Human-Robot Collaboration on the Assembly Line

Amal Jose Vallavanthara, Ramapriya Janardhan, Vinoth Kumar Elangovan, Mike Funk, Group 3.1

Technische Universität Berlin

***Abstract*—When humans and robots work together in the same place, the robot needs to change its speed and agility of operations based on the state of the human. To achieve this, it needs to know what the human is doing at all times. We do this by using a pose-recognition approach based on machine learning, to detect human actions. We split our work in two parts. First, we use a simulation environment to test if our approach works in general. Second, since the system is created to work in real time as well, we develop our action recognition algorithm also for real camera data. For the simulated approach, we create our own datasets for training and testing a machine learning algorithm. For the real camera data we use the already existing NTU-Dataset. In the end, as a proof of concept we will show, that it is even possible to determine a human state of mind from the recognized actions.**

I. Introduction

According to the International Labor Organization (ILO), more than 337 million accidents happen on the job each year, resulting in more than 2.3 million deaths annually. A major proportion of these accidents occur in factories. With introduction of more robots in factories, there is an immense need for increase in safety measures, since humans and robots work cooperatively on the same task. This can be dangerous as humans tend to get distracted, tired or even lazy, which can cause major accidents. The productivity of humans varies over a day, so the robot needs to change its speed and agility of operations based on the state of the human.

To achieve this adaptive behavior of the robot, we need to know what the human is doing at all times. This is done by a pose-recognition approach based on machine learning, to detect human actions. We choose human actions to be recognized according to the before mentioned scenario of a collaborated factory workspace. The human could either try to communicate actively with the robot or just be moving in a natural manner according to the task. As active communication we understand actions like a *'Warning'*-Movement, where human actively tries to get in contact with the robot when there is e.g. an emergency or dangerous situation. The other more passive movements to be recognized are task specific (in our case: Human-Robot Collaboration on the assembly line), and can be movements like *'Grasping an object'* or *'Walking Around'*. The robot has to be able to recognize these actions to take an adaptive decision what to do next, e.g. if human needs help.

We split our work in two parts. First, we use a simulation environment to test if our approach works in general. Second, since the system is created to work in real life as well, we develop our action recognition algorithm also for real camera data.

For the simulated approach, we create our own datasets for training and testing a machine learning algorithm with 6 different actions to be recognized. For the real camera data we use an already existing dataset (NTU Dataset [1]) with similar actions to determine if our algorithm works with this data as well. For both approaches, we can use the same code architecture, since only the input data and the trained machine learning model varies.

In the end, as a proof of concept we will show, that it is even possible to determine a human state of mind {*lazy, tired, active*} from the recognized actions. This is the next important step in Human-Robot Collaboration: not only recognizing human actions, but the state the human is in to predict the human behavior and guarantee a safe workplace.

II. Related Works

When Humans and Robots work together on the same task, the Robot has to estimate the Human intentions to be non-intrusive. For this, first it needs to recognize the Human in the environment, detect Human gestures and actions and take a decision from that input.

There are various methods to detect Humans in an image [2] [3]. The automotive industry is doing big research in pedestrian detection to make self-driving cars safer and improve driver assistance-systems. Another scenario is video surveillance, for example in public places like train stations, with crowds of people where occlusions happen frequently. But the goal in these use-cases is to detect Humans in general as a whole. Those pedestrian detection algorithms dont care about special poses so they are not quite accurate for our scenario.

A major challenge for algorithms is to deal with occlusions. Also in our case the Human can be occluded by the packages it is picking up, by the conveyor belt or other objects. Deformable Part Based Models would be a solution to tackle this problem [4]. It models all parts of the body separately and merges the result, so that partial occlusions dont have such a great impact on the overall result. In all research there is always the tradeoff between accuracy and speed [5]. To get higher accuracy, new Features and better learning algorithms are developed, often with the loss of speed. Since computational power is growing every year and neural networks increase the accuracy of algorithms dramatically, we decided to use a deep learning algorithm for our scenario.

A fast working algorithm is needed, so that the robot can detect human actions in real time and react according to it. There are several methods to achieve a higher speed while maintaining the needed accuracy, one of the most popular is exploiting the idea of cascades from Viola and Jones [6]. It

uses classifiers of different complexities to search the image faster for the desired object or person.

