4

Treating Image Loss by using the Vision/Motion Link: A Generic Framework

David Folio and Viviane Cadenat
CNRS; LAAS; 7, avenue du Colonel Roche, F-31077 Toulouse,
and Université de Toulouse; UPS: Toulouse.
France

1. Introduction

Visual servoing techniques aim at controlling the robot motion using vision data provided by a camera to reach a desired goal defined in the image (Chaumette & Hutchinson, 2006). Therefore, if the considered features are lost because of an occlusion or any other unexpected event, the desired task cannot be realized anymore. The literature provides many works dealing with this problem. A first common solution is to use methods allowing to preserve the visual features visibility during the whole mission. Most of them are dedicated to manipulator arms, and propose to treat this kind of problem by using redundancy (Marchand & Hager, 1998; Mansard & Chaumette, 2005), path-planning (Mezouar & Chaumette, 2002), specific degrees of freedom (DOF) (Corke & Hutchinson, 2001; Kyrki et al., 2004), zoom (Benhimane & Malis, 2003) or even by making a tradeoff with the nominal vision-based task (Remazeilles et al., 2006). In a mobile robotics context, the realization of a vision-based navigation task in a given environment requires to preserve not only the image data visibility, but also the robot safety. In that case, techniques allowing to avoid simultaneously collisions and visual data losses such as (Folio & Cadenat, 2005a; Folio & Cadenat, 2005b) appear to be limited, because they are restricted to missions where an avoidance motion exists without leading to local minima (Folio, 2007). As many robotic tasks cannot be performed if the visual data loss is not tolerated, a true extension of these works would be to provide methods that accept that occlusions may effectively occur without leading to a task failure. A first step towards this objective is to let some of the features appear and disappear temporarily from the image as done in (Garcia-Aracil et al., 2005). However, this approach is limited to partial losses and does not entirely solve the problem. Therefore, in this work, our main goal is to propose a generic framework allowing to reconstruct the visual data when they suddenly become unavailable during the task execution (camera or image processing failure, landmark loss, and so on). Thus, this work relies on the following central assumption: *the whole image is considered temporarily entirely unavailable*. This problem can be addressed using different methods such as tracking or signal processing techniques. However, we have chosen here to develop another approach for several reasons, which will be detailed in the chapter.
The proposed technique allows to reconstruct the visual features using the history of the camera motion and the last available features. It relies on the vision-motion link that is on the relation between the camera motion and the visual data evolution in the image.

Source: Computer Vision, Book edited by: Xiong Zhihui,
ISBN 978-953-7619-21-3, pp. 538, November 2008, I-Tech, Vienna, Austria

The chapter is organized as follows. We first state the problem and introduce a new general framework allowing to reconstruct the visual features when they become unavailable. Then, we apply it to design a controller able to perform a vision-based navigation task despite the temporary total loss of the landmark during the mission. Finally, we present simulation and experimental results validating the developed approach. We end the chapter by providing a comparative analysis of the different proposed methods.

2. Visual data estimation

In this section, we address the problem of estimating (all or some) visual data s whenever they become unavailable during a vision-based task. Thus, the key-assumption, which underlies our works, is that the whole image is considered to be temporarily completely unavailable. Hence, methods which only allow to treat partial losses of the visual features such as (Garcia-Aracil et al., 2005; Comport et al., 2004) are not suitable here. Following this reasoning, we have focused on techniques dedicated to image data reconstruction. Different approaches, such as signal processing techniques or tracking methods (Favaro & Soatto, 2003; Lepetit & Fua, 2006) may be used to deal with this kind of problem. Here, we have chosen to use a simpler approach for several reasons. First, most of the above techniques rely on measures from the image which are considered to be totally unavailable in our case. Second, we suppose that we have few errors on the model and on the measures[1]. Third, as it is intended to be used in a visual servoing context, the estimated features must be provided sufficiently rapidly wrt. the control law sampling period *Ts*. Another idea is to use a 3D model of the object together with projective geometry in order to deduce the lacking data. However, this choice would lead to depend on the considered landmark type and would require to localize the robot. This was unsuitable for us, as we want to make a minimum assumption on the landmark model. Thus, we have finally chosen to design a new approach to reconstruct the image data when they are entirely lost. It relies on the vision/motion link that relates the variation of the visual features in the image to the camera motion. In the sequel, we define more precisely this notion and then present our estimation method.

2.1 The vision/motion link

In this part, we focus on the vision/motion link. We consider a camera mounted on a given robot so that its motion is holonomic (see remark 1). The camera motion can then be characterized by its kinematic screw $\mathbf{v}_c$ as follows:

$$\mathbf{v}_c = \begin{bmatrix} V_{C/F_c}^{F_c} \\ \Omega_{F_c/F_0} \end{bmatrix} = \mathbf{J}\dot{\mathbf{q}} \tag{1}$$

where $V_{C/F_c}^{F_c} = \left(V_{\vec{\mathbf{x}}_c}, V_{\vec{\mathbf{y}}_c}, V_{\vec{\mathbf{z}}_c}\right)$ and $\Omega_{F_c/F_0}^{F_c} = \left(\Omega_{\vec{\mathbf{x}}_c}, \Omega_{\vec{\mathbf{y}}_c}, \Omega_{\vec{\mathbf{z}}_c}\right)$ represent the translational and rotational velocity of the camera frame wrt. the world frame expressed in *Fc* (see figure 1). $\mathbf{J}$ represents the robot jacobian, which relates $\mathbf{v}_c$ to the control input $\dot{\mathbf{q}}$.

[1] In case where this assumption is not fulfilled, different techniques such as Kalman filtering based methods for instance may be used to take into account explicitly the system noises.

Remark 1: We do not make any hypothesis about the robot on which is embedded the camera. Two cases may occur: either the robot is holonomic and so is the camera motion; or the robot is not, and we suppose that the camera is able to move independently from it (Pissard-Gibollet & Rives, 1995).

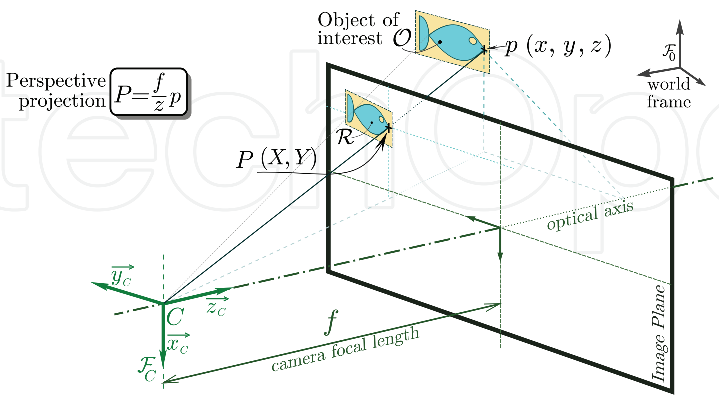


Fig. 1. The pinhole camera model.

Now, let us define the vision/motion link. In this work, we only consider fixed landmarks. We suppose that it can be characterized by a set of visual data s provided by the camera. We denote by **z** a vector describing its depth. As previously mentioned, the vision/motion link relates the variation of the visual signals $\dot{\mathbf{s}}$ to the camera motion. For a fixed landmark, we get the following general definition (Espiau et al., 1992):

$$\dot{\mathbf{s}} = \mathbf{L}_{(\mathbf{s},\mathbf{z})}\mathbf{v}_c = \mathbf{L}_{(\mathbf{s},\mathbf{z})}\mathbf{J}\dot{\mathbf{q}} \tag{2}$$

where $\mathbf{L}_{(\mathbf{s},\mathbf{z})}$ represents the interaction matrix. This matrix depends mainly on the type of considered visual data **s** and on the depth **z** representation. We suppose in the sequel that we will only use image features for which $L_{(s,z)}$ can be determined analytically. Such expressions are available for different kinds of features such as points, straight lines, circles in (Espiau et al., 1992), and for image moments in (Chaumette, 2004).
Our idea is to use the vision/motion link together with a history of the previous measures of image features and of the camera kinematic screw to reconstruct the visual data s. We have then to solve the differential system given by equation (2). However, this system depends not only on the visual features s but also on their depth **z**. Therefore, relation (2) cannot be directly solved and must be rewritten to take into account additional information about **z**. This information can be introduced in different ways, depending on the considered visual primitives. Therefore, in the sequel, we will first state the problem for different kinds of image data before presenting a generic formulation. We will then successively consider the case of points, of other common visual features and of image moments.

2.1.1 The most simple case: the point

The point is a very simple primitive, which can be easily extracted from the image. It is then commonly used in the visual servoing area. This is the reason why we first address this case. Therefore, we consider in this paragraph a visual landmark made of n interest points. Let us recall that, using the pinhole camera model, a 3D point p_i of coordinates (x_i, y_i, z_i) in F_c is projected into a point P_i (X_i, Y_i) in the image plane (see figure 1). We can then define the visual signals vector by a 2*n*-dimensional vector $\mathbf{s} = [X_1, Y_1, \cdots, X_n, Y_n]^T$, where ($X_i$, Y_i) are the coordinates of each projected point. In this case, the interaction matrix $\mathbf{L}_{(\mathbf{s},\mathbf{z})} = [\mathbf{L}_{(P_1,z_1)}, \cdots, \mathbf{L}_{(P_n,z_n)}]^T$ is directly deduced from the optic flow equations. $\mathbf{L}_{(P_i,z_i)}$ is given by (Espiau et al., 1992):

$$\mathbf{L}_{(P_i,z_i)} = \begin{bmatrix} \mathbf{L}_{(X_i,z_i)} \\ \mathbf{L}_{(Y_i,z_i)} \end{bmatrix} = \begin{bmatrix} -\frac{f}{z_i} & 0 & \frac{X_i}{z_i} & \frac{X_i Y_i}{f} & \left(f + \frac{X_i}{f}\right) & Y_i \\ 0 & -\frac{f}{z_i} & \frac{Y_i}{z_i} & \left(f + \frac{Y_i}{f}\right) & -\frac{X_i Y_i}{f} & -X_i \end{bmatrix} \tag{3}$$

where f is the camera focal length. As one can see, $\mathbf{L}_{(P,z)}$ *explicitly* requires a model or an estimation of the depth z_i of each considered point P_i. Several approaches may be used to determine it. The most obvious solution is to measure it using dedicated sensors such as telemeters or stereoscopic systems. However, if the robotic platform is not equipped with such sensors, other approaches must be considered. For instance, it is possible to use structure from motion (SFM) techniques (Jerian & Jain, 1991; Chaumette et al., 1996; Soatto & Perona, 1998; Oliensis, 2002), signal processing methods (Matthies et al., 1989), or even pose relative estimation (Thrun et al., 2001). Unfortunately, these approaches require to use measures from the image, and they cannot be applied anymore when it becomes completely unavailable. This is the reason why we propose another solution consisting in estimating depth **z** together with the visual data **s** (see remark 3). To this aim, we need to express the analytical relation between the variation of the depth and the camera motion. It can be easily shown that, for one 3D point p_i of coordinates (x_i, y_i, z_i) projected into a point P_i (X_i, Y_i) in the image plane as shown in figure 1, the depth variation $\dot{z}_i$ is related to the camera motion according to: $\dot{z}_{\mathrm{i}} = \mathbf{L}_{(\mathrm{z_i})}\,\mathbf{v}_{\mathrm{c}}$, with $\mathbf{L}_{(\mathrm{z_i})} = [0,\ 0,\ -1,\ {}^{z_iY_i}\!/_{f},\ {}^{z_iX_i}\!/_{f}, 0]$. Finally, the dynamic system to be solved for one point can be expressed as follows:

$$\begin{bmatrix} \dot{X}_i \\ \dot{Y}_i \\ \dot{z}_i \end{bmatrix} = \begin{bmatrix} \mathbf{L}_{(X_i,z_i)} \\ \mathbf{L}_{(Y_i,z_i)} \\ \mathbf{L}_{(z_i)} \end{bmatrix} \mathbf{v}_{\mathrm{c}} \tag{4}$$

In the case of a landmark made of *n* points, introducing $\psi = (\mathrm{X}_1, \mathrm{Y}_1, \mathrm{z}_1, \ldots, \mathrm{X}_n, \mathrm{Y}_n, \mathrm{z}_n)^{\mathrm{T}}$, we easily deduce that it suffices to integrate the above system for each considered point.

