Automatic MTM-Transcription in Virtual Reality Using the Digital Twin of a Workplace

Emmanouil Andreopoulos
https://orcid.org/0000-0002-1486-8279
Valentina Gorobets (vgorobets@ethz.ch)
ETH Zürich: Eidgenossische Technische Hochschule Zurich
Andreas Kunz

Research Article

Keywords: Methods-Time Measurement, Virtual Reality, Action Detection, Workplace Optimization

Posted Date: October 20th, 2022

DOI: https://doi.org/10.21203/rs.3.rs-2110194/v1

Automatic MTM-Transcription in Virtual Reality Using the Digital Twin of a Workplace

Emmanouil Andreopoulos[1*†], Valentina Gorobets[1†] and Andreas Kunz[1†]

[1]Innovation Center Virtual Reality, ETH Zurich, Leonhardstrasse 21, Zurich, 8092, Switzerland.

*Corresponding author(s). E-mail(s): eandreopo@ethz.ch;
Contributing authors: gorobets@iwf.mavt.ethz.ch; kunz@ethz.ch;
†These authors contributed equally to this work.

Abstract

Despite the high level of automation in industrial production, manual operations still play an important role and contribute significantly to the overall production costs. For the evaluation of these manual processes the "Methods-Time Measurement" (MTM) is widely used. This method is applied to real workplaces or mock-ups thereof, while also Virtual Reality (VR) can be used for the representation of such workplaces. However, the evaluation of the workers' performed actions is still done manually, which is a time-consuming and error-prone process. This paper introduces an approach to automatically detect full-body actions of users in VR and consequently derive the appropriate MTM values, without knowledge of a pre-existing workplan. The detection algorithm that was developed is explained in detail and its performance is analyzed through a user study with 30 participants.

Keywords: Methods-Time Measurement, Virtual Reality, Action Detection, Workplace Optimization

1 Introduction

Although industrial production processes are already highly automated, manual work continues playing an important role and is a significant factor towards the overall production cost. As an implication, a careful evaluation and optimization of such manual work processes is required [1, 2].

For evaluating manual work places, the work of Taylor [3] set the starting point towards the development of standardized evaluation methods. Originating from Taylor's work, many different methods were developed for the evaluation of manual work, as described by Caiza et al. [1]. One of the most widely used methods is the Methods-Time Measurement (MTM), which analyzes manual operations by dividing them into a sequence of so-called basic motions [4]. The time standard that is used for the MTM evaluation is referred to as "Time Measuring Units" (TMU), with one TMU being equivalent to 0.036 s. Thus, the granularity is fine enough for a detailed analysis [5].

Depending on the work process, several levels of detail of the MTM exist. The highest precision is achieved by the MTM-1, which offers 17 basic motions to decompose an overall more complex movement. Such a thorough analysis is mainly used for industrial mass production cases. For smaller production series, the analysis effort can be reduced by using the MTM-2, which only offers

9 basic movements. Thus, it is less precise, but the transcription effort is significantly lower. For even smaller production series, MTM-UAS or MTM-MEK can be used, which group a series of basic movements and are even faster to perform.

For an MTM evaluation, the workplace needs to be built or modeled by cardboard before the envisioned work can be performed. Thus, the evaluation of a new workplace is impossible without at least a physical mockup of it being developed first. When the environment is physically in place, the worker has to perform the planned tasks, while the process is being recorded. The videos are then analyzed by an expert, who manually transcribes all actions into the corresponding TMUs for the chosen MTM standard. However, this manual process is time-consuming, expensive, and error-prone. Moreover, using either cardboard models or the real workplace itself are options not well suited for small alterations, which would be necessary to optimize the workplace layout.

This paper introduces an algorithm that enables the automatic transcription of the performed tasks into the corresponding TMU values. Our approach relies on Virtual Reality (VR) to shift the early workspace planning into a digital form. Using VR automatically reduces the material waste of the cardboard modeling approach and allows for faster workplace alterations based on the MTM-2 evaluation. Additionally, using a Virtual Environment (VE) allows the integration of realistic components and tools which better resemble the real world compared to their cardboard alternatives.

After discussing about the related work on motion recognition and automatic MTM transcription in VR, this paper gives a detailed overview of the developed algorithm, followed by a description of the user study process. Next, the paper presents the analytic results of the study and concludes with a summary and an outlook on future work.

2 Related work

Tracking user movements has been of particular interest in VR, with the main objective being the enhancement of immersion. For instance, walking detection is important, since it can lead to more realistic navigation of the user in the VE. For the detection of walking, different approaches can be used, such as tracking the head movements as done by [6]. However, this and similar works like in [7–9] do not account for movements of the body in the VE and are usually based on a highly periodic pattern of walking, which is not necessarily observed in real walking scenarios where a displacement of the torso occurs. Moving towards real walking, Kunz et al. [10] showed that real walking can also be integrated in factory planning with the use of redirecting walking methods. This is considered as highly useful for planning and digital design of industrial facilities [11].

Similar to walking, also hand gesture recognition was researched. A survey by Sagayam et al. [12] give an overview of the different approaches. For this task contemporary works rely on computer vision and deep learning approaches [13, 14], which however require rich datasets for their training.

