robotics

MDPI

Article

Adaptive Kalman Filter Applied to Vision Based Head Gesture Tracking for Playing Video Games

Mohammadreza Asghari Oskoei

Department of Computer Science, Allameh Tabataba'i University, Tehran 1489684511, Iran; oskoei@atu.ac.ir

Received: 30 August 2017; Accepted: 13 November 2017; Published: 27 November 2017

Abstract: This paper proposes an adaptive Kalman filter (AKF) to improve the performance of a vision-based human machine interface (HMI) applied to a video game. The HMI identifies head gestures and decodes them into corresponding commands. Face detection and feature tracking algorithms are used to detect optical flow produced by head gestures. Such approaches often fail due to changes in head posture, occlusion and varying illumination. The adaptive Kalman filter is applied to estimate motion information and reduce the effect of missing frames in a real-time application. Failure in head gesture tracking eventually leads to malfunctioning game control, reducing the scores achieved, so the performance of the proposed vision-based HMI is examined using a game scoring mechanism. The experimental results show that the proposed interface has a good response time, and the adaptive Kalman filter improves the game scores by ten percent.

Keywords: optical flow; Kalman filter; visual human machine interface; video game

1. Introduction

People with motor infirmity or lack of full control of their upper limbs have problems working with physical user interfaces, such as gamepads or joysticks. They need an intuitive and hands-free human machine interface (HMI) to work with computer systems. This problem is escalated for pupils with cerebral palsy (CP), encephalitis or upper limb disabilities in using the controller of video games such as the Xbox [1]. To play games, they typically need a hands-free interface that does not require direct physical contact and motor motions. A vision-based human machine interface (VHMI) applies computer vision techniques to detect head gestures and render game commands. It could be used as an alternative to a physical gamepad and provides hands-free HMI to interact with video games without massive upper limb motions. This helps people with motor disability to interact with the game using head gestures [1–3].

Video streams are high-dimensional and often redundant data carrying rich information. Feature tracking (FT), also known as optical flow, is a computer vision-based method to identify motion in a sequence of frames. It detects the flow of hundreds of feature points in consecutive images, which, if properly adjusted, can be used to estimate motion information. For instance, it could be used to recognize head gestures in consecutive frames supplied by a camera [1]. Optical flow naturally involves huge data processing and this is a challenge for real-time applications. Feature selection limits the region of interest and makes the tracking algorithm efficient. Furthermore, a change in the appearance of a moving object, varying illumination and occlusion cut off the continuity of the optical flow and makes it malfunction. The Kalman filter is an optimal estimator and can be used to estimate the trajectory of a moving target in the presence of occlusion.

The application of vision-based HMI using either head gesture or hand movement has long been a favorite of many research works [4–6]. Ernst et al. [7] presented a real-time adaptive method of motion tracking for imaging and spectroscopy, and Huang et al. [8] applied temporal filters to develop a robust face tracking system. Optical flow was used in [9] to build 3D facial expression recognition.

Ali et al. [10] proposed a visual tracking system using the Kalman filter. Maleki and Ebrahim-zadeh [11] developed a visual mouse system using hand motion recognition in consecutive images and finally, Khandar et al. [12] proposed a mouse control system using vision-based head movement tracking.

A visual HMI was developed by Jia et al. [13] to manipulate an electrical wheelchair using head gestures. They built a combination of nose template matching, Camshift object tracking and face detection. Lu et al. [14] represented a vision-based HMI to distinguish nod and shake. The position and orientation of the head were initially identified using the multi-view model (MVM), and later, the hidden Markov model (HMM) was adopted to recognize head gestures using statistical inference. In the following, Lu et al. [15] applied the Bayesian network framework to MVM to identify head gestures. Later on, color information was inserted into the Bayesian network in order to enhance robustness and performance [16].

Head gesture-based HMI usually experiences a deficiency of robustness in performance. This is mainly due to interruption produced by the habitual behavior of eyes' target-tracking. This makes the head return to its original pose immediately after any gesture. This requires fast and precise recognition, in which any tiny head movements are spotted before interruption [1]. The method proposed by Jia et al. [13] could not spot small head movements, since it works based on nose position in the center of the face, and this needs to be large enough. Furthermore, the face detection approach is very dependent on head pose. If the head is facing directly towards the camera, face detection works properly, and it fails when movements make the face unrecognizable. Meanwhile, real-time control of a video game urges rapid and robust reactions, and this is a challenge in feature tracking, which involves a huge number of features under various occlusion and light conditions. Han et al. [17] proposed a real-time method to evaluate the weights of the features using the Kalman filter. It comprises inter-frame predication and single-frame measurement of the discriminative power of features. They showed that this method can stabilize the tracking performance when objects go across complex backgrounds [18,19].

To deal with problems raised due to occlusion, Han et al. [20] applied feature tracking in three steps: initially detecting the moving object, then tracing feature points within the object mask and finally estimating the trajectory of the object. Hua et al. [21] proposed a way to combine occlusion and motion estimation with a tracking-by-detection approach. The motion changes of the object between consecutive frames was estimated from the geometric relation between object trajectories. Real-time vision-based inertial navigation was presented by Mourikis and Roumeliotis [22]. They applied the extended Kalman filter (EKF) and deployed a measurement model to represent the geometric constraints. Shiuh-Ku et al. [23] designed an adaptive Kalman filter (AKF) to track the moving object. They used the rate of occlusion to adjust the error covariance of the Kalman filter, adaptively. The adaptive Kalman filter improves the tracking performance in situations such as changing lighting and partial and/or long-lasting occlusion in real-time applications [19].

This paper proposes the adaptive Kalman filter (AKF) to improve the performance of a vision-based head gesture interface to a video game. The vision-based interface detects the region of the head by face detection and applies optical flow to recognize head gestures. Head gestures are classified into five pre-trained classes corresponding to five commands that manipulate a car in a route with randomly appearing obstacles. Face detection often fails due to either the change in head posture, illumination or occlusion. Failure in face detection leads to missing optical flow and eventually malfunction in manipulating the car. In the video game, any safe movement is rewarded by a positive score, and a malfunction in manipulating the car and hitting obstacles are penalized by a negative score. Hence, the achieved scores of the game can be considered as a measure of the performance of the vision-based interface.

Failures in face detection, no matter the reason, are considered as a kind of occlusion and represented by the occlusion rate. The Kalman filter is to estimate head motion by adjusting in-between measurement and prediction. The measurement is obtained from optical flow, and the prediction is from estimated motion in the recent consecutive frames. The occlusion rate is applied

as an adaptation parameter of the Kalman filter to adjust in-between measurement and prediction. The experiment is designed to evaluate the improvement made by AKF applied to the vision-based interface while playing a video game. This paper is an extension of our previous work [1], where we developed head gesture-based HMI as an alternative to gamepads for the Xbox game.

