Towards a Professional Gesture Recognition with RGB-D from Smartphone

Pablo Vicente Moñivar[12], Sotiris Manitsaris[13], and Alina Glushkova[14]

[1] Centre for Robotics, MINES ParisTech, PSL Université Paris
[2] pablo.vicente_monivar@mines-paristech.fr
[3] sotiris.manitsaris@mines-paristech.fr
[4] alina.glushkova@mines-paristech.fr

Abstract. The goal of this work is to build the basis for a smartphone application that provides functionalities for recording human motion data, train machine learning algorithms and recognize professional gestures. First, we take advantage of the new mobile phone cameras, either infrared or stereoscopic, to record RGB-D data. Then, a bottom-up pose estimation algorithm based on Deep Learning extracts the 2D human skeleton and exports the 3rd dimension using the depth. Finally, we use a gesture recognition engine, which is based on K-means and Hidden Markov Models (HMMs). The performance of the machine learning algorithm has been tested with professional gestures using a silk-weaving and a TV-assembly datasets.

Keywords: pose estimation, depth map, gesture recognition, Hidden Markov Models, smartphone

1 Introduction

The role of professional actions, activities and gestures is of high importance in most industries. Motion sensing and machine learning have actively contributed to the capturing of gestures and the recognition of meaningful movement patterns by machines. Therefore, very interesting applications have emerged according to the industry. For example, in the factories of the future, the capabilities of workers will be augmented by machines that can continuously recognize their gestures and collaborate accordingly, whereas in the creative and cultural industries it is still a challenge to recognize and identify the motor skills of a given expert. Therefore, capturing the motion of workers or craftsmen using off-the-shelf devices, such as smartphones, has a great value. New smartphones are equipped with depth sensors and high power processors, which allow us to record data even without very sophisticated devices.

In this work, we aim to create a smartphone application that allows for recording gestures using RGB or RGB-D images, estimating human poses, training machine learning models by using only few shots and recognizing meaningful patterns. The motivation of this work is to give the possibility to the users to easily record, annotate, train and recognize human, actions, activities and gestures in professional environments.

2 State of art

2.1 Qualitative comparison between pose estimation frameworks

Different benchmarks of 2D human pose estimation are evaluated through annual challenges that aim to improve various key-parameters, such as the accuracy, how the algorithm performs with partial occlusions or using different keypoints, how to detect the pose of big number of individuals in the scene, etc. Some examples are the COCO Keypoint Challenge, MPII HumanPose and Posetrack.

The pose estimation in multi-person scenarios can be carried out by using a top-down or a bottom-up approach. In the first approach, a human detector is initiated and both the joints and the skeleton of each person are calculated separately. ***AlphaPose***[1] is a a top-down method based on regional multi-person pose estimation. Moreover, ***DensePose***[2], aims to map each person and extract a 3D surface from it using Mask R-CNN[3].

On the other hand, bottom-up approaches firstly detect and label all the joints candidates in the frame and secondly associate them to each individual person without using any person detector. ***DeepCut***[4] is an example of bottom-up approach, it uses a convolutional neural network (CNN) to extract a set of joint candidates, and then it labels and associates the joints with Non-Maximum Suppression.

Finally, ***OpenPose*** [5] is a real-time bottom-up approach for detecting 2D pose of multiple people on an image. First, it takes an RGB image and apply a fine-tuned version of the CNN VGG-19 [6] to generate the input features of the algorithm. Second, these features enter a multi-stage CNN to predict the set of confidence maps, where each map represents a joint, and the set of part affinities which represent the degree of association between joints. Lastly, bipartite matching is used to associate body part candidates and obtain the full 2D skeletons.

In our work, we have chosen to use OpenPose. The main reason is that top-down approaches suffer from an early commitment when the detector fails, and the computational power increases exponentially with the number of people in the scene. On the other hand, OpenPose includes a hand skeleton estimation, which has been considered an important perspective in our system. Table 1 shows a list of popular open source methods for 2D pose estimation and the classification obtained in their respective challenges.

