The Tongue-Drive System

Posted on **May 28, 2012** by **James Weber**

(Editor's note: This article will be published in the summer 2012 issue of The Point*, the publication of the Association of Professional Piercers. James Weber the article's author, have given BME permission to publish this article for the continued education of professionals and body art enthusiasts. Enjoy.)*

Late last February a rather curious news story made the rounds on Facebook and other social media sites and pop culture blogs. Various publications[1] reported on an article[2] about a project from Georgia Tech, one that enables a person with quadriplegia to control a wheelchair through the movement of the tongue by moving around a magnet worn in a tongue piercing. Piercers everywhere were sharing, reposting, and reblogging the article in a variety of places—including on my Facebook timeline. Fortunately, this was not news to me, as I've had the unique opportunity to be involved with the project as a consultant for several years. But after a dozen piercers forwarded me the article I realized it was time to write about my experience with the clinical trials of the Tongue Drive System.

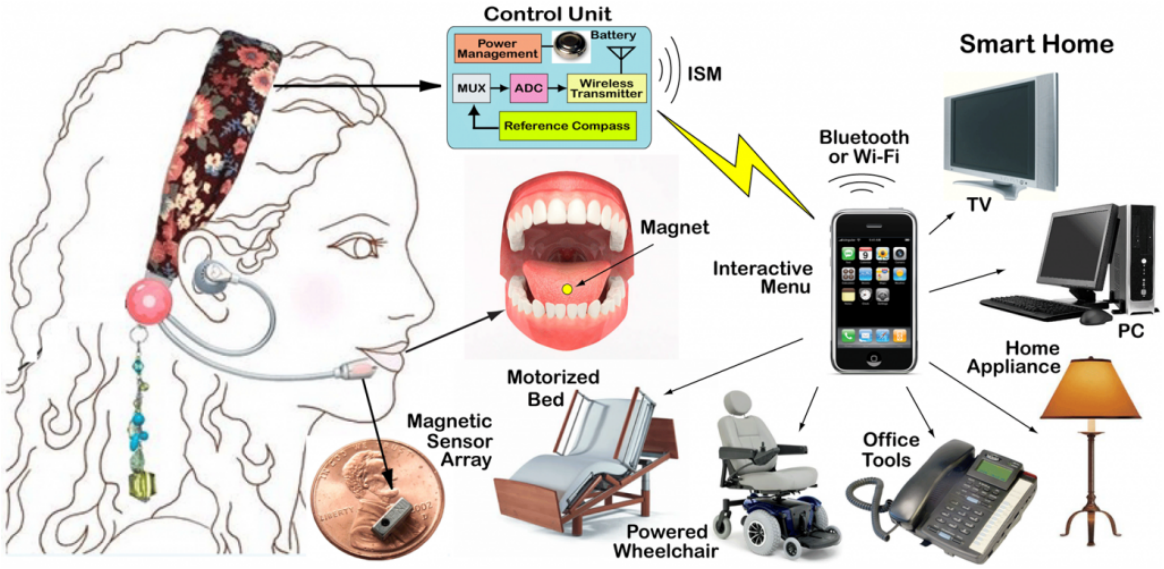
In late October of 2009 I was contacted by Dr. Maysam Ghovanloo, Associate Professor at the School of Electrical and Computer Engineering at the Georgia Institute of Technology. Over the phone he explained the project that he was working on, titled in the research protocol *Development and Translational Assessment of a Tongue-Based Assistive Neuro-Technology for Individuals with Severe Neurological Disorders*. Simply, this is a system that allows persons with quadriplegia to perform a variety of computer-aided tasks—including operating their wheelchairs—by changing the position of a small magnet inside their mouths. The magnet's changing position is monitored by a headpiece that looks like a double-sided, hands-free phone headset.

His team had, at that point, experimented with different ways to attach the magnet to the tongue with varying degrees of success. Adhesives were only effective for very short periods, and the idea of permanently implanting a magnet into the tongue was not considered a workable alternative.[3] This left a third option suggested by Dr. Anne Laumann: attaching a magnet to the tongue with a tongue piercing.

He then came to the reason for his call: he asked if I would be interested in being involved in the clinical trials as a member of the Data Safety Monitoring Board. As I listened to him describe the details of my involvement, I thought about the incredible places my life as a piercer—and my job as an APP Board member—have brought me. I enthusiastically and without hesitation said "Yes!"

(Note: The article is pretty lengthy, so we've put a break here to same some space. Click the Read More button to continue)

For those not familiar with clinical trials (and I was not when I initially agreed to be involved with the study), the Data Safety Monitoring Board (or DSMB, alternately called a Data Monitoring Committee) is a group of experts, independent of the study researchers, who monitor test-subject safety during a clinical trial. The DSMB does this by reviewing the study protocol and evaluating the study data, and will often make recommendations to those in charge of the study concerning the continuation, modification, or termination of the trial. The inclusion of a DSMB is required in studies involving human participants as specified by the Common Rule,[4] which is the baseline standard of ethics by which any government-funded research in the United States must abide. (The clinical trial is sponsored jointly by both the National Science Foundation and the National Institute of Health, but nearly all academic institutions hold their researchers to these statements of rights regardless of funding.)[5]



I was excited to be part of the project, and the following May I received the full details of the study. The clinical trial was to be performed in three phases, with three sets of participants. The first involved ten able-bodied individuals with existing tongue piercings. These participants were to test the hardware and software created by his team and to quantify the ability of those participants to operate the wheelchair with the specially-designed post[6] in their tongue piercing. The second group consisted of ten able-bodied volunteers without tongue piercings. These participants were to be pierced, given time to let the piercings heal, and then monitored operating the Tongue Drive System. The third group of participants was to be a selection of thirty people with quadriplegia—without existing tongue piercings—who were to be pierced and then monitored while the piercing healed. Afterward, they were to be evaluated on their ability to operate a computer and navigate an electric wheelchair through an obstacle course using the magnetic tongue jewelry.

The study was to be conducted in two different locations: in Atlanta, at the Georgia Institute of Technology and the Shepherd Center; and in Chicago, on the Northwestern Medical Center Campus and at the Rehabilitation Institute of Chicago, with half of the participants in each phase of the study coming from each location. (Five from each city for the first two phases, fifteen from each for the last.) Drs. Maysam Ghovanloo and Michael Jones were to oversee the trials in Atlanta, and Drs. Anne Laumann and Elliot Roth were to oversee the trials in Chicago.

The DSMB charter specified the eight people who had been drafted to be part of the DSMB: The board chair is a professor of rehabilitation science and technology; one member is a

director of a rehabilitation engineering research center; one a professor of rehabilitation medicine. There are two M.D.s: one a neurologist; one an associate professor of dermatology; two biostatisticians (one acting as study administrator); and me. Also included in the documents sent was the full study protocol. This document outlined the finer points of the study, including the protocol for tongue piercings to be performed by the doctors involved with the study. The email also specified the possible times of the first meeting of the DSMB, to be conducted via conference call.

As I participated in the conference call several weeks later it was hard not to feel I was out of my element. While I routinely lecture at several local universities, it's been quite a while since I've been in academia. But I soon realized I was not there for my academic credentials but for my position and experience—and as a de facto authority on piercing. This I could do.

During that first meeting I expressed the concerns I had about the piercing protocol, specifically about physicians performing the piercings—physicians with little or no experience doing so. "Do any of the members on the research team have prior piercing experience?" I wrote. "Even though it is not a complicated procedure, it is better for doctors who are involved in this task to have prior experience with tongue piercing."

I was told that the physician overseeing the piercings in Atlanta had performed at least thirty

tongue piercings in his private practice. And although Dr. Laumann—who was responsible for the tongue piercings in Chicago—had no prior piercing experience, she had conducted extensive research on piercing and tattooing[7] and had often observed professional piercers at work. (Furthermore, she is considered an expert among dermatologists in the field of piercing and tattooing.) While my concerns were addressed, I do remember feeling hesitant at the close of that meeting.

The second DSMB meeting was held six months later, in December of 2010. At this time the results of the first and second phases of the clinical trial were to be discussed. Before the meeting I was given information about the second study group and about the tongue piercing method performed at the Chicago location—and including images from both locations. From the images provided, I was concerned that the piercings performed by the physicians looked as if they were done by first-year piercing apprentices—which, in a way, they were.

Of the twenty-one study participants who received a tongue piercing, five were noted as complaining about the placement of the piercing, and three piercings resulted in embedded jewelry. Based on the photos I guessed this was because either the piercing had been placed too far back on the tongue or the length for initial jewelry was improper—or both. I pointed out to the committee this left only about 60% of the subjects who were both comfortable with the placement of the piercing (at least enough to not state the contrary to researchers) and who did not have problems with embedded jewelry. I stated I thought this was far too small a percentage to ensure the well-being of each research participant. Even though it was outside my role as a DSMB member, I further stated the results of the study may be affected by the improperly placed piercings, as more than a few of the study participants had taken out their jewelry and dropped out of the study within a few days of being pierced, saying they were either unhappy with the placement or found the position of the piercing uncomfortable.[8]

I went on to express concerns about the piercing protocols and to question whether piercers could perform these procedures instead of physicians. Unfortunately, I was told the parameters of the study, and the rules at the medical centers where the piercings were being performed, did not allow non-medical professionals to perform the piercing procedures.[9]

Despite my concerns, my suggestions and criticisms were well-received. Dr. Ghovanloo agreed to re-evaluate the piercing protocol and I offered him whatever help he needed. Most importantly, I got the impression the two doctors performing the piercings were somewhat humbled by the experience. While there was no doubt that these physicians have anatomical knowledge and surgical experience that far surpasses mine, they were quickly realizing this didn't make them proficient piercers.

Several months after that conference call, I had the opportunity to finally meet Dr. Ghovanloo in person. The quarterly meeting of the APP's board of directors was scheduled in Atlanta in February of 2010, and Dr. Ghovanloo arranged for me to meet some of the trial staff at the Shepherd Center. I had the sense he was excited as well, and he also arranged for the physician doing the piercings during the clinical trials in Atlanta to be there: Dr. Arthur Simon. As I was at a board meeting with Elayne Angel (the APP's then-Medical Liaison, current President, and resident expert on tongue piercings), I asked about having her attend as well. He readily agreed.

When Elayne and I arrived we were greeted by Shepherd staff member and study coordinator Erica Sutton, and we were soon led to our meeting with Dr. Ghovanloo and Dr. Simon. Compared to the necessary formality of the DSMB meetings, it was a friendly and relaxed meeting. Dr. Ghovanloo and his colleagues were somewhat starstruck by Elayne (she often does that to people) especially since her book, *The Piercing Bible*, was used so extensively in drafting the trial piercing protocols.

As we talked about the clinical trials, it was hard to not be affected by Dr. Ghovanloo's enthusiasm for the project. We spoke at length about the issues the doctors encountered when performing the piercings. Doctor Simon in particular was humbled after his experience. "How do you hold those little balls to screw on?" he asked at one point during the several hours we met, a little exasperated and only half joking. I can't speak for Elayne, but I left with an immense respect for Dr. Ghovanloo, his staff, and the whole project. I also left with the impression that they had a lot more knowledge of—and a little more respect for—what we do as well.

Since that time, stage three of the clinical trials has already taken place. I've been informed by Dr. Ghovanloo that the third and final meeting of the DSMB will be scheduled in the coming weeks. In fact, trials are being planned using a new prototype that allows users to wear a

dental retainer on the roof of their mouth embedded with sensors to control the system (instead of the headset),[10] with the signals from these sensors wirelessly transmitted to an iPod or iPhone. Software installed on the iPod then determines the relative position of the magnet with respect to the array of sensors in real time, and this information is used to control the movements of a computer cursor or a powered wheelchair.

I'm looking forward to hearing when the project is out of the trial phase and more widely available to all who can use it. When that happens, I'm sure I'll be hearing from Dr. Ghovanloo —and seeing the news again posted on Facebook.

More information about the current trials can be found on the Shepherd Center's web site: http://www.shepherdcentermagazine.org/q3_11/#/feature2/

Links:

1 http://www.wired.co.uk/news/archive/2012-02/21/tongue-drive-system, http://news.discovery.com/tech/tongue-drives-wheelchair-120222.html#mkcpgn=rssnws1, http://boingboing.net/2012/02/24/tongue-piercing-steers-wheelch.html

2 http://www.gatech.edu/newsroom/release.html?nid=110351

3 Unlike implants under the skin, the tongue has no "pockets" in which to encase a foreign object, and there was also concern about the need to remove the magnet for surgeries and MRIs.

4 http://www.hhs.gov/ohrp/humansubjects/commonrule/index.html

5 The history of research ethics in the country is simultaneously fascinating and shameful. Most of the modern rules now in place concerning clinical trials in the U.S. are as a result of the public outcry over the Tuskegee Syphilis Experiment, a study that ran for four decades, from 1932 and 1972, in Tuskegee, Alabama. This clinical trial was conducted by the U.S. Public Health Service and was set up to study untreated syphilis in poor, rural black men who thought they were receiving free health care from the U.S. government. The study was terminated only after an article in the New York Times brought it to the attention of the public. More information about the history of research ethics can be found here: http://research.unlv.edu/ORI-HSR/history-ethics.htm

6 In one of my early conversations with Dr. Ghovanloo I gave him the name of several manufacturers who I thought would be willing and/or able to make the jewelry needed for the trials. Barry Blanchard from Anatometal came through by manufacturing special barbells with a magnet encased in a laser-welded titanium ball fixed on top. Blue Mountain Steel also donated the barbells and piercing supplies for the initial piercings.

7 Dr. Laumann has co-written several published papers on body piercing and tattooing. The most recent is titled, "Body Piercing: Complications and Prevention of Health Risks."

8 Dr. Ghovanloo and the other physicians had suggestions for the reasons for the high dropout rate among healthy subjects. In response to an early draft of this article, he wrote, "We simply lost contact with a few subjects after piercing, and cannot say for sure what their motivation was in participating in the trial and consequently dropping out after receiving the piercing." Dr. Laumann, commenting on the Chicago site, wrote, "We prescreened thirty-two volunteers. Ten of these were screened and consented. Three of these were ineligible due to a short lingual frenulum, or 'tongue web.' This would have made the use of the TDS impracticable and for research it would have been considered inappropriate to cut the lingual frenulum. We pierced seven subjects and—you are correct—our first subject dropped out related to embedding of the jewelry and pain on the first day. After that we were careful to measure the thickness of the tongue and insert a barbell that allowed for 6.35 mm (1/4 inch) of swelling. Otherwise drop-outs came much later during the TDS testing phase related to scheduling and unrelated medical issues. One of the subjects, a piercer herself, was particularly pleased with the procedure, the tract placement and the appearance."

9 Though the protocols did not allow the procedure to be conducted by non-medical personnel, Gigi Gits, from Kolo, was present during one of the phase-two health subject's piercings and Bethra Szumski, from Virtue and Vice, was able to offer advice at the first phase-three piercing session in Atlanta.

10 Dr. Laumann: "The problem with headgear is that it needs to be removed at night, which means that the disabled individual cannot do anything in the morning until the headset is replaced and the TDS recalibrated. With secure intra-oral sensors, recalibration will not be necessary in the morning, nor will the sensors slip during use, which gives the wearer a great degree of independence. Of course, a dental retainer takes up space in the mouth and this may be difficult with a barbell in place."

This entry was posted in **ModBlog** and tagged **APP**, **Guest Column** by **James Weber**. Bookmark the **permalink [http://news.bme.com/2012/05/28/the-tongue-drive-system/]** .

About James Weber

James Weber is a professional body piercer. He started piercing professionally in 1993 and has been actively involved at an industry-wide level in body piercing education, legislation, and public relations projects for almost two decades. He is the founder and owner Infinite Body Piercing, Inc. Started in 1994, it is now one of the oldest—and busiest—piercing studios in the United States. James served two terms on the Board of Directors of the Association of Professional Piercers: his first as Medical Liaison, from 2005 until 2008; his second as President, from 2008 until 2011. He has overseen The Point: The Journal of the Professional Piercers as editor for seven years and twenty-seven issues.
View all posts by James Weber →

ONE THOUGHT ON "THE TONGUE-DRIVE SYSTEM"

SolidOak
on **June 1, 2012 at 3:27 pm** said:

Amazing!

Integrated Computer-Aided Engineering 18 (2011) 391–400
DOI 10.3233/ICA-2011-0384
IOS Press

A low-cost 3D human interface device using GPU-based optical flow algorithms

Rafael del Riego[a], José Otero[b] and José Ranilla[b,*]
[a]*CRAZYBITS STUDIOS, Avenida Argentina 132, Gijón, Spain*
[b]*Department of Computer Science, University of Oviedo, Campus de Viesques S/N, Gijón, Spain*

Abstract. Except for a few cases, nowadays it is very common to find a camera embedded in a consumer grade laptop, notebook, mobile internet device (MID), mobile phone or handheld game console. Some of them also have a Graphic Processing Unit (GPU) to handle 3D graphics and other related tasks. This trend will probably continue in the next future and the pair camera+GPU will be more and more frequent in the market. Because of this, the proposal of this work is to use these resources in order to build a low-cost software-based 3D Human Interface Device (3D HID) able to run in this kind of devices, in real time without degrading the overall performance. This is achieved implementing a parallel version of an existing Optical Flow Algorithm that runs fully in the GPU without using it at full power. In this way, usual graphic processes coexist with Optical Flow computations. To the best of author's knowledge, this approach (a software-based 3D HID that runs fully in a GPU) is not found in academic research nor in commercial products prototypes. Indeed, this is the salient contribution of this paper. The performance of the proposal is good enough to achieve real time in low grade computers.

Keywords: Virtual 3D-HID, optical flow algorithms, GPGPU

1. Introduction

Personal computers, laptops, notebooks or smartphones have an integrated low-resolution camera (a back high-resolution camera in the case of smartphones and in some other cases a frontal low resolution one). Besides, most of them have a graphic processing unit (GPU), a specialized processor that offloads 3D/2D graphics rendering from the microprocessor. Consequently, a new paradigm that uses the GPU has arisen [28]. This concept turns the massive floating-point computational power of modern graphics accelerators into a general-purpose computing power. In certain applications, this allows us to increase the performance in some orders of magnitude compared to a conventional CPU [26,28].

Being aware of these capabilities developers try to use these resources to build new Human Interface Devices (HIDs). This feature is also a trend in the gaming industry. Indeed, it is a subset of "motion gaming" concept that includes Nintendo Wii,[1] Sony PlayStation Move and Eye/Eye Toy[2] or Microsoft Kinect,[3] dedicated platforms that aim to produce seamless gaming experience [6].

Other approaches include accelerometers available in the gaming device itself and/or adding a video projector in order to use any available planar surface as screen. This is the case of Moject (Mo-tion and pro-ject portmanteau) by Innovation in Mobile Technologies,[4] which attaches a pico-projector to an iPhone, or Microvision pico-projector video game gun-controller.[5]

A common approach is using a multi-touch surface in order to provide a suitable sensor to give a real feeling of interaction [25].

There are also software solutions that add motion sensitivity using the on board camera, with no extra

*Corresponding author: Dr. J. Ranilla, Department of Computer Science, University of Oviedo, Campus de Viesques S/N, Gijón, Spain, E33204. E-mail: ranilla@uniovi.es.

[1] http://wii.com/.
[2] http://playstation.com
[3] http://www.xbox.com.
[4] http://moject.com.
[5] http://www.microvision.com.

ISSN 1069-2509/11/$27.50

hardware needed but degrading the performance of the device, loading the processor with extra computations like, for example, the proposal of Engine Software for Nintendo Dsi.[6]

All this examples fall in the 3D Human Interface Device (3D-HID) category and in the Vision-based motion estimation with mobile devices [10]. Speaking in general, Computer Vision based HIDs proposals are common from gesture detection [5,7,20] to eye tracking algorithms [19]. Surveillance systems [13] bear some relationship with the techniques in this paper because gestures or posture have sometimes to be detected but cannot be considered HID.

In this work, a virtual 3D-HID similar to the previously commented approach for Nintendo DSi is proposed but with a distinctive feature (no extra hardware is needed): it uses the GPU to unload the CPU with motion-estimation computations and avoiding accelerometers, gyroscopes, and special cameras. To the best of author's knowledge, in the context of 3D-HID, there is no other work to date that uses the GPU of a device to estimate the motion of the device itself without diminishing the graphic performance of the system and being completely unobtrusive to the user.

Apart from the GPU usage, the proposed approach can be considered an estimation of egomotion [2] (depth from motion dual problem [16]), more specifically, a qualitative estimation of camera motion from Optical Flow (OF) [23] using a parallel implementation [8,9] in real time but without specialized hardware [15]. Real time is the most important goal because it has direct impact in usability [14] other than 30 frames per second performance would add an unacceptable delay to user experience.

In order to point out the aim of this paper, its main aspects are going to be reviewed. Thus, in Section 2 explains the optical flow algorithms. Section 3 is devoted to the built system. The experimental results are showed in Section 4 and finally, Section 5 summarizes the conclusions.

2. Optical flow

Optical Flow (OF) is the 2D vector field projection of the 3D velocities of object points. In Fig. 1 a pair of frames of a classic test sequence is shown with the true optical flow over imposed. As can be seen, the motion of the objects in the scene is well represented by the optical flow.

[6] http://www.engine-software.com/.

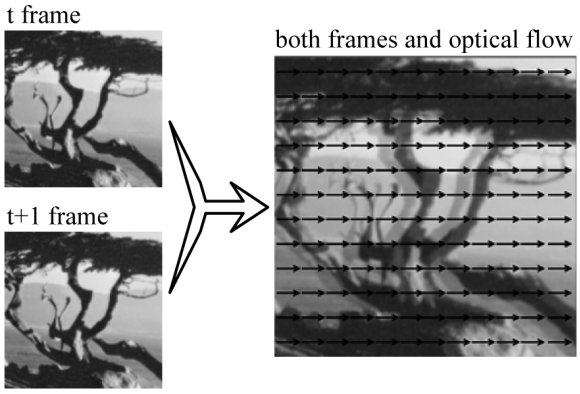


Fig. 1. OF example, from two frames (left) the motion of the scene is measured for each pixel. The resulting vector field (black arrows) is shown in the right side of the figure. Both frames are overimposed.

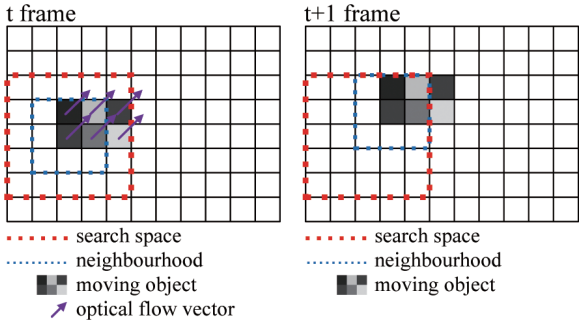


Fig. 2. OF computation using a block-matching algorithm. The neighborhood in the first frame (blue dotted square) is found in the second frame in different position. This displacement defines the OF vector for each pixel.

In the literature, OF algorithms are classified as follows: in correlation based techniques, in frequency based techniques and gradient based techniques.

Correlation based techniques or block matching algorithms (BMA) [1] try to maximize a measure of similarity between patches (taken from two consecutive frames) centered in a given pixel. The displacement that maximizes the selected measure divided by the time interval within the acquisition of the frames is the velocity of the pixel (see Fig. 2).

Frequency based techniques use a set of tuned spatiotemporal filters to search for the velocity of a pixel [11].

Gradient based techniques use the well known Optical Flow Constraint (OFC) shown in Eq. (1) in order to compute the OF [4]. In that equation, f is the gray level of a pixel, u and v are the components of that pixel OF in x and y axis, respectively.

$$-\frac{\partial f}{\partial t} = \frac{\partial f}{dx}\frac{dx}{dt} + \frac{\partial f}{\partial y}\frac{dy}{dt} = \frac{\partial f}{\partial x}u + \frac{\partial f}{\partial y}v \qquad (1)$$

Equation (1) makes the assumption that intensity changes (the spatiotemporal derivatives) in a sequence of images are only due to the movement of the objects in the scene: a single pixel will have constant brightness in the different positions that it takes during the sequence. Nonetheless, the Aperture Problem [17] states that there is no way to recover the complete OF vector using only local (one pixel) information, because Eq. (1) has two unknowns.