The paper is organized into five sections. Following the Introduction, Section 2 presents models and methods employed in this study, including the pyramidal implementation of feature tracking, feature selection and the adaptive Kalman filter algorithm. The structure and implementation of the proposed vision-based HMI is described in Section 3. Experiments are conducted in Section 4 to demonstrate the performance of the proposed vision-based HMI. Finally, the paper concludes with a brief discussion and future work in Section 5.

2. Models and Methods

This section describes models and methods that were employed in this study to build the vision-based human machine interface. Figure 1 depicts the schematic diagram of the proposed HMI, which is made up of three main components: feature selection, optical flow estimation and adaptive Kalman filter. Feature selection is to identify the major neighborhood of the moving pixels and solve the optical flow equations for just those areas. Face detection is applied to identify the neighborhood of pixels around the head. The optical flow method introduced by Lucas and Kanade [24] is a widely-used approach to identify motion in consecutive frames. It assumes the flow is constant in neighbor pixels. The adaptive Kalman filter estimates the motion information of the selected features under a weakening condition, such as occlusion.

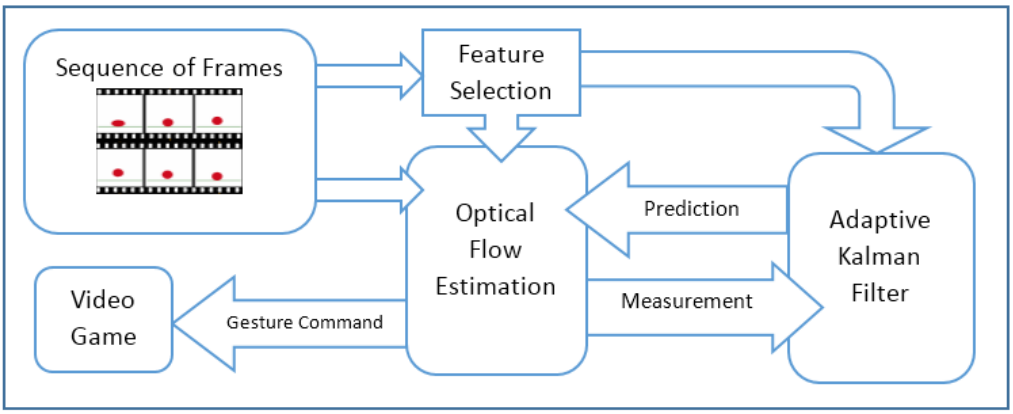


Figure 1. Schematic diagram of vision-based HMI made up of three main components: feature selection, feature tracker and adaptive Kalman filter.

2.1. Optical Flow Estimation

In computer vision, optical flow is the pattern of motions caused by relative movement between the camera and objects. The Lucas-Kanade algorithm [24] is a differential method to estimate optical flow in consecutive frames. It assumes that the flow is constant in the local neighborhood of pixels, known as features. The features are regions of interest for which the optical flow equations are solved. A head gesture in front of a static camera produces a set of local features with almost constant flow. Let us consider $I(u,t)$ representing a feature located at $u = \begin{bmatrix} x & y \end{bmatrix}^T$ in time frame t. The variables of x, y and t are discrete and bounded suitably. Generally, most of the consecutive frames are either similar or strongly related to each other, since they are taken at near time instants. In the case of smooth motion in the scene, it keeps the similarity or relation with slight relocation. Smooth motion

means relocation in the scene that can be captured in consecutive time frames and expressed by (1). Formally, this means that $I(u,t)$ is not arbitrary, but satisfies the following property:

$$I(u+d,t) \approx I(u,t+1) \tag{1}$$

To estimate optical flow in two successive grayscale frames, represented by $I(u)$ and $J(u)$, the objective is to find a point $v = u + d$ such that $I(u)$ and $J(v)$ are approximately similar. The displacement vector $d = \begin{bmatrix} d_x & d_y \end{bmatrix}^T$ is used to work out the velocity of the flow at the point of u. The similarity is defined in a sub-window centered at point u, with a size of $(2w_x + 1) \times (2w_y + 1)$, and the displacement vector of d is calculated in a way that minimizes the residual function (2).

$$e(d) = e(d_x, d_y) = \sum_{x=u_x-w_x}^{u_x+w_x} \sum_{y=u_y-w_y}^{u_y+w_y} [I(u) - J(u+d)]^2 \tag{2}$$

The sub-window is called the integration window and the displacement vector of d is calculated iteratively. The window size and initial guess of the displacement vector are both crucial parameters for the time complexity of optical flow estimation. Furthermore, the size of the integration window makes a trade-off between the accuracy and sensitivity of estimation. A small size of integration window enables the detection of very tiny motions around u; meanwhile, this may miss the detection of large motions. A large-sized integration window leads to the detection of displacements with low resolution. Practically, the integration window is chosen between 3×3 and 7×7. In general, displacements should be less than the window size, i.e., $d_x \leq w_x$ and $d_y \leq w_y$, but the pyramidal implementation of Lucas-Kanade algorithm [24] overcomes this limitation.

2.2. Pyramidal Lucas-Kanade Algorithm

An implementation of the Lucas-Kanade (LK) algorithm was presented by Bouguet [25]. Pyramidal implementation reconstructs an image window in higher levels with a smaller size. This allows examining windows at different levels and detecting motions larger than the size of the integration window. For an image with a size of $n_x \times n_y$ at the base level $I^0 = I$, the pyramidal implementation reconstructs images at higher levels with smaller sizes, recursively. For instance, I^1 is formed as a quarter size of I^0, and I^2 is formed as a quarter size of I^1, and so on. The pyramidal level I^L is formed from I^{L-1} based on (3). As shown in Figure 2, consecutive images were formed into pyramids with 2, 3 or 4 levels, and the LK algorithm was applied to the levels, respectively. The number of levels in the pyramid is chosen according to the maximum expected optical flow in the image [25].

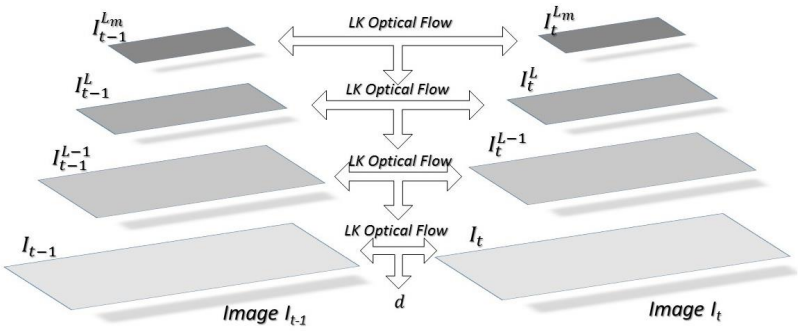


Figure 2. Pyramidal implementation of the Lucas-Kanade algorithm.

$$\begin{aligned} &I^L(x,y) = \tfrac{1}{4} I^{L-1}(2x,2y) + \\ &\tfrac{1}{8}[I^{L-1}(2x-1,2y) + I^{L-1}(2x+1,2y) + \\ &I^{L-1}(2x,2y-1) + I^{L-1}(2x,2y+1)] + \\ &\tfrac{1}{16}[I^{L-1}(2x-1,2y-1) + I^{L-1}(2x+1,2y+1) + \\ &I^{L-1}(2x-1,2y+1) + I^{L-1}(2x+1,2y-1)] \end{aligned} \tag{3}$$

By constructing pyramidal levels, the KL algorithm is initially applied at highest level L_m. It estimates optical flow by calculating the displacement vector that minimizes the residual function at this level. Then, it is used as the initial guess to speed up optical flow estimation in the level L_{m-1}. The optical flow is propagated to level L_{m-2}, and so on, up to the level *0*, the original image [25].

