International Journal of User-System Interaction 12(1) 2019, 54-66 © MatrixRom

Extracting human features to enhance the user experience on a training station for manual operations

Alin-Marius Cruceat[1], Alexandru Matei[1], Bogdan-Constantin Pîrvu[1], Alexandru Butean[2]

[1] Connected Intelligence Research Center, [2] Department of Computer Scienc
Lucian Blaga University of Sibiu
10 Victoriei Boulevard, Sibiu, Romania
E-mail: {alin.cruceat, alex.matei, bogdan.pirvu, alexandru.butean}@ulbsibiu.ro

Abstract. This paper presents an image analysis approach for extracting human features. These features are used in a physical training station for manual operations to support training environment adaptation based on predefined user profiles. A Kinect for Microsoft Xbox provides the RGBD data flow that is used to process the user features utilized furthermore for adjusting a training table and thus supporting the user in executing his training routine in an ergonomic and efficient manner. The development process and the integration of the training station with other smart devices will rely on these human features and specific user profiles to generate adapted training instructions.

Keywords: Training systems, motion training systems, image processing, feature extraction, visual feedback, Kinect for Microsoft Xbox 360.

1. Introduction

Human-computer interaction (HCI) is what the future brings through automatization. Current technological progress in speech recognition, gesture detection, augmented reality and virtual reality lay the foundation of a new era in which human activity can be drastically supported in daily life.

HCI applications have as general requirements the recognition, identification and feature extraction of the human and objects involved. This is also the case in the training station for manual operations that are currently being developed. The main target is to create training scenarios through which a trainee can easily learn without human intervention how to correctly assemble a product without any previous knowledge about it.

2. System overview

In its current version, the training station for manual assembly operations (see Figure 1) is composed out of a wide touch screen integrated into a table with adjustable height and multiple sensors and gadgets attached to the frame. The training scenario involves step-by-step user assistance with simple instructions that need to be followed to correctly assemble a modular tablet (current experiment). More details about the modular tablet can be found in a previous article Stanciu et. al. (2018).

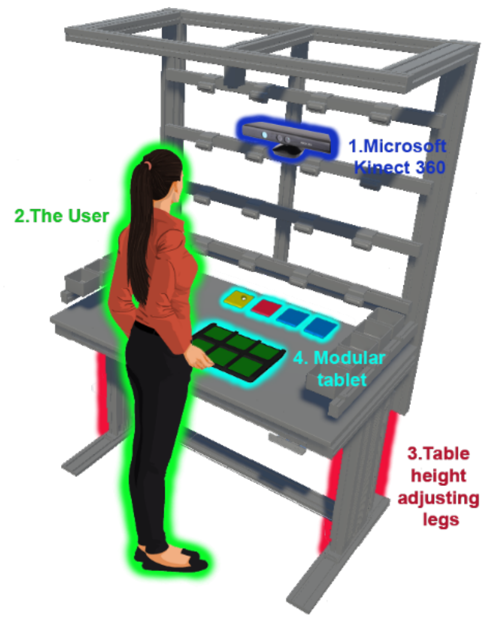

Figure 1. Manual assembly training station overview

The training station software architecture relies on decentralized microservices, following the architectural details from Francesco et. al. (2017). A microservice is a standalone module of an application that has a simple structure, can run independently and is easy to understand and manage. They can be instantiated multiple times depending on the specific service demand, following the containerized method presented by Venugopal (2017). The great advantage of this approach is that an application having multiple microservices does not completely shut down if one of the

microservices is not working properly as described in the Kinect Sensor Setup (2019). Moreover, using a service-oriented approach, the training scenario can be easily adapted by enabling and disabling microservices to ensure the best usability of our training station for various target groups and industry use-cases. For example, younger trainees might be satisfied by having several new features, while the elderly ones might find some of them distracting and thus resulting in lower training efficiency.
The implementation for extracting human characteristics is composed of a microservice containing the human detection implemented in C# and a separated Python script used for image analysis. The Python script is launched by the microservice at a specific time when certain conditions are met. The main libraries that are used for implementing this are the following:

- *Microsoft Kinect SDK* described by Microsoft Windows app development (2019) - is used for user identification via skeleton tracking and provides the coordinates for the joints;
- *Cvlib ,* OpenCV based library (2019) - used for selecting the faces of the users from the RGB data flow and also for the image pre-processing tasks of extracting human features.

