EAI Endorsed Transactions
on Creative Technologies
Research Article ICST.ORG

Head pose estimation & TV Context: current technology

Francois Rocca[1*], Matei Mancas[1], Fabien Grisard[1], Julien Leroy[1], Thierry Ravet[1] and Bernard Gosselin[1]

[1]University of Mons (UMONS), Faculty of Engineering (FPMs), 20, Place du Parc, 7000 Mons, Belgium

Abstract

With the arrival of low-cost high quality cameras, implicit user behaviour tracking is easier and it becomes very interesting for viewer modelling and content personalization in a TV context. In this paper, we present a comparison between three common algorithms of automatic head direction extraction for a person watching TV in a realistic context. Those algorithms compute the different rotation angles of the head (pitch, roll, yaw) in a non-invasive and continuous way based on 2D and/or 3D features acquired with low cost cameras. These results are compared with a reference based on the Qualisys motion capture commercial system which is a robust marker-based tracking system. The performances of the different algorithms are compared function of different configurations.
While our results show that full implicit behaviour tracking in real-life TV setups is still a challenge, with the arrival of next generation sensors (as the new Kinect one sensor), accurate TV personalization based on implicit behaviour is close to become a very interesting option.

Keywords: head pose estimation, viewer interest, face direction, Qualisys, Kinect, face tracking, 3D point cloud.

Received on 9 June 2014, accepted on 23 March 2015, published on 2 June 2015

doi: 10.4108/ct.2.3.e2

1. Introduction

The analysis of people interest is crucial for number of applications such as advertising, museums and public spaces displays, gaming technologies, TV experience, etc. Here we focus on context-aware TV experiences which can highly benefit from knowledge about viewer interest.

Viewer interest can be extracted in various ways. In computer vision there are two families of methods: one is marker-based and the other markerless. Here we focus on markerless face direction (or head pose) estimation techniques which begin to provide results for real-world applications at reasonable distances and illumination conditions in a non-invasive and transparent way.

Moreover, more and more TV and home setups (such as the XBOX) come with cameras. The acceptance of such sensors inside homes watching the viewers is higher and higher and people see less ethical issues in being observed by sensors if they have enhanced experience in return and if the data is processed locally in real-time without any recording and audio/visual data transmission. The conjunction of the arrival of new efficient low-cost sensors and of their high degree of acceptance open new potential applications in the TV domain. In this paper we thus focus on TV and present a state for the art of the main face direction estimation methods which can be used in TV setups. In addition to the gesture/voice recognition which can be now found on a lot of "smart" TVs, those setups can help enhancing viewer TV experience by modelling viewers behaviour to provide them with personalized media or enrichments in a single or multi-screen environment.

In section 2, we present the different techniques for head pose estimation and we describe markerless face direction methods which are used in this study. In section 3, we present the Qualisys system which was used to validate these techniques along with the experimental validation setup. Section 4 shows the validation results in our setup and presents a discussion on the state-of-the-art methods which allows to choose a method depending on the viewing situations (in terms of illumination changes or distance from the sensor for example) or analysis results (in terms of precision and framerate constraints). We finally conclude

*Corresponding author. Email:francois.rocca@umons.ac.be

in section 5 on the usability of the current and near future methods.

2. Head pose estimation

Head pose estimation and head movements are mainly captured with physical sensors and optical analysis as we can see in the animation industry.

Physical sensors such as accelerometers, gyroscopes and magnetometers are placed on the head to compute the head rotation [1] [2].

Another way consists in marker-based optical motion capture systems that are able to capture the subtlety of the motion. In these methods, markers are placed on the head of the actor and they are tracked through multiple cameras. The markers are often coloured dots or infrared reflective fiducials and the cameras depend on the markers type. Accurate tracking requires multiple cameras and specific software to compute head pose estimation. These systems are very complex and expensive, they need calibration and precise positioning of markers (Optitrack [3], Qualisys [4]) and they remain invasive. While they cannot be used in real-life TV setups, we use marker-based methods to evaluate the markerless methods which can be low-cost, transparent (no calibration needed) and non-invasive. More precisely we use the Qualisys motion capture system.

Markerless tracking is another approach to face motion capture and a wide range of methods exists. Some markerless equipment use infrared cameras to compute tracking of characteristic points. For example, FaceLAB gives the head orientation and the position of lips, eyes and eyebrows [5]. But there are also algorithms using only a webcam. We can cite FaceAPI [6] from the same company as FaceLAB. Markerless systems use classical cameras or infrared cameras to compute tracking of characteristic points. We choose several freely accessible methods in this paper for a fair comparison in a real-world TV context.

The first method that we use is based on the Microsoft Kinect SDK [7]. The Kinect SDK is free, easy to use and contains multiple tools for user tracking and behaviour modelling such as face tracking and head pose estimation. These tools combine 2D and 3D information obtained with the Kinect sensor.

Secondly, a head pose estimation solution based on 2D face tracking algorithm using the free library OpenCV [8]. The face tracking part of this method was developed by Jason Saragih and it is known under the name of "FaceTracker" [9]. The head pose estimation part was developed separately and it is explained in a study of this method for computer uses [10].

Finally we use a fully 3D method for real time head pose estimation from depth images [11] based on a free library called PCL (Point Cloud Library) [12].

2.1. MS Kinect solution (KinectSDK)

The Kinect sensor developed for the Xbox360 is a low-cost depth and RGB camera. It contains two CMOS sensors, one for the RGB image (640 x 480 pixels at 30 fps) and another for the infrared image from which the depth map is calculated, based on the deformation of an infrared projected pattern (λ = 830nm). The depth sensor has an optimal utilisation in a range of 1.2 meter (precision better than 10 mm) to 3.5 m (precision better than 30 mm) [13] and can be perturbed by other sources of infrared light.

Microsoft provides a Face Tracking module with the SDK which works with the Kinect SDK since the version 1.5. These SDKs can be used together to "*create applications that can track human faces in real time*" To achieve face tracking, at least the upper part of the user's Kinect skeleton has to be tracked in order to identify the position of the head.

The Get3DPose method returns two tables of three float numbers. The first one contains the Euler rotation angles in degrees for the pitch, roll and yaw as described in Figure 1, and the second contains the head position in meters. All the values are calculated relatively to the sensor which is the origin for the coordinates.

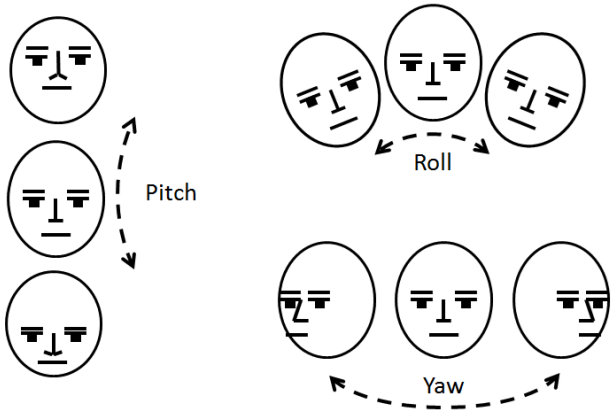

Figure 1. Three different degrees of freedom: pitch, roll and yaw [14]. All head motion can be obtained by combining these three basic movements.

