

Deep Instance Segmentation and Visual Servoing to Play Jenga with a Cost-Effective Robotic System

Original

Deep Instance Segmentation and Visual Servoing to Play Jenga with a Cost-Effective Robotic System / Marchionna, Luca; Pugliese, Giulio; Martini, Mauro; Angarano, Simone; Salvetti, Francesco; Chiaberge, Marcello. - In: SENSORS. - ISSN 1424-8220. - ELETTRONICO. - 32:2(2023). [10.3390/s23020752]

Availability:

This version is available at: 11583/2974584 since: 2023-01-13T11:11:51Z

Publisher:

MDPI

Published

DOI:10.3390/s23020752

Terms of use:


This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright

(Article begins on next page)

Article

Deep Instance Segmentation and Visual Servoing to Play Jenga with a Cost-Effective Robotic System

Luca Marchionna ^{1,†}, Giulio Pugliese ^{1,†}, Mauro Martini ^{1,2,*} , Simone Angarano ^{1,2} , Francesco Salvetti ^{1,2} 
and Marcello Chiaberge ^{1,2} 

¹ Department of Electronics and Telecommunications (DET), Politecnico di Torino, 10129 Torino, Italy

² PIC4SeR Interdepartmental Centre for Service Robotics, 10129 Torino, Italy

* Correspondence: mauro.martini@polito.it; Tel.: +39-3936286325

† These authors contributed equally to this work.

Abstract: The game of Jenga is a benchmark used for developing innovative manipulation solutions for complex tasks. Indeed, it encourages the study of novel robotics methods to successfully extract blocks from a tower. A Jenga game involves many traits of complex industrial and surgical manipulation tasks, requiring a multi-step strategy, the combination of visual and tactile data, and the highly precise motion of a robotic arm to perform a single block extraction. In this work, we propose a novel, cost-effective architecture for playing Jenga with e.Do, a 6DOF anthropomorphic manipulator manufactured by Comau, a standard depth camera, and an inexpensive monodirectional force sensor. Our solution focuses on a visual-based control strategy to accurately align the end-effector with the desired block, enabling block extraction by pushing. To this aim, we trained an instance segmentation deep learning model on a synthetic custom dataset to segment each piece of the Jenga tower, allowing for visual tracking of the desired block's pose during the motion of the manipulator. We integrated the visual-based strategy with a 1D force sensor to detect whether the block could be safely removed by identifying a force threshold value. Our experimentation shows that our low-cost solution allows e.DO to precisely reach removable blocks and perform up to 14 consecutive extractions in a row.

Keywords: Jenga; robotic arm; deep instance segmentation; visual servoing; sensor fusion



Citation: Marchionna, L.; Pugliese, G.; Martini, M.; Angarano, S.; Salvetti, F.; Chiaberge, M. Deep Instance Segmentation and Visual Servoing to Play Jenga with a Cost-Effective Robotic System. *Sensors* **2023**, *23*, 752. <https://doi.org/10.3390/s23020752>

Academic Editor: José María Martínez-Otzeta

Received: 14 November 2022

Revised: 4 January 2023

Accepted: 6 January 2023

Published: 9 January 2023



Copyright: © 2023 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/>).

1. Introduction

In recent years, visual-based control strategies have successfully spread in a wide variety of robotics contexts [1]. Nowadays, advances in computer vision for robotic perception are strictly tied to deep learning (DL). DL has been used in many robotics applications where objects must be detected [2] or segmented [3] to address manipulation tasks, demonstrating competitive advantages compared to classic image processing algorithms in terms of accuracy and robustness. For instance, relevant works in the precision agriculture field have been proposed in recent years to support autonomous navigation [4,5], harvesting [6], and spraying [7]. Intelligent DL-based behaviors are also desired for visual-based robotic surgery to detect and segment instruments [8,9], and in many industrial robotic tasks [10,11]. Nonetheless, to fill the gap between robot and human perception, multisensory approaches have recently been studied and evolved in novel robotic platforms, combining visual data with vocal interfaces [12,13], or tactile sensors [14–16].

The game of Jenga is a perfect example of a challenging benchmark for robotic perception and control. In recent years, researchers have tackled the game with disparate platforms and approaches, adopting sophisticated manipulators [17,18] and complex control systems [19,20]. The contribution to an effective robotic solution for Jenga goes beyond the fascinating dynamics of this popular game. Indeed, it can support the evolution of cutting-edge visual and multisensory control strategies for complex real-world tasks requiring human-level precision. The case of Jenga is not isolated in the historical advancement

of artificial intelligence (AI), where games are often used as a common benchmark for newly proposed learning algorithms [21–23]. The complexity of a round of Jenga resides in two different factors: first, it requires a multi-step policy to select a feasible block in the tower, approach it, and finalize its extraction. Second, all of these steps are based on the combination of real-time visual and tactile data processing and the highly precise motion of the end-effector for a single block extraction. According to this, it can be surely compared to real-world industrial [11], surgical [24], and agricultural [25,26] manipulation tasks.

This work presents a cost-effective system to play Jenga with the educational robotic manipulator—e.DO by Comau and a custom pushing finger as an end-effector. Our proposed solution is based on the combination of visual and tactile perception to handle the human-level complexity and the high precision required by the task. In particular, compared to previous attempts to play Jenga with a manipulator, we adopted a single RGB-D camera and a basic 1D force sensor as complementary hardware to the robot arm, considerably reducing the cost and complexity of the solution. An illustrative sequence of frames depicting our robotic system in action is shown in Figure 1. Overall, our perception and control system is composed of the following:

- A DL-based instance segmentation model fine-tuned on a custom Jenga tower dataset realized in simulation, which effectively allows the system to segment and select single blocks;
- An eye-in-hand visual control strategy that carefully handles the block extraction;
- A 1D force sensor to correctly evaluate the removability of a specific block.

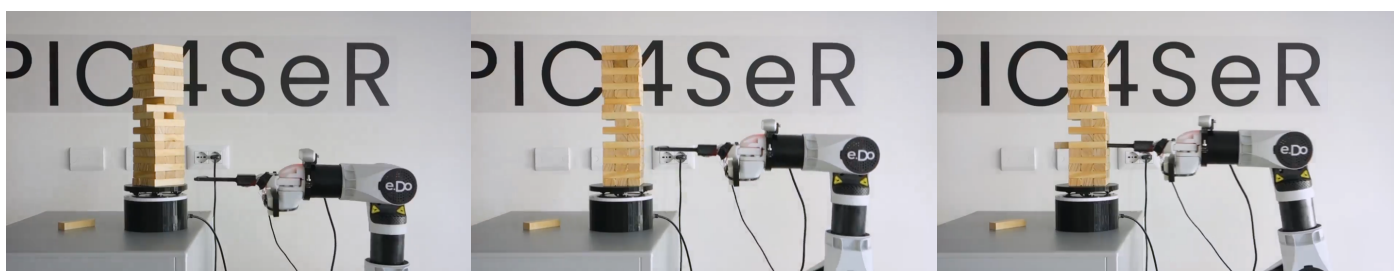


Figure 1. Our proposed solution in action: from left to right in the sequence, the e.DO robot selects a block to extract in the tower, adopting a visual-based control to approach it. Then, it starts pushing for the block extraction if it detects a low reaction force.

Our extensive experimentation validates all the sub-components of the proposed system. First, we study the force reaction on the 1D sensor and obtain a threshold-based decision policy to classify the extraction of the block as feasible or not. Then, we provide details on the training and testing of the instance segmentation model, comparing results obtained on simulated and real-world images. Moreover, we test our visual perception pipeline composed of segmentation and pose estimation of a group of blocks, measuring the tracking time during several runs. The adopted visual servoing control strategy is tested on the two major behavioral features of interest: precision, in terms of distance between the point of contact and the center of the block, and efficiency, estimated as the time required to align the end-effector to a block. Finally, we evaluate the overall performance of our solution by counting consecutive successful block extractions.

The paper is organized as follows. In Section 2, related works are presented in three subsections, discussing previous attempts to play Jenga with a manipulator and the state-of-the-art of deep instance segmentation, visual servoing, and a multisensory control strategy in robotics applications. Section 3 illustrates the overall strategy adopted to tackle the Jenga game and all the specific components of our solution. Finally, in Section 4, we present and discuss all the conducted experiments and the obtained results. Section 5 draws some conclusions and potential suggestions for future works.

2. Related Works

In this section, we first describe previous studies aimed at playing Jenga with a robot. Then we introduce the computer vision task of instance segmentation and its state-of-the-art and report similar works adopting visual servoing and multisensory control in robotic applications.

2.1. Playing Jenga with a Robot

Jenga is a common benchmark for robotic systems, allowing for a direct comparison of methods, experiments, and results. So far, few works have proposed a complete visual-based robotic system to play the game autonomously. At the same time, several studies focus on a partial aspect of the game with a specific solution.

Recently and most notably, Fazeli et al. [19] delved into the details of the manipulation and artificial intelligence capabilities needed to learn Jenga by sight and touch, achieving 20 consecutive block extractions; their system is made up of an end-effector that can push and pick blocks, a long-reach 6DOF industrial robot arm, an expensive 6-axis force sensor, and a fixed camera. The method is focused on learning from multisensory fusion and is compared to state-of-the-art learning paradigms with simulation and experiments, resulting in high performance and fast convergence. Their approach adheres to all the rules of Jenga with a tower in standard conditions and considers both block removal and placement on top of the tower. However, the hierarchical control strategy they adopt to carefully push the block during the extraction strongly relies on the use of an expensive 6-axis force sensor and a professional industrial arm, drastically increasing the overall cost of the solution.

