High Speed Parallel Kinematic Manipulator State Estimation from Legs Observation

Erol Özgür, Redwan Dahmouche, Nicolas Andreff, Philippe Martinet

***Abstract*— To control dynamics of a parallel robot, we should measure the state feedback accurately and fast. In this paper, we show how to estimate positions and velocities simultaneously (i.e., the state feedback) at a reasonable accuracy and speed. We did this using only the sequential visual contours of the legs. A single-iteration virtual visual servoing scheme regulates rapidly an error of these contours. We validated this theory, a step to control parallel robots at high speed by their leg kinematics, with simulations and experiments.**

I. INTRODUCTION

Dynamic control of robots requires fast feedback. Motor encoders feed back the joint values fast enough to control dynamics of serial robots. However, one still wishes to use vision-based control schemes [1] for better performances of robots, even though it is difficult to satisfy a reasonable accuracy and frequency for visual feedback.

In serial robots, the solution for vision-based dynamic control is found in a cascade of two control loops: (i) The first inner loop, and also the fast one, compensates for the dynamics. This inner loop uses an inverse dynamic model based on joint values provided by the motor encoders at high frequency. (ii) The second outer loop, slower one, uses feedback of a vision sensor. This outer loop is actually a kinematic control made possible by the inner loop which compensates for the dynamics of the serial robot.

This approach does not work for parallel robots. Because the joint values do not always represent the state of a parallel robot. The state of a parallel robot for dynamic control can be expressed by its end-effector pose and velocity – yet it is possible to express the state with different variables.

Therefore, we should now think how to compute pose and velocity at high speed with vision sensor. Some of the attempts to adapt vision for control schemes are as follows: Since the state of a parallel robot is expressed by its end-effector pose and velocity, these variables are computed with classical pose estimation algorithms. Unfortunately, these algorithms cannot give directly the velocity. The pose velocity is usually computed by numerical differentiation of the estimated pose, consequently introducing additional noise. Besides, predictive control techniques are exploited to adapt the visual sampling rate to the control sampling rate, but this increases the complexity [2]. Instead, from a control viewpoint, increasing visual feedback frequency to the control frequency is more appropriate [3], [4]. Then to do so, one compresses image data [5], builds fast communication interfaces, or embeds image processing unit closer to the camera [6], [7], [8], etc. These attempts complicate the system, too.

In previous work, we reviewed the standard dynamic control schemes for PKM [9] showing that theoretically observing the end-effector was far preferable to using joint encoders. Further, we demonstrated that observing the end-effector pose and velocity with a sequential grabbing strategy (one feature point after the other) could overpass the performances of the joint-based computed-torque control of a PKM [10]. Note that acceleration needs not be estimated in such control schemes since it is the input variable. We also demonstrated [11] that observing the kinematic elements (legs) of a PKM was a relevant alternative to observing the end-effector: less calibration, same accuracy as standard kinematic visual servoing. Of course, mixing vision with joint sensing *could* be of some interest (especially for vision-sceptic people), but we prefer to focus on the next challenge: dynamic control based on leg observation in a 100% vision way. We already showed in simulation [12] that the tracking of the legs associated to the control in the Cartesian space outperformed all other schemes (including joint-space control and Cartesian-space control with direct observation of the end-effector). The purpose of this paper is therefore to show how the tracking of the legs at high frequency can be achieved.

For this purpose we propose to grab small sub-images that allow only for local observations. In a parallel robot, since legs exist plentifully, this implies a *sequential grabbing strategy* (one by one) to collect enough information from the whole mechanism. In these sub-images, we observe contours of the legs. Image contours are one of the simplest visual features to detect. Then, required information is the contours of the legs which are grabbed sequentially at discrete instants of the motion of the robot.

However, although simultaneous multiple sub-images grabbing strategy (non-sequential) provides all the information at once, it is not preferable because of addressing problem and because it is slower for estimation. To compute the full state at each control sample time, the following methods are proposed: An extended Kalman filter predicts relative state of the robot by fusing a set of image feature points belonging to an object with some redundant measurements [6]. A CMOS Rolling Shutter camera captures a single image row by row. This causes image artifacts for the moving objects. Then, exploiting these visual artifacts, the pose and velocity of a moving object are simultaneously estimated with a non-

This work is supported by the ANR-VIRAGO project.

The authors are with Pascal Institute, CNRS, Clermont Universit*é*s, Clermont-Ferrand, France. N. Andreff is also with FEMTO-ST Institute, Universit*é* de Franche-Comt*é*, Besançon, France. firstname.lastname@lasmea.univ-bpclermont.fr

linear least squares method [13]. A virtual visual servoing scheme [14] estimates the state variables by sequentially grabbing blobs of a rigid pattern at high speed [15], [10].

Our work also exploits the virtual visual servoing scheme associated with sequential grabbing strategy as in [10]. And improves it to track an articulated set of legs of a parallel robot at high speed rather than to track a rigid pattern fastened to the end-effector at high speed. Yet, since we are good without any pattern, in a roundabout way, this gets rid of calibration between the pattern, the camera and the end-effector. The objective of this work is:

- to estimate the posture and velocity (without derivation) of a parallel robot through a copy virtual robot imitating its motion.