Some authors try to solve the aperture problem with the incorporation of some kind of global information involving a process of regularization [4].

Others perform a clustering of the OFCs themselves in order to find the most reliable one. Once obtained this, the corresponding normal flow to that OFC is achieved [18].

Another alternative is to analyze the measurements in the space of the velocities, that is, performing an estimation of the velocity with the results of many systems of OFC equations. Each system of equations is obtained from at least one pair of pixels in order to estimate the velocity. In this way, the analysis is performed directly in the domain of the data to be recovered, that is, the u, v space. Examples of this alternative are Otero et al. algorithm [21,22], Hierarchical Lucas-Kanade algorithm [6] or Schunck's proposal [24].

Recent approaches exist to improve OF algorithms results. For instance, in [27] procedures for the detection of areas where objects with different velocity overlap are proposed. Later, in this areas, the OF is improved using several diffusion filters.

Unfortunately, the OF accuracy improvement leads, obviously, to an increasing computational cost of the whole system. Recall that the aim of this paper is to achieve real time performance, because of this, approaches like this are discouraged.

In this paper, the approach in [24] was chosen as test bed, in order to ascertain if the execution speed up of an OF algorithm parallel implementation using a GPU is enough to achieve real time performance of the whole system. This approach has been chosen instead of the one in [21] because of the parallel version poor performance of this OF algorithm. This was found during the development of the system and it is not discussed here for simplicity's sake.

2.1. Hierarchical Lucas-Kanade algorithm

Hierarchical Lucas-Kanade (HLK) is composed of two procedures: an iterative one and a hierarchical one. The iterative part is very straightforward: arising

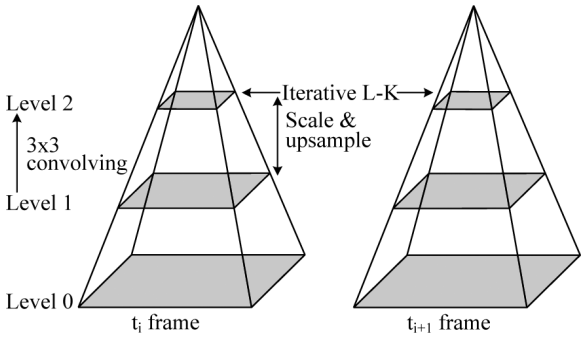


Fig. 3. How Hierarchical Lucas-Kanade algorithm works.

from OFC, the following equations Eq. (2) are obtained (see [6] for details):

$$G \doteq \sum_{x=p_x-w_x}^{p_x+w_x} \sum_{y=p_y-w_y}^{p_y+w_y} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

$$\bar{b} \doteq \sum_{x=p_x-w_x}^{p_x+w_x} \sum_{y=p_y-w_y}^{p_y+w_y} \begin{bmatrix} \delta I I_x \\ \delta I I_y \end{bmatrix} \tag{2}$$

$$\delta \bar{v}_{opt} = G^{-1} \bar{b}$$

where δI, I_x, I_y are spatiotemporal derivatives and w_x, w_y are integers that define the neighborhood size for the pixel at (p_x, p_y).

The goal is to find a motion vector that minimizes the error function in the following equation Eq. (3).

$$\varepsilon(\bar{v}_{opt}) = \sum_{x=p_x-w_x}^{p_x+w_x} \sum_{y=p_y-w_y}^{p_y+w_y} (A(x,y) - B(x+v_x, y+v_y))^2 \tag{3}$$

where A and B are usually termed the "first" and "second" images and v_x, v_y are the components of $\bar{v}_{opt}$.

The previous estimation Eq. (2) fails when the motion of the pixels is large. In order to refine the motion estimation, an iterative procedure is used: each movement estimation $\bar{v}_{opt}$ is used to warp the first frame making it closer to the second frame and an estimation of the *residual* of the motion vector is computed using again the previous equations with the transformed image as input. The procedure ends when the residual falls below a threshold (warped first frame very close to second frame) or the number of iterations is reached.

The hierarchical procedure uses what is called a "pyramid of images" (see Fig. 3): a set of different sized versions of an image. In particular, every "level" of the pyramid is an image constructed by halving the dimensions of the image in the level below starting

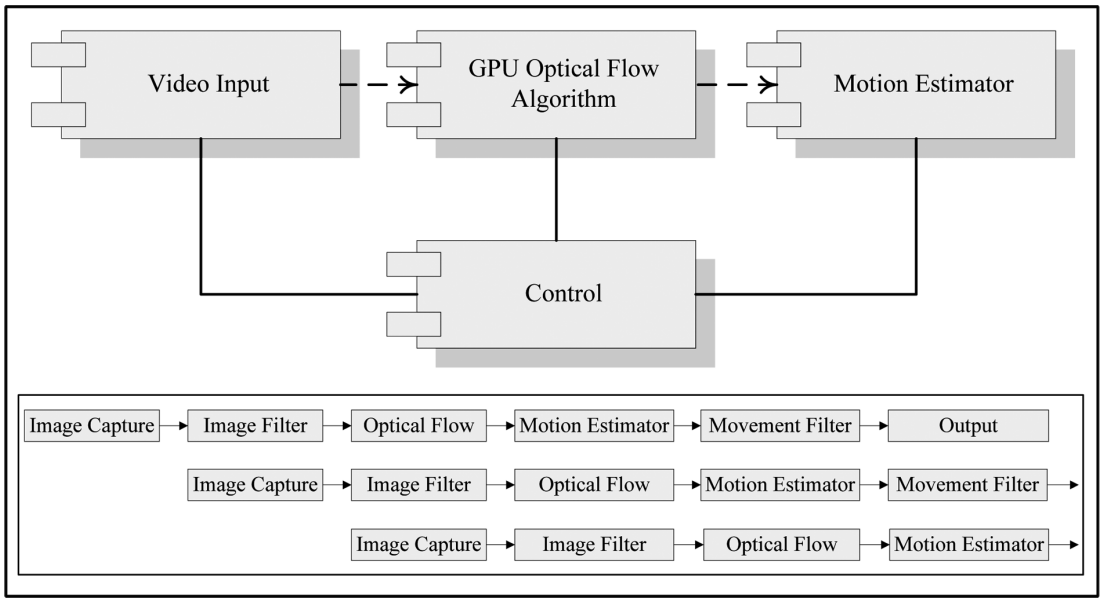


Fig. 4. Architecture of the proposed system.

from the original image in the first level. As the pixels of the image need to be sampled, smoothing of the input images is needed. The number of pixels at each level is four times lesser. The value of each pixel in a level is calculated convolving the level below using a 3 *by* 3 Gaussian kernel. Effectively, this smooths the image removing noise from the input. Moreover, the separability of this filtering benefits the parallelization of the whole system.

The other consequence is that the measurement of a movement in a level will be half the one in the level below. That means that the movement value is halved in every level. However as the same window size (in Lucas-Kanade algorithm) is applied, the search space is virtually doubled and it is now possible to track greater movements. This is important for a 2D-HID, otherwise, the user should self impose a maximum speed for his/her movements.

One thing to keep in mind is that as the images are smaller in each level and some points "vanish", it is also needed to remap the points where the motion will be estimated, exactly the opposite of the pyramidal decomposition.

At each frame, the process starts at the summit of the pyramid, making rough approximations of the movements. To accomplish these, the iterative process is applied at each level. The result of each level (multiplied by two) serves as starting point for the iterative procedure at the next level, until the base of the pyramid is reached, i.e. the full input image, is reached. The process ends when the last iteration is done at the base level, resulting in the final estimation of the motion.

3. Proposed system

The system presented in this work was designed with flexibility in mind, that is, it could be used to solve the proposed problem or to any other task that could be separated into pieces that inter-operate chained together in a lineal work-flow. Of course, non-lineal tasks could be adapted to fit into this design, although it is not the goal of this work.

The philosophy is not new; it has its background related to computer architecture designs, as it mimics the segmentation technique, common nowadays in the CPUs. Every step of the work-flow can be separated into tasks that are fitted into a "Segmentation Unit". In CPUs, the processing of every instruction is divided into small parts and then executed in specific components of the CPU. Thus, when an instruction has ended using the "Instruction Fetch" component, the next instruction can use it, having two instructions processed at the same time.

According to this philosophy, the system architecture is divided in four main subsystems: Video Input, Optical Flow Algorithm, Motion Estimator and Control. Each subsystem comprises several basic computing units (see Fig. 4). The "Segmentation Units" of the system are threads executed in parallel on the CPU

making a complete analogy with the CPU architecture. Of course, flaws and benefits of this framework are similar to those of CPU segmentation. Thus, good load balancing among pipelines is desired. It is up to the developers to know how the load of their work is best balanced.

As seen on Fig. 4, the use of this framework creates a work-flow that transforms an input image (provided it is not the first one of the sequence as, at least, two images are required in order to estimate the OF) into a set of values which quantify the motion estimated.

Other images are processed at the same time. This can be achieved even with a one-core CPU because the proposal makes the most of the combination CPU plus GPU.

The system delivers the heavier calculus of the production chain to the GPU. Thus, it releases the CPU from other tasks such as retrieving images from the input (from camera or from file during off-line testing), doing other minor calculations or lighter tasks as delivering the final result to the output.

At this point, it is easy to see that the proposed framework has two levels of parallelism. On the one hand, several functional units are working concurrently (including CPU and GPU) based on the pipelined technique. On the other hand, the OF computation unit runs inside the GPU exploiting data parallelism.

Each of these tasks is linked to the next by means of a buffer that lies between two "Segmentation Units". A buffer, as usually, is a place where information is stored to be used later. They manage the way the system behaves. Given the nature of the work, a ring buffer that could stop (i.e. put to sleep the internal threads, saving CPU time) the "Segmentation Units" when it has no data from the input buffer (or when it's output buffer is full) is an adequate solution.

Every "Segmentation Unit" executes several processes, so several logic parts can be processed in the same thread preventing overload due to thread management. It saves time as makes the most of the multi-core CPU capabilities without forcing the developer to know the internal mechanisms of threading.

The "Segmentation Units", which are "Control Units" of the processes itself, are in turn managed by a meta-controller unit, the Flow Controller. The users of the developed framework should emphasize on using this controller and other that can be obtained through this: the "Data Container". It acts as storage for data shared among Segmentation Units and can be used to pass some bits of info to the process and get info back from the process.

The following subsections provide details on the way the OF algorithm is parallelized (3.1 and 3.2) and how the output of the OF algorithm is evaluated (3.3).

3.1. Optical flow and the GPU

One of the core aspects of this work is the use of a heavily parallelized architecture, as the modern GPUs are, to gain extra computational power without degrading the performance of the CPU. This hardware is common nowadays in every PC. Even recent mobile devices present powerful GPUs capable to deliver these extra GFLOPs.

As seen in Section 2, algorithms based on the OF are computationally expensive. For example, in the case of a BMA algorithm, for a $n \times m$ pixels image, a search space of $p \times q$ pixels and a neighborhood of $s \times t$ pixels, the number of floating point operations is approximately $n \times m \times p \times q \times s \times t$. Because of this, a parallel implementation of HLK algorithm that runs on the GPU has been developed.

One key concept is that, due to the nature of each algorithm, not all of them are well suited for running efficient on GPUs. Recall that in this paper HLK [6] algorithm was chosen as a test bed. The exhaustive parallel implementation of the algorithms in [3] is an overwhelming task beyond the scope of this paper.

Finally, bear in mind that the *numerical* results of the parallelized algorithms are identical to the conventional implementations. Therefore, there is no need to compare the new implementations and the classic ones found in [3] in terms of numerical error. The salient aim of this paper is to achieve real time performance not an improvement on the OF accuracy.

3.2. CUDA implementations details

An API extension to the C programming language named CUDA (Compute Unified Device Architecture) is used to encode the algorithms that will be run on the GPU. CUDA, released by NVIDIA,[7] allows specified functions from a ordinary C program to run on compatible GPU's stream processors. This makes C programs capable of taking advantage of a GPU's ability to operate over substantial volume of data in parallel while still making use of the CPU when appropriate.

At first, the input image is loaded onto a CUDA "texture" which offers the improvements to manipulate im-

[7] http://www.nvidia.com.

ages such as interpolate between pixels needed for the algorithm. The next step after having the images loaded is building current image structure (i.e. pyramidal structure in case of HLK). Except for the first one, the following images and their structures are already loaded into memory as they were computed in the previous frames. When needed, the images are split into blocks that are rescaled independently (i.e. to form the next pyramid level in HLK).

According to CUDA documentation and the observed behavior during the development of the proposed system, one of the drawbacks of using graphics accelerators to improve computing performance is data loading process into (and back from) the GPU's memory because its high cost. The decision was to load the images as soon as possible and deliver to the GPU all the image-related calculus, as the CUDA offers some improvements to manipulate images. It is possible, due to the system architecture, to overlap communications (copy images from/to PC's memory to/from GUP's memory) and computation (floating point calculus on the GPU), that is, the movement of data between memories is made in asynchronous way to reduce its impact.

The next step involves the point filtering method to remove some unnecessary calculus for the next steps. As the calculus involves "reducing" the data to those that gather the useful information, some threads of the GPU are put to sleep while others gather the partial results. Finally, only one thread gathers the final result. Note that is very important not to get into thread synchronization too often writing to the same variable, as it incurs into performance penalty.

The core and final step of the GPU part of the global process ends with the estimation of the OF. As explained before, when parallelizing the point filter, the threads will operate independently among several parts of the image computing partial results of the matrices needed. Then, only some threads gather some partial results before only one thread summarizes all and retrieves the final real result. This is done at first for the needed matrices of the method (see [6]).

If the window size for the calculus is smaller enough to not overlap between different points where the OF will be estimated another optimization for the process is obtained. This is because it is not necessary to compute the spatial derivatives for the full image but only on the pixels in the window of each point which will be much lesser. The system used 240 points in an equally-distributed grid for 640 $\times$ 480 pixel images, with window sizes of 7 $\times$ 7 and 9 $\times$ 9. Thus, giving 11760 or 19440 pixels respectively against 307200 needed for the entire image, which represent a great performance gain. Therefore, the process is separated into parts of the image that are processed independently by replicated components of the GPU.

As CUDA abstracts some of the multithreading mechanisms implementation from the developer, only how the OF is estimated for one point has to be defined. The rest of the points are processed in parallel using the same code and sharing resources. Algorithm 1 shows the computational part of the CUDA's kernel to filter the points, where several points (as many as cores the GPU has) are processed at the same time (each point is automatically identified by local, threadIdx.x, blockDim.x, blockDim.y and blockIndex variables).

Algorithm 1. CUDA kernel.

```
__global__ void filterPoints(parameters){
...
  // Compute the differences for the pixels
  for (i = x; i < x + ratio; ++i)
    eT[local] = fabsf(tex2D(tex, i, y) -
                tex2D(texPre, i, y));
  __syncthreads();
  // Reduce the sum: first reduce each row
  if (threadIdx.x == 0) {
    for (i = local+1; i < window.width; ++i)
      eT[local] += eT[i];
  }
  __syncthreads();
  // Reduce the columns, hence get the total
  if (threadIdx.x == 0 && threadIdx.y == 0) {
    for (i = blockDim.x; i < blockDim.y;
                          i+=blockDim.x)
      eT[local] += eT[i];
    if (eT[local] < thresholdPoints)
      velocities[blockIndex] = CUDAPoint2D();
  }
  __syncthreads();
}
```

3.3. Motion estimators

Optical flow field estimation is used in order to measure/detect the motion in the image sequence acquired with the cameras. The goal is to use the translations in X, Y and Z along with rotation in Z (see Fig. 5) as input signals to the proposed virtual interface.

In order to discriminate the predominant motion in the scene, the following operators are used:

– X and Y translations are measured as the average OF in the image:

$$(X, Y)_T = \frac{\sum_{i=0}^{i=n} \sum_{j=0}^{j=m} (v_x, v_y)_{i,j}}{nm} \qquad (4)$$

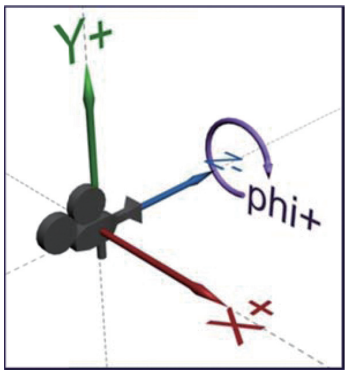


Fig. 5. Rotations and displacements to measure.

where $(X, Y)_T$ is the translation vector, $(v_x, v_y)_{i,j}$ is the OF vector for pixel (i, j), n, m are the number of rows and columns in the image.

– Z translation is measured with the divergence of the OF averaged across the image:

$$Z_T = \frac{\sum_{i=0}^{i=n} \sum_{j=0}^{j=m} \nabla \cdot (v_x, v_y)_{i,j}}{nm} \tag{5}$$

Equation (5) is evaluated and averaged for each OF value across the whole image.

– Z rotation is measured with the rotational of the OF averaged across the image:

$$Z_T = \frac{\sum_{i=0}^{i=n} \sum_{j=0}^{j=m} \nabla \times (v_x, v_y)_{i,j}}{nm} \tag{6}$$

The output of the OF algorithm is evaluated with the previous operators. The highest output defines the predominant motion in the scene.

All this calculations are done in a set of points uniformly distributed across the image for which their position is calculated by a simple internal algorithm. Given the desired number of points to accomplish the process, these points are always in the same position for the same image size. There is also another reason to proceed this way: the system does not focus on the movement of the elements of the scene but on the movement of the scene itself which is closer to the goal of the system.

Not all the points might be reliable to proceed. Points such as large uniform areas or points with minor changes between frames do not offer reliable information of the OF unless a too large window size is used. In order to prevent these points from been processed in a later stage of the process, a filter which removes them for the next step at each frame is proposed. It compares the value of the image intensity at each pixel and its neighborhood against a threshold Eq. (7), and then discards the points that do not fulfill the condition. What is more, the time inverted filtering is recovered as time saved in the heavier part of the calculus because some points are filtered before the main part of the process. This is summarized in Eq. (7), where $I(p_x, p_y)$ is image intensity at (p_x, p_y), I_{max} is the intensity maximum in the neighborhood and T is a threshold.

$$\sum_{x=p_x-w_x}^{p_x+w_x} \sum_{y=p_y-w_y}^{p_y+w_y} I(p_x, p_y) \leqslant \sum_{x=p_x-w_x}^{p_x+w_x} \sum_{y=p_y-w_y}^{p_y+w_y} T \cdot I_{max}(p_x, p_y) \tag{7}$$

In order to remove pixels with small changes between frames, for every point and its corresponding neighborhood the difference in its intensity with the previous image is computed, added up and compared against a threshold made using the input value specified. Lowering it will have more points passing the filter.

Low illumination leads to noise in the image and to the false detection of small movements due to the noisy pixels that appear and disappear. It was decided to filter the OF vectors that are below a threshold in order to minimize that error.

Table 1
Equipments used to evaluate the performance

Desktop (EQ1)	CPU Intel Core2 Quad Q9550 2.83 GHz
	RAM 4 GB
	GPU nVidia GeForce 9500 GT
	Web cam 1: Logitech Quickcam E2500
Laptop (EQ2)	CPU AMD Athlon 64 3000+ (1.8 GHz)
	RAM 1024 MB
	GPU nVidia GeForce 9500M
	Web cam 1: Logitech Quickcam E2500
	Web cam 2: Logitech Webcam C200

4. Experiments

Several experiments were done in order to test de proposed system with the chosen OF algorithm. Also, two equipments are used to evaluate the performance of the system (see Table 1). Note that in both equipments, the maximum frame rate is restricted by the camera quality or by the usb-connection speed.

One kind of experiments was made in real time with the webcams provided by the equipments in order to test the functionality of the system as a software-based 3D Human Interface Device. First, the behavior of the system was tested when the element endowed with motion is the webcam (see Fig. 6). Second, some tests

Fig. 6. Several frames extracted from a test sequence moving a webcam in an indoor environment. Top: camera moving from bottom to top. Shown frames span over 0.48 seconds. Bottom: camera moving from left to right. Shown frames span over 0.32 seconds.

Fig. 7. Several frames extracted from a test sequence with a subject moving in front of a web cam. The webcam is placed at right bottom corner. *Exactly* the same software is running. Only left to right movement is shown, vertical movements are detected but they are too small (as the jumps of the subject are) to be noticed printed. See the online video for details.

were made moving the hand/body of a person in front of the webcam (see Fig. 7). An URL[8] with small videos of this kind of experiment is provided, because (obviously) this kind of information is not well suited for a printed and static media. During the experiments it was found that translations are well detected.

In order to quantify the performance of the proposed system, and test if basic movements are detected correctly, making quantitative measurements, some additional tests were made. Thus, some synthetic videos of scenes were generated with a well defined, easy to perceive movement in an axis: X, Y, Z and the rotation along Z axis. In each video only one kind of movement was present. To make it more complete 50 frames per video are provided. The motion has constant velocity to avoid errors or noise. This procedure allows customizing the videos greatly adjusting the movement to the one to be tested.

In Fig. 8 some frames of synthetic videos are showed. Each one corresponds to a specific movement.

The system detects easily the motion in different axis. Some errors may appear only when velocity module falls below a threshold. This could happen with real images mainly due to image acquisition and illumination issues (flickering or low illumination).

[8] http://di002.edv.uniovi.es/~jotero/Video_3DInterface.avi.

During the experiments, it was found that X and Y translations are easily detected and measured. Z translation is only detected because small displacements in X or Y directions (by the user) lead to relatively high values compared with the divergence of the OF field. To avoid this, before divergence estimation using Eq. (5), the OF field is scaled by a suitable constant. For the other motion detectors, the OF field is kept unchanged. The amount of motion cannot be accurately measured but if a threshold is used it can simulate a click.

Rotation in Z axis is another kind of movement which is useful as input signal (click alike) in the virtual interface. System behavior is summarized in Table 2.

The performance in terms of number of frames per second processed was also tested along with computing power consumption. Using CUDA implementation the frame rate increases to 30 fps (the maximum frame rate of the camera or the usb-connection speed) in both systems. Moreover, standard GPU tasks are not degraded and the CPU can be fully dedicated to other tasks. Thus, the user experience is real-time alike. The results obtained for these experiments are shown in Table 3.

On the one hand, equipment EQ1 always gets 30 frames per second (fps) regardless of the usage of the GPU. However, the percentage of CPU usage is lower when the GPU is used as general-purpose computing device. On the other hand, EQ2 obtains 17 fps when

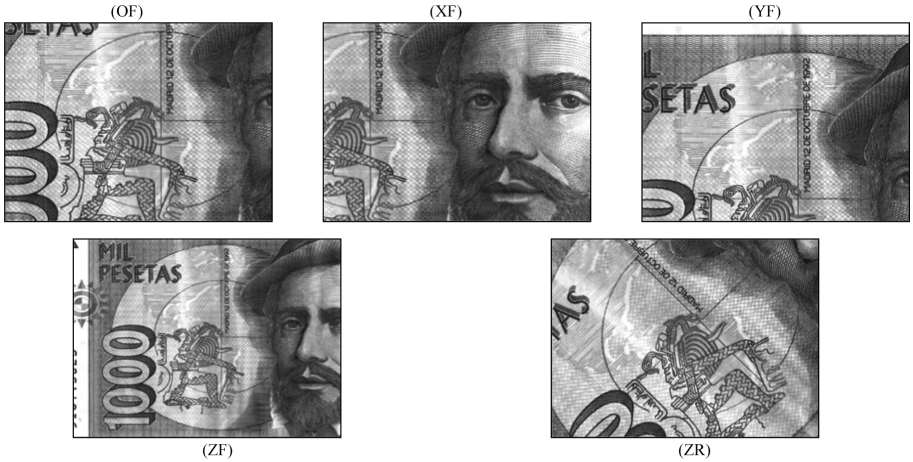


Fig. 8. Some frames of synthetic videos, each one corresponding to a specific movement. From top to bottom and left to right, original frame (OF), X (XF), Y (YF), and Z (ZF) translations and rotation around Z (ZR) are shown.

