



Article A Novel Velocity-Based Control in a Sensor Space for Parallel Manipulators

Antonio Loredo¹, Mauro Maya^{1,*}, Alejandro González², Antonio Cardenas¹, Emilio Gonzalez-Galvan¹ and Davide Piovesan³

- ¹ Facultad de Ingeniería, Universidad Autónoma de San Luis Potosí, San Luis Potosi 78290, Mexico
- ² Tecnologico de Monterrey, Escuela de Ingenieria y Ciencias, Queretaro 76130, Mexico
- ³ Biomedical Engineering, Gannon University, Erie, PA 16541, USA
- Correspondence: mauro.maya@uaslp.mx

Abstract: It is a challenging task to track objects moving along an unknown trajectory. Conventional model-based controllers require detailed knowledge of a robot's kinematics and the target's trajectory. Tracking precision heavily relies on kinematics to infer the trajectory. Control implementation in parallel robots is especially difficult due to their complex kinematics. Vision-based controllers are robust to uncertainties of a robot's kinematic model since they can correct end-point trajectories as error estimates become available. Robustness is guaranteed by taking the vision sensor's model into account when designing the control law. All camera space manipulation (CSM) models in the literature are position-based, where the mapping between the end effector position in the Cartesian space and sensor space is established. Such models are not appropriate for tracking moving targets because the relationship between the target and the end effector is a fixed point. The present work builds upon the literature by presenting a novel CSM velocity-based control that establishes a relationship between a movable trajectory and the end effector position. Its efficacy is shown on a Delta-type parallel robot. Three types of experiments were performed: (a) static tracking (average error of 1.09 mm); (b) constant speed linear trajectory tracking—speeds of 7, 9.5, and 12 cm/s-(tracking errors of 8.89, 11.76, and 18.65 mm, respectively); (c) freehand trajectory tracking (max tracking errors of 11.79 mm during motion and max static positioning errors of 1.44 mm once the object stopped). The resulting control cycle time was 48 ms. The results obtained show a reduction in the tracking errors for this robot with respect to previously published control strategies.

Keywords: vision-based control; camera-space manipulation; parallel robot

1. Introduction

Parallel robots are composed of mobile (end-effector) and fixed bases, which are linked together by at least two independent kinematic chains with *n* degrees of freedom [1]. By having more than one kinematic chain, parallel robots have some advantages over their serial counterparts, i.e., higher velocity and acceleration, improved rigidity, higher accuracy, better load-capacity-to-weight ratio, low inertia, good stability, and simple inverse kinematics. On the other hand, their main disadvantages are: reduced workspaces, complex kinematic models [2,3], limited workspaces, and multiple singular configurations in their reachable workspaces, resulting in limited maneuverability and low dexterity [4]. Traditional control strategies are hard to implement on parallel robots, due to: the robot's multiple solutions to the direct kinematic problem and modeling errors in the Jacobian as a result of inaccurate geometric calibration. Research on parallel robots has focused on mitigating the effects of these drawbacks on their precision and ease of control [5–8].

Cameras are often used as effective robotic sensors, as they allow for non-contact exteroceptive sensing. They can help to compensate for errors in the robot's kinematic model in positioning tasks as they provide measurements of the end-effector and the target.



Citation: Loredo, A.; Maya, M.; González, A.; Cardenas, A.; Gonzalez-Galvan, E.; Piovesan, D. A Novel Velocity-Based Control in a Sensor Space for Parallel Manipulators. *Sensors* **2022**, *22*, 7323. https://doi.org/10.3390/s22197323

Academic Editor: Jesús Ureña

Received: 31 August 2022 Accepted: 20 September 2022 Published: 27 September 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https:// creativecommons.org/licenses/by/ 4.0/). A control law based on these measurements can ensure precise positioning of the endeffector on the target despite uncertainties in the robot model. Visual Servoing (VS) and Camera Space Manipulation (CSM) are two of the most popular vision control approaches used in robotics. Vision-based control can be regarded as a sensor-based control strategy, which strongly resembles the process of our central nervous system [9].

VS control is used for real-time control, capable of integrating visual information obtained by one or more cameras in a closed-loop control cycle. Usually, VS algorithms are kinematic control schemes that do not include the stability analysis of closed-loop dynamics [10]. Its development allows for increased precision in manipulation tasks based on visual feedback [11–13]. VS can be classified based on the error signal it handles as: 3D (it is also known as "position-based" but this term is reserved in this work to describe control approaches where the control signal is the joint position), where the error is expressed in Cartesian coordinates, 2D (also called image-based, where the error is expressed in image coordinates), and some variants (mixing coordinates). With respect to the control signal it produces, VS can be classified as position-based in the sense that the control signal is the joint position (most 3D schemes are of this type) or velocity-based, in the sense that the control signal is the joint velocity (usually 2D schemes are of this type). Velocity-based controllers require the calculation of the robot Jacobian, and if using a 2D approach, it will also require what is known as the image Jacobian, i.e., the analytical transformation between the operational tangent space and the image tangent space. Image Jacobians depend on the visual features used in the approach and the camera parameters (a calibration process is required to obtain them) and can be complex and difficult to obtain [14]. Some work has also been done designing controllers robust to uncertainties in the Jacobian matrix as reported in [15].

On the other hand, CSM [16–18] and its variants, such as Linear Camera Model-Camera Space Manipulation (LCM-CSM) [19,20], are calibration-free and do not require the computation of the robot's Jacobian since they are position-based. They only require the (uncalibrated) robot's kinematic model and focus on controlling the end-effector position, refining its precision iteratively at each movement iteration. These approaches do not need the image Jacobian because they do not involve the velocities in the operational space or the image space since it is designed for static positioning of the robot using position commands. Previous attempts at tracking trajectories using vision-based control have used a dual-stage approach where velocity prediction was obtained in a first assessment stage, followed by a position control strategy trying to predict such velocity [21]. Tracking and movement using CSM require continuous updating of the view parameters, which can be performed non-linearly using a heuristic algorithm, such as a particle filter [22], or by local linearization conducive to the use of statistical estimators, such as the Kalman filter [18]. In [23], the authors presented a LCM-CSM variant where the tracking of moving objects was possible using positioning commands; however, no CSM velocity-based variant has been developed to this day. Velocity-based control could prove to be more energy efficient, suitable for low control frequencies, and robust to variations on the control signal [24,25].

Recent works in vision-based control have sought to enhance CSM with artificial intelligence (A.I.) techniques. The work reported in [26] combined the CSM method and a differential evolution technique to increase the precision of robot maneuvers; the authors of [27,28] used the CSM method and support the vector regression technique in robotics tasks to catch moving objects.

Vision-based systems with A.I. control strategies are very active research areas; examples are found in References [29–31]. In a recent study [32], the author presented a comprehensive survey on such topics. A drawback of these methods is the amount of data required to train the algorithm used for the tracking task. We opted for a more classical approach where the velocity of the target was estimated via a Kalman filter.

This work presents a new vision-based velocity control, inspired by LCM-CSM, meant to command the joint velocities of a robot to perform a task. This is done in a closed-loop control scheme by using the LCM-CSM vision parameters and the robot's Jacobian. The

contribution of the work presented herein involves the use of vision parameters to map the image space velocity vector to the end-effector velocity. This approach incorporates the benefits of velocity-based control while keeping some of the advantages of the traditional LCM-CSM, as presented in this paper.

