Human Movement Direction Classification using Virtual Reality and Eye Tracking

Available online at www.sciencedirect.com

ScienceDirect

Procedia Manufacturing 51 (2020) 95–102

Procedia MANUFACTURING

www.elsevier.com/locate/procedia

30th International Conference on Flexible Automation and Intelligent Manufacturing (FAIM2021)
15-18 June 2021, Athens, Greece.

Human Movement Direction Classification using Virtual Reality and Eye Tracking

Julius Pettersson[a,*], Petter Falkman[a]

[a]*Department of Electrical Engineering, Chalmers University of Technology, 412 96 Göteborg, Sweden*

Abstract

Collaborative robots are becoming increasingly more popular in industries, providing flexibility and increased productivity for complex tasks. However, the robots are not yet that interactive since they cannot yet interpret humans and adapt to their behaviour, mainly due to limited sensory input. Rapidly expanding research fields that could make collaborative robots smarter through an understanding of the operators intentions are; virtual reality, eye tracking, big data, and artificial intelligence. Prediction of human movement intentions could be one way to improve these robots. This can be broken down into the three stages, **Stage One:** *Movement Direction Classification*, **Stage Two:** *Movement Phase Classification*, and **Stage Three:** *Movement Intention Prediction*. This paper defines these stages and presents a solution to **Stage One** that shows that it is possible to collect gaze data and use that to classify a person's movement direction. The next step is naturally to develop the remaining two stages.

Keywords: virtual reality; eye tracking; movement classification; collaborative robots; human intention prediction

1. Introduction

Collaborative robots are becoming increasingly more popular in industries [8]. The advantages of having humans and robots in the same workspace interacting with each other are many, such as; increased flexibility [17] and increased productivity for complex tasks [17]. However, the robots are not yet that interactive since they cannot yet interpret humans and adapt to their swift changes in behaviour in a way that another human would do. The main reason is that the collaborative robots today are limited in their sensory input, which makes the human responsible to stay out of the way. Human intention prediction can be achieved using camera images and probabilistic state machines [4] with the goal of determining between explicit and implicit intent. Other ways are to monitor the gaze to predict an upcoming decision [13] or analyze bioelectric signals, such as electromyography, to predict human motion [5]. Other fields that have been rapidly expanding and could make collaborative robots smarter through an understanding of the operators behaviour and intentions are; virtual reality, eye tracking, gathering and management of large datasets, and artificial intelligence.

A way to gather insight in to how a person is reasoning is to measure and analyze where the person is looking [15] and the technique of doing this is called eye-tracking (ET). It is, for example, possible to gain insight into which alternatives the person is considering or which strategy a person is using while doing a task based on what a person is looking at. ET has, for example, been used in an industrial context to; use the gaze as the input for machine control [14], analyze industrial visualization of information [31], and evaluate new ways to facilitate human–robot communication [28].

Virtual Reality (VR) can be described as a technology through which visual, audible and haptic stimuli is able to give the user a real world experience in a virtual environment [7]. Benefits such as being able to provide more relevant content and present it in a suitable context [25] are reasons to promote the use of VR. It can, for example, be used; when making prototypes [1], to train operators in assembly [2], and improve remote maintenance [3].

* Corresponding author, *pjulius@chalmers.se*. This work has been supported by UNIFICATION, Vinnova, Produktion 2030.

10.1016/j.promfg.2020.10.015

The use of modern technologies such as ET and VR makes it possible to collect larger amounts of data, with higher accuracy and at a higher pace than before [24]. These large volumes of data, created at high speed, and with great variety [20] is referred to as Big Data. One area of artificial intelligence that can be used to process these huge datasets is called deep machine learning [26]. Big data and artificial intelligence has been shown to be important tools for the future to improve industrial manufacturing [21, 22].

Combining these areas to increase the intelligence of the collaborative robots can be broken down into the following three stages:

Stage One: *Movement Direction Classification*
Deep machine learning requires large amount of data to train the neural networks. The first step is therefore to create a virtual, measurable environment that is capable of gathering all the necessary data. The environment has to limit distractions and ambiguous stages to ensure that it is possible to evaluate any results and draw conclusions using domain knowledge. The end goal of the first stage is to be able to classify the movement direction of the test participant upon completion of the test.

Stage Two: *Movement Phase Classification*
The goal of the second stage is to improve the classification to be more precise. This means that the network should be able to classify the phase of a movement i.e. when there is; no movement, the beginning of a movement, an ongoing movement, and the end of a movement. The increased complexity of this second task requires that the environment developed in the first stage also is able to capture hand movements during the entire test sequence.

