Using Computer Vision to insure safety in Human-Robot Collaboration

Leo Bringer (lbringer@umich.edu)

Introduction

Human – Robot Collaboration (HRC) is a recent but prominent trend in the field of industrial and manufacturing robotics as a part of the strategy Industry 4.0. The main goal of this innovative strategy is to build up an environment for safety collaboration between humans and robots to improve efficiency in task execution. There is an area between manual manufacture and fully automated production where a human worker comes into contact with machine. This area has many limitations due to safety restrictions and this is where sensors like cameras for intelligent visual systems can step in to tackle this challenge. Initially, the machine is allowed to be at automatic work only if the operating personnel is out of its workspace. Collaborative robotics establishes new opportunities in the cooperation between humans and machines. Computer vision systems can allow robots to recognize their human collaborators and their position in space to adapt their movement accordingly and avoid collisions. Personnel shares the workspace with the robot where it helps him with non-ergonomic, repetitive, uncomfortable or even dangerous operations. This study lays the stress on the implementation of this kind of vision system using Human Pose Estimation to locate the human collaborator in 3D. With a data set of human body and robot joint coordinates in the same reference many methods allows to ensure safety and improve efficiency in HRC.

Project Overview

The first part of this study was to set up a vision system adapted to the workspace of the robot and that collects depths information of the human collaborator. Our system uses the Azur Kinect which is the latest version of this motion sensor that uses Microsoft own Time of Flight technology for depth generation. We implemented a ROS Driver that publishes sensor data from the Azure Kinect Developer Kit to the Robot Operating System (ROS) and thus allow to connect the Azure Kinect Developer Kit to the existing ROS installation of our Kawasaki robot (duAro). Then, we worked on Human Pose estimation in 3D to collect the body joint coordinates of the human collaborator in the robot workspace. The next challenge was to gather both the data from the camera and those from the robot in the same reference. We performed this Extrinsic Calibration using April tags and Froward Kinematics of the joint coordinates of the robot end effectors. Finally, we obtained a demo visualization in 3D using the ROS graphical interface and visualization widget RViz to confront the results of both data sets of coordinates in 3D. Finally, in this paper we briefly discuss some solutions that tackle Collision Avoidance and Human-robot interaction using the output of our method.

Background

Human Pose Estimation

Human Pose Estimation (HPE) is a Computer Vision technique that allows to identify and classify the joints in the human body. Essentially it is a way to capture a set of coordinates for each joint (arm, head, torso, etc.,) which is known as a key point that can describe a pose of a person. The connection between these points is known as a pair. The connection formed between the points has to be significant, which means not all points can form a pair. From the outset, the aim of HPE is to form a skeleton-like representation of a human body and then process it further in order to locate the human collaborator and the magnitude of its movements.
There are three types of approaches to model the human body depending on the type of application: skeleton-based model; contour-based model, volume-based model. In our case, since the use of the HPE is only to locate the articulations of the human collaborator's upper-body, the skeleton-based model will be sufficient. Moreover, a simplified model will be more advantageous in terms of computing power for our learning-based approach that is supposed to predict full 3D human pose from RGBD input.

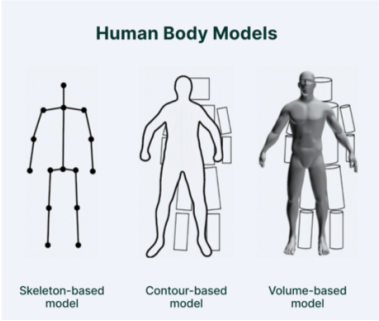

Figure 1: The different types of Models for HPE

HPE approaches are used to understand geometric and motion information of the human body, which can be very intricate. There are two different approaches in HPE that both have different pros and cons [2]: the classical approach and the deep learning-based approach.

Classical approaches usually refer to techniques and methods involving shallow machine learning algorithms. For instance, the Pictorial Structure Framework (PSF) [1] is commonly referred to as one of the traditional methods to estimate human pose. In essence, the PSF objective is to represent the human body as a collection of coordinates for each body part in a given input image. These kind of classical approach models work well when the input image has clear and visible limbs, however, they fail to capture and model limbs that are hidden or not visible from a certain angle.

