DETECTION OF 3D POSITION OF EYES THROUGH A CONSUMER RGB-D CAMERA FOR STEREOSCOPIC MIXED REALITY ENVIRONMENTS

Manuela Chessa[1], *Matteo Garibotti*[1], *Guido Maiello*[1,2,3]
Lorenzo Caroggio[1], *Huayi Huang*[1], *Silvio Sabatini*[1], *Fabio Solari*[1]

[1] Dept. of Informatics, Bioengineering, Robotics and System Engineering, University of Genoa, Italy
[2] Dept. of Psychology, Northeastern University, Boston, MA, USA.
[3] UCL Institute of Ophthalmology, University College London, London, UK.

ABSTRACT

A novel approach to track the 3D position of the user's eyes in stereoscopic virtual environments, where stereo glasses are worn, is proposed. Such an approach improves a state-of-the-art real-time face tracking algorithm by addressing the occlusion due the stereo glasses and providing estimation of eye position based on biometric features. More generally, our solution can be seen as a proof of concept for a more robust approach to improving motion tracking techniques. In particular, the proposed technique yields accurate and stable estimates of the 3D position of the user's eyes, while the user moves in front of the stereoscopic display. The correct tracking of both eyes' 3D position is a crucial step in order to achieve a more natural human-computer interaction which diminishes visual fatigue. The proposed approach is validated through quantitative tests: (i) we assessed the accuracy of our algorithm for tracking the 3D position of users' eyes with and without stereo glasses; (ii) we have performed a perceptual assessment of the natural interaction in the virtual environments through experimental sessions with several users.

Index Terms— Human-computer interactions, Virtual reality, Augmented reality, stereoscopic 3D entertainment system, computer vision

1. INTRODUCTION

In recent years there has been a growing interest in improving human-computer interaction by exploiting technologies for rendering stereoscopic 3D contents and employing affordable RGB-D camera as motion capture devices. Stereoscopic rendering is used in professional applications, such as data visualization and rehabilitation systems [2, 6], as well as in the entertainment field [11]. The current widespread use of visual content and human-computer interfaces is driving the development of techniques that simplify interaction and diminish visual fatigue (for recent reviews see [1, 5]). A more naturalistic interaction can be obtained through virtual environments that mimic the real-world (i.e. ecologically valid [2]). The intense use of digital devices that convey visual information through new multi-modal stimulation can lead to visual fatigue [13].

These considerations are especially relevant for stereoscopic visualizations and dynamic interaction systems where the user's whole body is tracked. The misperception of the 3D layout of the virtual objects and environment is the most important open issue when a user is free to move in front of a stereoscopic display. We have addressed this issue in recent studies [4, 12], where we have developed a novel approach to the rendering of 3D stimuli through a stereoscopic projection which takes into account the 3D position of the user's eyes, and thus significantly reduces the distortions associated with discrepancies between the positions of the virtual cameras and the eyes.

In this paper, we focus on the refinement of the low cost mixed reality system that enables natural human-computer interactions, that we have presented in [7]. It is worth noting that, in a stereoscopic 3D system, it is only necessary to know the position of the eyes, and not the gaze direction or the precise location of the iris, in order to optimally render the correct 3D perception of the scene. The underlying idea of using off-the-shelf technologies allows a widespread diffusion of the devised solutions not only in the research field, but also into everyday life devices and systems. The proposed system is composed of a common workstation, a 3D TV and a Kinect device employed as a color and depth (RGB-D) camera.

A crucial component of the system is the algorithm for accurate and stable estimation of the 3D position of both the eyes of the user. In order to obtain a natural interaction that minimizes visual fatigue, a virtual reality system has to take into account the biometric features of the specific user, e.g. see [10]. For example, different users have different inter-ocular distances, and the precise inter-ocular distance of a specific user is necessary to create the geometry which correctly renders virtual objects within a stereoscopic virtual reality system. The position of the eyes is often determined by using measurements based on the characteristics of the stereo glasses (e.g. see [3]) and not on the shape of the user's face. We wished to employ biometric features to tailor the system to an individual user. However, in most stereoscopic virtual environments the user wears stereo glasses, which can lead to tracking errors.

More generally, in applications in which it is essential to accurately track a user's biometric features, an important issue to address is what to do when tracking of said features fails. We propose that by first carefully mapping out the relationship between an individual user's biometric features, we can successfully resolve this issue. When tracking of the features of interest fails, other features can be employed to estimate the position of the features of interest. The approach we present in this manuscript can thus be seen as a proof of concept for a more general solution for improving motion tracking techniques.