Stage Three: *Movement Intention Prediction*
Finally, the third stage is to be able to predict the intended movement direction of a test participant ahead of time, building on the ability to classify different phases of a movement from the second stage. This is a key component to be able to utilize the intended functionality in real world applications such as collaborative manufacturing. The implementation of a real world application could be done using the same ET technology mounted to the safety glasses that the operator already wears.

The present paper aims to present the first stage that contains; the development of the test environment, the conduction of a pilot test study to collect data, and results that verifies that the test case is working as intended. The paper is organized as follows; backgrounds to the main areas of this project, ET, VR, convolutional neural networks (CNN:s), dropout, and uncertainty estimation (UE), are presented in Section 2. The VR and ET hardware, HTC-Vive and Tobii ET, and the modelling software, Unity, that has been used is described in Section 3 and Section 4 explains the development of the VR test environment. The description of the VR test execution and the data collection is provided in Section 5 and is followed by Section 6 that introduces the obtained dataset and the selection of features. Section 7 presents the neural network architecture and the classification results. Finally, a brief discussion and the conclusions can be seen in Section 8 and Section 9 respectively.

2. Background

This section gives a background to eye-tracking (ET), virtual reality (VR), convolutional neural networks (CNN:s), dropout and uncertainty estimation (UE).

2.1. Virtual reality

A device that is used to visualize the VR environment (VRE) to the user is called a head mounted display (HMD) [7]. The HMD is, according to [7], equipped with sensors that measure the user's head motions and also a display which is responsible of providing the user with the visual content. The system is also providing the user with audible and haptic stimuli to immerse the user in a real world experience [7] of the virtual environment.

2.2. Eye tracking

The eye gaze is an interesting biological marker because it is possible to analyze underlying neurophysiology based on the movement of the eyes [10]. Tracking gaze is therefore an appealing test method due to that insight and also because it is objective, painless, and noninvasive [10]. There are different types of eye movement, namely; fixation, saccade and mixed, which describe the coordinate of the pupil, quick movements from point to point and the relation between these [18]. The method used to measure these movements are called temporal, spatial and count methods, which analyze the gaze duration, time between each movement and the traveled distance of the gaze.

2.3. Convolutional neural networks

CNN:s are a type of artificial neural networks that are more robust to shift, scale, and distortion invariance [19] than fully connected networks, and are therefore better at detecting spatial and temporal features. This is achieved by convolving or sub-sampling the input to the layer with local receptive fields [19] (filters) of a given size [n x m]. Each filter has $n \cdot m$ number of trainable weights + a trainable bias and these are shared [19] for all filter outputs.

2.4. Dropout

Dropout is a deep machine learning method that is used to reduce overfitting [11]. This is done by randomly ignoring, with probability p, each neuron in a network every time a training case is presented to the network. The goal of randomly excluding some neurons for every training case is to make sure that the network learns generalized features instead of a co-adaptation between neurons [11]. The probability to be used for fully connected layers, suggested by [11], is $p = 0.5$.

2.5. Dropout as a Bayesian Approximation

The dropout method described above can, according to [9], be used to approximate Bayesian inference. This is done by enabling dropout at all times, not only during the training of the network, which means that the network will randomly omit some neurons also when making predictions causing variation. The mean prediction as well as the model uncertainty can be obtained by making N number of predictions [9] on the same data and collect the results. [9] claims that $N \in [10, 1000]$ should give reasonable results. Using this approach is useful since it gives a way to reason about model uncertainty that is easy to implement and less computationally expensive [9] than alternative methods. [9] suggests that the probability p for dropping a neuron should be in the range of $p \in [0.1, 0.5]$.

3. Experimental setup

This section describes the different components used in the developed system.

3.1. The VR-equipment

The VRE will be visualized using the HMD, together with the two hand-held controllers, that are part of the VR-kit, "*HTC-Vive*" [12]. HTC-Vive is a consumer-grade VR-system that consists of one HMD, two hand held controllers and two base stations. In the HMD, two images are shown to the user, one for the left eye and one for the right eye. The images show the same view that is slightly shifted between the two, such that they together create a stereoscopic image. The HMD and the hand held controllers are equipped with sensors that can be used for 360° motion capture [12]. The sensors in the HMD makes it possible to change the view in the virtual world in the same way that the user moves his or her head. The VR-system is tracking the position and the orientation of the HMD and the hand held controllers in the room. This is achieved using the two base stations and the "on-board" sensors [12].