3. User detection

For human detection, a Microsoft Kinect 360 device is integrated into the training system so that when the table is set up according to the height of the user, the user would have the device standing face to face with him (see Figure 1). Once the microservice is launched, the Kinect device must be connected. Next, people in Kinect's field of view are searched for and the closest one is selected as the focused user. After a user was selected, a signal from the user is needed to trigger the table adjustment process and the user features extraction.

4. Human Feature Extraction

The features that we estimate are:

- *Distance between the user and the training table* – used for starting the table height adjustment once the user is standing at a certain distance away from the training table

- *User elbow height from ground from standing position* – used for setting the height of the table
- *User height from standing position*
- *User gender*
- *User age*
- *User hair*
- *User skin color*

The *user height, gender*, *age*, *hair,* and *skin color* are features relevant for the user profile, features by which a certain trainee can be identified and track his progress and assign adapted training tasks or instructions. More details about the user profiles are out of the scope of this current paper. In order to make a correct table adjustment, the trainees need to keep a still position for a couple of seconds from the moment of reaching the predefined distance to the training table. If the trainee is standing too close or too far away from the Kinect device, the training system will instruct him to come closer or to move further away until he will be sitting at a proper distance.

Once the microservice is being started and the user triggers the start of the training the C# procedure takes the dataflow from Kinect and looks at the depth data, checking out whether the user is standing at 1.8 meters away (with ± 0.1 meters threshold) from the training table. This distance has been set in order to respect 2 criteria:

The face of the user from the RGB image from Kinect is well defined and clear - the closer the user he is by the camera the clearer are the features of the face allowing a better age and gender estimation;

The user distance from the device is enough for the user to be fully detected and correctly estimate the skeleton points as described in Kinect Sensor Setup (2019).

Once this distance condition is fulfilled, the user height is computed, the estimation of the table adjustment (i.e. table height) and finally a frame from the image flow is saved. A request is made to the Python script with the saved image as a parameter for making the feature extraction (see Figure 3).