2.1.2 A more generic case: common geometric visual features

Now, we consider the case of other geometric visual primitives such as lines, ellipses, spheres, and so on. As previously, our goal is to determine the common dynamic system to be solved to compute these primitives when they cannot be provided anymore by the camera. Thus, let O be the observed fixed landmark. We denote by R the projected region of O in the image plane, as described in figure 1. Assuming that R has a continuous surface and closed contour, it can be shown that the depth z_i of each point p_i of O can be related to the coordinates (X_i, Y_i) of each point P_i belonging to R by the following relation (Chaumette, 2004):

$$\frac{1}{z_i} = \sum_{p\geq 0, q\geq 0} A_{pq} X_i^p Y_i^q \qquad \forall (X_i, Y_i) \in \mathrm{R} \tag{5}$$

where parameters A_{pq} depend on the nature of object O. For instance, if we consider a planar object and exclude the degenerate case where the camera optical center belongs to it, the previous equation can be rewritten as follows:

$$\frac{1}{z_i} = AX_i + BY_i + C \qquad \forall (X_i, Y_i) \in \mathrm{R} \tag{6}$$

where $A= A_{10}$, $B= A_{01}$ and $C= A_{00}$ in this particular case.
Now, let us suppose that it is possible to associate to O a set of n visual primitives leading to $\mathbf{s} = [\pi_1, \cdots, \pi_n]^T$, and that the depth $\mathbf{z}$ can be expressed using equation (5). In such a case, relation (2) can be rewritten as follows:

$$\dot{\mathbf{s}} = \begin{bmatrix} \dot{\pi}_1 \\ \vdots \\ \dot{\pi}_n \end{bmatrix} = \begin{bmatrix} \mathbf{L}_{(\pi_1, A_{pq})} \\ \vdots \\ \mathbf{L}_{(\pi_n, A_{pq})} \end{bmatrix} \mathbf{v}_{\mathrm{c}} = \mathbf{L}_{(\pi_1, \cdots, \pi_n, A_{pq})} \mathbf{v}_{\mathrm{c}} = \mathbf{L}_{(\pi, A_{pq})} \mathbf{v}_{\mathrm{c}} \tag{7}$$

where $\mathbf{L}_{(\pi, A_{pq})}$ is the interaction matrix related to the visual primitives π of $\mathbf{s}$ (see remark 2). The interested reader will find in (Espiau et al., 1992) different expressions of the interaction matrix for numerous kinds of common visual features (lines, cylinders, spheres, etc.). It is important to note that, here, $\mathbf{L}_{(\pi, A_{pq})}$ depends *implicitly* on the object depth through the A_{pq} 3D parameters. In this case, the estimation of the visual features by integrating differential system (7) will require to determine A_{pq}. Different methods may be used. A first natural idea is to use the 3D model of the object if it is known. If not, another nice solution is provided by dynamic vision that allows to recover the 3D structure, as in (Chaumette et al. 1996). Unfortunately, this approach requires to define a particular motion to the camera, which is not suitable in a visual servoing context. Another solution would be to use a similar approach to the case of points. The idea is then to first relate $\dot{A}_{pq}$ to the camera kinematic screw $\mathbf{v}_c$, and then to estimate A_{pq} together with $\mathbf{s}$. However, as $\dot{A}_{pq}$ depends on the considered visual features, it would be difficult to design a comprehensive formulation of the estimation problem. Therefore, to provide a generic framework, we propose to use (5) to identify A_{pq}. To this aim, we consider a set of l points P_i (X_i, Y_i) belonging to the region R (see remark 2). The A_{pq} 3D parameters can then be determined on the base of the coordinates (X_i, Y_i) using any identification method such as least-squares techniques. Finally, the geometric visual features will be computed using the four-step algorithm 1.

```
Algorithm 1: Computation of geometric visual features
1. Considering a set of $l$ points belonging to R, define vector $\psi = (X_1, Y_1, z_1, \dots, X_l, Y_l, z_l)^T$
   as in the previous paragraph 2.1.1.
2. Solve system (4) to compute an estimation $\tilde{\psi}$ of $\psi$, that is an estimation of each triple
   (Xi, Yi, zi)
3. Use $\tilde{\psi}$ to identify the coefficients $A_{pq}$ of the surface which fits the best way the $l$
   chosen points using a least-square method for instance.
4. Knowing parameters $A_{pq}$, integrate dynamic system (7) and deduce an estimation of
   vector $\mathbf{s} = [\pi_1, \cdots, \pi_n]^T$
```

Remark 2: In some cases, several objects O could be used to design the set of visual primitives involved in (7). In such a case, a solution is to associate a set of l points to each observed object and to follow algorithm 1 for each of these sets.

2.1.3 The most difficult case: image moments

Although image moments have been widely used in computer vision, they have been considered only recently in visual servoing (Chaumette, 2004; Tahri & Chaumette 2005). Indeed, they offer several interesting properties. First, they provide a generic representation of any simple or complicated object. Moreover, in the specific context of visual servoing, it can be shown that designing control laws with such features significantly improves the decoupling between translation and rotation in the camera motion (Tahri & Chaumette 2005). Therefore, we have also treated this specific case.

In this section, we first briefly recall some definitions before proposing a method allowing to determine these primitives when they cannot be extracted from the image. As previously, we will consider a fixed object O and will denote by R the projected region of O in the image plane. We will also assume that R has a continuous surface and a closed contour. However, as the proposed reasoning can be easily extended to non-planar landmarks, we will consider here only planar objects for the sake of simplicity. Further details about more complex shapes and different image moments are available in (Chaumette, 2004; Tahri & Chaumette 2005).

The $(i+j)$th order image moment m_{ij} of R in the image is classically defined by:

$$m_{ij} = \iint_{R} X^i Y^j \, dx dy \tag{8}$$

It can be shown that $\dot{m}_{ij}$ can be related to the camera kinematic screw by:

$$\dot{m}_{ij} = \mathbf{L}_{(m_{ij},A,B,C)} \mathrm{v}_c \tag{9}$$

where $\mathbf{L}_{(m_{ij},A,B,C)}$ is the corresponding interaction matrix. The 3D parameters A, B, C allow to define the depth in the case of a planar object as described by equation (6). The analytical expression of $\mathbf{L}_{(m_{ij},A,B,C)}$ expresses as follows (Chaumette, 2004):

$$\mathbf{L}_{(m_{ij},A,B,C)} = \begin{cases} m_{V\vec{x}_c} &= -i\left(Am_{ij} + Bm_{i-1,j+1} + Cm_{i-1,j}\right) - Am_{ij} \\ m_{V\vec{y}_c} &= -j\left(Am_{i+1,j-1} + Bm_{i,j} + Cm_{i,j-1}\right) - Bm_{ij} \\ m_{V\vec{z}_c} &= (i+j+3)\left(Am_{i+1,j} + Bm_{i,j+1} + Cm_{i,j}\right) - Cm_{ij} \\ m_{\Omega\vec{x}_c} &= (i+j+3)m_{i,j+1} + jm_{i,j-1} \\ m_{\Omega\vec{y}_c} &= (i+j+3)m_{i+1,j} - im_{i-1,j} \\ m_{\Omega\vec{z}_c} &= im_{i-1,j+1} - jm_{i+1,j-1} \end{cases} \quad (10)$$

As one can see, the time variation of a (*i*+*j*)th order moment depends on moments of higher orders (up to *i*+*j*+1) and on *A*, *B* and *C*. Therefore, as previously, the interaction matrix $\mathbf{L}_{(m_{ij},A,B,C)}$ depends implicitly on the object depth through the *A*, *B* and *C* parameters. Note that the same results hold for centered and discrete moments (Chaumette, 2004; Folio, 2007). Now, it remains to express the differential system to be solved to determine the desired primitives. Defining the visual features vector by a set of image moments, that is: $\mathbf{s} = [m_1, \cdots, m_n]^T$,and using equation (9) leads to:

$$\dot{\mathbf{s}} = \begin{bmatrix} \dot{m}_1 \\ \vdots \\ \dot{m}_n \end{bmatrix} = \begin{bmatrix} \mathbf{L}_{(m_1,A,B,C)} \\ \vdots \\ \mathbf{L}_{(m_n,A,B,C)} \end{bmatrix} \mathbf{v}_c = \mathbf{L}_{(m_1,\cdots,m_n,A,B,C)} \mathbf{v}_c \quad (11)$$

However, as mentioned above, the time derivative $\dot{m}_{ij}$ of a (*i*+*j*)th order moment depends on the moments of higher orders. Therefore, it is impossible to reconstruct the image moments m_{ij} using the dynamic system (11). To avoid this problem, we propose to estimate a set of visual primitives from which it is possible to deduce the m_{ij} image moments in a second step. Here, following a similar approach to the previous case, we propose to use *l* points belonging to the contour of R to approximate the image moments. We present below the final chosen algorithm 2:

```
Algorithm 2: Computation of image moments
1. Considering a set of $l$ points belonging to the contour of R, define vector $\psi = (X_1, Y_1, z_1, \ldots, X_l, Y_l, z_l)^T$ as mentioned above.
2. Solve system (4) to compute an estimation $\tilde{\psi}$ of $\psi$, that is an estimation of each triple $(Xi, Yi, zi)$
3. Use $\tilde{\psi}$ to compute the different necessary moments $m_{ij}$ and deduce an estimation of vector: $\mathbf{s} = (m_1, \cdots, m_n)^T$
```

Notice that the *l* points of the contour of R must be carefully chosen. In the particular case of polygonal shapes, these points may be defined by the vertices of the considered polygon, and the corresponding image moments can then be determined using the methods described in (Singer, 1993) or in (Steger, 1996). For more complex shapes, it is usually possible to approximate the contour by a polygon and obtain an estimate of the image moments by the same process. Finally, the image moments can be always analytically determined for simple geometric primitives, such as circles, ellipses and so on.

2.2 Generic problem statement

In the previous subsections, we have addressed the problem of reconstructing image data from the vision/motion link. Our goal is now to propose a generic formulation of this problem. Three cases have been highlighted[2] :

a. The interaction matrix $\mathbf{L}_{(\mathbf{s},\mathbf{z})}$ requires *explicitly* the depth and we suppose that $\dot{\mathbf{z}}$ can be directly related to the camera kinematic screw $\mathbf{v}_c$. In that case, we are brought back to the case of the point and we only have to solve a system similar to the one given by (4).
b. The interaction matrix $\mathbf{L}_{(\mathbf{s},\mathbf{z})}$ depends *implicitly* on a model of the depth through the A_{pq} coefficients for instance. In such a case, to provide a generic framework, we propose to apply the results obtained for the points to compute the needed parameters. It is then possible to determine the desired visual features by solving (7).
c. It is difficult or impossible to characterize some (or all) the elements of the differential system. In this last case, the estimation problem may be solved by estimating other more suitable visual features from which we can deduce the desired image data. We then retrieve the first two cases.