Only little work can be found specifically on the topic of automatic MTM transcription for industrial applications. The first work is from Bellarbi et al. [15], in which an algorithm was developed to automatically generate MTM-UAS codes. The proposed algorithm follows a decision tree-like structure, where all the MTM basic motions are classified in groups, among which are “eye movement”, “hand movement”, and “body movement”. These three nodes contain a certain expected sequence of data, including 3D positions, rotations, and time. However, the authors state that their approach suffers from false positives and false negatives when unrelated actions, e.g., scratching, interrupt the main intended action. This is a problem stemming from the algorithm’s reliance on hard-coded sequences of data to converge to a detection. Another piece of work for an automated MTM transcription is from Benter & Kuhlang [16]. Their approach relies on Microsoft Kinect’s full body tracking data, which does not track sufficiently the leg movements, as stated by the authors. Consequently, the aforementioned drawback does not allow the detection of actions such as “Walk” and “Foot Motion”. In addition, the users always have to stay in the limited field of view of Microsoft Kinect, which makes the evaluation of larger workspaces impossible. Lastly, the developed algorithm works with only a limited subset of MTM-1 actions, and basic motions such as “Reach”, “Grasp”, or “Position” are not considered.

3 Methodology

3.1 Algorithm development

The aforementioned algorithm from Bellarbi et al. [15] works on data that were recorded during a virtual MTM session. However, this comes to the cost of recording a vast amount of data, which actually prohibits analyzing longer work sequences. Therefore, our algorithm will perform a real-time detection of the actions through the incoming data from the VR equipment. The VR equipment that is utilised consists of the HTC Vive Pro Head Mounted Display (HMD), 2 controllers, 4 base stations and 3 HTC Vive Trackers. One Vive Tracker was positioned at the back side of the users' hip, with the other 2 placed on each foot each. Our detection algorithm thus relies on a 5-point tracking (2 feet, 2 hands, 1 hip), with the tracking data from the HMD not being used.

Our virtual MTM application was developed in Unity[1], with the SteamVR plugin used to interface with the HTC Vive equipment. In order to increase the realism of the workspace optimization process in VR, a full-body animation through an avatar was incorporated. The avatar was developed using "MakeHuman"[2], which is an open source software tool for the prototyping of photorealistic avatars. In order to match the avatars movements to the movements of the user, the "Final IK" Unity Asset[3] was utilised. With this, the avatar was animated in real time, based on the positions of the 5 tracked devices on the user. For a precise animation, the avatar needs first to be adjusted to each individual user. This is a quick process, which lasts less than a minute and takes place during the so-called "calibration phase" before each user starts their session.

As mentioned above, the MTM-2 standard was chosen for our work, which includes the actions *Get*, *Put*, *Apply Pressure*, *Crank*, *Eye Action*, *Foot Motion*, *Step*, and *Bend & Arise*. From these basic movements, *Eye Action* was discarded due to the missing eye tracking hardware. Similarly, the *Crank* action, referring to a circular movement of the hands for the rotation of an object like a steering wheel, was also not considered - simply to reduce the effort for the initial version of the algorithm. In the following section, all considered MTM-2 action detections will be described in more detail.

3.1.1 Step and foot motion

One of the main challenges for the detection of the *Step* and *Foot Motion* actions is distinguishing between intended foot movements and "nervous feet". Furthermore, the intended foot movements should be separated between *Step* and *Foot Motion*. The former is a considered an action with the intention of moving the whole body with respect to the environment, while the latter is a smaller motion of the feet which does not intend to the displacement of the body. The flowchart of the decision process is shown in Figure 1.

For every frame, the shown sequence of decisions takes place for each foot tracker. First, the absolute velocity for each foot tracker is calculated. If this velocity exceeds 0.2 m/s, a threshold which was experimentally determined, then the so-called "Foot Movement" state is entered. As soon as the decision tree enters the "Foot Movement" state, it is firstly checked whether this state was already active or not. This is done to ensure that foot-related actions are not constantly being transcribed during the motion process, but only once the movement is completed. If "Foot Movement" was previously inactive, then the starting position in space and the time are both stored in memory. On the other hand, if the "Foot Movement" state was already entered, then the current foot position is stored and the distance from the starting point is updated. Additionally, if this distance is greater than the maximum displacement that was recorded so far during the ongoing "Foot Movement", then the maximum displacement is updated to reflect this change. This process is repeated until the foot velocity goes below 0.2 m/s for a time period longer than 0.4 s. This time buffer was set to avoid triggering the "Foot Movement" again in the case that the user momentarily pauses the motion of their foot. As soon as "Foot Movement" is completed, the maximum foot displacement is stored as the distance traveled during the motion process. If that distance is greater than 0.2 m, then a *Step* is detected. Otherwise, if it less than 0.2 m but still greater than 0.05 m, a

[1] http://www.unity.com; (accessed March 2022)
[2] http://www.makehumancommunity.org; (accessed March 2022)
[3] http://root-motion.com; (accessed March 2022)

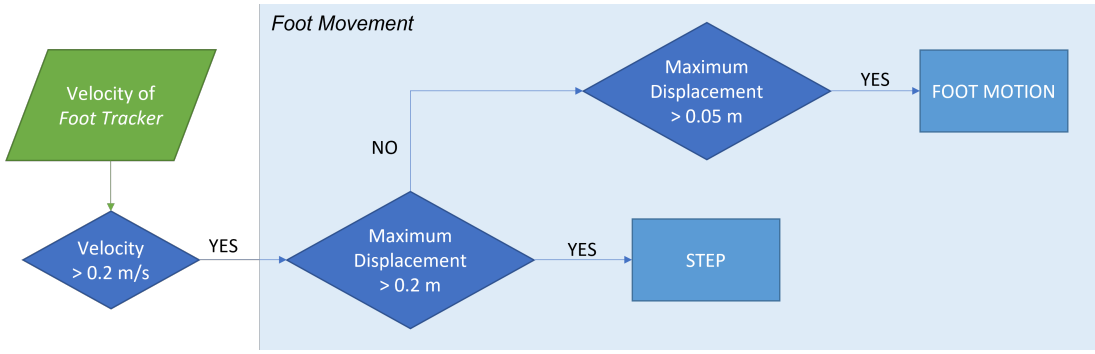

Fig. 1 *Step* and *Foot Motion* detection flowchart

Foot Motion is detected. All other foot motions are discarded as nervous foot movements.