This article is structured as follows: Section 2 presents a review of the kinematic model of Delta robots. Section 3 introduces the traditional CSM and LCM-CSM techniques while Section 4 describes the proposed modifications necessary to enable velocity control. Section 5 describes the experimental platform, and Section 5.4 details the experiments performed. Section 6 presents the results, and offers a brief discussion of them. Finally, Section 7 presents our conclusions and shows possible venues for future work.

2. Delta-Type Parallel Robots

Figure 1 shows the PARALLIX LKF-2040, an academic Delta-type robot composed of a mobile platform attached to a fixed base by three closed kinematic chains. Each chain has a parallelogram guaranteeing parallelism between the platforms [33]. PARALLIX LKF-2040 is capable of imparting three translational degrees of freedom to its end-effector. Figure 1 also shows the robot's geometric parameters required for its kinematic model and is described in [34]. These are: the radii of the fixed (*R*) and mobile platform (*r*), the lengths of the actuated links (*L*₁), the parallelogram (*L*₂), the constant angle of each arm with respect to the robot's coordinate system (α_i), the displacement angles of the active joints (θ_{i1}), and the passive joints (θ_{i2} and θ_{i3}).



Figure 1. The PARALLIX LKF-2040, an academic Delta-type robot.

2.1. Forward and Inverse Kinematic Model of the Delta Robot

The aforementioned geometric parameters define the configuration of each kinematic chain. With this information, it is possible to compute the robot's implicit forward kinematic model shown in (1) as well as its inverse kinematic model given by (3) [35] as follows:

$$(X_i - X_p)^2 + (Y_i - Y_p)^2 + (Z_i - Z_p)^2 = L_2^2$$
(1)

where:

$$X_{i} = (R + L_{1} \cos \theta_{i1} - r) \cos \alpha_{i}$$

$$Y_{i} = (R + L_{1} \cos \theta_{i1} - r) \sin \alpha_{i}$$

$$Z_{i} = -L_{1} \sin \theta_{i1}$$
(2)

and (X_p, Y_p, Z_p) denote the position of the center of the mobile platform expressed with respect to a frame attached to the center of the fixed platform.

$$\tan\frac{\theta_{i1}}{2} = \frac{-2Z_p \pm \sqrt{4Z_p^2 + 4R_1^2 - S^2 + Q_i^2 \left(1 - \frac{R_1^2}{L_1^2}\right) + Q_i \left(\frac{-2R_1S}{L_1} - R_1\right)}}{-2R_1 - Q_i \left(\frac{R_1}{L_1} - 1\right) - S}$$
(3)

with:

$$Q_{i} = 2X_{p} \cos \alpha_{i} + 2Y_{p} \sin \alpha_{i}$$

$$R_{1} = R - r$$

$$S = \frac{1}{L_{1}} \left(-X_{p}^{2} - Y_{p}^{2} - Z_{p}^{2} + L_{2}^{2} - L_{1}^{2} - R_{1}^{2} \right)$$
(4)

2.2. Computation of the Delta Robot's Jacobian

The Jacobian matrix provides a transformation between the end effector velocity vector (\dot{x}) in the Cartesian space, and the joint velocity vector (\dot{q}) . The Jacobian matrix of the Delta robot is given by [36] as:

$$\dot{\mathbf{q}} = \mathbf{J}_q^{-1} \mathbf{J}_x \dot{\mathbf{x}} \tag{5}$$

where J_x and J_q are the [3 × 3] matrices obtained from the derivation of the kinematic model. Their elements are given by:

$$\mathbf{J}_{x} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$
(6)

and

$$\mathbf{J}_q = diag(b_{11}, b_{22}, b_{33}) \tag{7}$$

where:

$$a_{i1} = \sin(\theta_{i3}) \cos(\theta_{i1} + \theta_{i2}) \cos(\alpha_i) + \cos(\theta_{i3}) \sin(\alpha_i) a_{i2} = \sin(\theta_{i3}) \cos(\theta_{i1} + \theta_{i2}) \sin(\alpha_i) - \cos(\theta_{i3}) \cos(\alpha_i) a_{i3} = -\sin(\theta_{i3}) \sin(\theta_{i1} + \theta_{i2}) b_{ii} = L_1 \sin(\theta_{i2}) \sin(\theta_{i3})$$
(8)

It should be mentioned that both J_x and J_q are dependent on the robot's current configurations (b_{ii} , a_{i1} , a_{i2} , and a_{i3}). That is, their numerical values depend on the instantaneous end-effector positions.

3. Camera-Space Manipulation with a Linear Camera Model

The LCM-CSM technique works by mapping visual markers from the operational space to the camera space. In an initial, offline stage, markers positioned in the robot's end effector are used. The corresponding three-dimensional positions of the markers are obtained by using the robot's forward kinematic model and, together with their correspondences in the camera space, are used to estimate the parameters of the mapping. If the robot has to perform a positioning task, such a mapping has to be determined for each of at least two participating cameras [19].

The LCM-CSM technique is a variant of the classic CSM method, which replaces the orthographic camera model with the pin-hole model [19]. LCM-CSM defines the so-called *'vision parameters'* by solving a set of linear equations.

In general, a relationship between points in three-dimensional spaces and their planar projections [37] in sensor spaces, can be written as,

$$\rho \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & 1 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ z_i \\ 1 \end{bmatrix} \tag{9}$$

where the center of the *i*th visual marker, located on the end-effector, is represented by the coordinates (u_i, v_i) expressed in pixels, while ρ is a scaling factor, which, under the model's assumptions, tends to a value of one. Cameras are placed far enough to neglect the effect of the camera perspective . For the work presented herein, the robot's workspace was small enough that it could be mostly sampled during the execution of a pre-planned trajectory (also known as "pre-plan") in which samples of the marker's three-dimensional locations and the corresponding camera-space locations were obtained. A requirement of such a trajectory is that samples encompass a large region of the manipulator's workspace. The matrix $\begin{bmatrix} p_{11} & p_{12} & p_{13} & \dots & p_{33} & 1 \end{bmatrix}$ encompasses the vision parameters and the vector $\begin{bmatrix} x_i & y_i & z_i \end{bmatrix}^T$ describes the three-dimensional position of the *i*th visual marker, with respect to the global reference frame.