Hence, in order to reconstruct the visual data $\mathbf{s}$, we propose to solve the following generic dynamic system:

$$\begin{cases} \dot{\psi} = \mathbf{L}_{(\psi)}\mathbf{v}_c = \varphi(\psi, t) \\ \psi(t_0) = \psi_0 \end{cases} \tag{12}$$

where ψ is the vector to be estimated and ψ_0 its initial value. Its expression depends on the previously mentioned cases:
in case (a), where the depth is explicitely required (e.g. points features): $\psi = (\mathbf{s}^T, \mathbf{z}^T)^T$. In the simple case of points, ψ is naturally given by $\psi = [P_1, \ldots, P_n, z_1, \ldots, z_n]^T$.
in case (b), where the depth is implicitely known, a two steps estimation process is performed: first, we set $\psi = [P_1, \ldots, P_l, z_1, \ldots, z_l]^T$ to reconstruct the l feature points P_i coordinates which allow to identify the $\widetilde{A}_{pq}$ parameters; then we fix $\psi = [\pi_1, \ldots, \pi_l, \widetilde{A}_{pq}]^T$ to estimate the desired set of visual primitives π (see algorithm 1).
in case (c), the expression of ψ is deduced either from case (a) or from case (b), depending in the primitives chosen to reconstruct the desired image features.
The previous analysis has then shown that the estimation of visual data can be seen as the resolution of the dynamic system given by expression (12). Recalling that ψ_0 is the initial value of ψ, it is important to note that it can be considered as known. Indeed, as the visual data is considered to be available at least at the beginning of the robotic task: $\mathbf{s}_0$ is directly given by the feature extraction processing, while the initial depth $\mathbf{z}_0$ can be characterized off-line (see remark 3).
Remark 3: While the image remains available (at least once at the begin), $\mathbf{s}$ is directly obtained from the image features extraction processing. In the case of points, their initial depth $\mathbf{z}_0$ can be computed using one of the previously mentioned methods (SFM methods, signal processing techniques, pose relative estimation approaches, etc). It follows that, for

[2] Note that the proposed approach can only be applied if an analytical expression of $\mathbf{L}_{(\mathbf{s},\mathbf{z})}$ is available.

other primitives, we can assume, without loss of generality, that parameters A_{pq} are known. Note also that ψ_0 being known, it is possible to determine ψ by iteratively applying our estimator from the task beginning. Finally, let us remark that, when the image is lost, ψ is provided by our estimator.
Now, let us address the resolution problem. A first idea is to integrate the above system (12) for any $t\in[t_0; t_f]$ where t_0 and t_f are respectively the initial and final instants of the task. However, in this case, the computation is then quite difficult to carry out. Therefore, we propose to discretize the problem and to solve system (12) during the time control interval $[t_k; t_{k+1}]$.

2.3 Resolution

2.3.1 Case of points: Towards an analytical solution

We focus here on specific features: the points. As previously shown, the differential system to be solved is given by equation (4). We will consider two approaches: in the first one, the obtained solution is independent from the robot structure on which is embedded the camera, while in the second one, it is closely related to it.
An analytical solution independent from the mechanical structure of the robot: In this part, our objective is to propose an analytical solution independent from the mechanical structure of the robot. It only depends on the type of the considered visual features, here points. A first idea is to consider that the camera kinematic screw $\mathbf{v}_c$ is constant during the control interval $[t_k; t_{k+1}]$. However, it is obvious that this property is not really fulfilled. This is the reason why we propose to sample this interval into $N\in$N┌ sub-intervals and to consider that $\mathbf{v}_c$ is constant during $[t_n; t_{n+1}]$ where $t_n = t_k+(n-k)T_N$, $T_N = \frac{T_s}{N}$ and T_s is the control law sampling period. In this way, we take into account (at least a bit!) the variation of $\mathbf{v}_c$ during T_s.
First of all, let us analytically solve system (4) on the sub-interval $[t_n; t_{n+1}]$. Considering that $\mathrm{v}_c(t_n)$ is constant during $[t_n; t_{n+1}]$ yields to:

$$\begin{cases} \dot{X} = -\frac{f}{z(t)}V_{\vec{x}_c}(t_n) + \frac{X(t)}{z(t)}V_{\vec{z}_c}(t_n) + \frac{X(t)Y(t)}{z(t)}\Omega_{\vec{x}_c}(t_n) + (f + \frac{X(t)^2}{f})\Omega_{\vec{y}_c}(t_n) + Y(t)\Omega_{\vec{z}_c}(t_n) \\ Y = -\frac{f}{z(t)}V_{\vec{y}_c}(t_n) + \frac{Y(t)}{z(t)}V_{\vec{z}_c}(t_n) + (f + \frac{Y(t)^2}{f})\Omega_{\vec{x}_c}(t_n) - \frac{X(t)Y(t)}{z(t)}\Omega_{\vec{y}_c}(t_n) - X(t)\Omega_{\vec{z}_c}(t_n) \\ \dot{z} = -V_{\vec{z}_c}(t_n) - \frac{z(t)Y(t)}{f}\Omega_{\vec{x}_c}(t_n) + \frac{z(t)X(t)}{f}\Omega_{\vec{y}_c}(t_n) \end{cases} \quad (13)$$

We set in the sequel $V_{\vec{x}_c} = V_{\vec{x}_c}(t_n)$, $V_{\vec{y}_c} = V_{\vec{y}_c}(t_n)$, $V_{\vec{z}_c} = V_{\vec{z}_c}(t_n)$, $\Omega_{\vec{x}_c} = \Omega_{\vec{x}_c}(t_n)$, $\Omega_{\vec{y}_c} = \Omega_{\vec{y}_c}(t_n)$ and $\Omega_{\vec{z}_c} = \Omega_{\vec{z}_c}(t_n)$ for the sake of simplicity. We also introduce the initial condition $X_n=X(t_n)$, $Y_n=Y(t_n)$ and $z_n=z(t_n)$, $\dot{z}_n=\dot{z}(t_n)$, and $\ddot{z}_n=\ddot{z}(t_n)$. It can be shown that (Folio, 2007)[3]:
1. If $\Omega_{\vec{x}_c}\neq 0$, $\Omega_{\vec{y}_c}\neq 0$, $\Omega_{\vec{z}_c}\neq 0$, hence $A_1\neq 0$, $A_2\neq 0$ and $A_3\neq 0$, then:

[3] The interested reader will find in (Folio, 2007) a detailed proof of the proposed results.

$$X(t) = \frac{f\Omega_{\vec{x}_c} A_4 + \Omega_{\vec{y}_c} A_5}{\Omega_{\vec{z}_c} A_3 z(t)}, \; Y(t) = \frac{f\Omega_{\vec{x}_c} A_4 - \Omega_{\vec{y}_c} A_5}{\Omega_{\vec{z}_c} A_3 z(t)} \text{ and}$$

$$z(t) = -\tfrac{c_1}{A_1}\cos\left(A_1(t-t_n)\right) + \tfrac{c_2}{A_1}\sin\left(A_1(t-t_n)\right) + \tfrac{A_2}{A_1}(t-t_n) + \tfrac{c_1}{A_1} + z_n,$$

with $A_4 = \ddot{z}_n - \Omega_{\vec{x}_c} V_{\vec{y}_c} + z(t) A_3$ and $A_5 = f\Omega_{\vec{z}_c}\left(\dot{z}_n + V_{\vec{z}_c}\right)$.

2. If $\Omega_{\vec{x}_c} = 0$, $\Omega_{\vec{y}_c} = 0$, $\Omega_{\vec{z}_c} = 0$, hence $A_1=0$, $A_2=0$ and $A_3=0$, then:

$$X(t) = \frac{X_n z_n - fV_{\vec{x}_c}(t-t_n)}{z(t)}, \; Y(t) = \frac{Y_n z_n - fV_{\vec{y}_c}(t-t_n)}{z(t)} \text{ and } z(t) = z_n - V_{\vec{z}_c}(t-t_n)$$

5. If $\Omega_{\vec{x}_c} \neq 0$, $\Omega_{\vec{y}_c} \neq 0$, $\Omega_{\vec{z}_c} = 0$, hence $A_1 = \sqrt{\Omega_{\vec{x}_c}{}^2 + \Omega_{\vec{y}_c}{}^2}$, $A_2=0$ and $A_3=0$, then:

$$X(t) = \frac{X_n z_n - fV_{\vec{x}_c}(t-t_n) - f\Omega_{\vec{y}_c} A_6}{z(t)}, \; Y(t) = \frac{Y_n z_n - fV_{\vec{y}_c}(t-t_n) - f\Omega_{\vec{x}_c} A_6}{z(t)}$$

with $A_6 = -\tfrac{c_1}{A_1{}^2}\sin\left(A_1(t-t_n)\right) - \tfrac{c_2}{A_1{}^2}\cos\left(A_1(t-t_n)\right) + (\tfrac{c_1}{A_1} + z_n)(t-t_n) + \tfrac{c_2}{A_1{}^2}$.

6. If $\Omega_{\vec{x}_c} = 0$, $\Omega_{\vec{y}_c} = 0$, $\Omega_{\vec{z}_c} \neq 0$, then:

$$X(t) = \frac{c_3 \Omega_{\vec{z}_c} \cos\left(\Omega_{\vec{z}_c}(t-t_n)\right) + c_4 \Omega_{\vec{z}_c} \sin\left(\Omega_{\vec{z}_c}(t-t_n)\right) - fV_{\vec{y}_c}}{\Omega_{\vec{z}_c} z(t)},$$

$$Y(t) = \frac{-c_3 \Omega_{\vec{z}_c} \sin\left(\Omega_{\vec{z}_c}(t-t_n)\right) + c_4 \Omega_{\vec{z}_c} \cos\left(\Omega_{\vec{z}_c}(t-t_n)\right) + fV_{\vec{x}_c}}{\Omega_{\vec{z}_c} z(t)}$$

where $A_1 = \sqrt{\Omega_{\vec{x}_c}{}^2 + \Omega_{\vec{y}_c}{}^2 + \Omega_{\vec{z}_c}{}^2}$, $A_2 = -\Omega_{\vec{z}_c}\left(V_{\vec{x}_c}\Omega_{\vec{x}_c} + V_{\vec{y}_c}\Omega_{\vec{y}_c} + V_{\vec{z}_c}\Omega_{\vec{z}_c}\right)$, $A_3 = \left(\Omega_{\vec{x}_c}{}^2 + \Omega_{\vec{y}_c}{}^2\right)$,

$c_1 = \ddot{z}$, $c_2 = \dot{z} - \frac{A_2}{A_1{}^2}$, $c_3 = -X_n z_n - f\frac{V_{\vec{y}c}}{\Omega_{\vec{z}c}}$ and $c_4 = Y_n z_n - f\frac{V_{\vec{x}c}}{\Omega_{\vec{z}c}}$.

The above expressions of ($X(t)$, $Y(t)$, $z(t)$) depend on the values of the camera kinematic screw at each instant t_n. These values must then be computed. To this aim, as $\mathbf{v}_c = \mathbf{J}(\mathbf{q})\dot{\mathbf{q}}$ (see equation (1)), it is necessary to evaluate the control input $\dot{\mathbf{q}}$ and $\mathbf{J}(\mathbf{q})$at t_n. Now, recalling that $\dot{\mathbf{q}}$ is hold during the time control interval, that is $\dot{\mathbf{q}}(t) = \dot{\mathbf{q}}(t_k)$, $\forall t \in [t_k; t_{k+1}]$, it is straightforward to show that $\dot{\mathbf{q}}(t_n) = \dot{\mathbf{q}}(t_k)$, $\forall t_n \in [t_k; t_{k+1}]$. It remains to compute $\mathbf{J}(\mathbf{q})$ and therefore the configuration $\mathbf{q}$ at instant t_n. To this aim, we propose to simply integrate the equation $\dot{\mathbf{q}}(t) = \dot{\mathbf{q}}(t_k)$ between t and t_k. It yields to:

$$\mathbf{q}(t) = (t - t_k)\dot{\mathbf{q}} + \mathbf{q}(t_k), \; \forall t \in [t_k; t_k{+}1] \tag{14}$$

where $\mathbf{q}(t_k)$ represents the robot configuration at instant t_k. The corresponding values can be measured using the embedded proprioceptive sensors. Then, on the base of $\mathbf{q}(t)$, it is possible to deduce the jacobian $\mathbf{J}(\mathbf{q}(t))$and therefore the camera kinematic screw on the sub-interval $[t_n; t_{n+1}]$. The latter being constant on this interval, the analytical expressions detailed above can be used to compute $\psi(t)=(X(t), Y(t), z(t))$ at t_{n+1}. The same computations must be performed at each instant t_n to obtain the value of ψ at t_{k+1}, which leads to algorithm 3.