The rest of this paper goes on as follows: Section II provides background information about the kinematics of legs. Section III explains the estimation of the state variables through a single-iteration virtual visual servoing scheme. Section IV shows the simulation and experimental results. Section V concludes the paper and offers some perspectives.

II. Edge Kinematics of a Cylindrical Leg

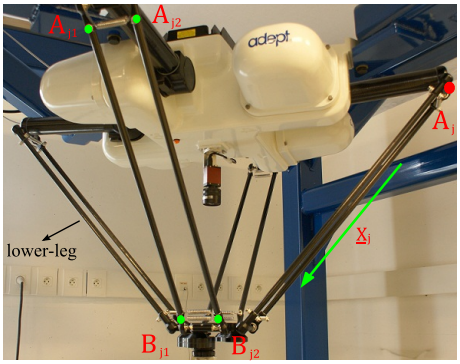

Fig. 1. The Quattro parallel robot with a base-mounted camera.

A. Notation

- $\mathbf{B} \in \Re^{3\times 1}$ is one of the tip points lying on the revolution axis of the cylindrical leg (see Fig. 1).
- $\underline{\mathbf{x}} \in \Re^{3\times 1}$ is the orientation unit vector and also corresponds to the revolution axis direction of the leg.
- $s \in \{L, R\}$ is the literal representation for the (L)eft or the (R)ight side of the leg seen from the camera.
- $\mathbf{p}_s \in \Re^{3\times 1}$ is a projection contour point lying on the image plane and located inside a visual edge of the cylindrical leg, $\mathbf{p}_s = [x, y, 1]^T$ (see Fig. 2).
- $\underline{\mathbf{n}}_s \in \Re^{3\times 1}$ is a unit vector orthogonal to the plane defined by the projection center $\mathbf{O}$ of the camera and by a side visual contour of the leg. Hence, it stands for a mathematical representation of such a visual contour.
- $\mathbb{X} \in \Re^{n\times 1}$ is the end-effector pose of a parallel robot.
- Unless specified in a left supper-script, all the variables are expressed in the camera frame.

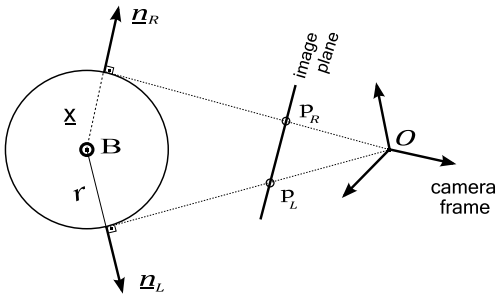

Fig. 2. View of the geometry of a cylindrical leg from its 3D orientation direction $\underline{\mathbf{x}}$ (perpendicular to the paper plane).

B. Edge Kinematics

Let $M_x \in \Re^{3\times n}$ and $L_B \in \Re^{3\times n}$ be the differential kinematic models that relate the velocity of the end-effector pose to the velocity of the orientation unit vector $\underline{\mathbf{x}}$ and to the velocity of the tip point $\mathbf{B}$ of the cylindrical leg in a kinematic chain:

$$\dot{\underline{\mathbf{x}}} = M_x\, \dot{\mathbb{X}}, \qquad \dot{\mathbf{B}} = L_B\, \dot{\mathbb{X}} \tag{1}$$

The geometry of a cylindrical leg (see Fig. 2) imposes the following constraints [16]:

$$\mathbf{B}^T\, \underline{\mathbf{n}}_s = -r, \qquad \underline{\mathbf{x}}^T\, \underline{\mathbf{n}}_s = 0, \qquad \mathbf{p}_s^T\, \underline{\mathbf{n}}_s = 0 \tag{2}$$

where r is radius of the leg. Since we would like to exploit contours of a leg, the third constraint will be used in the virtual visual servoing scheme. This requires completion of a differential model defined between an edge $\underline{\mathbf{n}}_s$ and the end-effector pose $\mathbb{X}$. Exploiting the second constraint and knowing that $\dot{\underline{\mathbf{n}}}_s^T\, \underline{\mathbf{n}}_s = 0$, the image velocity of the contour $\dot{\underline{\mathbf{n}}}_s$ is expressed as follows:

$$\dot{\underline{\mathbf{n}}}_s = \alpha\, \underline{\mathbf{x}} + \beta\, (\underline{\mathbf{x}} \times \underline{\mathbf{n}}_s) \tag{3}$$

where α and β are two unknown scalars. The α comes out by differentiating the second constraint in (2) and replacing (3) into the differentiated second constraint, which yields:

$$\dot{\underline{\mathbf{x}}}^T\, \underline{\mathbf{n}}_s + \underline{\mathbf{x}}^T\, (\alpha\, \underline{\mathbf{x}} + \beta\, (\underline{\mathbf{x}} \times \underline{\mathbf{n}}_s)) = 0 \tag{4}$$

that gives α as below:

$$\alpha = M_\alpha \begin{bmatrix} \dot{\mathbf{B}} \\ \dot{\underline{\mathbf{x}}} \end{bmatrix} \quad with \quad M_\alpha = \begin{bmatrix} \mathbf{0}_{1\times 3} & -\underline{\mathbf{n}}_s^T \end{bmatrix} \tag{5}$$

