Unsupervised Facial Expressions Recognition and Avatar Reconstruction from Kinect

Bassem Seddik¹, Houda Maâmatou¹, Sami Gazzah¹, Thierry Chateau², Najoua Essoukri Ben Amara¹

¹SAGE Research Unit
ENISo School of Engineers of Sousse
University of Sousse, Tunisia
bassem.seddik@ieee.org – houda.maamatou@yahoo.fr – sami_gazzah@yahoo.fr – najoua.benamara@eniso.rnu.tn

²Blaise Pascal University
63170 AUBIERE, France
thchateau@lasmea.univ-bpclermont.fr

Abstract—This paper presents a solution capable of recognizing the facial expressions performed by a person’s face and mapping them to a 3D face virtual model using the depth and RGB data captured from Microsoft’s Kinect sensor. This solution starts by detecting the face and segmenting its regions, then, it identifies the actual expression using EigenFaces metrics on the RGB images and reconstructs the face from the filtered Depth data. A new dataset relative to 20 human subjects is introduced for learning purposes. It contains the images and point clouds for the different facial expressions performed. The algorithm seeks and displays automatically the seven state of the art expressions including surprise, fear, disgust, anger, joy, sadness and the neutral appearance. As result our system shows a morphing sequence between the sets of 3D face avatar models.

Index Terms— Facial expressions, 3D reconstruction, Kinect, point clouds, EigenFaces.

I. INTRODUCTION

The field of facial emotions analysis is still an active issue in the latest research works. The new tools and technologies accessible to scientists are opening new paths for innovative solutions. Its applications include the design of better human/machine interfaces (Fig. 1), video gaming, computer generated animations and biometrical identification [1].

Literatures reviews classify the existing works according to their data entries, pre-processing steps, extracted feature types, classification methods and their post-processing output showing the final labeling results [2]. Most of the precedent works use the FACS system which introduces seven human expressions: happiness, fear, disgust, anger, surprise, sadness and the neutral state [3].

Pushpaja and Devendra [4] present a solution for the recognition of facial expressions based on the Eigenface technique and neural networks. They calculate the Eigenvectors to create the Eigenfaces then project all the learning images in the generated space to calculate their weights. The computed metrics are used as face descriptors, then, neural networks are employed for the facial expressions recognition.

More recent facial expression analysis toolboxes make use of the Active Shape Models (ASM) and the Active Appearance Models (AAM) [5] to learn about both the textural and geometric distributions inside 2D images.

Other solutions like the one developed by Blanz and Vetter [6] are oriented towards a user assisted generation of textured 3D face models from one or more photographs using dense one-to-one registration and modulation with an existing model. This idea allowed the introduction of more sophisticated facial animation retargeting solutions like those based on feature points extraction [7].

Other approaches use an example-based facial rigging algorithm on the same sensor. They proceed by an optimization of the gradient space in order to generate a blendshape model from a limited set of 3D facial poses that can be enriched later [8].

The same idea has been reused in order to create a system generating a virtual 3D avatar’s face that performs in real time [9]. It starts by prompting the user to save 19 different face expressions and feed them to the software using Microsoft’s Kinect sensor. To succeed in this process they combine the animation priors with geometry and texture registration by the mean of the expectation maximization algorithm. Their solution deals efficiently with the noisy data coming from the infra-red sensor but presents the limitation of requiring user supervised feeding of data in order to succeed in the learning process. It is obvious that a more automatic learning solution is needed.

Other recent works like the one proposed by Breidt et al. [10] make use of the Kinect sensor in order to provide an online reconstruction of the face state into a 3D virtual face.
To achieve robustness towards the sensor noise, they propose a semantic based correspondence between the depth data and determined target on the mesh to be animated.

In this paper, we will present a facial expression recognition and reconstruction solution that doesn’t require any prior data feeding from the user. It makes use of both the depth and RGB Kinect flows in order to animate a 3D face of an avatar. An overview of the different processing stages of our solution is illustrated in Fig. 2. The main contribution of this work is the unsupervised expression recognition and reconstruction from the Kinect sensor. We also introduce a new dataset of facial expressions containing both 2D images and 3D point clouds. Finally, we create a JAVA interface that shows a 3D avatar performing the animation corresponding to the actual user.

In the second section of this paper, we will present the details of our processing chain. The third section will describe our collected dataset and expose our experiments and results. Discussion and conclusions will be presented at the end.