Algorithm 3: Estimation of the points when vc is constant during $[t_n; t_{n+1}] \subset [t_k; t_{k+1}]$

```
Initialization: Determine $\psi(t_0)$ and set $t_k = t_0$ , $\psi(t_k) = \psi(t_0)$
For each control interval $[t_k; t_{k+1}]$ do
        Set $t_n = t_k$ , $\psi(t_n) = \psi(t_k)$
        For each integration interval $[t_n; t_n+1]$ do
                Compute $\mathbf{q}(t_n)$ thanks to relation (14)
                Deduce $\mathbf{J}(\mathbf{q}(tn))$ and $\mathbf{v}_c(t_n) = \mathbf{J}(\mathbf{q}(tn))\ \dot{\mathbf{q}}(t_n)$
                Evaluate $\psi(t_{n+1})$
        End for
End for
```

The proposed approach can be seen as a first step towards the introduction of the camera kinematic screw evolution into the reconstruction procedure. Its main advantage is that its realization is very simple and that it allows to remain independent from the robot mechanical structure. Naturally, the value of N must be carefully chosen: a large value reduces the estimation accuracy, especially if the control law sampling period Ts is large; a small value increases the computation time, but improves the precision, as it is getting closer to the true variable kinematic screw case.

Remark 4: Another equivalent idea to reconstruct the visual features in case of points would be to consider the exponential map approach and the direct measure of the camera velocity. This approach allows to determine the 3D coordinates of the point p_i (x_i, y_i, z_i) using the following relation (Soatto & Perona, 1998):

$$\dot{p}_i = -V_{C/F_c}^{F_c} - \Omega_{F_c/F_0}^{F_c} \wedge p_i(t) \leftrightarrow p_i(t_{k+1}) = \mathbf{R}(t_k)\, p_i(t_k) + \mathbf{t}(t_k)$$

where $\mathbf{R} \in SO_{(3)}$ [4] and $\mathbf{t} \in R^3$ define respectively the rotation and translation of the moving camera. Indeed, $\mathbf{R}$ and $\mathbf{t}$ are related to the camera rotation $\Omega_{F_c/F_0}^{F_c}$ and translation $V_{C/F_c}^{F_c}$ motion thanks to an exponential map (Murray et al., 1994), that is:

$$\begin{pmatrix} \mathbf{R} & \mathbf{t} \\ 0 & 1 \end{pmatrix} = \exp\begin{pmatrix} [\Omega]_\times & V_{C/F_0} \\ 0 & 0 \end{pmatrix}$$

where $[\Omega]_\times$ belongs to the set of 3×3 skew-matrices and is commonly used to describe the cross product of $\Omega_{F_c/F_0}^{F_c}$ with a vector in R^3. However, this approach can be used only if the camera kinematic screw $\mathbf{v}_c$ can be assumed constant during the sampling period Ts, which is not usually the case. We are then brought back to a similar approach to the one presented above.

An analytical solution integrating the mechanical structure of the robot: As previously mentioned, the above analytical solution is restricted to the case where the camera kinematic screw remains constant during $[t_n; t_{n+1}] \subset [t_k; t_{k+1}]$. However, although it offers a general result independent from the robot mechanical structure, this assumption is rarely completely fulfilled in a real context. Indeed, only the control input $\dot{\mathbf{q}}$ can be considered to be really hold between t_k and t_{k+1}, whereas the camera motion $\mathbf{v}_c$ evolves during the same

[4] $SO_{(3)}$: special orthogonal group of Transformations of R^3.

period. This is the reason why our objective in this part is to relax the assumption about **v**c. To this aim, it is necessary to consider a particular robotic system in order to express the corresponding jacobian **J**. We have chosen here the robot Super-Scout II on which we have always realized our experimental tests up to now. This is a nonholonomic cart-like vehicle equipped with a camera mounted on a pan-platform (see figure 2). Its mechanical structure is then simple and the expression of its jacobian is given by (23). In this case, the control input is defined by $\dot{\mathbf{q}} = (v, \omega, \varpi)^T$, where v and ω are the cart linear and angular velocities, while ϖ is the pan-platform angular velocity. Our robot will be more precisely described in the next section dedicated to the application context.

As previously shown, our goal is now to analytically solve the dynamic system given by (4). Considering the particular mechanical structure of our robot, it is impossible to Transactionlate the camera along $\vec{x}_c$ and rotate it around axes $\vec{y}_c$ and $\vec{z}_c$, which leads to $V_{\vec{x}_c} = \Omega_{\vec{y}_c} = \Omega_{\vec{z}_c} = 0$. Taking into account this result into equation (4), the dynamic system to be solved for one point $P(X, Y)$ expresses as follows:

$$\begin{cases} \dot{X} = \frac{X(t)}{z(t)} V_{\vec{z}_c}(t) + \frac{X(t)Y(t)}{f} \Omega_{\vec{x}_c}(t) \\ \dot{Y} = -\frac{f}{z(t)} V_{\vec{y}_c}(t) + \frac{Y(t)}{z(t)} V_{\vec{z}_c}(t) + (f + \frac{Y(t)^2}{f}) \Omega_{\vec{x}_c}(t) \\ \dot{z} = -V_{\vec{z}_c}(t) - \frac{z(t)Y(t)}{f} \Omega_{\vec{x}_c}(t) \end{cases} \tag{15}$$

where $V_{\vec{y}_c}(t)$, $V_{\vec{z}_c}(t)$ and $\Omega_{\vec{x}_c}(t)$ are given by relation (23).

Now, let us determine the analytical solution of the differential (15). We denote by v_k, ω_k and ϖ_k the values of the velocities applied to the robot at instant t_k. For the sake of simplicity, we also introduce the following notations: $X_k = X(t_k)$, $Y_k = Y(t_k)$, $z_k = z(t_k)$ and finally $\vartheta_k = \vartheta(t_k)$ where ϑ represents the pan-platform orientation with respect to the mobile base. After some computations (a detailed proof is available in (Folio, 2007)), we get the following results for $z(t)$ depending on the control inputs ω_k and ϖ_k:

1. If $\omega_k \neq -\varpi_k$ and $\omega_k \neq 0$, then:
   $z(t) = c_1 \sin(A_1(t - t_k)) + c_2 \cos(A_1(t - t_k)) - D_x \cos(\vartheta(t)) + \frac{v_k}{\omega_k} \sin(\vartheta(t)) - C_x$
2. If $\omega_k = -\varpi_k \neq 0$, then:
   $z(t) = \frac{\omega_k}{\varpi_k} D_x (\cos(\vartheta(t)) - \cos(\vartheta_k)) - \frac{v_k}{\varpi_k} (\sin(\vartheta(t)) - \sin(\vartheta_k)) + z_k$ (16)
3. If $\omega_k = 0$, then:
   $z(t) = c_3 \sin(\varpi_k(t - t_k)) + c_4 \cos(\varpi_k(t - t_k)) - v_k(t - t_k)\cos(\vartheta(t)) + \frac{v_k}{2\varpi_k} \sin(\vartheta(t)) - C_x$
4. If $\omega_k = \varpi_k = 0$, then:
   $z(t) = z_k - v_k(t - t_k)\cos(\vartheta_k)$

with[5] $c_1 = \frac{v_k}{\omega_k}\cos(\vartheta_k) + D_x \sin(\vartheta_k) - \frac{Y_k z_k}{f} - C_y$, $c_2 = -\frac{v_k}{\omega_k}\sin(\vartheta_k) + D_x \cos(\vartheta_k) + z_k + C_x$, $c_3 = -\frac{v_k}{2\varpi_k}\cos(\vartheta_k) + \frac{Y_k z_k}{f} - C_y$, $c_4 = -\frac{v_k}{2\varpi_k}\sin(\vartheta_k) + z_k + C_x$ and $A_1 = (\omega_k + \varpi_k)$.

[5] (C_x, C_y) and D_x are geometric parameters of the robot and of the camera (see figure 2).

Now, we consider the determination of $X(t)$. From (15) we can write: $\frac{\dot{X}}{X(t)} = -\frac{\dot{z}}{z(t)}$. $X(t)$ can then be computed by integrating this last relation for $\forall t \in [t_k; t_k+1]$, and we get:

$$X(t) = \frac{z_k X_k}{z(t)} \tag{17}$$

Finally, $Y(t)$ is easily deduced from (15) as follows: $Y(t) = -f\frac{\dot{z}+V_{\vec{z}c}(t)}{z(t)\Omega_{\vec{x}c}(t)}$. Using the solution $z(t)$ given by (16), $Y(t)$ expresses as:

1. If $\omega_k \neq -\varpi_k$ and $\omega_k \neq 0$, then
   $Y(t) = -\frac{f}{z(t)}\left(c_1\cos\left(A_1(t-t_k)\right) - c_2\sin\left(A_1(t-t_k)\right) - D_x\omega\sin\left(\vartheta(t)\right) - C_y(\omega+\varpi)\right)$
2. If $\omega_k = -\varpi_k \neq 0$, then
   $Y(t) = \frac{f}{z(t)\varpi_k}\left(v\cos\left(\vartheta(t)\right) - \cos\left(\vartheta_k\right) + \varpi_k z_k Y_k\right)$
3. If $\omega_k = 0$, then
   $Y(t) = -\frac{f}{z(t)}\left(c_1\cos\left(A_1(t-t_k)\right) - c_2\sin\left(A_1(t-t_k)\right) + v(t-t_k)\sin\left(\vartheta(t)\right) - \frac{v}{2\varpi}\cos\left(\vartheta(t)\right) + C_y\right)$
4. If $\omega_k = \varpi_k = 0$, then
   $Y(t) = -\frac{f}{z(t)}\left(v_k\left(t-t_k\right)\sin(\vartheta_k) + z_k Y_k\right)$

(18)

As one can see, the above solution requires the determination of $\vartheta(t)$. This angle can simply be computed by integrating $\vartheta = \varpi$ between t_k and t. Some straightforward calculus leads to $\vartheta(t) = \varpi_k(t-t_k) + \vartheta_k$, where ϑ_k is the pan-platform angular value at t_k, which is usually provided by the embedded encoder.
The proposed approach takes fully into account the evolution of the camera kinematic screw in the reconstruction process of the triple ($X(t)$, $Y(t)$, $z(t)$) for all $t \in [t_k; t_{k+1}]$. Although the proposed approach is restricted to the considered robotic system and to points, its main advantage lies in its accuracy. As shown in the next section devoted to applications, the obtained results are significantly better with this method.

2.3.2 Numerical resolution: a generic framework

As previously mentioned, the analytical solutions are restricted to the case of points and, for one of them, to the robotic system. The question is now: "What to do if other kinds of image features are used"? In this part, we address this problem and we aim at designing a generic framework allowing to solve the dynamic system (12) in a general context. In such case, it appears to be difficult to keep on proposing an analytical solution to the considered system. This is the reason why we propose to use numerical methods to solve (12). In order to increase the accuracy of the different considered schemes, we propose to divide the $[t_k; t_{k+1}]$ control law interval into $N \in N^*$ sub-intervals $[t_n; t_{n+1}] \subset [t_k; t_{k+1}]$. In this way, it is possible to define the integration step $T_N = \frac{T_s}{N} = t_{n+1} - t_n$.
Using numerical techniques to solve (12) requires to characterize ψ and the different related interaction matrix (see paragraph 2.2.). We suppose in the sequel that such a characterization is possible. Moreover, the numerical algorithms will not be used in the same way depending on the camera kinematic screw is considered to be constant or not on the interval $[t_k; t_{k+1}]$. We can then distinguish the two following cases:

1. The camera motion **v**c can be considered as constant during the control law sampling period T_s. In this case, only one evaluation of **v**c(t_k) at time t_k is needed to compute $\varphi(\psi, t)$, $\forall t \in [t_k; t_{k+1}]$.
2. The camera motion **v**c varies with time in $[t_k; t_{k+1}]$. It is then necessary to calculate it at each instant t_n. Therefore, the computation of $\varphi(\psi, t)$ require an evaluation of **v**c(t_n) for each time t_n. Recalling that only $\dot{\mathbf{q}}$ is hold during T_s, and thanks to (14) we can then compute **v**c(t_n) = **J**(**q**(t_n)) $\dot{\mathbf{q}}_k$ (or even using $\dot{\mathbf{q}}_n$ if available).