Table 1: List of popular open-source frameworks for 2D pose estimation

Method	Benchmark	Precision (%)	Rank
OpenPose	COCO Keypoint challenge 2016	60.5	1
AlphaPose	COCO Keypoint challenge 2018	70.2	11
DensePose	Posetrack multi-person pose estimation 2017	61.2	7

2.2 Machine and deep learning frameworks for mobile devices

A pose estimation embedded to a smartphone still represents a huge challenge for the scientific community. Neither Apple Store nor Google Play propose any application that provides pose estimation. The main reason is the computational power needed for running deep learning algorithms, as well as the lack of powerful graphical devices in smartphones. [7] shows different tests when running deep neural networks on Android smartphones. The use of TensorFlow-Lite, Caffe-Android or Torch-Android frameworks is currently possible for implementing CNNs on smartphones.

Nevertheless, at the end of 2017, Apple made a transition in the world of machine learning by launching the CoreML framework for iOS 11 that enable the running of machine learning models on mobile devices. The performance improved in the next version, CoreML2, at the end of 2018.

Today, there are available applications that do eye tracking based on gaze estimation by using CNNs [8]. Nevertheless, pose estimation requires much more computation power than eye tracking or face recognition. For this reason, the use of an external framework has been considered as a better solution within the context of this work.

2.3 Gesture recognition methods

The implementation of deep learning for gesture recognition has become the common practice and can lead to very good results. The ChaLearn LAP Large-scale Isolated Gesture Recognition Challenge from the ICCV 2017, crowned [9] [10] [11] as the best deep learning algorithms for gesture recognition. However, the need for large training databases is not compatible with the constraints professional gestures where datasets are quite small.

Dynamic Time Warping (DTW) and Hidden Markov Models (HMMs) are machine learning methods that are widely used in pattern recognition. DTW is a template-based approach that is based on a temporal re-scaling of a reference motion signal and its comparison with the input motion signal, such as in [12]. DTW can be good for doing one-shot learning, while HMMs is a robust duration-independent model-based approach.

Thus, in this research we chose to use our previous work, described in [13], which makes use of K-means to model the time series of motion data and HMMs for classifying and recognizing the gesture classes by using the Gesture Recognition Toolkit (GRT)[14].

3 Objectives

The general scope of this work is to build the basis of an application for a mobile device, which allows for data recording using its embedded sensors, estimate the human pose, extract the skeleton and recognize (offline) professional gestures. The pose estimation depends on the camera: RGB for 2D, and RGB-D, either infrared or stereoscopic, for 3D skeletons. The whole process can be controlled by the application. More precisely, the annotation, the joint selection, the skeleton visualization and the projection of the recognition results run locally on the phone while the pose estimation and machine learning algorithms are delegated to a GPU server.

4 Overall pipeline

The smartphone handles the input and output steps of the pipeline. It records the RGB-D frames and shows the results (body skeleton and gesture recognition accuracy). An external motion detection server receive the frames by using websockets and, then, it estimates the skeleton and uses the information provided by the body joints to train and test a gesture recognition engine. The overall architecture is shown in Figure 1.

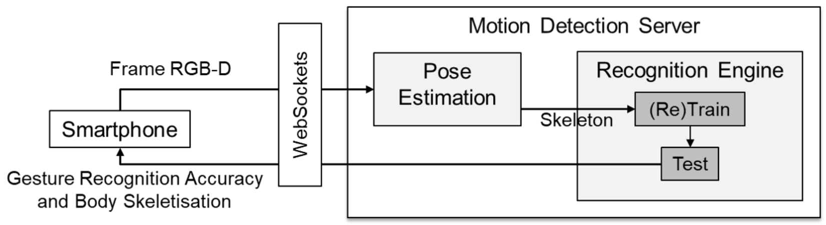


Fig. 1: Architecture of the overall pipeline

4.1 Video recording using the smartphone

A specific module for depth recording has been developed in order to use any iOS device equipped with an RGB and/or RGB-D sensor to record data. More precisely, the new iPhone XS has been used in this work.