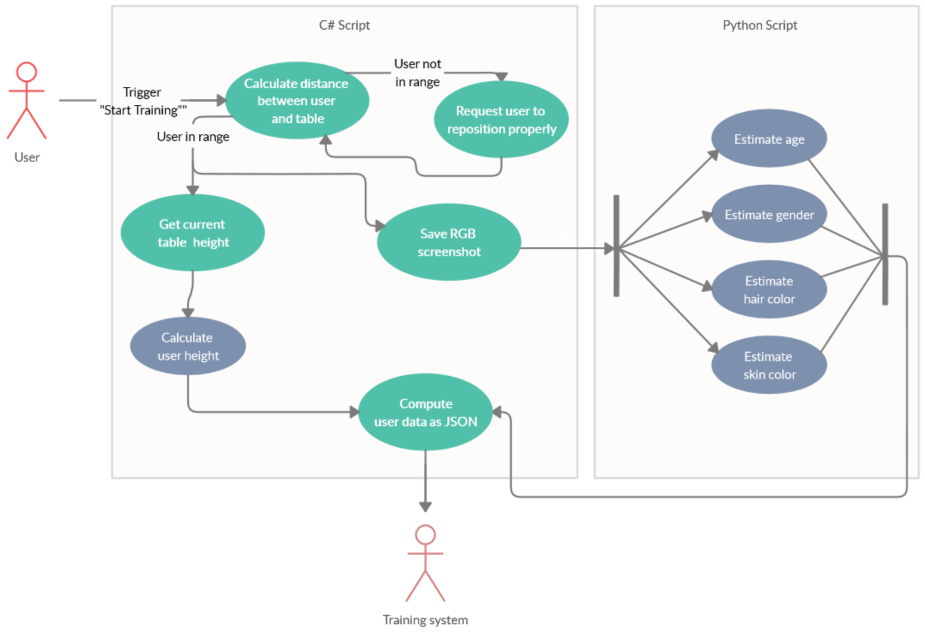

Figure 3. User Data Estimation Routine Logic

4.1 User Selection

Microsoft Kinect SDK v1.8 already comes with an implemented human detection algorithm which returns a data flow named Skeleton Points containing the calculated spatial coordinates of the joints of every human detected. From this data, the user that is the closest to the training system can be selected.

The second part of the user selection is being made by the Python script which works autonomously from the C# part of the microservice. Inside this script, a library called *cvlib* as presented by Ponnusamy (2019). Cvlib comes with a pre-trained model of a deep neural network that detects human faces from images and for each one of them offers information about the gender.

In order to get the main user from our image when other people are present, the area of each of the faces is calculated based on the returned values of the *detect_face* method of cvlib. Knowing that the closest person to the device is always the trainee and that the training scenarios are designed for adult people, the person with the greatest face frame area was selected.

4.2 User Height Estimation

The table height is set by a procedure at the beginning of the training session enabling the user to maintain an ergonomically correct position that allows him to easily reach all the modules that he needs to use to finalize the training. Setting up the training table at the corresponding height is made corresponding to ergonomic standards as described in Lifespan (2019) by having the table at the height of the user's elbow, allowing him to rest his arms on the table. Also, for analyzing the posture we took into consideration the information presented by Rahman (2014) and basic applied ergonomics introduced by Karhu et. al. (1977).

In Figure 4a, a representation of the joints from the Skeleton dataflow is presented. From this representation relevant are the elbow joint (lowest coordinate between left and right) and the head joint. The position of the head represented by the y coordinate of the head joint can be converted to user's height value, by adding it up to the previously provided table height. Because of this factor, the user height was one of the easiest elements to calculate, especially since the current table height is always known and that information about the user head joint is easily accessible. The training table comes equipped with a central command unit which provides the current height of the table. With all these parameters defined, the height of the user can be estimated using the following formula:

$$uh = th + (t + j) / d$$

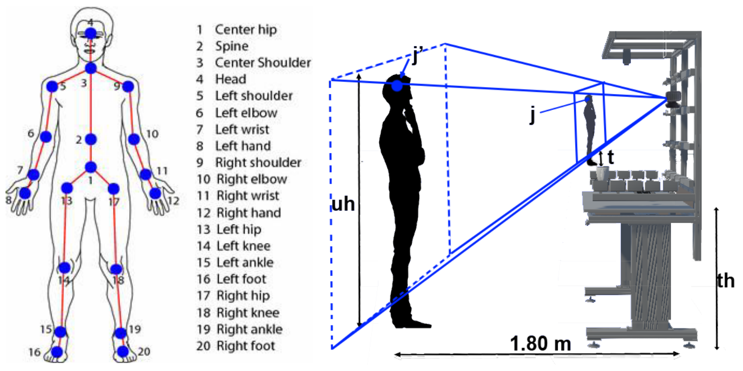

Figure 4. a) Skeleton points described by Bilban et. al. (2017) and

b) Height equation parameters, visual

The equation has the following parameters:

- uh - the user height (in meters)
- th - the current table height setup (in meters)
- t - approximated Kinect frame base height from table's current setup (in pixels)
- j - the user head joint's y coordinate in the Kinect frame (in pixels)
- d - estimated meters per pixels at the requested 1.80 m distance (meters per pixel)

Figure 4 b) gives a spatial representation of the equation's parameters for a better contextual understanding (related to the skeleton points).

4.3 User Gender Estimation

For gender estimation, we use cvlib library as presented by Ponnusamy (2019). For each detected human in the image, the confidence that is one of the two genders is returned. Using this information, we select the gender that has the greatest confidence. Not unexpected, the greater the resolution of the image, the better the confidence of the estimation is because of the high amount.