Kroger et al. [17], who adopt similar expensive hardware, achieved 29 extractions with a 6DOF industrial robot arm, a 6-axis force sensor, a 6-axis acceleration sensor, a laser distance sensor, and two static CCD cameras. The authors develop a modular control system based on a generic number of sensors with a primitive manipulation programming interface to play a standard Jenga game, manually recoloring the blocks to help the vision system. Both reaction force and tower-perturbed movements seen from cameras provide extraction feedback when a block is randomly chosen to be extracted. Pose estimation from cameras is refined with a laser distance profiler before gripping the block, which is then placed on top with force feedback.

Wang et al. [27] propose a simpler system to leverage inexpensive vision and manipulation hardware to develop a strategic planner based only on visual feedback. They achieve up to five consecutive extractions. The limitations of using classical computer vision with two CMOS cameras and a 5-DOF pioneer short-reach robotic arm without force measurements led the authors to choose a quite different and simplified Jenga setup compared to the real one. Target blocks were partially pre-pulled and distanced one from the other, and the tower had half the levels.

A two-fingers, anthropomorphic, 7-DOF industrial robot arm is used in [18]. An eye-in-hand omnidirectional camera detects the tower configuration, and a block is chosen using a stability criterion. The block is grasped by the robot hand, which mounts a 6-axis force sensor on each fingertip. The Jenga setup is not standard, as the tower has only 10 layers. However, the system can detect and place blocks on the top of the tower, presenting a pretty high autonomy level.

A fine-grained kinematic analysis of the physics behind weight, friction interactions, and stability of the Jenga tower is studied in [28], both during and after the extraction. With a 6DOF manipulator, a custom gripper, and a 6-axis force sensor, they compare the real forces with the modeled ones and achieve 14 consecutive extractions before breaking the tower. No vision or pose estimation system is used, so a human operator must provide poses and manually rotate the tower.

Differently, Yoshikawa et al. [20] investigated a reinforcement learning approach, using a deep Q-Network and a 6DOF manipulator to correctly push a block without a priori knowledge of the kinematics and stability of the tower. The work is done in simulation on an ideal Jenga tower. Negative rewards are extrapolated from how humans play the game,

such as pushing in the wrong direction and touching other blocks. The result is a policy for precise movement.

Similar to most related studies, our solution uses a force sensor to detect push failures and empirically check the removability of blocks. On the other hand, our perception system presents several novelties: a block identification approach based on an instance segmentation neural network, an eye-in-hand camera configuration, and a visual servoing control for the manipulator.

2.2. Deep Learning for Object Recognition

During the last few years, deep learning [29] has achieved state-of-the-art performance on various computer vision tasks. Different methods can be used to extract knowledge from visual data and give them a semantic interpretation. The literature refers to classification as the task of assigning a descriptive label to the whole image. Several approaches have been proposed to solve this problem, introducing architectural methodologies, such as convolutions [30], residual connections [31], feature [32], and space attention [33], or the more recent transformer-based architecture with self-attention [34,35].

Suppose a more fine-grained semantic description is needed. In that case, the object detection task has the objective of localizing instances that belong to target classes with the regression of bounding boxes. This detection allows the system to understand the scene hierarchically depicted in the image, assign multiple labels, and spatially identify the objects in the image reference frame. Popular methodologies for object detection [36–39] have focused on efficient and real-time execution to be used on continuous streams of images.

On the other hand, the semantic segmentation task aims at assigning a semantic label at the pixel level by predicting masks that identify the portion of the image belonging to a certain class. Classical approaches to this task are based on fully convolutional networks organized in an encoder–decoder fashion [40,41], which adopt successive downsampling and upsampling operations to predict labels at the pixel level. The main difference between semantic segmentation and object detection is that the former does not identify single instances but only regions that depict objects of the target classes.

Instance segmentation aims to localize single instances by predicting masks. This approach allows the most precise interpretation of the input image since it avoids coarse bounding box localization by identifying masks at a pixel level. Several methods have been proposed in the literature to solve this task. mask-RCNN [42] extends an object detection method called faster R-CNN [43] and is based on a two-stage approach that first proposes possible regions of interest (ROI) and predicts segmentation masks and classes in the second stage. Other approaches are based on one-stage architectures [44,45] and are inherently faster than two-stage methods. Other approaches solve a semantic segmentation task and then perform instance discrimination with boundary detection [46], clustering [47], or embedding learning [48]. Recently proposed YOLACT [49] and YOLACT++ [50] focus on a real-time approach to instance segmentation that extends an object detection approach with mask proposals. The combination of mask proposals and bounding boxes gives pixel-level instance localization. In this work, we adopted this approach due to its computational efficiency and ability to detect many near objects, typical of Jenga block segmentation.

2.3. Visual Servoing in Robotics

Industrial robotic tasks, such as assembly, welding, and painting represent standard scenarios where manipulators execute repeatable point-to-point motion by using off-line trajectories [51]. However, the variability and disturbances of different environments may affect the estimation of the target pose and lead to a degradation of task accuracy. Moreover, there are better strategies than this open-loop control technique for motion-based objects due to the target position and orientation variations.

Visual-based control strategies recently emerged as valid candidates for real-time trajectory computation and correction. In particular, visual servoing was introduced in 1979 [52] and refers to closed-loop systems where visual measurements are fed back into the controller to enhance task precision. Several works have proposed this approach for robotics applications in medical [53], agricultural [54], and aerospace contexts. The ability to move a robotic arm flexibly in high-precision surgery operations [55–57] confirms the potential of this technique in complex scenarios where small errors can compromise human health.

Visual servoing taxonomy distinguishes two approaches [58] according to the type of tracker used to generate visual features. Image-based visual servoing reconstructs the relative pose of the target in the manipulator reference frame using the camera field of view. This approach is widely used in agriculture applications [59,60], where occlusions of the camera can lead to poor visual feature extraction.

On the other hand, position-based visual servoing leverages a priori geometrical information on the object to derive the corresponding visual features. Recently, hybrid schemes have tried combining the two techniques and leveraging 2D and 3D visual features. In [61], an aerial vehicle with a robotic arm presents a hybrid visual servoing scheme to plug a bar into a fixed base. In this case, a marker detector provides the pose information of the object to be grasped, narrowing the possible field of use. Indeed, the robustness of tracking algorithms remains a central problem for visual-based control. Recently, in [62], the authors proposed a novel approach that uses augmented reality (AR) to generate 3D model-based tracking or 3D model-free tracking techniques to enhance the system's robustness.

3. Methodology

In this section, we frame the Jenga game and translate the rules of the game into a methodological set of requirements for the robotic system. First, a player's final goal is safely removing blocks from the tower. In the original game setting, a player has to place each extracted block at the top of the tower to validate its round and continue with a new block extraction. In our robotic experimental setting, we remove this rule and aim to extract as many blocks as possible from the tower without reallocation. This choice is mainly related to the limited workspace of the e.DO manipulator, designed for educational purposes, since it cannot reach the fallen blocks behind the tower. Moreover, our custom end-effector, similar to a human finger, cannot perform grasping and instead extracts blocks by pushing. As the first practical task, a Jenga player should be able to select one of the blocks of the tower to be extracted. To this end, each block of the tower is identified in our system using an instance segmentation deep neural network. Moreover, we define a heuristic block selection policy based on the idea that extracting multiple blocks from the same tower level is not recommended. Moreover, as better detailed in Section 3.2, blocks at different tower layers present diverse frictions and effects on the tower's configuration.

Therefore, our solution is based on the following assumptions:

- The identification and pose estimation of each block of the tower is the first step to selecting a suitable piece and approaching it;
- A removable block cannot be identified only by visual analysis: the integration of a tactile perception system is needed;
- A sufficiently precise alignment between the end-effector and the center of a target block allows the arm to extract it successfully by simply pushing.

A complete illustrative schematic of the system proposed to play Jenga is depicted in Figure 2.