Human action recognition in 3D skeleton sequences has attracted a lot of research attention. Recently, long short-term memory (LSTM) networks [7] have shown promising performance in this task due to their strengths in modeling the dependencies and dynamics in sequential data. As not all skeletal joints are informative for action recognition and the irrelevant joints often bring noise which can degrade the performance. In [8], a new class of LSTM network, global context aware attention LSTM, for skeleton-based action recognition has been proposed, which is capable of selectively focusing on the informative joints in each frame by using a global context memory cell. In [9] various predictive data mining algorithms and machine learning models for Human activity detection are discussed and compared for the accuracy and performances.

The tracking is only one part. To ensure a safe and pleasant Human-Robot collaboration, the robot has to interpret the Human actions correctly. This is done by integrating a Theory of Mind approach to the Adaptive Decision Making Process of the Robot [10]. In contrast to other research, where Human is assumed to always take Robots help and is doing all actions to achieve the task in our approach the Human behaviour is assumed to be stochastic and uncertain in all situations. So the Robot has to decide, based on his camera input, in what state Human is (e.g. tired, lazy, idle, distracted, active).

In our implementation, we dealt with determining human actions simulated from MORSE as well as actions captured from Kinect Camera. A DNN-Classifier Deployed Model was used for training in Tensorflow [11]. Initially a model was trained and tested to determine human action using 15 human joint information from NTU dataset [12]. To in order to improve accuracy for MORSE human actions, a neural network was created and trained for 300 diverse samples of MORSE human actions. In a similar manner, Tensorflow models were created for Kinect data to determine human action as well a state of mind recognition.

III. Methodology

To simulate the described scenario we use the MORSE simulation environment [13]. In this simulation environment, made for academic robotics, it is easy to create indoor or outdoor 3D simulated scenes with different robots and obstacles. It uses standard robotic sensors (e.g. cameras), actuators (e.g. speed controller) and robots (e.g. wheeled vehicle). The original goal was to implement the action recognition in MORSE alone. This required huge amount of data. The NTU dataset had millions of frames with over 60 action classes. This was an inspiration to use the already available dataset to train the neural network rather than create a whole new datset from MORSE. The task was tougher than anticipated as the dataset of NTU did not match the parameters of the MORSE. This led us to splitting the project into different tasks:

A. Task 1 - Create a MORSE dataset from scratch and train the whole network.

Since we want to detect human actions, we tried to use a virtual Microsoft Kinect in connection with the OpenNI Framework [14] as our sensor for human joint recognition in MORSE. Unfortunately the OpenNI tracker needs an actual Kinect camera connected to the computer to work properly. This is the reason why we could not use it with only a simulated camera in MORSE.

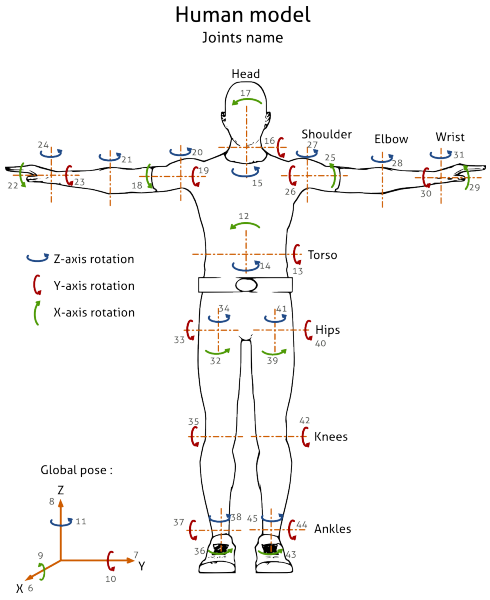
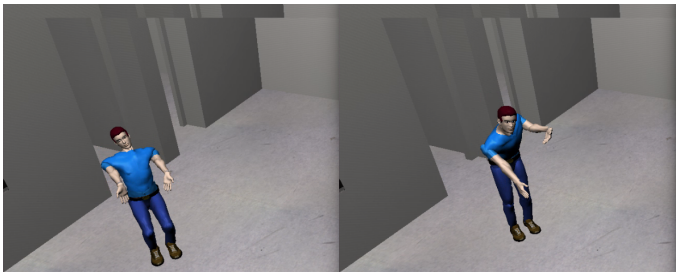


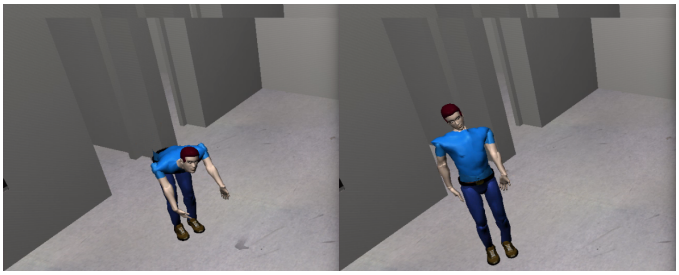
Fig. 1: Joints of HumanPosture Sensor in MORSE