The technique used to estimate the rotations and facial features tracking of the head (Figure 2) is not described by Microsoft, but the method uses the RGB image and depth map. The head position is located using 3D skeleton only on the depth map. The head pose estimation itself is mainly achieved on the RGB images. Consequently, the face tracking hardly works in bad light conditions (shadow, too much contrast, etc.).

By using the SDK, we obtain a head orientation measuring tool at 30 fps (frames per second). The experiment computer is a laptop with an Intel i7 2.40GHz, 8GB of RAM and running Windows 8. For this paper, this method will be called "KinectSDK".

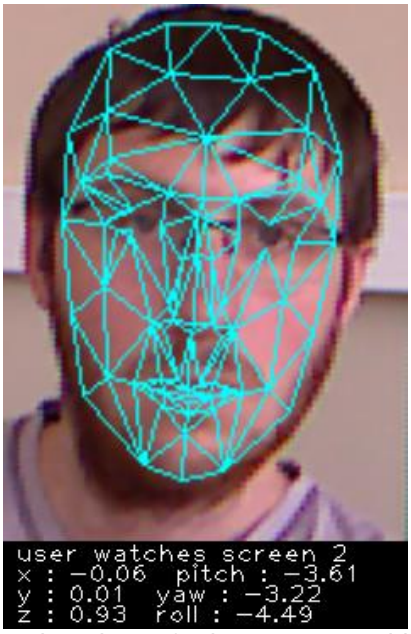

Figure 2. The Microsoft Kinect SDK provides facial features tracking and head pose estimation thought pitch yaw and roll.

2.2. Webcam solution (Facetracker)

This method is a combination of FaceTracker and a head pose estimation based of the features extraction from the face tracking part.

FaceTracker allows the identification and localization landmarks on a RGB image. These points can be assimilated to a facial mask allowing to track facial features like the edge of lips, facial contours, nose, eyes and eyebrows (Figure 3). Based on this, we apply the perspective-n-point (PNP) methods [11] to find the rotation matrix and 3D head pose estimation.

FaceTracker is a CLM-based C/C++ API for real-time generic non-rigid face alignment and tracking. The approach is an instance of the constrained local model with the subspace constrained mean-shifts algorithm as an optimization strategy [10].

The advantage is that FaceTracker does not require specific manipulation before the utilization and the algorithm makes an automatic detection of the user face based on a pre-trained model on database. FaceTracker is based on the OpenCV library [9]. It is compatible with any camera. In our setup we use a 480X640 pixel webcam. The initialization of the algorithm is based on the Haar classifiers [15], thus the face tracking is optimal if the face is centred in front of the camera and straight. We can also observe significant perturbations when an object starts occluding some landmarks or when head rotation is rapidly done with a wide angle.

To find the Euler angles of the rotation of the head we use 2D Points from Facetracker, 3D points from a 3D head model and we compute the rotation matrix based on the perspective-n-point method.

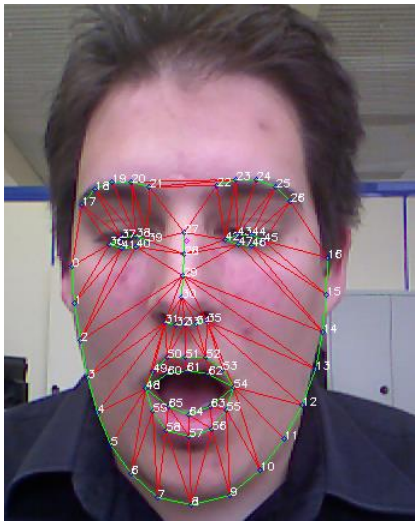

Figure 3. FaceTracker detects in real-time a set of 66 points. Points 0 to 16: lower facial contours, 17 to 21 and 22 to 26: right and left eyebrows, 27 to 35: nose, 36 to 41 and 42 to 47: right and left eyes, 48 to 65: edge of lips.

A Set of 7 points are taken among the 66 points from FaceTracker. These points were chosen because they are far enough and stable regardless of the expressions and movements of the face. In parallel to this, we use a 3D head model on which we extract 3D points corresponding to 2D previous points.

Once the seven 2D and 3D coordinates are set, and the camera matrix found, we can calculate the matrix of rotation and translation of 3D model by reporting the data from the face tracking (Figure 4). The pitch, roll and yaw can directly be extracted from the rotation matrix in real time about 24fps (from 19 to 28fps). The computing time per frame is about 50ms by single thread on a Linux OS with Intel Core i7 2.3GHz and 8GB of RAM. For the next steps of this analysis, this method is named "Facetracker".

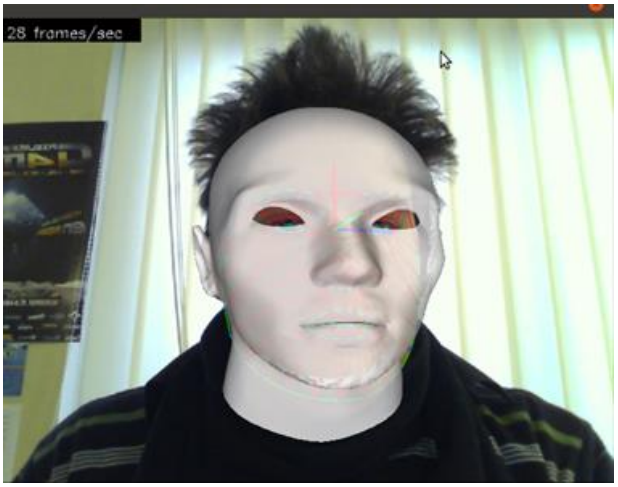

Figure 4. We have the projection of the 3D head model correctly superposed on the points from the face tracking.

2.3. Use of 3D point clouds (3DCloud)

The method used here is based on the approach developed in [16][17]. The implementation used in this case [18] is the version based on the PCL library. The main differences between the Fanelli method and the PCL implementation are the parameters of the algorithm and the training. The PCL implementation was online earlier, it is further maintained and was used for head pose estimation in TV context [11]. This solution relies on the use of random forest regression applied on a 3D cloud. This cloud is obtained with a RGB-D camera, such as MS Kinect or Asus Xtion. Random forests [19] are capable of handling large training sets, of generalization and fast computing time. In our case the random forests are extended by using a regression to simultaneously detect faces but also to estimate their orientations on the depth map.

The method consists of a training stage during which we build the random forest and an on-line detection stage where the patches extracted from the current frame are classified using the trained forests. The training process is done once and it is not requested for any user. The training stage is based on the BIWI dataset [20] containing over 15000 images of 20 people (6 females and 14 males). This dataset covers a large set of head pose (+-75 degrees yaw and +-60 degrees pitch) and generalizes the detection step. A leaf of the trees composing the forest stores the ratio of face patches that arrived to it during training as well as two multi-variate Gaussian distributions voting for the location and orientation of the head. A second processing step consists in registering a generic face cloud over the region corresponding to the estimated position of the head. This refinement can greatly increase the accuracy of the head tracker but requires more computing resources. A real-time mode is possible to use but it works at around 1 fps that is why we decided to run the system off-line (Figure 5). This allows a full processing of data corresponding to a recording of 20 fps with the refinement step.