Errors of a few centimeters in tracking the position of the user's eyes along the horizontal (x-axis) and vertical (y-axis) dimensions can be geometrically estimated and produce perceptual errors along the same axes of the same order of magnitude. Errors in estimating the interpupillary distance (IPD) have the same perceptual effect as errors in estimating the position of the eyes along the z coordinate (depth): the perceived position of the object is displaced in depth with respect to the intended position of the object. For example, Figure 1 shows the geometrically predicted magnitude of the perceptual error along the z-axis while viewing a virtual object placed 15 cm in front of the monitor when tracking errors of the user's eyes' position of up to 5 cm occur along the z-axis and up to 2 cm occur in estimating the user's IPD. When the user is closer to the monitor (Z=25 cm, Figure 1 left) errors in tracking the user's position along the z-axis produce a similar amount of perceptual error as errors in estimating the user's IPD. At farther distances (Z=75 cm, Figure 1 right), errors in tracking the user's position along the z-axis produces smaller perceptual errors than errors in estimating the user's IPD.

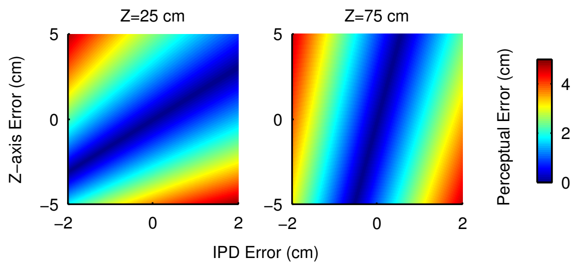

Fig. 1. Geometrical prediction of perceptual errors following incorrect 3D rendering due to errors in tracking the user's eyes true position. A virtual object is stereoscopically rendered to be 15 cm in front of the computer monitor while a user is at 25 cm (left image) or at 75 cm away (right image) from the surface of the monitor. The user's true IPD is 6.5 cm.

We wished to evaluate the impact of wearing stereo glasses on current state-of-the-art face-tracking algorithms, and propose a solution to the problem of correctly identifying the eyes of a user in 3D space during stereoscopic visualization with our proposed system. Considering our aim of using low-cost devices we were particularly interested in exploiting the already available features of the Kinect device. In [9] a 3D elastic surface with non-linear deformations is considered in order to model a human face, and the 3D face Candide model [1] is employed. The same model is adopted in the Microsoft Face Tracking SDK that we use in our proposed system (see Section 2.1.2 for details). Our proposed algorithm improves upon this system by bettering the reliability of the measure of the position the user's eyes in 3D space. Our solution also provides the framework to ask an interesting question about perceptual experience: do small ($\sim$ 1 cm) errors in the geometry of virtual reality rendering affect user perception and ability to naturally interact within a virtual reality system?

The main contributions of this paper are as follows:

- A technique to overcome the problem of the occlusion due to the stereo glasses (see Section 2.2). In particular, the estimation of the 3D position of the eyes is based on biometric characteristics of the specific user. This is achieved by improving the 3D Candide Model [8], as implemented in the library we use for face tracking. It is worth noting that such a model is the basis of current state-of-the-art algorithms (e.g. see [9]). Our solution can be extended to any other situation in which tracking of a particular biometric feature fails.
- A dataset of the RGB-D sequences of faces with different orientations with and without occlusions due to stereo glasses.
- An assessment of the effect of small geometric errors in virtual reality rendering on the performance of users in a simple hand-eye coordination task (see Section 3.2). We performed experiments with several users in order to evaluate whether functional perception of 3D virtual objects is robust to the initial error introduced by the face-tracking algorithm when users wear stereo glasses.

2. EYE DETECTION

The functions provided by the Kinect SDK to track human faces adopts a model (i.e. a mesh) composed of one hundred points. Two of the points in the model represent the center of the user's left and right eyes. The possibility of a user wearing glasses is not taken into account by the face Candide model employed by the Kinekt SDK. When a user wears stereo glasses to view stereoscopic content, the tracking performed by the functions of the Microsoft SDK fails to correctly identify the position of the user's eyes, as can be seen in Figure 2. Our first aim was to find a technique that would accurately track the eyes of a user wearing the stereo glasses.

[1]http://www.bk.isy.liu.se/candide/

One possible approach would have been to adapt the mesh face model to match a face with glasses. However this would have been a solution tailored to our specific problem. We thought a more useful approach would be to find a way to deal with tracking failures in general. Our proposed solution is to define the relationships between all tracked biometric features, and employ the features that are reliably tracked to estimate those for which tracking fails.

2.1. Hardware and software components

2.1.1. Hardware components

The proposed algorithm was implemented on a computer running on Windows 7 Professional composed of an Intel i7 870 CPU, 8GB of RAM, and an NVidia Quadro 600 GPU.