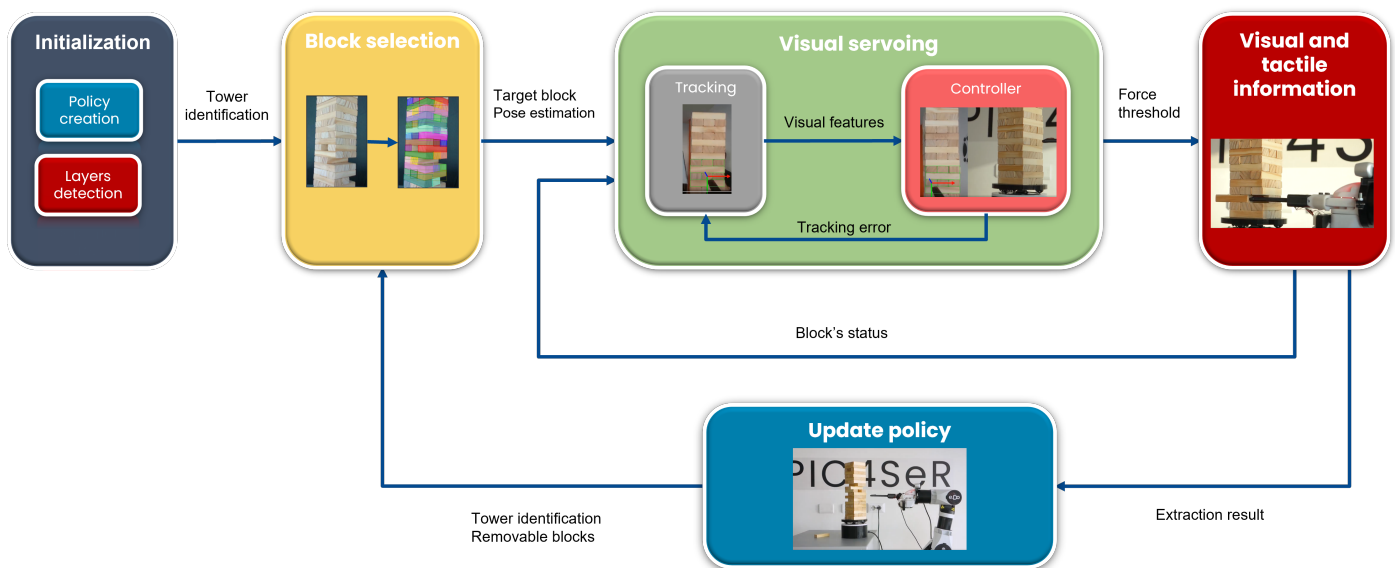


Figure 2. Block diagram of the proposed system architecture for Jenga: after tower identification, an RGB image of the tower is first used to extract the segmentation masks of each block with the deep instance segmentation neural network. A block selection policy chooses the block to extract while considering blocks that have already been tried. The selected block is precisely approached by the end-effector adopting a visual-based control strategy. Finally, the end-effector pushes the block, and an inexpensive 1D force sensor verifies its removability to stop or approve the extraction attempt.

3.1. Deep Instance Segmentation and Pose Estimation

The proposed methodology's first step relies on identifying the blocks present in the tower by visual analysis. We adopt a deep learning-based method to perform instance segmentation. The necessity of using a deep learning approach is caused by the fact that wooden blocks do not have easily-distinguishable features while having predictable positions in the camera frame instead. In this setup, the vision system can benefit from the ability of neural networks to generalize to different light conditions and points of view. Moreover, training the model on an exhaustive dataset can make it robust towards missing blocks and tower misplacement during the game. Among different approaches for image semantic analysis, instance segmentation aims at detecting all the objects of interest in an image at the pixel level. Given an input image of the tower, the model should output a set of possible Jenga block candidates, together with their respective pixel masks. Instance segmentation can therefore be seen as a combination of an object detection task with a mask prediction task. By translating the position of the blocks from the image reference frame to the robot reference frame, it is possible to achieve block localization.

Model Architecture

We select the YOLACT++ [50] architecture to implement the instance segmentation algorithm. This architecture has been chosen for its computational speed, capability to handle occlusion, and efficiency in detecting a high number of tightly packed same-class objects. The model is based on a ResNet-50 [31] + FPN [63] feature extraction backbone followed by two branches, one dealing with object detection and the other with mask prototype production. The detection branch outputs a set of anchor predictors a as possible Jenga block candidates. Each anchor prediction consists of the class confidence c , four bounding box regressors, and k mask coefficients. These mask coefficients are used to weigh the mask prototypes produced by the second branch. Both branches are based on convolutional layers applied to the features extracted by the common backbone.

The two branches are finally followed by a mask assembly block, which combines the masks with the coefficients predicted by each anchor to reach the final instances prediction. At inference time, anchor predictions are thresholded with a certain value t_c on their

confidence c score to produce the actual output. Moreover, as in standard object detection algorithms [36,37], an NMS (non-maximum suppression) method is applied to remove redundant predictions.

3.2. Block Selection Policy

Our solution defines a heuristic block selection policy based on physical and empirical considerations. Visual information cannot provide sufficient information to determine the status of a block. The imperceptible tolerances of each block cause minuscule variations in pressure that prevent visual-based systems from understanding which blocks are truly removable. The blocks in the higher layers are easier to extract, i.e., they can be pushed out of the tower by applying a smaller force. However, pushing from a decentralized contact point contributes to the formation of torques on the block that causes asymmetry in the tower. This effect is amplified on upper layers due to lower friction forces and may affect the stability of either adjacent blocks or the tower itself. Instead, the friction force increases for blocks located at lower levels, making the extraction harder and risking the Jenga tower falling.

Such considerations imply the need for a policy to select the block to extract. However, the policy must also consider the physical dimensions of our 6DOF anthropomorphic manipulator. Indeed, during the extraction primitive, the robot has to be parallel to the block, which implies a loss of DOFs. These orientation constraints restrict the robot's workspace, so the manipulator can only reach a limited range of tower levels. In order to overcome these issues, the policy divides the entire tower into two subspaces according to the robot's workspace. In particular, the subspaces correspond to the upper and lower levels, each with a predetermined number of tower levels. By convention, the numeration of tower levels carries in ascending order, where one corresponds to the lowest level. In addition to this vertical division, the policy is also initialized with the block direction for each level. Indeed, two possible tower orientations have a relative rotation of 90 deg between the two. The pose estimation described in Section 3.3, applied by the convention on the top, provides the reference to infer the actual orientation of the tower and all its levels.

As human players usually do, we initially adopt a random policy that selects the Jenga pieces in one of the subspaces. Then, the manipulator approaches the chosen block and starts pushing. At this point, force sensor data are collected to evaluate the block's status according to the adopted force threshold, as explained in Section 3.5. After each trial, a memory list updates the information on extracted and tested blocks. More in detail, a memory buffer stores the information about the blocks: the status (present, tested, or extracted), the threshold force to apply, and the current number of extraction attempts. In addition, it also keeps track of additional layers as the game goes on. Only one piece per level can be extracted as a further safety measure.

The policy repeats the process until all levels contained in the subspace are tested. After that, the system changes the subspace to test additional levels until the tower collapses. More in detail, selecting the higher layers as the starting subspace leads to three subspaces in total, with extra layers being included in the final subspace. Alternatively, if the process begins by extracting blocks from the lower layers, there will be only two subspaces, with extra layers being included in the high subspace. Figure 3 provides a minimal representation of the policy strategy. Except for the pick-and-place operation, the game implementation does not neglect any official rule.

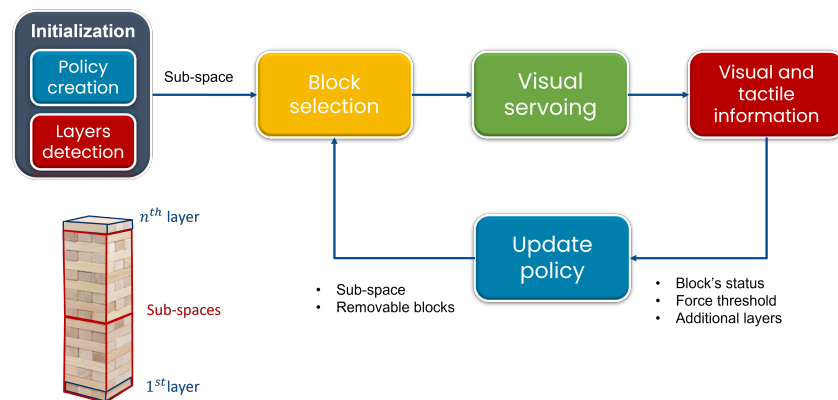


Figure 3. High-level input-output diagram describing the logic flow of the policy used to select the next block to extract from the Jenga tower. The layers of the tower are numbered from the 1st to the n th starting from the bottom, as shown in the schematic on the left. The policy starts to extract the blocks from one of the two subspaces of the tower, randomly chosen. It proceeds until all levels contained in the subspace are tested, to finally switch to the other subspace trying to avoid the tower to collapse.

3.3. Block Pose Estimation

Predicted block masks are the input of a post-processing unit implemented using OpenCV. It estimates the position and orientation of the desired block to be extracted with respect to the camera. This information is the first requirement to approach the block's face starting the manipulator's tracking and visual servo control. The pose estimation is performed with an intrinsic calibrated camera through the Perspective n Points (PnP) algorithm optimized for planar points [64]. PnP is applied to the four corners of the front face of the target block, separately identified from the others thanks to the predicted segmentation mask, and paired with the known dimensions of the block. To increase the robustness of the tracking algorithm and the overall precision of the visual control of the manipulator, corners of nearby blocks are also considered (if present). Adjacent blocks may have diverse mask shapes: smaller for occluded front-facing blocks and larger for side-facing ones. Hence, two different sets of points are considered to estimate their poses. Side-facing blocks can provide a significant advantage in the subsequent tracking phase, offering a more stable visual reference during the motion of the end-effector.

Therefore, the PnP algorithm provides an initial estimate of the 6DoF pose of the group of blocks (target and adjacent blocks) using segmentation mask corners to initialize the model-based tracker.