II. SOLUTION DEVELOPMENT

Our proposed solution starts by acquiring all the possible information from the Kinect sensor including RGB, depth and Infra Red (IR) flows. It applies a calibration step in order to get the different acquisitions aligned in the same coordinate space, and then goes for global bounding box face detection and searches for the different face parts inside of it. After that, an expression recognition process is applied using the Eigenface technique. Once the decision is made online about the mood of the user sitting opposite to the sensor, a 3D face avatar imitates the face expression using a morphing function between different pre-prepared models. Figure 2 gives an overview of our processing pipeline.

A. Depth and RGB calibration

In order to correctly extract the depth and RGB data corresponding to the face, we start by correcting the deviation caused by the slight shifting between the two Kinect sensors as in Fig. 3-a. To achieve this, we used the chessboard based calibration solution (see Fig. 3-b) offered by the OpenCV computer vision library, where the size and position of each square in the image are initially well known. We extracted 80 images coming from the Infra-red and RGB sensors with the Kinect kept static. We generated then the xml configuration files describing the distortion and the intrinsic Kinect parameters, and used them to adjust the existing deviation between the two obtained viewports as shown in Fig. 3-c. We also tested the calibration functionality offered by the OpenNI library [11] and integrated it in our workflow for ease of use.

B. Face detection and tracking

To detect the existence of a face in the Kinect video stream, we need a face detector (classifier) to look for features relating to face. In our case, we avoided the learning by exploiting the frontal and profile face classifiers offered by the OpenCV library [12]. They are the result of an Adaboost cascade learning process based on weak Haar-like features. The strength of this technique resides on its real time detection rates. Once the face detected, its bounding box details are acquired and passed to the next stage.

C. Face parts extraction

Emotional signs are localized in delimited face regions such as eyebrows, eyes, cheeks, chin, nose and mouth. In

Fig. 2. The processing steps of our pipe-line. We learn about expressions from 2D and 3D inputs and reconstruct the found emotion on a 3D avatar face.

Fig. 3. (a) Non-aligned images, (b) Chessboard images used to extract the calibration parameters between the two RGB and Depth entries, (c) Aligned images result after the calibration step

Fig. 4. Different facial regions extracted using the structure presented in Tab.1 and examples of saved images containing the face parts
order to separate those regions, we combined spatial relative positions [13] with Haar-cascades based eye and mouth detection in order to extract and save separately each facial region relative to a given expressive image. The geometry of a human face is always formed from top to bottom of a forehead, eyebrows, eyes, nose, mouth and chin. And in the horizontal direction, at the level of the nose, it presents the left ear, left cheek, nose, right cheek and right ear. Based on this distribution, we proposed the spatial segmentation of the face presented in Fig. 4. The face detection stage returns, according to the specifications of Tab. I, the (x, y) coordinates of the point i, the width (W) and the height (H) of each region. For example The mouth bounding box is determined using the following calculations:

\[
\text{Position}_{\text{mouth}}(x) = (x_1 - x) \times \frac{100}{W} \quad (1)
\]
\[
\text{Position}_{\text{mouth}}(y) = (y_1 - y) \times \frac{100}{H} \quad (2)
\]
\[
\text{Width}_{\text{mouth}} = (x_2 - x_1) \times \frac{100}{W} \quad (3)
\]
\[
\text{Height}_{\text{mouth}} = (y_2 - y_1) \times \frac{100}{H} \quad (4)
\]

D. Expression processing

1) Eigenface expression classification:

This step uses the Eigenface classification algorithm initially developed by Turk and Pentland for face recognition and adapted in recent researches to face expressions recognition by Robert et al. [14]. The Eigenface paradigm supposes that any desired face \( x \) can be reconstructed from a mean image \( m \) by adding a number of details \( u_i \). It reduces the contents of every processed image into set of vectors. Then the eigenvectors are obtained using the covariance matrix and PCA analysis is applied in order to produce the mean-like image. Any new face is then compared with it using simple distance calculation.

The output images of the PCA are called "EigenExpressions". Examples of training expressions and eigenExpressions generated are shown in Fig. 5.

2) Point cloud generation from depth:

The depth sensor generates a 640 x 480 grid that describing the distance of each pixel towards the IR projector. This map contains the Z coordinates of each point in the scene in addition to all the data relative to the human face and its surrounding background/objects. As shown in Fig. 6, this information allows us to calculate the x and y coordinates according to the field of view aperture.