Table 2
System response summary

Movement	System Response
X	Measured / Useful
Y	Measured / Useful
Z	Detected / Useful
Z Rot	Detected / Useful

Table 3
Performance achieved by the system with/without using the GPU. fps: Frames per second

	EQ1	EQ2
fps without using the GPU	30	17
% usage of the CPU	medium	high
fps using the GPU as computing device	30	30
% usage of the CPU	low	low

the GPU is not used, and the performance of other tasks is degraded (high percentage of CPU usage).

5. Conclusions

In this work is shown how to build a Virtual 3D Human Interface Device by using standard resources of the current computers: the integrated camera (or Web cam) and the graphic processing unit (GPU). The system applies OF techniques and uses CUDA to exploit the capabilities of the GPU as stream processor. This approach keeps the CPU of the computer fully available to other tasks and their performance is not degraded. The presented solution is simple and clean since no extra hardware is needed and relies only on software.

A HLK algorithm parallel version was designed and implemented to build the system together with adding a filtering stage to the original algorithms that avoids processing points in large blank areas or with minor changes between frames.

Using the proposed approach, the frame rate and resolution is now limited only by the characteristics of the camera. Because of this, no delay is added and user experience is real-time. Alike, computer's performance is not being degraded without the need of powerful/additional hardware.

As stated before, some issues exist regarding with the Z axis. One possible solution to address this is following the approach in [23]. Probably this will lead to accuracy *vs.* parallel efficiency trade-off: perhaps the most accurate OF algorithm is not well suited for a parallel implementation.

Acknowledgements

This work is supported by the Spanish Office of Science and FEDER, under the grants TIN2007-61273, TIN2008-06681-C06-04 and TIN2010-14971.

References

[1] P. Anandan, A Computational Framework and an Algorithm for the Measurement of Visual Motion, *International Journal of Computer Vision* **2** (1989), 283–310.
[2] X. Armangué, H. Araújo and J. Salvi, A Review on Egomotion by Means of Differential Epipolar Geometry Applied to the Movement of a Mobile Robot, *Pattern Recognition* **36**(12) (2003), 2927–2944.
[3] J.L. Barron, D.J. Fleet and S.S. Beauchemin, Performance of Optical Flow Techniques, *International Journal of Computer Vision* **12**(1) (1994), 43–77.

[4] B.K.P. Horn and B.G. Schunck, 1992. Determining Optical Flow, in: *Shape Recovery*, L.B. Wolff, S.A. Shafer and G.E. Healey, eds, Jones and Bartlett Publishers, Inc., USA, pp. 389-407.

[5] A. Besinger, T. Sztynda, S. Lal, C. Duthoit, J. Agbinya, B. Jap, D. Eager and G. Dissanayake, Optical Flow Based Analyses to Detect Emotion from Human Facial Image Data, Expert Systems with Applications, Volume 37, Issue 12, December 2010, Pages 8897-8902, ISSN 0957-4174, DOI: 10.1016/j.eswa.2010.05.063.

[6] J.-Y. Bouguet, Pyramidal Implementation of the Lucas-Kanade Feature Tracker, Description of the Algorithm, Technical report, Intel Corporation, Research Labs, 1994.

[7] L. Chern-Sheng, H. Chien-Wa, C. Kai-Chieh, S.-S. Hung, H.-J. Shei and M.-S. Yeh, A Novel Device for Head Gesture Measurement System in Combination with Eye-controlled Human-machine Interface, *Optics and Lasers in Engineering* **44**(6) (2006), 597–614.

[8] J. Díaz, E. Rosa, R. Agísa and J. Luis Bernier, Super-pipelined High-performance optical-flow Computation Architecture, *Computer Vision and Image Understanding* **112**(3) (2008), 262–273.

[9] M. Fleury, A.F. Clark and A.C. Downton, Evaluating Optical-flow Algorithms on a Parallel Machine, *Image and Vision Computing* **19**(3) (2001), 131–143.

[10] J. Hannuksel, P. Sangia and J. Heikkilä, Vision-based Motion Estimation for Interaction with Mobile Devices, *Computer Vision and Image Understanding* **108**(1–2) (2007), 188–195.

[11] D.J. Heeger, Optical Flow using Spatiotemporal Filters, *International Journal of Computer Vision* **1**(4) (1988), 279–302.

[12] J. Loviscach, Playing with All Senses: Human–Computer Interface Devices for Games, *Advances in Computers* **77** (2009), 79–115.

[13] Y. Liu and C. Pomalaza-Ráez, 2010. On-chip Body Posture Detection for Medical Care Applications using Low-cost CMOS cameras, *Integrated Computer-Aided Engineering* **17**(1) (January 2010), 3–13.

[14] C. Manresa-Yee, P. Ponsa, J. Varona and F.J. Perales, User Experience to Improve the Usability of a Vision Term-based Interface, Interacting with Computers, *Now Available* **22**(6) (2010), 594–605.

[15] J.L. Martín, A. Zuloaga, C. Cuadrado, J. Lázaro and U. Bidarte, Hardware Implementation of Optical Flow Constraint Equation using FPGAs, *Computer Vision and Image Understanding* **98**(3) (2005), 462–490.

[16] O. Millnert, T. Goedemé, T. Tuytelaars, L. Van Gool, A. Hüntemann and M. Nuttin, Range Determination for Mobile Robots using an Omidirectional Camera, *Integrated Computer-Aided Engineering* **14**(1) (2007), 63–72.

[17] D.W. Murray and B.F. Buxton, Experiments in the Machine Interpretation of Visual Motion, MIT Press, Cambridge, USA, 1990.

[18] P. Nesi, A. del Bimbo and D.B. Tzvi, Robust Algorithm for Optical Flow Estimation, Journal on Computer Vision, *Graphics and Image Processing: Image Understanding* **61**(2) (1995), 59–68.

[19] B. Noureddin, P.D. Lawrence and C.F. Man, A Non-contact Device for Tracking Gaze in a Human Computer Interface, *Computer Vision and Image Understanding* **98**(1) (2005), 52–82.

[20] R.G. O'Hagan, A. Zelinsky and S. Rougeaux, Visual Gesture Interfaces for Virtual Environments, *Interacting with Computers* **14**(3) (2002), 231–250.

[21] J. Otero, A. Otero and L. Sánchez, Mode Based Hierarchical Optical Flow Estimation, *Machine Graphics & Vision* **10**(4) (2001), 489–501.

[22] J. Otero, A. Otero and L. Sánchez, 3D Motion Estimation of Bubbles of Gas in Fluid Glass, Using an Optical Flow Gradient Technique Extended to a Third Dimension, *Machine Vision and Applications* **14**(3) (2003), 185–191.

[23] P. Sang-Cheol, L. Hyoung-Suk and L. Seong-Whan, Qualitative Estimation of Camera Motion Parameters from the Linear Composition of Optical Flow, *Pattern Recognition* **37**(4) (2004), 767–779.

[24] B.G. Schunck, Image Flow Segmentation and Estimation by Constraint Line and Clustering, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **11**(10) (1989), 1010–1027.

[25] D. Valkov, F. Steinicke, G. Bruder and K.H. Hinrichs, 2010. A Multi-Touch Enabled Human-Transporter Metaphor for Virtual 3D Traveling. In Proceedings of the 2010 IEEE Symposium on 3D User Interfaces, 79–82, IEEE Press.

[26] V. Volkov and J.W. Demmel, 2008, Benchmarking GPUs to tune dense linear algebra. In Proceedings of the 2008 ACM/IEEE conference on Supercomputing (SC '08), 1–11, IEEE Press, Piscataway, NJ, USA.

[27] S. Viet-Uyen Ha and J.W. Jeon, Readjusting Unstable Regions to Improve the Quality of High Accuracy Optical Flow, *IEEE Transactions on Circuits and Systems for Video Technology* **20**(4) (2010), 540–547.

[28] J.D. Owens, M. Houston, D. Luebke, S. Green, J.E. Stone and J.C. Phillips, GPU Computing, *Proceedings of the IEEE* **96**(5) (2008), 879–899.

Head Pose Estimation and Augmented Reality Tracking: An Integrated System and Evaluation for Monitoring Driver Awareness

Erik Murphy-Chutorian, *Member, IEEE*, and Mohan Manubhai Trivedi, *Fellow, IEEE*

***Abstract*—Driver distraction and inattention are prominent causes of automotive collisions. To enable driver-assistance systems to address these problems, we require new sensing approaches to infer a driver's focus of attention. In this paper, we present a new procedure for static head-pose estimation and a new algorithm for visual 3-D tracking. They are integrated into the novel real-time (30 fps) system for measuring the position and orientation of a driver's head. This system consists of three interconnected modules that detect the driver's head, provide initial estimates of the head's pose, and continuously track its position and orientation in six degrees of freedom. The head-detection module consists of an array of Haar-wavelet Adaboost cascades. The initial pose estimation module employs localized gradient orientation (LGO) histograms as input to support vector regressors (SVRs). The tracking module provides a fine estimate of the 3-D motion of the head using a new appearance-based particle filter for 3-D model tracking in an augmented reality environment. We describe our implementation that utilizes OpenGL-optimized graphics hardware to efficiently compute particle samples in real time. To demonstrate the suitability of this system for real driving situations, we provide a comprehensive evaluation with drivers of varying ages, race, and sex spanning daytime and nighttime conditions. To quantitatively measure the accuracy of system, we compare its estimation results to a marker-based cinematic motion-capture system installed in the automotive testbed.**

***Index Terms*—Active safety, graphics programming units, head pose estimation, human-computer interface, intelligent driver assistance, performance metrics and evaluation, real-time machine vision, support vector classifiers, 3-D face models and tracking.**

I. Introduction

VEHICULAR safety relies on the ability of people to maintain constant awareness of the environment as they drive. As new vehicles and obstacles move into the vicinity of the car, a driver must be cognizant of the change and be ready to respond as necessary. Although people have an astounding ability to cope with these changes, a driver is fundamentally limited by the field of view that he can observe at any one time. When a driver fails to notice a change to his environment, there is an increased potential for a life-threatening collision. It is reasonable to assume that this danger could be mitigated if the driver is notified when these situations arise. As evidence to this effect, a recent comprehensive survey on automotive collisions demonstrated that a driver was 31% less likely to cause an injury-related collision when he had one or more passengers who could alert him to unseen hazards [1]. Consequently, there is great potential for driver-assistance systems that act as virtual passengers, alerting the driver to potential dangers through aural or visual cues [2]. To design such a system in a manner that is neither distracting nor bothersome, these systems must act like real passengers, alerting the driver only in situations where he appears to be unaware of the possible hazard. This requires a context-aware system that simultaneously monitors the environment and actively interprets the behavior of the driver. By fusing information from inside and outside the vehicle, automotive systems can better model the circumstances that motivate driver behavior [3], [4].

With consideration for future driver assistance systems, we concentrate on one of the integral processes for monitoring driver awareness: estimation of the position and orientation of a driver's head. Head pose is a strong indicator of a driver's field of view and current focus of attention. It is intrinsically linked to visual *gaze estimation*, which is the ability to characterize the direction in which a person is looking [5], [6]. Intuitively, it might seem that looking at the driver's eyes might provide a better estimate of gaze direction, but in the case of lane-change intent prediction, for example, head dynamics were shown to be a more reliable cue [7]. In addition, implementing a vision system that focuses on a driver's eyes is impractical at many levels. In addition to the economic and technical challenges of integrating and calibrating multiple high-resolution cameras placed throughout the cabin (to view the eye from all head positions), it requires that the driver's eyes be visible at all times (e.g., sunglasses or other eye-occluding objects would cause the system to malfunction). Furthermore, we believe that the eyes can convey only the gaze direction relative to the direction of the head. Physiological studies demonstrate that this is clearly the case for human perception [8], and computational eye trackers typically require the subject to maintain a frontal head pose.

Manuscript received September 22, 2007; revised January 19, 2008, July 20, 2009, November 18, 2009, and January 6, 2010; accepted January 15, 2010. Date of publication April 5, 2010; date of current version May 25, 2010. The Associate Editor for this paper was Q. Ji.

E. Murphy-Chutorian was with the Computer Vision and Robotics Research Laboratory, Department of Electrical and Computer Engineering, University of California, San Diego, La Jolla, CA 92093 USA. He is now with Google Inc., Mountain View, CA 94043 USA (e-mail: erikmc@ucsd.edu; erikmc@google.com).

M. M. Trivedi is with the Computer Vision and Robotics Research Laboratory, Department of Electrical and Computer Engineering, University of California, San Diego, La Jolla, CA 92093 USA (e-mail: mtrivedi@ucsd.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TITS.2010.2044241

Computational head pose estimation remains a challenging vision problem, and there are no solutions that are both inexpensive and widely available. Among the research thrusts and commercial offerings that can provide a real-time estimate of head pose, most require multiple cameras to obtain a correspondence-based depth information, and none have been rigorously and quantitatively evaluated in an automobile. In a car, ever-shifting lighting conditions cause heavy shadows and illumination changes, and as a result, techniques that demonstrate high proficiency in stable lighting often will not work in these situations.

The novelty of this paper is threefold: First, we introduce a new procedure for head pose estimation and a new algorithm for 3-D head tracking. Second, we provide a systematic implementation of these two to create a hybrid head-pose estimation system. In this computational system, we only use a single video camera and provide a real-time (30 fps) implementation by optimizing the calculations for the parallel processors available on a consumer graphic processor. Third, we quantitatively demonstrate the success of this system on the road, comparing our markerless monocular head-pose estimator to ground truth obtained with a professional cinematic motion-capture system that we have configured for a vehicular testbed. To ensure a wide variety of driving conditions, we perform these experiments with drivers of varying age, race, and sex spanning daytime and nighttime drives.

In designing our system, we strove for a cost-efficient prototype that could be reasonably adapted for widely deployed automobiles. Although our prototype has been implemented in a full-size PC, the current evolution of embedded processors (and embedded graphics processors) would be the natural progression for the future of this technology. Our system was designed to meet the following design criteria:

1) Monocular: The system must be able to estimate head pose from a single camera. Although accuracy might be improved with stereo imagery, multiple cameras increase the cost and complexity of the system, and they require manual calibration that can drift as a result of vibrations and impacts.
2) Autonomous: There should be no manual initialization, and the system should operate without any human intervention. This criterion precludes the use of pure-tracking approaches that measure the relative head pose with respect to some initial configuration.
3) Fast: The system must be able to estimate a continuous range of head pose while driving, with real-time (30 fps) operation.
4) Identity and Lighting Invariant: The system must work across different drivers in varying lighting conditions.

II. Prior Work

Recently, there has been a great interest in driver-assistance systems that use computer vision technology to develop safer automobiles [3]. Within this scope, a large area of focus has been to direct cameras inside the vehicle and interpret the driver's state from video observations.

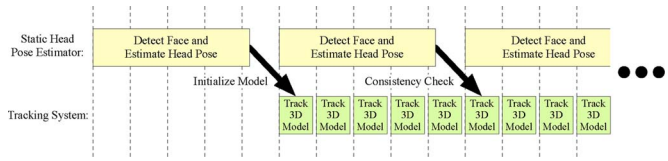


Fig. 1. Our hybrid head-pose-estimation scheme combines a static head-pose estimator with a real-time 3-D model-based tracking system. The static estimator initializes the tracker and provides periodic consistency checks as the two operations run in parallel.

In one prime example, the driver's eye closure blink frequency, nodding frequency, and 2-D face position have been used to estimate driver attentiveness [9]. This system uses infrared illuminators and Kalman filters to track the driver's pupil and a fuzzy classifier to provide an overall estimate of attentiveness. In another system, the driver's eyes and lip corners were initialized with color predicates and tracked in relation to a bounding box around the driver's head [10]. This was shown to provide an estimate of the driver's gaze as well as to estimate the driver attentiveness level using finite-state machines.

Driver head-motion estimation has also been used along with video-based lane detection and CAN bus data to predict the driver's intent to change lanes in advance of the actual movement of the vehicle [11]. This paper supplied these cues to a sparse Bayesian learning classifier that provides a probabilistic prediction of a lane change seconds in advance.

All these previous works use a coarse estimate of head motion as the input to a classifier that estimates an aspect of the driver's intent. In contrast, our system provides a fine absolute measure of the driver's head position that can directly be used to indicate the driver's focus of attention. As a result, we have put great attention into ensuring that the system is robust to variations in lighting and driver's appearance, and we have evaluated the accuracy of this system in varying conditions.

Our contributions in this paper also include new algorithms for head pose estimation and tracking, and we present a review of prior works in this area.

Our system is a hybrid approach that combines the initialization and stability properties of a static pose estimator with the highly accurate, jitter-free, and real-time capabilities of a tracking approach. The static estimator initializes the tracker from a single image frame and, as the head is tracked, continues to run in parallel, providing a periodic consistency check. If the tracking confidence falls below a threshold or the consistency check fails, then the static estimator automatically reinitializes the tracker. This process is illustrated in Fig. 1. Although very different in composition and scope, other works have espoused the advantages of hybrid systems [12]–[15].

The static head-pose estimator that we have developed is a nonlinear regression technique that directly estimates the head pose from a detected image patch. Nonlinear regression approaches provide continuous estimates of pose and have the some of the highest reported success in indoor environments [12]. Prior work in this area includes locally linear maps [16], multilayer perceptrons [17], and principal component analysis (PCA) projection with support vector regression [18]. From our experience with nonlinear regressors, we have observed that the most significant problem with these approaches is their

sensitivity to localization error. With noisy face localization (as is common with computational face detectors), the accuracy of these approaches diminishes. In our investigations, we found that we can mitigate this problem by using localized gradient orientation (LGO) histograms as the input to nonlinear regressors. These histograms provide explicit invariance to face localization error, as well as added invariance to lighting and appearance variation. In this paper, we provide experimental evaluation of the improvement in pose estimation by extracting these histograms.

Unlike static head pose estimation techniques, head tracking approaches operate on continuous video, estimating head pose by inferring the change in pose between consecutive frames of a video sequence. These approaches exploit the temporal continuity and smooth motion constraints to provide a jitter-free estimate of pose over time. These systems typically demonstrate much higher levels of accuracy than static pose estimation methods, but they require initialization from a known head position and are prone to drifting and losing track. Our system is an example of a top-down tracking approach that finds a global transformation that best accounts for the observed motion between video frames. With stereo imagery, for instance, the head pose can also be obtained with affine transformations by finding the translation and rotation that minimize the discrepancy in grayscale intensity and depth [14]. In addition to finding the transformation that minimizes the appearance between the model and the new camera frame, systems can also incorporate prior information about the dynamics of the head. Particle filters provide an approximation of the optimal track by maximizing the posterior probability of the movement from a simulated set of samples. Variations on particle filtering have been applied to head accurate real-time head-pose tracking in varying environments, including near-field video [19], low-resolution video with adaptive PCA subspaces [20], and near-field stereo with affine approximations [21], [22]. In this paper, we introduce a new dual-state particle filter to explicitly model the nonlinear motion of a driver. This motion model is able to simultaneously account for the observed jitter of a driver's head and the driver's intentional head movements. Compared with other particle tracking approaches, we have overcome many simplifications and limitations such that we have the following.

1) We only require monocular video, satisfying our design criterion and preventing the need for periodic stereo calibration.
2) We compute full projective transformations of the model, rather than affine approximations, improving performance by removing an artificial source of distortion.
3) We use a full textured-mapped 3-D model instead of a series of point samples, allowing a more complete comparison between the model and the observation.
4) We provide a real-time implementation, satisfying our design criterion for 30-fps tracking.

The system that we present in this paper is a novel software engine that advances the state of the art in fully autonomous head-pose estimation. This system has practical utility for many applications including intelligent meeting spaces [23], and in this paper, we focus our efforts on the automotive domain.

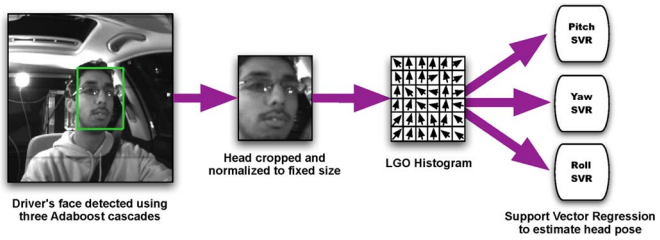


Fig. 2. Overview of our static head-pose-estimation procedure consisting of three steps: 1) The head is detected with a trio of cascaded Adaboost detectors. 2) An LGO histogram is extracted from the cropped head region. 3) The histogram is passed to SVRs for pitch, yaw, and roll.

To demonstrate the capacity of our system, we evaluated it on a wide range of natural driving situations with a cinematic motion-capture system providing a quantitative comparison. Although there have been other head-pose-estimation systems that have been applied to automotive imagery [24]–[29], they have been evaluated when the car is moving in specific scenarios only in situations where the car is not moving (indoors). It is unclear whether these approaches would require substantial modification to become viable options for real automotive use. In contrast, the data collection and evaluation that we have conducted in this paper are the first their kind, and we are able to demonstrate that our system attains a high level of accuracy during real-world driving.

The remainder of this paper is structured as follows: Section III details our methods for head detection and static head-pose estimation. Section IV introduces our augmented-reality head-tracking algorithm. Section V describes our hybrid head pose system and the real-time implementation of the tracker using optimized consumer-grade graphics hardware. Section VI introduces our automotive testbed and presents an evaluation of our methods. Section VII contains our concluding remarks.

III. Face Detection and Head-Pose Estimation

In the first stage of our system, we compute an initial estimate of the driver's head position and orientation. This consists of the following three steps.

1) A facial region is found using three cascaded Adaboost [30] face detectors applied to the grayscale video images.
2) The detected facial region is scale normalized to a fixed size and used to compute an LGO histogram.
3) The histogram is passed to three support vector regressors (SVRs) trained for head pitch, yaw, and roll.

A graphical overview of this procedure is presented in Fig. 2. It is run once to initialize the tracker and periodically repeated to check the consistency of the tracking estimate. In this following paragraphs, we describe these steps in more detail.

A. Facial Region Detector

To detect the location of the driver's head, we use three Adaboost cascades attuned to the left profile, frontal, and right profile faces [30], [31]. Each detector is capable of recognizing heads with enough deviation from its characteristic pose that when combined, they span the range of head poses in our

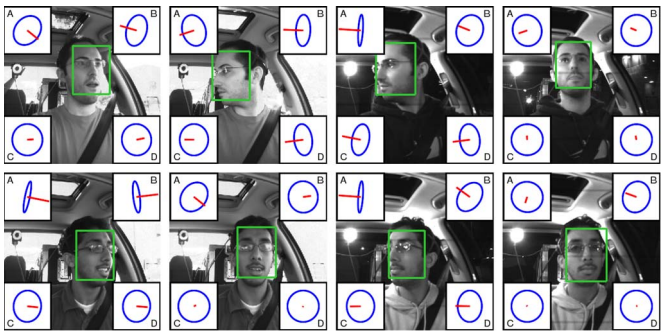


Fig. 3. Comparison of static head-pose estimation using the following methods: (A) NCC prototype matching. (B) Gradient PCA with support vector regression [18]. (C) LGO histograms with support vector regression. (D) Vicon motion capture ground truth. The center rectangle indicates the detected facial region using a trio of cascaded Adaboost face detectors, and the pose for each method is indicated by the direction of the thumbtack.

training data: -30° to 30° in pitch and roll -90° to 90° in yaw. For both training and testing, an uncompressed grayscale image is used as the input to the detectors, and we consider the largest detected rectangular region as the location of the driver's face. To ensure that the static pose estimation process is invariant to scale, every region is down sampled to a fixed size of 34×34 pixels. In an automobile, this makes the system invariant to the distance between the driver and the camera. In our experiments, this facial-detection scheme successfully detected a region in approximately 90% of the video frames. For the remaining frames, the initialization or consistency check is simply skipped until the next successful detection. No effort was made to prune false detections, although one could envision a production system with heuristics based on size, position, and color. From our experience with these detectors in driving video, false detections are quite rare, but when they do occur, the pose estimates are clearly incorrect until the next successful face detection. The detection examples are illustrated in Fig. 3.