So we decided for another sensor, the *HumanPosture* sensor [15]. It tracks all joint orientations directly from the human model in MORSE, which makes it easy to use. It provides 14 different joints with 34 joint orientations in total (see Fig. 1). To avoid collecting our own dataset for training and testing human action recognition, we tried to train on an already existing dataset - the *NTU Dataset* [1]. This dataset consists of 60 different actions, recorded multiple times, ending up with 4 millions frames of data. Since MORSE has only a limited number of implemented human actions to execute, we had to decide for actions which are also available in the NTU Dataset. We came up with 5 different actions: {*Sitting, Stand-Up / Sit-Down, Walking, Staying Idle, Grasping*} as in fig.2. Unfortunately the joint orientations of NTU and HumanPosture can not be compared with each other, since the NTU uses the Kinect camera as a reference frame, where the HumanPosture uses the joint orientations with respect to the body itself.

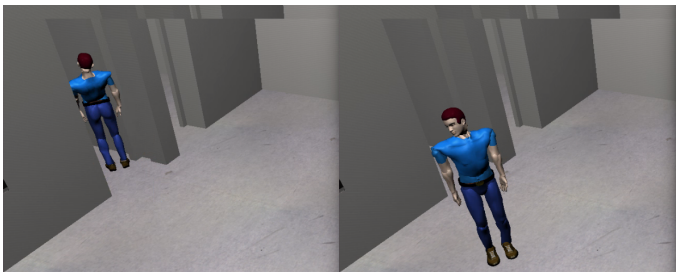
That is the reason why we come up with our own dataset for MORSE. We still use the HumanPosture sensor, but this time create our own dataset for training and testing. The actions we want to train for are already implemented for the human model in MORSE, but they are hard coded. So every action executed would be always exactly the same, which makes training and testing trivial. To come up with a sufficient dataset to train a Machine Learning algorithm, we add diversities to the human action movements in MORSE. This is done by adding some random factors to the hard coded kinematics of the human model, so that every execution of an action is slightly different.



(a) Sit (b) Warn



(c) Grasp (d) Idle



(e) Walk (f) Look Around

Fig. 2: Morse human model action simulations

We decide for 6 different actions to be recognized, all of them with a duration of 4 seconds: {*Sitting, Looking Around, Staying Idle, Walking, Grasping, Warning*} as shown in . We noticed that the frame rate varies for the MORSE sensor, but it stays around 30fps on average. From these 120 frames (4secs times 30fps), we extract exactly 30 periodic frames to train and test the system. We choose this amount of time, because the longest action takes almost 4 seconds to be completed. With the created diversities we were able to produce 300 diverse samples for each action to train the Neural Network.

B. Task 2 - Train the neural network with the NTU dataset and use the trained system to test the actions created by Kinect Camera through live feed.

For this, we actually spent time performing and recording the actions and creating a testing dataset with different human beings and a component of randomness. The frame rate is 30fps. This is exactly the same used in the NTU dataset. However, the joint information varies. The number of joints and the joint information available in both of the datasets varies. This is because the NTU dataset is extracted through Microsoft SDK library whereas we extract the node information through OpenNI tracker [14]. We tried to manually identify the joints and rule out information that was not common. Initially, the x, y, z positions and orientations of joints were considered for training. But the results were worse compared to just taking the x,y,z orientations alone. This could be attributed to the fact that it increased the dimensionality of the input vector without considerable increase in the information gain theory. We choose 6 actions from the NTU dataset from over 60 actions. We were able to get decent accuracy when tested with NTU dataset itself. However, on testing with the Kinect Camera dataset, the results showed a very low accuracy. This shows that the fault was not in the trained model we built or the ideology, but rather in the mismatch of the orientations. We believe that trying out the Microsoft SDK library to extract the joint information from the Kinect camera would yield us better results.