Afterwards, the β is computed by differentiating the first constraint in (2) and replacing (3) and (5) into the differentiated first constraint. This yields:

$$\dot{\mathbf{B}}^T\, \underline{\mathbf{n}}_s + \mathbf{B}^T\, (\alpha\, \underline{\mathbf{x}} + \beta\, (\underline{\mathbf{x}} \times \underline{\mathbf{n}}_s)) = 0 \tag{6}$$

which allows to calculate β as follows:

$$\beta = M_\beta \begin{bmatrix} \dot{\mathbf{B}} \\ \dot{\underline{\mathbf{x}}} \end{bmatrix} \quad with \quad M_\beta = \begin{bmatrix} \frac{-\underline{\mathbf{n}}_s^T}{\mathbf{B}^T\, (\underline{\mathbf{x}} \times \underline{\mathbf{n}}_s)} & \frac{\mathbf{B}^T\, \underline{\mathbf{x}}\, \underline{\mathbf{n}}_s^T}{\mathbf{B}^T\, (\underline{\mathbf{x}} \times \underline{\mathbf{n}}_s)} \end{bmatrix} \tag{7}$$

Thus, the differential model between an edge velocity and the end-effector pose velocity appears when (5), (7) and (1) are inserted into (3). This gives:

$$\dot{\underline{\mathbf{n}}}_s = M_s\, \dot{\mathbb{X}} \quad with \quad M_s = (\underline{\mathbf{x}}\, M_\alpha + (\underline{\mathbf{x}} \times \underline{\mathbf{n}}_s)\, M_\beta) \begin{bmatrix} L_B \\ M_x \end{bmatrix} \tag{8}$$

where $s \in \{L, R\}$ denotes for either left or right edge. Finally, the differential model for both of the left and right edges of a cylindrical leg is defined as follows:

$$\dot{\mathbf{n}} = M_n \, \dot{\mathbb{X}} \tag{9}$$

where $\mathbf{n} \in \Re^{6\times 1}$ and $M_n \in \Re^{6\times n}$ are as below:

$$\mathbf{n} = \begin{bmatrix} \underline{\mathbf{n}}_L \\ \underline{\mathbf{n}}_R \end{bmatrix}, \qquad M_n = \begin{bmatrix} M_L \\ M_R \end{bmatrix} \tag{10}$$

III. Virtual Visual Servoing

The non-sequential (simultaneous) acquisition approach gives the complete static pose of a robot straightaway. On the other hand, the sequential acquisition approach takes several (k) successive sub-images to collect the same information. Since the sequential acquisition approach gives the same information with k successive sub-images, one needs previous $k-1$ sub-images to be stored. Thus, the static pose can be computed at each sub-image grabbing instant with previously stored $k-1$ sub-images. Given the sampling time T_{camera}, the beauty of this scenario is that it provides simultaneously both the static pose and velocity. Furthermore, sequential approach is a lot faster than the non-sequential one. Table I compares requirements of the tasks of the sequential and non-sequential approaches for the computation of the static pose and velocity. To keep the comparison simple, a task time is assessed in either a sub-image processing time T_{sub} or a full image processing time T_{full}, where $T_{sub} \ll T_{full}$.

TABLE I

Comparison of sequential and non-sequential approaches.

Task	Sequential	Non – Sequential
Image acquisition	$1\,T_{sub}$	$1\,T_{full}$
Feature extraction	$1\,T_{sub}$	$1\,T_{full}$
Pose computation	$k\,T_{sub}$	$k\,T_{sub}$
Pose and velocity computation	$k\,T_{sub}$	$2k\,T_{sub}$

For example: If a 48×48 pixels sub-image is grabbed rather than a full 1024×1024 pixels image, then in sequential approach: (i) Image acquisition is reduced to about 455 times; (ii) Feature extraction is reduced to about 455 times; (iii) Pose computation time stays same; (iv) Pose and velocity computation time is reduced to about 2 times.

The estimation of the dynamic state of the robot is faster when the information grabbed is smaller. However, when smaller pieces of information are grabbed, less information is kept for the present time and less accuracy is expected. Consequently, an optimum should be determined regarding this tradeoff within theoretical and physical limits.

A. Notation

- $t \in \{t_c, \bar{t}_c\}$ denotes the time, where t_c is an acquisition instant of the camera and $\bar{t}_c$ is an estimation instant for the state variables of the virtual robot.
- $T \in \{T_c, \overline{T}_c\}$ are time periods of sub-image acquisition of the camera and of the update of the virtual robot state variables, respectively. The virtual time period $\overline{T}_c$ should be equal to or greater than the acquisition time period T_c of the camera ($\overline{T}_c \geqslant T_c$) so that the virtual robot can catch the motion of the real robot.
- $j(t) \in \{1, 2, \ldots\}$ is a function of time instants that indicates which cylindrical leg is observed at time t.

