applied sciences

MDPI

Article

Evaluation of Vision-Based Hand Tool Tracking Methods for Quality Assessment and Training in Human-Centered Industry 4.0

Irio De Feudis [1,2,†], Domenico Buongiorno [1,2,†], Stefano Grossi [3], Gianluca Losito [1], Antonio Brunetti [1,2], Nicola Longo [2,3], Giovanni Di Stefano [3] and Vitoantonio Bevilacqua [1,2,*]

1 Department of Electrical and Information Engineering (DEI), Polytechnic University of Bari, 70126 Bari, BA, Italy; irio.defeudis@poliba.it (I.D.F.); domenico.buongiorno@poliba.it (D.B.); g.losito3@studenti.poliba.it (G.L.); antonio.brunetti@poliba.it (A.B.)
2 Apulian Bioengineering s.r.l., Via delle Violette 14, 70026 Modugno, BA, Italy
3 Comau S.p.a., Via Rivalta 30, 10095 Grugliasco, TO, Italy; stefano.grossi2@external.comau.com (S.G.); nicola.longo@comau.com (N.L.); giovanni.distefano@comau.com (G.D.S.)
* Correspondence: vitoantonio.bevilacqua@poliba.it
† These authors contributed equally to this work.

Abstract: Smart industrial workstations for the training and evaluation of workers are an innovative approach to face the problems of manufacturing quality assessment and fast training. However, such products do not implement algorithms that are able to accurately track the pose of a hand tool that might also be partially occluded by the operator's hands. In the best case, the already proposed systems roughly track the position of the operator's hand center assuming that a certain task has been performed if the hand center position is close enough to a specified area. The problem of the pose estimation of 3D objects, including the hand tool, is an open and debated problem. The methods that lead to high accuracies are time consuming and require a 3D model of the object to detect, which is why they cannot be adopted for a real-time training system. The rise in deep learning has stimulated the search for better-performing vision-based solutions. Nevertheless, the problem of hand tool pose estimation for assembly and training procedures appears to not have been extensively investigated. In this study, four different vision-based methods based, respectively, on ArUco markers, OpenPose, Azure Kinect Body Tracking and the YOLO network have been proposed in order to estimate the position of a specific point of interest of the tool that has to be tracked in real-time during an assembly or maintenance procedure. The proposed approaches have been tested on a real scenario with four users handling a power drill simulating three different conditions during an assembly procedure. The performance of the methods has been evaluated and compared with the HTC Vive tracking system as a benchmark. Then, the advantages and drawbacks in terms of the accuracy and invasiveness of the method have been discussed. The authors can state that OpenPose is the most robust proposal arising from the study. The authors will investigate the OpenPose performance in more depth in further studies. The framework appears to be very interesting regarding its integration into a smart workstation for quality assessment and training.

Keywords: Industry 4.0; manufacturing; hand tool tracking; artificial vision; deep learning; tracker; quality; training

check for updates

Citation: De Feudis, I.; Buongiorno, D.; Grossi, S.; Losito, G.; Brunetti, A.; Longo, N.; Di Stefano, G.; Bevilacqua, V. Evaluation of Vision-Based Hand Tool Tracking Methods for Quality Assessment and Training in Human-Centred Industry 4.0. *Appl. Sci.* **2022**, *12*, 1796. https://doi.org/10.3390/app12041796

Academic Editor: Doriana Marilena D'Addona

Received: 22 January 2022
Accepted: 30 January 2022
Published: 9 February 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/).

1. Introduction

The Industry 4.0 revolution has affirmed the centrality of the human operator, proposing a new way to automate production processes based on human and robotic expertise synergy. The impact of new technologies, such as collaborative robots (co-bots), virtual reality (VR) and augmented reality (AR) [1], wearable devices, Internet of Things (IoT) and artificial vision [2,3], allows the companies to streamline their processes, ensuring better quality and enhancing the flexibility of their production potential [4,5]. From this

perspective, the human-centered approach has once again been reaffirmed in the automotive sector in the era of electric vehicles [6]; as an example, the technological processes of battery pack assembly require several manual procedures, especially in the completion of electrical connections.

The continuous evolution of production processes, not only in the automotive industry [7] but in the whole world of automation, together with the problem of employee turnover, has led to a progressive and ceaseless need to train novel operators on new techniques and to assess the quality of their job. The assembly and maintenance of industrial assets are complex tasks that require a large number of training hours in classrooms and hands-on sessions. Furthermore, especially in times of restrictions related to the pandemic, only a few trainees can access training on a physical asset at the same time, and their availability is always a trade-off with production line efficiency. Training all the operators of a plant on a piece of new equipment in a reasonable lead time is a huge challenge.

In addition to the personnel training, one of the key factors of success in the new industry is the traceability of manual operations to ensure product quality [8]. Indeed, in most of the manufacturing processes, it is important to collect data from the manual assembly and maintenance interventions and to store these data in a centralized data management system for subsequent analysis of the performance of the plants [9]. In this scenario, the companies that can accurately understand workers' behaviors and evaluate their performance in real-time will outperform their competitors.

Smart industrial workstations for training and evaluating the performance of the workers are drawing attention as an innovative approach to facing the problem. These systems are designed to guide a non-experienced operator in achieving the same level of precision and performance as the expert [10,11]. In addition, they ensure that procedures are carried out in the right way while collecting anonymous real-time data that can be used to verify technical parameters and to improve the working efficiency where needed.

The market proposes some commercial solutions for virtual guidance, such as Bosch's Active Assist system (https://www.boschrexroth.com/en/xc/products/product-groups/assembly-technology/news/activeassist-assistance-system/index/, accessed on 20 January 2022), Arkite HIM (https://arkite.com/product/, accessed on 20 January 2022), Rhinoassembly Light Guide System (https://www.rhinoassembly.com/en/catalog/product/-Light-Guide--LGS--LGS/, accessed on 20 January 2022) and Vir.GIL (https://www.comau.com/it/competencies/digital-initiatives/technologies/vir-gil/, accessed on 20 January 2022) (Figure 1). They are powerful and flexible systems that are able to guide the operator during the training of a new assembly, inspection or maintenance procedure by means of digital information projected on a workbench or directly on the asset to be manipulated. However, such innovative products do not implement algorithms that are able to accurately track the pose of a hand tool that might also be partially occluded by the operator's hands. In fact, in the best case, these industrial solutions roughly track the position of the hand center and assume that a certain task has been performed if the hand center position is enough close to a specified area. This can clearly lead to a rough performance evaluation and the learning of bad habits. In order to make a difference, it is fundamental to track, with high accuracy, the pose of the tools the worker handles during the learning session of the procedure [12]. As an example, if a worker is required to tighten several screws in sequence, an evaluation of whether the screws were tightened following the correct sequence without missing any screws might be requested.

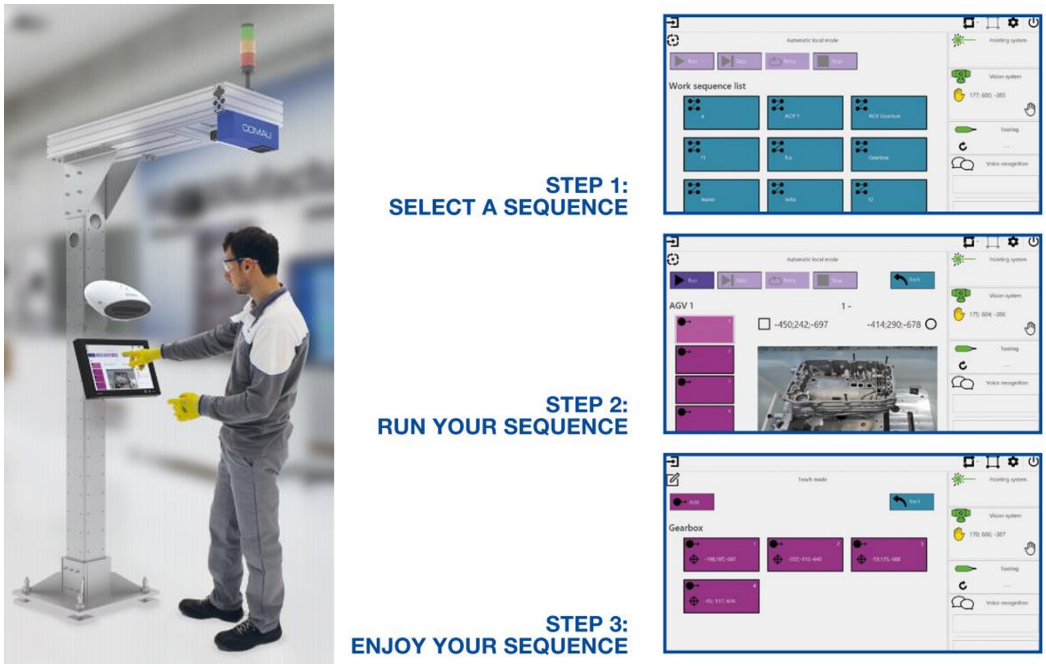

Figure 1. Vir.GIL virtual guidance system: (**Left**) hardware setup of the system; (**Right**) application UI workflow for starting a new procedure.

