{R}_{0\to 1}$. It is well known that an eigen-decomposition of a rotation matrix,

$$\mathbf{R}_{0\to 1} = \mathbf{U}\mathbf{\Lambda}(\theta)\mathbf{U}^*$$

is written in terms of rotation angle θ in the matrix $\mathbf{\Lambda}$ of eigenvalues and rotation axes $\mathbf{u_1}$ in the matrix $\mathbf{U}$, regrouping the eigenvectors. It is always possible to arrange the elements of $\mathbf{U}$ and $\mathbf{\Lambda}(\theta)$ such that

$$\mathbf{\Lambda}(\theta) := \begin{bmatrix} 1 & 0 & 0 \\ 0 & e^{j\theta} & 0 \\ 0 & 0 & e^{-j\theta} \end{bmatrix} \quad \text{and} \quad \mathbf{U} := \begin{bmatrix} \mathbf{u}_1 & \mathbf{u}_2 & \mathbf{u}_2^* \end{bmatrix}.$$

Hence, a continuous interpolation of $\mathbf{R}_{0\to 1}$ is given by

$$\mathbf{R}_{0\to\tau} = \mathbf{U}\,\mathbf{\Lambda}(\tau\theta)\,\mathbf{U}^*, \qquad \tau \in [0,1].$$

Consequently, an interpolation of $\mathbf{G}_{0\to 1}$ is given by

$$\mathbf{G}_{0\to\tau} = \mathbf{K}\mathbf{U}\,\mathbf{\Lambda}(\tau\theta)\,\mathbf{U}^*\mathbf{K}^{-1}, \qquad \tau \in [0,1],$$

where $\mathbf{K}$, $\mathbf{U}$ and θ satisfy $\mathbf{G}_{0\to 1} = \mathbf{K}\mathbf{R}_{0\to 1}\mathbf{K}^{-1}$. In practice, rotation $\mathbf{R}_{0\to 1}$ and calibration $\mathbf{K}$ are not known at a sufficient accuracy in order to interpolate $\mathbf{G}_{0\to 1}$ by the last equation.

On an other hand, since $\mathbf{G}_{0\to 1}$ and $\mathbf{R}_{0\to 1}$ are similar, their eigenvalues are equal. Thus, an interpolation of $\mathbf{G}_{0\to 1}$, resulting in a geodesic, is possible,

$$\mathbf{G}(\tau) := \mathbf{G}_{0\to\tau} = \mathbf{V}\,\mathbf{\Lambda}(\tau\theta))\mathbf{V}^{-1}, \qquad \tau \in [0,1], \tag{4}$$

where the diagonal matrix $\mathbf{\Lambda}(\theta)$ regroups again the eigenvalues of $\mathbf{G}_{0\to 1}$ and $\mathbf{V}$ a set of associated eigenvectors. Note that the eigen-decomposition of $\mathbf{G}_{0\to 1}$, $\mathbf{G}_{0\to 1} = \mathbf{V}\,\mathbf{\Lambda}(\theta)\,\mathbf{V}^{-1}$ does not require any knowledge of $\mathbf{K}$ nor $\mathbf{R}_{0\to 1}$.

III. Image Based Trajectory Planning

Now we can precisely state our objective :

> For image points $\mathbf{w}_i(\tau) := \mathbf{p}_i(\tau)Z_i(\tau)$, compute reference pathes $\mathbf{w}_i(\tau) \in \mathbb{R}^{3m}$, $\tau \in [0,1]$, which are *feasible*, *visible* and *compatible*.

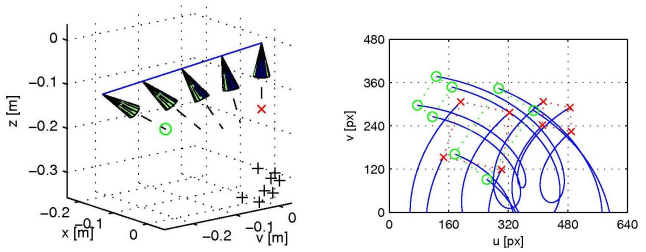
The following definitions are hereby used :

Definition 3.1 (compatible with boundary conditions): This means $\mathbf{w}_i(0) := \mathbf{w}_i^0 = \mathbf{p}_i^0 Z_i^0$ and $\mathbf{w}_i(1) := \mathbf{w}_i^1 = \mathbf{p}_i^1 Z_i^1$.

Definition 3.2 (feasible): A continuous path $\mathbf{w}_i(\tau)$ for m image points is called *feasible* if there exists a continuous camera motion $\mathbf{R}(\tau)$, $\mathbf{t}(\tau)$ and a constant camera $\mathbf{K}$ such that for any τ, the point set $\mathbf{p}_i(\tau)$ can be described as a projection of 3D points $\mathbf{x}_i(\tau)$ of a given rigid object.