The device used for motion tracking was the Kinect, which is a sensor produced by Microsoft as a peripheral for the Xbox 360. The Kinect is composed of an RGB camera with a resolution 640x480 pixels and a sensor for 3D mapping. The 3D sensor comprises an infrared projector and a camera (320x240 pixels) sensitive to the same infrared band with the ability to detect the infrared patterns reflected from the objects. In addition, the Kinect holds four microphones, a three axes accelerometer and the fan for heat dissipation. The main strength of the Kinect is the ability to combine many features at a very low price: indeed, despite being equipped with several hardware resources, it is sold at a very competitive price. Furthermore, accessibility is made easy by standard USB connection, which makes it compatible with most computers on the market.

2.1.2. Software components

Software was written in the C++ programming language and the integrated development environment (IDE) employed was Visual Studio 2012 Ultimate. The main libraries used for the development of the project were the Kinect for Windows Software Development Kit (SDK) (specifically the Face Tracking SDK [2]), and the OpenCV library [3] for image processing. Data analysis was performed in Matlab (MathWorks).

The *Microsoft SDK* allows programmers to build applications that can track human faces in real time. The Face Tracking module can parse an RGB and depth image from the Kinect and infers the pose of the head and facial expressions in real time. The 3D positions estimated by the tracking functions are referred to a coordinate system in which the origin is located at the center of the optical sensor, the Z axis is directed towards the user, while the Y axis is directed upward.

[2]http://msdn.microsoft.com/en-us/library/jj130970

[3]http://opencv.org/

2.1.3. Virtual Reality System

We tested our proposed solution for face tracking within the framework of the Virtual Reality system we have previously implemented in our laboratory. Our Virtual Reality system uses the position of the user's eyes to update the visualization accordingly to the user's point of view. Using a proprietary technique, this system lets the user perceive virtual objects in the most natural way possible, in terms of position, size and behavior of virtual objects versus real ones. The system is also capable of tracking the user's hands as he or she is interacting with real and virtual objects, which creates a Natural User Interaction System. In Figure 3 a picture of the Virtual Reality System used is shown.

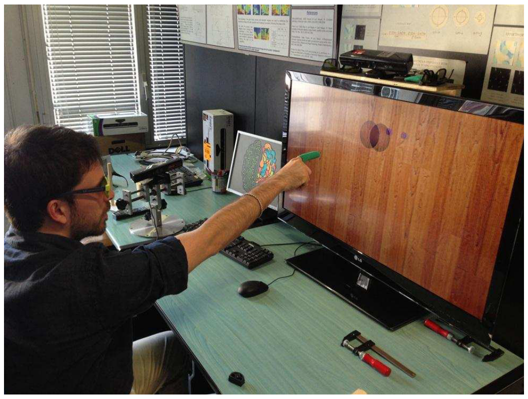

Fig. 3. Picture of the virtual reality system setup. It is possible to see a simple demo running on the 3D monitor, and the two Kinects needed for the tracking algorithm.

2.2. Detection of eye position

When the user wears stereo glasses, as previously said, the tracking function of the Microsoft SDK is destabilized. The geometry of surface of the face is modified by the presence of stereo glasses. The Microsoft face tracking functionality is forced to fit a distorted model of the face with a consequent error in estimating the eyes' position (see Fig. 2(a)).

However, we observed that the lower points of the face model, which are associated to nose, lips and chin, are correctly tracked. We took advantage of this observation and located the most stable point on the user face by comparing the position of the points, computed on the face by Microsoft SDK, first without and then with stereo glasses. The most stable point was found to be the upper lip (see Fig. 2(b)). From this point which was reliably tracked we were then able to estimate the correct position of the user's eyes even when wearing stereo glasses.

To correctly estimate the position of the user's eyes we developed a correction algorithm based on two other resources that the face tracking functionality of Microsoft SDK offers:

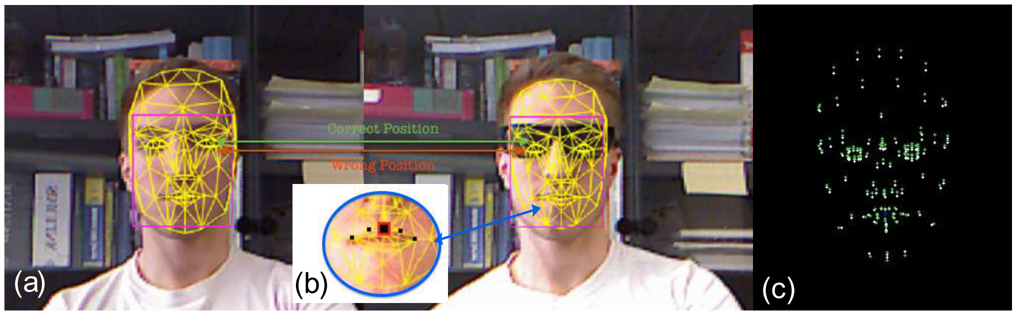

Fig. 2. Face tracking with the Microsoft Face Tracking SDK. (a) A user, sitting in the same position in 3D space, is tracked while not wearing glasses (left) and while wearing glasses (right). The face model is superimposed on the face of the user. When the user is not wearing glasses, the eye position is correct (green arrows). When the user is wearing glasses, the mesh is distorted and the eye position is estimated to be below the stereo glasses (red arrows). (b) The mesh, even when the user is wearing stereo glasses, is not distorted around the mouth of the user.

- the *translation* vector T that describes in the 3D world coordinates (X,Y,Z) the center of gravity of the point cloud computed on the user face with respect to the Kinect reference frame;
- the *rotation* matrix R of the point cloud about the center of gravity.

The proposed algorithm consists of two parts as follows.

2.2.1. *Calibration*

While a user initially does not wear stereo glasses the positions of the eyes with respect to the center point of the upper lip are computed as follows:

- Acquire, once the user is reliably being tracked, the 3D coordinates of both eyes and upper lip;
- Subtract from these three points the translation at the time of calibration T_C;
- Remove from these three points the rotation at the time of calibration R_C;
- Store the true position of the eyes with respect to the center of the upper lip.

Initial observations showed that the Microsoft SDK begins tracking the user face by applying a default modeled mesh of a face. After several seconds of tracking, the default mesh is adapted onto the user's biometric features, resulting in more precise tracking. Due to this behavior the calibration procedure was initiated after 5 seconds, so to use a more precise modeled mesh.

The coordinates of left eye X_L, right eye X_R and center of the upper lip X_U obtained by removing rotation and translation, i.e. the normalized coordinates (roto-translated in the frontal position with respect to the Kinect reference frame), are:

$$\begin{aligned} X_L &= R_C^{-1}(X_{LC} - T_C) \\ X_R &= R_C^{-1}(X_{RC} - T_C) \\ X_U &= R_C^{-1}(X_{UC} - T_C) \end{aligned} \quad (1)$$

where X_{LC}, X_{RC}, and X_{UC} denote the position of left eye, right eye and midpoint of the upper lip at the time of the calibration as estimated by the Kinect Microsoft SDK (Fig. 4, middle, shows a sketch of the calibration procedure, point 3).

The vectors that describe the position between eyes and lip are $D_{LU} = X_L - X_U$ and $D_{RU} = X_R - X_U$ that denote the lip to left eye and the lip to right eye vectors, respectively.

2.2.2. *Real-Time Correction*

During a working session, after having performed the calibration procedure, the user wears stereo glasses and for each frame the correct position of eyes is recomputed as follows (see Fig. 4):

- (1) For every frame i obtain the current position X_{Ui} of the upper lip;
- (2) Remove the current translation T_i and rotation R_i ;
- (3-4) Compute the position of both eyes $X_{Ri,new}$ and $X_{Li,new}$ by adding eyes-lip vectors D_{LU} and D_{RU} ;
- (5)Add rotation R_i and translation T_i to the eyes' points.

The wrong reconstruction due to the presence of stereo glasses causes a distortion of the modeled mesh on the face. However, the effect of this distortion on the translation vector and rotation matrix is minimal since these quantities are computed on the center of mass of the point cloud (comprised of one hundred points) representing the user's face with respect

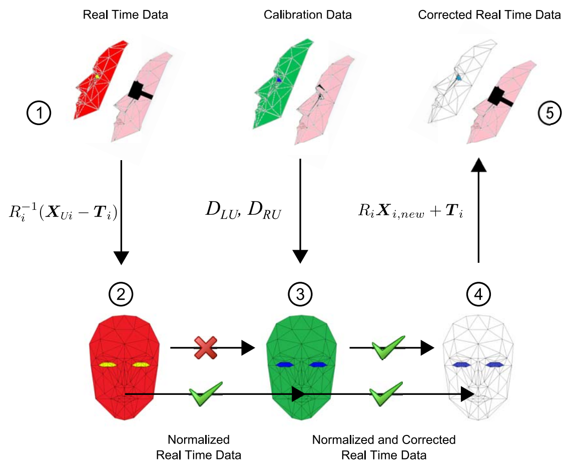

Fig. 4. Real-time correction procedure of eyes' position (see section 2.2.2 for details).