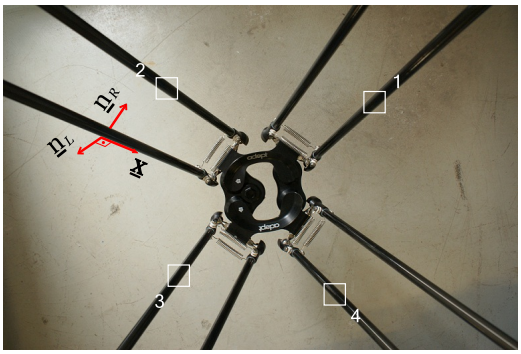

Fig. 3. A full image of lower-legs with their 48×48 pixel^2 sub-images from the base-mounted camera of the Quattro robot. These sub-images are grabbed consecutively at discrete time instants of the motion of the Quattro.

B. Sequential Postures Error

A posture error is formed with a pair of reference projection contour points (in metric units), $\{\mathbf{p}^*_{jL_i}, \mathbf{p}^*_{jR_i}\}$, extracted from the sub-image of a cylindrical leg of the real robot, and their associated edges (feedback signal) computed from the virtual robot's cylindrical leg:

$$\mathbf{e}_{ji} = \begin{bmatrix} \mathbf{p}^{*T}_{jL_i}\, \underline{\mathbf{n}}_{jL} \\ \\ \mathbf{p}^{*T}_{jR_i}\, \underline{\mathbf{n}}_{jR} \end{bmatrix} \tag{11}$$

where $i = 1, \ldots, m$ is the index of a detected contour point. Then, the error vector $\mathbf{e}_j \in \Re^{2m\times 1}$ of the j^{th} leg of the virtual robot is noted for all the contour points as follows:

$$\mathbf{e}_j = C^*_j\, \mathbf{n}_j \tag{12}$$

where $\mathbf{n}$ is as in (10) and $C^*_j \in \Re^{2m\times 6}$ is a constant reference contour matrix:

$$C^*_j = \begin{bmatrix} P^{*T}_{jL} & \mathbf{0} \\ \mathbf{0} & P^{*T}_{jR} \end{bmatrix} \tag{13}$$

with $\{P^*_{jL}, P^*_{jR}\}$ detected left and right side contours of the j^{th} cylindrical leg of the real robot at an instant of time:

$$P^*_{jL} = \begin{bmatrix} \mathbf{p}^*_{jL_1} & \cdots & \mathbf{p}^*_{jL_m} \end{bmatrix} \in \Re^{3\times m} \tag{14}$$

$$P^*_{jR} = \begin{bmatrix} \mathbf{p}^*_{jR_1} & \cdots & \mathbf{p}^*_{jR_m} \end{bmatrix} \in \Re^{3\times m} \tag{15}$$

Finally, having the sets of contour matrices from the real robot $\{\mathbf{C}^*\}$ and their corresponding feedback edge pairs $\{\mathbf{n}\}$ from the virtual robot which are saved at k sequential discrete instants, the complete error vector $\mathbf{e} \in \Re^{2km\times 1}$ is formed by stacking the last k posture errors of the legs:

$$\mathbf{e} = \begin{bmatrix} C^*_{j(t_c)}\, \mathbf{n}_{j(\bar{t}_c)} \\ \vdots \\ C^*_{j(t_c-(k-1)T_c)}\, \mathbf{n}_{j(\bar{t}_c-(k-1)\overline{T}_c)} \end{bmatrix} \tag{16}$$

where $j(\cdot)$ circular-wise enumerates the cylindrical legs at consecutive instants of time. Figure 3 shows an example for the enumeration of the lower-legs of the Quattro robot.

C. Approximated Edge Evolution Model

In order to regulate this time-space error to zero, a differential model should be defined between an edge $\mathbf{n}$ of time $t+\Delta t$ and the effector pose $\mathbb{X}$ of reference time t. To find this model, we first write small displacement of an edge:

$$\mathbf{n}_{t+\Delta t} = \mathbf{n}_t + \delta\mathbf{n}_t \tag{17}$$

where Δt tells how far in time the displaced edge is, and where $\delta\mathbf{n}_t$ is the displacement in the edge values with respect to reference time instant t. One can approximate the displacement $\delta\mathbf{n}_t$ through (9):

$$\delta\mathbf{n}_t \approx M_{n_t}\,\delta\mathbb{X}_t \tag{18}$$

The displacement in the end-effector pose $\delta\mathbb{X}_t$ can be approximated with a constant acceleration model as follows:

$$\delta\mathbb{X}_t \approx \Delta t\,\dot{\mathbb{X}}_t + \frac{1}{2}\Delta t^2\,\ddot{\mathbb{X}}_t \tag{19}$$