A linear system for the determination of the camera parameters was obtained from the previous expression, as follows:

$$\mathbf{Y}_{\mathbf{i}} = \mathbf{A}_{\mathbf{i}} \, \mathbf{P} \tag{10}$$

where

1

$$\mathbf{Y}_{\mathbf{i}} = \begin{bmatrix} u_i & v_i \end{bmatrix}^1 \tag{11}$$

$$\mathbf{A}_{\mathbf{i}} = \begin{bmatrix} x_i & y_i & z_i & 1 & 0 & 0 & 0 & 0 & -u_i x_i & -u_i y_i & -u_i z_i \\ 0 & 0 & 0 & 0 & x_i & y_i & z_i & 1 & -v_i x_i & -v_i y_i & -v_i z_i \end{bmatrix}$$
(12)

$$\mathbf{P} = \begin{bmatrix} p_{11} & p_{12} & \cdots & p_{32} & p_{33} \end{bmatrix}^{T}$$
(13)

In order to obtain vector **P**, at least 6 linearly-independent samples are required (i = 1, 2, ..., 6). If more than 6 points are used in the computation of **P**, an over-constrained system is obtained and a minimization process is required. The optimal value of **P** can be computed using least squares optimization:

$$\hat{\mathbf{P}} = \left(\mathbf{A}^T \mathbf{A}\right)^{-1} \mathbf{A}^T \mathbf{Y}$$
(14)

where $\mathbf{Y} = [u_1 v_1 \dots u_n v_n]^T$ are the measured or observed values, $\mathbf{A} = [\mathbf{A_1}^T \dots \mathbf{A_n}^T]^T$ and $n \ge 6$ is the number of points used. Typically, the view parameters are first (roughly) estimated using the set of (global) samples from the "pre-plan". It is immediate to see that the regression between the measurements in the camera space and the estimate of the markers in the operational space allows bypassing the need for a precise image Jacobian, which is *de facto* incorporated in the uncertainty of the view parameters. Then, as the robot approaches the (static) target, new (local) samples are obtained in the neighborhood of the final position and are incorporated in (14) to refine the parameters compensating for the uncertainties of the models. Usually, these local samples are given more significance by assigning them a weight (scalar factor) greater than one; each new sample has a higher weight.

Varying Weights

A problem that needs to be addressed, to enable an accurate estimation of the target point, is the updating of the view parameters when the target is in motion, as reported in [23]. Indeed, when the target is moving, the new samples are dispersed along the trajectory rather than accumulating near the target and are not as effective to refine the parameters.

In order to solve this problem, in the proposed approach, the set of samples obtained in the "pre-plan" was used throughout the experiment. A relative weight was applied to each sample and larger weights were given to samples located closer to the moving target in each image frame. Note that no new samples were required to refine the view parameters, thus the corresponding image acquisition and processing time were saved, yielding a lower computational cost compared to the usual approach. A drawback of this approach involves guaranteeing a sufficiently high number of pre-plan samples to cover the workspace appropriately. The weight given to each sample can be defined using a diagonal matrix W as,

$$\mathbf{W} = \begin{bmatrix} w_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & w_m \end{bmatrix}$$
(15)

where w_i , refers to the *i*th "pre-plan" sample. If **W** is used to estimate **P**, the following equation can be used for the estimation of the optimal parameters, with different weights associated with each sample,

$$\hat{\mathbf{P}} = \left(\mathbf{A}^T \mathbf{W} \mathbf{A}\right)^{-1} \mathbf{A}^T \mathbf{W} \mathbf{Y}$$
(16)

In the work presented herein, the following equation is proposed for the weight given to samples used to determine the camera parameters,

$$w_i = \frac{k}{l_i} \tag{17}$$

where, according to [38], such a definition enables a consistent positioning error. The parameter k is defined for the particular maneuver while l_i represents the Euclidean distance between each sample acquired during the pre-planned trajectory, and the location of the maneuvering objective.

4. CSM-Based Velocity Control

Control Law

Consider (9) for a point attached to the robot's end-effector; by taking its time derivative, a mapping relating the three-dimensional velocity (mm/s) of the end-effector and its associated velocity in the image plane (pixels/s) was obtained:

$$\begin{bmatrix} \dot{u} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \end{bmatrix} \begin{bmatrix} x \\ \dot{y} \\ \dot{z} \end{bmatrix}$$
(18)

г. ¬

Note that mapping (18) is not injective. By adding at least another camera, the injective mapping (19) could be established.

$$\begin{bmatrix} \dot{u}^{1} \\ \dot{v}^{1} \\ \dot{u}^{2} \\ \dot{v}^{2} \end{bmatrix} = \begin{bmatrix} p_{11}^{1} & p_{12}^{1} & p_{13}^{1} \\ p_{21}^{1} & p_{22}^{1} & p_{23}^{2} \\ p_{11}^{2} & p_{12}^{2} & p_{13}^{2} \\ p_{21}^{2} & p_{22}^{2} & p_{23}^{2} \end{bmatrix} \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix}$$
(19)

where the superindex identifies the corresponding camera. This procedure can be expanded to include any additional number of cameras, but for the sake of simplicity, the deduction was performed using only two cameras. Equation (19) can be written in a compact form,

$$\dot{\mathbf{s}} = \mathbf{J}_{csm} \dot{\mathbf{x}} \tag{20}$$

where $\dot{\mathbf{s}}$ is the time derivative of $\mathbf{s} = \begin{bmatrix} u^1 & v^1 & u^2 & v^2 \end{bmatrix}^T$, $\dot{\mathbf{x}}$ is the time derivative of $\mathbf{x} = \begin{bmatrix} x & y & z \end{bmatrix}^T$, and \mathbf{J}_{csm} is the *velocity CSM matrix*, defined as:

$$\mathbf{J}_{csm} = \begin{bmatrix} p_{11}^1 & p_{12}^1 & p_{13}^1 \\ p_{21}^1 & p_{22}^1 & p_{23}^2 \\ p_{21}^2 & p_{22}^2 & p_{23}^2 \\ p_{21}^2 & p_{22}^2 & p_{23}^2 \end{bmatrix}$$
(21)

For a given positioning task, the error signal in the camera space is given by:

$$\mathbf{e} = \mathbf{s}^* - \mathbf{s} \tag{22}$$

where $\mathbf{s}^* = \begin{bmatrix} u^{1*} & v^{1*} & u^{2*} & v^{2*} \end{bmatrix}^T$ is the target point and \mathbf{s} (as defined before) corresponds to a point attached to the end-effector. Both points are expressed by their pixel coordinates on each participating camera.

For a positioning task, by taking the time derivative of (22) and considering the velocity CSM matrix (21) the evolution of the error can be written as:

$$\dot{\mathbf{e}} = \dot{\mathbf{s}}^* - \mathbf{J}_{csm} \dot{\mathbf{x}} \tag{23}$$

By using (5), the previous equation can be rewritten as:

$$\dot{\mathbf{e}} = \dot{\mathbf{s}}^* - \mathbf{J}_{csm} \mathbf{J}_x^{-1} \mathbf{J}_q \dot{\mathbf{q}}$$
(24)

The closed-loop behavior of the error is proposed ideally as $\dot{\mathbf{e}} = -\mathbf{K}\mathbf{e}$ where **K** is a control gain matrix. Note that the system is such that if **K** is defined as a positive definite matrix, the error vector will exponentially decrease to zero. By substitution in (24), we find that:

$$-\mathbf{K}\mathbf{e} = \dot{\mathbf{s}}^* - \mathbf{J}_{csm}\mathbf{J}_x^{-1}\mathbf{J}_q\dot{\mathbf{q}}$$
(25)

Solving for \dot{q} and using (22), the following control law is synthesized:

$$\dot{\mathbf{q}} = \mathbf{J}_q^{-1} \mathbf{J}_x \mathbf{J}_{csm}^{\dagger} [\mathbf{K} (\mathbf{s}^* - \mathbf{s}) + \dot{\mathbf{s}}^*]$$
(26)

where J_{csm}^{\dagger} is the Moore–Penrose pseudoinverse of (21) (alternatively, the last row of J_{csm} can be eliminated, since it provides redundant information in the image space, yielding a square, invertible matrix and J_{csm}^{-1} can be used instead). In (26), \mathbf{s}^* is obtained by placing a marker in the target object and measuring its projection in the camera space, \mathbf{K} is user defined, $\dot{\mathbf{s}}^*$ is not known in the general case and cannot be directly measured (but could be estimated from measurements), and the rest of the elements are known as described before. Figure 2 shows the block diagram of the closed-loop control law.