3.1.2 Bend & arise action

Bend & Arise is the only action in MTM-2 that is not dependent on a single body part, but is essentially relevant to the entire upper body movement. For this reason, a combination of all the tracking objects was used for detecting this action. The flowchart of the decision process followed is shown in Figure 2. The detection process is based on initial calibration steps, where the distance between the hand controllers and the foot trackers is calculated for an upright posture. This gives the distance d_o shown in Figure 2.

After the calibration process, the same distance between the controllers and the foot trackers is continuously computed and stored as the current distance d_c. Then, the ratio R between d_c and d_o is calculated. If the ratio is less than a threshold of 70%, then the "Bend" state is entered. If this condition is not met, a second condition is evaluated, which uses the hip tracker to measure the forward bending angle. If this angle exceeds 30°, the "Bend" state is also entered. In order to deal with the edge case where the feet come closer to the hands – which should not be accounted as a "Bend" – an additional safety condition is applied, checking that the height of both the feet trackers is less than 20 cm. The 20 cm height threshold ensures that both feet are approximately on the ground, while leaving a safety margin for the placement of the foot trackers slightly higher than their expected position. The *Bend & Arise* action is finally transcribed as soon as the user returns to an upright position.

3.1.3 Get, regrasp & put actions

This section describes a set of actions, as they are mutually exclusive in their transcription. The *Get* action refers to an object being grasped by the user. Within MTM-2, this action can be further subdivided depending on the number of grasping actions undertaken and the weight/force applied. Since in our application the VR controllers are utilized, these two subcategories are discarded, and *Get* is only considered along with its "Distance Class" – which is the distance from the starting point of the hand's movement to the point where the grasping takes place. The *Regrasp* action refers to the user grasping again the same object that was previously in their hand. Finally, the "Put-Correction" is a special case of the *Put* action, which refers to immediate correction of an object's placement. For a "Put-Correction" to be detected by the algorithm, the user has to regrasp the object that was previously placed.

Figure 3 shows the inputs that are used for the detection of the aforementioned actions. First, the trigger of the controllers is used for grasping the objects. Hence, the first condition is to check whether the trigger is pressed or not. In order to avoid moving forward to the *Get* decision process when the trigger is accidentally pressed, an additional safety check is implemented that checks whether an interactable object is nearby. Apart from these conditions, a "Hand Movement" state was also implemented to manage the sequential relationship between the hand motion and the pressing of the trigger. This state is only entered when there was no previous hand motion and

[4] https://www.ikinema.com/docs/s317i325.html, (accessed March 2022)

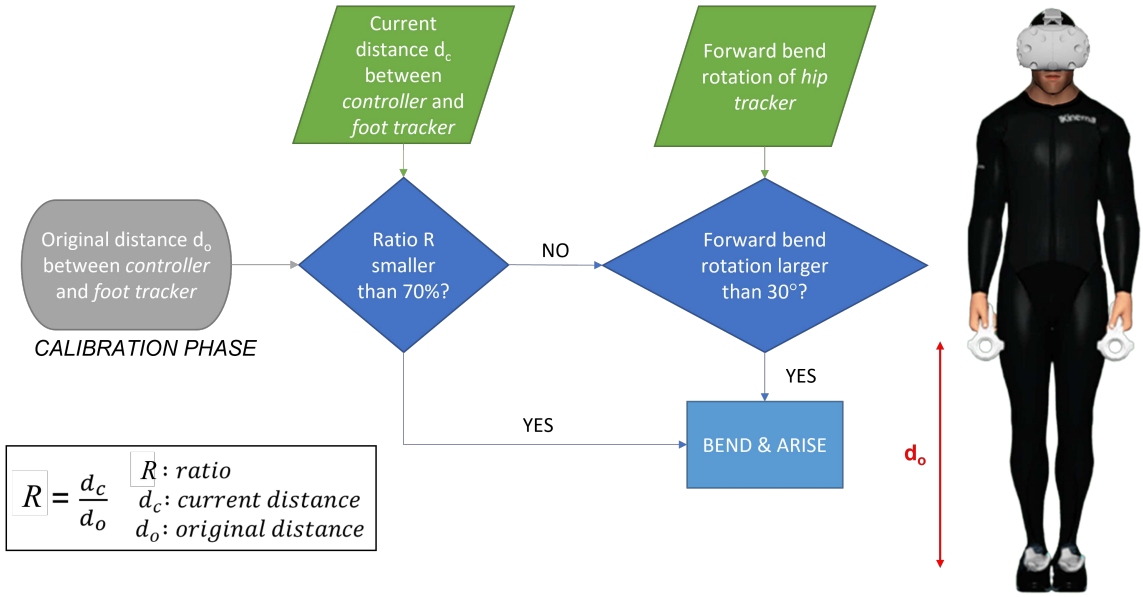

Fig. 2 *Bend & Arise* detection flowchart and body visualization[4]