If the representation of the end-effector pose $\mathbb{X}_t$ is not an element of a linear vector space, (19) is valid only if the rotational axis of the motion during Δt time remains constant. Otherwise, it will be still acceptable for a small Δt. Finally, the approximated differential model can be expressed using (17), (18) and (19) as below:

$$\dot{\mathbf{n}}_{t+\Delta t} \approx \frac{\mathbf{n}_{t+\Delta t} - \mathbf{n}_t}{\Delta t} = \frac{\delta\mathbf{n}_t}{\Delta t} \tag{20}$$

$$\dot{\mathbf{n}}_{t+\Delta t} \approx H_{(t,\,\Delta t)} \begin{bmatrix} \dot{\mathbb{X}}_t \\ \ddot{\mathbb{X}}_t \end{bmatrix} \tag{21}$$

where $H_{(t,\,\Delta t)} \in \Re^{6\times 2n}$ is the constant acceleration evolution model of the edge pair of a leg:

$$H_{(t,\,\Delta t)} = \begin{bmatrix} M_{n_t} & \frac{1}{2}\Delta t\, M_{n_t} \end{bmatrix} \tag{22}$$

M_{n_t} is defined in (10). This approximation allows to solve the inverse kinematic problem of a parallel robot only once instead of k times, where k is equal to or greater than the number of observed legs.

D. Visual Servoing Control Law

The control law which regulates the error is computed by first differentiating (12) with respect to time, which gives:

$$\dot{\mathbf{e}}_j = C_j^*\,\dot{\mathbf{n}}_j + \dot{C}_j^*\,\mathbf{n}_j \tag{23}$$

then replacing $\dot{\mathbf{n}}_j$ by (21) and imposing $\dot{\mathbf{e}}_j = -\lambda\,\mathbf{e}_j$ for an exponential convergence, (23) becomes:

$$-\lambda\,\mathbf{e}_j = L_{e_j} \begin{bmatrix} \dot{\mathbb{X}}_t \\ \ddot{\mathbb{X}}_t \end{bmatrix} + \dot{C}_j^*\,\mathbf{n}_j \tag{24}$$

where $L_{e_j} \in \Re^{2m\times 2n}$ is so-called interaction matrix which relates the end-effector pose velocity and its derivative to the error function of a leg:

$$L_{e_j} = C_j^*\,H_{j(t,\Delta t)} \tag{25}$$

Then, to converge to the state of the real robot, the control law $\mathbf{u} = \left[\dot{\mathbb{X}}_u^T, \ddot{\mathbb{X}}_u^T\right]^T$ is computed to update the pose and velocity of the virtual robot as follows:

$$\begin{bmatrix} \dot{\mathbb{X}}_u \\ \ddot{\mathbb{X}}_u \end{bmatrix} = -\lambda\,L_e^{\dagger}\,(\mathbf{e}-\tilde{\mathbf{e}}), \qquad \lambda > 0 \tag{26}$$

where $L_e \in \Re^{2km\times 2n}$ and $\tilde{\mathbf{e}} \in \Re^{2km\times 1}$ are as below:

$$L_e = \begin{bmatrix} L_{e_{j(\bar{t}_c)}} \\ \vdots \\ L_{e_{j(\bar{t}_c-(k-1)\overline{T}_c)}} \end{bmatrix}, \quad \tilde{\mathbf{e}} = \begin{bmatrix} \dot{C}^*_{j(t_c)}\,\mathbf{n}_{j(\bar{t}_c)} \\ \vdots \\ \dot{C}^*_{j(t_c-(k-1)T_c)}\,\mathbf{n}_{j(\bar{t}_c-(k-1)\overline{T}_c)} \end{bmatrix} \tag{27}$$

The term $\tilde{\mathbf{e}}$ in (26) is difficult to approximate, since we do not use any correspondence between successively detected reference contour points. Therefore, it is considered as a disturbance. Normally, (26) should not be directly calculated with an ordinary pseudo-inverse least squares regression. This can be unstable and slow. Here, (26) is just a representation of the numerical solution of (24). The linear system in (24) can be solved with damped total least squares and QR decomposition. This yields robust and fast solutions. Thus, one can write (24) as follows:

$$A\,\mathbf{u} = \mathbf{b} \tag{28}$$

where $A \in \Re^{(2km+2n)\times 2n}$ and $\mathbf{b}^{(2km+2n)\times 1}$ are the augmented coefficient matrix and the augmented error vector, respectively:

$$A = \begin{bmatrix} L_e \\ \mu\, I \end{bmatrix}, \qquad \mathbf{b} = \begin{bmatrix} -\lambda\,\mathbf{e} \\ \mathbf{0} \end{bmatrix} \tag{29}$$

with μ damping parameter, I ($2n$ by $2n$) identity matrix, and $\mathbf{0}$ ($2n$ by 1) zero vector. Linear system (28) can be now solved for $\mathbf{u}$ with QR decomposition.