C. Task 3 - Recognition of Human State of Mind

This is a done as a proof of concept. It takes in the last 30 actions executed by the human and decides based on these 30 actions for the state of mind the factory worker is in. The person could be distracted, tired, working or incapable. Actions such as repeated sitting and standing with just a few lifts or grasps show, the person is likely to be tired. Or for instance, the person is just looking around while standing for a large portion of the time, the person is more likely to be distracted rather than working. We created a dataset of 4000 samples. The entire sample was created from intuition alone. In practice we believe that it is necessary to include more parameters like the number of hours that the person has been working for and his average efficiency. These parameters we believe, let us know if the person is tired or just distracted. A person who has been working for long hours is more likely to be tired rather than distracted. In an ideal case, we should be able to create a dataset with actual workers in real factory scenarios. Since this is not a possibility, we believe that creating a dataset based on intuition alone is the only possibility for supervised learning. In future, with adequate dataset and a mix of supervised and reinforcement learning, recognition of state of mind can be achieved with commendable accuracy.

D. ROS Environment

The general flow of human robot interaction is shown in Fig. 3. Camera captures human actions and movements and streams it to the cloud. In the cloud, machine learning algorithms process these captured images and detemine the human action and state of mind. This decision from the cloud will be sent to robot. Robot takes necessary action such as pickup or drop package based on decision from cloud.

Fig 9 shows the system architecture for human robot interaction. The ROS nodes and topics created to implement human robot interaction are described below.

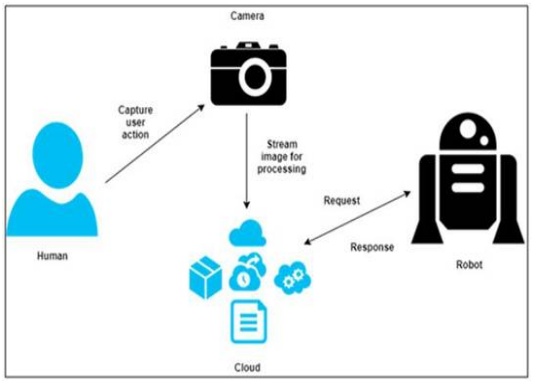


Fig. 3: System Design

Task Generator node controls the overall system behavior and execution. It assigns tasks to human control node and robot execution node through Task topic.

Human Control node generates the necessary token for the activation of an action. In the end, the camera is supposed to capture the actions of the human in real time and the robot decision making has to proceed based on that. But for simulation purposes, human action tokens, such as sit, stand-up, walk, warn, grasp, look-around etc. are manually requested or randomly generated and published to Human Action Execution topic.

Human Execution node subscribes to activation message from the Human Control node. That is, it subscribes and executes the action specified by Human Action Execution topic. The actions are executed by calling human action services in hrc-industry module.

Data Collection and Preprocessing node streams MORSE human joint information to Human Joint topic continuously when the human starts executing actions. Human actions can be traced with the joint information from the skeletal tracking model. This reduces the overhead of transferring large amount of data (like the full image of the human) over the network.

Human Action Recognition node node has two functions - creating data frames and deciding human action using machine learning service. A Convolutional Neural Network helps in identifying the command or movement of the human by means of its pretrained weights and layers. This possesses convolutional, linearization and maximization layers along with a fully connected top layer. This network is pre-trained with dataset consisting of actions that are pre-determined for each category of the use-case. In real time, it involves fewer computations to produce the necessary category of the action that is received as input. The Machine learning service/tensorflow is implemented in a cloud so as to reduce hardware complications that may arise due to huge computations if implemented in a robot. The node samples the data streamed to Human Joint topic for specified duration and create frames. The frame is then sent to a trained tensorflow model, which determines the human action related to the input frame. The decision by tensorflow is then published to Human Action Estimation topic.

Human State Recognition node is similar to Human Action Recognition node, but it samples the human actions streamed to Human Action Estimation topic for a specified duration and sends it to a trained tensorflow model to determine the human state related to the input frame. The decision by tensorflow is then published to Human State Estimation topic.

Robot Execution node node controls the actions executed by robot. It subscribes to Human Action Estimation topic and Human State Estimation topic. Based on human actions and human state of mind, the robot can decide to pick-up or drop the package or perform any assigned tasks.

IV. Results

A. Task 1

The neural network is trained with the dataset generated by MORSE. 85% of the dataset generated is utilized for training while 15% is utilized for testing the trained model. A total of 6 actions represented as 6 classes, with each category having a dataset of 300 samples. Each sample represents an action performed under the duration of 4 seconds, which is assumed to be the timeframe an action would require in realtime to get completed. The frame rate is fixed to be 30 frames/second so as to mimic the data rate of a Kinect camera.