3.2. Eye tracking-equipment

"*Tobii Eye Tracking VR Devkit*" [29] is an ET solution based on the HTC-Vive HMD. ET sensors are implemented behind the lenses inside the HMD, one for each eye, together with ten illuminators each. The information from the sensors is handled in the *Tobii EyeChip* located in the HMD which allows the ET calculations to be done without using the CPU on the host computer and the Tobii Pro SDK can be used to access the data.

The eye gaze is tracked with the *Binocular dark pupil tracking* and the output frequency is 120 Hz [29]. The ET can be performed in a 110° trackable field of view, which is the same field of view as in the HTC-Vive HMD [29], with an estimated accuracy of 0.5° and a delay of approximately 10 ms from the illumination of the eye to that the data is available in the SDK [29].

3.3. Software for virtual modelling

The 3D components in this project were created using the modelling software Blender [6] and the models are implemented in a VRE using Unity, a game creation engine [30]. Unity has support for both Tobii ET as well as VR through Tobii Pro SDK and Steam VR SDK respectively. It also supports custom written scripts in C# that makes it possible to implement all the desired functionality from the SDK:s as well as the intended test logic.

4. Development of VR test environment

The VRE designed to collect the data consists of four stages; language selection where the test participant selects whether the written instructions in the VRE should be given in Swedish or English, calibration of the eye-tracking sensors, an information form where the participant enters age and gender, and the last stage is the test itself. The test stage, Fig. 1, features a table in the form of half a circle where cubes will appear at random in 5 different zones.

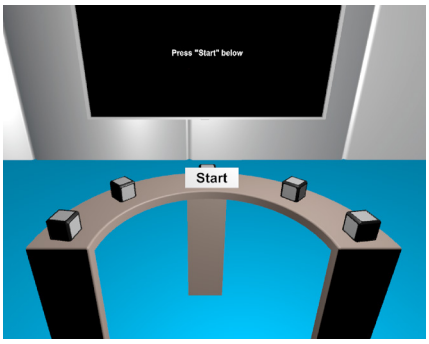

Fig. 1: An overview of the layout of the VR test environment.

The test stage has been designed in a way that is meant to force the test participant to look in the direction of the cube, make a movement towards the cube, and acknowledge that movement by touching the cube. The test has random selection of cubes and their positioning to make the test more unpredictable for the participant as well as a time delay between each cube appearing that helps to slow down the pace of the test execution. This is achieved by:

- A 45 degree spacing of the cubes makes for 5 zones in the half circle around the participant, labeled as 5 different movement directions.
- The cubes appear at arms length distance and requires the test person to touch it and simultaneously press a button to make the cube disappear.

- The zone that gets a cube is randomly selected every time a new cube is to be created.
- The positioning of the cube is also randomized in the interval $x \in [-x_s, x_s], y \in [-y_s, y_s]$, where x_s, y_S are the maximum deviations allowed around the center of each zone.
- After a cube has disappeared there is a time delay of 1s before the next cube appears.

The test is launched when the test participant presses the start button in the environment. Data is then collected during the time between two pressed cubes and saved as one data sample. This means that data is collected also during the time delays between the cubes, which is important since this time period could potentially include gaze data that shows strategies used to quickly find the next cube when it is shown.

The data that is collected from each test participant, each test and at every point in time is summarized in Table 1. This includes an anonymous participant ID, age, gender, the language that was used, as well as the date and time when the data was gathered. The test specific information in this case is the number (1-5) of the box that was clicked. The parameters obtained from the eyes at each timestamp, separate for left and right eye, are: eye gaze direction vector (EyeDirection), the coordinate in the virtual room where the gaze hits (EyeHitpoint), which object is gazed upon (EyeHitObject) as well as the size and position of the pupil. The head specific data that is collected are the position (HeadPosition) and rotation (HeadRotation), and the same data is also obtained from the two controllers that are held one in each hand (ControllerPosition, ControllerRotation).

Table 1: Table of data parameters.

Type	Parameter
Participant information	ID
Participant information	Age
Participant information	Gender
Participant information	Date & time
Participant information	LanguageIsEnglish
Test specific	BoxClicked
Test specific	Timestamp
ET	EyeDirection [x, y, z], (left, right)
ET	EyeHitpoint [x, y, z], (left, right)
ET	EyeHitObject (left, right)
ET	PupilPosition (left, right)
ET	PupilDiameter (left, right)
HMD & Controllers	HeadPosition [x, y, z]
HMD & Controllers	HeadRotation [x, y, z]
HMD & Controllers	ControllerPosition [x, y, z], (left, right)
HMD & Controllers	ControllerRotation [x, y, z], (left, right)

5. Description of test execution

The majority of the data was collected during the opening of a new facility at Chalmers University of Technology in Gothenburg. The test was performed by visitors of the opening in an open environment in the middle of the facility. All test participants were given the same instructions that are described below.

