Real Time Hand Gesture Recognition in Industry

William Dumoulin (https://orcid.org/0000-0002-0242-8612), Henallux Pierrard Industrial engineer, Belgium, **william@idt.be (mailto:william@idt.be)**
Nicolas Thiry (https://orcid.org/0000-0002-0689-4389), Henallux Pierrard Industrial engineer, Belgium, **nicolasthiry5@gmail.com (mailto:nicolasthiry5@gmail.com)**
Rim Slama (https://orcid.org/0000-0003-2723-9874), Fors, Henallux Pierrard Industrial engineer, Belgium, **rimslamarim@gmail.com (mailto:rimslamarim@gmail.com)**

DOI: https://doi.org/10.1145/3503961.3503962 (https://doi.org/10.1145/3503961.3503962)
VSIP 2021: 2021 3rd International Conference on Video, Signal and Image Processing (https://doi.org/10.1145/3503961), Wuhan, China, November 2021

With the 4th industrial revolution and the increased use of cobots in the industries comes many opportunities for new generation control panels. In this article, we proposed to develop a deep learning model to recognize in real time 10 different gestures that can be used to interact with a cobot. We proposed a new dataset containing gestures that can be used in an industrial context. The videos were taken from a computer webcam and then processed to remove the noise created by the background by isolating the movement of the gray scale images. We proposed to extract the spatio-temporal features by the combination of 3D convolution and LSTM layers. We also proposed a real time method to recognize our gestures, the frames are captured continuously and fed to the model to get a prediction every 2.4 seconds. Our experimental results show for 8 out of 10 gestures, a recognition rate of more than 90%. Furthermore, an interface was created to test our method in real time and to add new classes of gestures to be recognized by our model.

CCS Concepts: • **Computing methodologies** → **Activity recognition and understanding**;

KEYWORDS: Gesture recognition, Deep learning

ACM Reference Format:
William Dumoulin, Nicolas Thiry and Rim Slama. 2021. Real Time Hand Gesture Recognition in Industry. In *2021 3rd International Conference on Video, Signal and Image Processing (VSIP 2021), November 19-21, 2021, Wuhan, China.* ACM, New York, NY, USA, 11 Pages. https://doi.org/10.1145/3503961.3503962 **(https://doi.org/10.1145/3503961.3503962)**

1 INTRODUCTION

In recent years, the world of industry has witnessed a major change, referred to as "Industry 4.0". In fact, robots are now used in crowded environments and in shared areas with operators, which demands a focus on the development of communication techniques with robots like gesture recognition for human robot interaction. There are different levels of human robot interaction according to its degree: coexistence, synchronization, cooperation, and collaboration. Collaboration is the highest level of interaction among these levels where robot and operator share the same space and may work simultaneously on the same component. In this context, we need a specific algorithm able to recognize gestures needed to control a cobot without a control panel.

Existing human gesture datasets mostly contain gestures for sign language recognition or virtual reality context with a clear lack of industrial hand gesture recognition datasets and methods.

This paper proposes to bring a solution in this context. Indeed, we propose an interface that we used to acquire a new dataset with specific gestures that can be used in an industrial context. The collected dataset is then used within a convolutional approach to extract spatio-temporal features in order to classify gestures. Moreover, we propose a real time approach to recognize gestures using a friendly interface. The interface mainly shows a feed-back of the camera with the predictions from the model. There is also the possibility to create new videos and add new gestures in order to enrich the dataset with desired gestures.

The rest of the paper is organized as follows: Section 2 examines the state of the art methods for gesture recognition and used datasets. Section 3 introduces the proposed dataset that we created in an industrial context. Section 4 explains the proposed approach for gesture recognition. Section 5 presents the experimental results. Section 6 explains how to recognize gestures in real time. Finally, the last section concludes the paper with future research directions.

2 STATE-OF-THE-ART

Many different techniques were developed in the past years for hand gesture recognition in general. One of the first was an approach that separates the fingers from the palm [1]. The position of the hand on the static image was then deduced depending on the number of fingers found by the algorithm and their relative position.