B. LGO Histogram

To provide a robust description of each facial region, we compute the LGO histogram. A fixed-size version of this representation was first presented as part of the scale-invariant feature transform [32], which is intended for correspondence matching between regions surrounding scale- and rotation-invariant keypoints. It is a compact feature representation that is robust to minor deviations in region alignment, lighting, and shape [32], [33]. This is useful for automatic head pose estimation, since the explicit position invariance of the histogram offsets some of the localization error from the face detector. Additionally, the histogram is invariant to affine lighting changes, and the gradient operation emphasizes edge contours that are less influenced by identity than image texture. The merit of the generalized histogram has been demonstrated for human detection, where it has alternatively been called a histogram of oriented gradients [34]. In contrast to object recognition systems that represent an object as a configuration of multiple histogram descriptors [32], we use a single LGO histogram to represent the entire scale-normalized facial region. This descriptor consists of a 3-D histogram. The first two dimensions correspond to the vertical and horizontal positions in the image and the third to the gradient orientation. For an $M \times N \times O$ histogram, let the triplet (m, n, o) denote a specific bin in the histogram. The horizontal and vertical image gradients $\boldsymbol{X}_x(x, y)$ and $\boldsymbol{X}_y(x, y)$ are approximated by filtering with 3×3 pixel Sobel kernels. The image is then split into $M \times N$ discrete blocks, and for each pixel (x, y) in the (m, n) block, the absolute gradient orientation $o_{x,y}$ is quantized into one of O discrete levels

$$o_{x,y} = \left\lfloor O \times \left(\frac{1}{2\pi} \operatorname{atan2} \left(\boldsymbol{X}_y(x, y), \boldsymbol{X}_x(x, y) \right) + 0.5 \right) \right\rfloor \quad (1)$$

and used to increment the $(m, n, o_{x,y})$ histogram bin. After computing the histogram, it is smoothed with the $3 \times 3 \times 3$ kernel as

$$K(m, n, o) = \left(1 - \frac{g(m)}{M} \right) \left(1 - \frac{g(n)}{N} \right) \left(1 - \frac{g(o)}{O} \right) \quad (2)$$

to prevent aliasing effects, where $\{m, n, o \in \mathcal{B}\}$ for $\mathcal{B} = \{-1, 0, 1\}$, and $g(\cdot)$ is the complement impulse function

$$g(\lambda) = \begin{cases} 0, & \text{if } \lambda = 0 \\ 1, & \text{if } \lambda \neq 0. \end{cases} \quad (3)$$

The resulting *soft* histogram is subsequently reshaped and normalized to a unit vector. Finally, as suggested by Lowe [32], any vector component greater than 0.2 is truncated to 0.2, and the vector is renormalized if necessary. In our system, we use a 128-D vector, where $M = 4$, $N = 4$, and $O = 8$.

C. Support Vector Regression

To estimate the pose of the driver's head, we use support vector regression on the LGO histogram inputs. Support vector regression is a supervised learning technique for the nonlinear regression of a scalar function [35], [36].

An optimized software package [37] was used to train our system with radial basis function kernels. We generated three regressors trained for head pitch, yaw, and roll. The input to each is the LGO histogram described in Section III-B. To find the optimum regression parameters, we scale normalize each component of the input data such that it spans the range $[-1, 1]$ and then perform a cross-validation grid search across the parameter space. During testing, we use the same scaling parameters to normalize the new input before predicting the new pose.

IV. Head-Pose Tracking in Augmented Reality

We introduce a new procedure to track the driver's head in six degrees of freedom at 30 fps from a single video camera. Our approach uses an appearance-based particle filter in an *augmented reality*, which is a virtual environment that mimics the view space of a real camera [23]. Using an initial estimate of the head position and orientation, the system generates a texture-mapped 3-D model of the head from the most recent video image and places it into the environment. The model is subsequently rotated, translated, and rendered in perspective projection to match the view from each subsequent video frame. It would be computationally inefficient to exhaustively

search for the best transformation, so instead, we introduce an appearance-based particle filter framework to generate a set of virtual samples that together provide an optimal estimate of this transformation. The virtual samples are perspective projections of the head model at a specific rotation and translation and resemble small perturbations of the driver's face set against a solid background.

Although the 3-D construction and evaluation of these samples is a daunting computational challenge for a conventional computer processor, we show that it can be highly optimized for graphic processing units (GPUs), and we describe our real-time implementation that utilizes the 3-D virtualization and processing capabilities of a consumer-level GPU.

This section is organized in the following manner. Part A describes our dual-state motion model, and Part B details our particle-filtering approach to update this model.

A. State Model

We represent the driver's head as a rigid object constrained to six degrees of freedom in a 3-D world. This can be represented with respect to a fixed Cartesian coordinate system by the position, (x, y, z) and Euler angles (α, β, γ). To model the system using linear dynamics, we could define the state

$$\boldsymbol{x}_t = \begin{bmatrix} \boldsymbol{\theta_t} \\ \boldsymbol{\omega_t} \end{bmatrix} \tag{4}$$

where $\boldsymbol{\theta}_t = [x_t, y_t, z_t, \alpha_t, \beta_t, \gamma_t]^T$ represents the position and angle of the object at time t, and $\boldsymbol{\omega}_t$ represents the respective linear and angular velocity. In our head tracking application, however, motion is not well described by a linear system. Consider the typical motion of a person's head bobbling about in an automobile. For the most part, the subject is focused on a single location in the world, and his head is essentially static, subject only to small perturbations that can appear to be instantaneous when viewed at a sampling rate defined by a video camera. Only when the person conscientiously moves their head from one position to another can linear dynamics provide a good temporary approximation of the motion. The first situation can be modeled with a zero-velocity state model

$$\boldsymbol{x}_t^{(ZV)} = \begin{bmatrix} \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \boldsymbol{x}_{t-1} + \begin{bmatrix} \boldsymbol{\nu}_t \\ \mathbf{0} \end{bmatrix} \tag{5}$$

where $\boldsymbol{\nu}_t$ is a vector-valued random sample from an independent and identically distributed (i.i.d.) stochastic sequence that accounts for small instantaneous displacements of the head. The second situation can be described by a constant-velocity model

$$\boldsymbol{x}_t^{(CV)} = \begin{bmatrix} \mathbf{1} & \mathbf{1} \\ \mathbf{0} & \mathbf{1} \end{bmatrix} \boldsymbol{x}_{t-1} + \begin{bmatrix} \mathbf{0} \\ \boldsymbol{\eta}_t \end{bmatrix} \tag{6}$$

where $\boldsymbol{\eta}_t$ is a vector-valued sample from another i.i.d. stochastic sequence that accounts for any change in velocity of the head. At a practical level, we do not need to estimate whether the head is in a zero-velocity or constant-velocity mode, since we are only interested in the position and orientation of the head. Instead, these two models simultaneously constitute a mixed prior probability for the motion of the head.

To accommodate both of these motion models, we define the augmented state

$$\boldsymbol{y}_t = \{\boldsymbol{x}_t, \xi_t\} \tag{7}$$

where ξ_t is a binary variable $\{\xi_t : \xi_t \in 0, 1\}$ that specifies the head motion model at time t as

$$\boldsymbol{x}_t = (1 - \xi_t)\boldsymbol{x}_t^{(ZV)} + \xi_t \boldsymbol{x}_t^{(CV)}. \tag{8}$$

We can model ξ_t as a Markov chain, drawing each new sample from a probability distribution $f(\cdot)$ that only depends on the previous state

$$\xi_t \sim f(\xi_t \,|\, \xi_{t-1}). \tag{9}$$

Given this Markov property and the construction of (8), $\boldsymbol{y}_t$ is also a Markov process

$$p(\boldsymbol{y}_t | \boldsymbol{y}_0, \ldots, \boldsymbol{y}_{t-1}) = p(\boldsymbol{y}_t \,|\, \boldsymbol{y}_{t-1}). \tag{10}$$

In a classical tracking problem, the object's state $\boldsymbol{y}_t$ is observed at every time step but assumed to be noisy; hence, the optimal track can be found by maximizing the posterior probability of the movement given the previous states and observations. For a Markovian system that is perturbed by non-Gaussian noise, a sampling importance resampling (SIR) particle filter offers a practical approach that approximates the optimal track as a weighted sum of samples. These samples are drawn from the state transition density [see (10)], and the weight is set proportional to the posterior density of the observation given the samples. In our vision-based tracking problem, instead of observing a noisy sample of the object's state, we observe an image of the object. The observation noise is negligible, but the difficulty lies in inferring the object's state from the image pixels. The solution to this problem can be estimated using a similar SIR construction. We generate a set of state-space samples and use them to render virtual image samples using the fixed-function pipeline of a GPU. Each virtual image can directly be compared with the observed image, and these comparisons can be used to update the particle weights.

Given the existence of a set of N samples with known states $\{\boldsymbol{y}_t^{(l)} : l \in 0, \ldots, N-1\}$, we can devise the observation vector

$$\boldsymbol{z}_t = \begin{bmatrix} d\left(\boldsymbol{y}_t, \boldsymbol{y}_t^{(0)}\right) \\ \vdots \\ d\left(\boldsymbol{y}_t, \boldsymbol{y}_t^{(N-1)}\right) \end{bmatrix} \tag{11}$$

where $d(\boldsymbol{y}, \boldsymbol{y}')$ is an image-based distance metric. As with a classical SIR application, we are required to maintain and update a set of samples with a known state at every time step. We use these samples to update our observation vector, and with (10) and (11), we note that the observation is conditionally independent of all previous states and observations given the current state

$$p(\boldsymbol{z}_t \,|\, \boldsymbol{y}_0, \ldots, \boldsymbol{y}_t, \boldsymbol{z}_0, \ldots, \boldsymbol{z}_{t-1}) = p(\boldsymbol{z}_t \,|\, \boldsymbol{y}_t). \tag{12}$$

As a potential image comparison metric, normalized cross correlation (NCC) provides an appealing approach for comparing two image patches, having the desirable property that it is invariant to affine changes in pixel intensity in either patch. Given two image patches specified as M-dimensional vectors of intensity $\boldsymbol{\phi}$ and $\boldsymbol{\phi}'$, we can specify an NCC-based distance metric as follows:

$$d_{\mathrm{NCC}}(\boldsymbol{\phi}, \boldsymbol{\phi}') = 1 - \frac{1}{\sqrt{\sigma_{\phi}^2 \sigma_{\phi'}^2}} \sum_{i=0}^{M-1} (\phi_i - \mu_{\phi})(\phi_i' - \mu_{\phi'}) \quad (13)$$

where μ_{ϕ} is the mean of the intensity, and σ_{ϕ}^2 is the variance of the intensity. The unit constant and the minus sign are introduced to provide a positive distance measure in the range [0, 2].

When the lighting variation is nonaffine (e.g., specular reflections, shadowing, etc.), NCC poorly performs as a global image metric, since the transformation cannot be modeled by a global dc offset and scaling. If the image patches are small enough, however, then it is likely that they will be *locally* affine. As a consequence, better invariance to globally nonuniform lighting can be gained by using the average of a series of P small image patch NCC comparisons spread out over the object of interest. This is the basis of the mean NCC (MNCC) metric that we use in our tracking system as

$$d(\boldsymbol{y}, \boldsymbol{y}') = \frac{1}{P} \sum_{p=0}^{P-1} d_{\mathrm{NCC}}\left(\boldsymbol{\phi}_p, \boldsymbol{\phi}_p'\right). \quad (14)$$

We can directly relate these comparisons to the conditional observation probability if we can model the distribution such that it only depends on the current sample

$$p\left(\boldsymbol{z}_t \,\middle|\, \boldsymbol{y}_t^{(l)}\right) \propto h\left(z_{t,l}, \boldsymbol{y}_t^{(l)}\right) \quad (15)$$

where $z_{t,l}$ is the lth component of $\boldsymbol{z}_t$, and $h(\cdot,\cdot)$ is any valid distribution function.

In our head tracking system, we model the observation probability as a truncated Gaussian envelope windowed by the displacement between the current sample state and the sample with the smallest MNCC distance. Denote this latter sample as

$$\boldsymbol{y}_t^{(*)} = \left\{ \boldsymbol{y}^{(l)} : l = \underset{l}{\operatorname{argmax}}\, z_{t,l} \right\} \quad (16)$$

and define the state displacement as

$$s(\boldsymbol{y}, \boldsymbol{y}') = d_P(\boldsymbol{y}, \boldsymbol{y}') + \alpha\, d_A(\boldsymbol{y}, \boldsymbol{y}') \quad (17)$$

where $d_P(\cdot,\cdot)$ is the Euclidean distance between the position of the samples, and $d_A(\cdot,\cdot)$ is the angular displacement computed from the inverse cosine of the inner product of a quaternion representation of each sample's orientation. α is a parameter that scales the relative contribution of each measure. From these definitions, we formally define our distribution model as

$$h_t(z, \boldsymbol{y}) = \begin{cases} 0, & T_z < z \\ 0, & T_s < s\left(\boldsymbol{y}, \boldsymbol{y}_t^{(*)}\right) \\ \exp\left(-\frac{1}{2\sigma^2} z^2\right), & \text{otherwise} \end{cases} \quad (18)$$

where T_z and T_s are scalar thresholds, and σ is the standard deviation of the envelope. From qualitative analysis, we use the following parameters in our head tracking system: $\alpha = 0.01$, $T_z = 0.8$, $T_s = 0.012$, and $\sigma = 0.045$.

B. SIR

A particle filter is a Monte Carlo estimation method based on stochastic sampling [38], [39] that, regardless of the state model, converges to the Bayesian optimal solution as the number of samples increases toward infinity. The concept is to choose an appropriate set of weights and point samples

$$\left\{ \left(c_t^{(l)}, \boldsymbol{y}_t^{(l)}\right) : l \in 0, \ldots, N-1 \right\} \quad (19)$$

such that the *a priori* expectation of the state $\boldsymbol{y}_t$ can be approximated from the weighted average [40]

$$\mathrm{E}\left[\boldsymbol{y}_t \mid \boldsymbol{z}_0, \ldots, \boldsymbol{z}_t\right] \approx \sum_{l=0}^{N-1} c_t^{(l)} \boldsymbol{y}_t^{(l)}. \quad (20)$$

Let $p(\boldsymbol{y}_{0:t}|\boldsymbol{z}_{0:t})$ be the posterior probability distribution for all states up until time t. The samples can be drawn from an arbitrary *importance distribution* $\pi(\boldsymbol{y}_{0:t}|\boldsymbol{z}_{0:t})$, and the approximation is valid as long as the weights $c_t^{(l)}$ are proportional to the ratio between the posterior probability distribution and the importance distribution and $\sum_l c_t^{(l)} = 1$.

If we were to continue updating the sample weights, after only a few frames, most of the particle weights would approach zero. To practically account for this, we use a SIR filter that resamples the particles after every iteration. This is accomplished by drawing a new set of samples $\{\overline{\boldsymbol{y}}_t^{(l)} : l \in 0, \ldots, N-1\}$ from the distribution function

$$\rho\left(\overline{\boldsymbol{y}}_t \,\middle|\, c_t^{(0:N-1)}, \boldsymbol{y}_t^{(0:N-1)}\right) = \sum_{l=0}^{N-1} c_t^{(l)} \delta\left(\overline{\boldsymbol{y}}_t^{(l)} - \boldsymbol{y}_t^{(l)}\right) \quad (21)$$

where $\delta(\cdot)$ is the Kronecker delta function. After each resampling, the weight of each new sample is set to $1/N$. Given our probabilistic model and choice of $\pi(\boldsymbol{y}_t^{(l)}|\boldsymbol{y}_{0:t-1}^{(l)}, \boldsymbol{z}_{0:t}) = p(\boldsymbol{y}_t^{(l)}|\boldsymbol{y}_{t-1}^{(l)})$, the weight update equation can be reduced to

$$c_t^{(l)} \propto c_{t-1}^{(l)} p\left(\boldsymbol{z}_t | \boldsymbol{y}_t^{(l)}\right). \quad (22)$$

A full iteration of the SIR filter can be described as follows:

1) Update samples: $\boldsymbol{y}_t^{(l)} \sim p(\boldsymbol{y}_t \mid \overline{\boldsymbol{y}}_{t-1}^{(l)})$.
2) Calculate weights: $c_t^{(l)} = (p(\boldsymbol{z}_t|\boldsymbol{y}_t^{(l)}) / \sum_{l=0}^{N-1} p(\boldsymbol{z}_t|\boldsymbol{y}_t^{(l)}))$.
3) Estimate current state: $\widehat{\boldsymbol{x}}_t = \sum_{l=0}^{N-1} c_t^{(l)} \boldsymbol{x}_t^{(l)}$.
4) Resample: $\overline{\boldsymbol{y}}_t^{(l)} \sim \rho(\overline{\boldsymbol{y}}_t | c_t^{(0:N-1)}, \boldsymbol{y}_t^{(0:N-1)})$.

V. Hybrid System Implementation

Our proposed system uses a hybrid pose estimation scheme, combining our static head pose estimator procedure with a real-time implementation of our 3-D model-based tracking algorithm. The static estimator initializes the tracker from a single image frame and, as the head is tracked, continues to

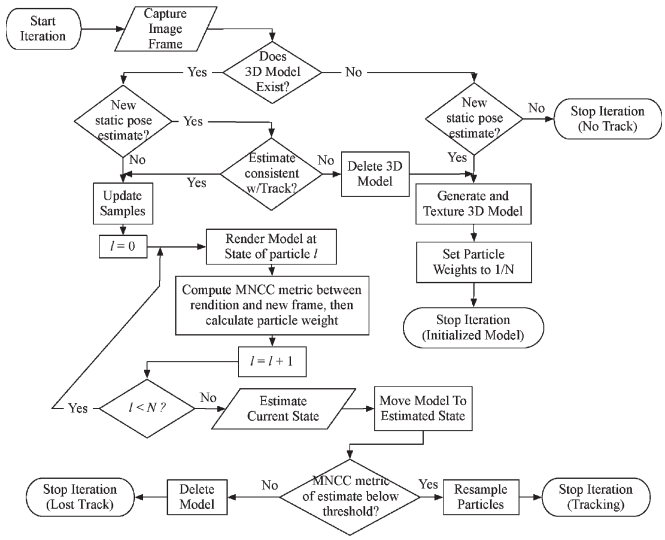


Fig. 4. Flowchart illustrating one iteration of our head tracking procedure. There are four potential results of each iteration denoted with the phase "stop iteration."

run in parallel, providing a periodic consistency check. If the tracking confidence falls below a threshold or the consistency check fails, then the static estimator automatically reinitializes the tracker. An overview of the full hybrid system is illustrated in Fig. 4.

A. Camera Perspective

The tracking system has been optimized to run on a GPU. First, we use the intrinsic parameters from our camera to model the perspective projection in the augmented reality. To correctly model the perspective projection of our camera, we must mimic the intrinsic camera parameters in our virtual environment. A camera can be modeled as an ideal perspective camera subject to spherical lens distortion. We refer the reader to [41] for more information. Many software packages are available to estimate and remove the distortion as well as estimate the intrinsic camera parameters that specify a linear projection from world coordinates into camera coordinates. We have calibrated our cameras using checkerboard calibration patterns and the facilities available in the OpenCV software library [42]. Each camera frame is undistorted before any further processing.

To project a 3-D model into the image plane with the same perspective distortion of the camera, we modify the set of projection matrices that define how a point in the virtual world is projected onto a rectilinear image, known as the *viewport*. In OpenGL, this is accomplished using two matrices:

1) ModelView matrix—A linear transformation that accounts for the position and direction of the camera relative to the model.
2) Projection matrix—A linear transformation that projects points into the viewport as *clip coordinates*. The parameters of this matrix affect the projection similar to how a lens affects a camera.

The conversion from an intrinsic matrix to the ModelView and Projection matrices requires a conversion from world coordinates to the *normalized view volume* coordinates used by OpenGL. We refer the reader to [43] for details on this conversion.

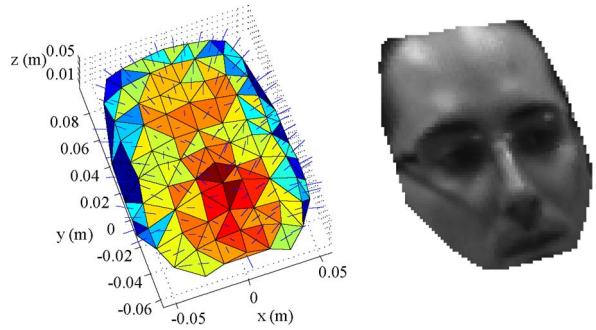


Fig. 5. (Left) Rigid facial model using for initialization and tracking. (Right) Example of the model as rendered by the tracking system.

B. 3-D Model

We represent the driver's head in our augmented reality framework as a texture-mapped 3-D model. The model consists of 3-D vertices that define a set of convex polygons approximating the surface of the head. Each vertex is assigned a texture coordinate that corresponds to a position in a 2-D image texture.

To create a new model and place it in the environment, we require a new set of polygons and an image texture. For our approach, we use a rigid anthropometric head model shown in Fig. 5. This model was created from a person excluded from the driving experiments, and although this single model is only an approximation of the facial shape of each driver, the texture-based tracking approach does not require a highly accurate fit. We show this in Section VI by comparing the rigid model to individualized models obtained by correspondence-based stereo vision. Once the model is textured, it is placed in the virtual environment with an inverse projection that puts it at the depth that corresponds to the observed width of the detected face. The static pose estimate is used to assign the initial pose angles. To ensure a symmetric view of the head, we only initialize the model if the estimated head pose is within 25° of the center; otherwise, the initialization is skipped until this constraint is satisfied.

C. Sample Generation and GPU-Based MNCC

After each new video frame is captured, it is copied into a texture object on the GPU. For each sample in the particle filter, we generate a virtual representation of the model and calculate the sample weight. To begin, the 3-D head model vertices are rotated and translated as described by the sample state. Next, the model is rendered to an off-screen framebuffer object using the fixed-function GPU pipeline (i.e., the basic procedure for rendering an object with the graphics API). Computing the MNCC distance metric described in (14) requires many computationally intensive pixel calculations. To obtain real-time speeds, we perform the calculation with the programmable pipeline of the GPU using the OpenGL Shading Language [44]. We render the vertices as individual points that compute the NCC of a local image patch using the programmable pipeline. More specifically, we compute the NCC with a *vertex shader*,

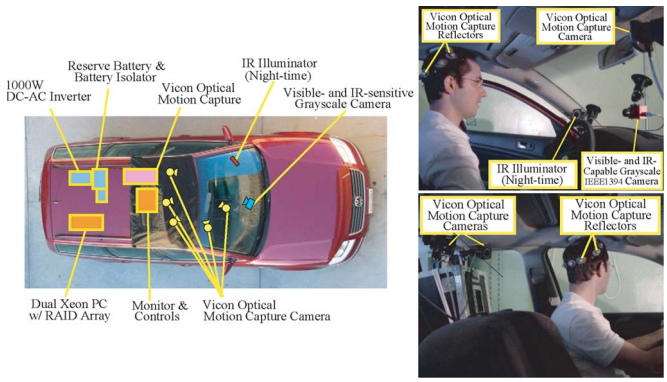


Fig. 6. LISA-P experimental testbed is a modified Volkswagen Passat equipped with a mobile computing platform and sensors for motion and video capture [45].

which is a program that operates on each vertex as it passes through the GPU pipeline. This enables hundreds of NCC windows to be computed in parallel. A full description of this GPU optimization is available in [43].