5.1. Instructions given to participants

The instructions for putting on the headset, performing the calibration of the eye tracking sensors, entering the required information, and starting the test was given as follows:

1. Put on the headset and adjust it such that the displays are centered in front of the eyes.
2. Receive a controller in the right hand. The controller is used to navigate the menus (using the laser pointer), destroying cubes during the test, and acknowledge all actions using the click function of the touchpad.
3. Now it is time to:
   (a) Choose the desired language, either Swedish or English by pressing the corresponding flag.
   (b) Calibrate the eye-tracker:
      i. Stand still with your head pointing forwards.
      ii. Press “Calibrate” using the laser pointer.
      iii. Focus your gaze on the red dots that appear on the screen until they disappear, without moving your head.
      iv. Press “Done” when the calibration is complete.
   (c) Enter the required information using the laser pointer.

Instructions for how to do the test follow below:

1. Press the “Start”-button on the screen by reaching towards it and touching it.
2. Use the controller to touch the cubes and press the touchpad while doing so to destroy them.
3. Keep destroying cubes until the “Done”-button appears.
4. Press the “Done”-button and remove the headset.

6. Description of dataset and selected features

This section will present some details about the gathered data, the process of selecting features to use as inputs to the network and preprocessing of the data.

6.1. The obtained dataset

The dataset consists of 720 data points obtained from 24 unique participants. Each participant provides 30 data points since that is the total number of cubes that are randomly displayed throughout the test.

The data has been collected at Chalmers University of Technology which resulted in a dataset with a majority of adults that have a higher level of education. The gender distribution of the collected data is 4% female, 96% male and 0% other. The variations in age ranged from the youngest participant being 22 and the oldest 63 years old with an average age of 34.

6.2. Selection of features

The features, shown in Table 2, that were used as input to the network are: the in-game timestamp, the gaze vector (x,y,z) for the left eye, the gaze vector (x,y,z) for the right eye, as well as the pupil diameters for each eye. These were chosen as a starting point of the analysis with the goal of investigating how much can be determined from just the eyes and then more features could be added if needed. The use of ET data alone would also make it easier to implement the functionality in a real world application. The number of the box (1-5) that was destroyed was used as the target label, since it is directly related to the main movement direction. The ID, age, and gender was used to manage the dataset as well as to provide some general information, these were however not used to train the network.

Table 2: Description of data used in classification.

Type	Feature
Input	Timestamp
Input	LeftEyeDirection.X
Input	LeftEyeDirection.Y
Input	LeftEyeDirection.Z
Input	RightEyeDirection.X
Input	RightEyeDirection.Y
Input	RightEyeDirection.Z
Input	LeftPupilDiameter
Input	RightPupilDiameter
Label	BoxClicked

6.3. Preprocessing of the data

The data from the tests was loaded into the computer memory from previous storage in files on the harddrive. From visual inspection of the histogram in Fig. 2, that shows the length of all data points, i.e. how many data samples that was collected from the time one box was destroyed until the next box had been destroyed. It can be seen that the dataset contains a few outliers. The shape of the data in the histogram can be approximated using a Beta distribution indicated by the black solid line in the figure. A threshold, dotted black line, was set to the mean plus three standard deviations (σ) of this distribution, with the values from Table 3, such that a maximum of Mean + $3*\sigma = 263+3*119 = 621$ data samples was allowed for the data point to be used in the classification. This is a reasonable limit since it corresponds to giving the test participant about 5 seconds to complete the task of finding the box and destroying it while at the same time keeping as many data points as possible from the dataset. The consequence of the filtering is that the dataset is reduced from 720 data points (N) to 702 data points (N) and that the maximum length (Max) of a data point decreases from 2417 to 602 as shown in Table 3. The minimum length (Min), however, stays the same. The histogram of the filtered data, which has a more reasonable distribution, can be seen in Fig. 3.

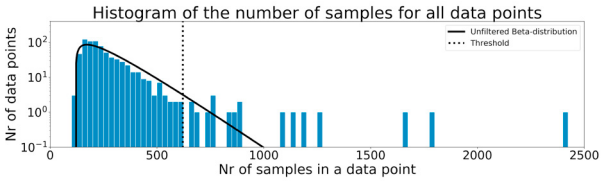

Fig. 2: Histogram of the number of samples for all data points.