Deep learning-based approaches are well defined by their ability to generalize any function (if a sufficient number of nodes are present in the given hidden layer). When it comes to computer vision tasks, deep convolutional neural networks (CNN) surpass all other algorithms, and this is especially true in HPE. Indeed, CNN has the ability to extract patterns and representations from the given input image with more precision and accuracy than any other algorithm; this makes CNN very useful for tasks such as classification, detection, and segmentation. Hence, unlike the classical approach, where the features were handcrafted; CNN can learn complex features when provided with enough training-validation-testing data. As a result, the deep learning-based approach might be the most relevant one in our case, since the human subject might be occluded by the robot arms during the task realization and thus our approach needs to be able to deal with occlusion. However, deep learning methods have high computational requirements and require training-validation-testing data. In our case, we began this study using the 3D Skeletal Tracking DNN developed by Microsoft [6] and implemented on the Software Development Kit of the Azure Kinect.

3D Skeletal Tracking on Azure Kinect

In this section, we are going to briefly study the functioning of the Skeleton tracking method developed by Microsoft [6]. First of all, the algorithm uses the Sensor SDK to gather IR image from the depth sensor. Then, the IR Image is used as an input to perform 2D pose estimation through a neural network based solution using ONNX Runtime which is a cross-platform inference and training machine-learning accelerator. As an output, the CNN gives both body part segmentation and 2D joints coordinates (32 in total) of the human bodies. Using the joint coordinates, the algorithm performs model fitting in the loop of the 2D joints on the depth image through a CMU Mocap database collected through synthetic generated data of body poses. The model fitting here is a Regression-based methods that uses a deep network to regress the model parameters and that give us as an output 3D joint coordinates of the human bodies.

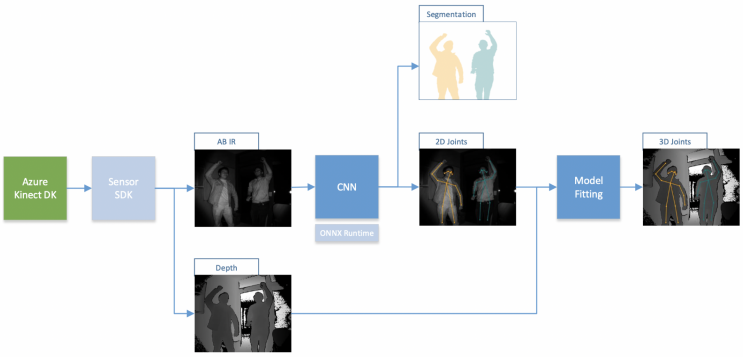

Figure 2: System Architecture of the Azure Kinect Body Tracking SDK

What has been done

Set up of the vision system

At first, the study has begun with the intent to use the Microsoft Kinect sensor for Xbox 360. This Kinect has a high-resolution depth and visual (RGB) sensing and has become available for widespread use. The complementary nature of the depth and visual information provided by the Kinect sensor has opened up a lot of opportunities to solve fundamental problems in computer vision. This invention as allowed recent approaches to deal with several challenges and has reported remarkable results. However, if the Kinect sensor for Xbox 360 can be a very useful for many Computer Vision Methods like 3D Mapping or Navigation, they are many issues related to skeleton tracking for research purposes with this sensor [3].