4.4 User Age Estimation

The age estimation was implemented also using a Convolutional Neural Network (CNN) based on *LearnOpenCV.com* methodologies described by Gupta (2019). This CNN predicts one of the following age groups: (0 – 2), (4 – 6), (8 – 12), (15 – 20), (25 – 32), (38 – 43), (48 – 53), (60 – 100). The precision of the estimation of this algorithm is given by the resolution of the processed image, higher resolution meaning a higher amount of detail resulting in higher estimated precision by the model.

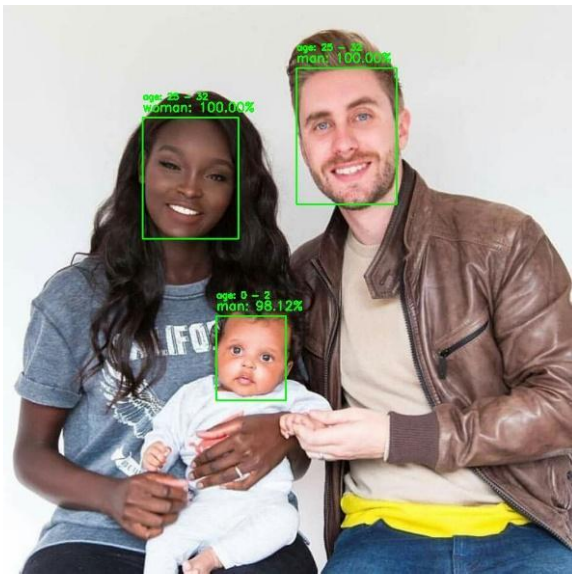

Figure 6. Age and gender estimation - high-resolution image

On our first attempt to test the algorithms we used images gathered from the internet as input containing people of diverse ages and genders, and because of the high resolution, the estimations were perfect. One example of these results can be seen in Figure 6. Switching to Microsoft Kinect as the image providing device, we conducted experiments using 10 people of age range 20-35 years, 5 of them male and 5 females. The results are presented in the next section and visible in Figure 7.

4.5 User hair and skin color estimation

Estimating the hair and skin color was problematic because the selected face frame was not always containing just those colors, but also the colors of the background, jewelry, hair ornaments, or even other high contrast colors like white teeth on dark-skinned people. In order to improve our estimation, the number of colors from the image was reduced by using K-Means clustering. In this way, the outcast colors from our frame were eliminated and the color palette was reduced to a smaller number.

Another critical situation was considering the length of the hair of the user. The issue became clear for men and short-haired women where the face frame also contained a portion of the background, drastically influencing the clustering process. In order to find a good solution for this issue, we decided to force a white background in the frame, easy to detect and eliminate, so that only 3 remaining colors would represent the skin of the face, the hair and some other shadows of the face.

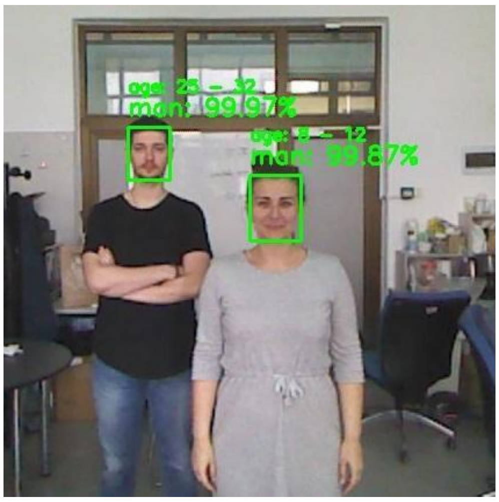

Figure 7. Age and gender estimation

For implementing this all colors were reduced into a cluster of 4 and checked whether any of them is close enough to white (HTML red green and blue - RGB- color code values higher than 240). If such a case is found, it means the user has short hair or tied in the back of his/her head, so we eliminate it, remaining just with the relevant colors for our analysis. On the other case if there is no color in the range of the defined white it means that the background is covered by the user's long hair, so we remake the clustering, grouping the colors straight to 3 colors, the only colors we need and available in our frame. The final result of the clustering is visible in Figure 8. The first image is an example of a person with short hair, where the background enters the frame and needs to be treated separately, whereas the second image represents a person with long hair in which the frame contains strictly the required colors.