A much more developed method is to apply a skeletonization method [2, 3]. The idea is to reduce an object to lines that are only one pixel wide. The images are therefore easier to analyse by the model. This method is applied to the videos before the training and the tests. An algorithm to create the skeleton of a shape need a Voronoi tessellation [4] but it's a heavy algorithm that slows down the entire process. A solution would be to use the Champfer distance transform [5] to reduce the computational cost.

N. Luthfil Hakim, T. K. Shih et al [6] created a model of gesture recognition for a television application. They used an RGB and a depth camera to get more valuable information. Their model is composed by multiple blocs of 3D convolution layers combined with dense and max pooling layers. The last part of their model is the Long Short-Term Memory layers [7] that find features that are time related which are crucial when dealing with videos.

Another basic method is the haar-like features [8] that are used for face or eyes detections. This method is called hand-crafted feature extraction method. If the color changes are not sharp enough, this method will show no results. Another method is to use the skin color to detect the hands [9].

The LSTM layer is an advanced type of RNN [10]. Enhanced methods of Recurrent Neural Network that could be used for gesture recognition were developed: Differential RNN [11], Hierarchical RNN [12], Bidirectional RNN [13].

Some articles [6, 14, 15] use the Kinect camera to get the depth information in addition to the 2D images.

Transformers [16, 17] are a type of layers that is designed to handle sequential data. Unlike LSTMs, they use a mechanism to give different weighs to the input data.

When working in challenging conditions, [18] can help to deal with it.

3 PROPOSED DATASET

3.1 Dataset Creation Protocol

In order to recognize gestures in an industrial context and for cobot controlling with gestures, we decided to create a new dataset. In order to collect this dataset, we created a program with a friendly interface that enables us to easily register videos with hand gestures (Figure 1).

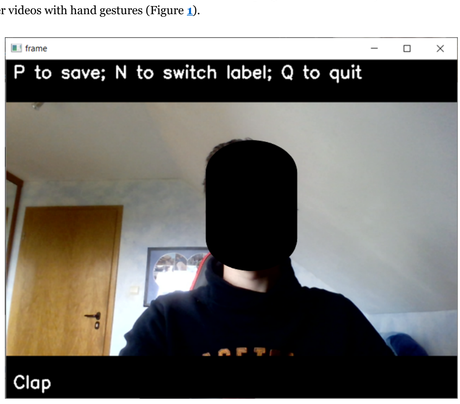

Figure 1: New video program

A dataset needs recordings of different people otherwise the model would only recognize the gesture of the person that did all of the data set videos. The smallest data set needs at least 12 different people and each one does each gesture twice. The result is a training set of almost 1 500 videos.

The data set is not split into a training and a test group. The data set is completely used for the training. Before the subjects recorded their movement, they were told to do the gesture the way that they wanted, with no other specification than the name of the movement and the time that they had to do it. The goal of this process was to create a dataset with meaningful gestures. However, when we reviewed the videos, it appeared that they were all very similar regarding the speed and position of the gesture in the image.

We decided to keep all these videos in the training set and to create ourselves the test set to have more challenging videos. The gestures are done slower or faster than the training set and in various places on the screen.

There are ten classes but there should be an eleventh: The non-movement class. This missing class is compensated by the prediction that gives a probability as an output for each gesture. If none of the gestures has more than 60%, it's not considered as a movement and is equivalent to the non-movement class.

3.2 Data Processing

The videos from the data set have 61 frames with a dimension of 640 pixels wide by 480 pixels height and 3 layers of depths for the red, green, and blue channels.

First step, conversion from RGB to gray image: Each pixel of the new gray image is calculated with the color channels of the source image with this formula:

$$Gray \Leftarrow 0.299 * Red + 0.587 * Green + 0.114 * Blue$$

The second step is to isolate the movement in the images. To do that, each image is compared with the next one. That make 60 frames of movement.

The pros of keeping only the movement are to remove the background. Of course, to make the training data, the background need to be still, otherwise it will be considered as part of the gesture.