On the one hand, iPhone XS uses a dual rear camera, composed by a wide-angle lens and a telephoto lens, to capture the disparity. The normalized disparity is defined as the inverse of the depth. In the Figure 2 on the left is shown how the rear camera captures the disparity. First, the observed object is reflected on the image plane of each camera. Then, the mathematical relation tying the depth (Z), baseline distance (B), disparity (D) and focal length (F), showed in equation 1, results in the normalized disparity. Finally, the iPhone needs to filter and post-process the disparity to smooth the edges and fill the holes, which requires a heavy computation.

$$\frac{B}{Z} = \frac{D}{F} \rightarrow \frac{1}{Z} = \frac{D}{BF} \tag{1}$$

On the other hand, the front camera, named True Depth Camera by Apple, is used to measure the depth in meters directly. It has a dot projector that launches over 30,000 dots onto the scene, generally the user face, which are then captured by an infrared camera. To ensure that the system works properly in the dark, there is an ambient light sensor and a flood illuminator which adds more infrared light when needed. The final result is more stable depth images with a higher resolution. Figure 2 on the right shows the architecture of the True Depth Camera.

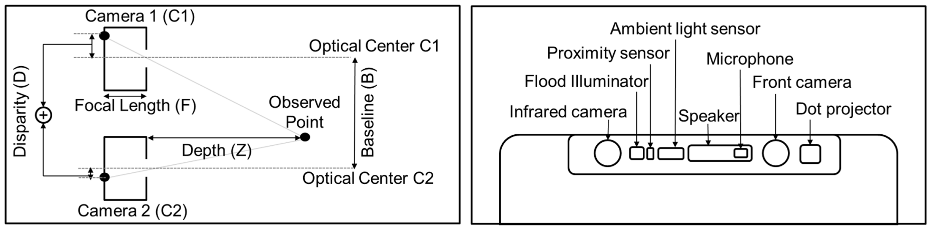


Fig. 2: IPhone XS back camera (Dual Rear Camera) capturing disparity (Left image) and IPhone XS front camera (True Depth Camera) capturing depth (Right image).

Finally, in order to capture the frames with both cameras, it is necessary to use the AVFoundation framework developed by Apple for working with temporal audiovisual media. It is necessary to create a session with at least one capture input (camera or microphone) and one capture output (photo, movie or audio). Then the measured distances (depth or disparity) need to be converted to pixel values in order to visualize depth maps.

4.2 Pose estimation

Once the data is recorded, it is sent to the GPU server. For compression purposes, the RGB-D data is converted to jpeg format, then websockets (RFC 6455) are used to send it. We chose to use the client library Starscream.

The goal of pose estimation is to obtain a series of keypoints that can be used by a gesture recognition engine as input, enabling to train a model that adapts to different situations or environments. OpenPose, in our case, estimates 25 body keypoints, 2x21 hand keypoints and 70 face keypoints. However, some of the estimated keypoints are useless for the recognition, either they are occluded or they do not carry any information about the gesture. Therefore, a joint selection module has been added to the application to give the possibility to the user to select the most appropriate keypoints depending on the use-case.

4.3 Gesture recognition

The joints obtained with the pose estimation and the data obtained from the depth camera are the input to the gesture classification algorithm. To make the recognition invariant to the position of each person in the frame, the neck joint has been taken as a reference point, and any frame without neck has been discarded.

The gesture recognition engine is based on supervised learning. Before making the gesture recognition, a labelled database can be manually created by the user, by manually selecting starting and ending time stamps of each gesture.

Once the database has been labelled, the gesture recognition engine uses k-Means to obtain discrete-valued observations. Then, Hidden Markov Models is used to train the discrete data and to determine a gesture recognition accuracy. The platform GRT has been used for the entire process.

4.4 Smartphone application

The graphical user interface of the application consists on four main modules:

– Recording module: allows to record the sequences and visualize depth maps. An example of this module is shown on Figure 3 right.
– Skeleton Visualization module: allows to visualize the skeleton estimated. Figure 3 left.
– Training module: allows to select the appropriate joints for the gesture recognition.
– Labelling module: interface similar to a photo gallery where you must select the gestures that you want to recognize.