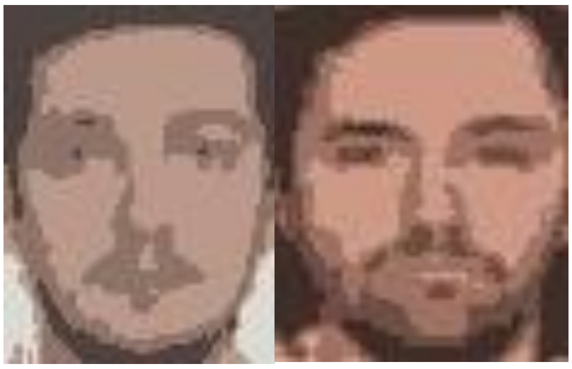
Figure 8. K-Means clustering for skin and hair color estimation

After obtaining these 3 colors, the next step was to assign the colors to the 3 defined categories. Because there is a frame containing only the user's face, our hypothesis is that the most dominant color should be either the color of the skin or the color of the skin shadow.

On this premise, it was assumed that any of the skin or its shadow color can be accepted to be the expected skin color. Thus, the remaining color which is significantly different to the other two can only be the hair color. This approach was implemented by representing the RGB colors in the 3D space and computing the distances between them. Both skin and its shadow color should be in this case, the two colors that have the smallest distance between them among the three.

5. Experimental results

While still in the conceptual stages, in order to test the efficiency of the solution, we conducted an experiment with 10 people of ages between 20 and 35 years old, 5 men and 5 women.

In Figure 8 it can be observed that the clustering algorithm made the colors of the hair and skin easily distinguishable to the bare eye. Just as we expected, the result is that the hair color is very different from the skin and lighted skin colors.

In order to test the efficiency of the feature extraction methods, a series of experiments on a group of subjects were conducted. The following combination of features was of interest:

- *Height* (within ± 0.05 meters threshold error)
- *Gender*
- *Age*
- *Skin color* (with ± 10 bits threshold error on the RGB pixel format)
- *Hair color* (with ± 10 bits threshold error on the RGB pixel format)

Table 1 - Estimation error

	Height	Gender	Age	Skin color	Hair color
Men	6.67%	0.00%	51.11%	6.56%	6.56%
Women	5.17%	34.48%	65.52%	10.53%	10.53%

Table 1 contains the total percentage of cases when the user features were not matching because of some poor estimations. The first obvious and disturbing result that we see is the age estimation, having a very poor result because of the low image quality and the lack of details. The predictions of the gender had also some results on which we could debate. One of the women had difficulties in having her gender correctly estimated (see Figure 7). For her, 9 out of 10 times she was estimated as being a man of age between 8-12. If we were to take her data out of the result, the gender detection algorithm error would go down to 21.28%. Testing on images of the same girl but made with a better resolution camera, a 100% accuracy in gender recognition was achieved, supporting our assumption that the lack of detail from the low-resolution Kinect camera was faulty for misinterpreting the gender and the age. This also proves that the training of the model is well done, as expected and that by changing the provider of the RGB dataflow, the predictions will definitely improve.

Since the gender estimation had errors on women, we chose to separate the results for men and women. The first element that was observed in our results is that the errors for estimating the hair color and skin color are the same, mainly because of our assumption that the skin color and the skin shadow color can be approximated as the same.

Here can be asserted that we initially overestimated the efficiency of the color detecting algorithm. As seen in Table 1, the skin and hair estimation algorithm had a few misinterpretation results and all of this because of people with hair covering their forehead, reason why the face frame would contain a higher percentage of hair color than face color.