The problem of the pose estimation of 3D objects, including the hand tool for allowing autonomous manipulation [13], has been studied for years and is an open and debated problem. The approaches that allow for high accuracies rely on a 3D model of the object that has to be detected within a 3D point cloud [14]. However, such methods are time consuming and may not be applicable due to the unavailability of the 3D model [14]. The unstoppable rise of deep learning in recent years has pushed the pose estimation research out of the the boundaries of classical computer vision techniques, and new approaches based on deep neural networks have been exploited [15]. Novel methods require minimal human intervention, improve the performance of 3D data-based approaches [16,17] and, in some cases, avoid the usage of 3D models for estimating the pose of an object [18,19]. To the best of the authors' knowledge, even if the problem has also been extended to the general situation of occluded objects, hand tool pose estimation and tracking in the industrial environment during assembly and maintenance procedures has not been extensively investigated.

Detecting and tracking a tool handled by an operator performing an assembly or maintenance procedure is not easy, mainly because of the tool's occlusion due to the operator's hands. In the authors' opinion, the problem of hand tool pose estimation can be solved with a direct or indirect approach. In a direct approach, the tool is detected and then tracked using some robust features that are visible even if it is occluded. In an indirect approach, body joints of the operator and their hands are detected and tracked; the tool pose is derived assuming a unique handling pose of the tool. In order to investigate both paradigms, four artificial vision-based systems have been design, implemented and compared. Each developed system has been evaluated with a set of experiments replicating real industrial scenarios, and their performances have been compared to analyze the pros and cons according to the task properties. Even though, in this work, the authors proposed general methodologies, the developed systems have been tested with a specific use case considering the estimation of the 3D position of a cordless power drill.

This study has the ambition to identify the best method(s) to integrate in the next cutting-edge workstations for training and assessment in Industry 4.0. Such systems will be designed with the following objectives:

- Training an operator on assembly and maintenance procedures with a recorded sequence of actions;

- Tracking an operator activity to validate each manual operation and certify the quality of the job;
- Detecting risky actions and behaviors in time and alerting the operator;
- Ergonomics monitoring during the procedure for always proposing to the operator the less stressful posture to perform the operations.

The main constraints of these solutions should be:

- Real-time execution;
- High accuracy.

The real-time execution is a key factor of a virtual guidance system because it has to be ready to warn the operator as soon as possible when their safety is compromised, when ergonomics guidelines are not respected [3,20] or when the process is going to be completed in the wrong way. On the other hand, a high level of accuracy is required to control the operator's movement in the assembling process to avoid errors.

The paper is organized as follows: in Section 2, the authors explain the technical characteristics of the selected methods, how they have been implemented, the experiments designed to evaluate the relative performance, the metrics computed and the statistical analysis conducted for the quantitative comparisons; Section 3 reports the results of the metrics computed on the data acquired during the experiment and the statistical analysis; in Section 4, the results are extensively discussed; Section 5 reports the conclusions of this paper and the future scope.

2. Materials and Methods

2.1. Systems for Hand Tool Tracking

In this work, four different systems based on either open-source or commercial solutions have been developed and compared:

1. The first system is a marker-based solution using the ArUco markers [21,22];
2. The second system is based on a deep learning model engineered for 2D detection problems and is called YOLO v4 [23,24];
3. The third system is based on the Azure Kinect Body Tracking (AKBT) service [25,26]
4. The fourth system considers the use of the OpenPose [27–30] library.

Each system has been designed to estimate a single interesting point of a hand-held tool, even though three out of four systems have the intrinsic capability of estimating the entire pose of the tool. It is worth citing that all four systems are based on RGBD camera, except for the method based on Aruco, which does not need the depth information. For this reason, the new Kinect Azure camera has been used either for acquiring RGBD data or estimating the human skeleton configuration with the Microsoft SDK "Azure Kinect Body Tracking".

2.1.1. ArUco-Based SYSTEM

The system based on ArUco marker considers the possibility of applying a 2D planar marker on the hand tool that must be always visible by a RGB 2D camera. The main benefit of this approach is that a single marker provides enough correspondences to obtain the camera pose and, from that, a tracked object pose can be derived. Once the position of the tracked tool's point within the marker reference frame is known, the path of such a point can be reconstructed by the marker frame pose. Such an approach is the most invasive one since it considers the introduction of a new object, i.e., the marker, within the scenario. The strength of this approach is that it can determine a robust 3D pose estimation using a simple 2D camera and some printed markers, keeping computational costs low and therefore saving hardware resources for other essential tasks. On the other hand, it is an invasive way to track an object because markers have to be installed on it. Furthermore, if the marker is not visible, it is not possible to detect the object and, as a consequence, to establish the correct pose.

ArUco [21,22] is a popular opensource library for detection of square fiducial markers and camera pose estimation mostly used in augmented reality applications. An ArUco

marker is a synthetic square marker composed of a wide black border and an inner binary matrix that determines its identifier (id). The black border facilitates its fast detection in the image, and the binary codification allows for its identification and the application of error detection and correction techniques. The marker size determines the size of the internal matrix. For instance, a marker size of 4×4 is composed of 16 bits. The ArUco decoding algorithm can locate, decode and estimate the pose in realtime of any ArUco markers in the camera's field of view, as shown in Figure 2. It is based on the knowledge of the matrix encoded in the square. Multiple matrices are encoded in a group of markers, identified in dictionaries. It is possible to choose among several predefined dictionaries or by generating one yourself.

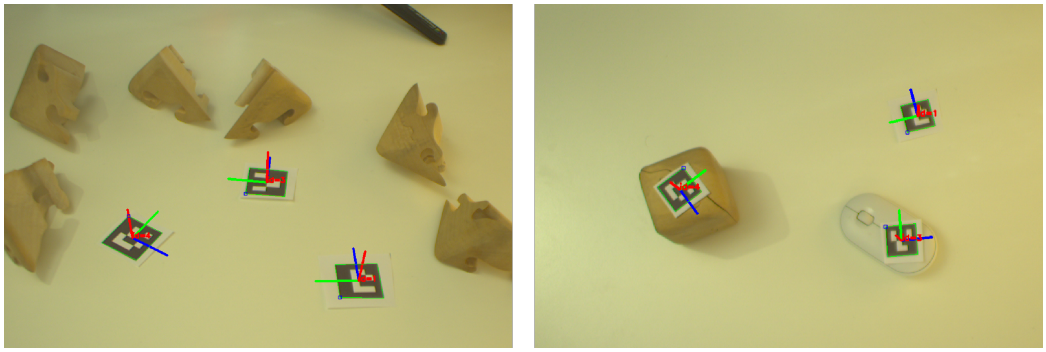

Figure 2. Example of ArUco markers pose estimation: (**Left**) three markers placed on a desk in different poses; (**Right**) three markers placed on three different objects.

For the sake of simplicity, in this study, a wooden support for the power drill was realized and three ArUco square markers with 5 cm side were stuck on it. Usage of a group of markers was preferred to a single marker in order to provide a more robust pose estimation: a single marker being visible but not recognized quickly in the scene, or not recognized at all because of an occlusion, could occur. When more markers of the group are detected in the same frame, only one marker's pose needs to be chosen as reference for the power drill chuck pose estimation. The area in pixels of a marker was used as selection criteria for the best marker because it is a matter of fact that AruCo algorithms have better performance the bigger the detected marker appears in the frame.

2.1.2. System Based on Deep Detection Model (YOLO)

The system that uses a deep neural network, i.e., YOLO, is based on two subsequent steps: (1) a deep neural network is trained to localize within a 2D RGB image the specific tool's area (or point) that has to be tracked, and (2) the 3D position of the tracked tool's point is estimated by computing the spacial centroid of the point cloud underlying the ROI found in the previous step.

You Only Look Once (YOLO) [23,24] is a family of convolutional neural networks (CNN) that achieve near state-of-the-art results with a single end-to-end model that can perform object detection in real-time. Compared to the approach taken by previous object detection algorithms, YOLO proposes the use of an end-to-end neural network that makes predictions of bounding boxes and class probabilities all at once (Figure 3). Starting from YOLO's pre-trained models, it is possible to re-train the last layers to introduce new classes that are not available in the original dataset to fit the object detector capabilities to new objects. In the re-training process, it is also possible to tune settings of the net and to adjust accuracy over performance, training time and batch iterations, choosing the best weight candidate. YOLO's performance has improved over time, starting from the v1 version. The original YOLO v1 was born as the first object detection network to combine the problem of drawing bounding boxes and identifying class labels in one end-to-end differentiable network. YOLO v4 outperforms most of the other object detection models [31]

by a significant margin, keeping frame-rate high and making it the best opportunity to detect objects in real-time.

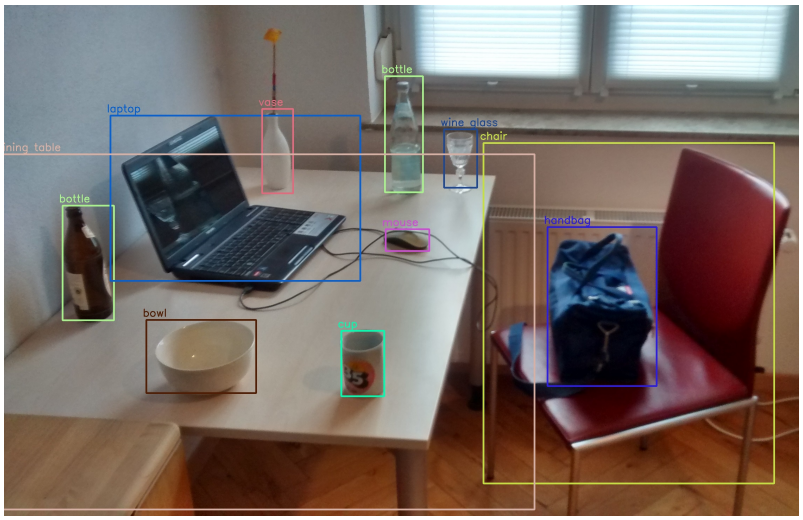

Figure 3. YOLO object detection output for office scene (from MTheiler, CC BY-SA 4.0, via Wikimedia Commons).