3.4. Tracking and Visual Servoing

The geometric dimensions of a standard piece of Jenga ($25 \times 15 \times 75$ mm) require precise movements to perform extraction successfully. Considering such dimensions and the width of the custom fingertip (i.e., 11 mm), it can be shown that the maximum position error from the center of the block must be smaller than 7 mm. Therefore, the maneuver of the end-effector requires accurate trajectory planning to approach a block precisely. Standard point-to-point planning and online control [65] can be considered valid methodologies. However, these control schemes adopt an open-loop control system which requires high precision on pose estimation and tiny mechanical tolerances to reach the desired point with a small error. On the other hand, visual servoing is a closed-loop control strategy that exploits visual measurements to correct the pose of the end-effector with respect to the target in real time. Continuous visual feedback is used to correct the end-effector trajectory and to align the relative pose of the camera with the target block. For this reason, visual servoing is a competitive and flexible strategy to accurately approach the desired object (in this case, the Jenga block to extract).

In this regard, a robust tracking algorithm is a fundamental and challenging component of visual servoing control, which is required to guarantee smooth trajectories and allow a faster convergence to the target. As described in the previous sections, the segmentation masks are used to estimate the position and orientation of the desired block. At this stage, the tracking system receives both the initial pose estimation and the segmentation masks and combines them with a 3D model of a Jenga block. Therefore, it detects the target to establish a continuous mapping of the 6DOF pose of the Jenga block in the camera field of view. Specifically, the adopted stereo model-based tracking method ViSP [66] combines several visual features, such as moving edges, key points, and depth information, to improve stability and robustness. The ViSP tracker requires the 3D model of a generic block to project its geometry into the image space and generate the visual features accordingly.

However, as mentioned in the previous Section 3.3, the tracking performance is only partially reliable if only the visible face of the desired block is used (Figure 4). For this reason, the tracking algorithm does not rely only on the visual information of the single block to extract, but it integrates adjacent blocks in a unique group model. As already discussed in the previous Section 3.3, this choice leads to more robust tracking thanks to the higher number of visual features extracted, especially from side-facing blocks.

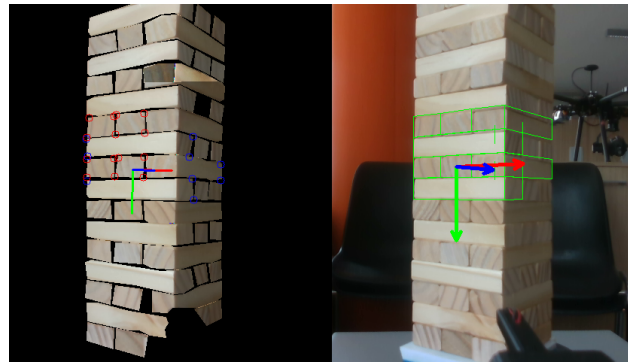


Figure 4. The initial pose estimation of the target block obtained by exploiting the segmented mask is shown on the left. The group model composed of multiple blocks to increase the robustness of the tracker is shown on the right.

Moreover, the visual features of the group of blocks of interest are collected from different perspectives before the tracking is started. This acquisition phase enables recovery tracking in case of sudden movements by exploiting the acquired pairings and re-initializing the block pose.

Our approach adopts the eye-in-hand visual-servo configuration, whose extrinsic camera parameters are estimated based on the end-effector design. Thus, we indicate with $s = (x, y, \log(Z/Z^*), \theta_u)$ the visual features constructed for the 2 1/2-D visual servoing tasks, where (x, y) are the coordinates of the target in the camera reference frame, Z and Z^* are the current and desired depth of the point, respectively, and θ_u is a 6×1 vector that identifies the rotation angles (expressed in the axis-angle convention) that the camera has to follow. Moreover, we refer to \widehat{L}_s^+ as the approximation of the interaction matrix and to e_s as the tracking error of the visual features, s , defined as $e_s = s - s'$, where s' denotes the desired visual features.

The 6×1 vector v represents the linear and angular camera velocities that are computed through the following control law:

$$v = -\lambda \widehat{L}_s^+ e_s \quad (1)$$

Hence, the above regulation control law defines the linear and angular velocities the camera has to follow to reach the target. Using forward kinematics, the vector $v \in \mathbb{R}^{6 \times 1}$ is converted into the task space and executed through a motion rate controller using inverse differential kinematics. The overall closed-loop system runs at about 9 Hz on the laptop i7-8750H CPU.

To further optimize the smoothness of the trajectory, we consider fixed constraints on the maximum joint velocities that the controller can predict. In Section 4, we show that the visual servoing pipeline we adopted increases the overall accuracy and robustness of the Jenga extraction system.

3.5. Tactile Perception

Tactile perception is a fundamental aspect of a human Jenga player since identifying potential removable blocks is not possible via visual analysis only. According to this, we decide to incorporate a low-cost 1D force sensor in our system, which only provides information about the perpendicular reaction force between the end-effector and the target block. Hence, while previous works adopted a 6D force sensor to adjust the direction of the end-effector while pushing [19], we take advantage of our visual-based control strategy and guarantee considerably good precision by simply pushing the block forward. The one-dimensional force sensor is mounted directly on the fingertip of the end-effector, as shown in Figure 5, and provides a digital output with a full-scale force span of 5 N. The sensor is activated when an interaction between the fingertip and the block occurs. The block can be either stuck or free to move, so the push primitive uses the reaction force to derive a binary block classification. The closed loop explained in Section 3.4 reads the measurements at 9 Hz while the Arduino microcontroller sends them at 20 Hz.

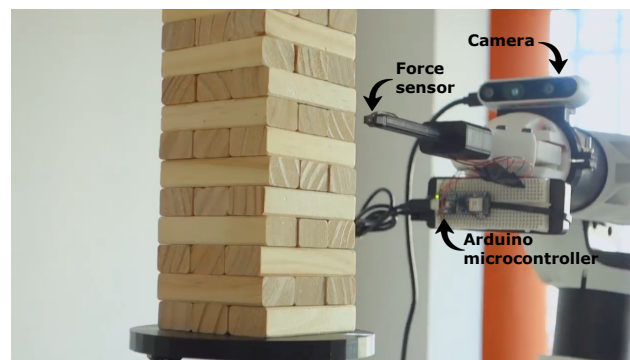


Figure 5. The manipulator has a human finger-inspired end-effector, which mounts a simple 1D force sensor on its tip. The force sensor controlled by an Arduino Nano board reveals the status of a block (removable or not) once it is approached and pushed. An eye-in-hand camera configuration enables the visual control pipeline to align the fingertip with the Jenga piece accurately.

A challenging aspect of the game is that the more push attempts are performed, the more unstable the tower becomes, as extracted blocks or rotations perturb its structure due to pushes and retractions. Therefore, two thresholds detect immobile or moving blocks in $thr_1 = 0.32$ N and $thr_2 = 0.18$ N. Such values are determined by combining theoretical [28] and empirical results, better depicted in Section 4.1. The threshold changes to a smaller conservative value in the second phase of the game when all the higher levels have been tested. Our choice can be defined as conservative, as it safeguards the stability of the tower rather than seeking more competitive performance.

4. Experiments

The experimental setup includes the anthropomorphic educational manipulator, e.DO manufactured by Comau, a depth camera Intel RealSense D435i, a MicroForce FMA piezoresistive force sensor (5 N full scale, 12-bit resolution for 0.002 N sensitivity), an Arduino Nano 33 BLE, and 3D printed components such as a rotating base, an end-effector design extension, and camera support. The software runs on a single computer with an i7-8750H CPU and 16 GB of RAM. Thus, one of our goals is to investigate and prove the effectiveness of state-of-the-art with low-cost equipment. Visual control adopts an eye-in-hand configuration with camera support, while the small force sensor is placed on top of the end-effector extension connected to the Arduino Nano.

4.1. Reaction Force Threshold

In this section, we first describe the experiments carried out to define the static force threshold used to check the removability of blocks. The real Jenga blocks present small differences in dimensions, generating a diverse pattern of friction forces in the tower each time it is rearranged for a new game. Although [28] tried to provide a rigorous mathematical formulation of friction forces in the tower, we prefer an experimental approach to identify the correct threshold values to detect whether or not the robot push affects the stability of the tower.

The measurements are taken with a complete tower configuration during the block extraction, starting the data collection of force reaction from when the contact between the block and end-effector starts and the force sensor detects a non-zero value. The experiment is run multiple times on different tower levels, mixing the blocks' disposition each time to test random friction conditions. The plot in Figure 6 shows the force profile of 15 blocks located at different tower positions over time.