Table 3: Distribution information for unfiltered and filtered data.

Type	Mean	σ	Min	Max	N
Unfiltered	263	119	116	2417	720
Filtered	238	79	116	602	702

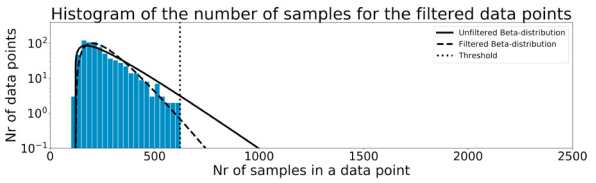

Fig. 3: Histogram of the number of samples for the filtered data points.

The next step was to filter out and discard all samples, within each data point, that contained NaN values. These occur when the ET fails to read the eye properly, the most common is that they are obtained as a result of the participant blinking.

The data points that were of shorter length than the decided threshold (621) were padded with zeros (ZP) at the end to guarantee that the structure of a data point that is fed to the network is a [621 x 9] matrix. Each row in the matrix corresponds to a data sample with the 9 features mentioned earlier. It was also tested to linearly upsample (US) the data points to achieve the desired length. A comparison of using these two methods for classification is presented in Table 4. US is, however, not an alternative for **Stage Three** and since ZP performed better, it was chosen as the preferred method.

After ZP, the data was normalized featurewise using the max-norm. The normalization makes sure that different value of magnitude between features does not bias the network to emphasize the importance of one feature over another.

Each data point is coupled with a label, to make the classification possible, that describes which cube that the test participant clicked on. The labels were one hot encoded such that it is possible to use them together with the loss function called `categorical crossentropy`.

Once the data was filtered, ZP:ed and normalized it was randomly split into three categories, training/validation/test. The proportions of the splits are: 45% of the data for training, 5% for validation, and the remaining 50% was used for testing and evaluation of the network.

7. Neural network design and classification results

A description of the development of the neural network architecture along with the classification results from the network will be provided in this section.

7.1. Neural network architecture

The neural network architecture that has been used to perform the classifications in this project is visualized in Fig. 4. The layout of this network takes the matrix X as the input and uses the Conv1D-layers, blue rectangles in the figure, as the base for extracting information from the data (the upper grey box in the figure) and the design itself is inspired by the inception modules from the network called `Inception-v3` [27]. The Conv1D-layers are used to analyze the time-dependency between a few nearby data samples across all the features. The thought behind using the structure of the inception modules is to calculate convolutional features with several filter sizes in parallel while at the same time reducing the number of parameters and the complexity of the network compared to using a fully connected (FC) neural network.

The network also contains pooling layers (red rounded rectangles) that are used to reduce the spatial size of the data fed to it, only keeping the most significant parts. This further helps reducing the number of parameters in the network. The purple parallelograms are responsible for managing the dimensions of the network connections through concatenation and flattening. The last parts of the network are the dense layers (yellow squares, corresponding to FC layers) and dropout layers (green diamonds, *training=True* means that it is used also when making predictions), followed by a final dense layer with as many neurons as there are output classes (5) that gives the output, $\hat{Y}$. The lower grey area in Fig. 4 is the part of the network that enables the UE.

All layers of the network uses the ReLU [23] function as the activation apart from the final dense layer which uses a softmax to enable multi-class classification.

The neural network has been trained using the `adam` optimizer [16] applying its default settings and `categorical crossentropy` as the loss function. The training was done until the validation loss stopped decreasing, terminating using early stopping.

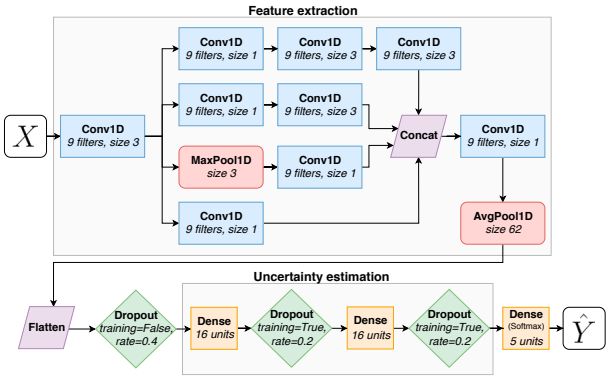

Fig. 4: A flowchart that describes the used network architecture.

7.2. Classification results

This section will provide the classification results, on the test set, from the trained network and a comparison that shows the impact of using UE [9].