The model doesn't get 60 frames for each video because it would make an over complicated model. The third step is to isolate a sample of the frames. The choice was made to take only 10 fps, that make a sample of 24 frames.

The fourth step is to resize and normalize the videos. The normalization takes all the pixels and map them from 0-255 to 0-1. After all the processing, the videos have 24 frames with a dimension of 120 pixels wide by 90 pixels height. All the pixels have a value between 0 and 1. See Figure 2 for comparison.

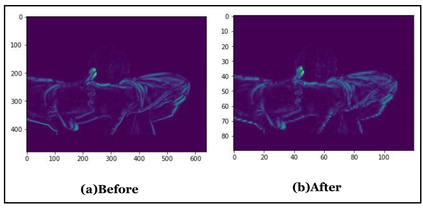

Figure 2: Normalization

The data set is available on the GitHub page [19]. To protect the identity of the persons in the data set, the videos are already processed like explained. The normalization is not done yet.

4 PROPOSED APPROACH

In order to recognize gestures on our collected dataset, we first proposed to test the model based on the work of N. L. Hakim, T. K. Shih, et al [6]. In this article, data are collected from an RGB and a depth camera. The images from the two cameras are processed separately and combined after the LSTM units [20]. The image processing is made with multiple 3D convolution layers.

The authors proposed to use a data set with thousands of videos and got a result of 91%. Our setup does not include a depth camera. The RGB images are therefore processed on their own. As expected, testing this approach on our data was not concluding. The problem was because of the small data set we are using and the large number of parameters proposed in this model. Thus, we simplified it to create another model more efficient in this context.

We trained 15 different models, each one had a small difference compared to the last one in order to understand how each parameter affects the model.

Our final model that we selected is composed of 3 blocs: The first two are an arrangement of Convolution 3d, dense, dropout, max pooling 3d and batch normalization. The last one has two ConvLSTM2D followed by a long-short term memory and a dense layer to have as many outputs as we have classes.

4.1 3D Convolution

In the convolution layers [21], [22], new channels are created with different kernels. A kernel is a squared matrix, whose size is generally 3x3 or 5x5. To analyse videos, we chose to use 3D convolution layers, with a size of 3x3x3 or 5x5x5.

The goal of this layers is to find the features of the images in the video.

4.2 Dense

The dense layer is also called fully connected layer. Each node of a layer is connected to all the nodes of the next layer. In our case a node is an entire image. Each channel of the last dimension is also an image. For example, the dimensions of the first dense layer are (24x90x120x256) which stands for 24 moments in time, 90 pixels height by 120 pixels wide and 256 images. The different images for one moment in time are created by the convolution layer.

The connection is defined by a weight with a value between minus one and one. In this architecture, the dense layer is placed after a convolution one. We can compare the dense layer to a filter.

The meaningful channels created in the 3D convolution layer have a weight with a high value, the useless ones have a weight close to zero.

4.3 Dropout

When this layer is placed after a dense layer, it removes a certain percentage of the weights by giving them new random values. This layer is added in order to reduce the risk of overfitting.

4.4 Max pooling 3D

This layer reduces the size of the data set by keeping the max value of a section of an image (or multiple images when it's 3D). A 3D Max pooling layer with a matrix (3x3x3): The dimensions go from (24,90,120,256) to (24,30,40,85) because the three last dimensions are divided by 3. This layer simplifies the data by keeping only the meaningful pixels.

4.5 Batch normalization

The layer transforms inputs so that they are standardized. This means that they have a mean value of zero and a standard deviation of one.

4.6 LSTM

Long Short-Term Memory [7] layers are a type of recurrent neural network capable of finding time dependent features.

This layer (and the ConvLSTM) is very important in this application because there are needed to differentiate "Swipe to the left" and "Swipe to the right" for example.

4.7 ConvLSTM 2D

Like the classic LSTM, the convLSTM is capable of learning order dependence in sequence but this layer is specialized for the videos analyses because it's the combination of LSTM and convolution layer.