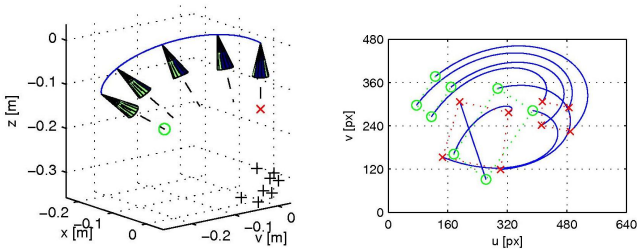
Definition 3.3 (visible): Given a FOV by its opposite corners $(\underline{u},\underline{v})$ and $(\overline{u},\overline{v})$ in pixel coordinates, a path $\mathbf{w}_i(\tau) = Z_i(\tau)[u_i(\tau) \;\; v_i(\tau) \;\; 1]^{\mathrm{T}}$ is said to be visible, if $\forall\tau \in [0,1]$:

$\underline{u} \le u_i(\tau) \le \overline{u}$ and $\underline{v} \le v_i(\tau) \le \overline{v}$, $\qquad i \in \{1,\ldots,m\}$



(a) Camera path. (b) Image Point Pathes.

Fig. 1: Algorithm A: Path is straight line for camera.



(a) Camera path. (b) Image Point Pathes.

Fig. 2: Algorithm B: straight line for selected image point $\mathcal{P}_s$.

Remark 1: Note that $\mathbf{w}_i(\tau) := \mathbf{p}_i(\tau)Z_i(\tau)$ can always be separated into $\mathbf{p}_i(\tau)$ and $Z_i(\tau)$ using $\mathbf{p}_{i,3} = 1$.

Remark 2: In the sequel, we suppose $\zeta = 1$. We revisit this issue when commenting the experiments in sec. IV.

A path can be ensured to be *feasible* choosing $\mathbf{G}(\tau)$, $\mathbf{g}(\tau)$ as parameters. Point pathes $\mathbf{w}_i(\tau)$ are then determined by

$$\mathbf{w}_i(\tau) := \big(\mathbf{G}(\tau)\big)^{-1}\big(\mathbf{w}_i^0 - \mathbf{g}(\tau)\big)$$

Compliance with boundary conditions is given, when $\mathbf{G}(\tau)$ and $\mathbf{g}(\tau)$ satisfy

$$\begin{aligned} \mathbf{G}(0) &= \mathbf{I}_3, & \mathbf{g}(0) &= \mathbf{0}_3, \\ \mathbf{G}(1) &= \mathbf{G}_{0\to1}, & \mathbf{g}(1) &= \mathbf{g}_{0\to1}. \end{aligned}$$

Algorithms will differ from each other only in planning the translation $\mathbf{g}_{0\to1}$.

In the next, two known concepts of calibration free path planning for image points are reviewed, before introducing our new planning scheme.

A. Previously Described Methods

1) Straight Line for Camera Centre (Algorithm A): [12] proposes to interpolate $\mathbf{g}_{0\to1}$ linearly:

$$\mathbf{g}^{\mathbf{A}}(\tau) := \tau\mathbf{g}_{0\to1}$$

Image point pathes are then computed by :

$$\mathbf{p}_i(\tau)Z_i(\tau) := \big(\mathbf{G}(\tau)\big)^{-1}\big(\mathbf{p}_i^0 Z_i^0 - \mathbf{g}^{\mathbf{A}}(\tau)\big) \qquad (5)$$

This yields a very satisfactory behaviour of planned camera motion (fig. 1a), namely a straight line for the optical centre in target space. On the other hand, there is no control in the image space and resulting point pathes tend to run out of the field of view even for modest camera motion (fig. 1b). Despite its simplicity, this approach is not very useful in practice.

2) Straight Line for Image Point (Algorithm B): It is well known [1] for visual servoing that controlling point features directly in the image plane (instead of in robot work space) increases chances to keep the target within the field of view. Following these lines, we have proposed recently to compute $\mathbf{g}(\tau) := \mathbf{g}^{\mathbf{B}}(\tau)$ such that the projection of an arbitrary selected image point $\mathcal{P}_s \in \{\mathcal{P}_1, \ldots, \mathcal{P}_m\}$ performs a straight line path on the image plane [19]:

$$\begin{aligned} \mathbf{p}_s(\tau) &:= \mathbf{p}_s^0 + \tau\big(\mathbf{p}_s^1 - \mathbf{p}_s^0\big), \\ Z_s(\tau) &:= Z_s^0 + \tau\big(Z_s^1 - Z_s^0\big), \end{aligned}$$

and, consequently,

$$\mathbf{g}^{\mathbf{B}}(\tau) := \mathbf{p}_s^0 Z_s^0 - \mathbf{G}(\tau)\mathbf{p}_s(\tau)Z_s(\tau).$$