In this study, YOLO was implemented in its fourth version. The last layers of the model were re-trained to recognize the power drill chuck as a new class, and then the performance of the model inference was evaluated. In order to train the model, a dataset consisting of 735 images was created. The images were acquired by Azure Kinect RGB camera with the power drill in different poses, with heterogeneous light conditions and in different environments. Considering the objective of study, the model was trained with images containing only the particular power drill used for our experiment, leading the net to recognize only the chuck of this tool and not that of other power drills. Data augmentation was performed in order to improve the performance of the model using flip upside-down, flip left-right, rotation and Gaussian blur. The training process was based on transfer learning and the CNN model was initialized with the weights retrieved from the GitHub page of YOLO v4 based on darknet framework. The net was trained with images of 512×512 resolution and max batches set to 2000. The starting weights were obtained from the training on the COCO dataset, containing 80,000 images and 80 different object classes.

The trained model was able to recognize the power drill chunk in every frame of real-time acquisition, returning the pixel coordinates x and y of the top-left corner of a box that contains it, the width and the height of the box and the confidence as the probability that the object was classified correctly. In order to estimate the 3D pose of the power drill chuck, the center point (x,y) of the box returned by YOLO model was calculated and the 3D coordinates of that point in the scene were computed by using the functions of Azure Kinect SDK.

2.1.3. System Based on Azure Kinect Body Tracking

The system based on the Azure Kinect Body Tracking exploits the 3D body configuration estimation performed by the AKBD SDK of Microsoft. The system is based on the tracking of the upper link main segments to estimate a specific tool's point if the relative position of such a point, with respect to the hand reference system, is known.

The Azure Kinect from Microsoft is a cutting-edge spatial computing developer kit with sophisticated computer vision and speech models, advanced AI sensors and a range of powerful SDKs. It is equipped with several sensors in order to sense the surrounding environment; the device integrates a 12-megapixel RGB camera supplemented by 1-megapixel depth camera, a 360-degree seven-microphone array and an orientation sensor. The main modules of the SDKs are the Sensor SDK and Body Tracking SDK. The first one is designed

for the interface with sensors and for managing data provided by them, automatically handling the problem of RGB and depth camera data alignment; the Body Tracking SDK is based on a complex deep learning model that supports body segmentation, human skeleton reconstruction, human body instance recognition and body tracking in real-time. The model recognizes 31 joints of the human body, each joint with its own reference system organized in a hierarchical structure, as shown in Figure 4. The deep learning model is able to reconstruct the entire human body model, where the higher the accuracy, the better the visibility of the body; the model has a good tolerance to the occlusions only for the highest joints of the hierarchical structure.

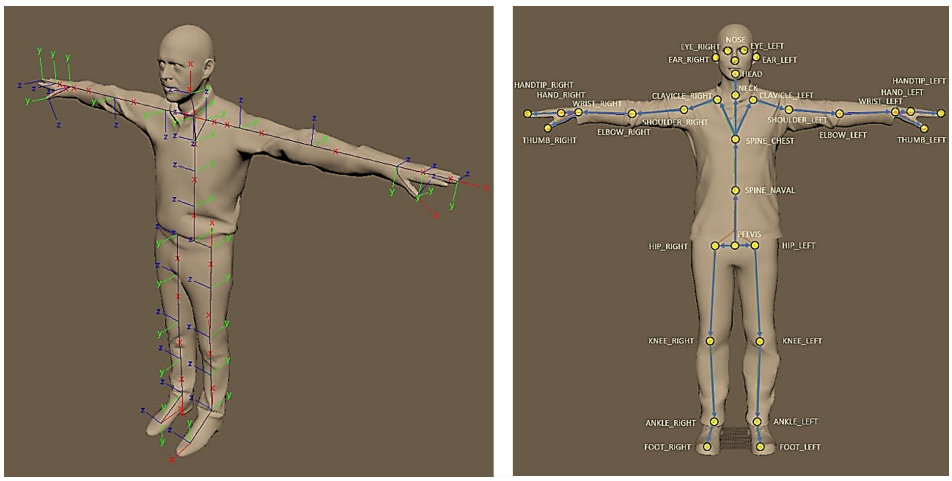

Figure 4. Azure Kinect Body Tracking detectable joints: (**Left**) joints with reference systems; (**Right**) joints' hierarchy.

Body Tracking SDK is able to detect and track all joints of a human body in the scene, returning position and orientation (in quaternion form) for each of them [25,26].

Intuitively, the kinect hand joints, i.e., n.8—left hand and n.15—right hand, should be the most useful to estimate the power drill chunk pose. Unfortunately, several tests led the authors to exclude the above-mentioned joints because experimental trials proved the poor quality of the orientation prediction. In fact, most of the time, the model predicted the power drill in the scene as part of the hand of the operator. Thus, the joints of the wrists (n.7—left and n.14 right) have been considered. The downside of this choice is that the tool pose changes relative to the wrist flexion/extension, and wrist radial–ulnar deviation movements are not considered.

2.1.4. System Based on OpenPose

This system, like the one based on the AKBT, uses the information of the operator upper limb pose to obtain the tool's point position. In detail, it relies on the 3D position of some hand's keypoints. OpenPose [27–30] can be considered as the state-of-the-art approach for real-time human pose estimation. It is the first framework that can jointly detect human body, hand, facial and foot keypoints on single images. OpenPose is a multi-stage CNN that uses a bottom-up approach to find every instance of a key point and then attempts to assemble groups of key points into skeletons of distinct humans. The deep learning OpenPose model is based on a CNN and follows a precise pipeline: in the first step, the model computes confidence maps for every body part detection; in the second part, it predicts part affinity fields (PAF) in order to associate every key point of every person in the frame; in the third step, bipartite matching is computed, so a several-graph connection is evaluated in order to choose the best performing PAF; in the last step, results are parsed and the skeleton is reconstructed. A confidence map is the 2D representation of the belief that a particular body part can be located. A single body part will be represented on a single map. Therefore, the number of maps is the same as the total number of body

parts, and is a number that depends on the dataset the model is trained on. Instead, PAF is a set of 2D vector fields that encode the location and orientation of limbs over the frame domain. The framework integrates three different trained models for body, hands and face keypoint estimation. They return, respectively, 25 keypoints for the body, 21 $\times$ 2 keypoints for hands and 70 keypoints for the face (135 keypoints).

For this study, the authors focused on OpenPose capability to track up to twenty keypoints of the hand, consisting of wrist, finger's knuckles and phalanges. The adopted configuration for the framework has the following characteristics: Body Network input size equal to 160 $\times$ 160, Hands Net input size equal to 368 $\times$ 368 and Face Net disabled. The keypoints are just recognized in the 2D frame and the output is expressed in pixel coordinates within the image. Hence, a registered 3D point cloud is used to obtain the 3D position of the keypoints.

Three out of all detected keypoints of the hand have been considered to compute the estimated hand's pose (Figure 5):

- Keypoint n.0 (wrist);
- Keypoint n.9 (middle finger knuckles);
- Keypoint n.17 (little finger knuckles).

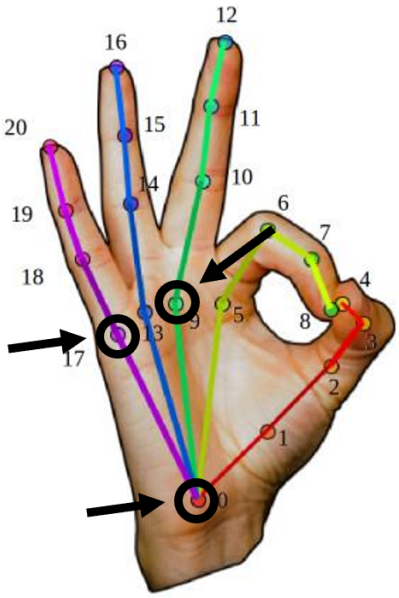

Figure 5. OpenPose keypoints for the hand, where the ones used for the calculation are highlighted. (Keypoint n.9) Middle finger knuckles; Keypoint n.0) wrist; (Keypoint n.17) little finger knuckles.

The reference frame of the operator's hand was computed as follows. First, two vectors are computed: one from the wrist point (kp_0) to the middle finger knuckle point (v_1) (Equation (1)) and one from the wrist point to the little finger knuckle point (v_2) (Equation (2)). The cross-product between v_1 and v_2 returns v_3, as shown in Equation (3); it is an orthogonal vector to v_1 and v_2 with a direction given by the right-hand rule outgoing from the back of the hand. Then, it was possible to calculate the vector whose direction is toward the thumb (v_4) as cross-product between v_1 and v_3 (Equation (4)).

$$\overline{v}_1 = [kp_{9x} - kp_{0x} \ , \ kp_{9y} - kp_{0y} \ , \ kp_{9z} - kp_{0z}] \tag{1}$$

$$\overline{v}_2 = [kp_{17x} - kp_{0x} \ , \ kp_{17y} - kp_{0y} \ , \ kp_{17z} - kp_{0z}] \tag{2}$$

$$\overline{v}_3 = \overline{v}_1 \times \overline{v}_2 \tag{3}$$

$$\overline{v}_4 = \overline{v}_1 \times \overline{v}_3 \tag{4}$$

with

$$x \equiv v_1 \ , \ y \equiv v_3 \ , \ z \equiv v_4 \tag{5}$$

2.1.5. Calibration of the Proposed Systems

All of the proposed methodologies consider the estimation of the 3D position of a specific point of interest of the hand tool. The system based on YOLO is the only one that directly estimate such a position. The other three systems are based on the knowledge of the position (that can be considered fixed) of such a point with respect to the estimated reference system. A specific calibration procedure is thus needed to acquire the position of such a point that, in this work, has been experimentally found by positioning the tool's point (while it was hand-held by the operator) in correspondence of a known position. However, other model-based techniques might be investigated.