In level L, given the initial guess $g^L = \begin{bmatrix} g_x^L & g_y^L \end{bmatrix}^T$ and optimum displacement $d^L = \begin{bmatrix} d_x^L & d_y^L \end{bmatrix}^T$ result in the initial guess in $L-1$ by $g^{L-1} = \left(g^L + d^L\right)/2$, the algorithm ends as it reaches level *0*. To initialize the algorithm, we set $g^{L_M} = \begin{bmatrix} 0 & 0 \end{bmatrix}^T$ and the final optical flow solution is obtained by $d = \left(g^0 + d^0\right)/2$. The algorithm is expressed by the extended form of (4).

$$d = \sum_{L=0}^{L_m} 2^L d^L \tag{4}$$

Equation (4) demonstrates the advantage of the pyramidal implementation. Even though, the small window size at higher levels ends up with a large displacement vector at lower levels. This supports a balance between the accuracy and sensitivity of the algorithm. Given d_{max} as the maximum displacement vector at each level, the pyramidal implementation can detect motions with displacement up to $d_{max\,final} = (2^{Lm+1} - 1)\, d_{max}$. For example, a pyramid with $L_m = 3$ can detect motions fifteen-times larger than of what ordinary LK can detect [24,25].

Now, let us review the calculation of the LK algorithm at each level. Consider two successive frames in level L, as A and B in (5), and the residual function that should be minimized at $p = [x\ y]^T = u^L$ with displacement $v = [v_x\ v_y]^T$ is defined in (6).

$$\begin{aligned} A(u) &\equiv I^L(u) \\ B(u) &\equiv J^L(u + g^L) \end{aligned} \tag{5}$$

$$e(v) = \sum_{x=p_x-w_x}^{p_x+w_x} \sum_{y=p_y-w_y}^{p_y+w_y} [A(p) - B(p+v)]^2 \tag{6}$$

The optical flow is displacement v^* that makes the first derivative of the residual function (6) equal to zero and is obtained by (7).

$$v^* = G^{-1} b \tag{7}$$

where:

$$\begin{aligned} G &\equiv \sum_{x=p_x-w_x}^{p_x+w_x} \sum_{y=p_y-w_y}^{p_y+w_y} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \\ b &\equiv \sum_{x=p_x-w_x}^{p_x+w_x} \sum_{y=p_y-w_y}^{p_y+w_y} \begin{bmatrix} \delta I\ I_x \\ \delta I\ I_y \end{bmatrix} \end{aligned} \tag{8}$$

and:

$$\begin{aligned} \delta I &= A(x,y) - B(x,y) \\ I_x &= \frac{\partial A(x,y)}{\partial x} = \frac{A(x+1,y) - A(x-1,y)}{2} \\ I_y &= \frac{\partial A(x,y)}{\partial y} = \frac{A(x,y+1) - A(x,y-1)}{2} \end{aligned} \tag{9}$$

Spatial derivatives I_x and I_y of the fixed frame are calculated once at the beginning of the iterations. Therefore, the matrix G remains fixed during the iterations, and this brings a computational advantage. The residual difference between the frames (b_k) needs to be calculated at each iteration. More details of the computations are presented in [25].

2.3. Feature Selection

In the previous section, optical flow was estimated using tracking features in the consecutive frames. Due to computational limits, it could not be applied to whole image pixels, and we should select points for tracking. This is called feature selection. Based on the KL algorithm, the G matrix of the selected feature, defined in (8), should be non-singular, and in practice, its minimum eigenvalue should be large enough. Therefore, this could be used as a criterion to select feature points, which are typically good for tracking. The feature points are chosen in a way that they are far enough from each other to avoid locality in the detected optical flow. The detailed procedure of feature selection is presented in [25].

Face detection often fails due to the change in head pose, illumination and occlusion. Failure in face detection leads to missing optical flow and eventually malfunction in manipulating the car. Failures in face detection, no matter the reason, are considered as a kind of occlusion and represented by the occlusion rate. The occlusion rate is in the range of [0, 1] and worked out by the number of frames without a detected face in the recent k frames divided by k. Its value of one means that there is no detected face in the last k frames, and the value of zero means all the recent frames had detected faces. The occlusion rate is applied as an adaptation parameter of the Kalman filter to adjust in-between measurement and prediction.

2.4. Adaptive Extended Kalman Filter

The Kalman filter is an optimal state estimator and has a wide range of applications in technology. It is a recursive algorithm that produces estimates of states that tend to be more precise than those based on a single measurement alone. The algorithm is built on two models, the state model and the measurement model, which are defined in (10) and (11), respectively.

$$X(t) = A(t)X(t-1) + W(t) \tag{10}$$

$$Z(t) = C(t)X(t) + V(t) \tag{11}$$

$$E\left\{W(k)W(l)^T\right\} = Q\delta_{kl},\ E\left\{V(k)V(l)^T\right\} = R\delta_{kl} \tag{12}$$

where $X(t)$ and $Z(t)$ are state and measurement vectors and $W(t)$ and $V(t)$ are white Gaussian noise with zero mean and the covariance matrix of Q and R, respectively, and δ_{kl} denotes the Kronecker delta function. The Kalman filter consists of two steps: prediction and correction. In the prediction step, it deploys the state model and produces the estimate of the current state and its uncertainty.

$$X^-(t) = A(t-1)X^+(t-1) \tag{13}$$

$$P^-(t) = A(t-1)P^+(t-1)A(t-1)^T + Q \tag{14}$$

where $X^-(t)$ and $X^+(t)$ are prior and posterior state estimates and $P^-(t)$ and $P^+(t)$ are the prior and posterior state covariance error, respectively. In the correction step, once the measurement is

observed, the posterior estimates are updated using a weighted average, in which more weight is given to estimates with higher certainty.

$$K(t) = P^{-}(t)C(t)^{T}\left(C(t)P^{-}(t)C(t)^{T} + R\right)^{-1} \tag{15}$$

$$X^{+}(t) = X^{-}(t) + K(t)\left(Z(t) - C(t)X^{-}(t)\right) \tag{16}$$

$$P^{+}(t) = (I - K(t)C(t))P^{-}(t) \tag{17}$$

The Kalman gain factor, $K(t)$, is in an inverse proportion of the measurement error R, and it adjusts weight in-between the measurement and the predicted estimate. The extended Kalman filter (EKF) has been developed to work on the system, in which both or one of the state and measurement models may be nonlinear. In EKF, the partial derivatives of nonlinear functions, which are the differentiable type, are applied to proceed with the prediction and correction steps. This process essentially linearizes the nonlinear function around the current estimate.