In this architecture, two ConvLSTM 2D are stacked.

The first ConvLSTM layer provides an output for each input while the second layer gives only one output for a sequence of input.

4.8 Reshape

The layer doesn't learn anything in the training process. It only changes the dimension of the data to pass them to the LSTM layer which needs specific input.

The diagram in Figure 3 summarizes the architecture of the system, from the capture of the video to the prediction by the model.

Figure 3

Figure 3: Complete process

4.9 Model Training

The videos are stored as they were recorded. They need to be processed when they are loaded to train the model. The data set of 1 500 videos is too heavy to be loaded in a standard computer's RAM, so the videos have to be load by package.

The labels of the videos are saved in a list which is randomly mixed. The videos can then be loaded in the order of the list. It's important that the model is trained on different gestures every time.

To train the model faster, the data processing and training are on two different threads. The process thread loads two packs of 40 videos each and waits for the training part to be done with the previous pack. Then the loaded pack is sent to the training section. The videos are loaded by small packs of 40, when the model finishes to train on them they are unloaded to avoid overloading the computer's memory.

For each pack of videos, there are multiple batches to train on: we chose a batch size of 20 videos.

5 EXPERIMENTS RESULTS

We can see on Figure 4 that the first two gestures are often confused with one another. This is probably due to the proximity of theses gesture. We can see that closing a hand and opening a hand is done at the same places on the videos. Furthermore, the difference between closing and opening a hand is much more subtle than the difference between swipe left and swipe right. (Table 1).

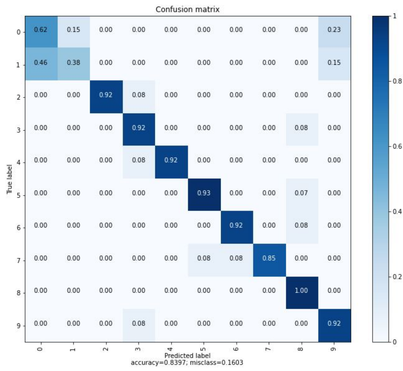

Figure 4: Confusion matrix

Table 1: List of gesture

No	Gesture Name
0	Closing one hand
1	Opening one hand
2	Clap with both hand
3	The right hand swipes to the left
4	The left hand swipes to the right
5	Crossed fingers
6	The hands start closed in the middle of the image and then go in opposite direction while they open
7	The right arm starts horizontally and rotates around the elbow, it ends when the hand is pointing towards the sky
8	The left arm starts horizontally and rotates around the elbow, it ends when the hand is pointing towards the sky
9	Waving

For all the other gestures, the model is very efficient. On the live test, we often got a precision greater than 90%.

The accuracy of our model is 84%. However, other works that are using different techniques have a higher accuracy. [9] uses many techniques in the preprocessing and achieves an accuracy of 98%. [6] uses also advanced pre-processing techniques and achieves 95.1 to 97.6% of accuracy with different models.

In our work, the pre-processing is simple and the accuracy is 84%. That is why we think that it might be interesting to continue the research on this pre-processing method.

6 REAL TIME RECOGNITION APPLICATION

6.1 Real Time Recognition Method

The application works with threads, there is one for the prediction and another one for the interface. The videos are recorded at 10 frames per seconds, which makes a pack of 24 frames which is sent to the prediction. 24 frames is the number of images that were given per video to train the model. The frames are processed directly when they are taken, the list of processed images is sent to the prediction thread.

A CPU Intel i7-1065G7 takes 1.5s while a GPU Nvidia gtx1050 takes 0.4s to get the prediction done.

More predictions could be done in parallel. For example, send a video for prediction every 12 frames. The 12 other frames would be kept from the end of the previous video. If the prediction thread gets the packs faster than the time to predict, it will create some delay between the gesture and the result.

6.2 Interface

The interface was coded using the python libraries tkinter [23] and OpenCV [24].