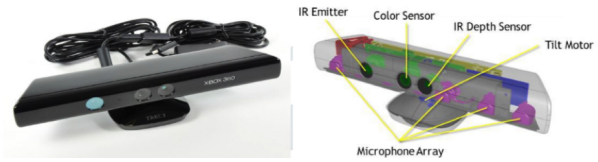

Figure 3: The Microsoft Kinect for Xbox 360 and its intern sensors

Indeed, if Skeleton Tracking methods are implemented in the Kinect for Xbox 360, they are restricted to game purposes-related projects. Indeed, there are two frameworks/ ROS packages that enables software support for the Xbox 360 Kinect:

1. OpenNI, which is a natural interaction framework associated with the NITE and SensorKinect drivers published by PrimeSense, manufacturer of the chips inside Kinect. PrimeSense was the company that originally developed the technology behind the Xbox 360 Kinect sensor, but they have since been acquired by Apple and shut down in 2016. Hence, this open source Software Development Kit (SDK) has been partly maintained for a while by a company that partnered with PrimeSense (Occipital), but is now quite outdated and

requires some unofficial source like NiTE to access functionality like skeleton tracking.

2. OpenKinect (also called freenect or libfreenect on ROS) is open source drivers developed by the OpenKinect community. They are the first drivers published specifically for Kinect. It doesn't have a clear corporate sponsor like OpenNI but has the benefit of being maintained and kept updated by an important community and supports many language wrappers and OS (Windows, Linux and Mac). However, this framework does not have skeleton tracking built in.

Both of these have been widely used for Kinect projects outside of Xbox 360 games but are now either outdated (does not support recent ROS versions) or requires unofficial sources with few documentation and thus are not reliable for serious projects that are intended to be maintained over the long term. As a result, a lot of research has been made and the solution adopted was to purchase a more recent RGBD sensor that would allow better efficiency and that supports the latest innovations and developed technologies in HPE.

We thus decided to decided to purchase an Azure Kinect which has both a Software Development Kit and a Robot Operating System (ROS) Driver that publishes sensor data from the Azure Kinect DK to ROS and thus allow us to connect the Azure Kinect DK to the existing ROS installation of our Kawasaki robot (duAro). We thus set up this ROS Driver on our own workspace and fine-tuned the parameters to obtain a database of information adapted to the ones we need for our study.

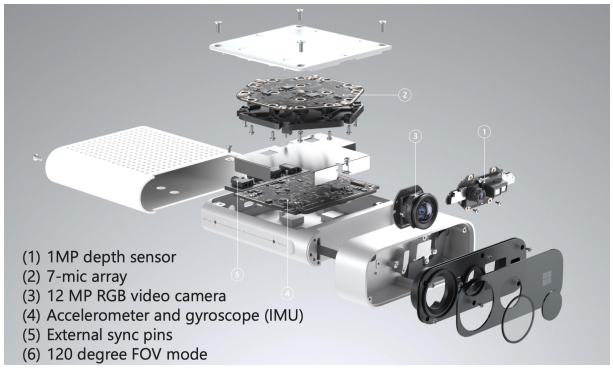

Figure 4: The Azure Kinect and its intern sensors

These internal sensors listed in that last figure allow to collect three different kind of visual data: RGB, Depth and Infrared images which combined result in the 3D camera that we can see in the next figure. Many methods are being used to extract very accurate, qualitative, and realistic models of 3D reconstruction but the main idea behind this process is a triangular grid creation process called triangulated irregular network (TIN) [5] spatial interpolation; a TIN is represented as a continuous surface consisting of triangular faces. To obtain these triangular faces the algorithm requires a simultaneous combination of its RGB-D sensors data, and then the features of the considered scene are detected and extracted. Then, homologous (key) points are sought between the camera frames and are matched. Initially, a sparse point cloud is created, and then its local coordinates are transformed into the global coordinate system using the co-linearity equations. The sparse point cloud has a low resolution. From the sparse cloud emerges the dense cloud, which has metric value, and its density depends on the number of frames. In the next step, a triangle grid is created among points of the dense cloud, and the resolution of the depth map is determined by the number of triangular surfaces. A texture is then given at each triangular surface. The position and orientation of the camera is calculated in real time for the desired reference system, and the three-dimensional reconstructed model is extracted.