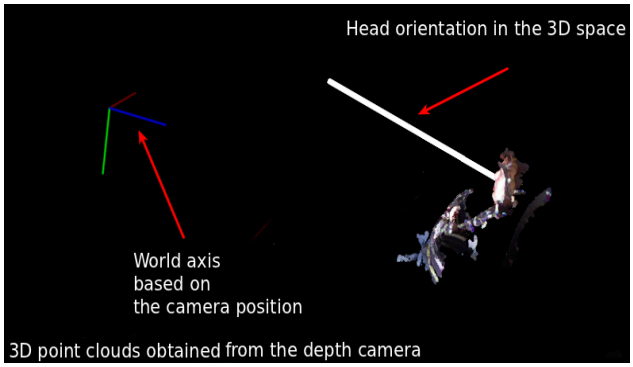

Figure 5. 3D rendering of the system. We can observe the 3D point cloud obtained with the depth camera and the application of the head pose estimation algorithm. When a face is detected, we retrieve a vector of the head direction.

The advantage of such a system is that it uses only geometric information from the 3D point cloud extracted by a RGB-D sensor, and is independent of the brightness. It can operate in the dark, which is rarely possible with face tracking systems working on colour image which are highly dependent on the illumination. This approach was chosen because it fits well in the scenario of TV interaction [11]. In addition, the use of 3D data will simplify the integration of future contextual information about the scene. For the analysis, this method is named "3DCloud".

3. A comparison: experimental setup

In this section we will first describe how we obtained the reference values with the Qualisys system.

3.1. Qualisys setup

System description

Every result of the experiments presented in this study was compared with an accurate measurement of the head movements. This ground truth was obtained thanks to an optical motion capture system from Qualisys [4]. The setup consists of eight cameras, which emit infrared light and which track the position of reflective markers placed on the head. Qualisys Track Manager Software (QTM) provides the possibility to define a rigid body and to characterize the movement of this body with six degrees of freedom (6DOF: three Cartesian coordinates for its position and three Euler angles - roll, pitch and yaw - for its orientation).

We used seven passive markers: Four markers were positioned on the TV screen and three markers were fixed to a rigid part of a hat (the three markers were placed with a distance of 72mm, 77mm and 86mm between them) (Figure 6 and Figure 7). Both TV screen and hat were defined as rigid bodies in QTM. The framerate tracking is constant at 150 fps, so it gives the values of the 6 degrees of freedom (DOF) each 0.007 seconds.

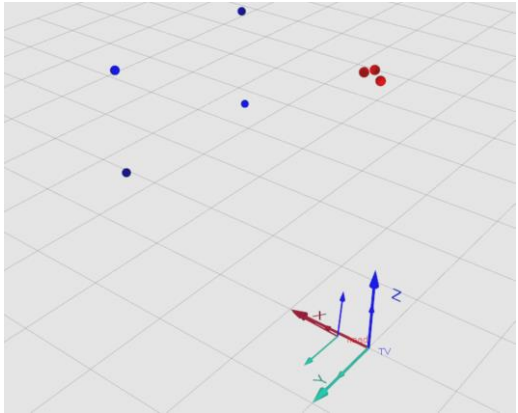

Figure 6. Qualisys Track Manager displays tracking of two rigid bodies (TV screen in blue on the left and head in red on the right).

Figure 7. Infrared reflectors on viewer hat sitting in front of the TV.

System calibration

Before each recording session, a calibration procedure was made: the subject, who wears the hat, sat in front of the screen and QTM nullified the 6DOF values for this head position. By this means, all the head movements were measured relatively to this initial starting position. To check the quality of the tracking data, QTM computes the different residuals of the 3D points compared to the rigid body definition. Over all the experiments, the average error of each head marker about 0.62mm.

3.2. Experimental setup

Qualisys produces marker-based accurate data in real-time for object tracking at about 150 frames per second. The infrared light and marker do not interfere with RGB image and with infrared pattern from the Kinect. The choice of Qualisys as reference has been done especially in order to compare markerless methods without interferences.

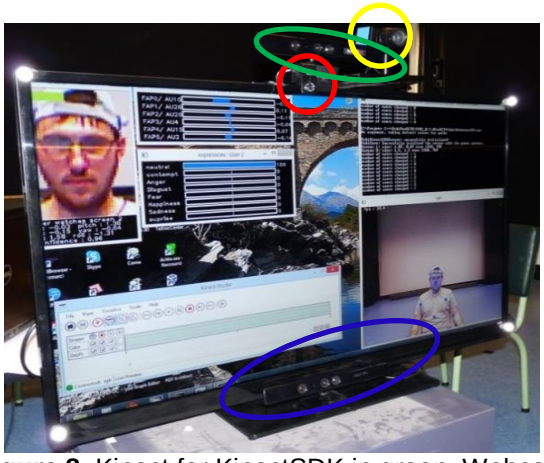

Figure 8. Kinect for KinectSDK in green, Webcam for the facetracker in red, Kinect for 3DCloud in blue, 2D camera synchronized with Qualisys in yellow.

We perform the recording of the KinectSDK and the Facetracker during the same time under normal conditions and correct face lighting. And we have chosen to perform the 3DCloud method separately from the first record because interferences are observed between 2 running Xbox360 Kinects heading in the same direction. This positioning is shown on Figure 8. The angles computed from the different methods are the Euler angles.

We made several recordings with 10 candidates. Each one does head movement sequence at 5 different distances from the screen: 1.20m, 1.50m, 2m, 2.5m and 3m. Movements performed are conventional rotations when we are facing a screen (pitch, roll, and yaw; combination of these movements; slow and fast rotation).

Six of them had light skin, others have dark skin. Three of them wear glasses and six of them wear beard or mustache. Table 1 summarizes these facial characteristics.

Table 1. Facial characteristics for the 10 candidates.

Candidates	1	2	3	4	5	6	7	8	9	10
Glasses	X	X		X						
Light skin	X	X	X	X		X				X
Beard	X				X	X		X	X	X

A preliminary test showed that the optimal position of the camera for Facetracker and KinectSDK is on top of the screen, while for 3DCloud which uses the shape of the jaw, is at the bottom. We thus decided to keep this two different positions in the following tests to maximize each method results. This does not change anything to the distances or viewing conditions and both positions could be valid in a real-life setup. However, we can notice that the top position would be more practical to avoid obstacles in people room such as objects on a table, etc.

4. Experiments results

After having synchronized the results obtained by all systems and the reference (temporal alignment and start offset suppression), as the sampling frequencies are different, we have interpolated the reference values to obtain similar data sampling for the different systems that we compare. To make the comparison between systems and the reference computed with Qualisys, we use two metrics: the Root Mean Square Error (RMSE) and the correlation score.