Figure 2. Block diagram of the proposed control scheme.

While tracking moving objects, if \dot{s}^* is not compensated (i.e., it is set to zero in the control law), tracking errors manifest as drag errors where the robot's end-effector follows the moving object at a certain distance. However, as mentioned before, it is possible to obtain an estimate of the object's camera space velocity through other means, for example, a Kalman filter (KF) as in [23]. The present work does not focus on this estimation; therefore, the implementation of [23] with previously tuned covariance matrices is used for this purpose in this control approach. The key elements of this Kalman filter targeting velocity estimation in camera space are presented in the next subsection, more details can be found in the above reference.

Please refer to Appendix A for more details regarding the computation of a Kalman filter.

5. Materials and Methods

5.1. Hardware

The developed control law was implemented on the PARALLIX LKF-2040, a 3 DOF Delta-type parallel robot (Figure 3), equipped with a variable speed conveyor belt as shown in Figure 4. A vision system consisting of two uEye UI-2210SE-M-GL USB digital cameras with a resolution of 640×480 pixels was used to provide the images for the controller. The cameras were placed 1.5 m away from the center of the robot's workspace and at a distance of 0.9 m from each other. Figure 3 shows the positions of the cameras with respect to the Delta PARALLIX LF-2040. The controller was implemented using an Intel Core i5-6600HQ, 2.5 GHz, 64-bit processor and 8 GB RAM computer running Windows 10.



Figure 3. Experimental setup. (**a**) Delta PARALLIX LF-2040 and camera setup. (**b**) Delta robot and camera position scheme.



Figure 4. Conveyor belt.

5.2. Software

The control system was developed in C++ using Visual Studio 2012 Express. Additionally, matrix operations implemented were conducted using the GNU-GSL libraries [39]; image acquisition and processing were performed using OpenCV [40]. Finally, the control system was implemented using multiple threads in order to decrease its cycle time. The main thread was devoted to the control law calculations, while secondary threads were used for image acquisition and processing. The resulting control cycle time was 48 ms.

5.3. Position Measuring System

In order to measure the 3D tracking error during the tasks, the following was used.

For static trajectories, a vernier caliper was used to measure the 3D positioning error. For moving trajectories, a vision-based measuring system (VMS) was implemented. The measuring system relies on the varying weights LCM-CSM algorithm (9)–(17). Two participating cameras (9) can be written as

$$\begin{bmatrix} u^{1} \\ v^{1} \\ u^{2} \\ v^{2} \\ v^{2} \end{bmatrix} = \begin{bmatrix} p_{11}^{1} & p_{12}^{1} & p_{13}^{1} \\ p_{21}^{1} & p_{22}^{1} & p_{23}^{1} \\ p_{21}^{2} & p_{22}^{2} & p_{23}^{2} \\ p_{21}^{2} & p_{22}^{2} & p_{23}^{2} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$
(27)

Once the corresponding vision parameters are obtained, it is possible to recover the 3D position of an observed image feature point.

The measuring system was calibrated using two visual markers placed on the end effector and separated 25 mm from each other. The markers were moved along a "pre-plan" trajectory sampling the robot's workspace. On each selected position along the pre-plan, the 3D position of each visual marker was obtained as well as the distance between them. An average error of 0.37833 mm with a standard deviation of 0.26058 mm was obtained for this vision-based measuring system. It is noteworthy that the precision of this system can be improved by using higher resolution images and/or more cameras. On the other hand, the system can be used to measure the tracking error while the target moves along an arbitrary trajectory.

A root mean squared error was obtained with the instantaneous measurements on each cycle of the system. Only the second half of the total iterations in each task were used to calculate this error as it was intended to obtain the steady-state error of the task.

5.4. System Operation

The cameras' vision parameters were estimated (see Section 3) before performing the positioning tasks. A total of 410 visual markers were used to initialize the vision parameters defined in (10), and their known three-dimensional and camera space positions were stored in memory. The markers were distributed evenly along the robot's workspace, taking care to span the available volume. In this way, the vision parameter could be determined using a weight-based scheme as a function of the position of the point of interest. This improved the performance of the LCM-CSM method. Under this scheme, the visual marker positions, which were closer to the point of interest, were deemed more important and, thus, carried a larger weight during the estimation of the vision parameters [38]. That is, for the results shown here, the vision parameters were estimated anew each time the point of interest changed position either in three-dimensional or the camera space.

The proposed control scheme was tested using a series of positioning tasks. Three different sets of experiments were performed:

- 1. The first set of experiments consisted of a series of static positioning tasks, i.e., using the proposed control, the robot tracked a static target placed randomly inside the robot's workspace. Ten tasks were executed, each repeated 3 times, for a total of 30 trials. Once the robot's end-effector was within 2 mm of the target position, the robot held its position for 50 control cycles, and the task's mean squared error was computed. During these experiments, the control gain matrix (**K**) was chosen of the form $\mathbf{K} = kI$ where *I* is the 3 × 3 identity matrix and k = 2.3; this value was tuned heuristically.
- 2. For the second set of experiments, the target was placed on a conveyor belt (approximately in the middle of the belt, width-wise) running at a constant speed. That is, the target moved following a linear trajectory referred to as "constant speed linear trajectory". Three different speeds were used: 7, 9.5, and 12 cm/s.

Additionally, three different control gain matrices (**K**) were tested, of the form $\mathbf{K} = kI$, where k = 2.3, 2.7, 3.1. The value of k was chosen as large as possible while maintaining no osculations on the task's positioning response.

For each speed, the task was performed 10 times with each of the possible **K** matrices. Finally, each task was also carried out under two conditions regarding the target's velocity (\dot{s}^*) compensation in the control law; (1) an estimate obtained by means of a Kalman filter was used, and (2) no estimation was used (the compensation was set to zero) yielding a simpler implementation but producing a larger error. However, this error can be reduced by increasing the control gain. In each case, the robot's tracking error was measured.

3. For the last set of experiments, the target was moved freehand along different trajectories inside the robot's workspace. This experiment was referred to as "freehanded trajectory". These trajectories were: a circle, a square, an eight shape, a lemniscate, a zig-zag, and a decreasing spiral. The control gain matrix (**K**) was chosen to be a diagonal matrix with a value of 2.7 on its non-zero terms.