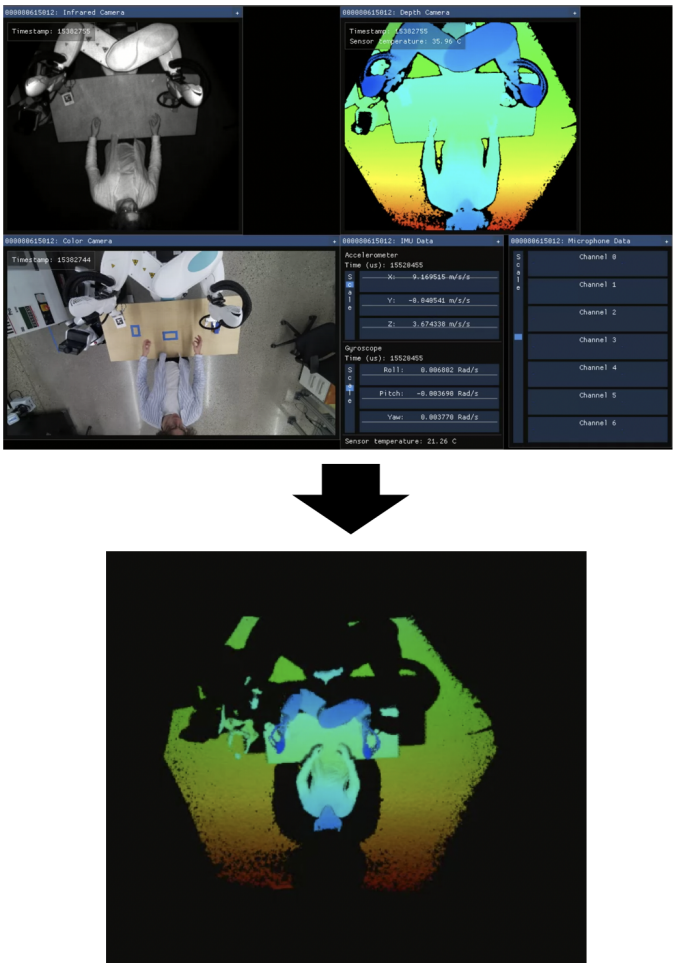

Figure 5: Combination of the intern sensors data to create 3D images

For our demonstration in the last part we use this point cloud 3D reconstruction and the Skeleton Tracking of the SDK [6] described in the Background study.

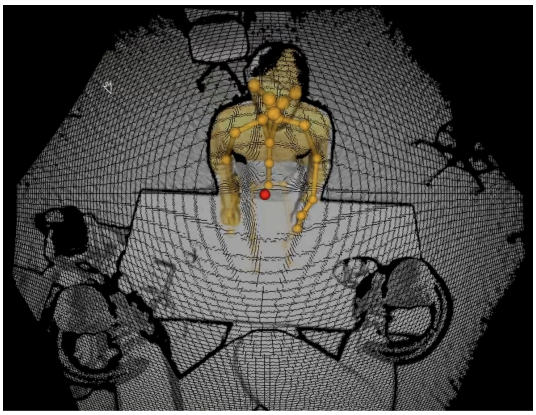

Figure 6: Set up of the Body Tracking SDK

Design of the camera hanging system

Since our Vision Sensor is supposed to both detect the human collaborator in the robot workspace and reference it in the base of the robot using AprilTags located on the robot end effectors (see next subsection), we considered that the best location for the main camera was from above with a camera "high angle" directed toward the human to best capture its real time poses. Hence, we had to find a way to design a stable system that hanged the camera to the ceiling. We also decided to provide the system with several Degree of Freedom (DoF) in rotation, in order to adapt the angle of the camera to the subject, in case the robot has to be moved for a demonstration or if many human collaborators are to be considered. We thus designed a very simple hanging system using a simple Tripod Extender (1 DoF of translation) and a double ball head (3 DoF of rotations).