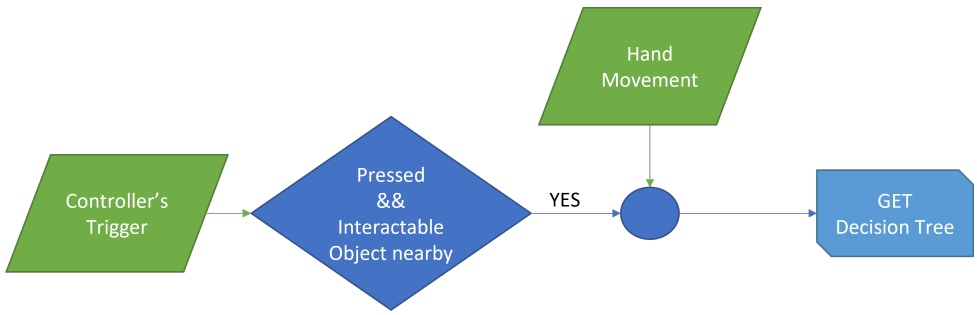

Fig. 3 *Get* detection input flowchart

there is no object that is already grasped. Furthermore, the controller's velocity has to exceed 0.1 m/s, and the time after the previous "Hand Movement" has to be larger than 0.2 s.

The first two conditions make sure that a "Get" state can be entered only when it is not already active. Then, the condition of the velocity threshold ensures that "Hand Movement" will be started when there is sufficient movement of the hand. Finally, the time buffer of 0.2 s ensures that there is some time for the user to press the trigger before a new "Hand Movement" is re-initiated. When the "Hand Movement" state is entered, the position of the controller is saved as the start position, which is later used for calculating the "Distance Class" for the *Get* action. In addition, the time is saved. For the "Hand Movement" state to be exited, the controller's velocity should drop below 0.1 m/s for more than 0.2 s to ensure that the user has not momentarily stopped the hand motion. Otherwise, if the velocity gets again higher than 0.1 m/s before 0.2 s of inactivity have passed then, the motion will continue to be associated with the previous "Hand Movement".

The previous conditions are forwarded to the "Get Decision Tree", which is shown in Figure 4, for it to decide which of the three actions should be transcribed. The forwarding of the inputs to the Get Decision Tree takes place when the trigger is pressed with an interactable object nearby. That is when the ongoing "Hand Movement" state is exited, and the current controller position is stored as the "Get Position". Additionally, the

time that has passed compared to the previous *Put* action is being computed and stored as the Get-to-Put time window. These two values are important decision parameters that will be used within the Get Decision Tree.

The Get Decision Tree follows a sequential process of conditional checks. The first conditional block is relevant to the detection of the "Put Correction" case associated to the previous *Put* action. For this, three conditions are evaluated. First, it is checked whether the grasped object is the same as the one from the previous *Put* action. Second, it is checked whether the same hand is grasping the object, and third, the action should not take place more than 1 s after the previous *Put* for it to be classified as a "Put Correction".

If the conditions for "Put Correction" are not satisfied, then the process will be checked for a "Regrasp" action. *Regrasp* shares the same first two conditions with the "Put Correction" case, since it also involves the grasping of the same object, while the hand stays relatively close to that object. The differentiating point between these 2 cases is the fact that *Regrasp* is not an action that is relevant to the time of occurrence, but mostly the movement of the user in space. Hence, the third condition checks whether the "Hand Movement" has started within a 0.35 m radius from the object after the previous *Put* action. If this is the case, then a *Regrasp* is transcribed. The 0.35 m distance threshold was decided after consultation by MTM experts.

If the conditional blocks for neither "Put Correction" nor *Regrasp* are fulfilled, then pressing the trigger is interpreted as a *Get* action. Since detecting a *Get* follows a modular approach, the algorithm can detect separate *Get* actions from the two hands that may simultaneously take place.

As shown in Figure 5, the *Get* action also includes the assignment of a "Distance Class", as well as the check for a special case named "Sequence of Get/Put". First, the "Distance Class" is calculated as the 3D Euclidean distance between the starting point of the associated "Hand Movement" and the spatial point when the trigger is pressed. For the "Sequence of Get/Put", a sequential grabbing of an object after the placement of another object nearby is checked, where it is assumed that the distance between the new *Get* and the previous *Put* is less than 0.3 m and the time passed is not more than 2 s. Also the case of passing an object from one hand to the other could be attributed to a "Sequence of Get/Put". This is simply done by checking whether the opposite controller is pressed concurrently with the one that was already pressed, and at the same time the two hands both attempt to grasp the same virtual object.

Detecting the *Put* action is much simpler than the *Get*, since it only checks whether the controller's trigger is released after a *Get* was already performed. Some parameters that are stored during the *Put* detection are the object that is placed and the time of placement. Moreover, the position of the controller when the trigger is released is also stored and used to calculate the "Distance Class" for the *Put*. This is the Euclidean distance between the position where the trigger was pressed (*Get*), and the currently saved position when the trigger is released (*Put*).

3.1.4 Apply pressure

Apply Pressure refers to the pressing of a button in the VE. The way that the interaction system was set up for the user to interact with virtual buttons was through the trackpad buttons of the controllers. These are big circular buttons that are placed at the top side of the controllers. The algorithm checks whether the trackpad buttons are pressed, while the hand is in a close proximity to a virtual button. If these two conditions are met for either of the two controllers, an *Apply Pressure* is transcribed.