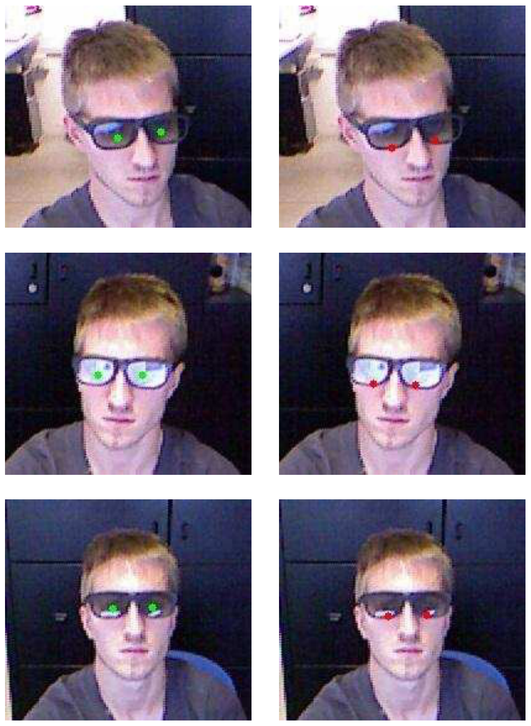

Fig. 5. Tracking the 3D eye position of a user wearing stereo glasses. Right column: errors in tracking with the Microsoft Face Tracking SDK. Left column: correct identification of both eyes' 3D position employing our proposed algorithm.

to the Kinect reference frame. Furthermore, our algorithm takes into account the current roto-translation, but relies on the position of the upper lip that is not affected by the presence of the stereo glasses, and the position of the eyes is computed by using the relative vectors D_{LU} and D_{RU}. It is worth pointing out that this algorithm, being composed of simple mathematical manipulations, does not introduce significant delays in the elaboration of eye position, managing to keep up with the Kinect's original frame rate.

3. RESULTS

3.1. Detection of eye position

Figure 5 qualitatively compares the tracking of the position of the eyes using the Microsoft Face Tracking SDK (right column) and our proposed algorithm (left). The Microsoft Face Tracking SDK has the tendency of incorrectly identifying the eye position on the lower rim of the stereo glasses. Our algorithm instead correctly identifies the position of both eyes.

We have collected a dataset of RGB-D sequences of faces for future comparisons. The dataset consists of acquisitions from multiple subjects, and we will continue to expand the dataset with new subjects in the future. Multiple acquisition, with and without stereo glasses and with different orientations are performed for each person, each of them lasts 10 seconds. Each acquisition is saved in a .xed file format and can be opened using Microsoft Kinect Studio, an application provided along with the Kinect SDK that is able to record the output from Kinect (RGBD data) and play back the recorded data at a later time. The dataset is available to the Computer Vision community, at `www.pspc.unige.it/Research/3Deye.html`.

To quantitatively asses the difference between our algorithm and the Microsoft Face Tracking SDK two carefully controlled data acquisition sessions were performed on one experienced subject. During the first recording session the subject was not wearing glasses. During the second session the subject wore the stereo glasses mid-session. During the first session the subject was free to move his heads. During the second session the subject had to hold his head still with his chin placed in a chin rest. The calibration procedure described in section 2.2.1 was performed before the acquisition of each clip and stored.

For each frame in each session, eye position was computed both with the Microsoft Face Tracking SDK and our proposed algorithm. To evaluate the performance of our proposed algorithm we computed the tracking error of the Microsoft Face Tracking SDK when the subject was wearing glasses, and compared this to the tracking error of our proposed solution when the subject both was and was not wearing glasses.

Fig. 6 shows the error along the three axes as well as the error in estimating IPD. The tracking error of the Microsoft Face Tracking SDK (Fig. 6 left) was defined as the absolute difference in measured eye position between when the subject was not wearing the stereo glasses and when the subject wore the stereo glasses. The tracking error of the proposed solution with glasses (Fig. 6 middle) was defined as the absolute difference in measured eye position between when the

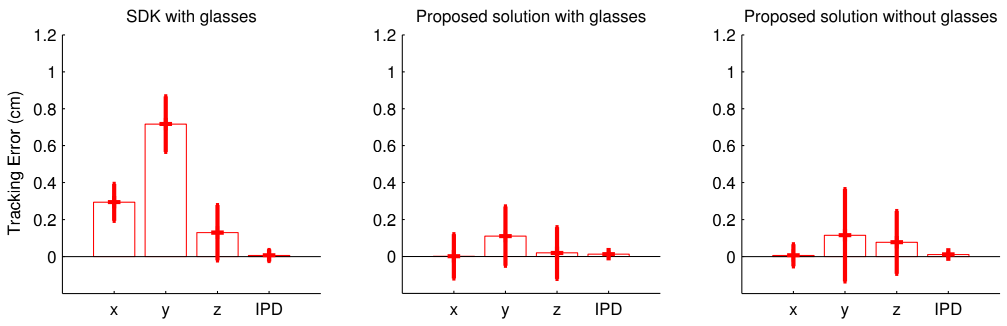

Fig. 6. Comparison of tracking errors when estimating eye position using the Microsoft SDK and the proposed technique. Data are the mean error along the x, y and z axes, as well as mean error in estimating user's IPD. Error bars represent ±1 standard deviation. Left: Tracking error of the Microsoft Face Tracking SDK with the stereo glasses. Middle: Tracking error of the proposed solution with the stereo glasses. Right: Tracking error of the proposed solution without the stereo glasses.