Point pathes are then computed by :

$$\mathbf{p}_i(\tau)Z_i(\tau) := \big(\mathbf{G}(\tau)\big)^{-1}\big(\mathbf{p}_i^0 Z_i^0 - \mathbf{g}^{\mathbf{B}}(\tau)\big) \qquad (6)$$

The resulting path is depicted on fig. 2. We have used the same boundary configurations as for fig. 1. Clearly all point pathes stay in the field of view (fig. 2b), at the cost of having lost direct control of the camera 3D motion, which becomes a geodesic of unknown radius (fig. 2a), eventually heavily deviating.

B. Novel method (Algorithm C)

In contrast, the new algorithm is explicitly designed such that the camera performs a straight line path "at the best possible". Deviations from this ideal path will only be conceded in order to preserve visibility. The central idea of this paper is thus to start from the solution of Algorithm A, the ideal straight line path for the camera, and to modify the translational planning, $\mathbf{g}^A(\tau)$ by an additional translation $\mathbf{\Delta g}(\tau)$, such that the new planning is feasible, visible and compatible. Reference pathes $\mathbf{w}_i(\tau)$ are then computed by :

$$\mathbf{w}_i(\tau) = \big(\mathbf{G}(\tau)\big)^{-1}\ \big(\ \mathbf{w}_i^0 - \mathbf{g}^{\mathbf{C}}(\tau)\ \big), \quad \tau \in [0,1], \qquad (7)$$

with $\mathbf{g}^{\mathbf{C}}(\tau) := \mathbf{g}^{\mathbf{A}}(\tau) + \mathbf{G}(\tau)\mathbf{\Delta g}(\tau)$. If Algorithm A yields a visible configuration for a certain τ, then $\mathbf{\Delta g}(\tau) = 0$ and solutions of Algorithms C and A are identical. In the remainderof this section, $\mathbf{\Delta g}(\tau)$ is determined as a translation with 3 dof.

The problem will be treated separately for the components of $\mathbf{\Delta g}(\tau)$ along image coordinate axes $\vec{u}$ and $\vec{v}$. To this end, 3D point $\mathbf{w}_i = [Z_i u_i \ \ Z_i v_i \ \ Z_i]^{\mathrm{T}}$ are projected on the plane $\Pi_{u,z}$, defined by $v = 0$, and on the plane $\Pi_{v,z}$, defined by $u = 0$. Homogeneous 2D coordinates of a projected 3D point $\mathbf{w}_i = [w_{1,i} \ \ w_{2,i} \ \ w_{3,i}]^{\mathrm{T}}$ will be called ${}^{\mathrm{u}}\mathbf{w}_i := [w_{1,i}, w_{3,i}, 1]^{\mathrm{T}}$ in the first case and ${}^{\mathrm{v}}\mathbf{w}_i := [w_{2,i}, w_{3,i}, 1]^{\mathrm{T}}$ in the later. Plane $\Pi_{u,z}$ with projected points ${}^{\mathrm{u}}\mathbf{w}_i$ is shown on fig. 3.

Likewise, the projection of the 3D field of view onto plane $\Pi_{u,z}$ is considered. Since plane $\Pi_{u,z}$ is defined by $v = 0$, the *projected* field of view is described by nothing more than the smallest and largest active pixel in direction $\vec{u}$, namely $\underline{u}$ and $\overline{u}$. The projected 2D field of view has the shape of

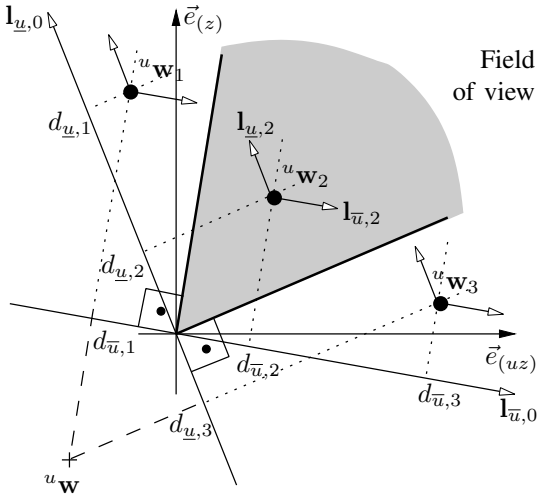


Fig. 3: Planar Problem in plane $\Pi_{u,z}$.