2.2. Experimental Validation

In order to evaluate and compare the proposed systems, the authors selected a specific scenario that considered an operator holding a cordless power drill; then, its mandrel was considered as the point of interest to be tracked. As will be deeply discussed below, the pose of the power drill and the position of the mandrel were also acquired with an accurate system in order to quantitatively evaluate the performance of each system.

2.2.1. Evaluation System Setup

The experiment environment was set up with the Azure Kinect positioned at approximately 1.8 m from the ground and two meters from a wall, tilted 15 degrees downwards with respect to the horizon. The distance of the operator from the Kinect camera could be in a range between 0.80 m and 2 m, compatible with the range defined in the official documentation of the device.

In order to evaluate the performance of the four frameworks under investigation, a highly accurate tracking of the power drill chunk pose was needed as reference. The HTC Vive (Figure 6) was selected as benchmark for the experiment since the precision of its tracking technologies has been tested to be around RMS 1.5 mm, and its accuracy around RMS 1.9 mm [32–34].

The HTC Vive system is used for rendering 3D virtual reality and it is developed by HTC in partnership with Valve. The headset (Figure 6—Top-left) uses room scale tracking technology (Lighthouse, as shown in Figure 6—Middle) for virtual reality experiences that allow users to freely move around a play area, accurately tracking the position and orientation of the user's head-mounted display and controllers, reflecting all real-life movement in the VR simulation environment. The tracking is possible thanks to two infrared signals emitters called base stations (Figure 6—Top-right) and special active sensors that cover the surface of the headset, and controllers that intercept the infrared pulses can autonomously track their own position and orientation in the workspace determined by the field of view of the stations. The HTC Vive capabilities can be extended by means of small devices called Vive trackers (Figure 6—Bottom) [35] that implement Lighthouse technology too. They allow for a high degree of flexibility, making it possible to track items or body parts if correctly configured [36].

HTC Vive base stations were placed in the space between Azure Kinect, delimiting a workspace that completely contained the field of view of the Kinect cameras to ensure the operator movements were in a monitored space. An HTC Vive tracker was placed on the power drill to track its pose. The position of the chunk with respect to the Vive tracker reference frame was found using the calibration procedure described above. The power drill Vive tracker was installed by means of a wooden support (Figure 7—Right) designed for the purpose.

Figure 6. HTC Vive components: (**Top-Left**) the VR headset with a controller; (**Top-Right**) front and rear of a base station; (**Middle**) Lighthouse tracking system setup; (**Bottom**) a HTC Vive tracker with details scheme.

Figure 7. (**Left**) The power drill used for the experiment. (**Right**) The wooden support mounted on the power drill with ArUco markers and HTC Vive tracker installed.

It is worth noting that the position of the tool's point of interest estimated by the proposed system is referred to as the reference frame of the kinect camera. Since the accurate measure of the power drill pose is with respect to the reference system of the HTC Vive, it is necessary to know the relative pose between the two reference systems in order to compare the accurate measure with the estimated one. Hence, a second Vive tracker was fixed on the Azure kinect chassis by using a 3D-printed support (Figure 8). Such a support designed by the authors allowed for the positioning of the Vive tracker with a well-known pose with respect to the camera frame.

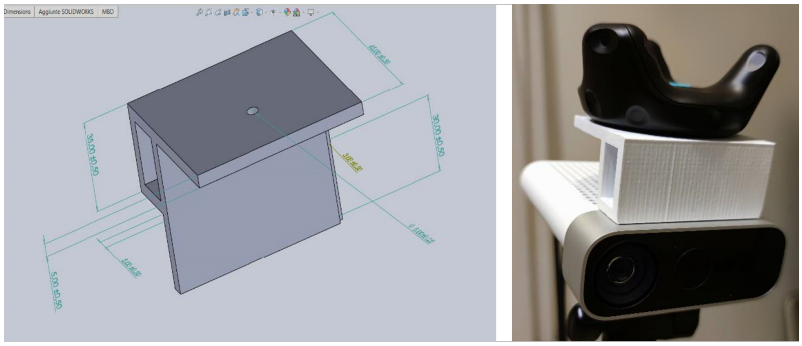

Figure 8. The support mounted on the Azure Kinect: (**Left**) the 3D model of the support; (**Right**) the 3D-printed support mounted on the Azure Kinect with the HTC Vive tracker.

2.2.2. Experiment Description

In order to evaluate and compare the pose estimation performance of the proposed systems, some experiments reproducing typical manual task performed by a worker were conducted. As already reported, a power drill (Figure 7—Left) was selected for the experiment because it is one of the most frequently used tools in industrial assembly/disassembly procedures.

Three different scenarios have been designed:

- In the first scenario, the operator holds the power drill in static position on a workbench (stationary position) (Figure 9—Left);
- In the second scenario, the operator follows a trajectory with the power drill on the platform, keeping the speed of the movement low (slow-motion condition) (Figure 9—Right);
- In the last scenario, the previous trajectory is considered, but the speed was increased (fast-motion condition).

For the second and third scenario, the trajectory was defined as a path from a starting point A to another point B on a horizontal workbench. Two different velocities were adopted for performing the same trajectory (approximately 4 cm/s for slow-motion condition; approximately 8 cm/s for fast-motion condition). A vision feedback on a monitor was used as a virtual reference to follow. Four users were involved in the experiment. Everyone was asked to perform three trials in each condition, for a total amount of twelve trials.

The execution of all of the proposed systems was performed by a real-time C++ application that both integrated all of the frameworks and synchronized the data acquired by the HTC Vive system with the estimation positions computed by each system (Figure 10).

The application ran on a computer with the following configuration: Intel i7-8750H, GTX 1060 with 6 GB memory and 16 GB RAM. The frame per second (FPS) performance for the frameworks on the machine was the following: YOLO 30+ FPS, AKBT 10 FPS, OpenPose 10 FPS and ArUco 30+ FPS. The leveling off of all the performances to the lowest FPS (10 FPS) was required. The final performance of the application was under 10 FPS.

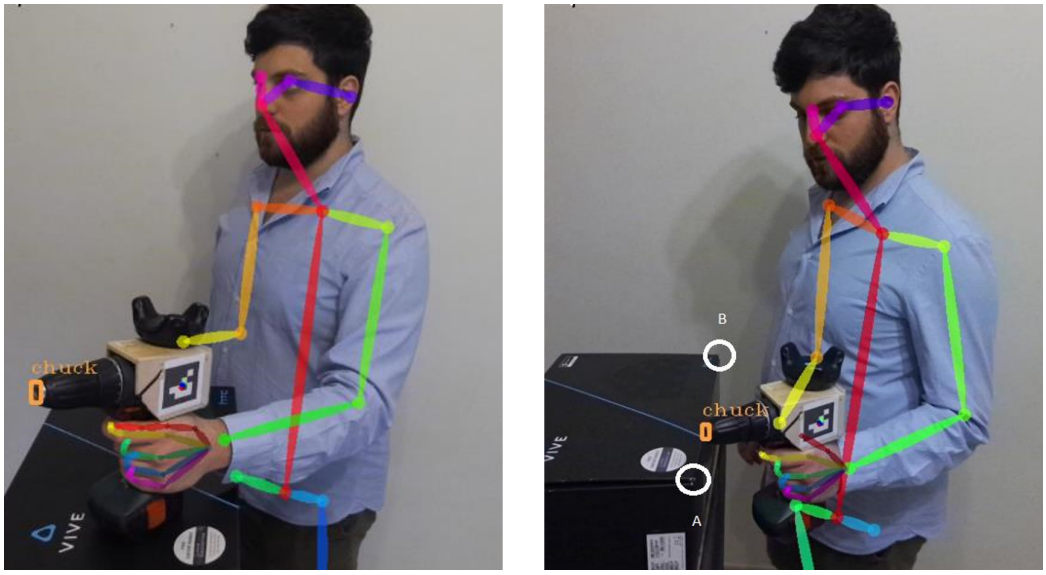

Figure 9. A user performing a trial in (**Left**) stationary condition and (**Right**) slow-motion condition.