The Root Mean Square Error is given by:

$$\sqrt{\sum \frac{(y_{pred} - y_{ref})^2}{N}} \qquad (1)$$

With the predicted values obtained by one system y_{pred}, the values from the reference y_{ref} and the total number of values N.

The correlation, based on the Pearson's coefficient, is given by:

$$\frac{\Sigma(y_{ref} - \bar{y}_{ref})(y_{pred} - \bar{y}_{pred})}{\sqrt{\Sigma(y_{ref} - \bar{y}_{ref})^2}\sqrt{\Sigma(y_{pred} - \bar{y}_{pred})^2}} \quad (2)$$

4.1. Raw data visualization

The Figures below show the results with the superposition of values from the different algorithms with the reference for one random recording session. The first series of three graphics show the KinectSDK, the Facetracker and the reference for pitch, yaw and roll (Figure 9). The second series show the 3DCloud method compared with the reference (Figure 10).

Each recording session contains a head movement sequence at 5 different distances. Figure 11 shows part of a session for the pitch, roll and yaw at the distance of 1m20 for KinectSDK. The sequence is: first a yaw movement follows by a pitch and a roll movement. Next movements are combination of previous basic movements.

The holes in the green plots on the Figure 11 come from loss of tracking by the KinectSDK. The algorithm provides no point when tracking is lost. The 3Dcloud method does not provide point in case of loss of tracking as the KinectSDK, but the Facetracker based method gives results even in case of loss of tracking, based on the latest detection (Figure 12).

We observe tracking losses in large and rapid angular movements. This is often due to the fact that part of the head is less visible or the brightness is reduced, and therefore the tracking based on features points from RGB image is more difficult to do. Figure 13 below shows angular errors with KinectSDK and the Facetracker

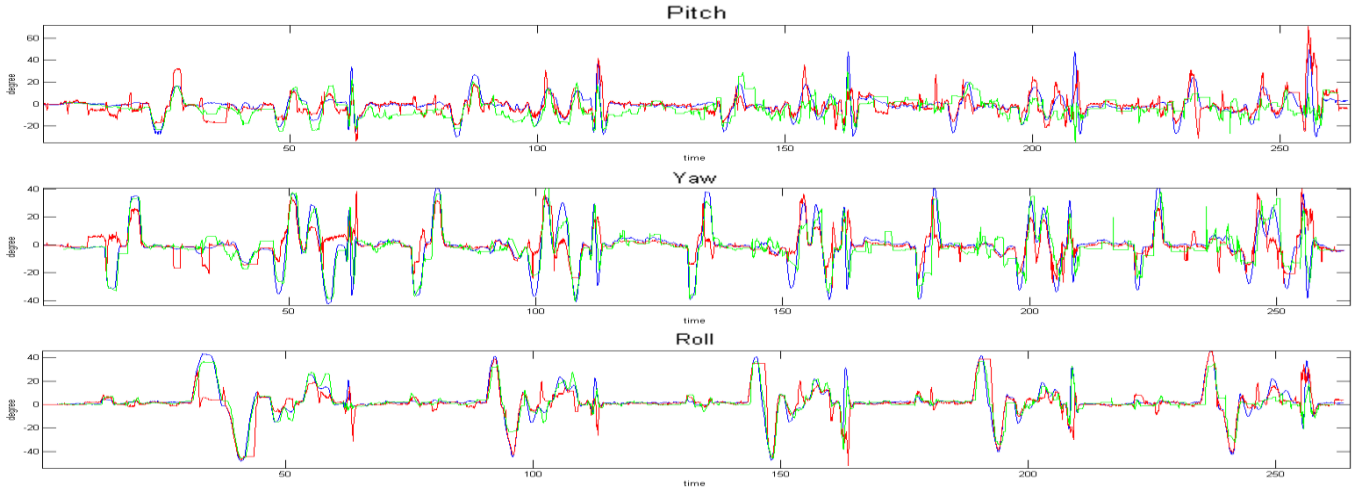

Figure 9. Reference: blue, KinectSDK: green, Facetracker: red. First row: pitch, second row: yaw, third row: roll.

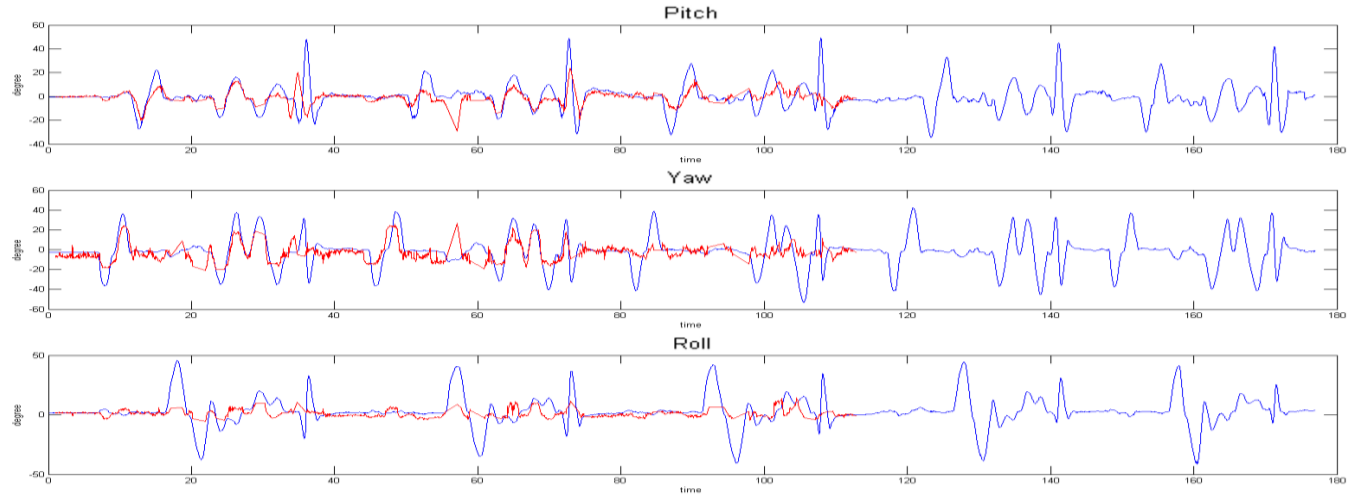

Figure 10. Reference: blue, 3DCloud: red. First row: pitch, second row: yaw, third row: roll. Tracking is lost for a distance greater than 2 meters for 3DCloud.