The classification results without UE can be seen in Fig. 5. The graph shows the largest contributor from the softmax output for each sample that was classified. The samples are sorted in increasing order, left to right, based on this value. A green bar represents a correctly classified sample whereas a red bar indicates that the sample was incorrectly classified.

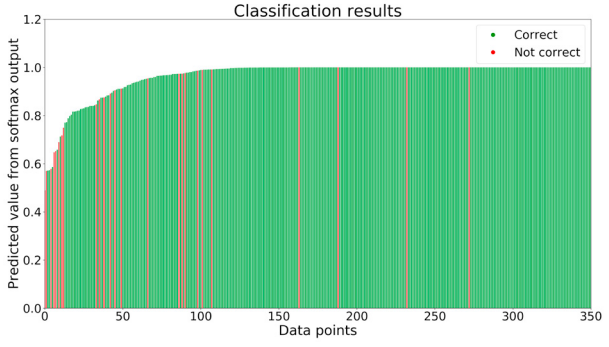

Fig. 5: A graph of the classification results without UE.

The difference when making predictions for UE is that many predictions are done on the same data such that it is possible to obtain a mean value and a standard deviation of the prediction. The pseudo code for this is shown in Algorithm 1.

The results obtained from the network using UE, with $nrOfPredictions = 1000$ as suggested by [9] to ensure good plots, can be seen in Fig. 6. The graph shows the largest contributor from the softmax output for each sample that was classified. The samples are sorted in increasing order, left to right, based on this value. The black interval displays two standard deviations of the prediction around

Algorithm 1 Pseudo code for predicting with UE.

Input: X, nrOfPredictions
Output: $\hat{Y}$, $\hat{Y}_{STD}$
1: predictions = []
2: **for** $i = 0$ to nrOfPredictions **do**
3: predictions[i] = model.predict(X)
4: **end for**
5: $\hat{Y}$, $\hat{Y}_{STD}$ = mean(predictions), std(predictions)
6: **return** $\hat{Y}$, $\hat{Y}_{STD}$

its mean value. A green bar represents a correctly classified sample whereas a red bar indicates that the sample was incorrectly classified.

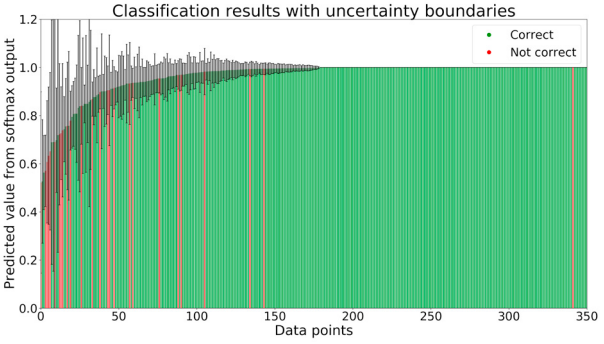

Fig. 6: A graph of the classification results with UE.

Once the mean and standard deviation has been obtained from the network these can be used to determine if the network is confident enough, high mean and low standard deviation, to make an accurate prediction. This was implemented as shown in Algorithm 2 where a prediction is accepted if the mean minus two standard deviations is larger than a chosen lower limit.

Algorithm 2 Pseudo code that accepts or discards a prediction.

Input: $\hat{Y}$, $\hat{Y}_{STD}$, lowerLimit
Output: $\hat{Y}$
1: **if** $\hat{Y} - 2 * \hat{Y}_{STD} >$ lowerLimit **then**
2: Accept $\hat{Y}$ as the prediction for this sample.
3: **else**
4: Discard $\hat{Y}$, the network is not confident enough.
5: **end if**

The classification results for different lower limits can be seen in Table 4. It is clear that the classification accuracy can be increased with this approach, however, at the cost of the network not being able to classify all samples.

8. Discussion

The classification results from Fig. 5 and Fig. 6 shows that the overall accuracy increases slightly when UE is used. This is due to the fact that the mean of the predictions gives a more stable result. However, more importantly, using UE enables reasoning regarding the network's uncertainty for the different predictions. The network does have some rather confident predictions that are wrong and these are likely a result of that it has learnt to classify certain situations the wrong way or that the data may be very similar for two directions that are close to each other.

Table 4: Comparison of classification results for different levels of filtering using UE for both US and ZP.