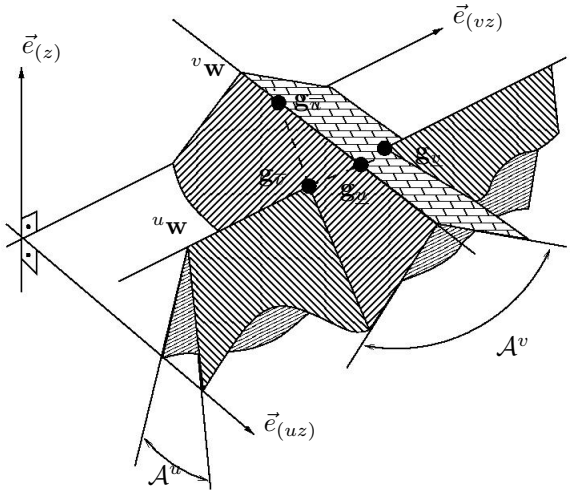


Fig. 4: Merging the two projected solutions.

an open radial segment at the optical centre, which can be described by its radial boundary lines $\mathbf{l}_{\underline{u},0}$ and $\mathbf{l}_{\overline{u},0}$

$$\mathbf{l}_{\underline{u},0} = \frac{1}{\sqrt{1+\underline{u}^2}}\begin{bmatrix}1 & -\underline{u} & 0\end{bmatrix}, \quad \mathbf{l}_{\overline{u},0} = \frac{1}{\sqrt{1+\overline{u}^2}}\begin{bmatrix}-1 & \overline{u} & 0\end{bmatrix},$$

defined as 2D homogeneous lines $\mathbf{l} = [-\sin\phi \quad -\cos\phi \quad d]$, where ϕ is the angle of the line normal with the abscissa and d the line distance from origin. Line normals are defined to point towards the interior of the 2D FOV (gray area in fig. 3).

So far we have determined the 2D FOV stemming from the basic solution (Algorithm A). In fig. 3 for instance, point $\mathcal{P}_2$ is visible with respect to $\underline{u}$ and $\overline{u}$, but not $\mathcal{P}_1$ and $\mathcal{P}_3$. Now we ask: how do we have to *translate* the camera, ie. the centre of the 2D field of view, such that all points are within (translated) radial segment? In fig. 3, by geometric obviousness, the 2D projection of the new camera centre would be the homogeneous 2D point ${}^{\mathrm{u}}\mathbf{w}$.

Let us compute its coordinates. We define for each projected point $\mathcal{P}_i$ a view cone $\mathcal{A}_i^u$, which describes all possible locations for the camera centre such that $\mathcal{P}_i$ is visible. The view cone is of the same shape as the field of view, but of opposite direction and centred at the point $\mathcal{P}_i$. Its interior is given by

$$\mathcal{A}_i^u = \left\{ {}^{\mathrm{u}}\mathbf{a}_i \middle| {}^{\mathrm{u}}\mathbf{a}_i = {}^{\mathrm{u}}\mathbf{w}_i + \lambda \mathbf{l}^*_{\underline{u},0} + \mu \mathbf{l}^*_{\overline{u},0}, \quad \lambda \le 0, \mu \le 0 \right\}$$

where the dual $\mathbf{l}^*$ to the line $\mathbf{l}$ is defined as column vector with $\mathbf{l}(\mathbf{l}^*) = 0$ subject to $||\mathbf{l}^*|| = ||\mathbf{l}||$. Alternatively, the boundaries of each view cone $\mathcal{A}_i^u$ can be computed by

$$\mathbf{l}_{\underline{u},i} = \mathbf{l}_{\underline{u},0} + \begin{bmatrix}0 & 0 & -d_{\underline{u},i}\end{bmatrix}, \quad \mathbf{l}_{\overline{u},i} = \mathbf{l}_{\overline{u},0} + \begin{bmatrix}0 & 0 & -d_{\overline{u},i}\end{bmatrix}, \tag{8}$$

with distance to the origin given by

$$d_{\underline{u},i} = \mathbf{l}_{\underline{u},0}\, {}^{\mathrm{u}}\mathbf{w}_i, \qquad d_{\overline{u},i} = \mathbf{l}_{\overline{u},0}\, {}^{\mathrm{u}}\mathbf{w}_i.$$