In this work, we used the adaptive Kalman filter that was developed by Shiuh-Ku et al. [23]. The proposed Kalman filter estimates motion information extracted by feature tracking. It adapts the filter parameters to overcome deteriorating conditions such as occlusion. This means that the Kalman gain factor is adjusted based on occlusion. The state model is a linear motion model, which is formed based on the optical flow of the last two frames. The outcome of the LK algorithm is used as the measurement for the correction step. In the consecutive frames, the time interval is very short, then velocity is assumed uniform and used instead of position in the model [23]. Therefore, we can model the displacement vector in time intervals as (18).

$$d(t) = d(t-1) + \nabla d(t-1) \tag{18}$$

where $\nabla d(t-1) = d(t-1) - d(t-2)$, $d(t)$ and $d(t-1)$ are the displacements vectors at frames t and $t-1$, respectively. Having displacement in the x-axis, y-axis directions (d_x, d_y), the state model and the measurement model are defined as (19) and (20), respectively.

$$X(t) = \begin{bmatrix} d_x(t) \\ d_y(t) \\ d_x(t-1) \\ d_y(t-1) \end{bmatrix} = \begin{bmatrix} 2 & 0 & -1 & 0 \\ 0 & 2 & 0 & -1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} d_x(t-1) \\ d_y(t-1) \\ d_x(t-2) \\ d_y(t-2) \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix} w(t) \tag{19}$$

$$Z(t) = \begin{bmatrix} d_x(t) \\ d_y(t) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} d_x(t) \\ d_y(t) \\ d_x(t-1) \\ d_y(t-1) \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \end{bmatrix} v(t) \tag{20}$$

Having the state model and the measurement, we can apply the AKF to track the features effectively in a sequence of frames. The AKF models are designed in such way that the estimate parameters adjust automatically. Occlusion makes the optical flow fail and leads to error measurement for the displacement vector. This means that the occlusion rate is proportional to measurement error and can be applied to adjust estimation parameters.

The Kalman gain factor K as shown in (15) is in inverse proportion to measurement error R. Where there is a lack of occlusion, measurement error of R and prediction error of Q could be assumed infinity and zero, respectively. In the presence of occlusion with a rate of α, the measurement error of R is set to α and the prediction error of Q is set to $1 - \alpha$, if it exceeds a predefined threshold. In other words, if the occlusion rate is less than the threshold, then the measurement result will be trusted more than the predicted one. Otherwise, the system will trust the predicted result completely. By this, the Kalman filter is adjusted based on the occlusion rate, adaptively.

At the end of this section, let us review the structure of the proposed vision-based HMI. Initially, face detection was applied to the stream of frames captured by the camera in front of the subject. It simultaneously identifies the region in which the head in and updates the certainty of face detection. Then, feature points were chosen in the region identified by face detection. Feature tracking was used to estimate the optical flow made by the head gesture. The adaptive Kalman filter is applied to estimate the feature points' motion, particularly in the case of occlusion. In the tracking procedure of the adaptive Kalman filter, a motion model is constructed to build the system state and is applied to the prediction step. The occlusion ratio is applied to adjust the prediction and measurement errors adaptively. The two errors will make the adaptive Kalman filter system trust the prediction or measurement more and more. The occlusion rate is extracted using face detection performance.

3. System Implementation

The proposed vision-based HMI is implemented to work in parallel with a gamepad to manipulate a car in either an Xbox or a home-developed video game. In both games, it detects user's head gestures and overrides commands from the gamepad using gestural instructions. In the Xbox game, the system runs two independent threads simultaneously: a thread that forwards the arriving commands from the gamepad to the video game and a thread that detects gestural commands and overrides some ordinary commands of the gamepad. It was implemented in Java, but equipped with a C++ core (as a DLL file) to handle video data processing and feature tracking using the OpenCV 2.x library [26]. This core was adopted by means of the Java native interface (JNI) in the main program. Figure 3 depicts the hardware setup, and Figure 4 illustrates a subject playing the Xbox game through the vision-based HMI. For the Xbox, the user could run the racing car forward, left and right using head gesture towards the up, left and right directions, respectively. The stop command was associated with the head's normal situation at the beginning. The Xbox Java SDK was deployed to read the controller connected through the USB into a dual-core 3.2-GHz PC, and then, the commands were forwarded to the Xbox via a hardware interface connected to the USB port using the Java Serial Communication SDK [1].

Figure 3. Hardware setup of the vision-based HMI for Xbox [1].

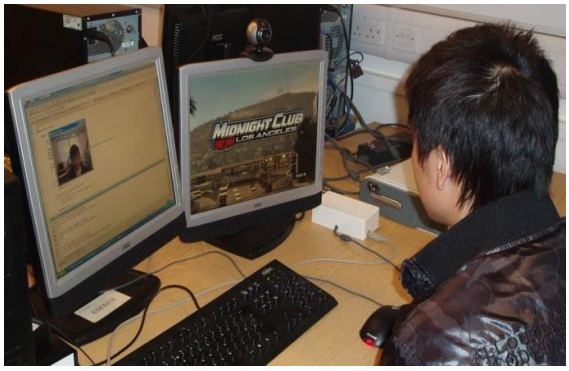
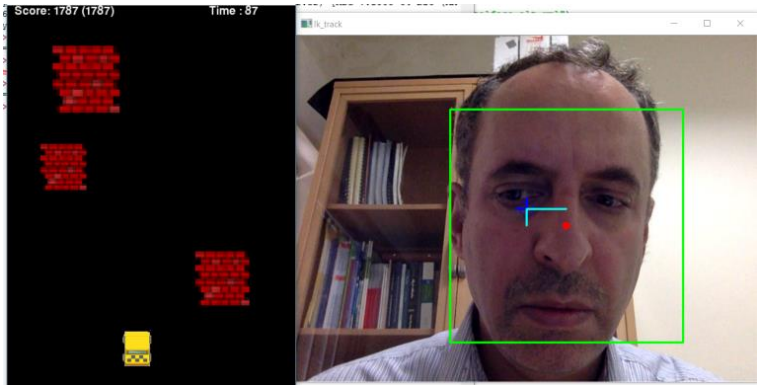


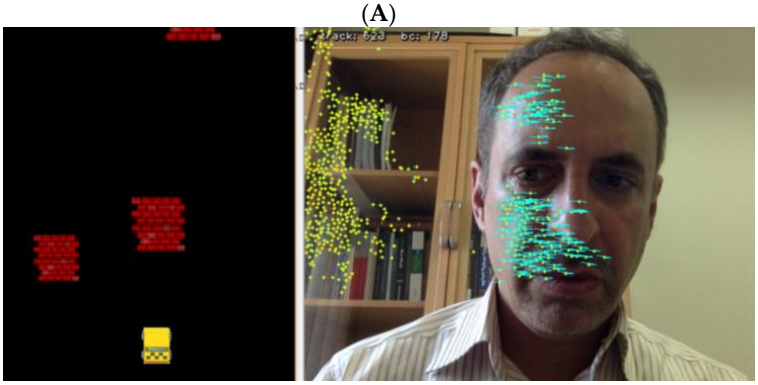
Figure 4. A user playing the Xbox game using the vision-based HMI [1].