VI. Data and Evaluation: Laboratory for Intelligent and Safe Automobiles-P Testbed

The Laboratory for Intelligent and Safe Automobiles (LISA)-P experimental testbed, as shown in Fig. 6, is used to collect real-world test data. The vehicle is a modified Volkswagen Passat. An IEEE1394 camera mounted on the windshield is used to capture face data, as shown in Fig. 6. This camera captures a 640×480 pixel grayscale video stream at 30 fps, and like most CCD imagers, it is naturally sensitive to both visible and near-IR light. For our specific camera, we were required to physically remove an IR filter installed above the imager.

For the purpose of illuminating the driver's face and stabilizing the lighting conditions at nighttime, a near-IR illuminator (light-emitting-diode array with plastic diffuser) was placed on the leftmost part of the windshield. Since the emitted light is not part of the visible spectrum, it does not serve as a distraction or cause any glare for the driver. In addition, the vehicle is instrumented with a Vicon optical motion capture system, with five sensors placed in various locations around the driver's head. This marker-based system is used to gather precise ground truth head pose data for evaluation. To prevent the reflective markers from appearing in the video, we created an unobtrusive headpiece for the subjects to wear on the back of their head, as shown in Fig. 6.

For the automotive experiments, we asked 14 subjects to drive the LISA-P while wearing the motion capture headpiece. The subjects consisted of 11 males and three females, spanning Caucasian, Asian, and south-Asian descent. The subjects ranged from 15 to 53 years of age, and five of them wore glasses.

Each of the subjects drove the vehicle on different round-trip routes through the University of California campus at different times, including drives from the morning, afternoon, dusk, and night. The cameras were set to autogain and autoexposure, but these adjustments have to compete with ever-shifting lighting conditions, and dramatic lighting shifts (e.g., sunlight diffracting around the driver or headlights from a neighboring vehicle) on occasion would completely saturate the image. All of these situations remain part of our experimental data, as they are typical phenomena that occur in natural driving.

TABLE I
Comparison of Mean Absolute Error Between Head Pose Estimation Approaches

Method	Laboratory		Day Driving		Night Driving	
	Pitch	*Yaw*	*Pitch*	*Yaw*	*Pitch*	*Yaw*
Norm. Cross-corr.	**5.19°**	21.25°	11.30°	14.90°	5.94°	16.49°
Grad. PCA + SVR	5.72°	13.46°	**3.62°**	12.19°	5.19°	13.11°
LGO Hist. + SVR	5.58°	**6.40°**	3.99°	**9.28°**	**5.18°**	**7.74°**

The automobile was set up to collect data during two periods: 1) half during the summer and 2) half during the winter. The placement of the cameras mildly varies between these two setups. The drives averaged 8 min in duration, amounting to approximately 200 000 video frames in all.

Experiment 1—Static Head Pose Comparison: We compare our static head pose estimation procedure to two alternative approaches for estimating the pitch and yaw of the driver's head. The first is a prototype matching scheme that uses NCC to compare the driver's face to each of the views in our training data. To make the system more robust to noise, we take the mean of the cross-correlation score for all the training images that share the same discrete pose, and we estimate the head pose as the pitch and yaw corresponding to the maximum score after bicubic interpolation.

The second comparative head pose estimation system is our implementation of the gradient PCA system described by Li *et al.* [18]. We chose this work for comparison since it is the most similar to our proposed system and it is capable of high accuracy and speed. This approach also uses two SVRs to estimate pitch and yaw. Instead of LGO histograms, the input to each regressor is the raw horizontal and vertical image gradient reduced to a 50-D vector using principal component analysis. The PCA basis is derived from the training data.

For both of these comparative approaches, we use the same array of Adaboost cascades described in Section III-A to locate and normalize the region of the image corresponding to the driver's face.

The data used for this evaluation is a 1-min excerpt from each of the six drives: two during the daytime and four during the nighttime. In addition, we provide a comparison for an indoor scenario to evaluate whether the differences between the algorithms are specific to the automotive imagery. For indoor experiments, ten people were asked to sit on a chair against a white background while facing an IEEE1394 video camera. Behind the camera, a projector displayed a grid of points on a screen in front of the subject, each point representing a specific pose at 5° intervals spanning $(-30°, 20°)$ in pitch and $(-80°, 80°)$ in yaw. Within this grid, we displayed an active cursor, which corresponds to the subject's current head pose as measured by the motion capture system. When a subject moved his/her head to any of the 363 grid point locations for the first time, the point would change color, and the camera would capture an image of the subject. In this fashion, we obtained a uniform sampling of all ten subjects across the pose space.

The results of these experiments are found in Table I. Here, we quantify each approach by the mean absolute error in pitch and yaw between the motion capture reference and the estimated orientation. In the laboratory experiment, all three systems provide a comparable level of error in pitch, and the LGO histograms demonstrate a 7.06° reduction in yaw error over the gradient PCA approach and a 14.85° reduction over the NCC approach. In the driving experiments, our algorithm again outperforms the other approaches in absolute yaw error: 9.28° compared with 14.90° and 12.19° during the daytime experiment, and 7.74° compared with 16.49° and 13.11° during the nighttime drives. We attribute the general improvement in yaw estimation with LGO histograms to the explicit invariance they provide to positional and orientational error caused by automatic face detection and localization. Although all three approaches are invariant to affine lighting changes, the normalized cross-correlation approach shows a significant reduction in pitch estimation during the daytime drives. We attribute this to the inability of this template matching approach to operate with nonaffine lighting caused by sunlight. The SVR-based approaches in comparison do not show a decrease in accuracy from indoors to outdoors. We attribute this to the representation ability of the regressors, which learn models that account for this lighting variation. Examples of all three systems along with the ground truth data are presented in Fig. 3, and although the day driving experiment had better pitch accuracy than the laboratory experiment, we attribute this in part to the Gaussian-like distribution of head orientations in the driving experiments compared with the uniform pitch variation in the laboratory evaluation. As shown in Fig. 8, the pitch error is typically smaller for the near-frontal orientations that are frequent during driving.

Experiment 2—Anthropometric 3-D Model Evaluation: To meet our design requirement for a monocular approach, we generate a textured 3-D model of the head from a 2-D image. This is accomplished by placing a generic anthropometric facial mesh in the projected location of the detected face at a depth that corresponds to the perspective width of the face. The texture is assigned to the model by projecting the first image of the tracking sequence onto this mesh. To verify that this generic model is a sufficient approximation for tracking, we compare it to individualized texture models that are generated using a commercial stereo correspondence algorithm to estimate the 3-D shape of the driver's face [46].

For this comparison, we evaluate the tracking system on 1-min excerpts from six of the drives in which we also captured a second video stream that can provide a binocular view of the driver. This allows us to create a stereo depth map of the driver's face. By sampling the map at a 10×10 pixel interval and computing the Delaunay triangulation, we create a triangular mesh that corresponds to the surface of the face.[1] A global center and orientation is assigned in the same fashion as was done for the generic model.

In our comparison, we ignore the error generated by lost tracks and compute the mean absolute error for all of the successfully tracked frames, which we define as any track where the estimate is within 30.0° of the true pitch, yaw, and roll. The results of this comparison are presented in Table II. This shows comparable tracking accuracy with either model, with the generic model slightly outperforming the individualized models slightly in yaw and roll estimation. As we would expect individualized methods to perform better than the generic model, we attribute this contradictory result to occasional correspondence errors in the stereo model that are potentially more detrimental to the tracker than the use of a single generic facial shape. From an implementation perspective, both fixed and dynamic models yield comparable performance, but the fixed-model approach has the advantage of using a single camera.

TABLE II
COMPARISON OF MONOCULAR GENERIC 3-D MODEL TO STEREO-BASED INDIVIDUALIZED MODELS USING MEAN ABSOLUTE ERROR

	Translation (cm)			Rotation (deg.)		
Model	*x*	*y*	*z*	*Pitch*	*Yaw*	*Roll*
Generic Model	0.88	1.35	1.78	4.10	5.64	2.42
Individualized Models	0.87	1.54	1.95	4.08	5.78	2.91

TABLE III
ISOLATED ERROR FOR STATIC HEAD POSE ESTIMATION

	Rotation (degrees)		
Statistic	*Pitch*	*Yaw*	*Roll*
Mean absolute error	5.71	7.53	7.34
Std. deviation of error	7.93	10.93	10.39

TABLE IV
ISOLATED ERROR OF TRACKING ALGORITHM

	Translation (cm)			Rotation (degrees)		
Statistic	*x*	*y*	*z*	*Pitch*	*Yaw*	*Roll*
Mean absolute error	0.92	0.88	1.44	3.39	4.67	2.38
Std. deviation of error	1.87	1.22	2.37	4.71	6.89	3.74

Experiment 3—Hybrid System Evaluation: In this experiment, we evaluate our tracking system on the full video footage obtained from all 14 drivers. To train the static head pose estimator, we separately extracted a uniform sampling of the pose space for pitch, yaw, and roll (approximately 300 images from each 10-degree interval where available, and all of the data from the intervals with fewer than 300 images) and used a cross-validation scheme to train with the data from 13 of the subjects while leaving the remainder for evaluation. This was repeated for every all-but-one combination.

We first present the results for the head pose estimator and the tracking algorithm independent of each other. Table III shows the mean absolute error and the standard deviation of the error for the static head pose estimator, and Table IV provides a similar treatment for the tracking system. To compute these latter statistics, the mean position and orientation are subtracted from the ground truth and the estimated track before calculating the mean absolute error between the two. We also exclude frames in which the tracker has suffered catastrophic error due to a lost track, which we again quantify as the frames where the tracked head orientation deviates more than 30° from the true pitch, yaw, or roll. Since this is the first system to be evaluated on these data, we cannot directly compare these results to other systems. Nevertheless, our tracking results on these challenging data are within one or two degrees of the error from prior systems evaluated on much simpler data sets (i.e.,

[1]Any points in the mesh that lie 20 cm beyond the median depth are considered as outliers and are discarded.

TABLE V
COMBINED ANGULAR ERROR FROM INITIALIZATION AND TRACKING

Statistic	Rotation (degrees)		
	Pitch	*Yaw*	*Roll*
Mean absolute error	8.57	11.24	8.29
Standard deviation of error	16.43	16.90	15.27

TABLE VI
COMPARISON OF STATIONARY JITTER BETWEEN STATIC POSE ESTIMATOR AND HYBRID SYSTEM

Method	Rotation (degrees)		
	Pitch	*Yaw*	*Roll*
Static Pose Estimation Only	5.75	8.20	6.72
Hybrid Pose Estimation	1.64	2.08	1.55

indoors with only a few subjects) [14], [21]. It is worth noting that these prior systems also require calibrated stereo cameras for depth information, whereas our system uses a single camera. From these tables, one can observe that the pose errors for the tracking system are substantially smaller than the pose errors for static head pose estimation procedure.

To evaluate the combined error for the full hybrid system, we directly compare the output of the system to the motion capture ground truth. These results are presented in Table V. This table only contains angular evaluation, since the ground truth is ambiguous as to the exact position of the face, which is not the same as the position of the motion capture headpiece. This result combines the errors of the static head pose estimator and the tracker, and it also contains the errors from any lost tracks. Although the mean absolute error is larger than the static head pose estimator by itself (as should be expected), the quality of the track is much better since the actual motion of the head is better captured by the addition of the visual tracking algorithm. We can quantify this in terms of the observed jitter. We define jitter as the mean absolute change in orientation between two successive frames (i.e., the derivative of the estimate) while the head is stationary. The ground truth is used to discover the stationary frames, and the jitter is presented in Table VI. The hybrid approach shows a large reduction in jitter, as it is very good at providing a smooth and accurate track of the head motion. This is important for applications in driver intent, where the motion of the driver's head provides cues to his/her intentions (e.g., predicting when the driver is about to perform a lane change). In Fig. 7, we plot the stationary jitter as a function of the pose angle. These plots show that the static pose estimator exhibits more jitter for poses that are far off-center, whereas the hybrid system is fairly consistent across the pose space.

To show the influence of head angle on our system, we plotted the pose estimate and the standard deviation for successful tracks as a function of the true orientations in Fig. 8. The histograms on the top of each plot show the relative frequency of each pose angle. For most of the pose space, the estimate is within one standard deviation of the true pose. The static head pose estimator exhibits a regular bias toward zero that causes the curve to deviate from a pure linear slope, but the error variance is relatively stable across the pose space.

To provide an example of lost tracks and reinitialization, we include a cross-sectional plot of head yaw for a challenging 1-min video excerpt in Fig. 10. Here, the true yaw is shown

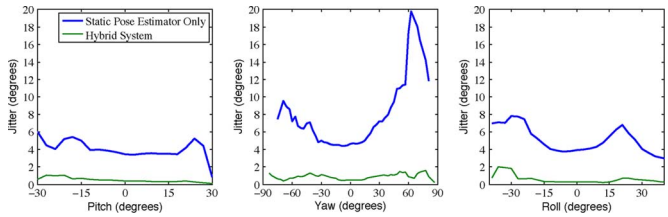


Fig. 7. Stationary jitter as a function of head position.

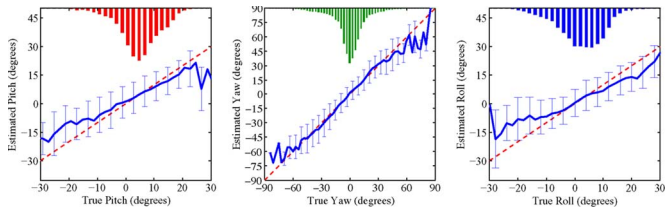


Fig. 8. Mean estimated head position as a function of the true angle. The error bars indicate one standard deviation, and the histograms show the frequency of each angle.

alongside the estimated yaw after removing any bias from the static pose estimator. In this excerpt, there are two situations in which the track is lost and reinitialized. Beginning from a successful track, the system closely follows the head until approximately 23 s. At this point, the driver makes an abrupt turn to the left and then an abrupt turn to the right. The track is lost at this point and then regained by reinitialization at about 28 s. A similar process occurs at 44 s. In these cases, the track is lost at the yaw extremes. This is difficult for the tracker since these far rotations project less of the model than frontal views. In addition, at 48 s, the tracker fails to keep up with the fast head movement but still maintains the track when the movement slows down.

Images of a tracking sequence are provided in Fig. 9. We also include example videos of the running system as supplementary material. We encourage the readers to view these videos as they provide better visualization of the system than is possible with the images alone.

VII. CONCLUSION

Robust systems for observing driver behavior will play a key role in the development of advanced driver assistance systems. In combination with environmental sensors, cars can be designed with the ability to supplement driver's awareness, preempting and preventing hazardous situations. In this paper, we have presented new algorithms for automotive head pose estimation and tracking, since head pose is a strong indicator of a driver's field of view and current focus of attention. The system satisfies all of our design criteria as it only requires monocular video for autonomous, real-time, identity-invariant, and lighting-invariant driver head pose estimation. It is an advancement in the state of the art, providing fine head pose estimation and ease of use.

We contribute two new processes that represent advances over previous head pose estimation approaches. Using LGO histograms to tolerate deviations caused by scale, position, rotation, and lighting, we demonstrated that they provide superior input to SVRs for robust head pose estimation in two degrees

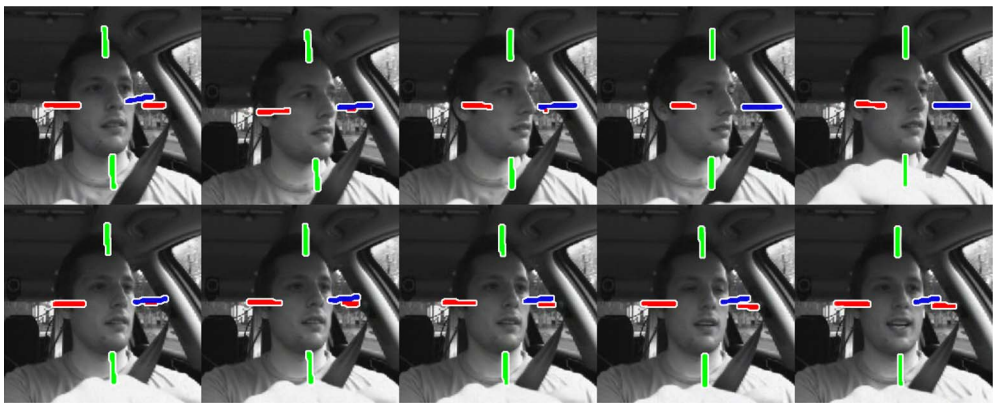


Fig. 9. Example images from a daytime tracking sequence. The images have been cropped around the driver's face to highlight the tracking estimate, which is indicated by the overlaid 3-D axes.

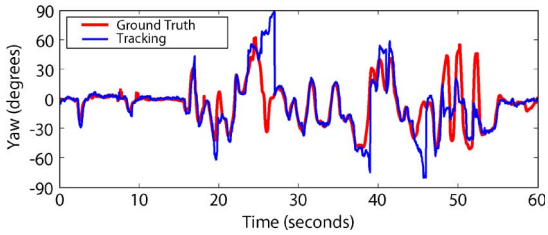


Fig. 10. Cross section of yaw during a daytime tracking sequence. The tracking bias is removed to exclude initialization error.

of freedom. The output of this static head pose estimator is used to reinitialize our particle filter-based head tracker. This real-time tracker updates a 3-D model of the head using a set of appearance-based comparisons that estimate the movement that minimizes the difference between a virtual projection of the model and the subsequent image frame.

Further extensions to this system could focus on model augmentation, since the initial model represents only a slice of the head that was visible from the perspective of a single camera when the model was created. As the head rotates, this region shifts out of view until there is very little texture remaining to continue the tracking. This effect is visible in Fig. 10, which shows how the track can become less reliable as the yaw of the head approaches 90° in either direction. As a possible solution, the model can be augmented by adding additional sets of polygons and textures. During tracking, if the rotation angle between the sample and the initial model exceeds a threshold and the MNCC score is sufficiently large to indicate an accurate track, then the initialization step can be repeated to add new polygons with a new texture to augment the original model. Care should be taken to prevent adding polygons that can overlap the existing model, and during this augmentation process, the global position and orientation do not need to be reestimated, since the are already established by the particle filter.

In conclusion, the system consists of a new method for estimating the pose of a human head that overcomes the difficulties inherent with varying lighting conditions in a moving car.

Acknowledgment

The authors would like to thank the Volkswagen Electronics Research Laboratory and the U.C. Discovery program for their support, A. Doshi for his help with experimental setup and data collection, S. Cheng for his contributions to the design and development of the LISA-P testbed and motion capture system, and the rest of their team members at the Computer Vision and Robotics Research Laboratory for their participation as test subjects and for their continuing support, comments, and assistance.

References

[1] T. Rueda-Domingo, P. Lardelli-Claret, J. L. del Castillo, J. Jiménez-Moleón, M. Garciá-Martín, and A. Bueno-Cavanillas, "The influence of passengers on the risk of the driver causing a car collision in Spain," *Accident Anal. Prevention*, vol. 36, no. 3, pp. 481–489, 2004.

[2] A. Doshi and M. M. Trivedi, "A novel active heads-up display for driver assistance," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 39, no. 1, pp. 85–93, Feb. 2009.

[3] M. Trivedi, T. Gandhi, and J. McCall, "Looking-in and looking-out of a vehicle: Computer-vision-based enhanced vehicle safety," *IEEE Trans. Intell. Transp. Syst.*, vol. 8, no. 1, pp. 108–120, Mar. 2007.

[4] S. Cheng and M. Trivedi, "Holistic sensing and dynamic active displays," *Computer*, vol. 40, no. 5, pp. 60–68, May 2007.

[5] R. Hammoud, A. Wilhelm, P. Malawey, and G. Witt, "Efficient real-time algorithms for eye state and head pose tracking in Advanced Driver Support systems," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2005, vol. 2, pp. 20–25.

[6] R. Hammoud, *Passive Eye Monitoring: Algorithms, Applications and Experiments*, 1st ed. Berlin, Germany: Springer-Verlag, 2008, ser. Signals and Communication Technology.

[7] A. Doshi and M. M. Trivedi, "On the roles of eye gaze and head pose in predicting driver's intent to change lanes," *IEEE Trans. Intell. Transp. Syst.*, vol. 10, no. 3, pp. 453–462, Sep. 2009.

[8] S. Langton, H. Honeyman, and E. Tessler, "The influence of head contour and nose angle on the perception of eye-gaze direction," *Percept. Psychophys.*, vol. 66, no. 5, pp. 752–771, Jul. 2004.

[9] L. Bergasa, J. Nuevo, M. Sotelo, R. Barea, and M. Lopez, "Real-time system for monitoring driver vigilance," *IEEE Trans. Intell. Transp. Syst.*, vol. 7, no. 1, pp. 63–77, Mar. 2006.

[10] P. Smith, M. Shah, and N. da Vitoria Lobo, "Determining driver visual attention with one camera," *IEEE Trans. Intell. Transp. Syst.*, vol. 4, no. 4, pp. 205–218, Dec. 2003.

[11] J. McCall, D. Wipf, M. M. Trivedi, and B. Rao, "Lane change intent analysis using robust operators and sparse Bayesian learning," *IEEE Trans. Intell. Transp. Syst.*, vol. 8, no. 3, pp. 431–440, Sep. 2007.

[12] E. Murphy-Chutorian and M. Trivedi, "Head pose estimation in computer vision: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 4, pp. 607–626, Apr. 2009.

[13] T. Jebara and A. Pentland, "Parameterized structure from motion for 3d adaptive feedback tracking of faces," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 1997, pp. 144–150.

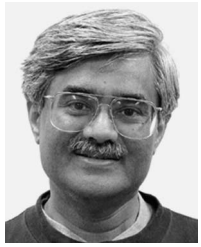
[14] L.-P. Morency, A. Rahimi, and T. Darrell, "Adaptive view-based appearance models," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2003, pp. 803–810.