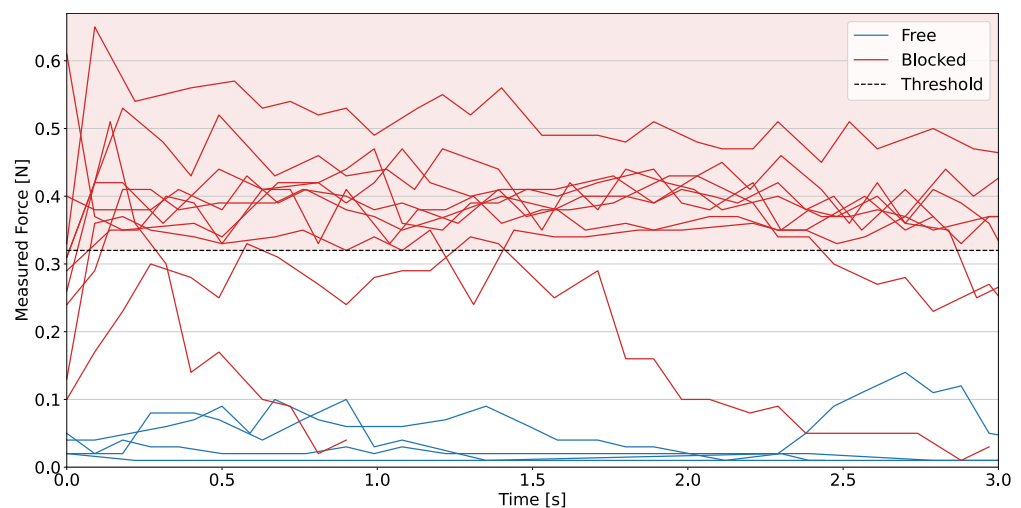


Figure 6. Reaction force measured over time during pushes for several blocks located in different layers and tower positions. The threshold force value of 0.32 N is also plotted to distinguish moving blocks in red from constrained ones in blue.

The choice of the threshold force values is not trivial, as it affects the system's ability to classify the block's removability state. By setting higher threshold values, more blocks can be classified as false positives, i.e., they are perceived as free to move, whereas they cannot be safely extracted. On the other hand, imposing lower threshold values affects system performance leading the system to avoid feasible block extractions.

As depicted in Figure 6, the initial threshold value is set to 0.32 N, making the system more likely to detect false positives than false negatives. In addition, this is reflected in a more aggressive strategy at the beginning of the game, enhancing performance at the cost of tower stability. After attempting to remove half of the levels, the threshold is reduced to preserve tower stability. In fact, after removing a certain number of blocks, the tower becomes increasingly unstable, and the friction forces change according to the position of the extracted blocks. Due to the static friction force, the superposition of these effects increases the probability of disrupting the tower as soon as the contact between the fingerprint and block occurs. Therefore, mechanics and empirical observations led us to lower the threshold value to 0.18 N. In general, the measurements are in the same range as the results obtained by the more extensive analysis of [18,19,28], with forces between 0 N and 1 N and thresholds ≥ 0.2 N.

4.2. Instance Segmentation Experiments

In this section, we report the method and the details of the procedure to train the instance segmentation model with a carefully devised synthetic dataset, as well as the results obtained by experimental validation. The experimentation aims to assess the quality of the deep learning model and its generalization to the real pictures of our experimental environment.

4.2.1. Training Setup

The training of the instance segmentation model is entirely performed on a synthetic dataset crafted from a 3D model of the Jenga tower. Using the 3D modeling software Blender and its Python APIs through BlenderProc [67], hundreds of photorealistic and varied images of the tower are produced with automatic pixel-perfect annotations. This approach makes it fast and easy to obtain hundreds of samples without manually labeling real images. The training and validation datasets are composed of 800 and 80 images of size 640×480 , respectively, rendered from a Blender synthetic scene composed of 48 cuboids arranged in a tower of 16 levels. We apply 12 different wood materials to the faces of these blocks to simulate the possible colors and wood line patterns with a realistic look. Each scene is loaded with blocks, a virtual camera, and a point light source. We design the following levels of scene randomization:

- The materials are randomly sampled and assigned to all 48 cuboids to change their look;
- A total of 6 to 24 cuboids are randomly displaced along their x and z axes by a distance between -4 mm and 4 mm;
- A point light source is positioned by randomly sampling a height of $0.1 \div 0.5$ m spanning a circular arc of 60 deg centered on the tower with a random radius between 0.4 and 0.7 m;
- A total of 2 to 9 random blocks are removed from the tower to create holes;
- Camera position is sampled on a circular arc of 20 deg with a height between 0.05 and 0.2 m and a radius between 0.25 to 0.45 m.

The camera is placed to capture two faces of the tower at the same time. For each configuration, 10 samples are acquired with the full tower and 10 with random missing blocks. The procedure is repeated for each random scene obtaining a diverse and complete dataset with different light conditions, block displacements, missing blocks, camera angles, and views of the experimental conditions. The scene's background is then filled with black, white, or gray. The render time for all the 880 images was 4 hours on an i7-9700K CPU. The test set is composed of 20 real manually-annotated pictures from our experimental setup for a total of around 800 segmented tower blocks. The images are taken in slightly different light conditions and camera positions.

The input 640×480 images are rescaled to 550×550 to be compatible with network input requirements. We adopt a ResNet-50 backbone with pre-trained weights on ImageNet. We perform 8000 training iterations (69 epochs) with a batch size of 8, SGD optimizer with a momentum of 0.9, and a weight decay of 5×10^{-4} . The initial learning rate of 10^{-3} is scaled down by a factor of 10 at iterations 5000, 6000, and 7000. We consider a positive intersection-over-union (IoU) value of 0.5 during training. The training is performed on a Tesla K80 GPU with Cuda 11.2.

4.2.2. Instance Segmentation Results

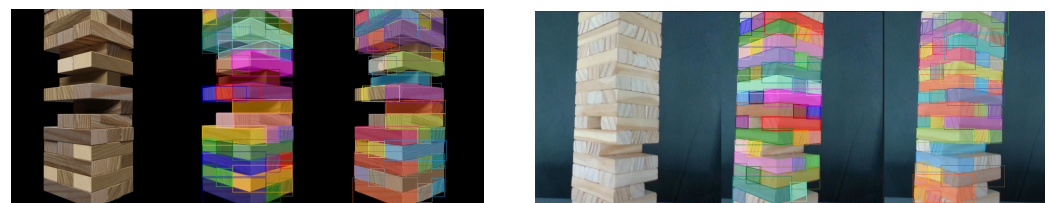
As the main metric to assess the quality of the instance segmentation, we adopt the widely used average precision at different values of intersection-over-union (AP_{IoU}). A predicted mask is considered a true positive (TP) if it has an IoU with the ground-truth mask over the given threshold. AP is then computed as the area under the curve of the precision–recall plot obtained varying the confidence threshold t_c .

Table 1 reports the AP results at different IoU values, both on the synthetic test set and the real manually-annotated dataset. As expected, increasing the IoU threshold results in a performance drop due to the stricter requirements asked of the model. Generally, we

observe a certain drop in performance when considering the real-world dataset, mainly due to border effects caused by approximate hand-made annotations and a decreased recall caused by the high number of instances in a single image. However, since a high recall is not required to perform block selection and tracking effectively, we state that the obtained real-world generalization is good enough for the target application. Visual comparison of a synthetic and a real image is reported in Figure 7a,b.

Table 1. Mask AP results at different values of IoU for the Jenga blocks instance segmentation model on both the synthetic and the real manually-annotated test datasets.

Dataset	AP ₅₀	AP ₈₀	AP ₉₀	Mean
Synth	90.08	87.76	63.2	78.37
Real	75.98	53.40	11.53	53.09



(a) Synthetic Dataset (AP = 69.9% for 80% IoU) (b) Real Images (AP = 88.1% for 80% IoU)

Figure 7. Segmentation masks on simulated (a) and real (b) images. From left to right: source image, ground truth, and predicted masks.

4.3. Tracking Robustness

This experiment tests the robustness of the stereo model-based tracker in two different configurations to assess its ability to keep track of the object's pose during movements. In particular, we compare our experimental visual set up with the basic functionalities of the ViSP library [66]. The main difference lies in constructing the 3D block model and the tracker initialization method. A CAD model of the single block takes only into account the target piece and requires the user to initialize the model manually. Instead, our tracking system merges different pieces around the target block according to the tower arrangement described in Section 3.4. For this test, we exploit a rotating base to automatically turn the Jenga tower around its vertical axis by 45 deg clockwise or counterclockwise at a constant speed. This rotation brings one of the faces perpendicularly to the camera axis, thus keeping the target block always in the field of view.

While the tower rotates, spanning the whole angle range, we test the tracker to follow the block moving in the images. We measure the projection error e_{proj} as the difference between the tracker's estimated rotation of the block and the actual rotation of the rotating base. Fixing the maximum acceptable error $err_{thr} = 25$ deg, the failure condition is reached when $e_{proj} < e_{thr}$. For each run, we report the percentage of the total time (60 s) for which the tracker follows the block without failures, including the target block's tower level. Since this test's ultimate goal is to highlight the tracker's robustness, we keep the same threshold value used in the game. The trials are performed with two different angular velocities, $\omega_1 = 2.5$ deg/s and $\omega_2 = 8.3$ deg/s for the same target.

The results in Table 2 suggest that ViSP [66] is a scalable library that can be further optimized according to the requirements of our task. Our tracking system not only overrides the point-to-point manual initialization leveraging the segmentation mask prediction, but it also significantly improves the tracking robustness up to 7.5 times.

Table 2. Largest tracking time comparison between the 3D CAD models generated as a single block model and our pattern-based multi-block model.