The model for training is a Deep Neural Network classifier with an input layer, 5 hidden layers and an output layer with each layer being fully connected. The number of hidden layers is chosen after experimenting with different numbers for the optimal performance of the neural network. A total of 1020 feature vectors are introduced for the network to learn the pattern from the input data and classify them into the corresponding categories. Initially, 100 samples of each action was created from MORSE. After training on the neural network we were able to achieve 73% accuracy. In the 2nd phase, we increased the datset to 300 samples and increased the epochs. This led to us to create 99% accuracy for the same set of actions. This showed us the importance of big data and the computation steps to increase the accuracy. The loss in the final iteration also decreased tremendously. Care was taken to make sure we did not have any overfitting. The loss for every 10 steps is logged and finally visualized in tensorboard. The loss convergence happens approximately after 200 steps as in fig.4 for the initial training and after 700 steps as in fig.5 for the enriched dataset after which local minima is found to have been identified and the loss does not vary any more. This training was performed on a 4 core Intel(R) Core(TM) i7-4790 CPU @ 3.60GHz processor.

1) Testing in Realtime: Although testing with datasets gives appreciable results, it has to be tested in real time as well. We utilized the trained model in the ROS network. The action to be performed by the human is controlled by a manual input. A live stream of joint nodes is continuously transmitted to the preprocessing state, where frames with 4 seconds of data (which equals one executed action) are conjoined together as a single real time test sample. This sample is then processed by the trained model to give an output specifying the predicted class.

The real time testing was performed on a 4 core Intel(R) Core(TM) i3-2348 CPU @ 2.30GHz. We recognized that the

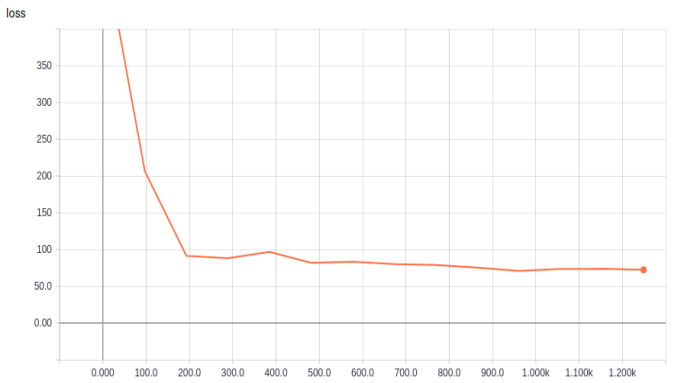


Fig. 4: Steps vs Loss Graph for Action recognition training from tensorboard with morse dataset with 100 samples per class.

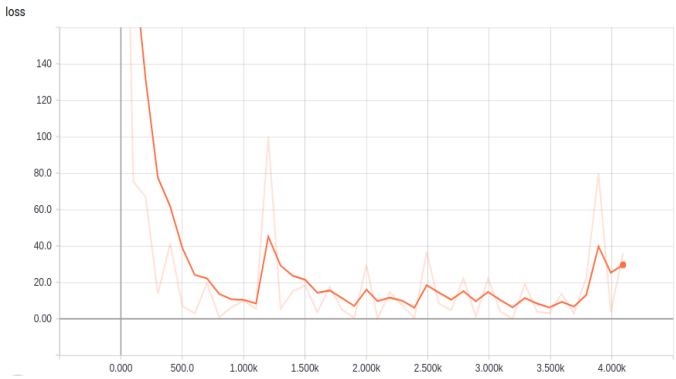
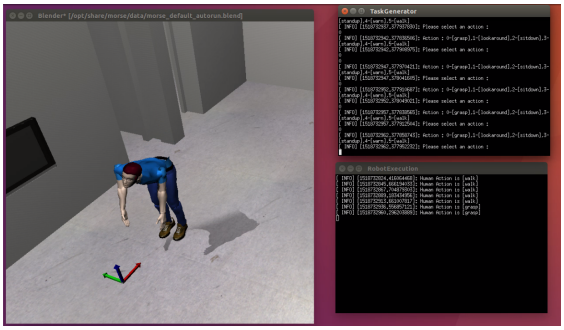
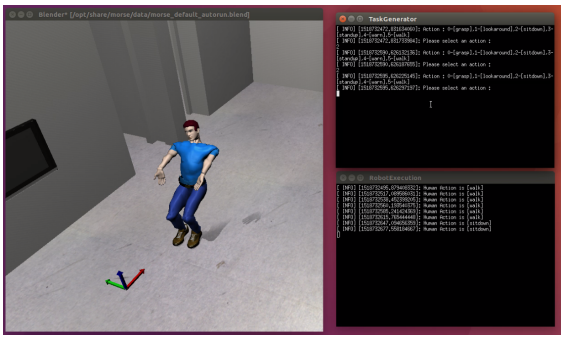