Then, the intersection $\mathcal{A}^u$ represents the smallest view cone such that all points are visible,

$$\mathcal{A}^u := \bigcap_{i=0}^{m} \mathcal{A}_i^u = \left\{ {}^{\mathrm{u}}\mathbf{a} \middle| {}^{\mathrm{u}}\mathbf{a} = {}^{\mathrm{u}}\mathbf{w} + \lambda \mathbf{l}^*_{\underline{u},0} + \mu \mathbf{l}^*_{\overline{u},0}, \quad \lambda \le 0, \mu \le 0 \right\} \tag{9a}$$

defined by its boundary lines $\mathbf{l}_{\underline{u}}$, $\mathbf{l}_{\overline{u}}$, and their intersection ${}^{\mathrm{u}}\mathbf{w}$,

$$\mathbf{l}_{\underline{u}} = \mathbf{l}_{\underline{u},j}, \qquad \text{such that } j = \arg\min_i d_{\underline{u},i}, \tag{9b}$$

$$\mathbf{l}_{\overline{u}} = \mathbf{l}_{\overline{u},k}, \qquad \text{such that } k = \arg\min_i d_{\overline{u},i}, \tag{9c}$$

$${}^{\mathrm{u}}\mathbf{w} = \mathbf{l}_{\underline{u}} \otimes \mathbf{l}_{\overline{u}}. \tag{9d}$$

where the operator $\otimes$ defines projective intersection [16], followed by point normalisation:

$$\otimes : \mathbf{x}, \mathbf{y} \to \mathbf{z} = \frac{\mathbf{x} \times \mathbf{y}}{(\mathbf{x} \times \mathbf{y})[0\ 0\ 1]^{\mathrm{T}}}, \qquad \mathbf{x}, \mathbf{y}, \mathbf{z} \in P^2.$$

Next, considering for the 3D points $\mathbf{w}_i$ their 2D projections ${}^{\mathrm{v}}\mathbf{w}_i$ onto the plane $\Pi_{v,z}$, defined by $u = 0$, a similar calculus yields

$$\mathcal{A}^v = \left\{ {}^{\mathrm{v}}\mathbf{a} \middle| {}^{\mathrm{v}}\mathbf{a} = {}^{\mathrm{v}}\mathbf{w} + \lambda \mathbf{l}^*_{\underline{v},0} + \mu \mathbf{l}^*_{\overline{v},0}, \quad \lambda \le 0, \mu \le 0 \right\},$$

and likewise the homogeneous 2D point ${}^{\mathrm{v}}\mathbf{w}$ represents the necessary amount of translation in the plane $\Pi_{v,z}$ such that all point coordinates v_i, are in the visible range, $v_i \in [\underline{v}, \overline{v}]$.

Up to now, we have computed separately the eventually required amount of translation for the planes $\Pi_{u,z}$ and $\Pi_{v,z}$. Fig. 4 shows in 3D-space the plane $\Pi_{u,z}$, given by $\vec{e}_{(uz)}$ and $\vec{e}_{(z)}$. Plane $\Pi_{v,z}$ is given by $\vec{e}_{(vz)}$ and $\vec{e}_{(z)}$. Since view cones $\mathcal{A}^v$ and $\mathcal{A}^u$ were the result of a projection along $\vec{e}_{(vz)}$ and $\vec{e}_{(uz)}$ respectively, they may be characterised by roof-like surfaces in 3D-space. Clearly, both partial solutions are coupled by the common axes, $\vec{e}_z$. The correct combination of both solutions is one of the four marked points in fig. 4,

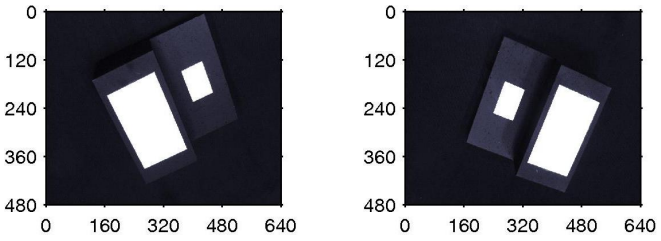


Fig. 5: Left:initial image points $\mathbf{p}_i^0$, right:reference points $\mathbf{p}_i^1$

designing the intersection of ridges with roof surfaces, which are computed by

$$ {}^{\mathrm{v}}\mathbf{g}_{\underline{\mathrm{v}}} = \begin{bmatrix} 0 & 1 & z_u \end{bmatrix} \otimes \mathbf{l}_{\underline{v}}, \quad {}^{\mathrm{v}}\mathbf{g}_{\overline{\mathrm{v}}} = \begin{bmatrix} 0 & 1 & z_u \end{bmatrix} \otimes \mathbf{l}_{\overline{v}}, \quad (10a) $$

$$ {}^{\mathrm{u}}\mathbf{g}_{\underline{\mathrm{u}}} = \begin{bmatrix} 1 & 0 & z_v \end{bmatrix} \otimes \mathbf{l}_{\underline{u}}, \quad {}^{\mathrm{u}}\mathbf{g}_{\overline{\mathrm{u}}} = \begin{bmatrix} 1 & 0 & z_v \end{bmatrix} \otimes \mathbf{l}_{\overline{u}}. \quad (10b) $$