The recent vision-based HMI and video game were both developed in Python 3.6 using the OpenCV 3.2 and PyGame libraries, respectively, and run in two threads in parallel using the Python Threading library. The vision-based HMI grabs the video stream captured by the camera, which is fixed on top of a screen illustrating the video game. It has two modules, face and optical flow detectors, and works in two modes: face detection mode and optical flow detection mode. In the former, head gestures are recognized through the shift of detected faces in consecutive frames, as shown in Figure 5A. While in the later, the average of optical flow made by features in the face ROI is used to address head gestures (Figure 5B). The output of the vision-based HMI is passed through two filters: a low-pass filter and the adaptive Kalman filter (AKF); both were implemented in Python using the OpenCV, SciPy and NumPy libraries. The video game is made up of a player yellow box (car) and randomly appearing red boxes (obstacles). The video game and vision-based HMI are run in parallel in multi-thread mode, and subjects should avoid hitting obstacles using head gesture to manipulate the car (Figure 5).

Since there is no direct access to the Xbox scoring system, we adopted a home-developed video game to directly record game scores and evaluate the HMI performance. The game has five ordinary commands: forward, backward, right, left and stop. The commands are produced by vision-based HMI using head gestures. The higher score shows quicker and safer forward driving. The scoring policy is designed for safe driving as much as possible by penalizing wrong movements. Forward movements reward with a positive score, and a wrong step, in which car hits an obstacle or border, is penalized by ten negative scores. The subjects were encouraged to advance to the highest score as much as possible. Hence, the game was played by the subjects until they were not able to get more scores continuously.



(**A**)



(**B**)

Figure 5. *Cont.*

(**C**)

Figure 5. Screenshot of the video game and the proposed vision-based HMI [1]. (**A**) video game and face detection mode of VHMI; (**B**) video game and optical flow detection mode of VHMI; (**C**) feature tracking.

In the operation mode, the vision-based HMI supersedes the gamepad and sends commands directly to the video game. It enables the user to control the game without the gamepad. In the video game, the X and Y axis of the gamepad are assigned to move the car towards the left and right, and the Z axis is for either forward or backward movements. Figure 5 illustrates frames captured by the camera to control the game. Initially, the HMI was calibrated by coordinating motions of the head. Using face detection, a sub-window is localized by the blue frame, and the vision-based HMI becomes activated. Green points depict selected features being tracked on the screen [1]. The blue rectangle shows margins to address various commands. If the green point is inside the rectangle, it means the stop command. Head gestures traced by the green point address X and Y axis values, while distance from the beginning point (red line) addresses the Z axis value (Figure 5). Besides manual selection, face detection is employed to select features in the head area. In the proposed HMI, head gestures are recognized using the feature tracking method boosted by an adaptive Kalman filter. The adaptive Kalman filter estimates the motion of feature points that may get lost due to occlusion. This prevents unwanted interruptions during game playing.

4. Experiments and Results

This section will evaluate the performance of the proposed vision-based HMI based on video games. The video game could involve subjects in a real-time interaction, in which they should produce proper and dexterous reactions with minimum danger for them. The first experiment was designed to compare the performance of the HMI with an ordinary joystick. It evaluates the performance by comparing achieved scores using vision-based HMI and the ordinary joystick while playing a car-racing Xbox game. The subjects were encouraged to gain the highest score as much as possible. Three subjects, aged 15, 21 and 25, participated in the experiments. They sat before a camera and a monitor connected to the video game (Figure 5), aiming to play a game, either with the vision-based HMI or a gamepad. For the vision-based HMI, they had to manipulate the car using head gestures. Increasing speed of the game naturally demands fast and intense reactions that lead to sudden changes in head postures in the

vision-based HMI and likely a high rate of missing data for optical flow estimation. This required adaptation for HMI, so the second and third experiments were designed to evaluate the performance of the vision-based HMI boosted by the adaptive Kalman filter.

4.1. Experiment I

The vision-based HMI is considered as an alternative to the gamepad for disabled people, and with regards to the intrinsic limits, it is not expected to perform as an ordinary game controller. To evaluate the performance, we compared the achieved scores using the proposed HMI with a standard gamepad. The comparison was conducted on scores achieved during playing an identical game, in a constant period of time (practically five minutes) and a unique route [1]. The sessions were repeated three times for each subject. To avoid biased results due to fatigue, the sessions were conducted on different days. A session was comprised of playing the Xbox game twice: the first with the gamepad and the second with the vision-based HMI. We recorded the achieved scores in the first and second games and calculated the rate of changes between them. Given $Score_{gPAD}$ and $Score_{vHMI}$ as the achieved score using the ordinary gamepad and the proposed HMI, respectively, the rate of change in the achieved scores is defined by (21).

$$R = \frac{Score_1 - Score_2}{Score_2} \times 100 \tag{21}$$

As shown in Table 1, the achieved scores using vision-based HMI are about half of the scores achieved by the standard gamepad. Figure 6 illustrates the scores of each session in the form of a bar chart. We applied statistical analyses to interpret the experimental results. As mentioned, the experiment was based on nine sessions and three subjects. Enough time was taken between sessions to provide nine independent observations with an identical distribution. The purpose of statistical analysis is to find statistically meaningful differences over observations with a certain significance.

Table 1. Achieved scores during game play using vision-based HMI ($Score_{gPAD}$) compared to the standard gamepad ($Score_{vHMI}$) in the 1st experiment [1].