The literature provides many methods allowing to numerically solve differential equations. A large overview of such methods is proposed for example in (Butcher, 2008). In this work, we have compared several common numerical schemes to select the most efficient technique. Here, we consider the Euler, Runge-Kutta (RK), Adams-Bashforth-Moulton (ABM) and Backward Differentiation Formulas (BDF) numerical techniques. Hence, we first recall briefly the description of these schemes.

Euler Scheme: It is the simplest numerical integration method, but usually the less accurate. As a consequence, it requires a small integration step. The Euler integration method is classically given by:

$$\tilde{\psi}_{k+1} = \psi_k + T_n \varphi(\psi_k, t_k) \tag{19}$$

Runge–Kutta Schemes: The Runge–Kutta methods are an important family of iterative algorithms dedicated to the approximation of solutions of differential equations. The most commonly used scheme is the fourth order one, which is often referred as RK4. It expresses as follows:

$$\tilde{\psi}_{k+1} = \psi_k + \tfrac{1}{6}\left(K_1 + 2K_2 + 2K_3 + K_4\right), \quad \text{with} : \begin{cases} K_1 = T_n(\psi_k, t_k) \\ K_2 = T_n(\psi_k + \tfrac{1}{2}K_1, t_k + \tfrac{1}{2}T_n) \\ K_3 = T_n(\psi_k + \tfrac{1}{2}K_2, t_k + \tfrac{1}{2}T_n) \\ K_4 = T_n(\psi_k + K_3, t_k + T_n) \end{cases} \tag{20}$$

The methods presented above are known as one-step schemes because they are based only on one previous value to estimate $\tilde{\psi}_{k+1}$. Contrary to this case, techniques using more than one of the previous values (ie. ψ_k, ψ_{k-1}, $\ldots\psi_{k-N}$) are referred as multi-steps approaches. Two of them are described below.

Adams Schemes: The Adams based techniques are multistep methods which approximate the solution of a given differential equations by a polynomial. It usually works as a predictor/corrector pair called the Adams–Bashforth–Moulton (ABM) schemes. The method consists in a two-step approach: the value $\hat{\psi}_{k+1}$ is first predicted with an Adams–Bashforth scheme, and then corrected thanks to an Adams–Moulton algorithm to obtain the estimation $\tilde{\psi}_{k+1}$. For instance the fourth order Adams–Bashforth–Moulton (ABM4) is given by:

$$\begin{aligned} \hat{\psi}_{k+1} &= \psi_k + \tfrac{T_n}{24}\left(55\varphi_{\mathrm{k}} - 59\varphi_{\mathrm{k-1}} + 37\varphi_{\mathrm{k-2}} - 9\varphi_{\mathrm{k-3}}\right) && \text{Adams–Bashforth predictor} \\ \tilde{\psi}_{k+1} &= \psi_k + \tfrac{T_n}{24}\left(9\varphi(\hat{\psi}_{k+1}, t_{k+1}) + 19\psi_k - 5\psi_{k-1} + \psi_{k-2}\right) && \text{Adams–Moulton corrector} \end{aligned} \tag{21}$$

Gear's Methods: The Gear's methods, also known as Backward Differentiation Formulas (BDF), consist in using a polynomial which interpolates the N previous values ψ_k, ψ_{k-1},…,

$\psi_{k\text{-}N}$ to estimate $\tilde{\psi}_{k+1}$. They are mainly dedicated to stiff differential equations (see remark 5). We recall below the fourth order BDF4 scheme which have been used hereafter:

$$\tilde{\psi}_{k+1} = \tfrac{48}{25}\psi_k - \tfrac{36}{25}\psi_{k-1} + \tfrac{16}{26}\psi_{k-2} - \tfrac{3}{25}\psi_{k-3} + \tfrac{12}{25}T_n\varphi(\hat{\psi}_{k+1}, t_{k+1}) \tag{22}$$

Remark 5: A problem is stiff if the numerical solution step size is more severely limited by the stability of the numerical technique than by its accuracy. Frequently, these problems occur in systems of differential equations involving several components which are varying at widely different rates. The interested reader will find more details in the survey by (Shampine & Gear 1979).

Therefore, these numerical methods provide an estimated value $\tilde{\psi}_{k+1}$ after N integration steps over one control law sampling period T_s. The algorithm 4 given below details the different steps of calculus.

Algorithm 4: Estimation of the visual features using numerical schemes.

```
Initialization: Determine $\psi(t_0)$ and set $t_k = t_0$ , $\psi(t_k) = \psi(t_0)$
For each control interval $[t_k; t_{k+1}]$ do
        If multiple step scheme and initialization not over then
                Initialization of the necessary previous values of $\psi$.
        End if
        If $\mathbf{v}_c$ is considered to be constant during $[t_k; t_{k+1}]$ then
                Evaluate $\mathbf{v}c$ only at instant $t_k$ .
        End if
        Set $t_n = t_k$ , $\psi(t_n) = \psi(t_k)$
        For each integration interval $[t_n; t_{n+1}]$ do
                If $\mathbf{v}_c$ varies during $[t_k; t_{k+1}]$ then
                        Evaluate $\mathbf{v}c$ at instant $t_n$ .
                End if
                Evaluate $\varphi(\psi(t_n), t_n)$
                Choose a numerical scheme and compute the corresponding value of $\psi(t_{n+1})$
        End for
End for
```

Finally, numerical schemes can be used since an expression of $\varphi(\psi, t)$ and a history of successive values of ψ is available. In this way, dynamic system (12) can be solved in a general context, that is for any kind of image features and any robot mechanical structures.

2.4 Conclusion

In this section, we have proposed a set of methods allowing to reconstruct the visual features when they cannot be provided anymore by the camera. Most of the works which address this problem classically rely on information based on the image dynamics. In this work, we have deliberately chosen to consider the case where the image becomes totally unavailable. We have then used the vision/motion to link to estimate the lacking data. Our first contribution lies in the modelling step. Indeed, we have stated the estimation problem for different kinds of visual features: points, common geometric primitives and image moments. On the base of this analysis, we have shown that the considered problem can be expressed as a dynamic system to be solved. Our second contribution consists in the

development of different techniques allowing to compute analytical and numerical solutions. The different proposed methods can be easily implemented on a real robot. Note that a careful choice of the different involved sampling periods is mandatory. Finally, these methods also require to have the necessary information for the algorithm initialization. This is particularly true for multi-step numerical methods which need a history of the values of ψ.
Now, our goal is to validate and compare the different proposed approaches.

3. Applications

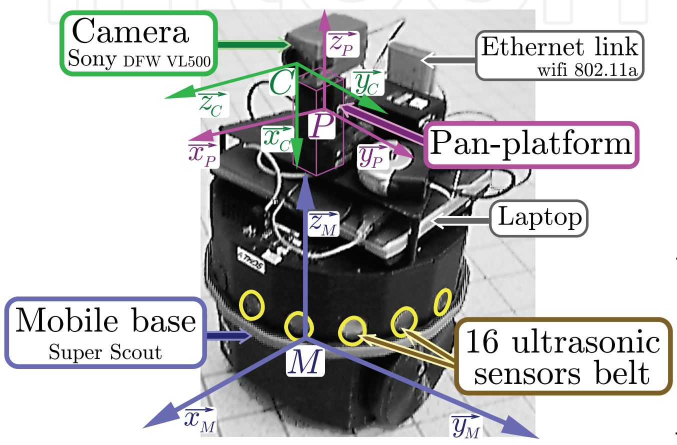
The proposed estimation methods can be used in many applications. For example, it has been recently successfully used to perform vision-based navigation tasks in cluttered environments. Indeed, in such a case, the visual features loss is mainly due to occlusions which occur when the obstacles enter the camera field of view. Integrating our reconstruction techniques in the control law allows to treat efficiently this problem and to realize the task despite the occlusion (Folio & Cadenat, 2007; Folio & Cadenat, 2008). Another interesting application area is to use the estimated visual features provided by our algorithms to refresh more frequently the control law. Indeed, as T_s is often smaller than the vision sensor sampling period T_c[6], the control law is computed with the same visual measures during several steps, which decreases its efficiency. Our work has then be successfully used to predict the visual features between two image acquisitions so as to improve the closed loop performances (Folio, 2007). Finally, it is also possible to apply our results to other related fields such as active vision, 3D reconstruction methods or even fault diagnosis for instance.
In this part, we still consider a visual servoing application but focus on a particular problem which may occur during the mission execution: the camera or the image processing failure. Our idea is here to use our estimation technique to recover from this problem so as the task can still be executed despite it. We first present the robotic system on which we have implemented our works. Then, we detail the mission to be realized and show how to introduce our estimation techniques in the classical visual servoing controller. Finally, we describe both simulation and experimental results which demonstrate the validity and the efficiency of our approach when a camera failure occurs.

3.1 Robotic system description and modelling

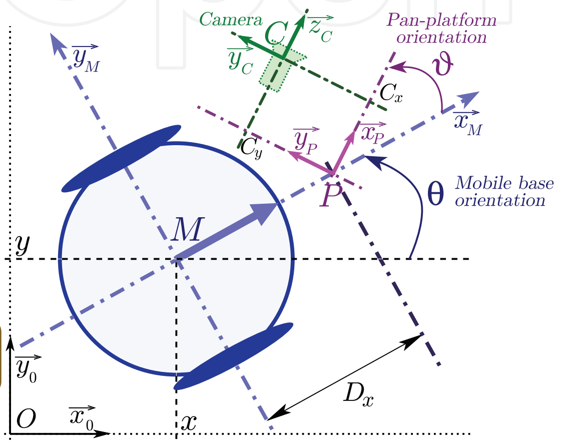
We consider the mobile robot Super-Scout II [7] equipped with a camera mounted on a pan-platform (see figure 2.a). It is a small cylindric cart-like vehicle, dedicated to indoor navigation. A DFW-VL500 Sony color digital IEEE1394 camera captures pictures in yuv 4:2:2 format with 640×480 resolution. An image processing module allows to extract the necessary visual features from the image. The robot is controlled by an on-board laptop running under Linux on which is installed a specific control architecture called G^{en}oM (Generator of Module) (Fleury and Herrb, 2001).
Now, let us model our system to express the camera kinematic screw. To this aim, considering figure 2.b, we define the successive frames: $F_M(M, \vec{x}_M, \vec{y}_M, \vec{z}_M)$ linked to the robot, $F_P(P, \vec{x}_P, \vec{y}_P, \vec{z}_P)$ attached to the pan-platform, and $F_C(C, \vec{x}_c, \vec{y}_c, \vec{z}_c)$ linked to the

[6] On our experimental platform, T_s=50ms while T_c is between 100 and 150 ms.
[7] The mobile robot Super-Scout II is provided by the AIP-PRIMECA of Toulouse.

camera. Let ϑ be the direction of the pan-platform wrt. $\vec{x}_M$, P the pan-platform center of rotation and D_x the distance between the robot reference point M and P. The control input is defined by: $\dot{\mathbf{q}} = (v, \omega, \varpi)^T$, where v and ω are the cart linear and angular velocities, and ϖ is the pan-platform angular velocity wrt. F_M. For this specific mechanical system, the kinematic screw $\mathbf{v}_c$ is related to the control input by the robot jacobian $\mathbf{J}$: $\mathbf{v_c} = \mathbf{J}\dot{\mathbf{q}}$. As the camera is constrained to move horizontally, it is sufficient to consider a reduced kinematic screw: $\mathbf{v_c^r} = \left(V_{\vec{y}_c}, V_{\vec{z}_c}, \Omega_{\vec{x}_c}\right)^T$ involving only the controllable DOF. The corresponding reduced jacobian matrix $\mathbf{J_c^r}$ expresses as follows:



2.a – Nomadic Super-Scout II.