3.2 User study

The main goal of the user study is to evaluate the performance of the algorithm in detecting the undertaken actions and transcribing them to the appropriate MTM values.

3.2.1 Technical setup

The VE was designed with a large walkable area to allow for plenty of *Steps* and *Foot Motions* to be performed. Furthermore, in order to measure the robustness of the grasping-related action recognition, objects with various sizes and geometries were placed within the VE, as shown in Figure 6. In detail, there are 2 tables, 2 boxes, 1 box cover, 2 hammers, 2 wrenches, 4 cubes of variable sizes, a target area for the cubes, 1 button, 3 screws, 1

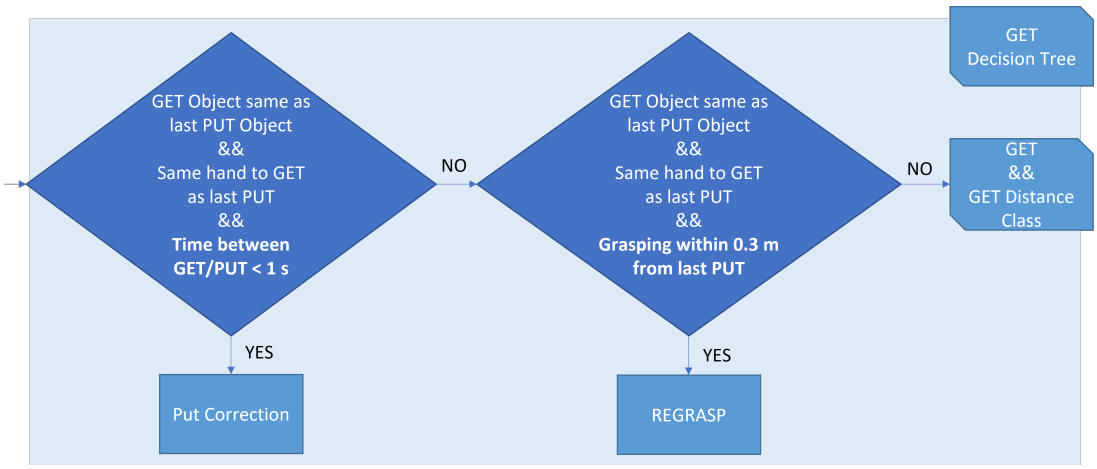

Fig. 4 *Get* decision tree

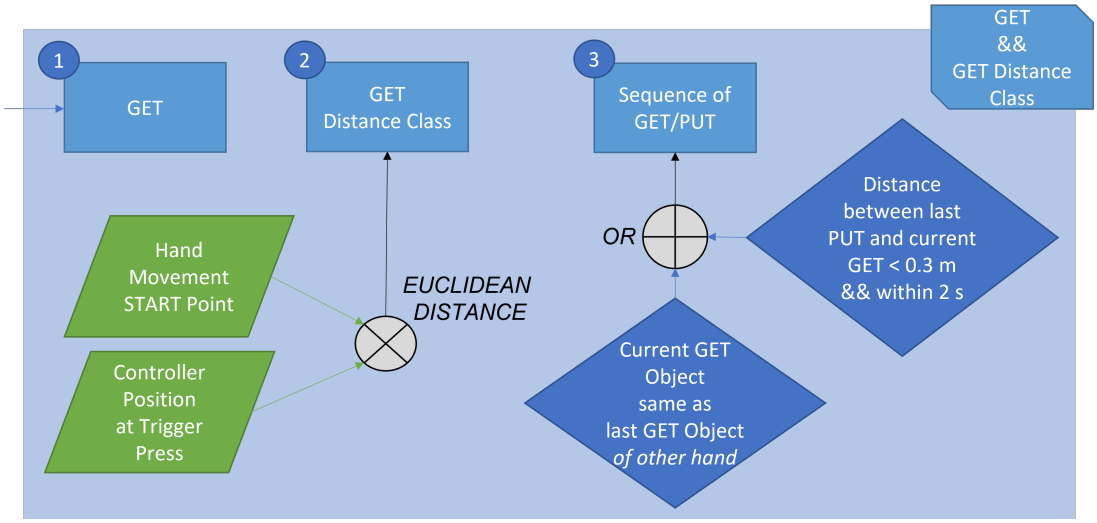

Fig. 5 *Get* action detection and extra features

pedal, 1 pallet and a start-position mark on the floor. From these, the 2 tables, the cubes' target area, the start-position mark, and the pallet are non-interactable objects, meaning that the users cannot alter their position. Some extra objects were also placed in the VE to increase realism, such as a fire extinguisher, some shelves and a bulletin board. However, these objects were not relevant for the user study.

3.2.2 User study design

The user study was designed to have a duration of approximately 30 minutes. Initially, the context and the purpose of the study was briefly explained to the participants. Then, they were instructed to fill out a pre-study questionnaire with demographic- and background-related questions, as well as their acquaintance with VR equipment. Further, a Simulator Sickness Questionnaire (SSQ) [17] was completed.

Before the actual user study takes place, the users had to perform a trial run to get familiar with the virtual environment and the controllers. The trial run consisted of all the interactions which would be required during the actual experiment, such as grasping and placing of objects, passing objects between the two hands and pressing of a button.

Fig. 6 The virtual environment for the user study

Fig. 7 Scene view of the user study

As soon as the trial run was completed, the users entered the actual run of the study, in which data got recorded. This started with a "calibration phase", where the avatar was adjusted to fit more realistically each user's body. Additionally, the vertical distance between the controllers and the feet trackers for both sides was stored, which is utilized as already mentioned for the *Bend & Arise* detection.