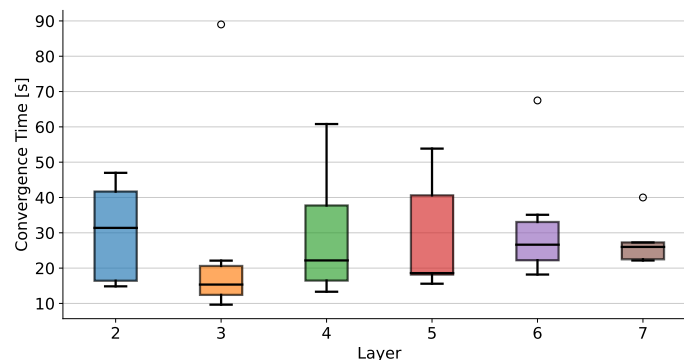
Level	Vel [deg/s]	Single [%]	Group [%]
5	ω_1	36.9	100
5	ω_2	17.0	100
6	ω_1	13.1	100
6	ω_2	8.0	100
8	ω_1	28.9	100
8	ω_2	10.6	100
9	ω_1	19.4	100
9	ω_2	29.5	100
10	ω_1	18.4	100
10	ω_2	8.1	100
11	ω_1	24.7	100
11	ω_2	14.8	100

4.4. Visual Servo Convergence and Accuracy

The visual servo control law is tested, in terms of time and spatial error, by bringing the end-effector to the computed goal. To this end, the tracking system estimates the block's pose (i.e., position and orientation) to extract the visual features and compute the 6×1 velocity vector in the camera reference frame. Such velocities are then converted in the end-effector's reference frame through the extrinsic parameters, estimated on our custom fingertip's design. Then, the linear and angular velocities are translated into joint velocities and actuated accordingly.

4.4.1. Timing Convergence

The experiment begins with the robot in a default pose at the same height as the tower's first level. Then, a timer starts and automatically stops when the end-effector reaches the desired pose with a fixed tolerance on the visual servo error magnitude as $tolerance = 0.00002$. This test is repeated for each tower level regardless of the block configuration. In Figure 8, we report the distribution of convergence time at different tower levels. The average convergence time is roughly constant among levels, while we observe a high variance between different observations. This is caused by the fact that most of the convergence time is spent reaching the desired orientation rather than the desired position. The oscillations in the tracker estimate are higher when the camera is close to the tower, and the 3D block model degenerates to a plane face. The tracker is set to tolerate up to 25 deg of mean reprojection error before failing, so the robot performs many corrections to the orientation, using all its joints to follow the oscillations. This sometimes generates longer convergence times. However, time is not a primary constraint in Jenga, so it is acceptable to trade convergence speed for more precise end-effector alignment.

**Figure 8.** Mean and standard deviation of convergence times for blocks on the same level, where the black line is the mean, and the dispersion of the values is shown. Circles show isolated events, in which tracker's oscillations caused longer convergence times of the robot pose.

4.4.2. Spatial Accuracy

The second part of the test aims to assess the manipulator's accuracy in pushing blocks. It consists in manually stopping the robot as soon as the end-effector reaches the target and measuring the distance between the contact point and the center of the block. Defining err_x and err_y as horizontal and vertical errors, we perform a statistical analysis of the system precision. The results depicted in Figure 9 are obtained from 21 trials on different blocks. The results demonstrate the accuracy and repeatability of our system, as a mean error below 0.2 mm and a standard deviation below 1 mm are compatible with the accuracy defined in Section 3.4. Moreover, it is worth noting that the mean value for both axes is close to zero, which indicates the deep focus on estimating the extrinsic camera-to-robot and intrinsic pixel-to-mm parameters. The differences between the two axes mainly depend on the manipulator's dexterity and its mechanical tolerances.

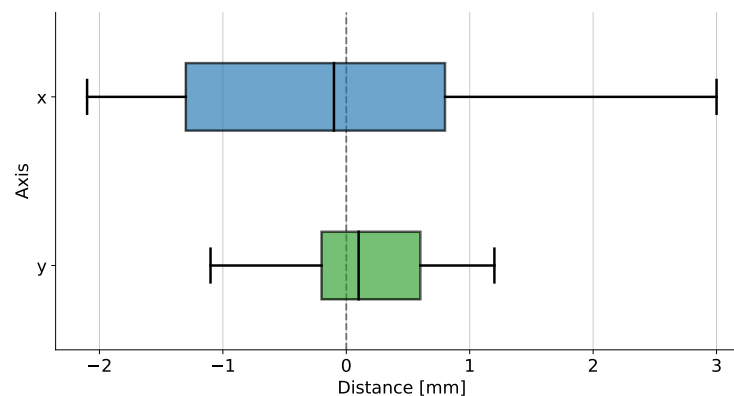


Figure 9. Mean and standard deviation of the distance between the contact point of the end-effector and the center of the selected block.

4.5. Consecutive Block Extractions

Finally, the different modules are integrated, and tests of the architecture are performed on the full system. The experiment includes 18 trials following the semi-automatic loop explained in Section 3.2. The goal is to extract as many blocks as possible without breaking the tower.

In order to properly validate our results, the chosen metric considers both correctly extracted blocks and correctly classified immobile blocks. Hence, an attempt is labeled as successful in one of the following cases:

1. A mobile block is correctly pushed without perturbing the stability of the tower;
2. The binary classification model correctly identifies the block as immobile, and the arm retracts without causing the tower to fall.

After each attempt, the policy updates the block status obtained from visual and tactile measurements during the extraction process. In particular, force sensor data are used to detect the block status, while visual measurements provide feedback on the result of the extraction primitive. The latter integrates pose estimation information and the direct kinematics of the manipulator in order to verify the complete extraction of the block from the tower. The minimization of the angular tracking error in the visual servo law allows positioning the end-effector as parallel as possible to the block by pushing it out of the tower (1) along the z-axis of the block. The robotic arm then retreats the long finger back along the same axis. This open-loop push primitive runs synchronously to the force sensor at a frequency of 20 Hz to eventually abort the execution if high reaction force values are detected. If the manipulator successfully extracts the block, the policy updates the memory buffer with the current block status and verifies the presence of other removable blocks in the subspace in order to start a new attempt. Conversely, in the case of immobile blocks, the manipulator pursues to extract the remaining blocks of the same layer. In this scenario, the policy removes the block from the list of removable blocks for future extractions.

In our setup, the operator has three interactions that make the game semi-autonomous: changing the tower's position to allow the robot to switch from higher to lower layers (or vice versa), placing the extracted blocks on top to form a new level, and aborting extractions when failures occur.

Minor failures do not stop the game, as they do not cause the tower to collapse. The operator can abort an attempt when the system fails for reasons unrelated to the extraction. In this case, the loop continues on the next iteration, and the policy is not updated. Possible minor failures are:

- The robot reaches a singularity position when the target block level is at the edge of the workspace;
- The tracking is lost as either the tower is in bad light conditions or the target block lies outside the field of view;
- The initial pose estimation is not precise enough because the segmentation model does not detect all the corners of the block correctly.

To minimize these failures and prevent the perturbation of the tower's stability, we make some conservative choices. Different force thresholds are assigned depending on the game's phase, i.e., which subspace of blocks is currently tackled. This way, the policy addresses the first subspace with an aggressive force threshold of 0.32 N. After all the first subspace levels have been tested, the policy addresses the second subspace with a cautious threshold of 0.18 N. The effect of adding the extracted blocks on top is twofold: their weight contributes to changing the friction between the blocks and shifting the center of gravity of the tower. Furthermore, each new level is considered in the policy for additional attempts. This way, we obtain a minimum of 42 attempts (3 per level) and 14 block extractions (1 per level), plus 3 additional attempts and 1 extraction for every newly formed level.

The results are reported in Figure 10 with details on the distribution of complete extractions, unmovable blocks correctly classified, and attempts ended with an error. Experiments show that 80% of the extraction attempts are successful. More specifically, 45% of the attempts lead to extracting the block correctly, while 35% of them find an unmovable block. The errors (20%) are mainly caused by the system losing the tracked block or by an imprecise initial pose estimation. Each time an error occurs, the current extraction attempt is aborted, and a new one is started.

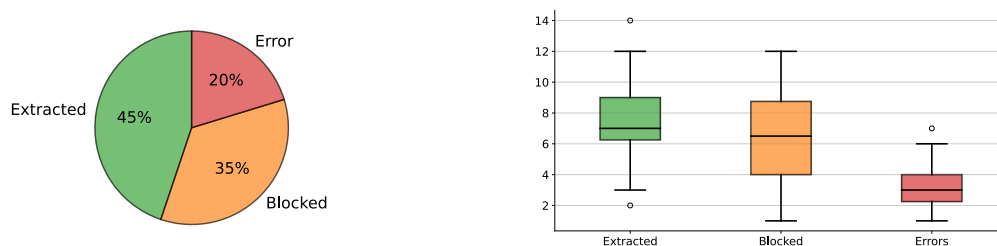


Figure 10. The plot on the left illustrates the average number of extracted blocks (green), the blocks correctly classified as unmovable (orange), and errors among 18 trials (red). On the right, the statistical distribution of the outcomes is plotted. Our system can perform successful extraction or correctly identify unmovable blocks in 80% of the attempts. The highest score achieved is 14 extracted blocks, accumulating more error in the long game, while the lowest score is only 2 extractions (white circles).