For a given "roof", we select the point which is nearer to the origin, since we are interested in a camera displacement as small as possible:

$$ {}^{\mathrm{v}}\mathbf{g} = \begin{cases} {}^{\mathrm{v}}\mathbf{g}_{\underline{\mathrm{v}}} & \text{if } ||{}^{\mathrm{v}}\mathbf{g}_{\underline{\mathrm{v}}}|| < ||{}^{\mathrm{v}}\mathbf{g}_{\overline{\mathrm{v}}}|| \\ {}^{\mathrm{v}}\mathbf{g}_{\overline{\mathrm{v}}} & \text{otherwise} \end{cases} \quad (11a) $$

$$ {}^{\mathrm{u}}\mathbf{g} = \begin{cases} {}^{\mathrm{v}}\mathbf{g}_{\underline{\mathrm{u}}} & \text{if } ||{}^{\mathrm{u}}\mathbf{g}_{\underline{\mathrm{u}}}|| < ||{}^{\mathrm{u}}\mathbf{g}_{\overline{\mathrm{u}}}|| \\ {}^{\mathrm{v}}\mathbf{g}_{\overline{\mathrm{u}}} & \text{otherwise} \end{cases} \quad (11b) $$

Between these remaining two points, each situated on a different ridge, we arbitrarily choose that one with the smaller component in direction $\vec{e}_z$. Indeed, it does not make sense to compare measurements among different axes until we know anything about the metric (squared pixels for instance). Finally, the solution is completed by taking complementary coordinates from the two partial solutions:

$$ -\mathbf{\Delta g} := \begin{cases} \begin{bmatrix} {}^{\mathrm{u}}\mathbf{w}^{\mathrm{T}} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} & {}^{\mathrm{v}}\mathbf{g}^{\mathrm{T}} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} & {}^{\mathrm{u}}\mathbf{w}^{\mathrm{T}} \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \end{bmatrix}^{\mathrm{T}} & \text{if } z_u < z_v, \\ \begin{bmatrix} {}^{\mathrm{u}}\mathbf{g}^{\mathrm{T}} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} & {}^{\mathrm{v}}\mathbf{w}^{\mathrm{T}} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} & {}^{\mathrm{v}}\mathbf{w}^{\mathrm{T}} \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \end{bmatrix}^{\mathrm{T}} & \text{otherwise.} \end{cases} $$

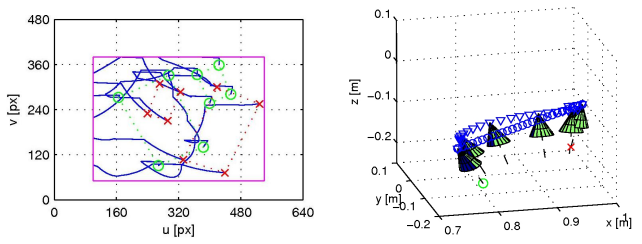
Once $\mathbf{\Delta g}(\tau)$ is computed, the new planning algorithm (7) is completed. It computes image point trajectories $\mathbf{p}_i(\tau)$ which are guaranteed to be *attainable*, *visible* and *compatible* with the image boundary conditions.

Remark 3: The new algorithm C performs computations in a 3D-space and 2D-spaces defined by image coordinate axes. Hence, they are not euclidian but affine spaces. However, our algorithm C uses only properties [16] of the affine space, namely comparing distances along parallel lines.

IV. Experimental Results

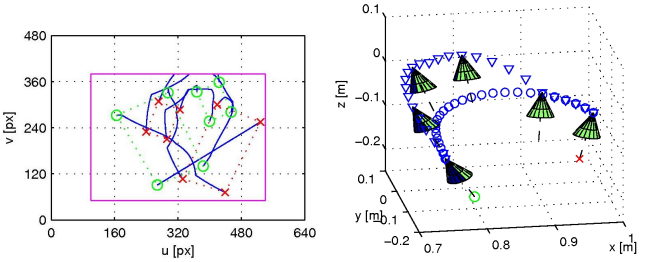
For the experimental setup we used an industrial robot of type *Stäubli RX 90* with a camera *Basler 301f* in eye-in-hand configuration in order to validate the planning algorithm C.

As target object we consider a manufactured piece with two rectangles. The coordinates of the eight corners which are in two different planes are thought to be controlled by a classic image based scheme [3]. The initial and reference images are shown in fig. 5. Respective initial and final camera positions are separated by $26cm$ and $134°$. Due to



(a) Image point pathes $\mathbf{p}_i^C(\tau)$. (b) Camera path $\bigtriangledown$.

Fig. 6: Pathes planned by novel Algorithm C.