6. Conclusion

This paper described an approach to extract human characteristics (height, age, gender) using image-based devices (depth, color). The results are bounded by a training station for industry workers were the presented solution is one of the many modules (services) that can be accessed in order to form a large variety of training use-cases for manual assembly operations.

As shown in the experiments section, some of the problems were caused by the low resolution of Kinect RGB camera which made the gender and age detection unreliable in some cases. Since the precision of the algorithms is determined by the details in the image, by replacing the Kinect 360 camera with Kinect One will result in improving the results of our human characteristics extraction and we are bound to accomplish this in the next few months. Also, in the next version, we consider using the methods presented by Haibin et. al. (2018)

The performance of the color detection can also be improved in the future by tracking the position of the color and see whether it is situated close to the center of the image, representing the face, or on the outside of it, representing the hair. This solution might be what we need to eliminate those special cases that were poorly estimated in the experiment session.

By having a microservices approach, linking the user profiles with real-time user data (based on e.g. biosensors) and combining them with training data from Product Lifecycle Management systems, we are confident that very good training performance at significantly lower costs compared with classical approaches can be achieved in the future.

Acknowledgment

This work is supported through the DiFiCIL project (contract no. 69/08.09.2016, ID P_37_771, web: http://dificil.grants.ulbsibiu.ro), co-funded by ERDF through the Competitiveness Operational Programme 2014-2020.

References

Bilban, M., Arikan, H., and Uzun, Y. (2017). An Example with Microsoft Kinect: City Modeling with Kinect. Journal of Multidisciplinary Engineering Science and Technology (JMEST). 4. 7554-7556.

Francesco, P.D., Malavolta, I., and Lago, P., "Research on Architecting Microservices: Trends, Focus, and Potential for Industrial Adoption," *2017 IEEE International Conference on Software Architecture (ICSA)*, Gothenburg, 2017, pp. 21-30.

Gupta, V. Learn OpenCV. Retrieved 2019, from: https://www.learnopencv.com/age-gender-classification-using-opencv-deep-learning-c-python

Karhu, O., Kansi, P. and Kuorinka, I. (1977) Correcting Working Postures in Industry: A Practical Method for Analysis. Applied Ergonomics, 8, 199-201.

Kinect Sensor Setup. Retrieved 2019, from: https://support.xbox.com/en-GB/xbox-360/kinect/kinect-sensor-setup.

Haibin, L., Yuchen, Y., Wenhua, D., and Ping, F., “Age Estimation of Face Images Based on CNN and Divide-and-Rule Strategy,” Mathematical Problems in Engineering, vol. 2018, Article ID 1712686, 8 pages, 2018.

Microservices Architecture: Advantages and Drawbacks. Retrieved 2019, from: https://cloudacademy.com/blog/microservices-architecture-challenge-advantage-drawback

Microsoft, Kinect - Windows app development. Retrieved 2019, from: https://developer.microsoft.com/en-us/windows/kinect.

OpenCV (2019). Retrieved from: https://opencv.org.

Ponnusamy, A.. GitHub repository. Retrieved 2019, from: https://github.com/arunponnusamy/cvlib

Rahman, C., (2014). Study and Analysis of Work Postures of Workers working in a Ceramic Industry through Rapid Upper Limb Assessment (RULA). International Journal of Engineering and Applied Sciences 2305-8269. 5. 14-20.

Stanciu, S., Petruse, R., and Pîrvu, B. (2018). Development Overview of a Smart Customizable Product, ACTA Universitatis Cibiniensis, 70(1), 36-42. doi: 10.2478/aucts-2018-0006.

Lifespan, Standing Desk Posture Treadmill Desk Ergonomics. Retrieved 2019, from: https://www.lifespanfitness.com/workplace/resources/articles/treadmill-desk-ergonomics.

Venugopal, M. (2017). Containerized Microservices architecture. International Journal of Engineering and Computer Science, 6(11), 23199-23208.

.