Each experiment ends when the tower falls. In our experiments, the fall occurred after 13.8 correct attempts on average (7.5 correct extractions and 6.3 detected unmovable blocks, respectively). In most experiments, the fall was caused by the increasing instability of the tower after multiple extractions. However, a few experiments ended earlier because of poor tracker performance in providing the position and orientation during the push movement or incorrect detection of mobile blocks (two experiments ended after only 5 extraction attempts). The highest registered score counts 14 successful extractions and 11 detected unmovable blocks. The results are comparable to those of [28], which scored 14 consecutive extractions using a 6-axis force sensor while being much higher than the 5 extractions

of [27], which also had inexpensive equipment. The score is far behind both the 20 and 29 of, respectively [19], more extensive architecture and artificial intelligence, and [17], more complex hardware and simpler Jenga game setup.

5. Conclusions

In this paper, we propose a complete system to play the challenging game of Jenga with cost-efficient robotic hardware. The contribution of this work goes beyond the Jenga game, proposing an advanced, adaptable solution for accurate manipulator control in delicate robotic tasks. We demonstrate that a visual-based approach for perception and control can provide the robot with significant benefits in terms of scene understanding and control accuracy.

Differently from previous works, the main components of our system are a deep instance segmentation neural network used to identify each block of the Jenga tower and a visual tracking and control pipeline to continuously adjust the pose of the end-effector as it approaches the block. A low-cost 1D force sensor is integrated into the system to check the removability of the target block, drastically reducing the cost and complexity of the overall system. Our extensive experimentation shows the remarkable performance of each fundamental component of the solution. After examining the accuracy and stability of the perception and control units, we evaluate the whole system by playing Jenga and reaching a maximum of 14 successive block extractions. Future works may see the advancement of the reasoning capability of the robot: reinforcement learning agents can be investigated to optimize the planning policy for the game. Moreover, a visual-based sensorimotor agent could eventually replace the controller by directly mapping segmented and depth images to velocity commands for the end-effector.

Author Contributions: Conceptualization, M.C.; methodology, L.M., G.P., M.M., S.A. and F.S.; software, L.M. and G.P.; validation, L.M. and G.P.; formal analysis, M.M., S.A. and F.S.; investigation, L.M., M.M., S.A. and F.S.; data curation, L.M. and G.P.; writing—original draft preparation, L.M., G.P., M.M., S.A. and F.S.; writing—review and editing, M.M., S.A. and F.S.; visualization, L.M., G.P., M.M. and S.A.; supervision, M.C.; project administration, M.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: This work was developed with the contribution of the Politecnico di Torino Interdepartmental Centre for Service Robotics (PIC4SeR) <https://pic4ser.polito.it/> (visited on 15 November 2022) and SmartData@Polito <https://smartdata.polito.it/> (visited on 15 November 2022).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Sun, X.; Zhu, X.; Wang, P.; Chen, H. A review of robot control with visual servoing. In Proceedings of the 2018 IEEE 8th Annual International Conference on CYBER Technology in Automation, Control, and Intelligent Systems (CYBER), Tianjin, China, 19–23 July 2018; pp. 116–121.
2. Zhao, Z.Q.; Zheng, P.; Xu, S.T.; Wu, X. Object detection with deep learning: A review. *IEEE Trans. Neural Netw. Learn. Syst.* **2019**, *30*, 3212–3232. [[CrossRef](#)] [[PubMed](#)]
3. Zhang, X.; Chen, Z.; Wu, Q.J.; Cai, L.; Lu, D.; Li, X. Fast semantic segmentation for scene perception. *IEEE Trans. Ind. Inform.* **2018**, *15*, 1183–1192. [[CrossRef](#)]
4. Martini, M.; Cerrato, S.; Salvetti, F.; Angarano, S.; Chiaberge, M. Position-Agnostic Autonomous Navigation in Vineyards with Deep Reinforcement Learning. In Proceedings of the 2022 IEEE 18th International Conference on Automation Science and Engineering (CASE), Mexico City, Mexico, 20–24 August 2022; pp. 477–484.
5. Salvetti, F.; Angarano, S.; Martini, M.; Cerrato, S.; Chiaberge, M. Waypoint Generation in Row-based Crops with Deep Learning and Contrastive Clustering. *arXiv* **2022**, arXiv:2206.11623.

6. Bac, C.W.; van Henten, E.J.; Hemming, J.; Edan, Y. Harvesting robots for high-value crops: State-of-the-art review and challenges ahead. *J. Field Robot.* **2014**, *31*, 888–911. [CrossRef]
7. Berenstein, R.; Shahar, O.B.; Shapiro, A.; Edan, Y. Grape clusters and foliage detection algorithms for autonomous selective vineyard sprayer. *Intell. Serv. Robot.* **2010**, *3*, 233–243. [CrossRef]
8. Kletz, S.; Schoeffmann, K.; Benois-Pineau, J.; Husslein, H. Identifying surgical instruments in laparoscopy using deep learning instance segmentation. In Proceedings of the 2019 International Conference on Content-Based Multimedia Indexing (CBMI), Dublin, Ireland, 4–6 September 2019; pp. 1–6.
9. Hasan, S.K.; Linte, C.A. U-NetPlus: A modified encoder-decoder U-Net architecture for semantic and instance segmentation of surgical instruments from laparoscopic images. In Proceedings of the 2019 41st Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), Berlin, Germany, 23–27 July 2019; pp. 7205–7211.
10. Chen, X.; Guhl, J. Industrial robot control with object recognition based on deep learning. *Procedia CIRP* **2018**, *76*, 149–154. [CrossRef]
11. Domae, Y. Recent trends in the research of industrial robots and future outlook. *J. Robot. Mechatronics* **2019**, *31*, 57–62. [CrossRef]
12. Juel, W.K.; Haarslev, F.; Ramirez, E.R.; Marchetti, E.; Fischer, K.; Shaikh, D.; Manoonpong, P.; Hauch, C.; Bodenhagen, L.; Krüger, N. Smooth robot: Design for a novel modular welfare robot. *J. Intell. Robot. Syst.* **2020**, *98*, 19–37. [CrossRef]
13. Eirale, A.; Martini, M.; Tagliavini, L.; Gandini, D.; Chiaberge, M.; Quaglia, G. Marvin: An Innovative Omni-Directional Robotic Assistant for Domestic Environments. *Sensors* **2022**, *22*, 5261. [CrossRef]
14. Yu, X.; He, W.; Li, Q.; Li, Y.; Li, B. Human-robot co-carrying using visual and force sensing. *IEEE Trans. Ind. Electron.* **2020**, *68*, 8657–8666. [CrossRef]
15. Goldau, F.F.; Shastha, T.K.; Kyrarini, M.; Gräser, A. Autonomous multi-sensory robotic assistant for a drinking task. In Proceedings of the 2019 IEEE 16th International Conference on Rehabilitation Robotics (ICORR), Toronto, ON, Canada, 24–28 June 2019; pp. 210–216.
16. Dong, J.; Cong, Y.; Sun, G.; Zhang, T. Lifelong robotic visual-tactile perception learning. *Pattern Recognit.* **2022**, *121*, 108176. [CrossRef]
17. Kroger, T.; Finkemeyer, B.; Winkelbach, S.; Eble, L.O.; Molkenstruck, S.; Wahl, F.M. A manipulator plays Jenga. *IEEE Robot. Autom. Mag.* **2008**, *15*, 79–84. [CrossRef]
18. Yoshikawa, T.; Shinoda, H.; Sugiyama, S.; Koeda, M. Jenga game by a manipulator with multiarticulated fingers. In Proceedings of the 2011 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM), Budapest, Hungary, 3–7 July 2011; pp. 960–965.
19. Fazeli, N.; Oller, M.; Wu, J.; Wu, Z.; Tenenbaum, J.B.; Rodriguez, A. See, feel, act: Hierarchical learning for complex manipulation skills with multisensory fusion. *Sci. Robot.* **2019**, *4*, eaav3123. [CrossRef] [PubMed]
20. Bauza, S.; Castillo, J.; Nanz, A.; Kambalur, B. Deep Q-Learning Applied to a Jenga Playing Robot. *Preprint ResearchGate* 2017 Available online: https://www.researchgate.net/publication/336778754_Deep_Q-Learning_Applied_to_a_Jenga_Playing_Robot (accessed on 5 November 2022).
21. Justesen, N.; Bontrager, P.; Togelius, J.; Risi, S. Deep learning for video game playing. *IEEE Trans. Games* **2019**, *12*, 1–20. [CrossRef]
22. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; Riedmiller, M. Playing atari with deep reinforcement learning. *arXiv* **2013**, arXiv:1312.5602.
23. Silver, D.; Hubert, T.; Schrittwieser, J.; Antonoglou, I.; Lai, M.; Guez, A.; Lanctot, M.; Sifre, L.; Kumaran, D.; Graepel, T.; et al. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv* **2017**, arXiv:1712.01815.
24. Caccianiga, G.; Mariani, A.; de Paratesi, C.G.; Menciassi, A.; De Momi, E. Multi-sensory guidance and feedback for simulation-based training in robot assisted surgery: A preliminary comparison of visual, haptic, and visuo-haptic. *IEEE Robot. Autom. Lett.* **2021**, *6*, 3801–3808. [CrossRef]
25. Zheng, C.; Chen, P.; Pang, J.; Yang, X.; Chen, C.; Tu, S.; Xue, Y. A mango picking vision algorithm on instance segmentation and key point detection from RGB images in an open orchard. *Biosyst. Eng.* **2021**, *206*, 32–54. [CrossRef]
26. Zheng, W.; Xie, Y.; Zhang, B.; Zhou, J.; Zhang, J. Dexterous robotic grasping of delicate fruits aided with a multi-sensory e-glove and manual grasping analysis for damage-free manipulation. *Comput. Electron. Agric.* **2021**, *190*, 106472. [CrossRef]
27. Wang, J.; Rogers, P.; Parker, L.; Brooks, D.; Stilman, M. Robot Jenga: Autonomous and strategic block extraction. In Proceedings of the 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, St. Louis, MO, USA, 10–15 October 2009; pp. 5248–5253.
28. Kimura, S.; Watanabe, T.; Aiyama, Y. Force based manipulation of Jenga blocks. In Proceedings of the 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems, Taipei, Taiwan, 18–22 October 2010; pp. 4287–4292.
29. LeCun, Y.; Bengio, Y.; Hinton, G. Deep learning. *Nature* **2015**, *521*, 436–444. [CrossRef]
30. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. Imagenet classification with deep convolutional neural networks. *Commun. ACM* **2017**, *60*, 84–90. [CrossRef]
31. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.
32. Hu, J.; Shen, L.; Sun, G. Squeeze-and-excitation networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 7132–7141.