[15] K. Huang and M. Trivedi, "Robust real-time detection, tracking, and pose estimation of faces in video streams," in *Proc. Int. Conf. Pattern Recog.*, 2004, pp. 965–968.
[16] R. Rae and H. Ritter, "Recognition of human head orientation based on artificial neural networks," *IEEE Trans. Neural Netw.*, vol. 9, no. 2, pp. 257–265, Mar. 1998.
[17] R. Stiefelhagen, J. Yang, and A. Waibel, "Modeling focus of attention for meeting indexing based on multiple cues," *IEEE Trans. Neural Netw.*, vol. 13, no. 4, pp. 928–938, Jul. 2002.
[18] Y. Li, S. Gong, and H. Liddell, "Support vector regression and classification based multi-view face detection and recognition," in *Proc. IEEE Int. Conf. Autom. Face Gesture Recog.*, 2000, pp. 300–305.
[19] F. Dornaika and F. Davoine, "Head and facial animation tracking using appearance-adaptive models and particle filters," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2004, pp. 153–162.
[20] J. Tu, T. Huang, and H. Tao, "Accurate head pose tracking in low resolution video," in *Proc. IEEE Int. Conf. Autom. Face Gesture Recog.*, 2006, pp. 573–578.
[21] K. Oka, Y. Sato, Y. Nakanishi, and H. Koike, "Head pose estimation system based on particle filtering with adaptive diffusion control," in *Proc. IAPR Conf. Mach. Vis. Appl.*, 2005, pp. 586–589.
[22] K. Oka and Y. Sato, "Real-time modeling of face deformation for 3d head pose estimation," in *Proc. IEEE Int. Workshop Anal. Model. Faces Gestures*, 2005, pp. 308–320.
[23] E. Murphy-Chutorian and M. M. Trivedi, "3d tracking and dynamic analysis of human head movements and attentional targets," in *Proc. ACM/IEEE Int. Conf. Distrib. Smart Cameras*, 2008, pp. 1–8.
[24] R. Pappu and P. Beardsley, "A qualitative approach to classifying gaze direction," in *Proc. IEEE Int. Conf. Autom. Face Gesture Recog.*, 1998, pp. 160–165.
[25] Y. Zhu and K. Fujimura, "Head pose estimation for driver monitoring," in *Proc. IEEE Intell. Veh. Symp.*, 2004, pp. 501–506.
[26] J. Wu and M. Trivedi, "Visual modules for head gesture analysis in intelligent vehicle systems," in *Proc. IEEE Intell. Veh. Symp.*, 2006, pp. 13–18.
[27] K. Huang and M. Trivedi, "Driver head pose and view estimation with single omnidirectional video stream," *Autom. Remote Control*, vol. 25, pp. 821–837, 2006.
[28] Z. Guo, H. Liu, Q. Wang, and J. Yang, "A fast algorithm face detection and head pose estimation for driver assistant system," in *Proc. Int. Conf. Signal Process.*, 2006, vol. 3.
[29] S. Baker, I. Matthews, J. Xiao, R. Gross, T. Kanade, and T. Ishikawa, "Real-time non-rigid driver head tracking for driver mental state estimation," in *Proc. 11th World Congr. Intell. Transp. Syst.*, 2004.
[30] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2001, pp. 511–518.
[31] R. Lienhart and J. Maydt, "An extended set of Haar-like features for rapid object detection," in *Proc. IEEE Int. Conf. Image Process.*, 2002, vol. 1, pp. 900–903.
[32] D. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vis.*, vol. 60, no. 2, pp. 91–110, 2004.
[33] K. Mikolajczyk and C. Schmid, "A performance evaluation of local descriptors," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 10, pp. 1615–1630, Oct. 2005.
[34] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2005, pp. 886–893.
[35] H. Drucker, C. Burges, L. Kaufman, A. Smola, and V. Vapnik, "Support vector regression machines," in *Proc. Adv. Neural Inf. Process. Syst.*, 1996, pp. 155–161.
[36] A. Smola and B. Scholkopf, "A tutorial on support vector regression," *Stat. Comput.*, vol. 14, no. 3, pp. 199–222, Aug. 2004.
[37] C.-C. Chang and C.-J. Lin, LIBSVM: A library for support vector machines, 2001. [Online]. Available: http://www.csie.ntu.edu.tw/~cjlin/libsvm
[38] S. Haykin, *Adaptive Filter Theory*, 4th ed. Englewood Cliffs, NJ: Prentice-Hall, 2002.
[39] S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for on-line non-linear/non-Gaussian Bayesian tracking," *IEEE Trans. Signal Process.*, vol. 50, no. 2, pp. 174–188, Feb. 2002.
[40] N. G. Arnaud Doucet and N. de Freitas, *Sequential Monte Carlo Methods in Practice*. New York: Springer-Verlag, 2001.
[41] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge, MA: Cambridge Univ. Press, 2004.
[42] OpenCV Computer Vision Library, 2007. [Online]. Available: http://sourceforge.net/projects/opencvlibrary/
[43] E. Murphy-Chutorian and M. M. Trivedi, "Particle filtering with rendered models: A two pass approach to multi-object 3D tracking with the GPU," in *Proc. Comput. Vis. Pattern Recog. Workshop, Vis. Comput. Vis. GPUs*, 2008, pp. 1–8.
[44] J. Kessenich, D. Baldwin, and R. Rost, The OpenGL Shading Language, ver. language 1.2, 3DLabs, Inc. Ltd., Milpitas, CA, 2006.
[45] S. Cheng and M. Trivedi, "Turn-intent analysis using body pose for intelligent driver assistance," *Pervasive Comput.*, vol. 5, no. 4, pp. 28–37, Oct. 2006.
[46] V. Design, Small Vision System Software. [Online]. Available: http://www.ai.sri.com/~konolige/svs/svs.htm

Erik Murphy-Chutorian (M'09) received the B.A. degree in engineering physics from Dartmouth College, Hanover, NH, in 2002 and the M.S. and Ph.D. degrees in electrical and computer engineering from the University of California, San Diego (UCSD), La Jolla, in 2005 and 2009, respectively.

At UCSD, he tackled problems in computer vision, including object recognition, invariant region detection, visual tracking, and head pose estimation. He has designed and implemented numerous real-time systems for human–computer interaction, intelligent environments, and driver assistance. He is currently a Software Engineer with Google Inc., Mountain View, CA, where he works on image content processing.

Dr. Murphy-Chutorian is actively involved as a Reviewer and program committee member for numerous computer vision and intelligent transportation publications and serves on the Computer Vision Technical Expert Task Group for the National Academies Transportation Research Board's Strategic Highway Research Program.



Mohan Manubhai Trivedi (F'09) received the B.E. degree (with honors) from the Birla Institute of Technology and Science, Pilani, India, and the Ph.D. degree from Utah State University, Logan.

He is currently a Professor of electrical and computer engineering and the Founding Director of the Computer Vision and Robotics Research Laboratory, University of California, San Diego (UCSD), La Jolla. He has established the Laboratory for Intelligent and Safe Automobiles, UCSD, to pursue a multidisciplinary research agenda. He and his team are currently pursuing research on machine and human perception, active machine learning, distributed video systems, multimodal affect and gesture analysis, human-centered interfaces, intelligent driver assistance, and transportation systems. He regularly serves as a consultant to industry and government agencies in the U.S. and abroad. He has given over 50 keynote/plenary talks. He served as the Editor-in-Chief of the *Machine Vision and Applications Journal* and is currently an Editor for *Image and Vision Computing*.

Prof. Trivedi is also currently an Editor for the IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS. He serves as the General Chair for the 2010 IEEE Intelligent Vehicles Symposium IV. He has received the Distinguished Alumnus Award from Utah State University, the Pioneer Award and Meritorious Service Award from the IEEE Computer Society, and a number of "Best Paper" Awards. He is a Fellow of The International Society for Optical Engineers.

Sensors **2012**, *12*, 3049-3067; doi:10.3390/s120303049

OPEN ACCESS

sensors

ISSN 1424-8220

www.mdpi.com/journal/sensors

Article

A Robust Kalman Algorithm to Facilitate Human-Computer Interaction for People with Cerebral Palsy, Using a New Interface Based on Inertial Sensors

Rafael Raya *, Eduardo Rocon, Juan A. Gallego, Ramón Ceres and Jose L. Pons

Bioengineering Group, CSIC, Crta. Campo Real Km 0.2 La Poveda, Arganda del Rey, Madrid 28500, Spain; E-Mails: e.rocon@csic.es (E.R.); gallego@iai.csic.es (J.A.G.); ceres@iai.csic.es (R.C.); jlpons@iai.csic.es (J.L.P.)

* Author to whom correspondence should be addressed; E-Mail: rafael.raya@csic.es; Tel.: +34-91-871-1900; Fax: +34-91-871-7050.

Received: 7 February 2012; in revised form: 24 February 2012 / Accepted: 27 February 2012 / Published: 6 March 2012

Abstract: This work aims to create an advanced human-computer interface called ENLAZA for people with cerebral palsy (CP). Although there are computer-access solutions for disabled people in general, there are few evidences from motor disabled community (e.g., CP) using these alternative interfaces. The proposed interface is based on inertial sensors in order to characterize involuntary motion in terms of time, frequency and range of motion. This characterization is used to design a filtering technique that reduces the effect of involuntary motion on person-computer interaction. This paper presents a robust Kalman filter (RKF) design to facilitate fine motor control based on the previous characterization. The filter increases mouse pointer directivity and the target acquisition time is reduced by a factor of ten. The interface is validated with CP users who were unable to control the computer using other interfaces. The interface ENLAZA and the RKF enabled them to use the computer.

Keywords: inertial sensors; human-computer interface; cerebral palsy; Kalman filter

---

1. Introduction

1.1. Definition and Classification of Cerebral Palsy

Cerebral palsy (CP) is the most common motor disability in childhood and involves a disorder of movement, posture and motor function. It is caused by a non-progressive interference, lesion, or abnormality in the immature, developing brain. CP involves a group of disorders that are permanent but not unchanging [1]. The prevalence of CP is internationally 1.5–2.0 cases per 1,000 births. More than 500,000 people in the United States have CP [2]. In Europe these figures are even higher [3]. CP is an umbrella term that involves a wide variety of diseases. It can be classified according to the pathology of brain injury or according to the timing of brain injury. The "Surveillance of cerebral palsy in Europe (SCPE): a collaboration of cerebral palsy surveys and registers" presented a consensus on the definition, classification and description of CP [4,5]. The classification based on predominant neuromotor abnormality divides CP into: spastic, dyskinetic or ataxic, with dyskinesia further differentiated into dystonia and choreoathetosis. The changing nature of symptoms and signs makes the clinical classification difficult in the first years of life as the pattern of movement and muscle tone may change completely.

The WHO International Classification of Functioning, Disability and Health (ICF) along with several other recent publications have sensitized health professionals to the importance of evaluating the functional consequences of different health states. The SCPE also recommended describing functional severity in legs and arms according to standardized scores. For ambulation, the Gross Motor Function Classification System (GMFCS) [6] has been widely employed internationally to group individuals with CP into one of five levels based on functional mobility or activity limitation. So has the bimanual fine motor function system BFMF [7], or, in prospective studies, the Manual Ability Classification System MACS [8].

1.2. Human-Computer Interfaces for People with CP

People with CP often have severe limitations using conventional human-computer interfaces (HCI), thus diminishing their opportunities to communicate and learn through computers [9]. Davies *et al*. presented a systematic review of the development, use and effectiveness of devices and technologies that enable or enhance self-directed computer access by individuals with CP [10]. They divided HCI into five categories:

- pointing devices,
- keyboard modifications,
- screen interface options,
- speech and gesture recognition software and
- algorithms and filtering mechanisms

Touch screens [11], switches with scanning approaches [9] and joysticks [12] are examples of the first category. Man *et al*. [9], presented a study to compare four different computer-access solutions: the *CameraMouse*, (head tracking using a webcam [13]), the *ASL Head Array*, (a mouse emulator using head switches), the *CrossScanner* (1–2 switch mouse emulator), and the *Quick Glance Eye Tracking*

System (eye tracking using infrared sensors). Two students with quadriplegic CP with dyskinetic athetosis participated.

The *CrossScanner* showed the highest rate of accuracy among the four systems and across the two participants. The *CameraMouse* was considered an attractive tool for postural training. The capture field of the *Quick Glance Eye Tracking System* is limited by the transmission angle of the infrared light. The participants had athetosis, which literally means "without fixed position" [14]. *Quick Glance Eye* showed low performance because the subjects continually moved out of the capture field.

Eye and face tracking interfaces are powerful pointing devices for people with motor disorders. They often succeed in improving human-computer interaction [15]. They have the potential to be a very natural form of pointing, as people tend to look at the object they wish to interact with. However, they often present low performance with people with severe motor disorders.

As regards keyboard-based solutions, some studies have demonstrated that keyboard adaptations improve speed and accuracy [16,17]. The category "screen interface options" includes the interfaces that scan through screen icons or dynamically change the icon position. Children with significant physical impairments (who are unable to point) use visual scanning and switches to select symbols. Symbol-prediction software is a method of access that involves highlighting a specific symbol within an array on the basis of an expected or predicted response [18]. The prediction software reduces the response time required for participants but there is a trade-off between speed and accuracy.

Some devices are voice-based human-computer interfaces in which a set of commands can be executed by the voice of the user. Speech-recognition software is difficult to customize for users with CP who have dysarthric speech. A combination of feedback information through auditory repeat and visual feedback may help users to reduce variability in dysarthric utterances and enable increased recognition by speech-recognition software [19,20].

Algorithms and filtering mechanisms are focused on improving the accuracy of computer recognition of keyboard input or tracking of the pointer motion. In connection with the techniques to facilitate pointer control, there are two different approaches: (1) *target-aware* and (2) *target-agnostic*. Target-aware techniques require the mouse cursor to know about and respond to the locations and dimensions of on-screen targets. Examples are gravity wells [21], force fields [22] and bubble cursors [23]. In contrast, few techniques are target-agnostic, meaning that the mouse cursor can remain ignorant of all on-screen targets, and targets themselves are not directly manipulated. Conventional pointer acceleration is by far the most common target-agnostic technique found in all modern commercial systems. Wobbrock *et al*. [24] have designed an algorithm that adjusts the mouse gain based on the deviation of angles. People with CP and other disabilities (e.g., Parkinson's or Friedreich's ataxia) participated. The authors concluded that the algorithm improved pointing throughput by 10.3% over the Windows default mouse and 11% over sticky icons. Some mathematical analyses showed that additional modeling and filters within the computer software could theoretically improve icon selection using a mouse as the input, but this was not tested in real time [25].

Access solutions for individuals with CP are in the early stages of development and future work should include assessment of end-user comfort, effort and performance, as well as design features [10]. A fundamental conclusion is that there is a wide diversity of solutions but their authors frequently assert that usability decreases dramatically when users have a severe motor disability.

1.3. Motivation and Objective

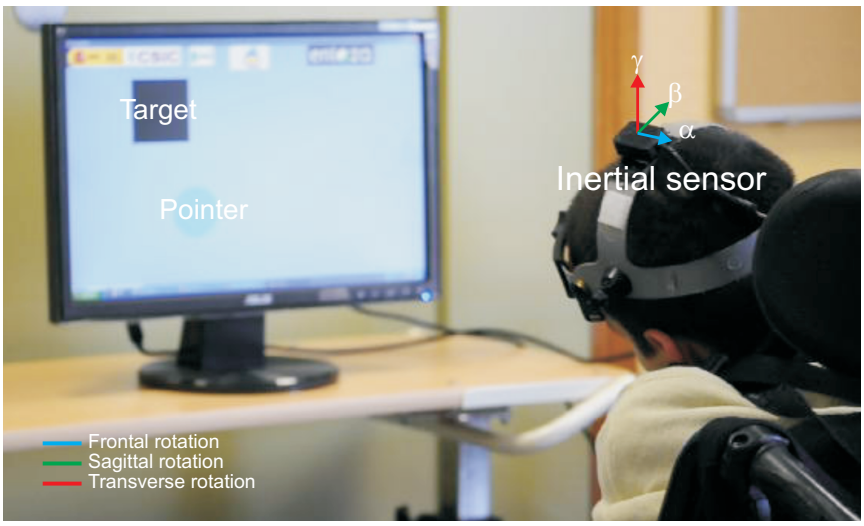
Human-machine interaction is often limited by the disability caused by CP. There are many technologies that can be useful to create alternative interfaces. This work presents an adaptive algorithm to reduce the effect of involuntary movements on human-machine interaction. This aim can be achieved as follows:

- characterizing the motor disorders in terms of motor control, frequency and range of motion (ROM). This work was presented in [26] (summarized in Section 3),
- designing the filtering algorithm based on the previous characterization (Section 4),
- validating the filtering algorithm with people with CP in real time (Section 5).

2. A New Interface Based on Inertial Sensors: The ENLAZA Interface

The ENLAZA interface is based on inertial technology. The inertial measurement unit (IMU), developed jointly by the authors and Technaid S.L., integrates a three-axis gyroscope, a three-axis accelerometer and a three-axis magnetometer. The rate gyroscope measures angular velocity by measuring capacitance based on the Coriolis force principle. The accelerometer measures the gravity and the acceleration caused by motions (by Hookes law). The magnetometer measures the Earth's magnetic field. The 3D IMU is based on microelectromechanical systems (MEMS) and is available in a package measuring 27 mm $\times$ 35 mm $\times$ 13 mm and it weighs 27 g, which is less than other sensors used in the field [27,28]. The 3D IMU is capable of sensing ± 2.0 Gauss, $\pm 500^{\circ}/s$ angular rate and ± 3 g acceleration about three axes independently. It has an angular resolution of 0.05°, a static accuracy less than 1° and a dynamic accuracy about 2° RMS. The interface consists of a headset with a commercial helmet and an IMU. Figure 1 depicts a person with CP using the inertial interface.

Figure 1. Experiments with the inertial interface at Cantabria ASPACE (Spain).



The IMU is used to estimate the sagittal and transverse rotations that are translated to pointer positions by means of a calibrated range of motion. These rotations can be calculated by the Euler angles using the equations:

$$\begin{aligned} R_{GS} &= R_s \cdot (R_G)^{-1} \\ \alpha &= \arctan(-R_{GS}(2,3)/R_{GS}(3,3)) \\ \beta &= \arcsin(R_{GS}(1,3)) \\ \gamma &= \arctan(-R_{GS}(1,2)/R_{GS}(1,1)) \end{aligned} \tag{1}$$

where R_G is the rotation matrix during calibration (global reference corresponding to the center of the screen). R_s is the rotation matrix during the task. The β and γ angles (sagittal and transverse rotations respectively) are translated to vertical and horizontal pointer positions. The frontal rotation (α) does not cause any displacement.

The inertial interface was validated by healthy users [29]. The metric used was the *Throughput* that is defined by the standard "*ISO 9241-Part 9. Requirements for non-keyboard input devices*" to assess the usability of the HCI. The *Throughput* is specifically described for target-reaching tasks. It is assumed to be a reliable estimation of goal-directed motor coordination. The metric is based on Fitts's law that models human psychomotor behavior based on Shannon's Theorem [30]. Human psychomotor behavior is simulated as a channel for information transmission measured in bits/s. A signal is transmitted through a non-ideal medium and is perturbed by noise. The effect of the noise is to reduce the information capacity of the channel. Fitts's law proposes a logarithm-linear relationship between the amplitude of the movement, the target width and the average movement time (Equation (2)).

$$T = a + b log_2(1 + \frac{D}{W}) \tag{2}$$

where T is the average time taken to complete the movement, a is the intercept and defines delay in the psychomotor system or reaction time and b is the slope coefficient. These constants can be determined experimentally by fitting a straight line to measured data. D is the distance from the starting point to the center of the target and W is the width of the target measured along the axis of motion. The ENLAZA interface was validated on 5 healthy participants using the protocol presented in [29]. The average *Throughput* was about 2 bits/s. This result is in agreement with the literature. Table 1 shows the *Throughput* values for similar interfaces.

Table 1. Usability of advanced interfaces in the literature according to the metric *Throughput*.

Device	**Throughput (bits/s)**
Mouse	3.7–4.5
ViewPoint (eyetracker) [31]	2.3–3.7
Touchpad [32]	2.9
GyroPoint (device based on gyroscope) [33]	2.8
ENLAZA interface	2
Joystick [32]	1.8
RemotePoint (isometric joystick) [33]	1.4
HeadJoystick [34]	0.92–1.93

3. Characterizing CP Motor Disorders

3.1. Objective

The characterization of abnormal head movements was presented in [26]. This section presents a summary of this characterization because it is the basis of the filter design.

3.2. Involuntary Movement and Posture

Head movement may be affected by any of the five basic types of dyskinesia: tremor, tic, chorea, myoclonus, and dystonia [35,36]. In addition, the head is subject to two dyskinesias that we call "flopping" and "nodding". Head tremor is an active, wholly involuntary, sustained pendular oscillation. Myoclonus can either be jerks (rapid contractions) or rhythmical (resembling tremor). Flopping is a passive, involuntary movement characterized by transient, exponentially decaying occurring at the end of active head movement. Tic and nodding are acquired behavioral patterns. A tic is a single, rapid, stereotyped movement. Nodding is an active, regular, sustained, usually pendular oscillation.

3.3. Main Outcome Measures

The main outcome measures for characterization were selected according to three domains: time, frequency and space.

1. Time-domain analysis. Motor disorders can be evidenced by muscle tone variations, causing difficulties in fulfilling the speed-accuracy trade-off stated by Fitts's law. The metric for the time-domain measures the correlation between the voluntary psychomotor model and the captured data (R-squared, R2).
2. Frequency-domain analysis. A frequency-domain analysis is necessary because motor disorders are frequency- and time-varying. Components calculated from Fast Fourier Transform (FFT) are (1) frequency at peak amplitude and (2) frequency corresponding to the first 75% of the area of the spectral energy. A low-pass filter at a cutoff frequency of 10 Hz was introduced to reduce the influence of electrical noise on the measurement. This frequency leaves the movement unaltered, because most voluntary and involuntary movements range from 0 to 4 Hz [36].
3. Abnormal postures can be identified by measuring the spatial variables such as predominant head orientation and ROM. Neck/head motion is clinically described as rotations about 3 orthogonal axes embedded in the head. Euler angles are useful for describing human motion such as head movement because they define rotation using three angles that can easily be physically related to frontal, sagittal, and transverse (α, β, γ) axes that are calculated using Equation (1). ROM is defined as the difference between the maximum and minimum values of α, β and γ. We calculate the ratio between the range needed to reach a target and the total range. If the ratio is less than 1, the subject moved his/her head between the screen limits; otherwise the head was out of range. α is given as absolute value because frontal rotation does not produce pointer displacements.

3.4. Participants

Four people with severe CP were recruited (Table 2). Their mean age was 29 years (range: 26–35). They were unable to use the mouse pointer or the keyboard. Three healthy users participated to extract the normalized patterns for comparison. Patient trials were carried out at Cantabria ASPACE, a specialized CP center with expertise in using alternative human-computer interfaces. Tests with healthy users were carried out at Bioengineering Lab-CSIC (Madrid, Spain). Experiments and protocols were approved by the Cantabria ASPACE expert committee.

Table 2. Characteristics of subjects with CP.

User	Motor disorder		
	Cervical tone	**General tone**	**Associated movements**
CP1	Extensor hypertonia	Extensor hypertonia	Athetosis
CP2	Dystonia	Dystonia	Ballistics
CP3	Hypotonia	Hypotonia	No
CP4	Hypotonia	Dystonia	Dystonia

3.5. Methods

Participants were instructed to locate the cursor over the target as quickly as possible using head motion. The target then changed its position, following a sequential order. One session consisted of reaching 15 targets. There were 5 sessions per user during a week, thus the target-reaching task was carried out 75 times in total. The target-reaching task is attractive because it provides a statistical description of the involuntary movements made during voluntary activity.

3.6. Results

The time-domain analysis showed relevant information about voluntary behavior during a reaching task. The reaching task can be modeled by:

1. an initial movement that rapidly covers distance and
2. a slower homing-in phase.

This voluntary movement describes a log-linear law between amplitude of movement and time in order to maintain the trade-off between speed and accuracy. The motion can consist of several "sub-movements" especially to home on the target. The "sub-movements" following the initial movement are usually performed with lower velocity and correspond to the trajectory correction. The R2 correlation between Fitts's model and psychomotor behavior was about 83% for the healthy subjects. The CP participants had lower correlation compared to the healthy users, implying that they had more difficulties in maintaining the speed-accuracy trade-off. The correlation was lower for CP1 (32%) with respect to the remaining users (50–75%). Gross motor control was better than fine motor control because there was an initial movement that rapidly covers distance. However, there were many overshoots and undershoots causing many sub-movements around the target.

The frequency-domain analysis showed that the predominant component for the healthy users was about 0.3 Hz. Three-quarters of total amplitude frequency ranged between 1.5 Hz and 3.5 Hz with a mean about 2 Hz (75% of the total power spectral density ranges from 0 to 2 Hz [26]). This result has been found to be in agreement with the literature [36]. Analyzing the movement of the people with CP, an important conclusion can be obtained: voluntary and involuntary movements share the same bandwidth. This fact must be considered to design the filtering technique. Nevertheless, the movement of CP1 presented a higher predominant frequency (>1 Hz) compared to the rest of the users with CP due to the athetotic movements associated with hypertonia. Dystonia (CP2 and CP4) causes jerky movements with irregular amplitudes and variable frequency. These results are dissimilar to other motor disorders such as Parkinson's resting tremor because the frequency of the tremor is relatively constant in any one patient in the range of 3–7 Hz. Table 3 summarizes the frequency data.

Table 3. Maximum frequency (f_{max}) y 75% (f_{75}) spectral density (Hz) mean (std).