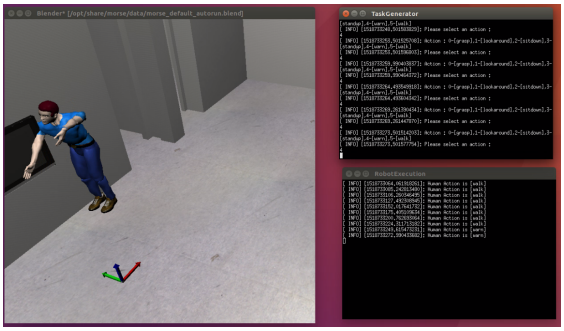
Fig. 5: Steps vs Loss Graph for Action recognition training from tensorboard with morse dataset with 300 samples per class.



(a) Grasp



(b) Sit



(c) Warn

Fig. 6: Action recognition with real time morse data

actions *walk, idle* and *look around* were being identified as the same action - *walk*. Investigation over the dataset revealed that the data produced by MORSE in real time does not have the proper node information for the joints in the leg. This makes it difficult for the network to classify between these three actions as the other two actions; idle and look around do not involve movements in the joints of the leg.

Actions *grasp* and *warn* are identified properly although with a delay. This is because the used processor lacks any GPU which affects the performance of the tensorflow. This results in a 20 seconds delay before the model can output the category. The *grasp* action begins with the idle position, proceeds with a grasp position and the resets back to the idle position, all within 4 seconds. Since the tensorflow takes 20 seconds to produce the output, the data frames of the grasp action are overwritten by the *idle* action and results in wrong output. To avoid this we build a 20 seconds buffer of the same action by continuously repeating it as input. The same problem and rectification happens for *warn* action also. This procedure of buffering the same action helps the model to predict the action correctly. A fast performing processor equipped with GPU will rectify this delay and help the model provide the predictions immediately. Fig 6 represents the actions that were successfully identified

B. Task 2

A Deep Neural Network classifier with an input layer, 5 hidden layers and an output layer with each layer being fully connected is used for this task too. With 6 actions being represented as 6 classes, the dataset of these actions are fed into the neural network for learning. Upon training for 6 actions an accuracy of 84% is achieved with the test dataset which is 15% of the total number of available samples.

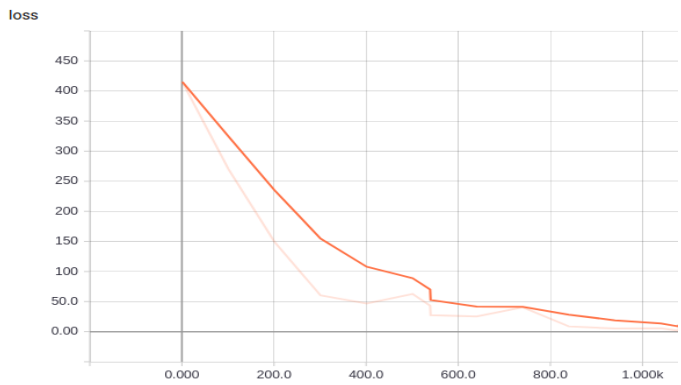


Fig. 7: Steps vs Loss Graph from tensorboard for Action recognition training with NTU dataset

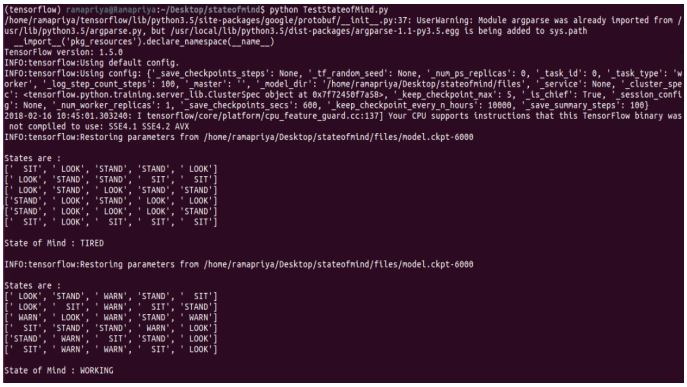


Fig. 8: State of Mind detection based on set of 30 continuous actions

The loss function is logged for every 10 steps for the loss value. The loss seems to converge after 1000 steps as shown in fig.7. One major drawback is, that the full coordinate information available along with the coordinate position and the orientation can not be utilized for training. This is, because it makes the dimensions of the input vector higher, but the number of samples available for each category are very less relatively. This results in producing an input vector with lower dimensions by clipping of the coordinates and using only the orientations information. However as described in III-B, the performance of the real time data could not be tested.