Lower Limit	US - accuracy	US - % samples classified	ZP - accuracy	ZP - % samples classified
0	88.37%	100.00%	93.28%	100.00%
0.10	89.03%	98.97%	93.52%	99.74%
0.20	89.36%	97.16%	93.73%	98.97%
0.30	91.71%	93.54%	93.70%	98.45%
0.40	93.68%	89.92%	94.44%	97.67%
0.50	94.67%	87.34%	95.39%	95.35%
0.60	94.82%	84.75%	95.59%	93.80%
0.70	95.82%	80.36%	96.31%	90.96%
0.80	96.22%	75.19%	96.76%	87.60%
0.90	96.76%	63.82%	98.33%	77.26%

The comparison between using US and ZP for different levels of uncertainty thresholding is shown in Table 4. It is clear that ZP gives a higher overall performance and is therefore the preferred method. The table also shows that there is a trade-off between increasing the accuracy through uncertainty thresholding and the number of samples that will be classified. The level of thresholding is, therefore, dependent on the application of the network. Consider the following two scenarios:

Scenario 1: A network that will be used as one of many inputs to an overarching control system should only provide its superior with high certainty information to avoid unnecessary interference. However, it does not have to classify all samples since their are other inputs in-place to cover for these cases, i.e. classification accuracy should be prioritized.

Scenario 2: A network that will be used as the only input to an overarching control system has to provide information at all times to ensure that the system is always running. On the other hand, it can afford to make some mistakes, i.e. have a lower accuracy, if the severity of a mistake is low.

9. Conclusions

Combining the areas of virtual reality, eye-tracking and machine learning can be one way to increase the intelligence of collaborative robots. This can be broken down into the three stages, **Stage One:** *Movement Direction Classification*, **Stage Two:** *Movement Phase Classification*, and **Stage Three:** *Movement Intention Prediction*, described in the introduction. This paper gives a solution to the first stage and shows that it is possible to collect

eye gaze data and use that to classify a person's movement direction. The results clearly shows that it is possible to combine VR and ET into a platform for testing and analysis of human behaviour, which can be beneficial in multiple areas of research. It is also shown that the implementation of UE improves the network and gives a way to improve the classification accuracy, at the cost of the percentage of samples classified, to obtain a more confident network. The next step is naturally to develop the second and third stage defined in this paper.

References

[1] Abidi, M., Al-Ahmari, A., El-Tamimi, A., Darwish, S., Ahmad, A., 2016. Development and evaluation of the virtual prototype of the first saudi arabian-designed car. Computers 5, 26.

[2] Al-Ahmari, A.M., Abidi, M.H., Ahmad, A., Darmoul, S., 2016. Development of a virtual manufacturing assembly simulation system. Advances in Mechanical Engineering 8, 1687814016639824.

[3] Aschenbrenner, D., Maltry, N., Kimmel, J., Albert, M., Scharnagl, J., Schilling, K., 2016. Artab-using virtual and augmented reality methods for an improved situation awareness for telemaintenance. IFAC-PapersOnLine 49, 204–209.

[4] Awais, M., Henrich, D., 2010. Human-robot collaboration by intention recognition using probabilistic state machines, in: 19th International Workshop on Robotics in Alpe-Adria-Danube Region (RAAD 2010), IEEE. pp. 75–80.

[5] Bi, L., Guan, C., et al., 2019. A review on emg-based motor intention prediction of continuous human upper limb motion for human-robot collaboration. Biomedical Signal Processing and Control 51, 113–127.

[6] Blender Documentation Team, . Blender 2.82 Reference Manual. Blender Foundation. License: CC-BY-SA v4.0. Accessed on: Feb. 13, 2020. [Online]. Available: https://docs.blender.org/manual/en/dev/.

[7] Dahl, M., Albo, A., Eriksson, J., Pettersson, J., Falkman, P., 2017. Virtual reality commissioning in production systems preparation, in: 22nd IEEE International Conference on Emerging Technologies And Factory Automation, September 12-15, 2017, Limassol, Cyprus, IEEE. pp. 1–7.

[8] El Makrini, I., Merckaert, K., Lefeber, D., Vanderborght, B., 2017. Design of a collaborative architecture for human-robot assembly tasks, in: 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE. pp. 1624–1629.

[9] Gal, Y., Ghahramani, Z., 2016. Dropout as a bayesian approximation: Representing model uncertainty in deep learning, in: international conference on machine learning, pp. 1050–1059.

[10] Gould, T.D., Bastain, T.M., Israel, M.E., Hommer, D.W., Castellanos, F.X., 2001. Altered performance on an ocular fixation task in attention-deficit/hyperactivity disorder. Biological psychiatry 50, 633–635.

[11] Hinton, G.E., Srivastava, N., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.R., 2012. Improving neural networks by preventing co-adaptation of feature detectors. arXiv preprint arXiv:1207.0580 .