E. Virtual Robot Motion Evolution

Pseudo-update rule for the dynamic state of the virtual robot using the control law (26) can be written as follows:

$$\mathbb{X}_{\bar{t}_c} = \mathbb{X}_{(\bar{t}_c-\overline{T}_c)} + \overline{T}_c\,\dot{\mathbb{X}}_u, \qquad \dot{\mathbb{X}}_{\bar{t}_c} = \dot{\mathbb{X}}_{(\bar{t}_c-\overline{T}_c)} + \overline{T}_c\,\ddot{\mathbb{X}}_u \tag{30}$$

This update rule is valid as long as representation of the end-effector pose forms a vector space. In order to assemble the feedback edges $\mathbf{n}$, the virtual robot's pose is displaced back in time with a *constant velocity motion model*, since the updated dynamic state is up to velocity. Using the new updated state variables and differential kinematic models for the time t_c, the feedback edge set is calculated as follows:

$$\mathbf{n}_{\bar{t}_c+\Delta t} \approx \mathbf{n}_{\bar{t}_c} + \Delta t\,M_{n_{\bar{t}_c}}\,\dot{\mathbb{X}}_{\bar{t}_c} \tag{31}$$

where $\Delta t = i\overline{T}_c$ is the virtual time displacement with an index $i = 0,-1,\ldots,-(k-1)$ counting backwards.

F. Predicting Future Sub-Image Location

The next dynamic state estimation needs a future sub-image to be grabbed at the next future sampling time t_c+T_c. The position of this sub-image on the image plane must be correctly predicted from the current dynamic state $\{\mathbb{X},\dot{\mathbb{X}}\}$ computed for the time instant t_c. Otherwise there will not be any useful signal in the grabbed sub-image and the tracking

will fail. Achieving a correct prediction is, itself, a proof of the correct performance of the proposed method. In order to predict any of the corresponding sub-image positions of the legs, we first find the likely future pose of the real robot:

$$\widehat{\mathbb{X}}_{(t_c+T_c)} = \mathbb{X}_{\bar{t}_c} + \overline{T}_c \dot{\mathbb{X}}_{\bar{t}_c} \tag{32}$$

Once the likely future end-effector pose $\widehat{\mathbb{X}}_{(t_c+T_c)}$ is found, it is solved for the tip point $\widehat{\mathbf{B}}_j$ and for the orientation unit vector $\widehat{\underline{\mathbf{x}}}_j$ of a leg through the inverse kinematic model (IKM). Subsequently, the next location of the corresponding sub-image center is calculated as follows:

$$z_j \begin{bmatrix} {}^{im}\mathbf{w}_j \\ 1 \end{bmatrix} = K(\widehat{\mathbf{B}}_j - d_j \widehat{\underline{\mathbf{x}}}_j) \tag{33}$$

where ${}^{im}\mathbf{w}_j \in \Re^{2\times 1}$ is the next predicted location of a sub-image center in pixel units, d_j is a distance that indicates how far along the cylindrical leg to the leg's tip point the observed region is, z_j is the projective scale factor, and K is the camera intrinsic matrix.

IV. RESULTS

We conducted simulation and experimentation on the Quattro robot (see Fig. 1). The Quattro robot encodes its pose in the contours of its 4 lower-leg rods. Thus, we estimated the dynamic state of the Quattro robot by using at least 4 sub-images grabbed from each of the lower-legs at consecutive discrete instants of the motion. For example, the first estimation used sub-images of the lower-legs $\{1,2,3,4\}$, the second used $\{2,3,4,1\}$ and so on. We set the camera sub-image acquisition frequency to $500\,Hz$. Each sub-image was a 48×48 pixel^2. Figure 3 shows these 4 sub-images on the lower-legs. Furthermore, while even the lower-leg rod being observed is partially out of the field of view of the camera, we can still measure visual contours by keeping the sub-image location in the visible region via parameter d_j and/or by observing the other rod if the leg is a parallelogram type. Figure 4 shows the simple block representation of the estimation algorithm. We did each estimation with a single-iteration virtual visual servoing. We

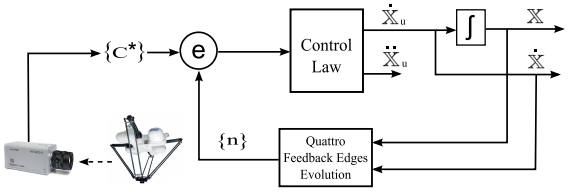

Fig. 4. Single-iteration virtual visual servoing for fast estimation.

noticed that the computation of the velocity control law $\ddot{\mathbb{X}}_u$ was very ill-conditioned. Therefore, we assigned the pose update control law $\dot{\mathbb{X}}_u$ to the pose velocity. We evaluated the performance of the estimated states with root-mean-square of residuals (RMSE). The end-effector of the Quattro robot has 3 translational (xyz) and 1 rotational (θ) degrees of freedom. Thus, we examined accuracies in these two parts.