C. Task 3

The Deep Neural Network classifier is fed with data samples created manually from the actions executed by the human in MORSE. A set of of 30 actions represents one state of mind. 30 actions are chosen because at the rate of 30 fps and a timeframe of 4 seconds per action, 30 actions will have a timeframe of 2 minutes. So for every 2 minutes the human state of mind is monitored. To increase or decrease the timeframe for detection, it would be necessary to increase or decrease the input vector. Different combinations of 6 actions are used to produce the sample with 30 actions and distributed among the four different states of mind which are represented as classes. These four states are *distracted, tired, working* and *incapable*. We were able to test it by feeding a combination of 30 actions as in fig.8 randomly generated from within the available 6 categories of Task 1, but since this approach was based on intuition, it is not possible to evaluate the dataset.

V. Conclusion

We have shown the methodology for human action recognition for the actions generated via MORSE and the NTU dataset. We splitted our main goal into 3 parts as using NTU dataset for action recognition of the human model in MORSE was not possible. For prediction model with MORSE dataset, we overcame the irregularities for the MORSE human and generated datasets for the 6 use case actions *grasp*, *warn*, *sit*, *idle*, *walk* and *look around*. An accuracy of 99% was achieved with the MORSE dataset, however the prediction suffered in real time due to the processor speed. We rectified it by introducing buffer and repetitive actions. However this issue can be solved by utilizing a processor with GPU to have a higher computational speed. For the NTU dataset, 84% accuracy was achieved in recognizing the actions. Testing in real time with self recorded Kinect data was not possible due to a mismatch in the node information while using OpenNI library and Microsoft SDK library to generate the skeletal nodes. Finally we proposed a model for the recognition of state of mind as a proof of concept. For the future development, dataset generation of actions that are related to an industrial workplace is important. The neural network model then has to be trained and fine tuned with this dataset for realtime application. To be able to use the NTU dataset, it is crucial to match the NTU joint data with the live Kinect data. We assume it works with the Microsoft SDK, but this still has to be tested. Although the real life data can not be mapped directly to the MORSE data, the simulation environment offers a good solution to test the action recognition and theory of mind approach.

References

[1] "Ntu rgb+d action recognition dataset," http://rose1.ntu.edu.sg/Datasets/actionRecognition.asp, accessed: 2018-03-01.

[2] S. Zhang, Z. Wei, J. Nie, L. Huang, S. Wang, and Z. Li, "A review on human activity recognition using vision-based method," *Journal of healthcare engineering*, vol. 2017, 2017.

[3] R. Benenson, M. Omran, J. Hosang, and B. Schiele, "Ten years of pedestrian detection, what have we learned?" in *European Conference on Computer Vision*. Springer, 2014, pp. 613–627.

[4] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *IEEE transactions on pattern analysis and machine intelligence*, vol. 32, no. 9, pp. 1627–1645, 2010.

[5] A. Angelova, A. Krizhevsky, V. Vanhoucke, A. S. Ogale, and D. Ferguson, "Real-time pedestrian detection with deep network cascades." in *BMVC*, vol. 2, 2015, p. 4.

[6] P. Viola, M. J. Jones, and D. Snow, "Detecting pedestrians using patterns of motion and appearance," in *null*. IEEE, 2003, p. 734.

[7] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[8] J. Liu, G. Wang, L.-Y. Duan, K. Abdiyeva, and A. C. Kot, "Skeleton based human action recognition with global context-aware attention lstm networks," *IEEE Transactions on Image Processing*, 2017.

[9] J. B. Rana, R. Shetty, and T. Jha, "Application of machine learning techniques in human activity recognition," *arXiv preprint arXiv:1510.05577*, 2015.

[10] O. C. Görür, B. S. Rosman, G. Hoffman, and S. Albayrak, "Toward integrating theory of mind into adaptive decision-making of social robots to understand human intention," 2017.

[11] "Dnn classifier," https://www.tensorflow.org/versions/r1.2/api_docs/python/tf/contrib/learn/DNNClassifier, accessed: 2018-03-01.