User	f_{max} (Hz)			$f_{75\%}$ (Hz)		
	Frontal	**Sagittal**	**Transverse**	**Frontal**	**Sagittal**	**Transverse**
CP1	1.28 (0.20)	1.59 (0.50)	0.95 (0.30)	3.32 (0.45)	3.70 (0.20)	3.24 (0.26)
CP2	0.39 (0.15)	0.39 (0.18)	0.65 (0.37)	3.31 (0.48)	4.16 (0.32)	3.37 (0.24)
CP3	0.17 (0.16)	0.26 (0.06)	0.39 (0.16)	3.16 (0.38)	1.92 (0.23)	2.39 (0.15)
CP4	0.50 (0.04)	0.325 (0.18)	0.57 (0.11)	3.12 (0.34)	2.43 (0.03)	2.29 (0.17)
HS1	0.35 (0.10)	0.38 (0.04)	0.33 (0.01)	3.52 (0.56)	1.49 (0.15)	1.74 (0.21)
HS2	0.37 (0.09)	0.35 (0.09)	0.33 (0.03)	2.16 (0.24)	1.74 (0.24)	1.72 (0.15)
HS3	0.36 (0.05)	0.39 (0.07)	0.34 (0.06)	2.65 (0.47)	2.68 (0.61)	1.95 (0.17)

The spatial-domain analysis showed that the healthy users had good postural control. The range of motion offers useful information to evaluate postural stability. The ROM analysis for people with CP revealed a meaningful difference for hypotonia. Hypotonia is a decreased muscle tone that causes the head to drop forward. Sagittal ROM is more unbalanced due to the gravity effect.

According to the analysis in time, frequency and spatial domain, the following conclusions can be obtained:

- Hypertonia and athetosis movements cause involuntary movements at higher peak frequency than voluntary movements.
- Hypotonia is characterized by abnormal postural activity. The frequency of hypotonic movements is similar to the frequency of voluntary motion.
- Voluntary- and involuntary-movement frequencies share the same bandwidth.
- The spatial-domain analysis showed that the people with CP had a higher difficulty (greater difficulty) for postural control in the sagittal plane. This result was especially observed for hypotonic cases because of the low muscle tone and the pull of gravity.
- Time-domain analysis revealed that fine motor control is more affected than gross motor control.

Table 4 summarizes the design principles of the filtering technique based on motor and posture characterization.

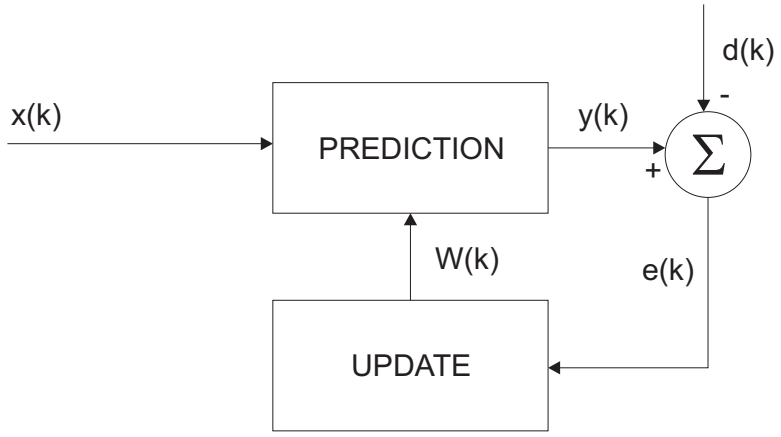
Table 4. Design principles of the filtering technique based on characterization.

Features	Design principles
(1) **Heterogeneity**	**Adaptive interface**
	Recognition of the particular user's needs
(2) **Time domain**	**Enhancement of the fine motor control**
	Reduction of the sub-movements around the target
(3) **Frequency domain**	**Definition of a voluntary control model**
	Filters based on separating frequency bands are not adequate
(4) **Spatial domain**	**Bidimensional filter**
	Independent vertical and horizontal effects

4. Filtering Algorithms

The proposed hypothesis states that long trajectories of the pointer correspond to voluntary movements whereas rapid changes are involuntary. As a consequence, the filter should dynamically adapt its gain according to trajectory deviations. An adaptive filter is time-varying since its transfer function is continually adjusted and driven by a reference signal that depends on the application. The general adaptive-filtering block diagram consists of the prediction and update steps as depicted in Figure 2. The parameter k is the iteration number, $x(k)$ denotes the input signal, $y(k)$ is the output signal and $d(k)$ defines the desired signal. The error signal $e(k)$ is the difference between d(k) and y(k). The filter coefficients $W(k)$ are updated as stated by the error signal.

Figure 2. Adaptive-filtering block diagram.



The equation parameters can be adjusted to track the movements of the mouse pointer. Some works assume a constant velocity model to describe the dynamics of the mouse pointer [37]. This assumption is reasonable when that sample period is very small compared with the movement speeds [38]. In our case, the sample period adopted was 20 ms and the dominant frequency of voluntary movement is 0.3 Hz meaning that this assumption is adequate.

The Benedict–Bordner filter [39] and the Kalman filter are adaptive filters commonly used in tracking applications. These algorithms were successfully applied by some authors for tremor

suppression [40–42]. The purpose of this investigation is to determine the feasibility of using these adaptive filters for reducing the motor disability effects caused by CP. In addition, we propose a *robust Kalman filter* that theoretically improves the performance of the classic Kalman in the presence of data outliers.

4.1. Benedict–Bordner Filter (BBF)

The g-h filter (sometimes called α-β filter) is a simple recursive adaptive filter assuming that velocity remains approximately constant. It is used extensively as a tracking filter. The g-h algorithm consists of a set of update equations:

$$\dot{x}^*_{k,k} = \dot{x}^*_{k,k-1} + h_k \left(\frac{y_k - x^*_{k,k-1}}{T} \right) \tag{3}$$

$$x^*_{k,k} = x^*_{k,k-1} + g_k \left(y_k - x^*_{k,k-1} \right) \tag{4}$$

and prediction equations [38]:

$$\dot{x}^*_{k+1,k} = \dot{x}^*_{k,k} \tag{5}$$

$$x^*_{k+1,k} = x^*_{k,k} + T\dot{x}^*_{k+1,k} \tag{6}$$

The tracking update equations or estimation equations (Equations (3) and (4)) provide the mouse pointer speed and position. The predicted position is an estimation of x_{k+1} based on past states and prediction, Equations (5) and (6), and it takes into account current measurement using updated states. T is the sample period. The selection of the parameters g and h determines whether we put the combined estimate closer to y_k or to $x^*_{k,k-1}$.

The Benedict–Bordner estimator is designed to minimize the transient error. Therefore, it responds faster to changes in movement speed and is slightly under damped [43]. The relation between filter parameters is defined by Equation (7):

$$h = \frac{g^2}{2 - g} \tag{7}$$

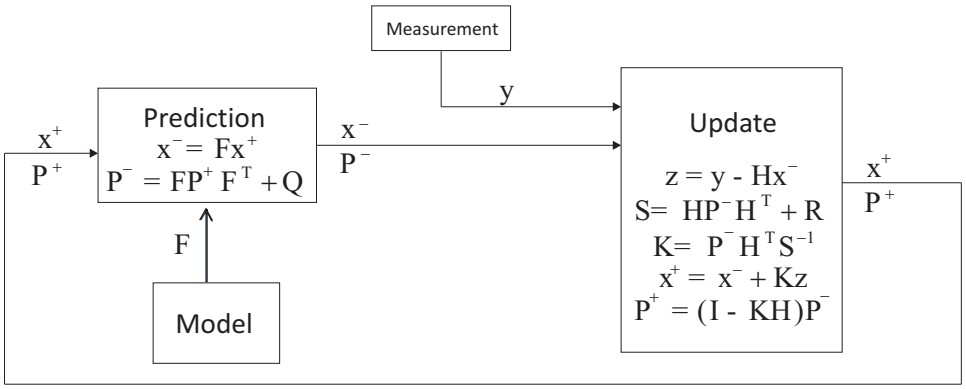
g-h gains are manually selected and static.

4.2. Kalman Filter (KF)

The application of a standard linear Kalman filter requires that the dynamics of the target is represented as a state space model [44]. A simple kinematics approach based on the assumption of the constant velocity process is suggested by some authors, and is shown to track voluntary movements correctly. Figure 3 illustrates the block diagram of the Kalman filter.

The main difference with respect to g-h filters is that a Kalman filter uses covariance noise models for states and observations. A time-dependent estimate of state covariance is updated automatically, and from this the Kalman gain matrix terms are calculated.

Figure 3. Block diagram of the Kalman filter.



4.3. Robust Kalman Filter (RKF)

The Kalman filter is commonly used for real-time tracking, but it is not robust to outliers. The sub-movements around the target region caused by motor disorders can be considered as outliers. In our application, it is difficult to define a complete model of the pathological patterns because they are not repetitive or stereotyped. We propose establishing a model of voluntary control and considering the observations that lie outside the pattern of normal distribution as outliers.

The robustification is based on the methodology of the M-estimators following Huber's function [45]. The difference between the measurement and the estimation is weighted according to Huber's function. For scalar observations, Huber's function ψ_H is [46]:

$$\varphi(Kz) = Kz \cdot min(1, b/|Kz|)$$

where $|Kz|$ is the norm. Figure 4 illustrates Huber's function.

Figure 4. Huber's function.

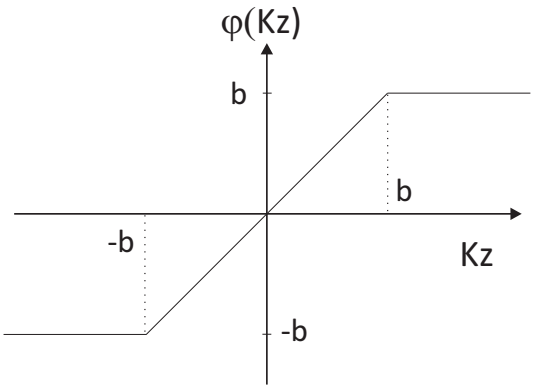
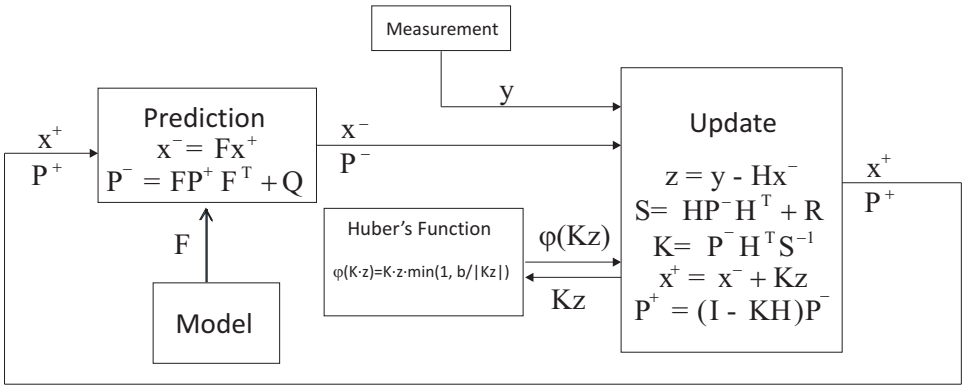


Figure 5 depicts the block diagram of the robust Kalman filter. The result is a very easy implementation and simple derivation of the classic Kalman algorithm that includes the detection and elimination of undesirable data by an iterative downweighting of the outlying observations within the least squares method. The selection of the threshold b is a trade-off between outlier rejection and control delay. Using this trade-off, b was empirically selected ($b = 5$).

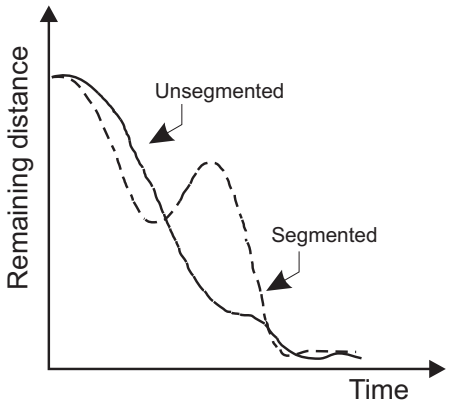
Figure 5. Block diagram of the robust Kalman filter.



4.4. Evaluation of the Filtering Techniques

The filtering algorithms BBF, KF and RKF were applied offline to the previously captured data (Section 3). The performance was compared using a metric called segmentation [47]. Segmentation measures the decomposition of a complex motion into a sequence of simpler movements that are called sub-movements. It is based on extracting maximum distance points during the reaching task. By means of segmentation, overshoots and undershoots can be detected. This metric offers information on the motor performance of both healthy subjects and persons with CP. Figure 6 illustrates an example of segmentation.

Figure 6. Kinematic descriptor of the improvement introduced by the adaptive filter (movement segmentation).



The indicators of movement segmentation can be partially interpreted as artifacts that are considered as outliers. Segmentation is estimated calculating the number of local maxima separated by 200 ms from the function "Remaining distance *versus* time". The mean of sub-movements was M = 1.41 ± 0.18 for the healthy subjects. As expected, the results showed a higher number of sub-movements for the CP subjects. The mean was about 8 sub-movements for CP1, CP2 and CP3 and slightly higher for CP4. Table 5 summarizes the number of sub-movements without and with filters. All filters considerably reduce the number of sub-movements. The RKF reduces movement segmentation up to 65%.

Table 5. Mean number of sub-movements per filtering algorithm (std).

User	Filtering algorithms (number of sub-movements (Mean(std)))			
	Without filter	**BBF**	**KF**	**RKF**
CP1	7.92 (1.26)	4.83 (0.96)	3.93 (0.70)	3.5 (0.77)
CP2	8.06 (3.38)	3.97 (1.71)	3.10 (0.98)	2.83 (0.85)
CP3	7.82 (1.54)	4.08 (1.22)	3.04 (0.77)	2.77 (0.76)
CP4	14.35 (8.07)	7.02 (4.09)	4.73 (2.42)	4.64 (2.39)

The effect of the filtering techniques can be shown graphically. Figure 7 depicts the target-reaching trajectory without the adaptive filter and with BBF, KF and RKF. Figure 8 shows the remaining distance *versus* time without and with adaptive filters for four consecutive targets. Table 5 shows that the RKF had the best performance followed by the KF and BBF. Although the BBF is able to filter high-frequency movements, it had lower performance than RKF. The reason is that BBF responds faster to changes in movement (involuntary movements) that is undesired. The detection and elimination of outliers (sub-movements following the initial movement), included by the RKF, are more adequate for this application. The gain filter is modulated in real-time and is lower during straight paths in which the prediction error is smaller. By means of outliers suppression and the dynamic gain filter, the initial movement that rapidly covers distance is smoothly filtered, whereas the movements around the target are filtered more strongly. As a consequence, the filtering technique facilitates fine control.

Figure 7. Pointer path performed by CP1 to move the cursor from a target (green circle) to the next target (blue circle). (**a**) without filtering (**b**) with BBF (**c**) with KF (**d**) with RKF.

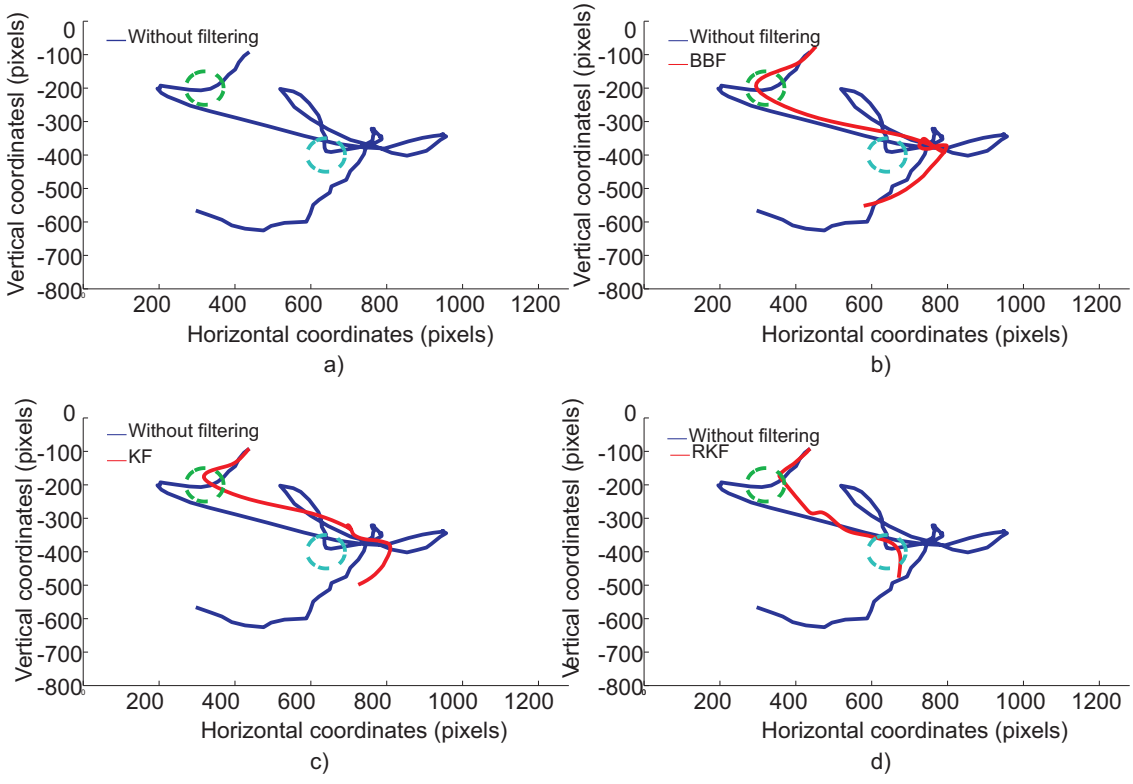
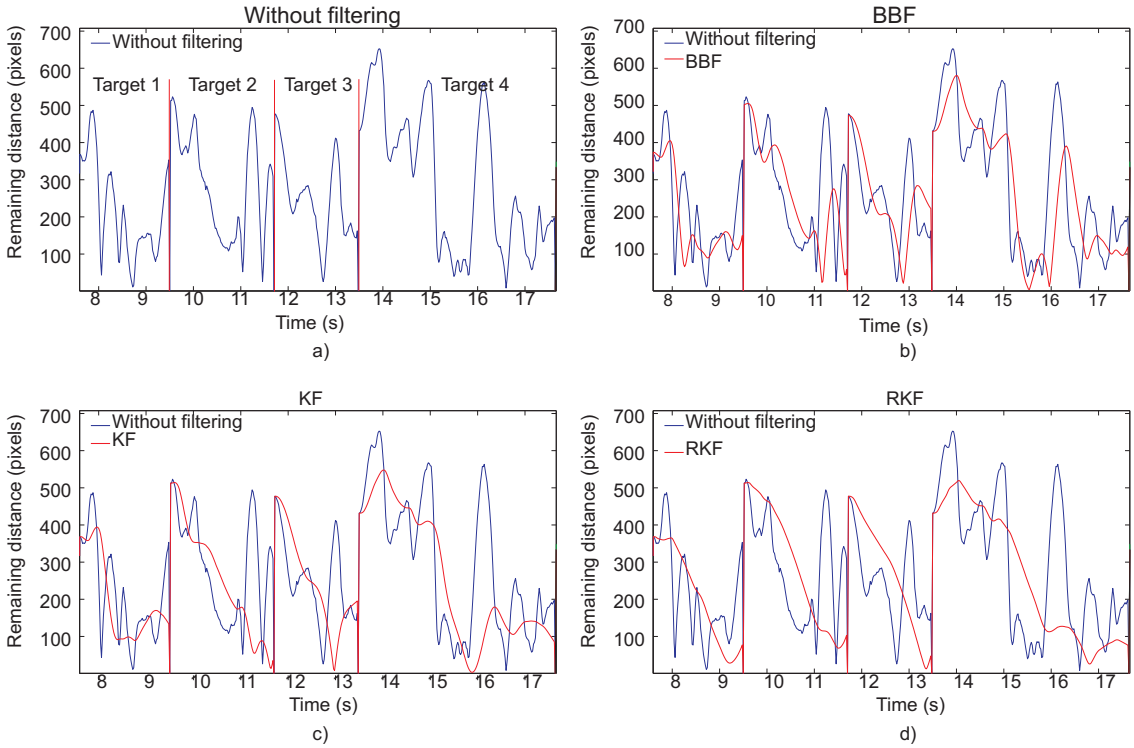


Figure 8. Remaining distance *versus* time for 4 consecutive reaching tasks performed by CP1, (**a**) without filtering, (**b**) with BBF, (**c**) with KF, (**d**) with RKF.



5. Evaluation of the Inertial Interface and RKF Algorithm with People with CP

Once the filtering algorithm has been designed, the ENLAZA interface is evaluated as a computer pointing device.

5.1. Participants and Methods

The users CP1, CP3 and CP4 participated in this experiment. CP2 was unable to participate because he had left the Cantabria ASPACE center. CP1 usually controls an eye tracking system to use the computer. CP3 and CP4 cannot use any interface (e.g., mouse or keyboard) or even advanced interfaces such as the aforementioned eye tracking system. The users CP3 and CP4 used a pointing magnifier [48]. The experiments and protocols were approved by Cantabria ASPACE expert committee.

The task consisted of reaching the target by clicking on it. The click was performed when the cursor was placed in a region of 60 pixels for 3.5 s. The metric used was the target-reaching time, as a measurement of the acquisition speed. The following three methods were compared:

- Target-reaching task without filter
- Target-reaching task with robust Kalman filter
- Target-reaching task using incremental method

The incremental method represents a simple way of filtering involuntary movements. In this mode, the pointer increases its position gradually according to the head pose. Figure 9 illustrates this control mode. The experiments were randomized in order to reduce the learning effect (Table 6).

Figure 9. Control space of incremental method. If the user looks at the grey area, the pointer stops. If the user looks at the red region, the pointer moves towards the right.

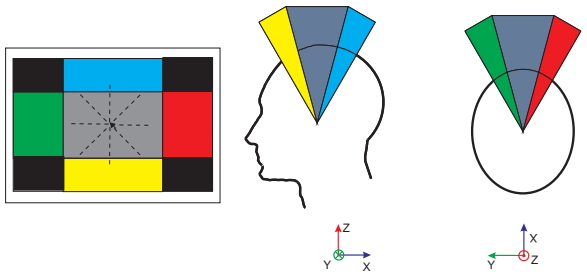


Table 6. Randomized design of the experiments. (**A**) Without filter; (**B**) With RKF; (**C**) With incremental method.

User	Day				
	1	**2**	**3**	**4**	**5**
CP1	CBA	BAC	ABC	ACB	BCA
CP3	BCA	BAC	BCA	BAC	ABC
CP4	BCA	BAC	CBA	ABC	CBA

5.2. Results

Table 7 summarizes the reaching time for each user and method. The largest difference was observed in CP1. The adaptive RKF reduces the reaching time by a factor of 10. As described in Section 3 the fundamental frequency of CP1 is higher than voluntary movement (Table 3). The filter reduces the effect of high-frequency movements, thus fine control is improved. The reaching time was reduced by a factor of 2 for CP3 and CP4 (hypotonic cases). According to the results in Section 3, correlation with the voluntary motor control was higher for the hypotonic cases. The incremental method also facilitates this control. The RKF presented a better trade-off between reaching and selection than incremental method.

Table 7. Reaching time (seconds) for each user and method.

User	Target reaching time (s). Mean(std)		
	Without filter	**Incremental method**	**RKF**
CP1	109 (10.98)	15.67 (11.70)	8.67 (4.78)
CP3	44.16 (34.77)	19.23 (6.74)	18.08 (14.82)
CP4	43.26 (37.30)	39.97 (21.26)	17.43 (12.20)

In conclusion, the inertial interface ENLAZA is an effective HCI for people with motor disorders. People who were unable to control conventional interfaces were able to control the computer with an average reaching time between 8 and 18 s. The robust Kalman filter facilitates target acquisition reducing the effect of the involuntary movements on the control.