For each positioning task, the following steps were developed:

- 1. We obtained the coordinates of the objective point in pixels (*s**) and loaded the vision parameters previously estimated during the "pre-plan".
- 2. The program entered the control cycle, setting a convergence criterion of error in each camera coordinate (u_i, v_i) less than 1 pixel.
- 3. The centroid of the visual marker attached to the robot's end-effector (s^*) was obtained and the error was computed (22).
- 4. We read the robot's encoders and found $\begin{bmatrix} \theta_{i1} & \theta_{i2} & \theta_{i3} \end{bmatrix}$ to later calculate the Jacobian, as shown in (6) and (7).
- 5. We performed (19) to obtain the speeds to be injected into the robot's controller.
- 6. We repeated until the convergence was obtained.

Figure 5 shows a flow diagram illustrating the previous steps.



Figure 5. Control algorithm.

6. Results and Discussion

6.1. Results

The LCM-CSM view parameters were initially estimated for the robot's workspace. A total of 410 evenly distributed point positions in the Delta PARALLIX workspace had dimensions of $250 \times 250 \times 225$ mm (l × w × h). The workspace was sampled using a grid with a 50 by 50 mm separation between the adjacent points on the *x* – *y* plane, and 25 mm on the *z*-axis.

The proposed control law (based only on the observations of the camera space and the robot's joint space) was applied to the robot, and its performance was then evaluated in three different tasks.

For a set of 30 trials of *static positioning tasks*, an average positioning error of 1.086 mm was found (see Table 1). Figure 6a shows the decreasing error signal, measured in pixels, in the camera space. This was observed as an approach of the end-effector to the target position, visible in the same figure. Figure 6b shows the joint velocity vector. Note that the robot's end-effector stopped as it reached close to the target.

Table 1. Positioning error for static targets.

Error (mm)		
Average	1.086	
Max.	1.36	
Min.	0.76	
Std. Dev.	0.195	



(a) Positioning error in camera space.

(b) Joint velocities.

Figure 6. Error in camera space and joint velocities during a typical static positioning task.

For the *constant speed*, *linear trajectory tracking tasks*, a tracking or dragging error ranging from 9 to 28 mm was found while running 30 experiments of this type. This error seemed to be a function of the target/conveyor speed and the control gain (**K**). These experiments, and their associated dragging errors, are summarized in Table 2. Figure 7 shows a typical example of the position error's evolution of error in both camera space and Cartesian space. This figure presents results at different target speeds. It is worth noting that the dragging error increases as a function of the target speed.

Finally, tracking results for the *freehand trajectories* are shown in Table 3. In this case, the robot's end-effector followed the target before it stopped at a point inside the robot's workspace. The positioning accuracy for this final point is shown in Table 3 labeled as the *final error*. A RMS tracking error on the order of 10 mm during the motion of the target and a final positioning error on the order of 1.4 mm were observed for these experiments. Figure 8 shows the freehand trajectories as well as the position errors in both the camera and the Cartesian spaces.

Conveyor Speed	<i>g</i> ii	RMS Tracking Error (mm)
	2.3	11.2137
7 cm/s	2.7	9.6279
	3.1	8.8899
9.5 cm/s	2.3	14.8628
	2.7	12.3435
	3.1	11.7613
12 cm/s	2.3	27.8823
	2.7	20.9840
	3.1	18.6458

Table 2. Tracking error for a target with constant speed linear motion.



(a) Image error (7 cm/s)



(**b**) Image error (9.5 cm/s)



(c) Image error (12 cm/s)

Figure 7. Tracking errors in 2D and 3D for a target in linear motion at a constant speed (the number in the parenthesis indicates the target speed).



Figure 8. Measured target trajectories during freehand motion.

Trajectory	RMS Tracking Error (mm)	Final Error (mm)
circle	7.521	1.44
square	10.471	1.41
decreasing spiral	9.021	1.23
lemniscate	9.661	1.16
zig-zag	10.788	1.29

Table 3. Tracking and final positioning errors for freehand moving target along different trajectories.

6.2. Discussion

From the results obtained for the *static positioning tasks*, on a set of 30 positioning maneuvers, the precision achieved by the implemented control was on the order of 1 mm. This is consistent with existent implementations of CSM-based position control laws, which are sufficiently accurate for many industrial tasks [20]. Finally, it is worth noting that the parallel robot used in this work was an academic prototype with a precision of 3.17 mm for static positioning when its original kinematic controls were used [23]. Moreover, the original kinematic controls did not allow for the tracking of a-priori unknown trajectories. Based on this, we can say that the proposed control scheme improves the performance of the platform.

For the experiments involving *constant speed linear target trajectories*, there were relatively high tracking/dragging errors throughout the experiment. This was the result of the object not stopping during the maneuver and was likely a consequence of an incorrect estimation of the target speed. Several factors may contribute to this, above all, the constant control cycle time. As the target's speed increased, the convergence of the Kalman filter degraded. This is because the distance traveled by the target was larger between samples while the sampling rate remained constant. Another factor was the small but rapid variations of the conveyor belt's speed. This might be due to the electronics in the speed variator driving the conveyor's motor at a very low speed. These issues were not particular to the proposed control scheme. In any controlled system, the efficiency of the control scheme implementation was dependent on the hardware as well as on the relative speed of the controller dynamics and the dynamics of the physical systems. From the results summarized in Table 2, it can be observed that the tracking error increased as the target's speed increased. It can also be seen that the tracking error decreased as the control gain increased. This was to be expected as the control gain acts directly on the positioning error. The obtained error can still be manageable for a number of industrial applications and it can be further reduced by improving the speed estimation using a higher rate sampling (provided the use of higher performance hardware) and finer tuning of the control gain.

For the case of the *freehand trajectories*, there was no control applied on the target trajectory and, therefore, they could be considered similar to arbitrary trajectories. The tracking errors for these types of trajectories are of the same order as those of linear trajectories, even when there is no knowledge of the target speed (the target motion was performed at speeds close to 7 cm/s). The final positioning precision was consistent with that achieved during the static positioning tasks (c.f. Table 1). These errors could be reduced by decreasing the image acquisition times, which, in this implementation, were hardware-dependent.

These results corroborate the ability of the proposed velocity-based CSM variant to track moving objects, something not achievable by most traditional CSM approaches or the PARALLIX's original kinematic control scheme. Therefore, the proposed control scheme is an improvement over existing CSM approaches and the original control of the robot. For static tasks, the proposed approach achieves a positioning error of the same order of magnitude as other existing CSM approaches; this error is smaller than the one reported for the original robot non-CSM control [23].

In Figure 9 and Table 4 is shown a comparison of this work with significant works reported in the literature regarding vision-based robot controllers for different robotic architectures. In the figure, the coordinates of the center of each ball are given by the

cycle time (in ms) and the average positioning error (mm); the radius of each ball is the standard deviation (mm) of the positioning error. It can be seen that this work produces a low positioning error with small standard deviation and an good cycle time. Table 4 shows the comparison in numbers as well as tracking errors for moving targets and provides the references to those works.



Figure 9	. Static	error	vs.	cycle	time.
----------	----------	-------	-----	-------	-------

Table 4. Cycle time and tracking error comparison.