Experimentation: We coded the estimation algorithm in C++ with nT_2 matrix library [17]. The test path was a $8\,cm$ by $8\,cm$ square motion. The path crosses XY, XZ and YZ planes. The motion was without rotation. The maximum velocity and acceleration of the motion were $25\,cm/s$ and $1\,m/s^2$, respectively. We calibrated the camera such that a non-iterative model compensates for distortions rapidly. Figures 5 and 6 show the reference and estimated Cartesian poses and velocities. Table II lists accuracies of the position and orientation estimations. Table III tabulates the approximate

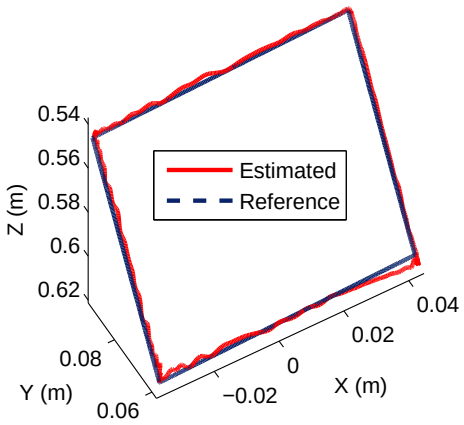

Fig. 5. Reference (blue dotted line) and estimated (red solid line) Cartesian space curves in the camera frame (for the results of Table II).

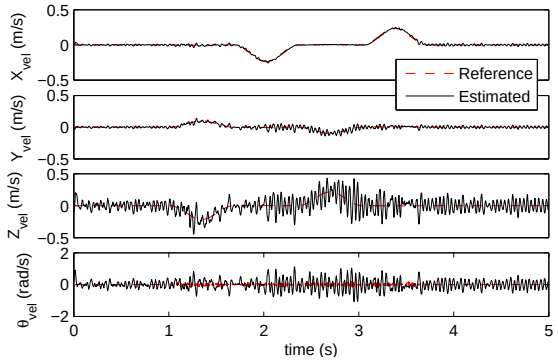

Fig. 6. Superimposed reference (red dotted line) and estimated (black solid line) Cartesian velocities in the camera frame (for the results of Table II).

TABLE II

RMSE OF ESTIMATIONS IN THE EXPERIMENT.

Path	Pose Errors		Velocity Errors	
square	xyz (mm)	θ (deg)	$\dot{xyz}$ (cm/s)	$\dot{\theta}$ (deg/s)
$25\,cm/s,\ 1\,m/s^2$	4	1.5	9.8	17.8

taken times of the processes required to estimate a dynamic state of the Quattro robot using sequential 48×48 pixel^2 sub-images. An estimation costed about $1400\,\mu s$. This means that the posture and velocity of the Quattro can be computed faster than $500\,Hz$.

TABLE III

TAKEN TIMES FOR AN ESTIMATION WITH 48×48 SUB-IMAGES.

ROI exposure	ROI transfer	Edge detection	Estimation
500 μs	100 μs	200 μs	600 μs

Simulations: We also validated the proposed estimation algorithm in Matlab by simulations. The test path was the same square motion as in the experimentation part. We also repeated this square motion with 3 different maximum velocity and acceleration values: $(60cm/s,\ 1G)$, $(1.2m/s,\ 5G)$, and $(1.7m/s,\ 10G)$. In order to test the robustness, we generated the following moderate noises: (i) we deflected the camera orientation with 1° degree around an arbitrary axis; (ii) we displaced the camera position $5mm$ away along an arbitrary direction; (iii) we perturbed the contours [-1,+1] pixel orthogonally and uniformly. Tables IV and V list the accuracies of the position and orientation estimations without and with noise, respectively.

TABLE IV

RMSE OF ESTIMATIONS IN THE SIMULATIONS (WITHOUT NOISE).

Path	Pose Errors		Velocity Errors	
square	$xyz\ (mm)$	$\theta\ (deg)$	$\dot{xyz}\ (cm/s)$	$\dot{\theta}\ (deg/s)$
$25cm/s,\ 1m/s^2$	0.4	0.02	0.27	2.7
$60cm/s,\ 1G$	1.5	0.07	3.3	5.8
$1.2m/s,\ 5G$	2.6	0.12	8.7	8.7
$1.7m/s,\ 10G$	3.4	0.16	16	16.3

TABLE V

RMSE OF ESTIMATIONS IN THE SIMULATIONS (WITH NOISE).

Path	Pose Errors		Velocity Errors	
square	$xyz\ (mm)$	$\theta\ (deg)$	$\dot{xyz}\ (cm/s)$	$\dot{\theta}\ (deg/s)$
$25cm/s,\ 1m/s^2$	3.4	3.03	9.8	21.04
$60cm/s,\ 1G$	3.9	3.03	10.3	29.12
$1.2m/s,\ 5G$	4.4	2.98	12.8	34.26
$1.7m/s,\ 10G$	5	2.95	20.3	36.20

Discussion: In Table IV, we see that even if there is no noise, the errors increase. This is because the motion model used for tracking does not take into account acceleration. If we look at the first test results of the Table V, we can see that these results match with the experiment. In Table V, we also see that the errors remain at the same order of magnitude and they come from the calibration errors. So, we can expect that there will not be performance degradation in experiments where the robot moves on test paths with faster motion. Moreover, experimentation revealed that the depth (i.e., along optical axis of the camera) and consequently the orientation estimations are very sensitive to noise (see Fig. 6). These errors in depth and orientation might appear because of a cylindrical leg whose radius is relatively smaller than its observational distance from the camera. This makes estimations, especially the depth, more sensitive to small noises. We can eliminate this sensitivity to noise by placing multiple cameras (e.g., a camera per leg) in different viewpoints. This also allows us to explore the entire robot workspace. Yet, we can improve these estimations in a dynamic control scenario of a robot by exploiting the computed-torque control law as a feed-forward acceleration term for better update of state estimations [10]. Generally, the source of errors comes from: (i) the use of approximated theoretical models; (ii) the calibration of camera extrinsic parameters; (iii) the image noise while detecting visual contours.