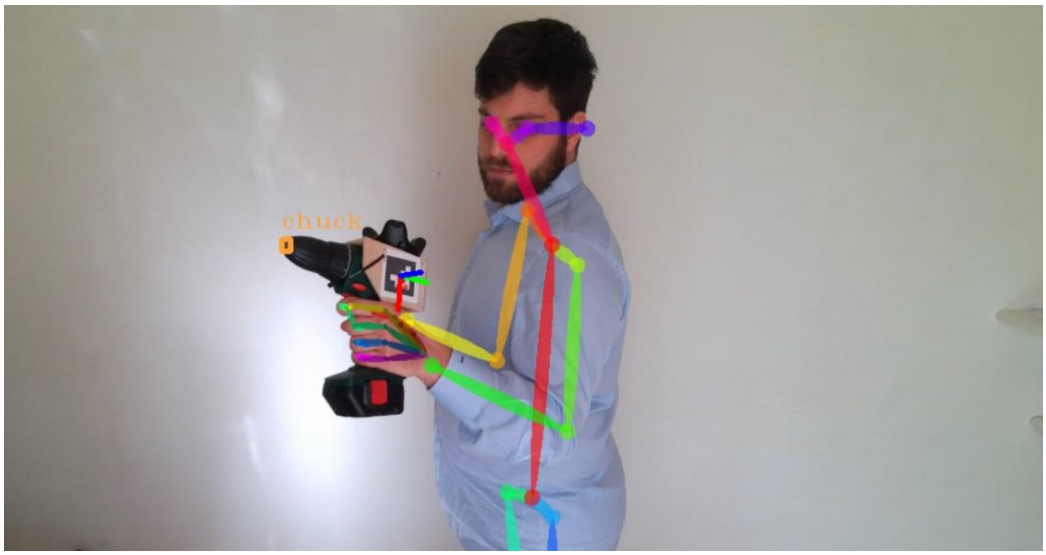

Figure 10. Output of the application for a scene with a user holding the power drill.

2.2.3. Evaluation Metrics

The performance of each proposed methodology was evaluated considering both the root mean square point to point distance (D. RMS) and the multivariate R^2 between the estimated and measured tool path. In particular, such metrics have been independently computed for each trajectory of a trial. It is worth remembering that the tool's pose acquired by the HTC Vive system was considered as the measured pose.

The multivariate R^2 index represents how much variability of the estimated path components is explained by the variability of the measured path components. It is then a global indicator of the goodness of the estimation. The R^2 was computed as follows:

$$R^2 = 1 - \frac{SSE}{SST} = 1 - \frac{\sum_{C=X,Y,Z} \sum_{k=1}^{N} \left\| path_C(t_k) - path_C^{est}(t_k) \right\|^2}{\sum_{C=X,Y,Z} \sum_{k=1}^{N} \left\| path_C(t_k) - \bar{path}_C \right\|^2} \quad (6)$$

where $path_C(t_k)$ is the value of the measured path component (on the X, Y or Z axis) at the specific sample time t_k, $path_C^{est}(t_k)$ is the value of the estimated path component (on the X, Y or Z axis) at the specific sample time t_k, SSE is the sum of the squared errors and SST is the sum of the squared residuals from the mean $\bar{path}_C$.

2.2.4. Statistics

For a deeper look into the difference in the methods, the authors performed the Friedman test and the Dunn's pairwise post hoc tests with Bonferroni correction on root mean square point to point distance (D. RMS) and multivariate R^2 data for each condition, comparing the four methods. The significance level was set to 0.05. Non-parametric tests were adopted since the assumptions underlying parametric tests resulted in being violated for all sets of data. All the analyses were performed using the SPSS software (Version 21). The authors left out the correlation data of stationary condition from this analysis, considering it pointless to apply the test in that particular condition because its variability is already explained by standard deviation previously calculated.

3. Results

For each trajectory estimated by the four proposed systems during the 12 trials, the root mean square point to point distance (D. RMS) and the multivariate R^2 (see Equation (6)) between the measured and estimated trajectory were computed.

The results obtained for the stationary, slow-motion and fast-motion conditions are reported in Tables 1, 2 and 3, respectively.

Table 1. Mean and standard deviation of root mean square point to point distance (D. RMS) expressed in cm calculated on each trajectory for all methods (stationary condition).

Traj. #	ArUco	AKBT	OpenPose	YOLO
1	2.8 (0.4)	6.1 (1.0)	6.1 (0.4)	13.1 (0.5)
2	2.7 (0.4)	8.9 (0.9)	5.3 (0.2)	14.5 (0.4)
3	3.0 (0.4)	6.9 (2.3)	5.6 (0.2)	24.4 (0.5)
4	2.5 (0.5)	5.4 (1.2)	5.9 (0.2)	25.1 (0.4)
5	4.7 (0.1)	2.1 (1.5)	10.1 (0.7)	13.4 (0.1)
6	4.6 (0.8)	11.2 (1.6)	5.1 (0.8)	8.7 (0.4)
7	4.6 (0.9)	10.3 (1.0)	4.1 (0.4)	8.5 (0.3)
8	3.1 (1.3)	7.3 (0.4)	7.6 (0.2)	7.9 (0.1)
9	3.0 (1.0)	9.2 (0.7)	7.4 (0.2)	7.9 (0.1)
10	2.6 (0.4)	8.7 (0.2)	7.2 (0.1)	7.9 (0.1)
11	3.1 (0.3)	8.9 (0.3)	7.1 (0.1)	8.0 (0.1)
12	1.5 (0.1)	15.3 (0.4)	8.1 (0.2)	7.9 (0.1)
	3.1 (1.0)	8.9 (3.8)	6.7 (1.6)	11.9 (6.1)

Table 2. Mean and standard deviation of root mean square point to point distance (D. RMS) expressed in cm and multivariate R^2 calculated on each trajectory for all methods (slow-motion condition).

Traj. #	ArUco		AKBT		OpenPose		YOLO	
	D. RMS	R^2	D. RMS	R^2	D. RMS	R^2	D. RMS	R^2
1	7.0 (1.1)	0.99	20.6 (5.3)	0.98	11.2 (1.3)	0.99	14.7 (1.8)	0.99
2	9.0 (3.3)	0.99	23.1 (2.7)	0.97	12.4 (1.3)	0.99	11.6 (1.9)	0.99
3	8.2 (3.5)	0.99	20.0 (1.7)	0.98	11.1 (0.7)	0.99	12.1 (1.2)	0.99
4	9.8 (4.6)	0.99	22.4 (1.5)	0.97	12.6 (0.9)	0.99	13.4 (1.2)	0.99
5	5.0 (2.7)	0.99	18.3 (0.9)	0.92	9.1 (1.9)	0.98	13.4 (3.4)	0.96
6	6.2 (1.9)	0.99	27.5 (15.6)	0.84	10.0 (2.2)	0.98	15.7 (4.9)	0.95
7	9.0 (2.3)	0.98	13.1 (1.4)	0.97	10.7 (1.3)	0.98	12.5 (0.9)	0.97
8	9.7 (4.5)	0.98	15.0 (2.9)	0.97	6.8 (1.5)	0.99	14.9 (6.4)	0.96
9	11.0 (4.7)	0.98	14.1 (3.6)	0.97	6.6 (1.1)	0.99	18.5 (4.5)	0.95
10	8.9 (3.8)	0.98	12.3 (2.3)	0.97	8.1 (1.5)	0.98	13.4 (2.0)	0.97
11	7.4 (2.6)	0.99	31.6 (12.3)	0.83	7.3 (1.8)	0.99	13.0 (1.7)	0.97
12	4.6 (1.9)	0.99	26.9 (2.4)	0.75	8.3 (1.0)	0.97	9.8 (1.4)	0.96
M (SD)	7.8 (2.0)	-	20.0 (6.1)	-	13.3 (2.4)	-	13.3 (2.3)	-

Table 3. Mean and standard deviation of root mean square point to point distance (D. RMS) expressed in cm and multivariate R^2 calculated on each trajectory for all methods (fast-motion condition).

Traj. #	ArUco		AKBT		OpenPose		YOLO	
	D. RMS	R^2	D. RMS	R^2	D. RMS	R^2	D. RMS	R^2
1	12.5 (6.3)	0.99	25.2 (3.3)	0.97	13.4 (1.8)	0.99	16.6 (2.8)	0.98
2	5.4 (1.7)	0.99	28.7 (1.2)	0.96	11.7 (0.9)	0.99	11.9 (1.1)	0.99
3	12.2 (4.7)	0.99	21.3 (3.1)	0.98	13.1 (2.0)	0.99	14.5 (1.6)	0.99
4	6.3 (1.9)	0.99	21.8 (2.4)	0.97	13.1 (1.8)	0.99	13.5 (2.4)	0.99
5	5.6 (1.0)	0.99	22.6 (0.9)	0.97	11.6 (0.5)	0.99	11.6 (1.7)	0.99
6	13.6 (8.5)	0.98	20.5 (2.9)	0.98	13.0 (1.7)	0.99	15.3 (2.4)	0.99
7	8.7 (6.1)	0.99	18.1 (6.5)	0.98	11.4 (1.6)	0.99	16.3 (4.5)	0.98
8	10.1 (4.0)	0.99	15.5 (5.0)	0.98	11.7 (1.8)	0.99	19.6 (10.1)	0.97
9	9.4 (4.2)	0.99	17.1 (5.6)	0.98	11.6 (1.3)	0.99	14.6 (1.7)	0.99
10	10.7 (6.5)	0.99	15.0 (2.7)	0.99	11.6 (1.4)	0.99	13.9 (2.2)	0.99
11	4.0 (0.6)	0.99	17.4 (4.3)	0.94	10.2 (0.6)	0.98	11.6 (1.8)	0.97
12	5.9 (2.8)	0.99	12.5 (5.4)	0.96	10.0 (0.7)	0.98	10.9 (2.3)	0.97
M (SD)	8.7 (3.2)	-	19.6 (4.6)	-	11.8 (1.1)	-	14.2 (2.5)	-

In order to compare the performance of the methods side by side in the three different conditions, the authors represented the results in boxplot charts. This is a useful way to visualize differences between groups and to quickly identify information, such as median and data dispersion. The resulting boxplots are shown in Figures 11 and 12. Then, the Friedman test and the Dunn's pairwise post hoc tests with Bonferroni correction were performed on the data to compare one-to-one the methods and to understand if their distributions were significantly different (Tables 4 and 5).