The main window, that can be seen at the Figure 5, shows the live feed from the camera. It also shows the prediction that have the highest score at the bottom of the screen. Furthermore, the window shows the 3 first predictions for the same video with the percentage of recognition. If the first prediction given by the model is lower than 60%, the predictions are not displayed and a message is written on the screen explaining that the model did not recognize the gesture.

Figure 5: Main Window

The last information that is shown is the number of frames that are already captured in the video. This allows the person in front of the camera to perform the full gesture on one video. As soon as the video is fully captured, another video is being captured.

The main window also shows two buttons. The first one freezes the prediction. In other words, the videos are no longer captured and the prediction stays on the window. The second button opens a second window. While the second window is active, the first is completely freezed to avoid wasting the resources of the computer.

The second window, that can be seen at the Figure 6, displays an example of each gesture that the model is able to recognize with its name. It also allows the user to add a new gesture. The first step is to create a new label or title for the gesture and add it to the list through the text field on the window. If the label already exists on the list, a message is displayed on the screen.

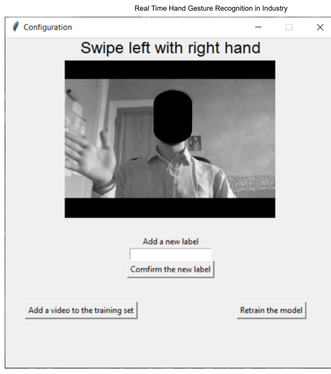

Figure 6: Parameter's window

On this window, there are two buttons. The first one opens program to record a video and add it to the training set. This program is explained at the third point: Proposed Dataset. The second button retrains the model. This operation takes a lot of time. Therefore, a safety needs to be installed to avoid miss-clicks.

7 CONCLUSIONS AND FUTURE WORKS

In this paper, we proposed a dataset collected within a new friendly used interface in order to acquire hand gestures. The collected data aims to be used for hand gesture recognition for controlling cobots in industrial context. We also proposed a well-adapted architecture of a deep learning approach that is able to recognize in real time such gestures with efficiency.

The result we obtained allows us to obtain an overall accuracy of 84% and 90% for 8 gestures out of 10. In future works, we intend to collect more data in the industrial context using our developed interface. We also have to focus on resolving the problem of gesture confusion by improving the model performances.

REFERENCES

[1] Z.-h. Chen, J.-T. Kim, J. Liang, J. Zhang, and Y.-B. Yuan, “Real-time hand gesture recognition using finger segmentation,” The Scientific World Journal, vol. 2014, 2014. Navigate to

[2] R. L. Ogniewicz and M. Ilg, “Voronoi skeletons: theory and applications.” In CVPR, vol. 92, 1992, pp. 63–69. Navigate to

[3] B. Ionescu, D. Coquin, P. Lambert, and V. Buzuloiu, “Dynamic hand gesture recognition using the skeleton of the hand,” EURASIP Journal on Advances in Signal Processing, vol. 2005, no. 13, pp. 1–9, 2005. Navigate to

[4] A. Beristain Iraola, Skeleton based visual pattern recognition. Applications to tabletop interaction. Servicio Editorial de la Universidad del Pais Vasco/Euskal Herriko..., 2009. Navigate to

[5] M. A. Butt and P. Maragos, “Optimum design of chamfer distance transforms,” IEEE Transactions on Image Processing, vol. 7, no. 10, pp. 1477–1484, 1998. Navigate to

[6] N. L. Hakim, T. K. Shih, S. P. Kasthuri Arachchi, W. Aditya, Y.-C. Chen, and C.-Y. Lin, “Dynamic hand gesture recognition using 3dcnn and lstm with fsm context-aware model, “Sensors, vol. 19, no. 24, p.5429, 2019. Navigate to

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

[8] Q. Chen, N. D. Georganas, and E. M. Petriu, “Real-time vision-based hand gesture recognition using haar-like features,” in 2007 IEEE instrumentation & measurement technology conference IMTC 2007.IEEE, 2007, pp. 1–6. Navigate to