V. CONCLUSIONS

This paper has shown that we can estimate the postures and velocities of all parts of a parallel robot rapidly with reasonable accuracy through sequential observation of the leg contours. This method does not need any artificial pattern since the legs are observed, and is applicable for any slim prism-shaped legs. It is feasible by an edge detection in a small and structured sub-image. The sub-image contains only a portion of the slim leg. Some future perspectives to increase the accuracy of estimation are as follows: (i) to investigate the grabbing strategy with different number, size and order of the sub-images; (ii) to explore the number of rods (4 or 8) and the number of sub-images on each rod.

Finally, we conclude that this work will guide us to control the dynamics of parallel robots from their leg observations.

REFERENCES

[1] F. Chaumette, S. Hutchinson, "Visual servo control, part II: Advanced approaches", *IEEE Robotics and Automation Magazine*, 2007.

[2] J. Gangloff, M. de Mathelin, "High-speed visual servoing of a 6 DOF manipulator using multivariate predictive control", *Advanced Robotics. Special Issue: advanced 3D vision and its application to robotics.*, pp. 993-1021, 2003.

[3] M. Vincze, "Dynamics and system performance of visual servoing", *IEEE Int. Conf. on Robotics and Automation, (ICRA'00)*, USA, 2000.

[4] P. I. Corke, "Dynamic issues in robot visual-servo systems", *Int. Symp. on Robotics Research, ISSR'95*, pp. 488-498, Springer, 1995.

[5] I. Richardson, "H.264 and MPEG-4 Video Compression: Video Coding for Next-generation Multimedia", J. Wiley, 2003.

[6] Wilson L.W, Hulls C.W, Bell G. S, "Relative end-effector control using cartesian position based visual servoing", *IEEE Transactions on Robotics and Automation*, pp. 684-696, October, 1996.

[7] Y. Nakabo, M. Ishikawa, H. Toyoda, S. Mizuno, "1ms column parallel vision system and its application of high speed target tracking", *IEEE Int. Conf. on Robotics and Automation, (ICRA'00)*, USA, 2000.

[8] P. Chalimbaud, F. Berry, "Embedded active vision system based on FGPA achitecture.", *In EURASIP Journal on Embedded Systems*, 2007.

[9] F. Paccot, N. Andreff, P. Martinet, "A review on dynamic control of parallel kinematic machines: theory and experiments", *International Journal of Robotics Research, (IJRR'09)*, pp. 395-416, 2009.

[10] R. Dahmouche, N. Andreff, Y. Mezouar, O. Ait-Aider, P. Martinet "Dynamic visual servoing from sequential regions of interest acquisition", *International Journal of Robotics Research, (IJRR'12)*, vol. 31, no. 4, pp. 520-537, 2012.

[11] N. Andreff, T. Dallej, P. Martinet, "Image-based visual servoing of a Gough-Stewart parallel manipulator using leg observations", *International Journal of Robotics Research. Special Issue on Vision and Robotics Joint with the International Journal on Computer Vision, (IJRR'07)*, vol. 26, no. 7, pp. 677-687, 2007.

[12] E. Özgür, N. Bouton, N. Andreff, P. Martinet "Dynamic Control of the Quattro Robot by the Leg Edges", *IEEE Int. Conf. on Robotics and Automation, (ICRA'11)*, China, 2011.

[13] O. Ait-Aider, N. Andreff, J.M. Lavest, P. Martinet, "Simultaneous object pose and velocity computation using a single view from a rolling shutter camera", *The 9th European Conference on Computer Vision, (ECCV'06)*, pp. 56-68, Austria, 2006.

[14] E. Marchand, F. Chaumette, "Virtual visual servoing: A framework for real-time augmented reality", *EUROGRAPHICS 2002 Conference Proceeding*, pp. 289-298, Germany, 2002.

[15] R. Dahmouche, N. Andreff, Y. Mezouar, P. Martinet "3D Pose and Velocity Visual Tracking Based on Sequential Region of Interest Acquisition", *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, (IROS'09)*, USA, 2009.

[16] N. Andreff, P. Martinet, "Unifying Kinematic Modeling, Identification, and Control of a GoughStewart Parallel Robot Into a Vision-Based Framework", *IEEE Transactions on Robotics, vol. 22, no. 6, pp. 1077-1086*, 2006.

[17] J. Falcou, J. Lapresté, T. Chateau, J. Sérot, "NT2: A High-performance library for computer vision", 2007.