To achieve our point cloud reconstruction, we used the following equations in order to generate the x and y coordinates from the depth data [15]:

\[
x = z \times 2 \times \tan \left( \frac{\text{fov}_Y}{2} \right) \times \frac{y_p}{\text{yres}} \quad (5)
\]
\[
y = z \times 2 \times \tan \left( \frac{\text{fov}_X}{2} \right) \times \frac{x_p}{\text{xres}} \quad (6)
\]

Where \( x_p \) and \( y_p \) denote the coordinates of a given pixel, \( x_{\text{res}} \) and \( y_{\text{res}} \) save the horizontal and vertical resolution of the depth map, and, \( \text{fov}_Y \) and \( \text{fov}_X \) indicate the horizontal and vertical field of vision of the Kinect (in radians).

E. Face reconstruction

1) Filtering of the face point cloud:

After generation of the pcd file containing the point cloud of the hole scene, we obtain the data illustrated in Fig. 7-a. We perform, then, two stages of filtering in order to obtain the clouds relative to the face. First, we start by selecting

<table>
<thead>
<tr>
<th>Direction</th>
<th>Region</th>
<th>\text{Pos}_X %</th>
<th>\text{Pos}_Y %</th>
<th>Width %</th>
<th>Height %</th>
</tr>
</thead>
<tbody>
<tr>
<td>vertical</td>
<td>Forehead</td>
<td>8</td>
<td>0</td>
<td>83</td>
<td>22</td>
</tr>
<tr>
<td></td>
<td>Eyebrows</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Left</td>
<td>6</td>
<td>24</td>
<td>37</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>Right</td>
<td>56</td>
<td>24</td>
<td>37</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>Eyes</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Left</td>
<td>17</td>
<td>33</td>
<td>25</td>
<td>13</td>
</tr>
<tr>
<td></td>
<td>Right</td>
<td>58</td>
<td>33</td>
<td>25</td>
<td>13</td>
</tr>
<tr>
<td></td>
<td>Nose</td>
<td>36</td>
<td>45</td>
<td>27</td>
<td>22</td>
</tr>
<tr>
<td></td>
<td>Mouth</td>
<td>33</td>
<td>70</td>
<td>33</td>
<td>17</td>
</tr>
<tr>
<td></td>
<td>Chin</td>
<td>31</td>
<td>90</td>
<td>38</td>
<td>14</td>
</tr>
<tr>
<td></td>
<td>Left ear</td>
<td>0</td>
<td>31</td>
<td>8</td>
<td>33</td>
</tr>
<tr>
<td></td>
<td>Right cheek</td>
<td>64</td>
<td>44</td>
<td>20</td>
<td>25</td>
</tr>
<tr>
<td></td>
<td>Right ear</td>
<td>92</td>
<td>31</td>
<td>8</td>
<td>33</td>
</tr>
<tr>
<td>Horizontal</td>
<td>Face</td>
<td>0</td>
<td>0</td>
<td>100</td>
<td>100</td>
</tr>
</tbody>
</table>

Fig. 5. EigenFace expression classification; a- Samples of training expressions, b- Some generated EigenExpressions.
only the point cloud contents that fall on the previously calibrated bounding box of the face obtained from the face detection stages. Then, we perform a depth filtering assuming that any data that is further than a threshold is part of the background. Figure 7-b shows the obtained point clouds relative to the faces. The obtained 3D data can also be displayed instead of the avatar.

2) Morphing of 3D avatar model:
Classic 3D facial animation techniques are based on the generation of a multitude of identical 3D faces representing the different key shapes of the face parts for each expression. In order to perform the transition from a given state to another, a morphing function is applied as the sum of the different face poses multiplied by a coefficient $a_i$, where:

$$\sum_{i=0}^{n} a_i = 1 \quad (7)$$

and $n$ is the number of facial poses. Using our obtained decision about the current facial expression, we are able to perform a morphing between the already prepared face avatar models showing the seven possible expressions for the same avatar. Figure 8 shows the different 3D avatar models used for each expression. We created and animated 2 basic models in the "obj" format.

III. EXPERIMENTATION AND RESULTS

A. KiFaEx dataset description
Part of this dataset creation work is motivated by the weak recognition results obtained from already trained classifiers as they have been created using different acquisition conditions. Our Kinect Facial Expression dataset (KiFaEx) of images and point clouds contains information relative to 20 different subjects showing each one 7 different expressions (Fig. 9 shows examples among them). Two different shots were obtained in average for each different emotion, but, not all the data was used for expression fidelity reasons. The selected figures were divided into 132 expressions used during the learning process and 65 other ones used for the test. All data acquisition has been taken in a fixed position under the same lighting conditions in order to get the best performance rates from Eigenfaces recognition.