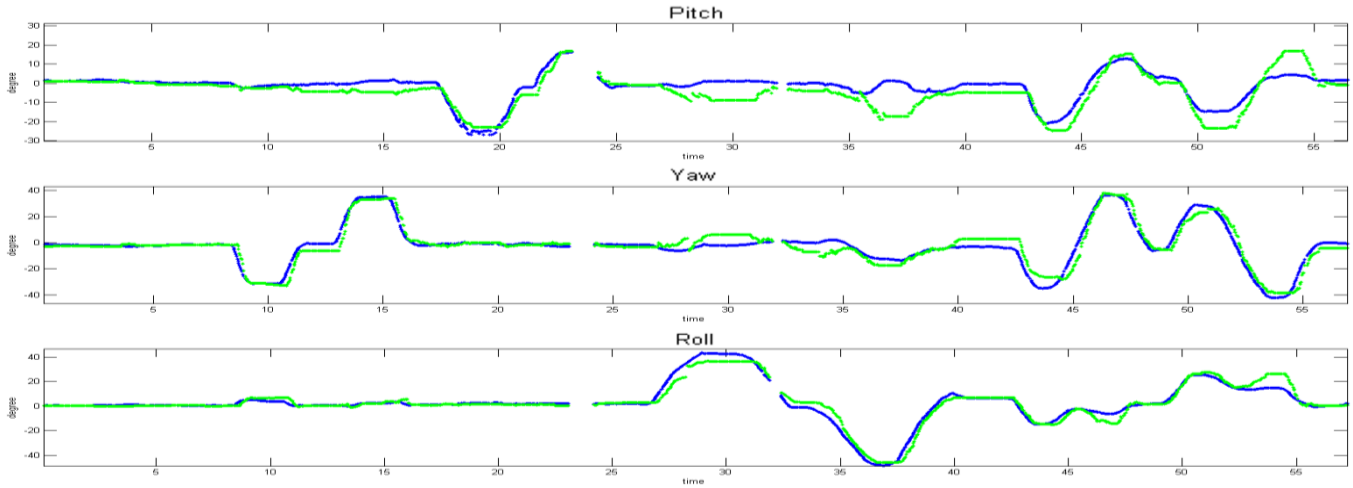

Figure 11. Head movement sequence for a distance of 1m20. Errors appear like holes in the green plots. Reference is in blue and KinectSDK is green.

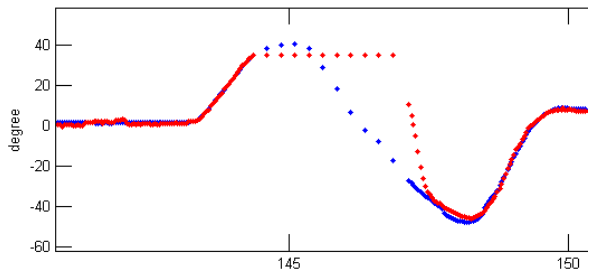

Figure 12. Results given by Facetracker (red) in case of loss of tracking.

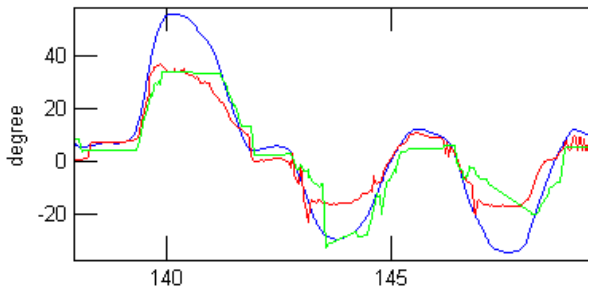

Figure 13. Errors are observed with some angular movement due to loss of tracking. The reference is blue, KinectSDK is green and Facetracker is red.

In addition to errors on large angles, we observed that 3DCloud achieves significant errors in the roll movement (Figure 14). This is caused by the shape of 3D head model used in this method. The model is mainly flat with nose prominent and it is good for pitch and yaw but less good for the roll rotation.

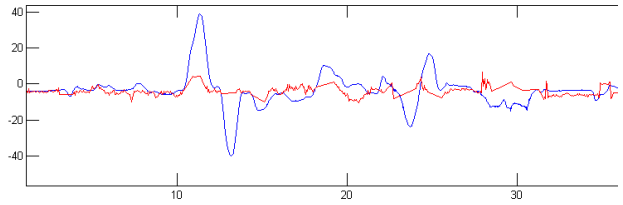

Figure 14. Errors observed on Roll: 3DCloud in red, reference in blue.

4.2. Head pose estimation: a comparison

Based on the raw data we compute the Root Mean Square Error (RMSE) and the correlation of the three methods compared to the Qualisys reference. The results are summarized in Figures 15 to 20.

Correlation and RMSE function of the distance

The correlation is a good indicator used to establish the link between a set of given values and its reference. It is interesting to analyse the correlation value obtained for each distance, with average for all candidates, to know which methods are better correlated with the reference data. If the correlation value is equal to 1, the two signals are totally correlated. If the correlation is between 0.5 and 1, we consider a strong dependence. The 0 value shows that the two signals are independent and de -1 value correspond to the opposite of the signal. Figure 15 shows the correlation for pitch, Figure 16 for yaw and Figure 17 for roll. The three plots from KinectSDK, Facetracker and 3DCloud are compared with the Qualisys reference.

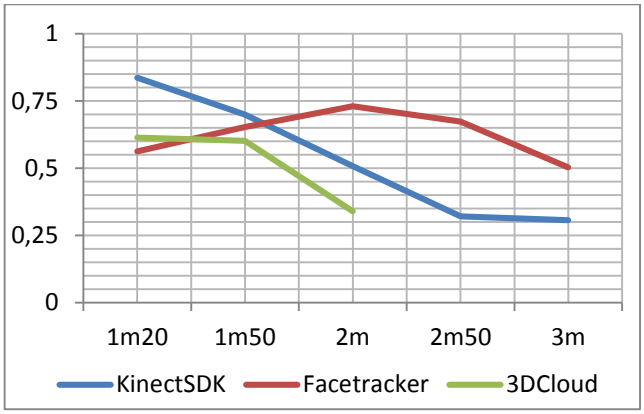

Figure 15. Mean correlation for the pitch function of the viewer distance from TV (in m).

On Figure 15, we observe that the pitch (up-down movement) of the KinectSDK has a good correlation (0.84) at a distance of 1m20. The Facetracker and 3DCloud are lower with values about 0.6. We observe that the facetracker stays stable with the distance between 0.5 to 0.73. But KinectSDK and 3Dcloud decrease with the distance under the correlation value of 0.5 for KinectSDK at 2m50 with 0.32, and for the 3DCloud at 2m with 0.34.

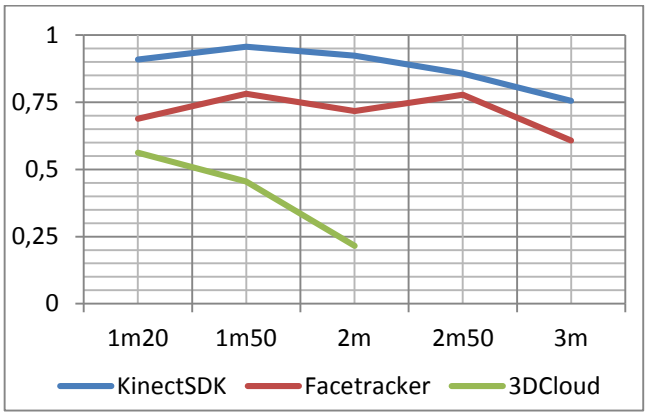

Figure 16. Mean correlation for the yaw function of the viewer distance from TV (in m).