subject was not wearing the stereo glasses and tracking was performed with the Microsoft Face Tracking SDK, against when the subject wore the stereo glasses and tracking was performed with our proposed solution. The tracking error of the proposed solution without glasses (Fig. 6 right) was defined as the absolute difference in measured eye position between when the subject was not wearing the stereo glasses and tracking was performed with the Microsoft Face Tracking SDK, against when the subject was not wearing the stereo glasses and tracking was performed with our proposed solution.

These data confirm that our proposed solution successfully remove tracking errors. Figure 6 (left) shows how, when stereo glasses are worn, the Microsoft SDK introduces errors along the x, y and z axes of 0.7, 0.3 and 0.1 cm respectively, whereas no measurable error is introduced in the estimate of the user's IPD. The errors in tracking with our proposed technique (Fig. 6 middle) when stereo glasses are worn are reduced along all axes. In particular, the error along the x and z axes is nullified, and there is a seven fold reduction in error along the y axis. Furthermore, our proposed technique introduces no measurable error when stereo glasses are not worn (Fig. 6 right). The estimate of the user's IPD, which is a particularly important parameter for correct 3D rendering, is never affected. These results prove that our technique, both with and without stereo glasses, performed comparably to the Microsoft Face Tracking SDK without stereo glasses, thus eliminating the tracking errors introduced by the Microsoft SDK with stereo glasses. It is worth noting that the tracking error of the Microsoft Face Tracking SDK and the tracking error of our proposed solution with glasses had to be estimated by comparing different image sequences from the second session of recording, which could introduce noise in the estimate of the tracking errors (but should not bias the result). To avoid this issue as much as possible, the second session of recording was carefully controlled for head movements by requiring the subject to maintain fixed head position using a chin rest.

3.2. Experimental tests

Having validated our tracking technique, we asked whether small errors in positioning of the user's eyes in the geometry of stereoscopic rendering has any impact on the perceptual and functional experience of the user. To answer this question we devised a visuo-motor experimental setup. The testing setup can be seen in figure 3. Two Kinect devices were employed. One Kinect tracked the eye position of the user, while the other tracked the user's hand to monitor interaction with virtual objects. Stimuli were displayed on a 42-inch passive 3D TV. Ten subjects (ages ranging 22 to 47 years) participated in the experiment. All subjects had normal or corrected-to-normal vision and normal stereo vision. The experiment consisted of two consecutive sessions.

In the first test session, each subject was sitting centered in front of the monitor, with head unconstrained. During the second session subjects were randomly placed to the left or right of the center of the monitor. At the beginning of each session the tracking system was calibrated. The task for each subject was to reach with a finger to the perceived position of a virtual target object, which, due to the stereoscopic 3D visualization technique employed [12], had a stable position within the 3D world. The target objects were stereoscopic cubes, 10 cm wide. Observers completed 100 trials per session. For each trial the tracking technique employed was pseudo-randomly chosen between the the Microsoft SDK technique and our proposed solution (50 trials employed the Microsoft SDK technique and 50 employed our proposed technique for eye tracking). For every trial, the position of the cubes was randomly selected from a uniform distribution of space within a defined volume ($30 \times 30 \times 20$ cm). The systems is able to track the position of the finger of the user, in order to verify the error of the perceived position against the imposed position.

Figure 7 shows the raw error data from all participants in the form of scatter plots. From visual inspection of the raw data no apparent difference can be observed between the