33. Woo, S.; Park, J.; Lee, J.Y.; Kweon, I.S. Cbam: Convolutional block attention module. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 3–19.
34. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention is all you need. In Proceedings of the Advances in Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017; Volume 30.
35. Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; et al. An Image is Worth 16 × 16 Words: Transformers for Image Recognition at Scale. In Proceedings of the International Conference on Learning Representations, Virtual Event, 3–7 May 2021.
36. Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.Y.; Berg, A.C. Ssd: Single shot multibox detector. In Proceedings of the European Conference on Computer Vision, Amsterdam, The Netherlands, 11–14 October 2016; pp. 21–37.
37. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You only look once: Unified, real-time object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 779–788.
38. Redmon, J.; Farhadi, A. YOLO9000: Better, faster, stronger. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 7263–7271.
39. Redmon, J.; Farhadi, A. Yolov3: An incremental improvement. *arXiv* **2018**, arXiv:1804.02767.
40. Long, J.; Shelhamer, E.; Darrell, T. Fully convolutional networks for semantic segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015; pp. 3431–3440.
41. Ronneberger, O.; Fischer, P.; Brox, T. U-net: Convolutional networks for biomedical image segmentation. In Proceedings of the International Conference on Medical Image Computing and Computer-Assisted Intervention, Munich, Germany, 5–9 October 2015; pp. 234–241.
42. He, K.; Gkioxari, G.; Dollár, P.; Girshick, R. Mask r-cnn. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 2961–2969.
43. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster r-cnn: Towards real-time object detection with region proposal networks. In Proceedings of the Advances in Neural Information Processing Systems, Montreal, QC, Canada, 7–12 December 2015; Volume 28.
44. Li, Y.; Qi, H.; Dai, J.; Ji, X.; Wei, Y. Fully convolutional instance-aware semantic segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 2359–2367.
45. Chen, L.C.; Hermans, A.; Papandreou, G.; Schroff, F.; Wang, P.; Adam, H. Masklab: Instance segmentation by refining object detection with semantic and direction features. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018; pp. 4013–4022.
46. Kirillov, A.; Levinkov, E.; Andres, B.; Savchynskyy, B.; Rother, C. Instancecut: From edges to instances with multicut. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 5008–5017.
47. Liang, X.; Lin, L.; Wei, Y.; Shen, X.; Yang, J.; Yan, S. Proposal-free network for instance-level object segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *40*, 2978–2991. [[CrossRef](#)] [[PubMed](#)]
48. Newell, A.; Huang, Z.; Deng, J. Associative embedding: End-to-end learning for joint detection and grouping. In Proceedings of the Advances in Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017; Volume 30.
49. Bolya, D.; Zhou, C.; Xiao, F.; Lee, Y.J. Yolact: Real-time instance segmentation. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Seoul, Korea, 27 October–2 November 2019; pp. 9157–9166.
50. Bolya, D.; Zhou, C.; Xiao, F.; Lee, Y.J. YOLACT++: Better Real-time Instance Segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.* **2020**, *44*, 1108–1121. [[CrossRef](#)] [[PubMed](#)]
51. Evjemo, L.D.; Gjerstad, T.; Grøtli, E.I.; Sziebig, G. Trends in smart manufacturing: Role of humans and industrial robots in smart factories. *Curr. Robot. Rep.* **2020**, *1*, 35–41. [[CrossRef](#)]
52. Hill, J.; Park, W.T. Real Time Control of a Robot with a Mobile Camera. In Proceedings of the 9th ISIR, Washington, DC, USA, 13–15 March 1979.
53. Azizian, M.; Khoshnam, M.; Najmaei, N.; Patel, R.V. Visual servoing in medical robotics: A survey. Part I: Endoscopic and direct vision imaging-techniques and applications: Visual servoing in medical robotics: A survey (Part I). *Int. J. Med. Robot. Comput. Assist. Surg.* **2014**, *10*, 263–274. [[CrossRef](#)] [[PubMed](#)]
54. Dewi, T.; Risma, P.; Oktarina, Y.; Muslimin, S. Visual Servoing Design and Control for Agriculture Robot; a Review. In Proceedings of the 2018 International Conference on Electrical Engineering and Computer Science (ICECOS), Malang, Indonesia, 2–4 October 2018; pp. 57–62.
55. Staub, C.; Osa, T.; Knoll, A.; Bauernschmitt, R. Automation of tissue piercing using circular needles and vision guidance for computer aided laparoscopic surgery. In Proceedings of the 2010 IEEE International Conference on Robotics and Automation, Anchorage, AK, USA, 3–7 May 2010; pp. 4585–4590.
56. Voros, S.; Haber, G.P.; Menuet, J.F.; Long, J.A.; Cinquin, P. ViKY Robotic Scope Holder: Initial Clinical Experience and Preliminary Results Using Instrument Tracking. *IEEE/ASME Trans. Mechatronics* **2010**, *15*, 879–886. [[CrossRef](#)]
57. Krupa, A.; Gangloff, J.; Doignon, C.; de Mathelin, M.; Morel, G.; Leroy, J.; Soler, L.; Marescaux, J. Autonomous 3D positioning of surgical instruments in robotized laparoscopic surgery using visual servoing. *IEEE Trans. Robot. Autom.* **2003**, *19*, 842–853. [[CrossRef](#)]
58. Hutchinson, S.; Hager, G.; Corke, P. A tutorial on visual servo control. *IEEE Trans. Robot. Autom.* **1996**, *12*, 651–670. [[CrossRef](#)]
59. Barth, R.; Hemming, J.; van Henten, E.J. Design of an eye-in-hand sensing and servo control framework for harvesting robotics in dense vegetation. *Biosyst. Eng.* **2016**, *146*, 71–84. [[CrossRef](#)]

60. Mehta, S.; MacKunis, W.; Burks, T. Robust visual servo control in the presence of fruit motion for robotic citrus harvesting. *Comput. Electron. Agric.* **2016**, *123*, 362–375. [[CrossRef](#)]
61. Lippiello, V.; Cacace, J.; Santamaria-Navarro, A.; Andrade-Cetto, J.; Trujillo, M.Á.; Rodríguez Esteves, Y.R.; Viguria, A. Hybrid Visual Servoing With Hierarchical Task Composition for Aerial Manipulation. *IEEE Robot. Autom. Lett.* **2016**, *1*, 259–266. [[CrossRef](#)]
62. Comport, A.; Marchand, E.; Pressigout, M.; Chaumette, F. Real-time markerless tracking for augmented reality: The virtual visual servoing framework. *IEEE Trans. Vis. Comput. Graph.* **2006**, *12*, 615–628. [[CrossRef](#)] [[PubMed](#)]
63. Lin, T.Y.; Dollár, P.; Girshick, R.; He, K.; Hariharan, B.; Belongie, S. Feature pyramid networks for object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 2117–2125.
64. Collins, T.; Bartoli, A. Infinitesimal plane-based pose estimation. *Int. J. Comput. Vis.* **2014**, *109*, 252–286. [[CrossRef](#)]
65. Siciliano, B.; Sciavicco, L.; Villani, L.; Oriolo, G. *Robotics: Modelling, Planning and Control*; Advanced textbooks in control and signal processing; Springer: London, UK, 2010.
66. Marchand, E.; Spindler, F.; Chaumette, F. ViSP for visual servoing: A generic software platform with a wide class of robot control skills. *IEEE Robot. I Autom. Mag.* **2005**, *12*, 40–52. [[CrossRef](#)]
67. Denninger, M.; Sundermeyer, M.; Winkelbauer, D.; Olefir, D.; Hodan, T.; Zidan, Y.; Elbadrawy, M.; Knauer, M.; Katam, H.; Lodhi, A. BlenderProc: Reducing the Reality Gap with Photorealistic Rendering. In Proceedings of the Robotics: Science and Systems (RSS), Virtual Event, 12–16 July 2020.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.