For the second angle, the yaw, corresponding to a right-left movement, we have on the Figure 16 good results for the KinectSDK with values upper than 0.9 for 1m20, 1m50 and 2m. Then de values decrease from 0.85 for 2m50 to 0.76 for 3m. The plot of the Facetracker is similar but less good with values around 0.75. 3DCloud achieves the worse performance with 0.61 at the beginning and less after.

As mentioned in Section 4.1, the 3DCloud provides bad values for the roll. The KinectSDK have good correlation as for the yaw curve (0.93 to 0.7). Facetracker correlation is also good but with lower result than KinectSDK with about 0.65 (Figure 17).

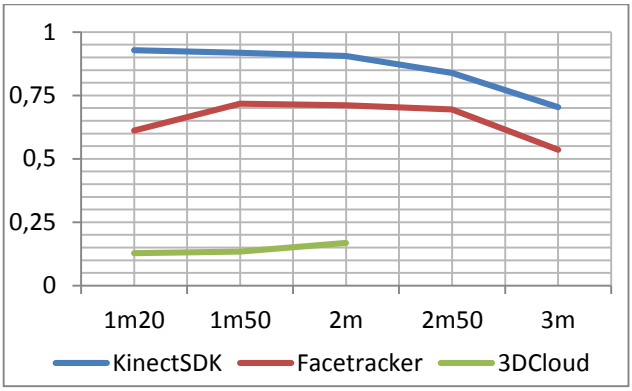

Figure 17. Mean correlation for the roll function of the viewer distance from TV (in m).

After watching the correlation values, it is also interesting to look at the mean error made by each system. Indeed, a method with a big correlation and low RMSE is considered very well for head pose estimation. Figure 18 shows the RMSE for pitch, Figure 19 for yaw and Figure 20 for roll.

We observe a RMSE similar for the pitch about 10 to 15 degrees for each method (Figure 18). But the KinectSDK is good at 1m20 with 5.9 degrees. The error logically grows with the ditance.

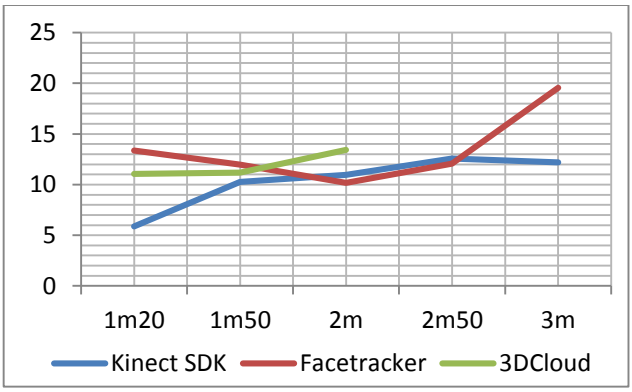

Figure 18. Mean RMSE (in degrees) for the pitch function of the viewer distance from TV (in m).

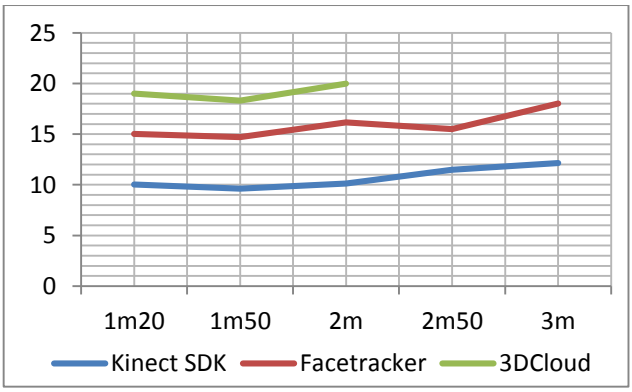

Figure 19. Mean RMSE (in degrees) for the yaw function of the viewer distance from TV (in m).

On the yaw, we observe on Figure 19 a slight increase of the error with the distance. But the KinectSDK is better with RMSE from 10 to 12 degrees, 15 to 18 degrees for Facetracker and around 20 for 3DCloud.

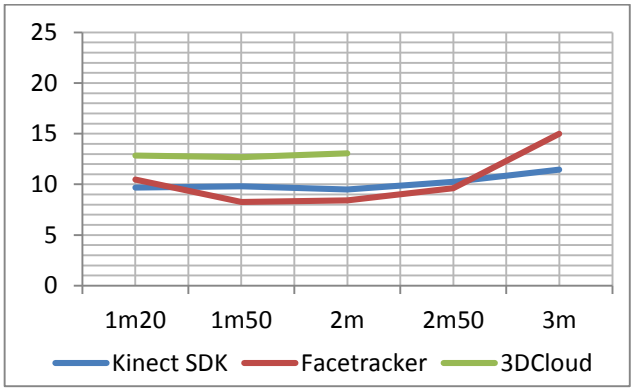

Figure 20. Mean RMSE (in degrees) for the roll function of the viewer distance from TV (in m).

In the case of roll, the RMSE is similar for Facetracker and KinectSDK (around 10 degrees with a smaller error at 3m for KinectSDK). The error of 3DCloud is arround 13 degrees (Figure 20). This error can be put in perspective because the correlation for the roll was poor.

Correlation and RMSE function of the viewer

After watching the values of the root means square error and correlation according to the different distances, it is interesting to look at the average values of these two indicators for each individual to link some observation to candidates facial features previously described in Table 1. Below, we have all three graphs (Figures 21, 22 and 23) for the test according to the correlation, followed by the three graphs of the RMSE (Figures 24, 25 and 26).

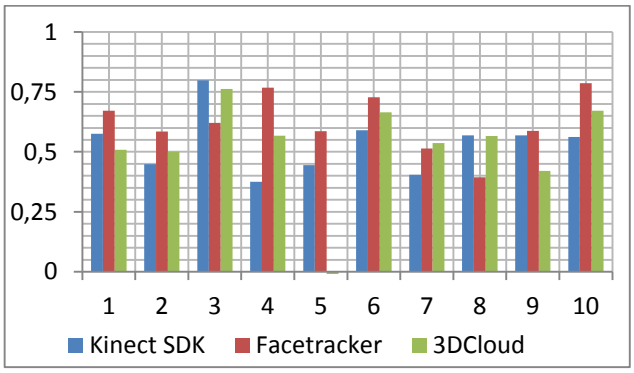

Figure 21. Mean correlation for the pitch for each candidate.

In Figure 21, we observe that the correlation for each individual is about 0.6. All these values are similar. But a correlation about 0 is observed for the candidate number 5 for the 3DCloud method which means that the pitch did not work at all.

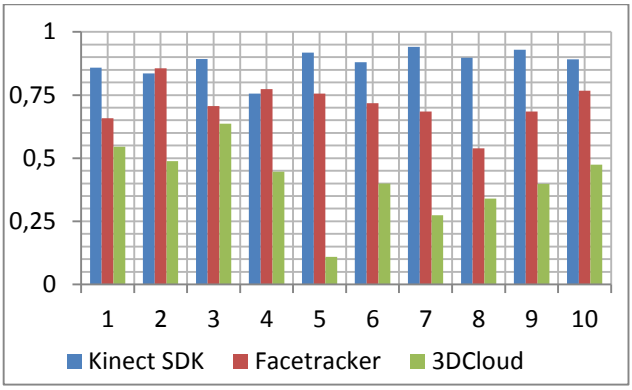

Figure 22. Mean correlation for the yaw for each candidate.