Figure 7: Hanging system

April tags extrinsic calibration

Since the goal of this study is to obtain a dataset of coordinates of both the human joints and the robot arms in the same reference to confront those coordinates, measure distances and ensure safety in the robot workspace, we needed to obtain all of our data previously collected in the same reference (here we chose the robot basis). In that order, we performed extrinsic calibration using April Tags.
April Tag is a visual fiducial system, useful for a wide variety of tasks including augmented reality, robotics, and camera calibration. Targets can be created from an ordinary printer, and the April Tag detection software computes the precise 3D position, orientation, and identity of the tags relative to the camera.

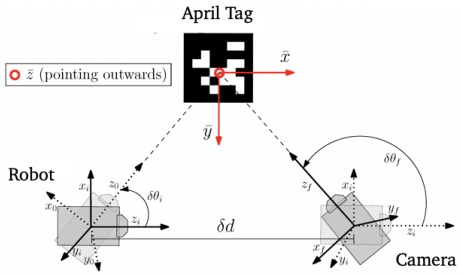

Figure 8: April Tag Calibration

Hence, since the coordinates of the end effectors of the duAro are easily obtainable in the basis of the robot we decided to stick an April Tag (only one as a permanent solution but more of them allows better precision in the calibration) on an end effector of the robot in order to obtain the position and orientation of that end effector in the reference of the camera. To do that, we used a ROS package that enables to perform a continuous detection and obtain in real time the position and orientation of that April Tag (which is also those of the end effector with an offset) in the reference of the camera.
Then, using Forward Kinematics we obtained the position and orientation of that same end effector in the reference of the robot from its joint coordinates.

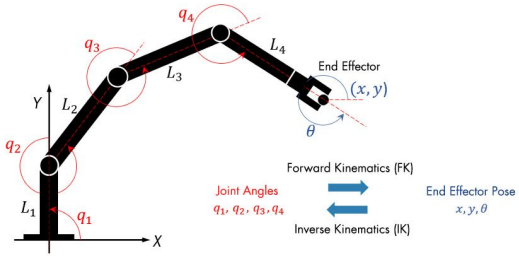

Figure 9: Forward Kinematics

So, let us name T a transformation matrix between two different coordinate system:

$$T = \begin{bmatrix} R_{3x3} & t_{3x1} \\ 0_{1x3} & 1 \end{bmatrix}$$

where R_{3x3} is a rotation matrix and t_{3x1} is a translation vector.

So if we name T_{rf} the transformation matrix between the robot coordinate system and the fiducial coordinate system (known), T_{fc} the transformation matrix between the fiducial coordinate system and the camera coordinate system (also known) and T_{rc} the transformation matrix between the robot coordinate system and the camera coordinate system(the one we seek for), we have:

$$T_{rc} = T_{rf} \times T_{fc}$$

Demonstration and Visualization on RViz

Finally, after performing the extrinsic calibration (Fig 10) and a data collection of both the joints of human pose estimation and 3D point cloud reconstruction, we performed a demonstration on RViz of a fusion of all datasets in the same reference:

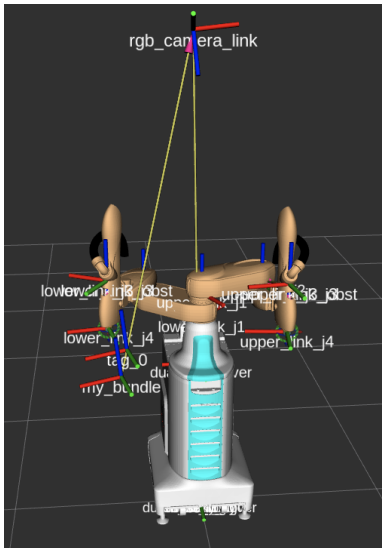

Figure 10: Visualization on RViz of the Calibration of the camera in the reference of the robot

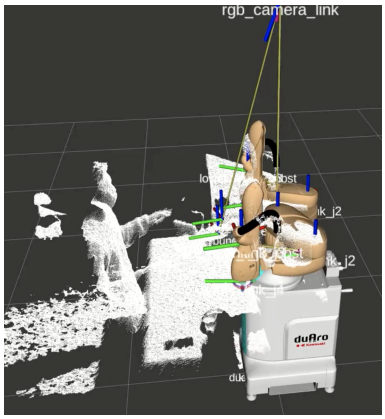

Figure 11: Visualization on RViz of the 3D point cloud reconstruction in the robot reference system