(a) Image point pathes (b) Camera path $\bigtriangledown$ (wo. correction: $\circ$)

Fig. 7: For comparison: Pathes planned by [19].

the important error on rotation, this is a difficult task for 2D controller [3] operating in set point mode.

Depth distributions $\hat{Z}_i^0 = \zeta Z_i^0$, $\hat{Z}_i^1 = \zeta Z_i^1$ can then determined by one of the methods mentioned in sec. II-B. In the experiments, a linear model based depth reconstruction was used. Now algorithm C can be applied and point pathes $\mathbf{p}_i(\tau)$ are computed (blue lines in fig. 6a), interpolating initial position (green circles) and reference position (red crosses). Contours of the object rectangles are indicated by dotted lines. Note that in order to make clear the effect of Algorithm C, we have artificially reduced the FOV by 50 pixels on the bottom and by 100 pixels on the three other images sides, marked by the pink rectangle. Thus, planned pathes reach eventually the reduced border, but never leave it. Fig. 6b shows the resulting camera path. Intermediate camera orientation is given for $\tau \in \{0, 0.2, 0.4, 0.6, 0.8, 1.0\}$ for the path $\bigtriangledown$, planned by algorithm C. Deviation from path $\circ$ (algorithm A) represents the translation correction $\mathbf{\Delta g}$, assuring visibility. Despite the heavy deformation of image point pathes, camera centre is at most deviated by $10cm$.

Note that straight line path A is not visible thus not useable. Here it is only shown for comparison. Even if visible, injecting path A into an IBVS will yield a camera straight line only if depth scale ζ *and* external calibration $\mathbf{T_N}$ are perfectly known. In all experiments, ζ is subject to an error of about 20% and $\mathbf{T_N}$ also is only coarsely known.

For comparison, we consider the path planning described in [19], which yields as well *feasible*, *visible* and *compatible* image point pathes. This algorithm imposes for a selected image point a straight line path in image space (fig. 7a).

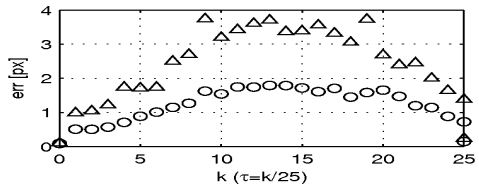


Fig. 8: Algorithm C: Pixel error for 26 intermediate point positions (after convergence). ○: mean error, △: max. error

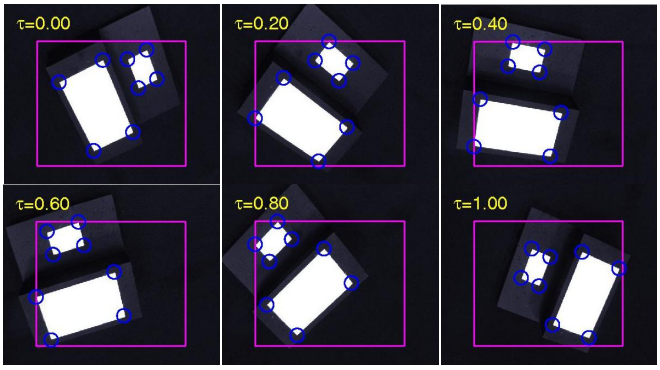


Fig. 9: Object identified during visual servoing along path planned with Algorithm C at instant τ, seen by the camera.

However, this yields generally larger deformation of the camera path (fig. 7b); in this example up to $30cm$ from straight line for camera centre. Note that the algorithm in [19] has only one dof for translating the camera whereas the novel Algorithm C can modify $\mathbf{\Delta g}(\tau)$ with 3 dof.

Finally, let us verify that a path planned by Algorithm C is indeed feasible. To this end, we select 26 intermediate configurations $\mathbf{p}_i^C(\tau_k)$ on the planned path at $\tau_k = k/25$ for $k \in [0, 25]$. Each of these configurations is successively servoed by 2D image point visual control [3]. After the servoing has settled, the error e_k is computed, based on measured points $\mathbf{p}_i(\tau_k)$ and reference points $\mathbf{p}_i^C(\tau_k)$:

$$e_i(k) := ||\mathbf{p}_i^C(\tau_k) - \mathbf{p}_i(\tau_k)|| \qquad k \in [0, 25].$$

Fig. 8 shows for each instant τ the mean value $\frac{1}{m}\sum_{i=1}^{m} e_i(k)$ (circles ○), and the maximum value $\max_i e_i(k)$ (triangles △). These errors being small, all the computed intermediate reference configurations, based on projective interpolation of $\mathbf{G}$, and $\mathbf{g}$, correspond indeed to *feasible* camera configurations. Note that $e_k = 0$ in the case of perfect reconstruction of $\mathbf{G}$, $\mathbf{g}$, i.e. perfect pinhole projection of the real system.