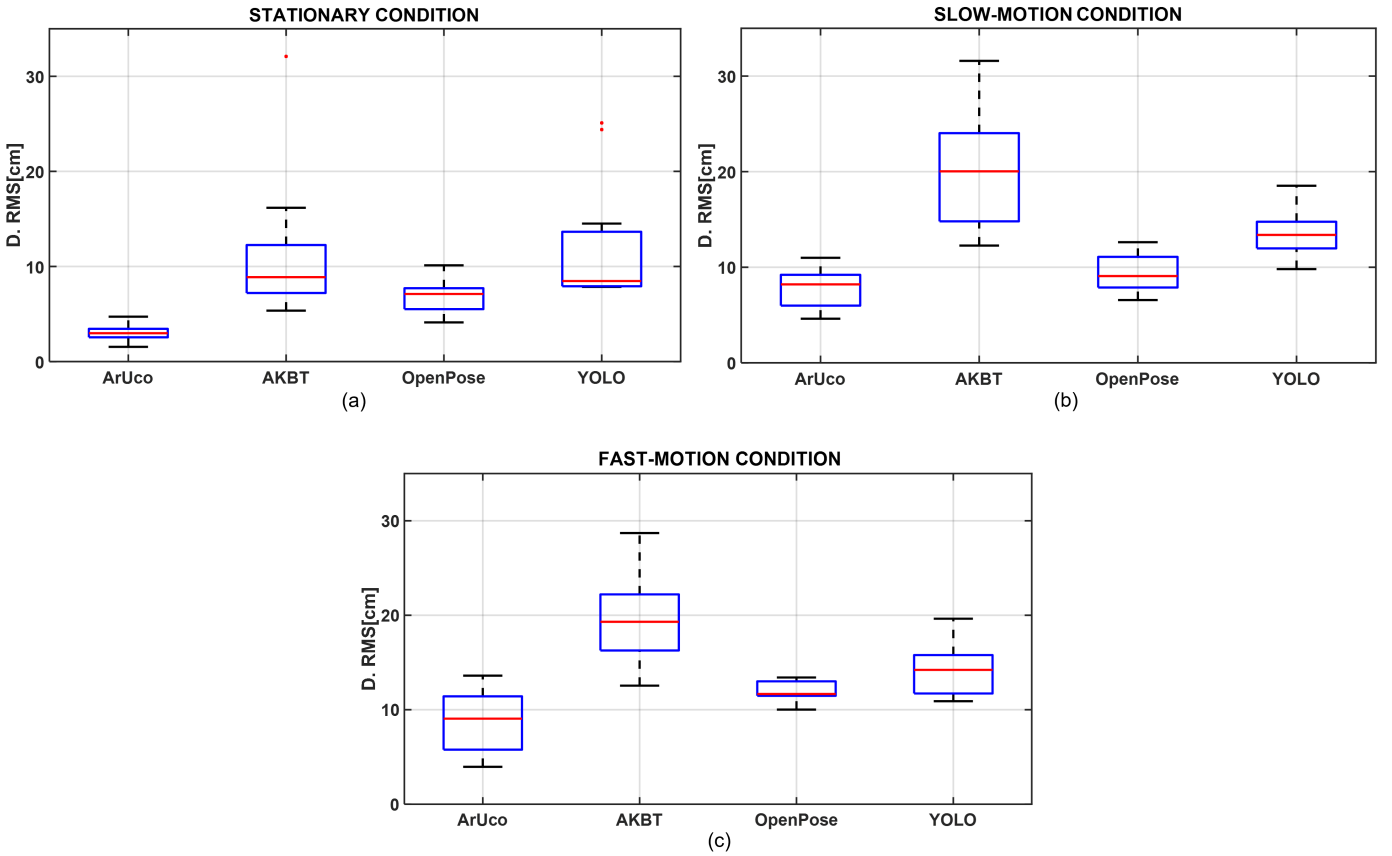

Figure 11. Box plots of root mean square point to point distance (D. RMS) expressed in *cm* calculated for (**a**) stationary condition, (**b**) slow-motion condition and (**c**) fast-motion condition.

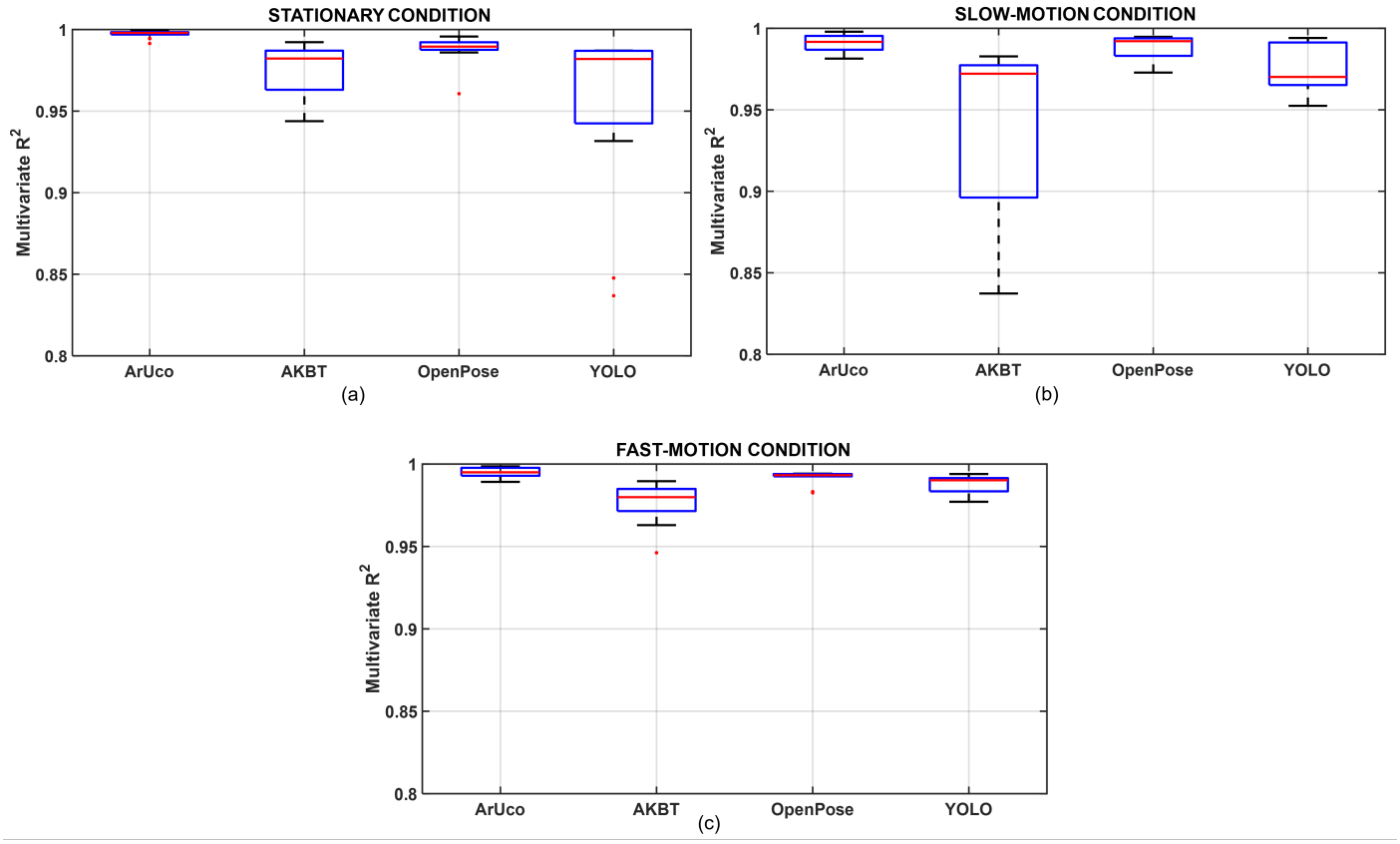

Figure 12. Box plots of multivariate R^2 index values calculated for (**a**) stationary condition, (**b**) slow-motion condition and (**c**) fast-motion condition.

Table 4. *p*-values of root mean square point to point distance (D. RMS) Bonferroni-corrected post hoc methods comparison for dynamic conditions. Significant results ($p \leq 0.05$) are highlighted in bold.

Pairwise Comparison	Stationary	Slow	Fast
ArUco vs. YOLO	**<0.001**	**0.003**	**0.002**
ArUco vs. AKBT	**<0.001**	**<0.001**	**<0.001**
ArUco vs. OpenPose	0.09	1.00	0.492
YOLO vs. AKBT	1.00	0.772	0.492
YOLO vs. OpenPose	0.41	**0.05**	0.347
AKBT vs. OpenPose	0.201	**<0.001**	**0.002**

Table 5. *p*-values of multivariate R^2 Bonferroni-corrected post hoc methods comparison for dynamic conditions. Significant results ($p \leq 0.05$) are highlighted in bold.