During the study, the tasks that the users were required to perform, were verbally communicated by the study facilitator in a step-by-step manner. These consisted of a variety of tasks which would enable the detection of all the actions under consideration. Throughout the time in which the participants were performing the instructed tasks, the VR scene was recorded, with the view shown in Figure 7. This included a virtual dashboard with the results of the automatic transcription of all the actions being displayed in real-time. Through the comparison of these values with the motion of the avatar, the automatic transcription can be manually evaluated offline, after all the user studies took place.

As soon as the users completed the tasks, they removed the VR equipment and filled out a post-study questionnaire. The questionnaire consisted of a second SSQ, a task load index SIM-TLX [18], and questions from the System Usability Scale (SUS) [19].

3.2.3 User study participants

30 participants took part in the user study, mainly recruited from the university staff and students. From them, 20 were male and 10 were female. The average age was 28 with a SD = 11.1 years. 23.3% of the participants were wearing glasses during the user study, 10% contact lenses, and 66.7% had normal vision.

4 Results

This paragraph will present the results of both - the questionnaires and the transcription algorithm.

4.1 SSQ results

The SSQ reveals how certain feelings of the users change after performing the tasks in VR. The responses to the questions were given on a Likert scale between 0 and 3. As it can be seen from Table 1, there is no significant difference in the answers before and after the VR session, which means that the users did not show any extreme reactions due to the user study.

4.2 Task load index results

The outcomes of the SIM-TLX questionnaire show that the user study was considered neither physically nor mentally challenging. Furthermore, the users considered that they were successful in performing the instructed tasks, meaning that the shift to VR did not influence their morale in a negative way. In addition, the presence-related questions showed that the user experience was considered as particularly immersive, which is an important factor to ensure that the MTM evaluation in VR is comparative to the real-world scenario.

Table 1 Results of the SSQ

	General discomfort	Fatigue	Headache	Eye strain	Difficulty focusing	Sweating	Nausea	Difficulty concentrating	Dizziness eyes open	Blurred vision	Dizziness eyes closed	Stomach discomfort	Burp
before	0.23	0.5	0.27	0.4	0.33	0.23	0.07	0.33	0.1	0.17	0.13	0.07	0.03
after	0.37	0.4	0.17	0.3	0.17	0.2	0.27	0.2	0.2	0.17	0.27	0.23	0.07
Δ	0.13	-0.1	-0.1	-0.1	-0.17	-0.03	0.2	-0.13	0.1	0	0.13	0.17	0.03

4.3 Statistical metrics and manual data transcription

In order to measure the performance of the automatic MTM-2 transcription, the results of a manual transcription are used as ground truth and then compared to the results of the algorithm. This comparison was done trough the calculation of the *True Positives* (TP), the *False Positives* (FP), and the *False Negatives* (FN) for all of the considered MTM-2 actions. *True Positives* are the actions that were actually done by the user and were also successfully detected by the algorithm. *False Positives* are the actions that were detected by the algorithm without the user actually performing them. *False Negatives* are the actions that were done by the user, but not detected by the algorithm. These three metrics are used extensively in literature for the evaluation of prediction tasks, and thus are also used here to make the results easily comparable with similar works. Additionally, they can be used for the calculation of two important metrics, namely the *Precision* and *Recall* of the detections. The metrics *Precision* and *Recall* are defined as:

$$Precision = \frac{\sum TP}{\sum (TP + FP)} \tag{1}$$

$$Recall = \frac{\sum TP}{\sum (TP + FN)}. \tag{2}$$

Precision is the ratio of the detected actions which are actually relevant, since it compares the correct detections to the sum of correctly and incorrectly detected actions. A high *Precision* means that the algorithm has not falsely detected many actions that did not take place. On the other hand, *Recall* says how many of the actually performed actions are detected by comparing the detected actions to the sum of the detected and undetected actions. A high *Recall* means that only a few of the performed actions were missed by the algorithm.

For all the individual actions, the cumulated TPs, FPs, and FNs are listed in Figures 8, 9, 10, and 11, together with the calculated *Precision* and *Recall* values. The colors represent the quality of the results and easily show where there is still room for improvement.

Bend & Arise			Step			Foot Motion		
TP	FP	FN	TP	FP	FN	TP	FP	FN
176	29	2	3060	59	26	1184	51	23
Precision	Recall		Precision	Recall		Precision	Recall	
0,859	0,989		0,981	0,992		0,959	0,981	

Fig. 8 Transcription results for lower and main body actions - *Bend & Arise, Step, Foot Motion*

Get			Regrasp			Put		
TP	FP	FN	TP	FP	FN	TP	FP	FN
536	21	4	106	24	3	643	39	9
Precision	Recall		Precision	Recall		Precision	Recall	
0,962	0,993		0,815	0,972		0,943	0,986	

Fig. 9 Transcription results for hands actions - *Get, Regrasp, Put*

Put Correction			Sequence Get-Put			Apply Pressure		
TP	FP	FN	TP	FP	FN	TP	FP	FN
61	58	1	69	13	2	63	0	0
Precision	Recall		Precision	Recall		Precision	Recall	
0,513	0,984		0,841	0,972		1	1	

Fig. 10 Transcription results for hands actions - *Put Correction, Sequence Get-Put, Apply Pressure*

Pedal Presses				
TP	FP	FN	Average Steps	Average Foot Motions
33	12	1	1,6	0,667
Precision	Recall			
0,733	0,971			