In Figure 22, the KinectSDK gives a coorelation higher than 0.75 for each candidate followed by Facetracker with values higher than 0.5. 3DCloud method gives the worse correlation with values between 0.1 and 0.64. For this method, candidate number 5 gives also the worse correlation for the 3DCloud.

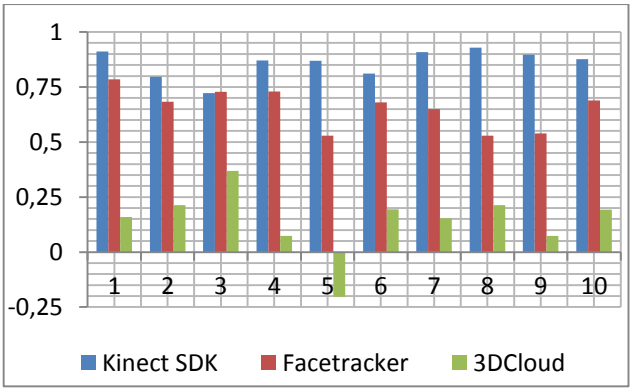

Figure 23. Mean correlation for the pitch for each candidate.

In Figure 23, we observe that the KinectSDK and the Factracker method give good values higher than 0.5, with a better coorelation for KinectSDK. Results for the 3DCloud are worse, as already seen on other graphics regarding the roll (Figure 17).

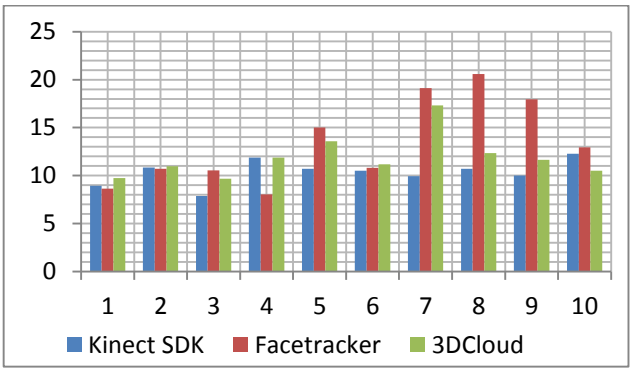

Figure 24. Mean RMSE (in degrees) for the pitch for each candidate.

RMSE is about 10 for the KinectSDK and 3DCloud for all candidates in Figure 24. We observe that the error on the pitch for the Facetracker method is higher for candidates 5,7,8 and 9, these candidates have darker skin (Table 1). The KinectSDK has more homogenous results.

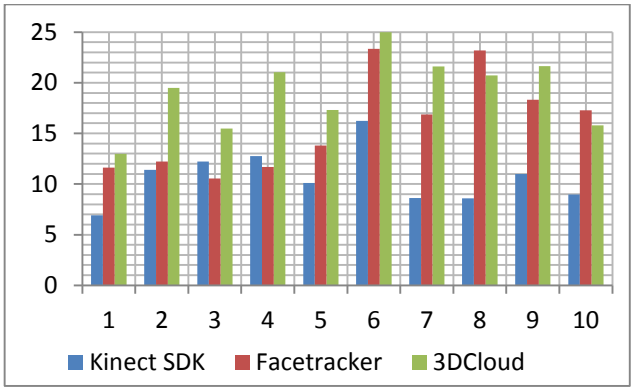

Figure 25. Mean RMSE (in degrees) for the yaw for each candidate.

In Figure 25, the 3DCloud gives worse results than KinectSDK and Facetracker. We also observe bigger error for darker skin for the Facetracker method. Again KinectSDk seems to be less sensitive to the viewer skin color.

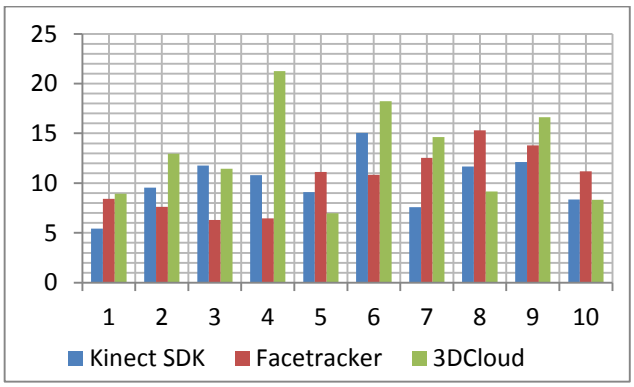

Figure 26. Mean RMSE (in degrees) for the roll for each candidate.

On this roll graph (Figure 26), the error is about 10 degrees for KinectSDK and Facetracker and greater for the 3DCloud method.

4.3. Face direction methods analysis

After analyzing all data obtained by the three different methods we are able to establish the advantage and the drawbacks for each method in a TV context.

These results show that the better correlation values are obtained with the KinectSDK. The Facetracker based method also gives good result. We also have similar errors for these methods. A previous study has shown that the Facetracker method gives very good result for a distance under 1 meter [10]. At this distance the KinectSDK is not able to track the head because on one hand the sensor had a blind zone up to 60cm [21] and on the other hand the field of view is too small and it is hard to detect correctly a user under a distance of 1 meter. Concerning the third method, 3DCloud, the RMSE and the correlation are worse than the two other methods and do not work at a distance of more than 2m from the screen. The estimation of roll is also of poor quality. Concerning skin color, KinectSDK seems to be the most homogenous methods while the two others (mostly FaceTracker) might work less well in case of dark skin.

For all these methods, errors are mainly due to face tracking errors and tracking losses. If we cut all sections with bad detection of the head and the characteristics point of the face, the RMSE will decline significantly and the correlation will increase. But in our context, we want to get results without post-processing corrections. We can also say that from a distance of 1.50m, an error of 10 degrees generates a gaze tracking error on the screen of 26cm ($150\sin(26°)$). This is quite acceptable for whether a person looks at a screen, or any other object. However, this error let us hope to be able to detect the screen which is attended and not precisely what region of the screen is attended.

About the benefits of these different methods, we can say that the Facetracker method requires a basic camera while the two other work with a 3D sensor. The advantage of the 3D sensor for the KinectSDK is in the robust people and head tracking. Thanks to this, KinectSDK rarely loses the head position, provided being able to detect and track the user skeleton. The 3DCloud method allows head pose estimation in all kind of illumination and also in darkness because it works only on the point cloud obtained by the 3D sensor. Facetracker and KinectSDK work in real-time while the 3DCloud requires about 1 second per frame.