Pairwise Comparison	Slow	Fast
ArUco vs. YOLO	**0.002**	**0.009**
ArUco vs. AKBT	**<0.001**	**<0.001**
ArUco vs. OpenPose	1.00	1.00
YOLO vs. AKBT	1.00	0.492
YOLO vs. OpenPose	**0.037**	0.106
AKBT vs. OpenPose	**<0.001**	**<0.001**

The results show that, in all of the conditions, the ArUco method has a great performance. Indeed, it shows a significantly lower root mean square point to point distance (D.RMS) than AKBT and YOLO (stationary ArUco vs. YOLO: *t*-stat = −2.154, $p < 0.001$; stationary ArUco vs. AKBT: *t*-stat = −2.308, $p < 0.001$; slow-motion ArUco vs. YOLO: *t*-stat = −1.769, $p = 0.003$; slow-motion ArUco vs. AKBT: *t*-stat = −2.538, $p < 0.001$; fast-motion

ArUco vs. YOLO: t-stat = −1.917, p = 0.002; fast-motion ArUco vs. AKBT: t-stat = −2.833, $p < 0.001$), but not OpenPose; the ArUco variability increases in dynamic conditions. In addition, OpenPose obtained notable results, showing a significantly lower D.RMS than AKBT and YOLO in the slow-motion condition (OpenPose vs. AKBT: t-stat = 2.077, $p < 0.001$; OpenPose vs. YOLO: t-stat = −1.308, p = 0.05) and only AKBT in the fast-motion condition (t-stat = 1.917, p = 0.002). It appears to be the most reliable in terms of variability compared to all of the methods. On the other hand, the multivariate R^2 results (Figure 12, Table 5) show that the ArUco performance is significantly better than YOLO and AKBT (slow-motion ArUco vs. YOLO: t-stat = 1.846, p = 0.002; slow-motion ArUco vs. AKBT: t-stat = 2.462, $p < 0.001$; fast-motion ArUco vs. YOLO: t-stat = 1.667, p = 0.009; fast-motion ArUco vs. AKBT: t-stat = 2.583, $p < 0.001$) but is comparable with OpenPose in dynamic conditions, and also confirm the difference between OpenPose and Kinect (slow-motion OpenPose vs. YOLO: t-stat = 1.385, p = 0.037; slow-motion OpenPose vs. AKBT: t-stat = −2.0, $p < 0.001$; fast-motion OpenPose vs. AKBT: t-stat = −2.167, $p < 0.001$).

4. Discussion

In order to evaluate and compare ArUco-, OpenPose-, YOLO- and AKBT-based methods, the authors designed a system composed of an Azure Kinect device, HTC Vive tracking system (the benchmark) and an application with all of the frameworks in one that runs on a computer with this configuration: Intel I7-8750H, GTX 1060 with 6GB memory and 16 GB RAM.

The frame per second (FPS) performance for the frameworks on the machine were the following: YOLO 30+ FPS, AKBT 10 FPS, OpenPose 10 FPS and ArUco 30+ FPS. The leveling off of all performances to the lowest FPS (10 FPS) was required. The final performance of the application was under 10 FPS because it was subjected to the overhead of the post-processing phases and the simultaneous execution of the methods.

The authors designed an experiment in which a user handling a power drill simulated three different conditions during an assembly procedure: holding the tool in a static position (stationary condition) and moving the tool at two different velocities (slow-motion condition, 4 cm/s, and fast-motion condition, 8 cm/s).

The performance of the four methods has been evaluated on two metrics: the root mean square point to point distance (D. RMS) and the multivariate R^2 of a trajectory compared to the benchmark system (HTC Vive tracking system).The results have been reported as boxplot charts and a statistical analysis has been performed.

As reported in Figure 11, ArUco resulted in a very accurate method in the stationary condition, showing a D. RMS lower than all the other methods and a low variability. The slow-motion and fast-motion conditions boxplot charts (Figure 11) show that the better accuracy of ArUco is also preserved in dynamic conditions, even if the variability increases, becoming comparable with the one of OpenPose and YOLO for the slow-motion condition and even being the worst overall for the fast-motion condition. The correlation boxplots in Figure 12 confirm the same situations, and the ranking of the methods seems, globally, the following: ArUco, OpenPose, YOLO, AKBT.

The authors could presume that ArUco would be the most accurate of the frameworks because it implements a marker-based technique, unlike the others. It confirms the results of other studies, such as [37]. Nevertheless, it is not enough, because an invasive method cannot be seriously considered in a real industrial application; it is rarely accepted, for safety and ergonomics reasons, to install markers on a hand tool and to handle it in a way that always keeps them visible.

Although OpenPose does not have the best performance in term of D. RMS, it could be considered a valid framework for building a virtual guidance system; the low variability of the D. RMS suggests it is a reliable method if the user can accept a D.RMS of around 10 cm. Furthermore, its non-invasiveness and flexibility make OpenPose a brilliant option to adopt. As a matter of fact, its performance is probably reduced by the limited hardware used for the experiment; a network with a higher resolution should improve its performance.

YOLO v4 did not show an exceptional performance, but it has a good variability of D.RMS for dynamic conditions. The main problem of the framework is that it requires being retrained with many pictures of the hand tool under consideration (735 images were used in this study) in order to make the model able to recognize the point/s of interest. This framework would require a deeper investigation exploiting different points of interest for the selected hand tool to be detected and considering a more effective reinforcement learning for the DL model.

Azure Kinect Body tracking returned the worst results. It is certainly an innovative, compact, lightweight and well-documented framework; its DL model capabilities and accuracy improve with time thanks to continuous support from Microsoft. A study could be designed, in order to reinforce the model performance, on tracking some interesting points of the upper limb for more accurately deriving the hand tool position.

5. Conclusions

In this study, the authors conducted a literature review and market research, looking for studies and out-of-the-box solutions that proposed or implemented methodologies for hand tool pose estimation during assembly and maintenance procedures in the industrial field. They discovered that, even if there are plenty of studies concerning static and dynamic object pose estimation in the literature, the problem of occluded hand tool pose estimation and tracking in the industrial environment is not extensively investigated. Furthermore, it emerged that all of the commercial solutions do not really implement algorithms that are able to accurately track the pose of a tool partially occluded by the operator's hands, but they roughly derive it by the hand pose or by using color matching techniques.

For this reason, the authors selected four of the most promising computer vision and deep learning frameworks (ArUco, OpenPose, YOLO and AKBT) in the field to evaluate their performance in the task of industrial hand tool detection and pose estimation in real-time during an assembly or maintenance procedure. Two different approaches have been considered: a direct approach, in which the tool is directly detected and tracked using some robust features that are visible, and an indirect approach, in which the body joints of the operator and their hands are detected and tracked, and the tool pose is derived by considering the unique handling pose of the tool. To the best of the authors' knowledge, this study is the first that compares the performance of ArUco-, OpenPose-, YOLO- and AKBT-based methods side by side in the task of occluded hand tool tracking.

In order to fairly compare these frameworks, the objective was redefined, reducing the pose estimation problem to only 3D position estimation, because it is intuitive that a method such as YOLO, which only returns a position estimation, requires being complemented with another technique for orientation estimation. A complete pose estimation investigation is worth a further study.

The results of the study suggests OpenPose is the most complete proposal, thanks to its acceptable root mean square point to point distance, D. RMS (approximately 12 cm) and low variability in dynamic conditions, even when a limited network resolution is adopted. This framework is worth a dedicated study in order to exploit all model capabilities in extracting the information with a higher accuracy. OpenPose could be used to implement a tool pose estimation module in a smart workstation for training and assessment. A system designed with this feature would have a great impact in the automotive industry, especially for critical procedures that require monitoring, with high accuracy, the movements of the operator and the correct usage of hand tools for battery-pack assembling, engine repairing and overhauling, glue smearing and adhesive and sealant application. In future studies, the authors aim to investigate the problem of complete pose estimation (including orientation) and to conduct a deeper study on OpenPose, implementing a pose estimation module based on it and evaluating the performance in experiments that simulate procedures at different levels of complexity.

Author Contributions: Conceptualization, I.D.F., D.B., N.L., G.D.S. and V.B.; data curation, I.D.F., D.B., S.G. and G.L.; formal analysis, I.D.F., D.B., A.B., N.L. and G.D.S.; investigation, I.D.F. and D.B.; methodology, I.D.F., D.B., S.G. and G.L.; resources, I.D.F.; software, I.D.F., D.B., S.G. and G.L.; supervision, V.B.; validation, I.D.F., D.B., S.G., G.L., A.B., N.L., G.D.S. and V.B.; visualization, I.D.F. and D.B.; writing—original draft, I.D.F., D.B., A.B., N.L., G.D.S. and V.B.; writing—review and editing, I.D.F., D.B., A.B., N.L., G.D.S. and V.B. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Informed Consent Statement: Informed consent was obtained from all subjects involved in the study.