Fig. 9 shows the target where image points are at most on the reduced visibility border, but they do never trespass.

V. Conclusions

The proposed algorithm computes point pathes in image space in order to allow for large displacements to be controlled by image based visual servoing (IBVS). When correcting the motion to ensure visibility, the proposed algorithm takes advantage of all 3 translational dof; moreover it is based on a controlled camera displacement, thus yielding in general compacter pathes than previous planners like [19]. This can also be seen by a "better" exploitation the sensor range *i.e.* image space. Contrarily to other schemes, Homography decomposition is not used to perform sensitive partial euclidean reconstruction; the algorithm relies only on properties of affine space. Note that the scheme is a direct calculus, only using a few elementary operations at run-time, after the single off-line eigen decomposition of $\mathbf{G}$. Contrarily to other planners like potential field methods our scheme is a global planning scheme and thus provides always a solution. Even if the assumption of a perfect reconstruction of affine parameters $\mathbf{G}$, $\mathbf{g}$ is unrealistic, experimental results show that standard machine vision is enough to compute pathes near to feasible ones.

References

[1] S. Hutchinson, G. Hager, and P. Corke. A Tutorial on Visual Servo Control. *IEEE Trans. on Robotics and Automation*, 12(5):651–670, 1996.

[2] F. Chaumette. Potential problems of stability and convergence in image-based and position based visual servoing. In D. Kriegman, G. Hagar, and Morse A., editors, *The Confluence of Vision and Control (LNCS 237)*, pages 66–78. Springer, Berlin Heidelberg, 1998.

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

[4] E. Malis and P. Rives. Robustness of Image-Based Visual Servoing with Respect to Depth Distribution Errors. In *IEEE ICRA*, volume 2, pages 1056–1061, Taipei, Taiwan, 2003.

[5] Y. Mezouar and F. Chaumette. Path planning for robust image-based control. *IEEE Trans. on Robotics and Automation*, 18(4):534–544, 2002.

[6] N. R. Gans and S. A. Hutchinson. An asymptotically stable switched system visual controller for eye in hand robots. In *IEEE/RSJ IROS*, pages 735–742, Las Vegas, Nevada, USA, 2003.

[7] K. Hashimoto and T. Noritsugu. Potential switching control in visual servoing. In *IEEE ICRA*, pages 2765–2770, USA, 2000.

[8] G. Chesi, K. Hashimoto, D. Prattichizzo, and A. Vicino. A switching control law for keeping features in the field of view in eye-in-hand viusal servoing. In *IEEE ICRA*, pages 3929–3934, Taiwan, 2003.

[9] N. Mansard and F. Chaumette. A new redundancy formalism for avoidance in visual servoing. In *IEEE/RSJ IROS*, volume 2, pages 1694–1700, Edmonton, Canada, August 2005.

[10] B. Thuilot, P. Martinet, L. Cordesses, and J. Gallice. Position based viusal servoing: keeping the object in the field of vision. In *IEEE ICRA*, pages 1624–1629, Washington DC, USA, May 2002.

[11] V. Kyrki, D. Kragic, and H. I. Christensen. New shortest-path approaches to visual servoing. In *IEEE/RSJ IROS*, pages 349–354, Sendai, Japan, 2004.

[12] Y. Mezouar, A. Remazeilles, P. Gros, and F. Chaumette. Image interpolation for image-based control under large displacement. In *IEEE ICRA*, volume 3, pages 3787–3794, Washington DC, USA, 2002.

[13] B. Allotta and D. Fioravanti. 3D Motion Planning for Image-Based Visual Servoing Tasks. In *IEEE ICRA*, pages 2185–2190, Spain, 2005.

[14] G. Chesi and A. Vicino. Visual Servoing for Large Camera Displacements. *IEEE Trans. on Robotics*, 20(4):724–735, 2004.

[15] M. Iwatsuki and N. Okyiama. A new formulation of visual servoing based on cylindrical coordinate system. *IEEE Trans. on Robotics*, 21(2):266–273, 2005.

[16] R. Hartley and A. Zisermann. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2000.

[17] Y. Mezouar and F. Chaumette. Optimal Camera Trajectory with Image-Based Control. *Int J of Robotics Research*, 22(10/11):781–804, 2003.

[18] E. Malis. Visual servoing invariant to changes in camera intrinsic parameters. *IEEE Trans. on Robotics and Automation*, 20(1):72–81, 2004.

[19] F. Schramm and G. Morel. Ensuring Visibility in Calibration-Free Path Planning for Image-Based Visual Servoing. *IEEE Trans. on Robotics*, 22(4):848–854, 2006.