	$\mathbf{Score_{gPAD}}$	$\mathbf{Score_{vHMI}}$	$\mathbf{R = \frac{Score_{vHMI} - Score_{gPAD}}{Score_{gPAD}}}$
Subject 1			
Session 1	1307	591	−54.8%
Session 2	1142	589	−48.4%
Session 3	1305	366	−71.9%
Subject 2			
Session 1	2234	764	−65.8%
Session 2	1664	925	−44.4%
Session 3	1686	834	−50.5%
Subject 3			
Session 1	2088	646	−69.1%
Session 2	2091	984	−52.9%
Session 3	1812	954	−47.3%
Mean ± STD			**−56.1 ± 10%**

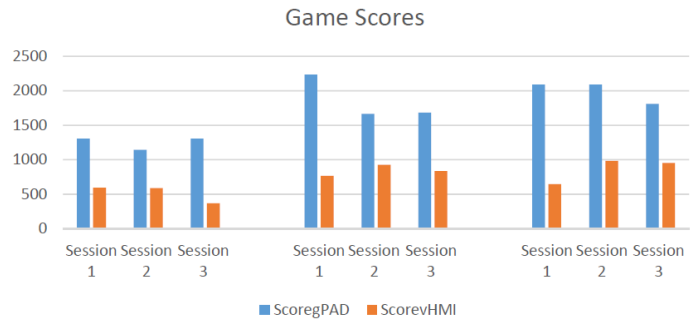


Figure 6. Comparison of the achieved scores using the standard Xbox gamepad and vision-based HMI in the first experiment.

Analysis of variance (ANOVA) is a statistical approach in which the variance of observations is partitioned into components due to different explanatory variables. It is a powerful tool, but only applicable for data with a normal distribution. To analyze the results of the experiments in this paper, due to a low rate of observations and their unknown distribution, non-parametric approaches were used [27]. Wilcoxon rank-sum and Kruskal-Wallis are two non-parametric statistical methods that are adopted to compare samples from two and more than two groups, respectively [1–3].

Wilcoxon rank-sum [27] is a two-sided test for two groups of data with independent samples and an identical distribution to recognize whether their medians are equal. Kruskal-Wallis tests the equality of samples' median among groups. Intuitively, it is identical to the one-way ANOVA and the extension of the Wilcoxon rank-sum for more than two groups. In both, data are replaced by their ranks in the analyses. Both methods provide a p-value. The p-value is a probability that the groups are drawn from the same population or from different populations with the same median. If the p-value is near zero, this suggests that at least one group's median is significantly different from the other median. The critical p-value, which determines whether the result is judged as "statistically significant", is chosen to be 0.05 in this paper [1].

Figure 7 demonstrates the statistical analysis applied to the experimental results. As is shown, the discrepancy of the achieved scores using the vision-based HMI is less than the standard gamepad, and this presents the repeatability of the experiments during different sessions and subjects. By the way, it is shown that the achieved scores declined by about 50% by altering the standard HMI to the proposed vision-based HMI. This means the application of the vision-based HMI imposes an overload of about 50% compared with the standard gamepad.

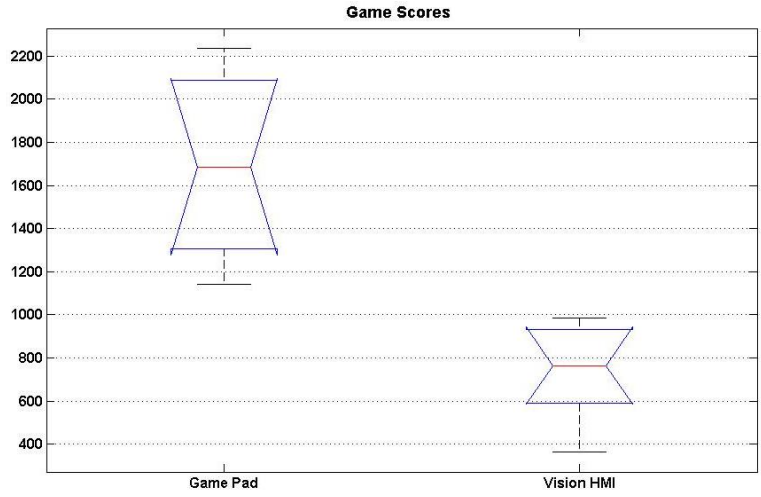


Figure 7. Statistical analysis applied to the comparison of the achieved scores using the standard Xbox gamepad and the vision-based HMI in the first experiment [1].

4.2. Experiment II

The second experiment was to evaluate the improvement made by AKF. A challenging video game that requires quick and precise reactions was considered. The game was developed in our lab and configured in a way so as to ensure that there was no systematic bias between the sessions. To play the game, subjects should use head gestures to drive a car in a route with randomly appearing obstacles. They were able to either move the car forward or bear it to the left/right to avoid hitting obstacles/the border. The car was intended to have a gentle acceleration, and subjects were able to adjust the speed and even to stop it. There was no move backward option. Any safe movement including forward, left and right was rewarded with one positive score, while hitting an obstacle/border was penalized with ten negative scores. The obstacles were set to appear randomly with a uniform distribution, and the subjects had enough pre-training and motivation to achieve a score as high as they could obtain. Error in the vision-based interface leads to lesser scores, and improvement of the vision-based interface raises the scores.

The subjects had to run the game in two sessions: vision-based interface with and without AKF. The sessions had identical settings and a fixed length. According to the identical conditions in the two sessions, the rate of increasing of the achieved scores was considered as a measure of improvement made by AKF. Subjects ran the experiment three times with different orders of the sessions (i.e., with and without AKF). The experiment was conducted by three subjects, and the results are shown in Table 2. The rate of change between scores of the sessions was worked out through (21).

Table 2. Achieved scores while playing the game using vision-based HMI with ($Score_{vHMI\text{-}AKF}$) and without ($Score_{vHMI}$) the adaptive Kalman filter, in the 2nd experiment.

	$\mathbf{Score_{vHMI}}$	$\mathbf{Score_{vHMI\text{-}AKF}}$	$\mathbf{R = \frac{Score_{vHMI\text{-}AKF} - Score_{vHMI}}{Score_{vHMI}}}$
Subject 1			
Session 1	408	453	11%
Session 2	381	430	13%
Session 3	415	445	7%
Subject 2			
Session 1	303	325	7%
Session 2	289	342	18%
Session 3	268	315	18%
Subject 3			
Session 1	389	406	4%
Session 2	410	449	10%
Session 3	395	413	5%
Mean ± STD			**10.3 ± 5%**

Table 2 represents the results of the second experiment. As shown, applying the adaptive Kalman filter to the vision-based HMI has made the first subject's scores improve by about 10%, while the third subject's scores were only improved by 6%. According to the experimental results, the achieved scores in the video game employing the adaptive Kalman filter applied to the vision-based HMI notably improve. The range of improvement is about 4–18% with an average rate of 10%. Figure 8 depicts the result of the statistical analysis (Kruskal-Wallis) applied to the achieved scores by subjects, individually. It shows the 10% significant improvement made by AKF for Subjects 1 and 2.