[9] Trong-Nguyen Nguyen, Huu-Hung Huynh, and Jean Meunier , "Static Hand Gesture Recognition Using Artificial Neural Network," Journal of Image and Graphics, Vol. 1, No. 1, pp. 34-38, March 2013. doi: 10.12720/joig.1.1.34-38 Navigate to

[10] C. L. Giles, G. M. Kuhn, and R. J. Williams, “Dynamic recurrent neural networks: Theory and applications,” IEEE Transactions on Neural Networks, vol. 5, no. 2, pp. 153–156, 1994. Navigate to

[11] V. Veeriah, N. Zhuang, and G.-J. Qi, “Differential recurrent neural net-works for action recognition,” in Proceedings of the IEEE international conference on computer vision, 2015, pp. 4041–4049. Navigate to

[12] Y. Du, W. Wang, and L. Wang, “Hierarchical recurrent neural network for skeleton based action recognition,” in Proceedings of the IEEE conference on computer vision and pattern recognition, 2015, pp. 1110–1118. Navigate to

[13] L. Pigou, A. Van Den Oord, S. Dieleman, M. Van Herreweghe, and J. Dambre, “Beyond temporal pooling: Recurrence and temporal convolutions for gesture recognition in video,” International Journal of Computer Vision, vol. 126, no. 2, pp. 430–439, 2018. Navigate to

[14] Z. Ren, J. Meng, J. Yuan, and Z. Zhang, “Robust hand gesture recognition with Kinect sensor,” in Proceedings of the 19th ACM international conference on Multimedia, 2011, pp. 759–760. Navigate to

[15] Jiaqing Liu, Kotaro Furusawa, Tomoko Tateyama, Yutaro Iwamoto, and Yen-wei Chen, "An Improved Kinect-Based Real-Time Gesture Recognition Using Deep Convolutional Neural Networks for Touchless Visualization of Hepatic Anatomical Mode," Journal of Image and Graphics, Vol. 7, No. 2, pp. 45-49, June 2019. doi: 10.18178/joig.7.2.45-49 Navigate to

[16] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” arXiv preprint**arXiv:1706.03762 (arXiv:1706.03762)**, 2017. Navigate to

[17] M. De Coster, M. Van Herreweghe, and J. Dambre, “Sign language recognition with transformer networks,” in12th International Conference on Language Resources and Evaluation, 2020. Navigate to

[18] Mygel Andrei M. Martija, Jakov Ivan S. Dumbrique, and Prospero C. Naval, Jr, "Underwater Gesture Recognition Using Classical Computer Vision and Deep Learning Techniques," Journal of Image and Graphics, Vol. 8, No. 1, pp. 9-14, March 2020. doi: 10.18178/joig.8.1.9-14 Navigate to

[19] GitHub, 2021, last consulted July 29, 2021 at **https://github.com/KurukW/AI_Project1_G7/tree/main/DATA (https://github.com/KurukW/AI_Project1_G7/tree/main/DATA)** Navigate to

[20] A. Ullah, J. Ahmad, K. Muhammad, M. Sajjad, and S. W. Baik, “Action recognition in video sequences using deep bi-directional lstm with cnn features,” IEEE access, vol. 6, pp. 1155–1166, 2017. Navigate to

[21] S. Ji, W. Xu, M. Yang, and K. Yu, “3d convolutional neural networks for human action recognition,” IEEE transactions on pattern analysis and machine intelligence, vol. 35, no. 1, pp. 221–231, 2012. Navigate to

[22] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, “Learning spatiotemporal features with 3d convolutional networks,” in Proceedings of the IEEE international conference on computer vision, 2015, pp.4489–4497. Navigate to

[23] Tkinter documentation: **https://docs.python.org/3/library/tkinter.html (https://docs.python.org/3/library/tkinter.html)** Navigate to

[24] OpenCV : **https://opencv.org/ (https://opencv.org/)** Navigate to

VSIP 2021, November 19–21, 2021, Wuhan, China

ACM ISBN 978-1-4503-8588-6/21/11…$15.00.
DOI: **https://doi.org/10.1145/3503961.3503962 (https://doi.org/10.1145/3503961.3503962)**