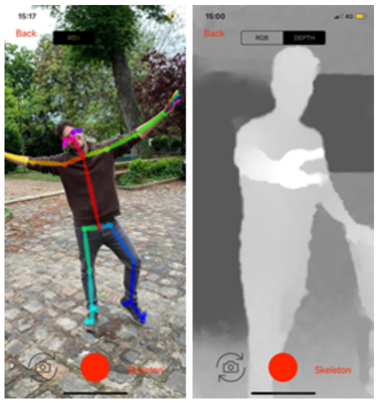
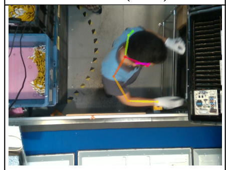
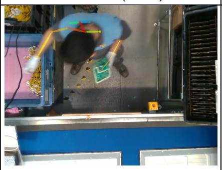
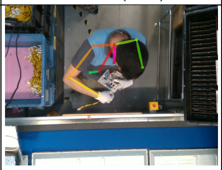
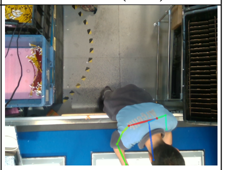


Fig. 3: Example of the smartphone application using the skeleton visualization module and the rear camera (left image) and the recording module of the application enabling the depth visualization with the true depth camera (right image)

5 Datasets

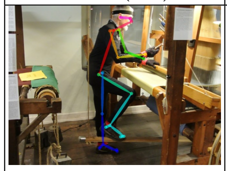
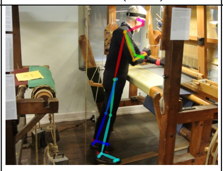
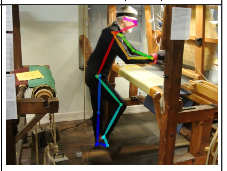
The first dataset used, TV Assembly dataset (TVA), is made up of RGB-D sequences recorded from a top mounted view at a conveyor surface factory. Two different users have been recorded. Each sequence contains around 10.000 RGB frames together with the depth. Table 2 shows the four gestures identified and labeled during the sequences along with the skeleton of each of them.

Table 2: Example of frames from the conveyor surface dataset

Gesture 1 (G1)	Gesture 2 (G2)	Gesture 3 (G3)	Gesture 4 (G4)
			
Take the card from the left side box	Take the wire from the right side box	Connect the wire with the card	Place the card on the TV chassis

The second dataset, Silk Weaving dataset (SW), contains sequences recorded from a lateral view and three different positions in a silk weaver museum. Three clear gestures have been identified and showed in 3 with their corresponding skeleton.

Table 3: Example of frames from the silk weaver museum dataset

Gesture 5 (G5)	Gesture 6 (G6)	Gesture 7 (G7)
		
Press the treadle and push the batten	Move the shuttle sideways	Leave the treadle and pull the batten

6 Results

In order to evaluate the performance of our pipeline and the algorithms we use the 80%-20% evaluation criteria. We randomly divide our dataset in 80% training set and 20% as testing set, repeating this procedure 10 times and computing the average values to generate the confusion matrix. We also use the Recall (Rc), Precision (Pc) and f-score metrics.

The TVA dataset contains 48 repetitions of each gesture and the SW dataset contains 88 repetitions. Moreover, for the gesture recognition engine, we selected 30 clusters for the K-Means algorithm and 4 states for the HMMs, which follow an ergodic topology. Finally, five different tests have been done in order to compare the gesture recognition accuracy using the following criteria: 2D against 3D, 2 joints against 7 joints, different camera positions and mixing gestures from the two datasets.

6.1 Pose estimation comparison using the TVA and SW datasets

A comparison of the pose estimation using the two datasets has been made. We compared the number of frames per gestures with 1. the percentage of frames without any estimated skeleton, thus without estimation at all; 2. the percentage of frames without any reference point, thus without the neck and 3. the percentage of frames having the minimum useful estimation, thus at least the neck.

The results are shown on Table 4 and we can affirm that the lateral views provided by the SW dataset have a better potential, than the TVA dataset, since a full skeleton has been estimated for all the frames. In the top mounted view of the TVA dataset, the algorithm struggled to estimate any information in 43% of frames for G2 and in 36% of frames for G4, because the user is not captured in many frames. With regard to the TVA dataset, we would expect that these results might have an impact in the gesture recognition accuracy. We would also expect that the small duration of the G5 might also affect its recognition accuracy.