Control Cycle Time [ms]	Static Error (mm)	Std Dev (mm)	Tracking Error (mm) @ vel (mm/s)	Method
325	1.26	0.34	NA	Traditional CSM
242	1.11	0.35	NA	Gonzalez et al. [18]
8.33	4.21	2	20@800	Visual Servo—Trashlosheros [41]
NA	0.4	0.21	NA	CSM—Bonilla [42]
1.4	4	1	NA	Visual Servo—Özgür [43]
58.8	3.5	1.6	NA	Stereo—Chen [44]
57.3	1.48	0.43	NA	LCSM—Lara et al. [23]
48	1.09	0.19	8.89@70, 11.76@95, 18.65@120	CSM—Vel

We also compared our work with some recent work in the literature that utilized both vision-based and non-vision-based controllers for Delta-type parallel robots.

The work in [3] focused on the tracking error of a 3DOF parallel manipulator using a visual control. Figures 9–11 in [3] report the error and displacement on each axis were plotted. The graphs show that the ratio between the maximum error and the maximum displacement within the workspace was about 0.01.

In [45], the tracking error of a Delta robot was controlled using computer vision. In this case, the error was calculated as the difference between the actual and desired angle of the joint. The error was presented as an angle where the peak error at the joint was 0.025 rad. Given that the trajectory in the Cartesian space was a circle with a diameter of 0.1 m we estimated that the rotation of the motor could not be larger than 1 rad given the dimension of the robot. Hence, the ratio of error to workspace was about 0.025.

In [8], the same type of Delta robot was controlled with a non-linear control. It is easy to see from Figures 8 and 9 in [3] at the maximum error of the end effector position was 0.006 m for a maximum displacement of 0.15 m, for a ratio of 0.04.

Finally, in [41], the error of a parallel robot in the 3D visual tracking of a ball, at velocities up to 1 m/s, controlled with a visual system, is presented. The oscillation of the ball in each direction was about 200 mm, and the tracking error was up to 15 mm, producing a ratio of 0.075

In our work, the tracking error varied from 8.89 mm at 7 cm/s to a maximum of 18.65 mm at 12 cm/s. Considering that each axis of the Delta robot in question was 250 mm, the ratio of our controller was capable of varying between 0.035 and 0.075. It can be seen that depending on the speed of our controller, it is comparable to other controllers presented

in the literature. We demonstrated that CSM controllers, which are usually utilized for point-to-point pick-and-place, could also be used in tracking tasks, using the modification we proposed to the controlling algorithm.

7. Conclusions

This article presents the development and implementation of a novel velocity-based control on Camera Space Manipulation (CSM) applied to a Delta-type parallel robot. While CSM is often implemented as a position-based control, the proposed new methodology uses velocity-based control, which is the main contribution of this work. This contrasts with previous work, such as [23], where the control signal was defined in the position only. The proposed control has the ability to manipulate static and moving targets, even when the trajectory of the target is not known in advance, in contrast, CSM was originally designed to manipulate only static targets.

It should be noted that image-based Visual Servoing (VS) control techniques share some characteristics with the proposed velocity-based CSM technique. However, in VS, the image Jacobian has to be recalculated in each control cycle because it depends on the position of the target point within the image plane and the corresponding location along the focal axis of the camera. In contrast, in the case presented herein, the Jacobian matrix associated with the vision scheme (J_{csm}) can be regarded as a constant if the positioning task is performed inside the volume that contains the pre-plan trajectory.

Using the proposed control strategy, in the case of static objectives, the average positioning error was 1.086 mm with a standard deviation of 0.195 mm and an average task duration of 1.79 s. These results are competitive when compared against a number of industrial applications. Regarding mobile object tracking tasks, the results show tracking errors of 8.89, 11.76, and 18.65 mm for targets moving along the linear trajectory at speeds of 7, 9.5, and 12 cm/s, respectively; and 11.79 mm in the case of an arbitrary freehand trajectory during motion, with a maximum final static positioning error of 1.44 mm once the object stopped. Although the tracking error is relatively large while the target is moving, when the target stops, the proposed control can achieve similar precisions for static positioning tasks.

Traditional control schemes are hardly applicable to Delta robots and other parallel robots in general, due to open problems in the literature, such as the determination of the direct kinematic and dynamic models and singularity analyses, among others. Commonly, Delta robots are used in industrial applications for "pick-and-place" tasks; however, these applications require online adjustments due to the handling of different products on the same line, specifications by the client, and the displacements of the moving targets. Moreover, vision control techniques replace traditional robot control schemes; the authors believe that, in these cases, the proposed vision-based control via CSM can provide a suitable solution.

Author Contributions: Conceptualization, A.G., A.C. and E.G.-G.; Formal analysis, M.M. and E.G.-G.; Investigation, A.L., M.M., A.G. and A.C.; Methodology, A.L. and D.P.; Project administration, M.M.; Supervision, M.M., A.C. and D.P.; Validation, A.L.; Writing—original draft, A.L., M.M. and A.G.; Writing—review & editing, M.M., A.G., A.C., E.G.-G. and D.P.; visualization, A.G. and M.M.; funding acquisition, M.M., A.C. and E.G.-G. All authors have read and agreed to the published version of the manuscript.

Funding: This work was partially supported by Consejo Nacional de Ciencia y Tecnologia (CONA-CyT) grant number Catedras-2016/972. The APC was funded by Tecnologico de Monterrey, Vicerrectory of Research and Technology Transfer.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

Appendix A

Appendix A.1. Target Velocity Estimation in Camera Space via Kalman Filter

The KF algorithm estimates the n - dimensional state x of a discrete system described by

$$\mathbf{x}_k = \mathbf{A}\mathbf{x}_{k-1} + \mathbf{B}\mathbf{u}_{k-1} + \mathbf{w}_k \tag{A1}$$

$$\mathbf{z}_k = \mathbf{H}\mathbf{x}_k + \mathbf{v}_k \tag{A2}$$

where *k* represents the discrete time instant being analyzed, **A** is the state transition matrix, \mathbf{u}_{k-1} is a control input, **B** the corresponding input matrix, and \mathbf{w}_k is a process noise with a covariance **Q**. State measurements **z** are functions of the system's current state following a model represented by **H** and the measurement noise denoted by \mathbf{v}_k with a covariance **R**. Both \mathbf{w}_k and \mathbf{v}_k have a zero-mean and are normally distributed. The KF does not only track the state vector but it also offers a statistical estimation of its uncertainty \mathbf{P}_k , known as the estimation covariance.

The KF comprises two stages as depicted in Figure A1. The first stage is a *prediction* phase where the state vector is calculated based on the system's model; moreover, the covariance matrix was calculated, resulting in the *a-priori* estimation, as indicated by the superindex "–":

$$\mathbf{x}_{k}^{-} = \mathbf{A}\mathbf{x}_{k-1}^{+} + \mathbf{B}\mathbf{u}_{k-1} \tag{A3}$$

$$\mathbf{P}_{k}^{-} = \mathbf{A}\mathbf{P}_{k-1}^{+}\mathbf{A}^{T} + \mathbf{Q}_{k} \tag{A4}$$

In the second stage, known as the *correction* phase, the state vector is updated based on the new state measurements. The *a-posteriori* estimation, indicated by the superindex "+", is obtained from this phase as follows:

$$\mathbf{K}_{k} = \mathbf{P}_{k}^{-} \mathbf{H}^{T} \left(\mathbf{H} \mathbf{P}_{k}^{-} \mathbf{H}^{T} + \mathbf{R}_{k} \right)^{-1}$$
(A5)

$$\mathbf{x}_{k}^{+} = \mathbf{x}_{k}^{-} + \mathbf{K}_{k} (\mathbf{z}_{k} - \mathbf{H}\mathbf{x}_{k}^{-}) \tag{A6}$$

$$\mathbf{P}_k^+ = (\mathbf{I} - \mathbf{K}_k \mathbf{H}) \mathbf{P}_k^- \tag{A7}$$

where **K** is the Kalman filter gain, and **I** is an identity matrix of suitable size. Note that the filter's gain is a function of the current measurement model Equation (A5), and the current estimation covariance.