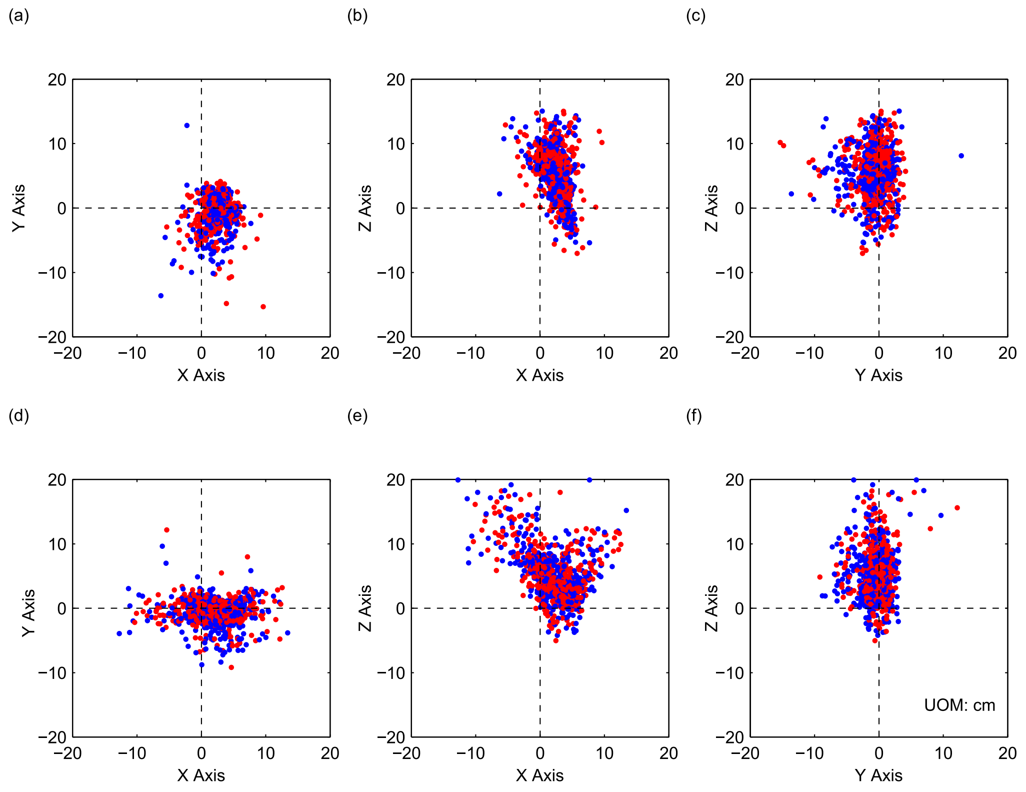

Fig. 7. Plots of perceptual error, computed as the perceived position relative to virtual object position. Blue dots are the errors when the Microsoft SDK Face Tracking was employed, red dots are the errors when our proposed system was employed. More points are near zero axis, the greater the accuracy of the results is. First row shows the error of each trial for the first test session along the x,y axes (a), x,z axes (b) and y,z axes (c). Second row shows the result for each trial of the second session, using the same representation as the first row. Units of measurement are expressed in centimeters.

visuo-motor error produced by the Microsoft SDK Face Tracking and the proposed technique. For each participant, mean magnitude of the error vector was computed for both tracking techniques and for both sessions. Mean error across participants for the first session was 6.8 ± 1.7 cm for the Microsoft technique and 6.8 ± 1.5 cm for the proposed technique, which were statistically indistinguishable (p=0.945, paired samples t-test). Mean error across participants for the second session was 6.7 ± 2.8 cm for the Microsoft SDK and 6.8 ± 2.8 cm for the proposed technique, which again were not significantly different (p=0.52, paired samples t-test). Given our sample size, the variability in our data, and our within subjects design, power calculations determined that there was a 80% probability of detecting a difference in error magnitude of 1 cm, which was our expected effect size, at the 0.05 level. The relatively large size of the target objects, which were 10 cm wide, did not require a lot of precision to localize, which can account for the ~ 7 cm error in pointing responses.

The visuo-motor error distributions shown in Figure 7 qualitatively describe the reaching strategies employed by the users. Since our experimental paradigm was intended to test visuo-motor interaction in an augmented reality setup, the participants could see the target stimulus and their hand at the same time. Thus, the reaching task could theoretically be achieved simply by nulling the binocular disparity between the target and the finger. However, observers did not appear to fully employ this strategy. The centroids of the error distributions along the Z axis (Figure 7 b,c,e,f) are shifted towards positive values, suggesting that observers were unwilling to reach into the virtual target, but had a tendency to stop short. Furthermore, the centroids of the error distributions along the X axis (Figure 7 a,b,d,e) are also shifted towards positive values, which reflects the fact that all observers employed their right hand for the reaching task.

Figure 7 also shows how the visuo-motor error distributions differ along the three main axes and according to subject positioning with respect to the center of the screen. The first row of Figure 7 shows how, when the user is centered with respect to the monitor, the error distribution is (a) roughly equal along the X and Y axes, and (b), (c) wider along the Z axis. The second row of Figure 7 shows that when the users are distributed to the left and right of the center of the monitor, the error distribution widens along the X axis (a). Figure 7 (b) shows a peculiar ”V” pattern, which suggests that the widening of the error distribution along the X axis is due to motor-programming differences when reaching towards a virtual target from the side.