Table 4: OpenPose results on the TVA and SW datasets

Dataset	***TV Assembly***				***Silk Weaving***		
Gesture	**G1**	**G2**	**G3**	**G4**	**G5**	**G6**	**G7**
# Frames per sample	80.62	67.89	88.69	164.27	30.85	41.81	66.16
% Frames without any skeleton estimated	0.88	12.27	1.31	14.39	0	0	0
% Frames without reference point (without neck)	2.71	31.05	1.41	22.83	0	0	0
% Frames with at least the minimum useful estimation (at least the neck)	**96.41**	**56.67**	**97.28**	**62.78**	**100**	**100**	**100**

6.2 Gesture recognition comparisons using the TVA dataset: 2D vs 3D and 2 vs 7 joints

The joints selected for training the gesture recognition engine with the TVA dataset are the upper-body joints. Table 5 compares the recognition accuracy by using 2 (wrists) or 7 (wrists, elbow, shoulders and head) joints in the 2D or 3D space. The highest results are obtained by using 7 joints in 3D, while the worst with 2 joints in 2D.

Moreover, as we expected, the low number of frames with the minimal useful pose estimation for G2 and G4 impacted the recognition accuracy for these gestures, while G1 and G3, which have data that give good pose estimation, achieved very high accuracy.

Additionally, on one hand, we notice that, while what we gain with the 3D is not so important compared with the 2D, the potential error in the accuracy (standard deviation) decreases for approximately 40% with the 3D. On the other hand, if we use 7 joints instead of 2, we increase the accuracy for more than 10%, meaning that not mostly the hands are involved into the effective gestures. In any case, the way the 3rd dimension is extracted is biased by the fact that OpenPose is already pre-trained using only the RGB, meaning that a complete re-training of the OpenPose with the depth might give better results. Finally, the number of joints that give better accuracy really depends on the nature of gestures.

6.3 Gesture recognition comparison using three different camera positions from the SW dataset

The accuracy in the Table 6 is 100%, meaning that the recognizer works perfectly for the gestures of the SW dataset. We think that the difference between the accuracy of the two datasets is mostly due two main reasons: 1. the top mounted view used in the TVA dataset and 2. the fact that in a number of frames the user is not captured in the TVA dataset, thus there is no any pose estimation. In addition, the three gestures made in the SW dataset have a greater variance in space than in the TVA dataset.

6.4 Comparison mixing gestures and data from the TVA and the SW datasets

The last experiment we tried is mixing gestures and data from the two datasets. In Table 7, we calculate the precision, recall and f-score for each gesture

Table 5: Comparison in the gesture recognition using different joints and dimension and TV Assembly dataset

2J-2D	*G1*	*G2*	*G3*	*G4*	**Pr(%)**
HMM1	81	1	3	6	89.0
HMM2	0	70	1	29	70.0
HMM3	17	12	53	5	60.9
HMM4	2	14	0	96	85.7
Rc(%)	81.0	72.2	92.3	70.6	76.9 ± 7.9

2J-3D	*G1*	*G2*	*G3*	*G4*	**Pr(%)**
HMM1	80	1	8	3	87.0
HMM2	0	73	1	15	82.0
HMM3	1	28	0	85	66.3
HMM4	1	28	0	85	74.6
Rc(%)	84.2	65.0	87.5	77.3	77.2 ± 4.7

7J-2D	*G1*	*G2*	*G3*	*G4*	**Pr(%)**
HMM1	85	1	4	0	94.4
HMM2	0	84	0	11	88.4
HMM3	4	16	71	8	71.7
HMM4	0	6	0	100	94.3
Rc(%)	95.5	78.5	94.7	84.0	87.2 ± 6.0

7J-3D	*G1*	*G2*	*G3*	*G4*	**Pr(%)**
HMM1	98	1	1	0	98.0
HMM2	0	82	1	5	93.2
HMM3	4	7	68	7	79.1
HMM4	6	114	0	94	82.5
Rc(%)	90.7	78.8	97.1	88.7	88.1 ± 3.4

Table 6: Gesture recognition using all joints and 2 dimensions on Silk Weaving dataset

2J-2D	*G4*	*G5*	*G7*	**Pr(%)**
HMM1	183	0	0	100
HMM2	0	166	0	100
HMM3	0	0	181	100
Rc(%)	100	100	100	100 ± 0

when we train the gesture recognition engine with samples from both datasets. As a general conclusion, we notice that there is a decrease of accuracy for every gesture. Nevertheless, this decrease is not important given the fact that we have many users, thus a high variance in the way the gestures are executed, and different camera positions.