[12] HTC, . VIVE VR System. HTC. Accessed on: Feb. 13, 2020. [Online]. Available: https://www.vive.com/us/product/vive-virtual-reality-system/.

[13] Huang, C.M., Mutlu, B., 2016. Anticipatory robot control for efficient human-robot collaboration, in: 2016 11th ACM/IEEE international conference on human-robot interaction (HRI), IEEE. pp. 83–90.

[14] Jungwirth, F., Murauer, M., Haslgrübler, M., Ferscha, A., 2018. Eyes are different than hands: An analysis of gaze as input modality for industrial man-machine interactions, in: Proceedings of the 11th PErvasive Technologies Related to Assistive Environments Conference, ACM. pp. 303–310.

[15] Karatekin, C., 2007. Eye tracking studies of normative and atypical development. Developmental review 27, 283–348.

[16] Kingma, D.P., Ba, J., 2014. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 .

[17] Krüger, J., Lien, T.K., Verl, A., 2009. Cooperation of human and machines in assembly lines. CIRP annals 58, 628–646.

[18] Lai, M.L., Tsai, M.J., Yang, F.Y., Hsu, C.Y., Liu, T.C., Lee, S.W.Y., Lee, M.H., Chiou, G.L., Liang, J.C., Tsai, C.C., 2013. A review of using eye-tracking technology in exploring learning from 2000 to 2012. Educational Research Review 10, 90–115.

[19] LeCun, Y., Bottou, L., Bengio, Y., Haffner, P., et al., 1998. Gradient-based learning applied to document recognition. Proceedings of the IEEE 86, 2278–2324.

[20] McAfee, A., Brynjolfsson, E., Davenport, T.H., Patil, D., Barton, D., 2012. Big data: the management revolution. Harvard business review 90, 60–68.

[21] Morariu, O., Morariu, C., Borangiu, T., Răileanu, S., 2018. Manufacturing systems at scale with big data streaming and online machine learning, in: Service Orientation in Holonic and Multi-Agent Manufacturing. Springer, pp. 253–264.

[22] Nagorny, K., Lima-Monteiro, P., Barata, J., Colombo, A.W., 2017. Big data analysis in smart manufacturing: a review. International Journal of Communications, Network and System Sciences 10, 31–58.

[23] Nair, V., Hinton, G.E., 2010. Rectified linear units improve restricted boltzmann machines, in: Proceedings of the 27th international conference on machine learning (ICML-10), pp. 807–814.

[24] Pettersson, J., Albo, A., Eriksson, J., Larsson, P., Falkman, K., Falkman, P., 2018. Cognitive ability evaluation using virtual reality and eye tracking, in: 2018 IEEE International Conference on Computational Intelligence and Virtual Environments for Measurement Systems and Applications (CIVEMSA), IEEE. pp. 1–6.

[25] Rizzo, A.A., Schultheis, M., Kerns, K.A., Mateer, C., 2004. Analysis of assets for virtual reality applications in neuropsychology. Neuropsychological Rehabilitation 14, 207–239.

[26] Samek, W., Wiegand, T., Müller, K.R., 2017. Explainable artificial intelligence: Understanding, visualizing and interpreting deep learning models. arXiv preprint arXiv:1708.08296 .

[27] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z., 2016. Rethinking the inception architecture for computer vision, in: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 2818–2826.

[28] Tang, G., Webb, P., Thrower, J., 2019. The development and evaluation of robot light skin: A novel robot signalling system to improve communication in industrial human–robot collaboration. Robotics and Computer-Integrated Manufacturing 56, 85–94.

[29] Tobii AB, . Tobii Pro VR Integration – based on HTC Vive Development Kit Description. v.1.7 - en-us ed. Tobii AB. Accessed on: Feb. 13, 2020. [Online]. Available: https://www.tobiipro.com/siteassets/tobii-pro/product-descriptions/tobii-pro-vr-integration-product-description.pdf/?v=1.7.

[30] Unity Technologies, . Unity User Manual (2018.1). 2018.1-002n ed. Unity Technologies. Accessed on: Feb. 13, 2020. [Online]. Available: https://docs.unity3d.com/2018.1/Documentation/Manual/index.html.

[31] Wu, L., Guo, L., Fang, H., Mou, L., 2018. Bullet graph versus gauges graph: Evaluation human information processing of industrial visualization based on eye-tracking methods, in: International Conference on Applied Human Factors and Ergonomics, Springer. pp. 752–762.