[12] A. Shahroudy, J. Liu, T.-T. Ng, and G. Wang, "Ntu rgb+ d: A large scale dataset for 3d human activity analysis," *arXiv preprint arXiv:1604.02808*, 2016.

[13] "The morse simulator documentation," https://www.openrobots.org/morse/doc/latest/morse.html, accessed: 2018-03-01.

[14] "Openni," http://openni.ru/index.html, accessed: 2018-03-01.

[15] "Human posture," http://www.openrobots.org/morse/doc/1.4/user/sensors/human_posture.html#configuration-parameters-for-human-posture, accessed: 2018-03-01.

VI. Appendix

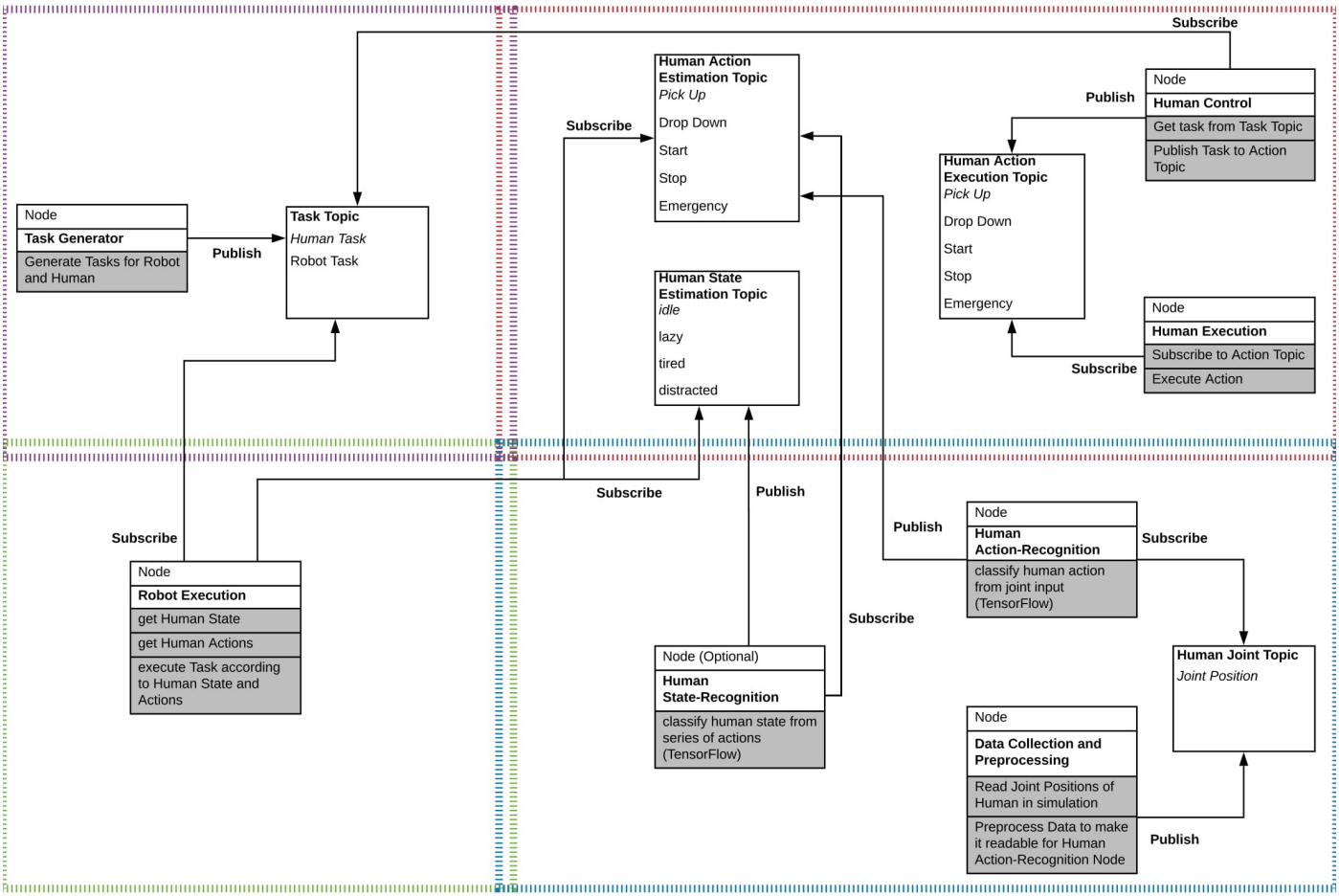


Fig. 9: Appendix: SystemArchitecture