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

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

References

1. García-Pereira, I.; Casanova-Salas, P.; Gimeno, J.; Morillo, P.; Reiners, D. Cross-Device Augmented Reality Annotations Method for Asynchronous Collaboration in Unprepared Environments. *Information* **2021**, *12*, 519. [CrossRef]
2. Brunetti, A.; Buongiorno, D.; Trotta, G.F.; Bevilacqua, V. Computer vision and deep learning techniques for pedestrian detection and tracking: A survey. *Neurocomputing* **2018**, *300*, 17–33. [CrossRef]
3. Manghisi, V.M.; Uva, A.E.; Fiorentino, M.; Bevilacqua, V.; Trotta, G.F.; Monno, G. Real time RULA assessment using Kinect v2 sensor. *Appl. Ergon.* **2017**, *65*, 481–491. [CrossRef]
4. Oztemel, E.; Gursev, S. Literature review of Industry 4.0 and related technologies. *J. Intell. Manuf.* **2020**, *31*, 127–182. [CrossRef]
5. Xu, L.D.; Xu, E.L.; Li, L. Industry 4.0: State of the art and future trends. *Int. J. Prod. Res.* **2018**, *56*, 2941–2962. [CrossRef]
6. Quevedo, W.X.; Sánchez, J.S.; Arteaga, O.; Álvarez, M.; Zambrano, V.D.; Sánchez, C.R.; Andaluz, V.H. Virtual reality system for training in automotive mechanics. In Proceedings of the International Conference on Augmented Reality, Virtual Reality and Computer Graphics (AVR 2017), Ugento, Italy, 12–15 June 2017; Springer: Cham, Switzerland, 2017; pp. 185–198. [CrossRef]
7. Kumar, R.; Banga, H.K.; Kumar, R.; Singh, S.; Singh, S.; Scutaru, M.L.; Pruncu, C.I. Ergonomic evaluation of workstation design using taguchi experimental approach: A case of an automotive industry. *Int. J. Interact. Des. Manuf. (IJIDeM)* **2021**, *15*, 481–498. [CrossRef]
8. Cao, Y.; Jia, F.; Manogaran, G. Efficient traceability systems of steel products using blockchain-based industrial Internet of Things. *IEEE Trans. Ind. Inform.* **2019**, *16*, 6004–6012. [CrossRef]
9. Kostakis, P.; Kargas, A. Big-Data Management: A Driver for Digital Transformation? *Information* **2021**, *12*, 411. [CrossRef]
10. Webel, S.; Bockholt, U.; Keil, J. Design criteria for AR-based training of maintenance and assembly tasks. In Proceedings of the International Conference on Virtual and Mixed Reality Held as Part of HCI International 2011 (VMR 2011), Orlando, FL, USA, 9–14 July 2011; Springer: Berlin/Heidelberg, Germany, 2011; pp. 123–132. [CrossRef]
11. Lee, K. Augmented reality in education and training. *TechTrends* **2012**, *56*, 13–21. [CrossRef]
12. Zajec, P.; Rožanec, J.M.; Trajkova, E.; Novalija, I.; Kenda, K.; Fortuna, B.; Mladenić, D. Help Me Learn! Architecture and Strategies to Combine Recommendations and Active Learning in Manufacturing. *Information* **2021**, *12*, 473. [CrossRef]
13. Holz, D.; Ichim, A.E.; Tombari, F.; Rusu, R.B.; Behnke, S. Registration with the point cloud library: A modular framework for aligning in 3-D. *IEEE Robot. Autom. Mag.* **2015**, *22*, 110–124. [CrossRef]
14. Du, G.; Wang, K.; Lian, S.; Zhao, K. Vision-based robotic grasping from object localization, object pose estimation to grasp estimation for parallel grippers: A review. *Artif. Intell. Rev.* **2021**, *54*, 1677–1734. [CrossRef]
15. Altini, N.; De Giosa, G.; Fragasso, N.; Coscia, C.; Sibilano, E.; Prencipe, B.; Hussain, S.M.; Brunetti, A.; Buongiorno, D.; Guerriero, A.; et al. Segmentation and Identification of Vertebrae in CT Scans Using CNN, k-Means Clustering and k-NN. *Informatics* **2021**, *8*, 40. [CrossRef]
16. Zhu, M.; Derpanis, K.G.; Yang, Y.; Brahmbhatt, S.; Zhang, M.; Phillips, C.; Lecce, M.; Daniilidis, K. Single image 3D object detection and pose estimation for grasping. In Proceedings of the 2014 IEEE International Conference on Robotics and Automation (ICRA), Hong Kong, China, 31 May–7 June 2014; pp. 3936–3943.
17. Schwarz, M.; Schulz, H.; Behnke, S. RGB-D object recognition and pose estimation based on pre-trained convolutional neural network features. In Proceedings of the 2015 IEEE international conference on robotics and automation (ICRA), Seattle, WA, USA, 26–30 May 2015; pp. 1329–1335.
18. Periyasamy, A.S.; Schwarz, M.; Behnke, S. Robust 6D object pose estimation in cluttered scenes using semantic segmentation and pose regression networks. In Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, 1–5 October 2018; pp. 6660–6666.
19. Kendall, A.; Grimes, M.; Cipolla, R. Posenet: A convolutional network for real-time 6-dof camera relocalization. In Proceedings of the IEEE International Conference on Computer Vision, Santiago, Chile, 7–13 December 2015; pp. 2938–2946.

20. Banga, H.K.; Goel, P.; Kumar, R.; Kumar, V.; Kalra, P.; Singh, S.; Singh, S.; Prakash, C.; Pruncu, C. Vibration Exposure and Transmissibility on Dentist's Anatomy: A Study of Micro Motors and Air-Turbines. *Int. J. Environ. Res. Public Health* **2021**, *18*, 4084. [CrossRef] [PubMed]
21. Romero-Ramirez, F.J.; Muñoz-Salinas, R.; Medina-Carnicer, R. Speeded up detection of squared fiducial markers. *Image Vis. Comput.* **2018**, *76*, 38–47. [CrossRef]
22. Garrido-Jurado, S.; Munoz-Salinas, R.; Madrid-Cuevas, F.J.; Medina-Carnicer, R. Generation of fiducial marker dictionaries using mixed integer linear programming. *Pattern Recognit.* **2016**, *51*, 481–491. [CrossRef]
23. Bochkovskiy, A.; Wang, C.Y.; Liao, H.Y.M. Yolov4: Optimal speed and accuracy of object detection. *arXiv* **2020**, arXiv:2004.10934.
24. Wang, C.Y.; Bochkovskiy, A.; Liao, H.Y.M. Scaled-yolov4: Scaling cross stage partial network. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Nashville, TN, USA, 20–25 June 2021; pp. 13029–13038.
25. Romeo, L.; Marani, R.; Malosio, M.; Perri, A.G.; D'Orazio, T. Performance analysis of body tracking with the microsoft azure kinect. In Proceedings of the 2021 29th Mediterranean Conference on Control and Automation (MED), Puglia, Italy, 22–25 June 2021; pp. 572–577.
26. Tölgyessy, M.; Dekan, M.; Chovanec, L.; Hubinskỳ, P. Evaluation of the azure Kinect and its comparison to Kinect V1 and Kinect V2. *Sensors* **2021**, *21*, 413. [CrossRef]
27. Cao, Z.; Hidalgo Martinez, G.; Simon, T.; Wei, S.; Sheikh, Y.A. OpenPose: Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields. *IEEE Trans. Pattern Anal. Mach. Intell.* **2019**, *43*, 172–186. [CrossRef]
28. Simon, T.; Joo, H.; Matthews, I.; Sheikh, Y. Hand Keypoint Detection in Single Images using Multiview Bootstrapping. In Proceedings of the 2017 Hand Keypoint Detection in Single Images using Multiview Bootstrapping (CVPR), Honolulu, HI, USA, 21–26 July 2017.
29. Cao, Z.; Simon, T.; Wei, S.E.; Sheikh, Y. Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields. In Proceedings of the 2017 Hand Keypoint Detection in Single Images Using Multiview Bootstrapping (CVPR), Honolulu, HI, USA, 21–26 July 2017.
30. Wei, S.E.; Ramakrishna, V.; Kanade, T.; Sheikh, Y. Convolutional pose machines. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016.
31. Altini, N.; Cascarano, G.D.; Brunetti, A.; De Feudis, I.; Buongiorno, D.; Rossini, M.; Pesce, F.; Gesualdo, L.; Bevilacqua, V. A Deep Learning Instance Segmentation Approach for Global Glomerulosclerosis Assessment in Donor Kidney Biopsies. *Electronics* **2020**, *9*, 1768. [CrossRef]
32. Ikbal, M.S.; Ramadoss, V.; Zoppi, M. Dynamic Pose Tracking Performance Evaluation of HTC Vive Virtual Reality System. *IEEE Access* **2020**, *9*, 3798–3815. [CrossRef]
33. Niehorster, D.C.; Li, L.; Lappe, M. The accuracy and precision of position and orientation tracking in the HTC vive virtual reality system for scientific research. *i-Perception* **2017**, *8*, 2041669517708205. [CrossRef] [PubMed]
34. Borges, M.; Symington, A.; Coltin, B.; Smith, T.; Ventura, R. HTC vive: Analysis and accuracy improvement. In Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, 1–5 October 2018; pp. 2610–2615.
35. Lwowski, J.; Majumdat, A.; Benavidez, P.; Prevost, J.J.; Jamshidi, M. HTC Vive Tracker: Accuracy for Indoor Localization. *IEEE Syst. Man Cybern. Mag.* **2020**, *6*, 15–22. [CrossRef]
36. De Feudis, I.; Buongiorno, D.; Cascarano, G.D.; Brunetti, A.; Micele, D.; Bevilacqua, V. A Nonlinear Autoencoder for Kinematic Synergy Extraction from Movement Data Acquired with HTC Vive Trackers. In *Progresses in Artificial Intelligence and Neural Systems*; Springer: Singapore, 2021; pp. 231–241. [CrossRef]
37. Smirnov, A. Hand Tracking for Mobile Virtual Reality. Bachelor's Thesis, Charles University, Prague, Czech Republic, 2021.