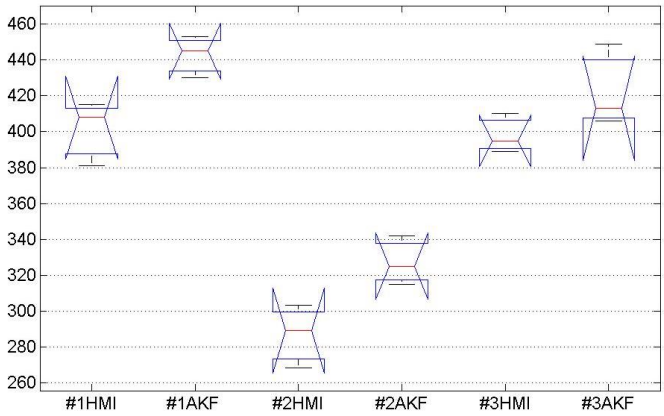


Figure 8. Statistical analysis applied to the achieved scores by subject, individually, in the second experiment. Subjects 1–3 HMI without and with the adaptive Kalman filter (AKF).

4.3. Experiment III

The third experiment is to examine the performance of the proposed HMI in a condition requiring rapid reactions. Therefore, the game is configured in a way that obstacles appear densely and fast. Too many moving obstacles (red boxes) cover the ground, and the player's car (yellow box) should bear left/right to avoid hitting obstacles (Figure 9). Corresponding to bearing left/right commands, the target value is the displacement made by the head gesture in the X-axis direction (dx), which is measured through optical flow. Due to unintended head motions and environmental noise, there are artefacts that should be removed from the output of the HMI. A threshold and low-pass filter were adopted to remove the drifts; meanwhile, AKF is applied to generate the optimum estimation. Figure 10 depicts the outputs of HMI for a period of the game in two modes: without and with AKF. The blue line shows measured displacement in the x-direction, and the red line represents the filtered output. The green lines represent positive or negative scores, caused by either safe driving or hitting obstacles, respectively. It shows that the applied filters remove the artefacts and make the output stable.

To evaluate the performance of the HMI, we used the achieved score by subjects, since it represents the accuracy and response time of the HMI along with the level of the subjects' skill to manipulate the car. To make a comparative study, subjects were asked to repeat the game in two modes: with and without AKF. Table 3 illustrates the results of experiments conducted in nine sessions of the game. It confirms that the HMI with AKF outperforms in most sessions on average with a 5% improvement.

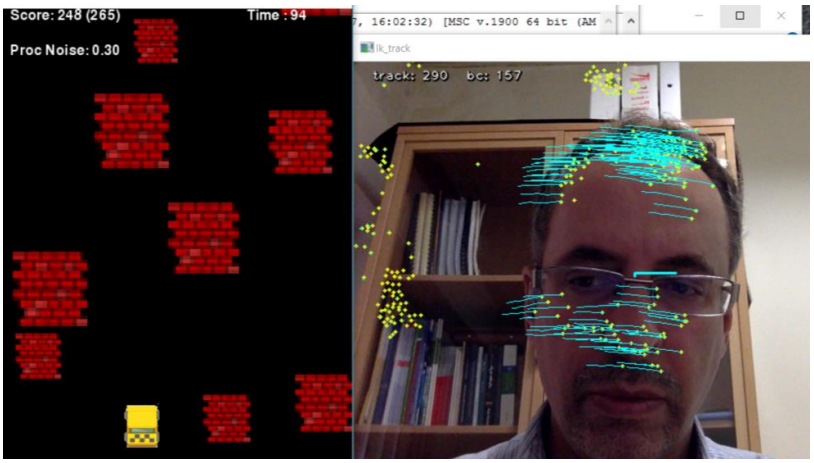
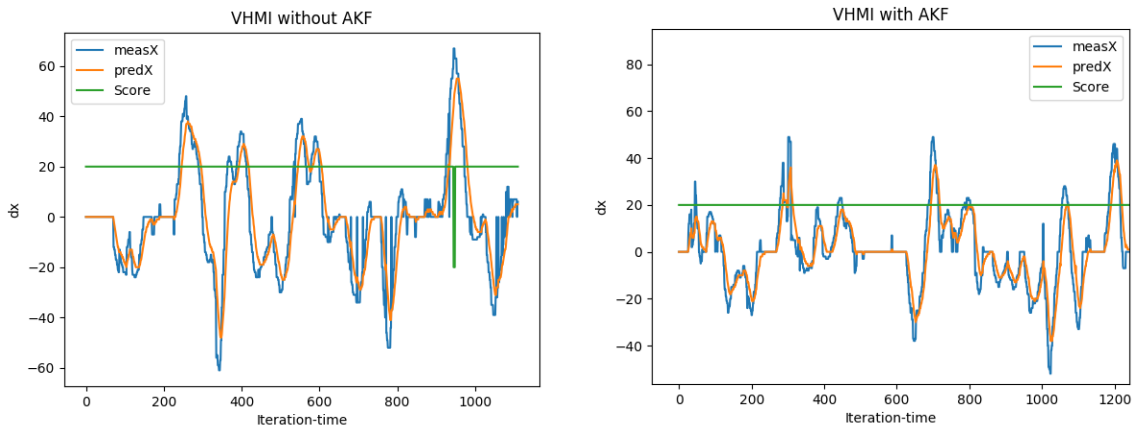


Figure 9. Screenshot of the rapid game, in which dense obstacles (red boxes) require rapid reactions from the player's car (yellow box) in the third experiment.



(**A**) (**B**)

Figure 10. Output of VHMI in the x-direction (*dx*): blue is pure (non-filtered); red is filtered; and green is the achieved scores: (**A**) without AKF; (**B**) with AKF.

Table 3. Achieved scores during rapid games using VHMI with and without AKF, in the 3rd experiment.

Session	$\mathbf{Score_{vHMI}}$	$\mathbf{Score_{vHMI\text{-}AKF}}$	$\mathbf{R = \frac{Score_{vHMI\text{-}AKF} - Score_{vHMI}}{Score_{vHMI}}}$
1	248	281	13.3%
2	189	209	10.6%
3	289	272	−5.9%
4	161	185	14.9%
5	241	237	−1.7%
6	202	189	−6.4%
7	311	346	11.3%
8	266	259	−2.6%
9	197	233	18.3%
Mean ± STD			**6% ± 3%**

5. Conclusions

This paper examines the performance of a vision-based HMI applied to playing video games. The proposed system employs face detection and feature tracking approaches to trace the motions produced by the player's head gesture and applies the adaptive Kalman filter to overcome deficiency raised by occlusion, illumination, etc. The head gestures were recognized and translated into commands for manipulating a car in the video game. We compared the performance of the proposed vision-based HMI with the standard gamepad using the achieved scores in identical games. It is shown that the proposed vision-based HMI is an acceptable alternative for people that are not able to use standard pads, although it imposes an overload of about 50%. The adaptive Kalman filter improves the performance of the proposed HMI and the achieved scores in various conditions. Our future research will focus on fusing the visual and audio data to develop a multimodal HMI with more functionality and reliability.