The pitch and the yaw are the two important rotations in a context on TV watching because we are generally straight face at the TV, so roll is generally close to 0. In this case the Pitch describes the up-down movement. This movement is important to know if the viewer looks at the main TV screen or if he watches a second screen on his knees like a smartphone or a tablet. The yaw corresponds to a left-right movement, usable to know if the viewer watches the main TV screen or if his attention is drawn on the sides of the screen, for example to talk with somebody else. Combination of pitch and yaw indicate the direction of the face allowing to know where the user look on the TV, but given the error of 26 cm at a distance of 1m50, by using the current technique one can hardly get usable screen position information and only the attended screen can be extracted in real TV setups.

Although our tests present the viewer properly sited on a classical chair, we produced some preliminary tests in much more relaxed position on a sofa and daylight in the back (Figure 27). The first results show that while 3DCloud and FaceTracker perform poorly, the KinectSDK performs less well, but still the data extracted using the facial mask makes sense.

Figure 27. Preliminary test in relaxed positions on a sofa with a tablet as second screen (KinectSDK).

5. Head pose estimation in a TV context: a conclusion

This study aims to show the advantages and weaknesses of three markerless head pose estimation methods in a TV context. This assessment is achieved using a highly accurate marker-based MOCAP system (Qualisys). These three methods were chosen because they are easy to use, low-cost and the codes are freely available.

The study of accuracy is made on individuals with different facial characteristic. As we work in the context of TV watching for user attention detection, we worked over distances from 1.20m to 3m and we analyzed the rotation of the head along three angles: pitch, roll and yaw.

This study focuses on Facetracker a method operating on RGB image, the 2D-3D method from the KinectSDK and a full 3D method based on Point Cloud Library (3DCloud). The results proved that the most accurate method is the KinectSDK with the best correlation and the smaller mean error. These accuracy is due to the 3D user and skeleton detection which provides precisely the head position. Based on this robust head position, the estimation of angles of rotation is made easier. The second best result is obtained by the Facetracker method. The error is a bit higher and correlation slightly lower than KinectSDK due to wrong face detection. These two methods have weaknesses in face illumination variations and occlusions. Concerning the full 3D method we observed the worse results. But this method has a major advantage because it works only on the point cloud and it is insensitive to brightness changes and also works in complete darkness. We also notice that the methods are sensitive to facial characteristic for head pose estimation. Glasses and beard create minor errors. Only the color of the skin has a slight effect with the method for face tracking which was less stable.

The choice of one of these methods is therefore based on the context of use. If the illumination is bad or if it must operate in the dark, the chosen method will be 3DCloud. This method however has the disadvantage that requires more computation time while the other two methods work perfectly in real time. In the case of a classical TV setup, the user attention is better computed by the KinectSDK. If we are interested in head pose estimation with straight face in front of a computer screen (like a webcalm computer setup) the KinectSDK is better if it is possible to track the user skeleton (not too close to the camera). Otherwise the FaceTracker will be the best method for computer uses.

Our tests show that the current technologies can provide a first prototype of implicit viewer behavior in the context of a TV setup. However, reaching good extraction quality in real-life setups with natural positions and lightning are only possible by using a robust sensor as a RGB-D camera. Nevertheless, with the arrival of second generation RGB-D sensors as the Kinect one (second version of the Kinect sensor which provides better depth sensor, better RGB definition and operates in more complex illumination conditions), the implicit viewer behavior acquisition in real-life TV setups becomes possible.

The head pose estimation allows to know the user interest (or disinterest) on the media displayed on the screen which is of crucial importance in TV content personalization. In addition to one viewer head pose estimation, other features as body movements, postures or joint attention can be extracted from the skeleton to provide additional features to the TV viewer behavior analysis. Joint attention appears when two individual share the focus on the same object, in this case the object is the screen.

Acknowledgements

This work is supported by the Integrated Project LinkedTV (www.linkedtv.eu) funded by the European Commission through the 7th Framework Programme (FP7-287911).

References

[1] Persa, S.-F. (2006) Sensor fusion in head pose tracking.
[2] Emotiv EPOC headset Features, http://emotiv.com/epoc/features.php
[3] OptiTrack, Optical motion tracking solutions. http://www.naturalpoint.com/optitrack/
[4] Qualisys. Products and services based on optical motion capture. http://www.qualisys.com/
[5] FaceLAB 5. Face and eye tracking application. http://www.seeingmachines.com/product/facelab
[6] FaceAPI. Markerless face tracking application. http://www.seeingmachines.com/product/faceapi/
[7] Microsoft Kinect Software Development Kit, http://www.microsoft.com/en-us/kinectforwindowsdev/Start.aspx
[8] Bradski, G. (2000) The opencv library. *Doctor Dobbs Journal of Software Tools 25.11* 120-126.

[9] Saragih, J. M., Lucey, S., & Cohn, J. F. (2011) Deformable model fitting by regularized landmark mean-shift. *International Journal of Computer Vision, 91(2)*, 200-215.
[10] Rocca, F., Mancas, M., & Gosselin, B. (2014). Head Pose Estimation by Perspective-n-Point Solution Based on 2D Markerless Face Tracking. *Intelligent Technologies for Interactive Entertainment: 6th International ICST Conference, INTETAIN 2014*
[11] Leroy, J., Rocca, F., Mancas, M., & Gosselin, B. (2013) Second Screen Interaction: an Approach to Infer TV Watcher's Interest Using 3D Head Pose Estimation. *WWW13 Companion: Proceedings of the 22nd international conference on World Wide Web companion* 465-468
[12] Rusu, R. B., & Cousins, S. (2011, May). 3d is here: Point cloud library (pcl). *In Robotics and Automation (ICRA), 2011 IEEE International Conference, IEEE*, 1-4
[13] Gonzalez-Jorge, H., Riveiro, B., Vazquez-Fernandez, E., Martínez-Sánchez, J., & Arias, P. (2013). *Metrological evaluation of microsoft kinect and asus xtion sensors. Measurement, 46(6)*, 1800-1806.
[14] Face Tracking. Microsoft Developper Network. http://msdn.microsoft.com/en-us/library/jj130970.aspx
[15] Viola, P., & Jones, M. J. (2004) Robust real-time face detection. *International journal of computer vision, 57(2)*, 137-154.
[16] Fanelli, G., Dantone, M., Gall, J., Fossati, A., and Gool, L. (2013) Random forests for real time 3d face analysis. *International Journal of Computer Vision, 101:*437-458,
[17] Fanelli, G., Gall, J., and Van Gool, L. (2011) Real time head pose estimation with random regression forests. *CVPR*, 617-624
[18] Aldoma, A. (2012) 3d face detection and pose estimation in pcl.
[19] Breiman, L. (2001) Random forests. *Machine Learning, 45(1):*5-32,
[20] Fanelli, G., Weise, T., Gall, J., and Gool, L.(2011) Real time head pose estimation from consumer depth cameras. *Proceedings of the 33rd international conference on Pattern recognition, DAGM'11*, (Berlin, Heidelberg : Springer-Verlag), 101-110
[21] Viager, M. (2011) Analysis of Kinect for mobile robots. *Technical report, Technical University of Denmark, Lyngby, Denmark, Mar*.