B. Expression recognition and reconstruction evaluation
Table II shows an example based evaluation for 65 different expressions retained for the test stage. The obtained results are also improved by combining the data of our dataset with those of the JAFFE dataset [16] as shown in Tab. III. Average accuracy percentage is 64% and it reaches 81% for the smiling emotion. As the reconstruction takes only in consideration an input every 5 frames, the obtained 3D animation is satisfactory for our reconstruction purposes. The morphing transition using expression recognition swaps fluidly between the different avatar emotions. We also notice that many weak classifications are due to logical similarities in the expression formulation. For example, disgust is almost similar to sadness in many cases. Fear and the neutral state can be easily confused due to the approximative similarities in the mouth and eyes dispositions. For this reason, the obtained 3D animation is visually acceptable by the performing user. Figure 10 shows examples of well recognized expressions in addition to their corresponding reconstructions. The Face’s point clouds have not been used during the description process and are only displayed optionally in the developed software. Future works will focus more on their exploitation.

![Fig. 7. a- Raw point cloud data obtained. Visualization is realized using the PCL library functions. The different colors indicate the proximity of each group of points from the source IR projector; b- Results obtained from our facial filtering distance based stage: a point cloud containing the face expression depth information is obtained.](image)

**TABLE II. OUR DATASET RECOGNITION CONFUSION MATRIX**

<table>
<thead>
<tr>
<th>KiFaEx &amp; JAFFE</th>
<th>neu</th>
<th>sur</th>
<th>hap</th>
<th>dis</th>
<th>fea</th>
<th>ang</th>
<th>sad</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>neu</td>
<td>17</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>1</td>
<td>-</td>
<td>3 81</td>
</tr>
<tr>
<td>sur</td>
<td>15</td>
<td>1</td>
<td>-</td>
<td>-</td>
<td>1</td>
<td>-</td>
<td>-</td>
<td>4 67</td>
</tr>
<tr>
<td>hap</td>
<td>3</td>
<td>8</td>
<td>1</td>
<td>1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>50</td>
</tr>
<tr>
<td>dis</td>
<td>3</td>
<td>-</td>
<td>14</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>67</td>
</tr>
<tr>
<td>fea</td>
<td>5</td>
<td>-</td>
<td>2</td>
<td>11</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>48</td>
</tr>
<tr>
<td>ang</td>
<td>-</td>
<td>-</td>
<td>1</td>
<td>-</td>
<td>9</td>
<td>-</td>
<td>-</td>
<td>3 69</td>
</tr>
<tr>
<td>sad</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>-</td>
<td>12</td>
<td>63</td>
<td></td>
</tr>
</tbody>
</table>

![Image](image)
IV. CONCLUSION AND PERSPECTIVES

We presented in this paper a solution for the automatic recognition and reconstruction of facial expressions using the 2D RGB and 3D depth data obtained from Microsoft’s Kinect. The developed interface offers the advantage of unsupervised expression reconstruction from the sensor and shows near to the state of the art performances. The results are printed in real time on a 3D viewport (see Fig. 10) showing an avatar face that imitates the automatically recognized emotions.

Future works include using the Semantic Shape Context [17] (SSC) approaches in order to recognize the facial expression from the Depth data. As the point clouds were not used during the learning process, applying a spatio-temporal SSC can reveal offer the information describing the facial expressions revealed by the point clouds extracted in parallel with the RGB flows.

ACKNOWLEDGMENT

The dataset acquisition for this paper has been possible thanks to the help of the IEEE ENISo student branch.

Fig. 8. Different facial expressions prepared for the 3D morphing stage. Each Model is prepared manually by an artist and the blendshapes take the charge of creating the morphing transition between the different emotions. The choice of the avatar is up to the user.

Fig. 9. Snapshots of the expressions saved for the 9 female and 11 male subjects in the KiFaEx dataset.

Fig. 10. Our JAVA based software is capable of (1) building an RGB-D dataset of different expressions for each person, (2) showing the results obtained for each data flow after the calibration, filtering and thresholding steps, (3) showing the 3D animated model corresponding to the captured expression and (4) visualizing the corresponding point clouds.

Fig. 10. Our JAVA based software is capable of (1) building an RGB-D dataset of different expressions for each person, (2) showing the results obtained for each data flow after the calibration, filtering and thresholding steps, (3) showing the 3D animated model corresponding to the captured expression and (4) visualizing the corresponding point clouds.
REFERENCES