2.b – Cart-like robot with a camera mounted on a pan-platform.

Fig. 2. The robotic system.

$$\mathbf{v_c^r} = \begin{pmatrix} V_{\vec{y}_c}(t) \\ V_{\vec{z}_c}(t) \\ \Omega_{\vec{x}_c}(t) \end{pmatrix} = \begin{pmatrix} -\sin(\vartheta(t)) & D_x\cos(\vartheta(t)) + C_x & C_x \\ \cos(\vartheta(t)) & D_x\sin(\vartheta(t)) - C_y & -C_y \\ 0 & -1 & -1 \end{pmatrix} \begin{pmatrix} v \\ \omega \\ \varpi \end{pmatrix} = \mathbf{J_c^r}\dot{\mathbf{q}} \tag{23}$$

where C_x and C_y are the coordinates of C along axes $\vec{x}_P$ and $\vec{y}_P$ (see figure 2.b). Notice that $\mathbf{J_c^r}$ is a regular matrix as det($\mathbf{J_c^r}$)=$D_x \neq 0$.

3.2 Execution of a vision-based task despite camera failure

Our objective is to perform a vision-based task despite camera failure. We first describe the considered mission and state the estimation problem for this particular task before presenting the obtained results.

3.2.1 Vision-based task

Our goal is to position the embedded camera with respect to a visual landmark. To this aim, we have applied the visual servoing technique given in (Espiau et al., 1992) to mobile robots as in (Pissard-Gibollet & Rives, 1995). The proposed approach relies on the task function formalism (Samson et al., 1991) and consists in expressing the visual servoing task by the following task function to be regulated to zero:

$$e_{vs} = \mathbf{C}(\mathbf{s} - \mathbf{s}^*)$$

where **s*** represents the desired value of the visual signal, while **C** is a full-rank combination matrix allowing to take into account more visual features than available DOFs (Espiau et al., 1992). A classical controller $\dot{\mathbf{q}}_{(\mathbf{s})}$ making e_{vs} vanish can be designed by imposing an exponential decrease, that is: $\dot{e}_{\text{vs}} = -\lambda e_{\text{vs}} = \mathbf{CL}_{(\mathbf{s},\mathbf{z})}\mathbf{J}_{\mathbf{c}}^{\mathbf{r}}\dot{\mathbf{q}}_{(\mathbf{s})}$, where λ_{vs} is a positive scalar or a positive definite matrix. Fixing **C**= $\mathbf{L}_{(\mathbf{s}^*,\mathbf{z}^*)}$ as in (Espiau et al., 1992), the visual servoing controller $\dot{\mathbf{q}}_{(\mathbf{s})}$ can be written as follows:

$$\dot{\mathbf{q}}_{(\mathbf{s})} = \left(\mathbf{CL}_{(\mathbf{s},\mathbf{z})}\mathbf{J}_{\mathbf{c}}^{\mathbf{r}}\right)^{-1}(-\lambda)\mathbf{L}_{(\mathbf{s}^*,\mathbf{z}^*)}{}^{+}(\mathbf{s}-\mathbf{s}^*) \tag{24}$$

3.2.2 Control strategy

As previously mentioned, the goal is to perform a positioning vision-based task with respect to a landmark, despite visual data loss due to a camera failure. The robot will be controlled in different ways, depending on the visual data availability. Two cases may occur: either the camera is able to provide the visual data or not. In the first case, controller (24) can be directly applied to the robot and the task is executed as usually done. In the second case, we use our estimation technique to compute an estimation of the visual data vector $\tilde{\mathbf{s}}$. It is then possible to evaluate controller (24). Hence, during a camera failure, the vehicle is driven by a new controller: $\dot{\mathbf{q}}_{(\tilde{\mathbf{s}})} = \left(\mathbf{CL}_{(\tilde{\mathbf{s}},\mathbf{z})}\mathbf{J}_{\mathbf{c}}^{\mathbf{r}}\right)^{-1}(-\lambda)\mathbf{L}_{(\mathbf{s}^*,\mathbf{z}^*)}{}^{+}(\tilde{\mathbf{s}}-\mathbf{s}^*)$. Therefore, we propose to use the following global visual servoing controller:

$$\dot{\mathbf{q}}_{\text{vs}} = (1-\sigma)\dot{\mathbf{q}}_{(\mathbf{s})} + \sigma\dot{\mathbf{q}}_{(\tilde{\mathbf{s}})} \tag{25}$$

where $\sigma \in [0; 1]$ is set to 1 when the image is unavailable. This reasoning leads to the control architecture shown in figure 3. Note that, generally, there is no need to smooth controller (25) when the image features are lost and recovered (if the camera failure is temporary). Indeed, when the failure occurs, as the last provided information are used to feed our reconstruction algorithm, the values of **s** and $\tilde{\mathbf{s}}$ are close and so are $\dot{\mathbf{q}}_{(\mathbf{s})}$ and $\dot{\mathbf{q}}_{(\tilde{\mathbf{s}})}$. Usually, the same reasoning holds when the visual features are available anew. However, some smoothing may be useful if the camera motion has been unexpectedly perturbed during the estimation phase or if the algorithm has been given too inaccurate initial conditions. In such a case, it will be necessary to smooth the controller by defining σ as a continuous function of time *t* for instance.

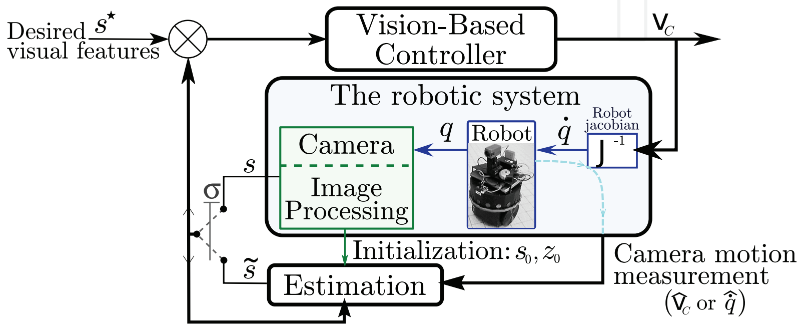


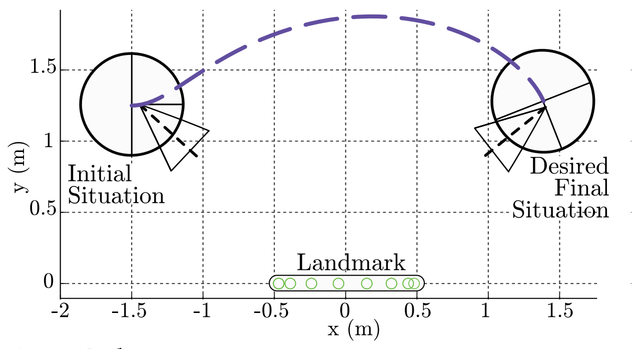
Fig. 3. The chosen control architecture.

3.3 Results

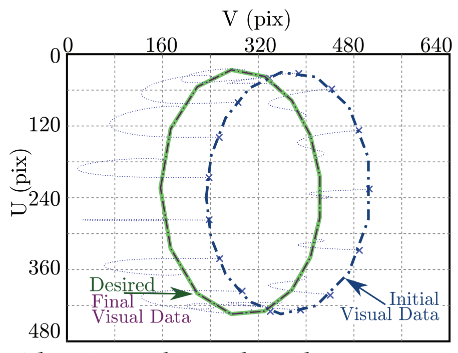
Our goal is then to position the embedded camera with respect to a given landmark despite camera failure. We will use controller (25). We propose hereafter some simulation results together with experimental tests to show the interest and the efficiency of the different proposed estimation techniques. We first present the obtained results in the two following subsections. Precise comments together with a detailed comparative analysis are provided in next part.

3.3.1 Simulation results

To validate our work, we have first realized numerous simulations using Matlab software. We have considered different kinds of visual features: points, ellipses and image moments. For each case, we have performed the same robotic task, starting from the same initial configuration to reach the same goal **s*** (see Figure 4). In a similar way, the camera failure occurs after 50 steps and lasts until the end of the mission. In this way, we guarantee that the multi-step numerical schemes can be correctly initialized6. The control law sampling period has been chosen equal to T_s = 50ms, which is close to its value on the robotic platform. The control interval has been divided in N=10 integration step, that is T_n=5ms.



4.a – Robot trajectory.



4.b – Visual Landmark.

Fig. 4. Simulated robotic task.

Our goal is here to perform the vision-based navigation task represented on the figure 4.a. It consists in positioning the embedded camera with respect to a landmark made of one ellipse described on figure 4.b. This landmark can be modelled by three different visual primitives: a set of points belonging to the ellipse, the ellipse features itself, and finally (at least) two image moments. This example allows us to illustrate the different ways of stating the estimation problem shown in section 2. We present below the obtained results for each of these primitives.

Case of points: In this case, we consider 10 points belonging to the ellipse. Thus, the image features vector is defined by: $\mathbf{s}=\left[X_1,Y_1,\cdots,X_{10},Y_{10}\right]^T$, where ($X_i$, Y_i) are the coordinates of each projected point P_i. Let us recall that the dynamic system to be solved is given by (4).

The table 1 synthesis the simulation results obtained using the proposed numerical and analytical methods for points. More precisely, it shows the maximal and the standard deviation (std) error of the euclidean norm of the set of points (i.e. $\left\|\mathbf{s}-\tilde{\mathbf{s}}\right\|$) and of their depth (i.e. $\left\|\mathbf{z}-\tilde{\mathbf{z}}\right\|$).

	Euler error		RK4 error		ABM4 error	
	std	max	std	max	std	max
$\|\mathbf{s}\|$	$0.947 .10^{-3}$	$2.450 . 10^{-3}$	$0.299 . 10^{-3}$	$1.133 . 10^{-3}$	$0.136 . 10^{-3}$	$0.556 . 10^{-3}$
$\|\mathbf{z}\|$	$1.935 . 10^{-3}$	$8.949 . 10^{-3}$	$1.829 . 10^{-3}$	$7.624 . 10^{-3}$	$0.379 . 10^{-3}$	$1.906 . 10^{-3}$

	BDF4 error		Analytic error ($\mathbf{v}_c$ constant)		Analytic error ($\mathbf{v}_c$ variable)	
	std	max	std	max	std	max
$\|\mathbf{s}\|$	$0.311 .10^{-3}$	$2.109 . 10^{-3}$	$0.199 . 10^{-3}$	$0.863 . 10^{-3}$	$1.779 . 10^{-12}$	$31.724 . 10^{-12}$
$\|\mathbf{z}\|$	$0.892 . 10^{-3}$	$7.784 . 10^{-3}$	$0.968 . 10^{-3}$	$6.456 . 10^{-3}$	$8.795 . 10^{-12}$	$96.367 . 10^{-12}$

Table 1. Point simulation results ($|\mathbf{s}|$ in pixel, and $|\mathbf{z}|$ in mm).