Figure 12: Comparison of the visualization on RViz of the 3D point cloud coupled with the HPE body joints and the real time data from the RGB camera of the Azure Kinect

Study of precision and accuracy in the data collection

First and foremost, since we are going to primarily use this data-set to insure safety in Human Robot Collaboration we need to know what accuracy we can expect from our real time data in order to prevent unexpected collisions and secure the collaborative workspace.
There are multiple potential sources of inaccuracy in the fused data-set but the only possibly significant ones are the following ones :

- The first one is related to the visual fiducial marker system (April-Tag), the backbone of the state estimation of the robot position at a given time instance in the camera's reference [8]. It has been proved that the angular rotation of the camera (yaw angle) about its vertical axis is the primary source of error that decreases the precision to the point where the marker system is not potentially viable for sub-decimeter precise tasks. It is also observed that the ideal scenario for April-Tag accuracy is when the camera is pointing towards the center of the tag or camera z-axis lies toward the center of April-Tag. The distance from the camera to the April-tag and the size of the April-tag itself also obviously dictates the accuracy expected in the measurements, however in a reasonable range of distance with a reasonable marker size the accuracy expected does not fluctuate very significantly. As a result, we have adapted our set up accordingly so that the camera points directly toward the center of the April-Tag used for the extrinsic calibration with a low yaw angle. Also we have added another tripod extender with a double ball head to reduce the distance that is now of 1.11m (44 inches) for a 54mm (2 inches) marker. To test the accuracy of our April-Tag Based State Estimation we have used another April-Tag, placed both of them on end-effectors of the robot and used the relative position of both end-effectors compared to the relative position of both fiducial markers state estimation. In our current set up we have obtained a position estima-

tion error of +/-1mm on each x, y and z axis and a rotation error of 1.5 ° for the roll angle, 1.2 ° for the pitch angle and 1.7 ° for the yaw angle.

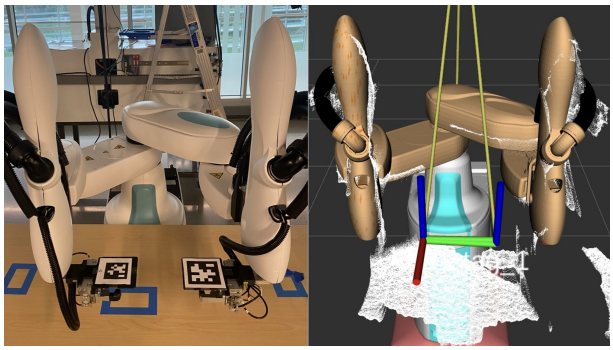

Figure 13: Measurement system of accuracy in April-Tag Based State Estimation

- The second potential source of inaccuracy is related to the data collection of the point cloud for the point cloud processing and this error coupled with algorithmic inaccuracy of HPE for the body joints. For point cloud processing, according to previous studies [9], in a range of 1.5m from the camera (in its field of view) the systematic spatial error over the horizontal plane is less than 2mm for the Azure Kinect sensor and less than 4mm in a range of 3m. To be more precise, the errors are in the range of 0.6 to 3.7 mm (standard deviation) for distances from 1.0 to 5.0 m. However, this error analysis only incorporated the center of the target surface, whereas the largest random errors are typically observed toward object boundaries. In our study, both the April-Tag and the human collaborator are centered in the camera frame but it is important to note that in more general terms the inaccuracy is higher as we move toward the boundaries of the camera's view. As for the body joints estimation, previous studies [10] about Pose Tracking Performance review of the Azure Kinect, have shown that the inaccuracy of the spatial position of the body joints is considerably more significant than the rest of the other inaccuracies. Indeed, only for upper body part joint estimation (which is usually more accurate) the range of precision is around 10 to 20 mm (not including the hands that could attain higher inaccuracy). Compared to the State of the Art of tracking error, these ranges are actually quite reasonable but compared to the rest of our study, this error is much more significant and requires to be taken in consideration for safety measures.