,



Figure A1. Kalman filter stages.

Appendix A.2. Target Velocity Estimation

The set of images of the target position was acquired by two cameras. The position of the visual marker was determined after processing the camera images. In this way, the initial position and velocity of the target, expressed in the camera space, were obtained.

In the camera space, the motion of the moving target between two frames could be approximated as follows:

$$u^{*} = u_{0}^{*} + \dot{u}^{*} \cdot \Delta t + \frac{1}{2} \ddot{u}^{*} \cdot \Delta t^{2}$$
(A8)

$$v^* = v_0^* + \dot{v}^* \cdot \Delta t + \frac{1}{2} \ddot{v}^* \cdot \Delta t^2$$
 (A9)

where (u^*, v^*) , (\dot{u}^*, \dot{v}^*) , and (\ddot{u}^*, \ddot{v}^*) , respectively, denote the target's current position, velocity, and acceleration at time *t* in the camera space. A Kalman filter for estimating the target's position u^* and velocity \dot{u}^* can be written by setting:

$$\mathbf{x} = \begin{bmatrix} u^* \\ \dot{u}^* \end{bmatrix}, \ \mathbf{u} = \dot{u}^*, \ \mathbf{A} = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix}, \ \mathbf{B} = \begin{bmatrix} \frac{1}{2}\Delta t^2 \\ \Delta t \end{bmatrix}$$
(A10)

where Δt denotes the control's cycle time and is assumed to be constant. Additionally, u_0^* in (A8) represents the *a*-posteriori position obtained from the last observation (u_{k-1}^+) . An identical filter was set up for the v image coordinate. This was done for each participating camera. The control loop used the KF to estimate the target's velocity in the camera space (\dot{u}^*, \dot{v}^*) .

References

- 1. Merlet, J.P. Parallel Robots; Springer Science & Business Media: Dordrecht, The Netherlands, 2006.
- 2. Pandilov, Z.; Dukovski, V. Several open problems in parallel robotics. Acta Tech. Corviniensis-Bull. Eng. 2011, 4, 77.
- 3. Lee, L.W.; Chiang, H.H.; Li, I.H. Development and Control of a Pneumatic-Actuator 3-DOF Translational Parallel Manipulator with Robot Vision. *Sensors* **2019**, *19*, 1459. [CrossRef] [PubMed]
- Mostashiri, N.; Dhupia, J.S.; Verl, A.W.; Xu, W. A review of research aspects of redundantly actuated parallel robotsw for enabling further applications. *IEEE/ASME Trans. Mechatronics* 2018, 23, 1259–1269. [CrossRef]
- 5. Bonev, I.A.; Ryu, J.; Kim, S.G.; Lee, S.K. A closed-form solution to the direct kinematics of nearly general parallel manipulators with optimally located three linear extra sensors. *IEEE Trans. Robot. Autom.* **2001**, *17*, 148–156. [CrossRef]
- Bentaleb, T.; Iqbal, J. On the improvement of calibration accuracy of parallel robots—Modeling and optimization. *J. Theor. Appl. Mech.* 2020, 58, 261–272. [CrossRef]
- 7. Huynh, B.P.; Kuo, Y.L. Dynamic Hybrid Filter for Vision-Based Pose Estimation of a Hexa Parallel Robot. *J. Sens.* 2021, 2021, 9990403. [CrossRef]
- Amjad, J.; Humaidi, A.I.A. Design of Augmented Nonlinear PD Controller of Delta/Par4-Like Robot. J. Control Sci. Eng. 2019, 2019, 11. [CrossRef]
- 9. Cherubini, A.; Navarro-Alarcon, D. Sensor-based control for collaborative robots: Fundamentals, challenges, and opportunities. *Front. Neurorobot.* **2021**, 113. [CrossRef]
- Bonilla, I.; Reyes, F.; Mendoza, M.; González-Galván, E.J. A dynamic-compensation approach to impedance control of robot manipulators. J. Intell. Robot. Syst. 2011, 63, 51–73. [CrossRef]
- 11. Chaumette, F.; Hutchinson, S. Visual servo control. I. Basic approaches. IEEE Robot. Autom. Mag. 2006, 13, 82–90. [CrossRef]
- 12. Weiss, L.; Sanderson, A.; Neuman, C. Dynamic sensor-based control of robots with visual feedback. *IEEE J. Robot. Autom.* **1987**, 3, 404–417. [CrossRef]
- 13. Lin, C.J.; Shaw, J.; Tsou, P.C.; Liu, C.C. Vision servo based Delta robot to pick-and-place moving parts. In Proceedings of the 2016 IEEE International Conference on Industrial Technology (ICIT), Taipei, Taiwan, 14–17 March 2016; pp. 1626–1631. [CrossRef]
- 14. Xiaolin, R.; Hongwen, L. Uncalibrated Image-Based Visual Servoing Control with Maximum Correntropy Kalman Filter. *IFAC-PapersOnLine* **2020**, *53*, 560–565. [CrossRef]
- 15. Garrido, R.; Trujano, M.A. Stability Analysis of a Visual PID Controller Applied to a Planar Parallel Robot. *Int. J. Control Autom. Syst.* **2019**, *17*, 1589–1598. [CrossRef]
- 16. Skaar, S.B.; Brockman, W.H.; Hanson, R. Camera-space manipulation. Int. J. Robot. Res. 1987, 6, 20–32. [CrossRef]
- González-Galván, E.J.; Cruz-Ramırez, S.R.; Seelinger, M.J.; Cervantes-Sánchez, J.J. An efficient multi-camera, multi-target scheme for the three-dimensional control of robots using uncalibrated vision. *Robot. Comput.-Integr. Manuf.* 2003, 19, 387–400. [CrossRef]