Case of ellipse: We consider now the ellipse itself described on figure 4.b. Hence, we first recall that an ellipse can be defined by the following quadric equation:

$$X_i^2 + E_1 Y_i^2 + E_2 X_i Y_i + E_3 X_i + 2E_4 Y_i + E_5 = 0 \tag{26}$$

The visual features vector can expresses as: $\mathbf{s} = [E_1, E_2, E_3, E_4, E_5, X_1, Y_1, \cdots, X_{25}, Y_{25}]^T$, where ($X_i$, Y_i) are the coordinates of the points belonging to the ellipse. This set of l=25 points allows to identify the ellipse A_{pq} 3D parameters (see algorithm 2). The differential system to be solved can then be expressed as follows:

$$\dot{\mathbf{s}} = \begin{bmatrix} \mathbf{L}_{(E_1, A_{pq})} \\ \\ \mathbf{L}_{(E_5, A_{pq})} \\ \mathbf{L}_{(P_1, z_1)} \\ \\ \mathbf{L}_{(P_{25}, z_{25})} \end{bmatrix} \mathbf{v}_c = \mathbf{L}_{(E_1, E_2, E_3, E_4, E_5, A_{pq}, X_1, Y_1, \cdots, X_{25}, Y_{25})} \mathbf{v}_c \tag{27}$$

where $\mathbf{L}_{(P_i, z_i)}$ is the interaction matrix related to the point P_i given by (3) , and $\mathbf{L}_{(E_i, A_{pq})}$ is the interaction matrix related to the E_i ellipse quadric parameters. The $\mathbf{L}_{(E_i, A_{pq})}$ expressions are available in (Espiau et al., 1992). As the points were presented in the previous paragraph, we focus here on the ellipse. Thus, we focus on the ellipse quadric parameters E_i estimation results, for which only numerical techniques can be applied. Moreover, each estimated point P_i belonging to the ellipse has to fit the relation (26). Therefore, in this case, the estimator efficiency can be evaluated according to the following relation:

$$\varepsilon_{El} = \max_i \left(X_i^2 + E_1 Y_i^2 + E_2 X_i Y_i + E_3 X_i + 2E_4 Y_i + E_5 \right) \tag{28}$$

Hence, the table 2 summarizes the estimation error of E_i parameters and the ε_{El} error for each considered numerical scheme.

Case of moments: The landmark shown on figure 4.b has been characterized by points and ellipses. It is also possible to use the two following image moments: the area (i.e. m_{00}), and the gravity center, and the gravity center ($X_g = \frac{m_{10}}{m_{00}}, Y_g = \frac{m_{01}}{m_{00}}$) of the ellipse. In this case, the visual features vector is set to: $\mathbf{s} = [m_{00}, X_g, Y_g, X_1, Y_1, \cdots, X_{15}, Y_{15}]^T$. As previously mentioned the set of l=15 points allows to approximate the image moments features. The table 3 describes the image moments estimation error.

	Euler error		RK4 error		ABM4 error		BDF4 error	
	std	max	std	max	std	max	std	max
E_1	$6.194 \,.\, 10^{-4}$	$6.353 \,.\, 10^{-3}$	$2.181.\, 10^{-4}$	$5.469 \,.\, 10^{-3}$	$1.821 \,.\, 10^{-4}$	$3.846 \,.\, 10^{-3}$	$1.825 \,.\, 10^{-4}$	$3.477 \,.\, 10^{-3}$
E_2	$4.741 \,.\, 10^{-5}$	$2.315 \,.\, 10^{-4}$	$3.517.\, 10^{-5}$	$1.233 \,.\, 10^{-4}$	$2.878 \,.\, 10^{-5}$	$1.032 \,.\, 10^{-4}$	$3.092 \,.\, 10^{-5}$	$1.011 \,.\, 10^{-4}$
E_3	$4.967 \,.\, 10^{-5}$	$2.199 \,.\, 10^{-4}$	$3.995 \,.\, 10^{-5}$	$1.891 \,.\, 10^{-4}$	$3.179 \,.\, 10^{-5}$	$1.299 \,.\, 10^{-4}$	$3.101 \,.\, 10^{-5}$	$1.184 \,.\, 10^{-4}$
E_4	$4.569 \,.\, 10^{-4}$	$2.181 \,.\, 10^{-3}$	$3.157 \,.\, 10^{-4}$	$1.177 \,.\, 10^{-3}$	$2.357 \,.\, 10^{-4}$	$1.067 \,.\, 10^{-3}$	$2.109 \,.\, 10^{-4}$	$1.019 \,.\, 10^{-3}$
E_5	$2.328 \,.\, 10^{-4}$	$6.741 \,.\, 10^{-4}$	$1.314 \,.\, 10^{-4}$	$5.762 \,.\, 10^{-4}$	$1.087 \,.\, 10^{-4}$	$5.096 \,.\, 10^{-4}$	$1.006 \,.\, 10^{-4}$	$4.934 \,.\, 10^{-4}$
ε_{El}	0.2027	0.8398	0.1512	0.7616	0.1284	0.5756	0.1071	0.6056

Table 2. Ellipse features simulation results.

	Euler error		RK4 error		ABM4 error		BDF4 error	
	std	max	std	max	std	max	std	max
m_{00}	6.1044	19.983	5.8346	18.1033	4.194	16.769	2.4298	10.3209
X_g	$1.768 \,.10^{-3}$	$4.253 \,.10^{-3}$	$1.278 \,.10^{-3}$	$3.526 \,.10^{-3}$	$0.805 \,.10^{-3}$	$2.404 \,.10^{-3}$	$0.449 \,.10^{-3}$	$1.923 \,.10^{-3}$
Y_g	$4.337 \,.10^{-3}$	$13.243 \,.10^{-3}$	$2.371 \,.10^{-3}$	$12.304 \,.10^{-3}$	$1.503 \,.10^{-3}$	$10.115 \,.10^{-3}$	$1.345 \,.10^{-3}$	$6.834 \,.10^{-3}$

Table 3. Image moments features simulation results (area m_{00} in pixel², and (X_g, Y_g) in pixel)

3.3.2 Experimental results

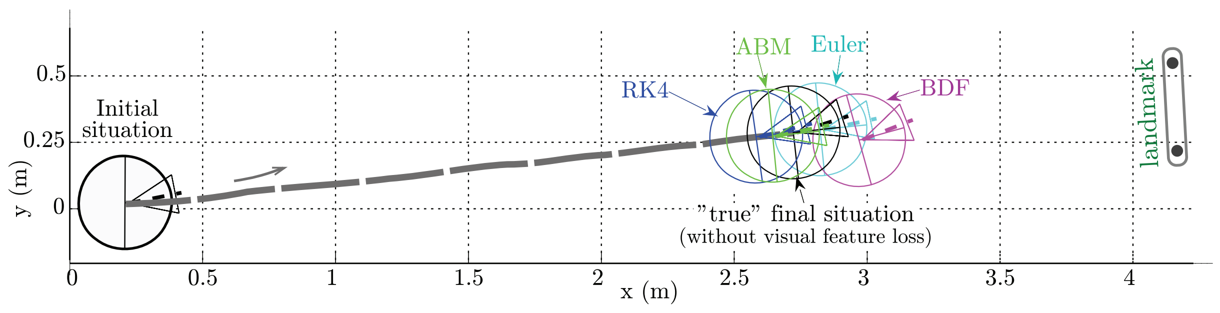


Fig. 5. Robot trajectory, using numerical schemes.

We have also experimented our approaches on our Super-Scout II. We have considered once again a vision-based navigation task which consists in positioning the embedded camera in front of a given landmark made of n=4 points. First of all, we address the validation of the proposed numerical schemes. For each of them, we have performed the same navigation task: start from the same configuration using the same **s*** (see figure 5). At the beginning of the mission, the robot is driven using the visual features available from the camera and starts converging towards the target. At the same time, the numerical algorithms are initialized and launched. After 10 steps, the landmark is artificially occluded to simulate a camera failure and, if nothing is done, it is impossible to perform the task. The controller is then evaluated using the computed values provided by our proposed method.

In a second step, we have validated the analytical method which take into account the vc time variation on our robotic platform (see figure 6). We have followed the same experimental procedure as for numerical schemes. Indeed, as previously, the visual data are available at the beginning of the task and the robot is controlled using (24). After a few steps, the landmark is manually occluded. At this time, the visual signals are computed by our estimation procedure and the robot is driven using controller. It is then possible to keep on executing a task which would have aborted otherwise.

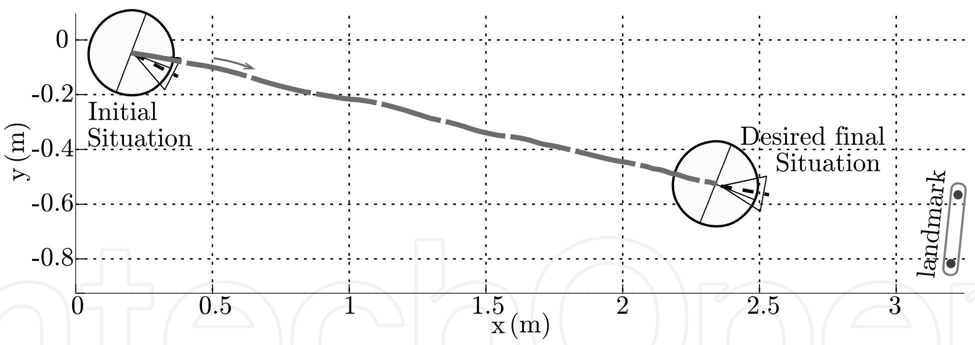


Fig. 6. Robot trajectory, using analytical method ($\mathbf{v}_c$ variable during [t_k; t_{k+1}])

	Euler error		RK4 error		ABM4 error		BDF4 error		Analytic error ($\mathbf{v}_c$ variable)	
	std	max	std	max	std	max	std	max	std	max
$\lvert \mathbf{s} \rvert$	1.0021	9.6842	0.9092	7.0202	0.9003	5.9256	1.1172	7.6969	0.1275	0.7657
$\lvert \mathbf{z} \rvert$	0.0883	0.7233	0.0721	0.6385	0.0572	0.5064	0.1016	0.5989	0.0143	0.0637

Table 4. Experimental results ($\lvert \mathbf{s} \rvert$ in pixel, and $\lvert \mathbf{z} \rvert$ in m).

The table 4 summarizes the whole results obtained in the case of points. These errors remain small, which means that there are few perturbations on the system and, thanks to our estimation method, it is possible to reach a neighborhood of the desired goal despite the camera failure. Once again the analytical method gives the best estimation results. Moreover, for the proposed task, the ABM4 scheme is the most efficient numerical method, as it leads to the least standard deviation error (std) and to the smallest maximal error. The RK4 algorithm gives also correct performances, while Euler method remains the less accurate scheme as expected. As T_s is rather small, the BDF4 technique provides correct results but has been proven to be much more efficient when there are sudden variations in the kinematic screw (stiff context).

3.4 Comparative analysis and additional comments

The previous part has been devoted to the validation of the different reconstruction algorithms in the visual servoing context proposed in section 2. To this aim, we have focused on a specific problem which may occur during a vision-based task: the loss of the visual features due to a camera or an image processing failure. The presented simulation and experimental results have demonstrated the efficiency and the interest of our approach.
Now, on the base of these results, our objective is to compare the different approaches and to exhibit their advantages and drawbacks. We also aim at giving some elements allowing to select the most suitable method depending on the context (considered visual features, task to be realized, and so on). All the tests have shown that the analytical solution integrating the true variation of the camera kinematic screw is the most accurate approach. This result is quite consistent because this solution explicitely takes into account the robot mechanical structure and appears to be the closest to the real system. Thus, the estimation errors appear to be negligible. In a similar way, the other analytical solution also leads to nice results (equivalent to the ones obtained with numerical approaches), but is less precise than the previous one. However, these two approaches are restricted to the case where the considered image features are points and cannot be used for other kinds of visual

primitives. Moreover, as previously mentioned, one of them depends strongly on the mechanical structure of the robot on which is embedded the camera. To relax these restrictions, it is necessary to treat the problem using numerical schemes. Among them, the Gear's method (BDF4) and the Adams–Bashforth–Moulton scheme (ABM4) are the most accurate. It can be shown that the first one is particularly efficient when the dynamics of the system to be solved varies rapidly, that is when the problem becomes stiff. For example, this phenomenon has been observed when the robot has to avoid obstacles to safely perform the desired vision-based navigation task (Folio & Cadenat, 2007). In other cases (few variations on the camera kinematic screw), ABM4 scheme appears to provide better results, although it is sometimes limited by cumulative errors which occur when the visual primitives are reconstructed following several steps as in algorithms 1 and 2. Finally, the Runge–Kutta fourth order (RK4) algorithm and Euler schemes are the less accurate because they do not consider any history of the image features to be estimated. The above table 5 summarizes this comparative analysis.