Table 7: Gesture recognition mixing gestures from SW and TVA datasets and using 7 joints and 2D

	G1	*G2*	*G3*	*G4*	*G5*	*G6*	*G7*
Pr(%)	81.9	56.7	80.9	62.8	98.3	98.8	99.5
Rc(%)	84.6	77.5	68.5	69.9	96.6	97.6	95.4
f-score(%)	**83.2**	**65.5**	**74.2**	**66.2**	**97.4**	**98.2**	**97.4**

7 Conclusions and Future work

In this work, we developed the first version of a smartphone application that allows users to record human motion using the RGB or the RGB-D sensors, annotate them, estimate the pose and recognize them. The use of a smartphone in industrial or professional context is much easier then the use of highly intrusive body tracking systems. The long term goal of this application is to permit to industrial actors to record their own professional gestures, to annotate them and to use a user-friendly system for their recognition. We developed a module that extracts the 3rd dimension from a depth. We concluded that the 3rd dimension improves the recognition stability, decreasing by 40% the standard deviation in the accuracy. In addition to this, we observed that while hand are involved in most of cases, using 7 instead of 2 joints can give better recognition results, especially when the camera is top-mounted.

Our future work will be focused not only on improving the application, thus improving the interface, but also on the further development of professional gestures dataset. With a large dataset, we will be able to consider also the use of Deep Learning. Finally, we also plan to extend our pose estimation and gesture recognition system towards the direction of using finger motion as well.

Acknowledgement

The research leading to these results has received funding by the EU Horizon 2020 Research and Innovation Programme under grant agreement No. 820767, CoLLaboratE project, and No. 822336, Mingei project. We acknowledge also the Arelik factory and the Museum Haus der Seidenkultur for providing as with the use-cases as well as the Foundation for Research and Technology Hellas for contributing to the motion capturing.

References

1. H. Fang, S. Xie, and C. Lu, "RMPE: regional multi-person pose estimation," 2016.
2. R. A. Güler, N. Neverova, and I. Kokkinos, "Densepose: Dense human pose estimation in the wild," 2018.
3. W. Abdulla, "Mask r-cnn for object detection and instance segmentation on keras and tensorflow," *GitHub repository*, 2017.
4. L. Pishchulin, E. Insafutdinov, S. Tang, B. Andres, M. Andriluka, P. V. Gehler, and B. Schiele, "Deepcut: Joint subset partition and labeling for multi person pose estimation," 2015.
5. Z. Cao, G. Hidalgo, T. Simon, S. Wei, and Y. Sheikh, "Openpose: Realtime multi-person 2d pose estimation using part affinity fields," 2018.
6. K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014.
7. A. Ignatov, R. Timofte, W. Chou, K. Wang, M. Wu, T. Hartley, and L. V. Gool, "AI benchmark: Running deep neural networks on android smartphones," 2018.
8. K. Krafka, A. Khosla, P. Kellnhofer, H. Kannan, S. Bhandarkar, W. Matusik, and A. Torralba, "Eye tracking for everyone," 2016.
9. L. Zhang, G. Zhu, P. Shen, and J. Song, "Learning spatiotemporal features using 3dcnn and convolutional lstm for gesture recognition," *ICCV workshop*, 2017.
10. H. Wang, P. Wang, Z. Song, and W. Li, "Large-scale multimodal gesture recognition using heterogeneous networks," *ICCV 2017 Worhshop*, pp. 3129–3137, 2017.
11. P. Wang, W. Li, S. Liu, Z. Gao, C. Tang, and P. Ogunbona, "Large-scale isolated gesture recognition using convolutional neural networks," 2017.
12. A. Corradini, "Dynamic time warping for off-line recognition of a small gesture vocabulary," pp. 82–, 2001.
13. E. Coupeté, F. Moutarde, and S. Manitsaris, "Multi-users online recognition of technical gestures for natural human–robot collaboration in manufacturing," *Autonomous Robots*, Feb 2018.
14. N. Gillian and J. A. Paradiso, "The gesture recognition toolkit," *Journal of Machine Learning Research 15*, 2014.