Fig. 11 Transcription results for pedal press

5 Discussion

It is evident that the results of the transcription algorithm are very notable, with most of the action detections exceeding 95% for *Precision* and 97% for *Recall*. The high *Recall* numbers indicate that the algorithm in general does not miss actions from the user. However, it is sometimes overly sensitive to certain actions, which is reflected in the *Precision* results. In particular, the actions related to the leg trackers, i.e., *Step* and *Foot Motion*, show remarkable results for both *Precision* and *Recall*. This means that incorporating a "Foot Movement" state with given time and velocity thresholds was successful towards separately detecting the actions of each foot. It is important to note that a large number of FPs of these two actions originates from the "Pedal Presses", specifically resulting in 48 out of the 59 FPs for *Step*, and 20 out of 51 FPs for *Foot Motion*. This means that if these two actions were deactivated when the pedal was pressed, then the *Precision* for *Step* detection would rise from 98.1% to 99.2%, and for *Foot Motion* from 95.9% to 97.4% respectively. Hence, the *Step* and *Foot Motion* detection method on its own constitutes a very powerful concept, but some more work needs to be done to properly integrate the logic of the "Pedal Press" to the overall detection framework. Additionally, in order to make *Foot Motion* less sensitive in its detection and further eliminate its FPs, the distance threshold of 0.05 m could be slightly increased. Finally, very few times there have been FPs because the trackers momentarily lost connection to the base stations due to potential occlusions, leading to a small displacement of their position, which did not actually take place in reality.

As for the *Bend & Arise* action, the detection is also very good with a *Precision* of 85.9% and a *Recall* of 98.9%. It is worth noting that for this action 17 out of total 29 FPs occurred to 3 out of the 30 users, due to occlusions of the trackers. Hence, if these occlusions were avoided, the performance of *Bend & Arise* would improve drastically.

Moving to the hand-related action detections, almost all FPs relevant to *Get*, *Put*, *Regrasp* and the special classes "Put-Correction", and "Sequence of Get/Put" are due to a mismatch in the interfacing between the detection algorithm and the SteamVR Interaction system. This caused the user to see the highlighting of an object slightly earlier than the detection algorithm considered the object as ready-to-grasp. This can be easily corrected in future work.

In spite of the aforementioned problem, the performance of both *Get* and *Put* detections remained at very high levels, with the *Precision* ranging at 95% and the *Recall* at 99%. The *Regrasp* action was affected slightly more by this issue, because many times the users did not approach the objects close enough during the second grasping. Despite that, the *Precision* remained at 84.1%, and the *Recall* at 97.2%. On the other hand, the "Put Correction" transcription was affected the most from this inconsistency, with the *Precision* dropping to 51.3%. This comes from the fact that for a "Put Correction" an instantaneous regrasping of the object that was previously placed is required, and the early object-highlighting led many times the users to not go all the way back to the object's position. Finally, the *Precision* and *Recall* for the detection of the "Sequence of Get/Put" special case were 84.1% and 97.2%, respectively, which are quite satisfactory numbers.

As for the *Apply Pressure* detection it can be seen that the approach followed brought perfect results for both *Precision* and *Recall*. This can be attributed to the use of Unity tags to differentiate the buttons from the rest of the interactable objects. As such, the detections remained robust against potential erroneous actions from the users, such as confusing the controller inputs. Pedal Presses were detected with a *Precision* of 73.3% and a *Recall* of 97.1%. As visible from the results, the average number of *Step* actions detected during a Pedal Press is 1.6, while for *Foot Motion* 0.7. This means that when users were instructed to press the virtual pedal, they were more frequently approaching the pedal from further away distances, resulting in larger leg movements.

6 Conclusion and future work

In this paper, an automatic MTM-2 transcription algorithm for a work process within a virtual environment was introduced and evaluated with a corresponding user study. The algorithm is based on the spatial input data of a VR tracking system consisting of a Head Mounted Display, 2 controllers, 1 hip tracker, and 2 feet trackers. The tracking signals were also used to animate the avatar through inverse kinematics to increase immersion for the user.

For the user study, a workflow was designed including actions that are expected to take place in a real workplace. These actions were then automatically detected by the developed algorithm in real-time, which considered most of the MTM-2 basic motions, such as *Get*, *Put*, *Apply Pressure*, *Foot Motion*, *Step*, and *Bend & Arise* actions. As the results from the user study show, the introduced algorithm works remarkably well, with most of the action detections exceeding 95% for *Precision* and 97% for *Recall*. Some lower numbers were only detected for the hand-related actions, which mainly stem from an inconsistency between the algorithm and the interaction system in VR. Hence, it is considered that with some minor modifications the algorithm can achieve an even higher *Precision*, setting a promising transcription framework for the MTM-2 analysis of virtual workplaces.

Future work will tackle the mentioned inconsistency between the algorithm and SteamVR, but also focus on further improving the algorithm itself, such as for the detection of Pedal Presses. Another future extension will be the implementation of a dataset capturing system. Such a recorded dataset could be used to replay a user's session in the virtual environment and allow experts to see in a first-person view the actions that are being performed. This could be an interesting feature for MTM experts to evaluate first-hand the efficiency of the overall workflow, even while being in a remote location.

Finally, the extension of the algorithm to support the automatic transcription of the actions in a real workplace needs to be investigated. This would require certain modifications to both the hardware and the detection process of certain actions, such as grasping. The controllers cannot be used in a real-world setting, therefore motion capture gloves could be considered. Another option would be to track and detect hand-related actions via cameras. All these and other options need to be further evaluated in order to decide which is an optimal approach to extend the algorithm to the real-world case.