Methods	Advantages	Drawbacks	Interest
Analytical solution $\mathbf{v}_c$ variable during $[t_k; t_{k+1}]$	High accuracy	Solution specific to a Super-Scout II-like robotic system and to points	+++
Analytical solution $\mathbf{v}_c$ constant during $[t_k; t_{k+1}]$	• Allows to consider different robotic systems • Good accuracy	Solution specific to points	+
Euler Scheme	Easy to implement	• The least accurate • Require small sampling period	--
RK4 Scheme	More accurate than Euler scheme		-
ABM4 Scheme	• Reconstruction based on a predictor/corrector pair. • The local truncation can be estimated.	• The scheme initialization requires a history of values of ψ obtained at previous instants. • Less accurate when successive approximations are performed.	+
BDF4 Scheme	The most efficient approach when $\mathbf{v}_c$ significantly varies	The scheme initialization requires a history of values of ψ obtained at previous instants.	++

Table 5. Comparative table

Only 4th order numerical schemes have been considered in this work. It is important to note that using greater orders does not necessarily lead to better results. Usually, a more suitable strategy is to reduce the integration step T_n. However, it must be noticed that in such a case the approximations are more and more cumulated and that the estimation error does not decrease anymore with T_n as one could have expected. Therefore, the most efficient way allowing to improve the quality of our algorithms is to reduce the control law sampling period.

Furthermore, as one could have expected, simulation provides better results than experimentation. In particular, when testing our algorithm, we have to deal with several constraints related to our robotic platform. First, the implementation of our algorithms requires to have the necessary data for initialization and to know precisely the values of T_s and T_n. These conditions are not necessarily fulfilled on experimental platforms, as it mainly depends on the operating system running on them. For instance, on the Super-Scout II, T_s cannot be precisely obtained. Moreover, our modelling does not take into account the noises which appear on the image features extraction processing and on the measurement of the robot velocities $\dot{\mathbf{q}}$.

Finally, let us recall that our main goal was to provide a generic framework to reconstruct the visual features whenever they become unavailable. In particular, we have shown that common visual primitives cannot be computed if the A_{pq} 3D parameters are not previously estimated. We have then proposed a solution consisting in using points to identify them. However, our validation work has demonstrated that this solution does not provide a totally efficient estimation because of the successive approximations induced by the procedure. Thus, the estimation algorithm could be improved by computing A_{pq} together with s as done for points. However, in such a case, the result would be restricted to the considered visual primitives, whereas the proposed solution based on points presents the advantage of the genericity.

In conclusion, the proposed reconstruction algorithms have been successfully validated in the vision-based navigation task. The obtained results have demonstrated the efficience and the relevancy of our approach to treat the specific problem of image features loss during a visual servoing task. Thanks to our method, as soon as an analytical expression of $\mathbf{L}_{(\mathbf{s},\mathbf{z})}$ is available, it is possible to reconstruct the visual data when needed and to keep on performing a task which should have been aborted otherwise.

4. Conclusion

In this chapter, we addressed the problem of computing the image features when they become unavailable during a vision-based task. To this aim, in a first step, we have elaborated different algorithms able to reconstruct the visual signals when they are lost. The proposed techniques rely on the camera kinematic screw and on the last measured perceptual cues. We have then shown that the problem can be expressed as the resolution of a dynamic system and we have developed different techniques allowing to solve it. We have proposed both analytical and numerical solutions. The first ones are very accurate, but appear to be limited to specific image features and dedicated to a particular robot mechanical structure. The second ones are less precise but present the advantage of genericity, as they can be applied in a general context (any kind of visual data and of robotic systems). It is then possible to obtain an estimation of any kind of image features independently from the robot on which is embedded the camera. In a second step, we have demonstrated the validity of our algorithm in the visual servoing context, considering the case of a positioning task during which a camera failure occurs. The obtained simulation and experimentation results have demonstrated the relevancy of our techniques to keep on performing the mission despite such a problem. Finally, we have ended our study by proposing a comparative analysis of the different elaborated algorithms.

These works have opened several interesting issues. First, the designed analytical solutions are restricted to the case of points and, for one of them, to the considered robotic system.

Although it is a priori difficult to develop such a solution for the general case, a natural extension would then to solve this problem for other kinds of robots and of visual primitives. Moreover, as the analytical solution directly provides an expression of the depth, it would be interesting to use it together with approaches such as tracking algorithms or camera pose reconstruction techniques. Finally, our results could also be successfully applied in other related fields than visual servoing. For example, it would be interesting to use them in a fault tolerance context to detect and correct errors in image processing algorithms.

5. References

Benhimane, S. & Malis, E. (2003). Vision-based control with respect to planar and non-planar objects using a zooming camera. *Proceedings of International Conference on Advanced Robotics*,Vol. 2, pp. 991–996, Coimbra, Portugal.

Butcher, J. C. (2008). *Numerical Methods for Ordinary Differential Equations* (2nd ed.). Wiley, ISBN:9780470723357.

Chaumette, F. (2004). Image moments: a general and useful set of features for visual servoing. *IEEE Transaction on Robotics*, Vol. 20, No. 4, pp. 713–723.

Chaumette, F.; Boukir, S.; Bouthemy, P. & Juvin, D. (1996, May). Structure from controlled motion. *IEEE Transaction Pattern Anal. Machine Intell.*, Vol. 18, No. 5, pp. 492–504.

Chaumette, F. & Hutchinson, S. (2006). Visual servo control, part I: Basic approaches. *IEEE Robotics and Automation Magazine*, Vol. 13, No. 4, pp. 82–90.

Comport, A. I.; Marchand, E. & Chaumette, F. (2004). Robust model-based tracking for robot vision. *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, Vol. 1, pp. 692–697, Sendai, Japan.

Corke, P. & Hutchinson, S. (2001). A new partitioned approach to image-based visual servo control. *IEEE Transaction Robotics and Automation*, Vol. 17, pp. 507–515.

Espiau, B., Chaumette, F.;, & Rives, P. (1992, June). A new approach to visual servoing in robotics. *IEEE Transaction Robotics and Automation*, Vol. 8, pp. 313–326.

Favaro, P. & Soatto, S. (2003). Seeing beyond occlusions (and other marvels of a finite lens aperture). *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Vol. 2, pp. II-579–II-586, Saint -Louis, USA.

Fleury, S. & Herrb, M. (2001). GenoM: User Manual. LAAS-CNRS.

Folio, D. (2007). Stratégies de commande référencées multi-capteurs et gestion de la perte du signal visuel pour la navigation d'un robot mobile. PhD thesis, University of Toulouse, France.

Folio, D. & Cadenat, V. (2005a). A controller to avoid both occlusions and obstacles during a vision-based navigation task in a cluttered environment. *Proceedings of European Control Conference*, pp. 3898–3903, Seville, Spain.

Folio, D. & Cadenat, V. (2005b). Using redundancy to avoid simultaneously occlusions and collisions while performing a vision-based task amidst obstacles. *Proceedings of European Conference on Mobile Robots*, pp. 92–97, Ancona, Italy.

Folio, D. & Cadenat, V. (2007). A new controller to perform safe vision-based navigation tasks amidst possibly occluding obstacles. *Proceedings of European Control Conference*, pp. 1448–1454, Kos, Greece.

Folio, D. & Cadenat, V. (2008). A sensor-based controller able to treat total image loss and to guarantee non-collision during a vision-based navigation tasks. *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, Nice, France.

Garcia-Aracil, N.; Malis, E.; Aracil-Santonja, R. & Perez-Vidal, C. (2005). Continuous visual servoing despite the changes of visibility in image features. *IEEE Transaction Robotics and Automation*, Vol. 21, pp. 1214–1220.

Jerian, C. & Jain, R. (1991). Structure from motion: A critical analysis of methods. *IEEE Transaction on Systems, Man and Cybernetics*, Vol. 21, No. 3, pp. 572–588.

Kyrki, V., Kragic, D. & Christensen, H. (2004, October). New shortest-path approaches to visual servoing. *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 349–355*.

Lepetit, V. & Fua, P. (2006). Monocular model-based 3d tracking of rigid objects. *Foundations and Trends in Computer Graphics and Vision*, Vol. 1, No. 1, pp. 1–89.

Mansard, N. & Chaumette, F. (2005). A new redundancy formalism for avoidance in visual servoing. *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, Vol. 2, pp. 1694–1700, Edmonton, Canada.

Marchand, E. & Hager, G. (1998). Dynamic sensor planning in visual servoing. *Proceedings of IEEE International Conference on Robotics and Automat.*, Vol. 3, pp. 1988–1993, Leuven, Belgium.

Matthies, L.; Kanade, T. & Szeliski, R. (1989). Kalman filter-based algorithms for estimating depth in image sequences. *International Journal of Computer Vision*, Vol. 3, No. 3, pp. 209–238.

Mezouar, Y. & Chaumette, F. (2002). Avoiding self-occlusions and preserving visibility by path planning in the image. *Robotics and Autonomous Systems*, Vol. 41, No. 2, pp. 77–87.

Murray, R.; Li, Z. & Sastry, S. (1994). *A mathematical introduction to robotic manipulation* (1st ed.). CRC Press, ISBN:0849379814, Boca Raton, FL, USA.

Oliensis, J. (2002, December). Exact two-image structure from controlled motion. *IEEE Transaction Pattern Anal. Machine Intell.*, Vol. 24, No. 12, pp. 1618–1633.

Pissard-Gibollet, R. & Rives, P. (1995). Applying visual servoing techniques to control a mobile hand-eye system. *Proceedings of IEEE International Conference on Robotics and Automation*, Vol. 1, pp. 166–171, Nagoya, Japan.

Remazeilles, A.; Mansard, N. & Chaumette, F. (2006). Qualitative visual servoing:application to the visibility constraInternational *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4297–4303, Beijing, China.

Samson, C.; Borgne, M. L.; & Espiau, B. (1991). *Robot Control : The task function approach*. Oxford University Press, ISBN:0198538057, Oxford.

Shampine, L. F. & Gear, C. W. (1979). A user's view of solving stiff ordinary differential equations. *Society for Industrial and Applied Mathematics Review*, Vol. 21, No. 1, pp. 1–17.

Singer, M. (1993). Reducing structure from motion: A general framework for dynamic vision part 2: Implementation and experimental assessment. *Pattern Recognition*, Vol. 26, No. 7, pp. 1019–1028.

Soatto, S. & Perona, P. (1998). Reducing structure from motion: A general framework for dynamic vision part 2: Implementation and experimental assessment. *IEEE Transaction Pattern Analysis and Machine Intelligence*, Vol. 20, No. 9, pp. 943–960.

Steger, C. (1996). On the calculation of arbitrary moments of polygons. FGBV-96-05, Technische Universität München.

Tahri, O. & Chaumette, F. (2005). Point-based and region-based image moments for visual servoing of planar objects. *IEEE Transaction on Robotics*, Vol. 21, No. 6, pp. 1116–1127.

Thrun, S.; Fox, D.; Burgard, W.; & Dallaert, F. (2001). Robust mote-carlo localization for mobile robots. *Artifial Intelligence*, Vol. 128, No. 1-2, pp. 99–141.