Acknowledgments: The author would like to thank Hamid Mahmoudabadi for his help in developing the video game in Python.

Conflicts of Interest: The author declares no conflict of interest.

References

1. Oskoei, M.A.; Hu, H. Application of feature tracking in a vision based human machine interface for XBOX. In Proceedings of the 2009 IEEE International Conference on Robotics and Biomimetics, Guilin, China, 19–23 December 2009.
2. Oskoei, M.A.; Hu, H. Myoelectric based virtual joystick applied to electric powered wheelchair. In Proceedings of the IROS 2008, IEEE/RSJ International Conference on Intelligent Robots and Systems, Nice, France, 22–26 September 2008; pp. 2374–2379.
3. Oskoei, M.A.; Hu, H. Adaptive myoelectric control applied to video game. *Biomed. Signal Process. Control* **2015**, *18*, 153–160. [CrossRef]
4. Rautaray, S.S.; Anupam, A. Vision based hand gesture recognition for human computer interaction: A survey. *Artif. Intell. Rev.* **2015**, *43*, 1–54. [CrossRef]
5. Al-Rahayfeh, A.M.E.R.; Faezipour, M.I.A.D. Eye tracking and head movement detection: A state-of-art survey. *IEEE J. Transl. Eng. Health Med.* **2013**, *1*, 2100212. [CrossRef] [PubMed]
6. Zhu, Y.M.; Yang, Z.B.; Yuan, B. Vision based hand gesture recognition. In Proceedings of the 2013 IEEE International Conference on Service Sciences (ICSS), Shenzhen, China, 11–13 April 2013; pp. 260–265.
7. Ernst, T.M.; Prieto, T.E.; Armstrong, B.S.R. Motion Tracking System for Real Time Adaptive Imaging and Spectroscopy. U.S. Patent 9,138,175, 22 September 2015.
8. Chen, H.; Ding, X.P.; Fang, C. Pose robust face tracking by combining view-based AAMs and temporal filters. *Comput. Vis. Image Underst.* **2012**, *116*, 777–792.
9. Kung, S.H.; Zohdy, M.A.; Bouchaffra, D. 3D HMM-based Facial Expression Recognition using Histogram of Oriented Optical Flow. *Trans. Mach. Learn. Artif. Intell.* **2016**, *3*, 42. [CrossRef]
10. Ahmad, A.; Jalil, A.; Ahmed, J.; Iftikhar, M.A.; Hussain, M. Correlation, Kalman filter and adaptive fast mean shift based heuristic approach for robust visual tracking. *Signal Image Video Process.* **2015**, *9*, 1567–1585.
11. Maleki, B.; Ebrahimnezhad, H. Intelligent visual mouse system based on hand pose trajectory recognition in video sequences. *Multimed. Syst.* **2015**, *21*, 581–601. [CrossRef]
12. Khandar, D.G.; Chatur, P. Vision based head movement tracking for mouse control. *Int. J. Adv. Res. Comput. Sci.* **2015**, *6*, 85–87.
13. Jia, P.; Hu, H.; Lu, T.; Yuan, K. Head gesture recognition for hands-free control of an intelligent wheelchair. *J. Ind. Robot* **2007**, *34*. [CrossRef]
14. Lu, P.; Zhang, M.; Zhu, X.; Wang, Y. Head nod and shake recognition based on multi-view model and hidden Markov model. In Proceedings of the International Conference on Computer Graphics, Imaging and Visualization (CGIV'05), Beijing, China, 26–29 July 2005; pp. 61–64.
15. Lu, P.; Huang, X.; Zhu, X.; Wang, Y. Head Gesture recognition based on Bayesian network. In Proceedings of the 2nd Iberian Conference on Pattern Recognition and Image Analysis (IbPRIA'2005), Estoril, Portugal, 7–9 June 2005.
16. Lu, P.; Zeng, X.; Huang, X.; Wang, Y. Navigation in 3D game by Markov model based head pose estimating. In Proceedings of the 3rd International Conference on Image and Graphics (ICIG'2004), Hong Kong, China, 18–20 December 2004.
17. Han, Z.J.; Ye, Q.X.; Jiao, J.B. Online feature evaluation for object tracking using Kalman Filter. In Proceedings of the 19th International Conference on Pattern Recognition (ICPR 2008), Tampa, Florida, 8–11 December 2008; pp. 1–4.
18. Fagiani, C.; Betke, M.; Gips, J. Evaluation of tracking methods for human-computer interaction. In Proceedings of the Sixth IEEE Workshop on Applications of Computer Vision (WACV 2002), Washington, DC, USA, 3–4 December 2002; pp. 121–126.
19. Wang, C.T.; Kim, J.H.; Byun, K.Y.; Ni, J.Q.; Ko, S.J. Robust digital image stabilization using the Kalman filter. *IEEE Trans. Consum. Electron.* **2009**, *55*, 6–14. [CrossRef]
20. Han, B.; Christopher, P.C.; Lu, T.R.; Wu, D.P.; Li, J. Tracking of multiple objects under partial occlusion. In *SPIE Defense, Security, and Sensing*; International Society for Optics and Photonics: Bellingham, WA, USA, 2009; p. 733515.
21. Yang, H.; Alahari, K.; Schmid, C. Occlusion and motion reasoning for long-term tracking. In *European Conference on Computer Vision*; Springer International Publishing: Cham, Switzerland, 2014; pp. 172–187.

22. Mourikis, A.; Roumeliotis, S. A multi-state constraint Kalman filter for vision-aided inertial navigation. In Proceedings of the 2007 IEEE International Conference on Robotics and Automation, Roma, Italy, 10–14 April 2007; pp. 3565–3572.
23. Weng, S.K.; Kuo, C.M.; Tu, S.K. Video object tracking using adaptive Kalman filter. *J. Vis. Commun. Image Represent.* **2006**, *17*, 1190–1208. [CrossRef]
24. Lucas, B.D.; Kanade, T. An iterative image registration technique with an application to stereo vision. In Proceedings of the 7th International Joint Conference on Artificial Intelligence, Vancouver, BC, Canada, 24–28 August 1981.
25. Bouguet, J.Y. Pyramidal Implementation of the Affine Lucas Kanade Feature Tracker Description of the Algorithm. Available online: https://pdfs.semanticscholar.org/aa97/2b40c0f8e20b07e02d1fd320bc7ebadfdfc7.pdf (accessed on 13 November 2017).
26. Bradski, G.; Kaebler, A. *Learning Computer Vision with the OpenCv Library*; O'Reilly Media Inc.: Sebastopol, CA, USA, 2008.
27. DeGroot, M.H.; Schervish, M.J. *Probability and Statistics*; Addison Wesley: Boston, MA, USA, 2002; pp. 589–694.

© 2017 by the author. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0/).