Acknowledgments. The authors want to thank Acél & Partner AG for their continuous support in all MTM-related questions.

Declarations

Funding This work is funded internally by ETH Zurich.
Conflicts of interest/Competing interests The authors declare no competing interests.
Availability of data and material Not applicable.
Code availability Not applicable.
Ethics approval The user studies were performed under the Swiss ethics declaration BASEC Req-2022-00678.
Consent to participate Not applicable.
Consent for publication The authors approve the publication of the obtained results.
Authors' contributions These authors contributed equally to this work.

References

[1] Caiza, G., Ronquillo-Freire, P.V., Garcia, C.A., Garcia, M.V.: Scoping review of the work measurement for improving processes and simulation of standards. In: International Conference on Information Technology & Systems, pp. 543–560 (2021). https://doi.org/10.1007/978-3-030-68285-9_51. Springer

[2] Andrianakos, G., Dimitropoulos, N., Michalos, G., Makris, S.: An approach for monitoring the execution of human based assembly operations using machine learning. Procedia Cirp **86**, 198–203 (2019). https://doi.org/10.1016/j.procir.2020.01.040

[3] Taylor, F.W.: The Principles of Scientific Management. Harper & brothers, New York (1919)

[4] Maynard, H.B., Stegemerten, G.J., Schwab, J.L.: Methods-time measurement (1948)

[5] Gorobets, V., Holzwarth, V., Hirt, C., Jufer, N., Kunz, A.: A vr-based approach in conducting mtm for manual workplaces. The International Journal of Advanced Manufacturing Technology **117**(7), 2501–2510 (2021). https://doi.org/10.1007/s00170-021-07260-7

[6] Caserman, P., Krabbe, P., Wojtusch, J., von Stryk, O.: Real-time step detection using the integrated sensors of a head-mounted display. In: 2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC), pp. 003510–003515 (2016). https://doi.org/10.1109/SMC.2016.7844777. IEEE

[7] Tregillus, S., Folmer, E.: Vr-step: Walking-in-place using inertial sensing for hands free navigation in mobile vr environments. In: Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems, pp. 1250–1255 (2016). https://doi.org/10.1145/2858036.2858084

[8] Bhandari, J., Tregillus, S., Folmer, E.: Legomotion: Scalable walking-based virtual locomotion. In: Proceedings of the 23rd ACM Symposium on Virtual Reality Software and Technology, pp. 1–8 (2017). https://doi.org/10.1145/3139131.3139133

[9] Lee, J., Ahn, S.C., Hwang, J.-I.: A walking-in-place method for virtual reality using position and orientation tracking. Sensors **18**(9), 2832 (2018). https://doi.org/10.3390/s18092832

[10] Kunz, A., Zank, M., Fjeld, M., Nescher, T.: Real walking in virtual environments for factory planning and evaluation. Procedia Cirp **44**, 257–262 (2016). https://doi.org/10.1016/j.procir.2016.02.086

[11] Mujber, T.S., Szecsi, T., Hashmi, M.S.: Virtual reality applications in manufacturing process simulation. Journal of materials processing technology **155**, 1834–1838 (2004). https://doi.org/10.1016/j.jmatprotec.2004.04.401

[12] Sagayam, K.M., Hemanth, D.J.: Hand posture and gesture recognition techniques for virtual reality applications: A survey. Virtual Reality **21**(2), 91–107 (2017). https://doi.org/10.1007/s10055-016-0301-0

[13] Oudah, M., Al-Naji, A., Chahl, J.: Hand gesture recognition based on computer vision: A review of techniques. journal of Imaging **6**(8), 73 (2020). https://doi.org/10.3390/jimaging6080073

[14] Sun, J.-H., Ji, T.-T., Zhang, S.-B., Yang, J.-K., Ji, G.-R.: Research on the hand gesture recognition based on deep learning. In: 2018 12th International Symposium on Antennas, Propagation and EM Theory (ISAPE), pp. 1–4 (2018). https://doi.org/10.1109/ISAPE.2018.8634348. IEEE

[15] Bellarbi, A., Jessel, J.-P., Da Dalto, L.: Towards method time measurement identification using virtual reality and gesture recognition. In: 2019 IEEE International Conference on Artificial Intelligence and Virtual Reality (AIVR), pp. 191–1913 (2019). https://doi.org/10.1109/AIVR46125.2019.00040. IEEE

[16] Benter, M., Kuhlang, P.: Analysing body motions using motion capture data. In: Nunes, I.L. (ed.) Advances in Human Factors and Systems Interaction, pp. 128–140. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-20040-4_12

[17] Kennedy, R.S., Lane, N.E., Berbaum, K.S., Lilienthal, M.G.: Simulator sickness questionnaire: An enhanced method for quantifying simulator sickness. The international journal of aviation psychology **3**(3), 203–220 (1993). https://doi.org/10.1207/s15327108ijap0303_3

[18] Harris, D., Wilson, M., Vine, S.: Development and validation of a simulation workload measure: The simulation task load index (sim-tlx). Virtual Reality **24**(4), 557–566 (2020). https://doi.org/10.1007/s10055-019-00422-9

[19] Slater, M., Usoh, M., Steed, A.: Depth

of presence in virtual environments. Presence: Teleoperators & Virtual Environments **3**(2), 130–144 (1994). https://doi.org/10.1162/pres.1994.3.2.130