- González, A.; Gonzalez-Galvan, E.J.; Maya, M.; Cardenas, A.; Piovesan, D. Estimation of camera-space manipulation parameters by means of an extended Kalman filter: Applications to parallel robots. *Int. J. Adv. Robot. Syst.* 2019, 16, 1729881419842987. [CrossRef]
- Rendón-Mancha, J.M.; Cárdenas, A.; García, M.A.; Lara, B. Robot positioning using camera-space manipulation with a linear camera model. *IEEE Trans. Robot.* 2010, 26, 726–733. [CrossRef]
- Coronado, E.; Maya, M.; Cardenas, A.; Guarneros, O.; Piovesan, D. Vision-based Control of a Delta Parallel Robot via Linear Camera-Space Manipulation. J. Intell. Robot. Syst. 2017, 85, 93–106. [CrossRef]
- Huynh, P.; Arai, T.; Koyachi, N.; Sendai, T. Optimal velocity based control of a parallel manipulator with fixed linear actuators. In Proceedings of the 1997 IEEE/RSJ International Conference on Intelligent Robot and Systems. Innovative Robotics for Real-World Applications. IROS'97, Grenoble, France, 11 September 1997; Volume 2, pp. 1125–1130. [CrossRef]
- 22. Chavez-Romero, R.; Cardenas, A.; Maya, M.; Sanchez, A.; Piovesan, D. Camera Space Particle Filter for the Robust and Precise Indoor Localization of a Wheelchair. *J. Sens.* 2016, 2016, 1–11. [CrossRef]
- 23. Lopez-Lara, J.G.; Maya, M.E.; González, A.; Cardenas, A.; Felix, L. Image-based control of Delta parallel robots via enhanced LCM-CSM to track moving objects. *Ind. Robot. Int. J. Robot. Res. Appl.* **2020**, *47*, 559–567. [CrossRef]
- Zelenak, A.; Peterson, C.; Thompson, J.; Pryor, M. The advantages of velocity control for reactive robot motion. In Proceedings of the ASME 2015 Dynamic Systems and Control Conference, DSCC 2015, Columbus, OH, USA, 28–30 October 2015; p. V003T43A003. [CrossRef]
- Duchaine, V.; Gosselin, C.M. General Model of Human-Robot Cooperation Using a Novel Velocity Based Variable Impedance Control. In Proceedings of the Second Joint EuroHaptics Conference and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems (WHC'07), Tsukuba, Japan, 22–24 March 2007; pp. 446–451. [CrossRef]
- Xie, Y.; Wang, W.; Zhao, C.; Skaar, S.B.; Wang, Q. A Differential Evolution Approach for Camera Space Manipulation. In Proceedings of the 2020 2nd World Symposium on Artificial Intelligence (WSAI), Guangzhou, China, 27–29 June 2020; pp. 103–107.
- Wang, Y.; Wang, Y.; Liu, Y. Catching Object in Flight Based on Trajectory Prediction on Camera Space. In Proceedings of the 2018 IEEE International Conference on Real-Time Computing and Robotics (RCAR), Kandima, Maldives, 1–5 August 2018; pp. 304–309.
- Xue, T.; Liu, Y. Trajectory Prediction of a Flying Object Based on Hybrid Mapping Between Robot and Camera Space. In Proceedings of the 2018 IEEE International Conference on Real-Time Computing and Robotics (RCAR), Kandima, Maldives, 1–5 August 2018; pp. 567–572.
- 29. Sekkat, H.; Tigani, S.; Saadane, R.; Chehri, A. Vision-Based Robotic Arm Control Algorithm Using Deep Reinforcement Learning for Autonomous Objects Grasping. *Appl. Sci.* 2021, *11*, 7917. [CrossRef]
- Xin, J.; Cheng, H.; Ran, B. Visual servoing of robot manipulator with weak field-of-view constraints. Int. J. Adv. Robot. Syst. 2021, 18, 1729881421990320. [CrossRef]
- Mok, C.; Baek, I.; Cho, Y.S.; Kim, Y.; Kim, S.B. Pallet Recognition with Multi-Task Learning for Automated Guided Vehicles. *Appl. Sci.* 2021, 11, 11808. [CrossRef]
- Singh, A.; Kalaichelvi, V.; Karthikeyan, R. A survey on vision guided robotic systems with intelligent control strategies for autonomous tasks. *Cogent Eng.* 2022, 9, 2050020. [CrossRef]
- Clavel, R. DELTA, a fast robot with parallel geometry. In Proceedings of the 18th International Symposium on Industrial Robots, Lusanne, Switzerland, 26–28 April 1988; Burckhardt, C.W., Ed.; Springer: New York, NY, USA, 1988; pp. 91–100.
- 34. Balmaceda-Santamaría, A.L.; Castillo-Castaneda, E.; Gallardo-Alvarado, J. A Novel Reconfiguration Strategy of a Delta-Type Parallel Manipulator. *Int. J. Adv. Robot. Syst.* **2016**, *13*, 15. [CrossRef]
- 35. Maya, M.; Castillo, E.; Lomelí, A.; González-Galván, E.; Cárdenas, A. Workspace and payload-capacity of a new reconfigurable Delta parallel robot. *Int. J. Adv. Robot. Syst.* 2013, *10*, 56. [CrossRef]
- López, M.; Castillo, E.; García, G.; Bashir, A. Delta robot: Inverse, direct, and intermediate Jacobians. Proc. Inst. Mech. Eng. Part C J. Mech. Eng. Sci. 2006, 220, 103–109. [CrossRef]
- 37. Hartley, R.; Zisserman, A. Multiple View Geometry in Computer Vision; Cambridge University Press: Cambridge, UK, 2003.
- 38. Liu, Y.; Shi, D.; Skaar, S.B. Robust industrial robot real-time positioning system using VW-camera-space manipulation method. *Ind. Robot. Int. J.* **2014**, *41*, 70–81. [CrossRef]
- 39. Galassi, M.; Theiler, J. GNU Scientific Library. 1996. Available online: https://www.gnu.org/software/gsl/ (accessed on 1 June 2017).
- 40. Bradski, G. The OpenCV Library. Dr. Dobb's Journal of Software Tools. 2000. Available online: http://opencv.org/ (accessed on 1 July 2016).
- Angel, L.; Traslosheros, A.; Sebastian, J.M.; Pari, L.; Carelli, R.; Roberti, F. Vision-Based Control of the RoboTenis System. In Recent Progress in Robotics: Viable Robotic Service to Human: An Edition of the Selected Papers from the 13th International Conference on Advanced Robotics; Lee, S., Suh, I.H., Kim, M.S., Eds.; Springer: Berlin/Heidelberg, Germany, 2008; pp. 229–240. [CrossRef]
- Bonilla, I.; Mendoza, M.; González-Galván, E.J.; Chávez-Olivares, C.; Loredo-Flores, A.; Reyes, F. Path-tracking maneuvers with industrial robot manipulators using uncalibrated vision and impedance control. *IEEE Trans. Syst. Man Cybern. Part C Appl. Rev.* 2012, 42, 1716–1729. [CrossRef]
- Özgür, E.; Dahmouche, R.; Andreff, N.; Martinet, P. High speed parallel kinematic manipulator state estimation from legs observation. In Proceedings of the 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, Tokyo, Japan, 3–7 November 2013; pp. 424–429. [CrossRef]

- Chen, G.; Peng, R.; Wang, Z.; Zhao, W. Pallet recognition and localization method for vision guided forklift. In Proceedings of the 2012 8th International Conference on Wireless Communications, Networking and Mobile Computing, Shanghai, China, 21–23 September 2012; Volume 2. [CrossRef]
- 45. Lu, X.; Liu, M. Optimal Design and Tuning of PID-Type Interval Type-2 Fuzzy Logic Controllers for Delta Parallel Robots. *Int. J. Adv. Robot. Syst.* **2016**, *13*, 96. [CrossRef]