4. CONCLUSIONS

In this paper, we have addressed the issue related to tracking 3D eye position in stereoscopic virtual environments, where the user wears stereo glasses, thus the estimated model of the face is distorted and the position of the eyes is inaccurate. In this situation, the state-of-the-art algorithms show degraded performances. We proposed a technique that allows accurate and stable tracking. We assessed our approach both with and without stereo glasses. Results show that our technique outperforms the Microsoft SDK Face Tracking at correctly identifying the 3D eye-position when users are wearing stereo glasses.

The success of our suggested solution suggests that it is generalizable with regards to improving other motion tracking applications. If multiple biometric features are being tracked, a careful mapping of all the tracked features with regards to each other is a robust way of protecting the system against tracking failures on a subset of the features. If due to occlusions or other issues tracking is lost or unreliable on some of the tracked features, the position of the lost features can be recovered from the rest.

We employed the proposed technique to address the question of visuo-motor misperception in a stereoscopic mixed reality system. Our results show that small ($\sim$ 1 cm) errors in the geometry of stereoscopic rendering have no effect on the functional perception of virtual 3D space. We did observe qualitative differences in the error distributions dependent on the position of the users with respect to the monitor. Since the stereoscopic technique employed has been shown to minimize misperception of the virtual environment [12, 4, 7], these differences are likely due to different motor programming strategies when viewing virtual objects from the side. These results have implications for future assessments of mixed-reality systems which require active user interaction.

5. REFERENCES

[1] Takehiko Bando, Atsuhiko Iijima, and Sumio Yano. Visual fatigue caused by stereoscopic images and the search for the requirement to prevent them: A review. *Displays*, 33(2):76 – 83, 2012.

[2] Corey J. Bohil, Bradly Alicea, and Frank A. Biocca. Virtual reality in neuroscience research and therapy. *Nat Rev Neurosci*, 12(12):752–762, 2011.

[3] Andrea Canessa, Manuela Chessa, Agostino Gibaldi, Silvio P. Sabatini, and Fabio Solari. Calibrated depth and color cameras for accurate 3D interaction in a stereoscopic augmented reality environment. *Journal of Visual Communication and Image Representation*, 25(1):227 – 237, 2014.

[4] Manuela Chessa, Matteo Garibotti, Andrea Canessa, Agostino Gibaldi, Silvio P. Sabatini, and Fabio Solari. A stereoscopic augmented reality system for the veridical perception of the 3D scene layout. In *International Conference on Computer Vision Theory and Applications (VISAPP)*, pages 15–23, 2012.

[5] Eun Joung Cho, Kwan Min Lee, Sung Min Cho, and Yang Hyun Choi. Effects of stereoscopic movies: The positions of stereoscopic objects and the viewing conditions. *Displays*, 35(2):59 – 65, 2014.

[6] P.M. Ferre, R. Aracil, and M.A. Sanchez-Uran. Stereoscopic human interfaces. *IEEE Robotics & Automation Magazine*, 15(4):50–57, 2008.

[7] Matteo Garibotti, Manuela Chessa, Silvio P. Sabatini, and Fabio Solari. An affordable stereoscopic 3d augmented reality system for life-like interaction. In *Proceedings of the 10th European Conference on Visual Media Production*, CVMP '13, pages 5:1–5:10, New York, NY, USA, 2013. ACM.

[8] H. Li, P. Roivainen, and R. Forchheimer. 3-D motion estimation in model-based facial image coding. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 15(6):545–555, 1993.

[9] Javier Orozco, Ognjen Rudovic, Jordi Gonzlez, and Maja Pantic. Hierarchical on-line appearance-based tracking for 3D head pose, eyebrows, lips, eyelids and irises. *Image and Vision Computing*, 31(4):322 – 340, 2013.

[10] K. Ponto, M. Gleicher, R.G. Radwin, and Hyun Joon Shin. Perceptual calibration for immersive display environments. *Visualization and Computer Graphics, IEEE Transactions on*, 19(4):691–700, 2013.

[11] Joe Rivett and Nicolas Holliman. Stereoscopic game design and evaluation. In *Proc. SPIE 8648, Stereoscopic Displays and Applications XXIV*, pages 1–13, 2013.

[12] F. Solari, M. Chessa, M. Garibotti, and S. P. Sabatini. Natural perception in dynamic stereoscopic augmented reality environments. *Displays*, 34(2):142–152, 2013.

[13] K Ukai and P Howarth. Visual fatigue caused by viewing stereoscopic motion images: Background, theories, and observations. *Displays*, 29(2):106–116, 2008.