- Previous extensive experiments [8], [9] show that the main source of inaccuracy during dynamic collaboration is the frame inconsistency due to motion during calibration and which significantly reduces the performance. The reason is that April-Tag fiducial system is coded in such a way that the output frame of reference is dependent upon the yaw angle orientation of the camera. As the orientation of the moving marker changes, the output frame also changes, making it hard to have a consistent frame of reference. Hence, our current set up performs the calibration only once in the initial position of the system, since the camera is static in the robot basis' reference¿

Potential applications for Obstacle Avoidance

Finally, now that we have a data set with both coordinates of linked body joints of the human collaborator and coordinates of the robot arms in the same reference, there are many possible applications for obstacle avoidance and even for possible human robot collaborations to be considered.
The most common ideas in the area of HRC for obstacle avoidance are the following:

- Creating a kinetostatic repulsive danger field around the arm of the robot. In this solution, when the human arms come close (with gradually increasing intensity depending on the euclidean distance) to the repulsive field, the robot arms are being repulsed and thus have to adapt their trajectory in order to avoid collision with the human collaborator. The challenges related to this solution is that the repulsive force, hinder the robot's trajectory which forces him to adapt its trajectory in real time and might jeopardize its task execution.

- The second solution is to create gradual danger zones around the robot (or robot arms for better insurance). When the human collaborator's arms penetrate the danger zones, the velocity of the movement of the robot is gradually decreased in order for the human to adapt its movement and avoid collisions.

- Another solution would be to use real-time vision algorithm for reactive avoidance [7] of moving obstacles that determines the euclidean distance of the closest coordinates of the human and the robot arms to either adapt the trajectory of the robot in real time or temporarily stop it.

References

[1] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, R. Moore, A. Kipman, A. Blake, "Real-Time Human Pose Recognition in Parts from Single Depth Images" Microsoft Research Cambridge & Xbox Incubation, 2011.

[2] N. Barla, A Comprehensive Guide to Human Pose Estimation, v7labs, 2021.

[3] V. Bhati, "A Review Study On Human Pose Estimation Using Kinect Sensor," in JETIR (ISSN-2349-5162), vol. IV, Issue 11, 2017-Nov.

[4] Z. Cao, G. Hidalgo, T. Simon, S. Wei, and Y. Shei, "OpenPose: Realtime Multi-Person 2D Pose Estimation using Part Affinity Field,", IEEE Transactions On Pattern Analysis And Machine Intelligence, Edition 30 May 2019.

[5] C. Zimmermann, T. Welschehold, C. Dornhege, W. Burgard and T. Brox, "3D Human Pose Estimation in RGBD Images for Robotic Task Learning," IEEE Transactions On Pattern Analysis And Machine Intelligence, edition 13 Mar 2018.

[6] Z. Liu, "3D Skeletal Tracking on Azure Kinect," Microsoft-Research, 2019.

[7] D. Kulić and E. Croft, "Pre-collision safety strategies for human-robot interaction,", , AutonomousRobots, pp. 149–164, 2007.

[8] S. Abbas, S. Aslam, K. Berns and A. Muhammad, "Analysis and Improvements in AprilTag Based State Estimation,", MDPI, Basel, Switzerland., in Sensors 2019.

[9] G. Kurillo, E. Hemingway, M. Cheng and L. Cheng, "Evaluating the Accuracy of the Azure Kinect and Kinect v2,", MDPI, Basel, Switzerland., in Sensors 2022.

[10] J. A. Albert, V. Owolabi, A. Gebel, C. M. Brahms, U. Granacher and B. Arnrich, "Evaluation of the Pose Tracking Performance of the Azure Kinect and Kinect v2 for Gait Analysis in Comparison with a Gold Standard: A Pilot Study,", MDPI, Basel, Switzerland., in Sensors 2020.