6. Conclusions and Future Work

This work provides the following contributions for the inertial interface. The state-of-the art showed that although there are effective solutions, there is a lack of usability for users with severe motor limitations. The design of the ENLAZA interface demonstrated that inertial technology makes the extraction of pathological patterns possible. These patterns were used to define a user's needs in terms of motor control, frequency and range of motion. This work provides the following contributions for filtering techniques. A review of the state-of-the-art facilitation techniques for human-computer interaction was presented. The performance of different algorithms to reduce the effects of involuntary movements was studied. As a result, filtering techniques were selected according to the characterization of involuntary movement and posture of people with CP. Finally, a new filtering technique (RKF) based on accurately detecting and reducing deviations in the cursor trajectory was proposed and evaluated. The proposed technique improved fine motor control. Functional evaluation of the ENLAZA interface as a computer pointing device was carried out. Those subjects who were unable to control conventional interfaces were able to control the computer with the ENLAZA interface. Using the RKF algorithm, the reaching time was reduced by a factor of ten for CP1 and two for CP3 and CP4. The results illustrated that the average reaching time ranged between 8 and 18 s. The results and problems that this work faced suggest a field of work that must be addressed in the future. Long-term experiments will be interesting to analyze how physical and cognitive learning affects the device control. The filtering strategy was developed independently of the target location on the screen. As a complement to facilitate the interaction, we will study the application of the filtering strategy with other techniques based on the adaptation of the environment (*i.e.*, click crossing). According to the Cantabria ASPACE team, some users are unable to control eye tracking HCI because of their involuntary movements. Therefore, the RKF will be applied to these alternative interfaces in order to improve the accessibility of alternative HCI. The criterion for the inclusion of participants was the existence of a motor disability that limits possible interaction with assistive products. It will be interesting to extend both the number of users with CP and other groups with similar disabilities (e.g., spinal cord injury or stroke) who often have limited access to the computer. Inertial technology provides a new opportunity for analysis and extraction of kinematic patterns of voluntary and pathological movement. The development of a motion tracking system for full-body analysis is envisaged. The impact of therapies will be evaluated with objective parameters as a complement to the functional and subjective evaluation of the therapists. Motion capture and virtual representation via biofeedback methods motivate users during exercise therapy.

Acknowledgements

The authors would like to thank Cantabria ASPACE and especially Teresa González and Antonio Ruiz. This work was funded by the ENLAZA (IMSERSO-Spain), HYPER (Consolider-Ingenio, MICINN, Spain) and ABC EU-FP7 projects, and JAE-Predoc program (CSIC).

References

1. Bax, M. Terminology and classification of cerebral palsy. *Dev. Med. Child Neurol.* **1964**, *11*, 295–297.

2. Winter, S.; Autry, A.; Yeargin-Allsopp, M. Trends in the prevalence of cerebral palsy in a population-based study. *Pediatric* **2002**, *110*, 1220–1225.
3. Johnson, A. Prevalence and characteristics of children with cerebral palsy in Europe. *Dev. Med. Child Neurol.* **2002**, *44*, 633–640.
4. Cans, C. Surveillance of cerebral palsy in Europe: A collaboration of cerebral palsy surveys and registers. *Dev. Med. Child Neurol.* **2000**, *42*, 816–824.
5. Krageloh-Mann, I.; Cans, C. Cerebral palsy update. *Brain Dev.* **2009**, *31*, 537–544.
6. Palisano, R.; Rosenbaum, P.; Walte, S.; Russell, D.; Word, E.; Galuppi. Gross motor function classification system. *Dev. Med. Child Neurol.* **1997**, *39*, 214–233.
7. Beckung, E.; Hagberg, G. Correlation between ICIDH handicap code and gross motor function classification system in children with cerebral palsy. *Dev. Med. Child Neurol.* **2000**, *42*, 669–673.
8. Eliasson, A.C.; Krumlinde-Sundholm, L.; Rösblad, B.; Beckung, E.; Arner, M.; Ohrvall, A.M.; Rosenbaum, P. The Manual Ability Classification System (MACS) for children with cerebral palsy: Scale development and evidence of validity and reliability. *Dev. Med. Child Neurol.* **2006**, *48*, 549–554.
9. Man, D.W.K.; Wong, M.L. Evaluation of computer-access solutions for students with quadriplegic athetoid cerebral palsy. *Am. J. Occup. Ther.* **2007**, *61*, 355–364.
10. Davies, C.; Mudge, S.; Ameratunga, S.; Stott, S. Enabling self-directed computer use for individuals with cerebral palsy: A systematic review of assistive devices and technologies. *Dev. Med. Child Neurol.* **2010**, *52*, 510–516.
11. Durfee, J.; Billingsley, F. A comparison of two computer input devices for uppercase letter matching. *Am. J. Occup. Ther.* **1999**, *5*, 214–220.
12. Rao, R.; Seliktar, R.; Rahman, T. Evaluation of an isometric and a position joystick in a target acquisition task for individuals with cerebral palsy. *IEEE Trans. Rehabil. Eng.* **2000**, *8*, 118–125.
13. Betke, M.; Gips, J.; Fleming, P. The camera mouse: Visual tracking of body features to provide computer access for people with severe disabilities. *IEEE Trans. Neural Syst. Rehabil. Eng.* **2002**, *10*, 1–10.
14. Singer, H.; Mink, J.; Gilbert, D.; Jankovic, J. *Movement Disorder in Childhood*; Saunders Elsevier: Philadelphia, PA, USA, 2010.
15. Mauri, C.; Granollers, T.; Lorés, J.; García, M. Computer vision interaction for people with severe movement restrictions. *Interdiscip. J. Hum. ICT Environ.* **2006**, *2*, 38–54.
16. Lin, Y.; Chen, M.; Yeh, C.; Yeh, Y.; Wang, H. Assisting an Adolescent with Cerebral Palsy to Entry Text by Using the Chorded Keyboard. In *Proceedings of the 11th International Conference (ICCHP '08)*, Linz, Austria, 9–11 July 2008; In *Computers Helping People with Special Needs*; Miesenberger, K., Klaus, J., Zagler, W., Karshmer, A., Eds.; Springer-Verlag: Heidelberg, Germany, 2008; pp. 1177–1183.
17. McCormack, D. The effects of keyguard use and pelvic positioning on typing speed and accuracy in a boy with cerebral palsy. *Am. J. Occup. Ther.* **1990**, *44*, 312–315.
18. Stewart, H.; Wilcock, A. Improving the communication rate for symbol based, scanning voice output device users. *Technol. Disabil.* **2000**, *13*, 141–150.

19. Havstam, C.; Buchholz, M.; Hartelius, L. Speech recognition and dysarthria: A single subject study of two individuals with profound impairment of speech and motor control. *Logop. Phoniatr. Vocol.* **2003**, *28*, 81–90.
20. Parker, M.; Cunningham, S.; Enderby, P.; Hawley, M.; Green, P. Automatic speech recognition and training for severely dysarthric users of assistive technology: The STARDUST project. *Clin. Linguist. Phon.* **2006**, *20*, 149–156.
21. Hwang, F.; Keates, S.; Langdon, P.; Clarkson, P. Multiple Haptic Targets for Motion-Impaired Computer Users. In *Proceedings of the CHI '03*, Fort Lauderdale, FL, USA, 5–10 April 2003; pp. 41–48.
22. Ahlström, D.; Hitz, M.; Leitner, G. An Evaluation of Sticky and Force Enhanced Targets in Multitarget Situations. In *Proceedings of the 4th Nordic Conference on Human-Computer Interaction: Changing Roles (NordiCHI '06)*, Oslo, Norway, 14–18 October 2006; pp. 58–67.
23. Grossman, T.; Balakrishnan, R. The Bubble Cursor: Enhancing Target Acquisition by Dynamic Resizing of the Cursor's Activation Area. In *Proceedings of CHI '05*, Portland, OR, USA, 2–7 April 2005; pp. 281–290.
24. Wobbrock, J.O.; Gajos, K.Z. A Comparison of Area Pointing and Goal Crossing for People with and without Motor Impairments. In *Proceedings of 9th International ACM SIGACCESS Conference on Computers and Accessibility*, Tempe, AZ, USA, 22–24 October 2007.
25. Olds, K.; Sibenaller, S.; Cooper, R.; Ding, D.; Riviere, C. Target Prediction for Icon Clicking by Athetoid Persons. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '08)*, Pasadena, CA, USA, 19–23 May 2008; pp. 2043–2048.
26. Raya, R.; Rocon, E.; Ceres, R.; Harlaar, J.; Geytenbeek, J. Characterizing Head Motor Disorders to Create Novel Interfaces for People with Cerebral Palsy. In *Proceedings of the IEEE International Conference on Rehabilitation Robotics (ICORR '11)*, Zurich, Switzerland, 29 June–1 July 2011.
27. Rocon, E.; Ruiz, A.; Pons, J. On the Use of Rate Gyroscopes for Tremor Sensing in the Human Upper Limb. In *Proceedings of the International Conference Eurosensors XIX*, Barcelona, Spain, 1–14 September 2005.
28. Roetenberg, D.; Luinge, H.; C.Baten.; Veltink, H. Compensation of magnetic disturbances improves inertial and magnetic sensing of human body segment orientation. *IEEE Trans. Neural Syst. Rehabil. Eng.* **2005**, *13*, 395–405.
29. Raya, R.; Roa, J.O.; Rocon, E.; Ceres, R.; Pons, J.L. Wearable inertial mouse for children with physical and cognitive impairments. *Sens. Actuat. A Phys.* **2010**, *162*, 248–259.
30. Fitts, P. The information capacity of the human motor system in controlling the amplitude of movement. *J. Exp. Psychol.* **1954**, *47*, 381–391.
31. Zhang, X.; MacKenzie, S. Evaluating Eye Tracking with ISO 9241-Part 9. In *Proceedings of the HCI International*, Beijing, China, 22–27 July 2007; pp. 779–788.
32. Douglas, S.A.; Kirkpatrick, A.E.; MacKenzie, I.S. Testing Pointing Device Performance and User Assessment with the ISO9241, Part 9 Standard. In *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI '99)*, Pennsylvania, PA, USA, 15–20 May 1999; pp. 215–222.

33. MacKenzie, I.S.; Jusoh, S. An Evaluation of Two Input Devices for Remote Pointing. In *Proceedings of the 8th IFIP Working Conference on Engineering for Human-Computer Interaction (EHCI '01)*, Toronto, ON, Canada, 1–13 May 2001; pp. 235–249.
34. Music, J.; Cecic, M.; Bonkovic, M. Testing inertial sensor performance as hands-free human-computer interface. *WSEAS Trans. Comput.* **2009**, *8*, 715–724.
35. Marsden, C.D.; Parkes, J.D. Abnormal movement disorders. *Br. J. Hosp. Med.* **1973**, *10*, 428–450.
36. Gresty, M.A.; Halmagyi, G.M. Abnormal head movements. *J. Neurol. Neurosurg. Psychiatry* **1979**, *42*, 705–714.
37. Baldwin, P.; Basu, A.; Zhang, H. Predictive Windows for Delay Compensation in Telepresence Applications. In *Proceedings of the 1998 IEEE International Conference on Robotics & Automation*, Leuven, Belgium, 16–20 May 1998; Volume 1, pp. 2884–2889.
38. Brookner, E. *Tracking and Kalman Filtering Made Easy*; Wiley-Interscience: Malden, MA, USA, 1998.
39. Benedict, T.; Bordner, G. Synthesis of an optimal set of radar track-while-scan smoothing equations. *IRE Trans. Autom. Control,* **1962**, *7*, 27–32.
40. Riviere, C.; Thakor, N. Modeling and canceling tremor in human-machine interfaces. *IEEE Eng. Med. Biol.* **1998**, *1*, 29–36.
41. Pons, J.; Rocon, E.; Ruiz, A.; Moreno, J. *Upper-Limb Robotic Rehabilitation Exoskeleton: Tremor Suppression*; Intech Education and Publishing: Vienna, Austria, 2007; Chapter 25, p. 648.
42. Gallego, J.; Rocon, E.; Roa, J.; Moreno, J.; Pons, J. Real-time estimation of pathological tremor parameters from gyroscope data. *Sensors* **2010**, *10*, 2129–2149.
43. Bar-Shalom, Y.; Li, X. *Estimation and Tracking: Principles, Techniques, and Software*; Artech House Publishers: Boston, MA, USA, 1998.
44. Kalman, R. A new approach to linear filtering and prediction problems. *J. Basic Eng. Trans. ASME* **1960**, *82*, 35–45.
45. Huber, P. *Robust Statistics*; Wiley and Sons: Hoboken, NJ, USA, 1981.
46. Cipra, T.; Romera, R. Robust Kalman filter and its application in time series analysis. *Kybernetika* **1991**, *27*, 481–494.
47. McCrea, P.; Eng, J. Consequences of increased neuromotor noise for reaching movements in persons with stroke. *Exp. Brain Res.* **2005**, *162*, 70–77.
48. Findlater, L.; Jansen, A.; Shinohara, K.; Dixon, M.; Kamb, P.; Rakita, J.; Wobbrock, J. Enhanced Area Cursors: Reducing Fine-Pointing Demands for People with Motor Impairments. In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST '10)*, New York, NY, USA, 3–6 October 2010; pp. 153–162.

Designing a Human Computer Interface Using Laser Track Pad (LTP) for the Physically Challenged People

Arghya Ghosh[1], Chandan Garai[2], Prasun Hajra[3], Pranam Paul[4]

[1,2,3] *Dept. of Computer Science and Engineering,*
National Institute of Technical Teachers' Training and Research, Kolkata, West Bengal, India

[4] *Dept. of Computer Application,*
Narula Institute of Technology, West Bengal, India

[1] ghosh_ar@yahoo.co.in, [2] chandangarai@hotmail.com, [3] prasun.hajra@ymail.com,
[4] pranam.paul@gmail.com

***Abstract* — In this paper, we have proposed a Human Computer Interface, replacing all the basic mouse operations by simple laser beam operations controlled by a Laser Pen (LP) & tracked by the Laser Track Pad (LTP). This approach helps to overcome the limitations of the physically challenged people for interacting with the computer systems. The main emphasis of this work is on mapping the Display Units (DUs) of various resolutions with our fixed-resolution LTP. We have introduced an algorithm to solve the Intermediate Pixel Reachability problem with minimal movement of the laser beam on the LTP (i.e., also with minimal movement of the body part holding the laser pen).The potentiality and possibilities for future works are considered.**

***Keywords*— Human Computer Interface, Laser Beam, Image Processing, Intermediate Pixel Reachability.**

I. INTRODUCTION

Research on Human Computer Interaction (HCI) has gone to a certain extent in the last two-three decades. In hand held devices the key pads are being replaced by the virtual key boards. Motion sensors [1] are being used to automatically set the screen position. Newer technologies are being introduced to ease the use of computer. Mouse is replaced by touchpad. The living standards of the physically challenged people also have been improved a lot within this period. Nowadays, using computer systems has become one of the essential requirements of modern daily life. The physically challenged people who cannot hold mouse or cannot use the key board has got some alternatives like "Camera Mouse" [2], [13]-[16], "hMouse" [3] etc. These solutions, which are currently available, are mostly hardware based and more focused towards the movement of the head position or the movement of a particular body part of the user.

Some popular methods like "Eye-gaze" [4] tracks the position of the eye balls as the users move their heads. "Camera mouse" is one which requires no body-attachments as it uses the web camera of the computer system. "hMouse" calculates the user's head roll, tilt, yaw and scaling, horizontal and vertical motion for mouse control based on the reliable tracking result of the hMouse tracker. A new approach "Black Pearl" [5] has recently been launched in the market. But it is almost using the same technology as of the "Camera Mouse". All these technologies depend on the movement of the users' head positions. But, practically this needs a huge movement of the head to navigate the mouse cursor between the extreme corners of the display units, which sometimes becomes impossible for a physically challenged people.

Laser Track Pad (LTP) focuses greatly on reducing the head movement/body part movement of the user and accurately tracks the position of the laser beam [6] on the track pad through a camera fitted within it. A physically challenged people can grip a pen in the mouth or between his toes (Hallux & Second toe) and can write or paint something. Similarly a Laser Pen is also very easy to grip & operate almost in the same way.

In this paper, the basic components required for the purpose are defined. Working principle along with the laser beam actions (equivalent to the conventional mouse operations) are also being mentioned in brief. In section-V the algorithms for mapping of the LTP with the DU have been proposed and analyzed with the help of an example. Section-VI discusses the implementation details & lastly, section-VII concludes the paper.

II. DESCRIPTION OF THE COMPONENTS

To implement all the different operations of a mouse we require a LTP fitted with a web camera and a source of laser light (i.e., a laser pen). According to the movement of the laser beam on the LTP, a mouse cursor is navigated to point the desired location on the DU. Interactions of the LP and the LTP with the DU are being shown in the Fig. 1.

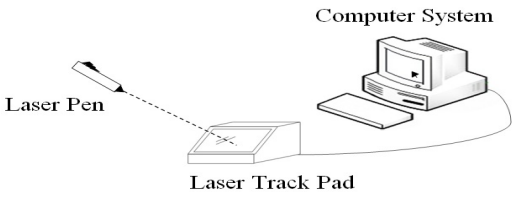


Fig. 1 Components and Their Interaction.

A. *Description of the Laser Pen*

The low-cost Laser Pen (of height 50 mm & diameter 13 mm), used in this work is very common & easily available in the market. It comes with an ON/OFF switch which can be easily operated by an effortless pressing (even by the lips). Fig. 2 shows the basic architecture of the Laser Pen.

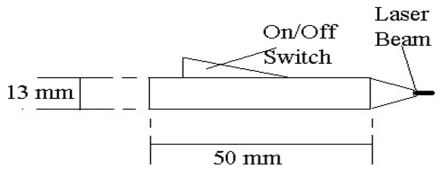


Fig. 2 Laser Pen

B. *Description of the Laser Track Pad*

A LTP is composed of a standard web-camera placed under a frosted glass sheet, focusing the centre of the glass sheet and maintaining a fixed distance (depending on the focal length of the lens used) from it such that the total area of the LTP comes within the Angle of view of the lens [11]. Angle of view can be measured horizontally, vertically or diagonally. Fig. 3 shows the basic architecture of the Laser Track Pad.

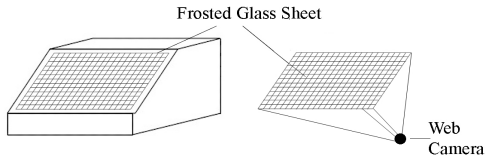


Fig. 3 Laser Track Pad

III. WORKING PRINCIPLE

In this system, we have first mapped the LTP with the DU. That is, through the incidence of a laser beam on the LTP, we can point to any of the desired locations on the DU. Now a LP is used as a source of the laser light to point on the frosted glass sheet of the LTP. The distance between the laser source and the track pad does not affect the degree of accuracy for the identification of the incident point of the laser beam [12] on the LTP.

Then from the video streams captured by the web camera fitted inside the LTP box, the co-ordinates of the laser incident point are identified and the mouse is navigated to the equivalent position on the DU by applying the LTP to DU Mapping Algorithms.

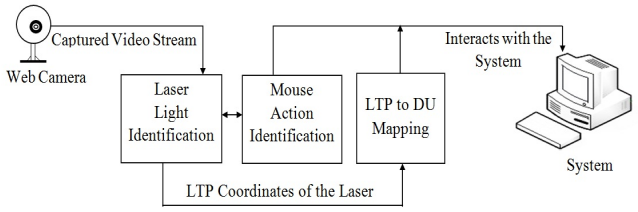


Fig. 4 Block Diagram of the System

Mean while, by analyzing the video streams we identify the basic mouse actions (e.g., mouse pointer move, single click, double click and right click) performed by the user at that point and accordingly the action is performed. So, in this way the physically challenged people can interact with the computer system with minimal head/body part movement. The block diagram of the system's work flow is shown in Fig.- 4.

IV. BASIC MOUSE OPERATIONS REPLACED BY THE LP ACTIONS

Table-I
LIST OF MOUSE ACTION

Mouse Operation	Laser Pen Action	Time Span
Mouse Pointer Move	{ON-MOVE}	–
Single click	{ON-OFF}	t
Double Click	{ON-OFF}, {ON-OFF}	t
Right Click	{ON-OFF}, {ON-OFF}, {ON- OFF}	t

1st column of the Table-I represents the basic mouse operations and the 2nd column represents the corresponding actions of the Laser Pen, (the action states are defined in the Table-II). 3rd column represents the fixed time span (t) for successful execution of different mouse operations.

Table-II
DESCRIPTION OF ACTION STATES

States	Description
ON	Start of laser beam emission.
OFF	Stop of emission.
ON-OFF	ON before OFF
MOVE	Movement of the laser beam.

According to the working principle, the number of {ON-OFF} pair(s) within the fixed time span 't' is counted and accordingly the corresponding mouse operation is executed, i.e. if the {ON-OFF} pair occurs once within the specified time span 't', it is treated as a single click. If it is counted as 2 or 3, then the mouse double click or mouse right click is executed respectively.

V. ALGORITHM

In this work, algorithms for laser light identification and mapping between LTP & DU are proposed & implemented.

A. *Laser Light Identification*

INPUT: Images captured by the web camera (at minimum speed of 7fps).

OUTPUT: Co-ordinates of the laser incident point.

Step1: Convert each and every image captured by the web camera in to gray scale.

Step2: Filter out everything except laser light of specific intensity range, using the Fourier transformation on each and every gray scale images.

Step3: Retrieve the position of the specific intensity laser dot within the image.

Step4: Return the co- ordinates and stop.

B. *LTP to DU Mapping*

This algorithm is to map the LTP of a fixed resolution with display devices of different resolutions (e.g.: 640 X 480, 800 X 600, 1280 X 1024 etc.). Otherwise, if the mapping was limited between the LTP & the DU of a specific resolution,

then sometimes it might happen that the total area of the display screen is not covered by the laser beam movement on the LTP.

To map the x & y co-ordinates of the LTP with the same of the DU, two mathematical factors (RI_W & RI_H) have been proposed. In Fig. 5 we have shown the mapping of the above mentioned display devices (having two different resolutions & aspect ratios) with our fixed-sized LTP.

The LTP to DU mapping has been achieved by two algorithms -- "Basic LTP to DU Mapping Algorithm" and "Intermediate Pixel Reachability Algorithm".

1. Basic LTP to DU Mapping Algorithm

Now, let us denote the following terms,

PW_{LTP} : No. of pixels along the width of the LTP.
PH_{LTP} : No. of pixels along the height of the LTP.
PW_{DU} : No. of pixels along the width of the DU.
PH_{DU} : No. of pixels along the height of the DU.

IW_{LTP} : No. of intervals along the width of the LTP.
IH_{LTP} : No. of intervals along the height of the LTP.
IW_{DU} : No. of intervals along the width of the DU.
IH_{DU} : No. of intervals along the height of the DU.

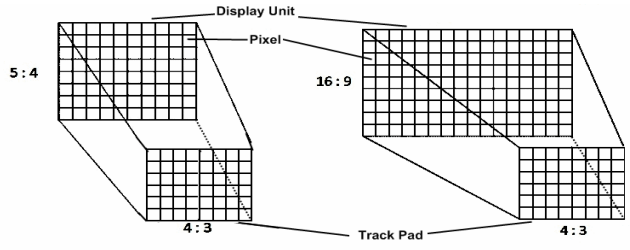


Fig. 5 Mapping of different displays with the fixed sized LTP.

Then, $IW_{LTP} = PW_{LTP} - 1$
$IH_{LTP} = PH_{LTP} - 1$
$IW_{DU} = PW_{DU} - 1$
$IH_{DU} = PH_{DU} - 1$

$RI_W = IW_{LTP} : IW_{DU}$
$= 1 : (IW_{DU} / IW_{LTP})$ ……..(1)
$RI_H = IH_{LTP} : IH_{DU}$
$= 1 : (IH_{DU} / IH_{LTP})$ ……..(2)

Therefore, if the point (x, y) on the LTP is mapped to the pixel position (X, Y) on the DU, then

$X = (1/ RI_W) * x$ and
$Y = (1/ RI_H) * y$

Example:

Let the resolution of the LTP be 4 x 4 and the resolution of the DU be 16 x 12. We have to map the LTP with the DU. Fig. 6 shows the pixel wise mapping.

Then, $RI_W = \frac{(4-1)}{(16-1)} = \frac{1}{5}$ …… from Eqn. (1),

$RI_H = \frac{(4-1)}{(12-1)} = \frac{1}{3.67}$ ……. from Eqn. (2).

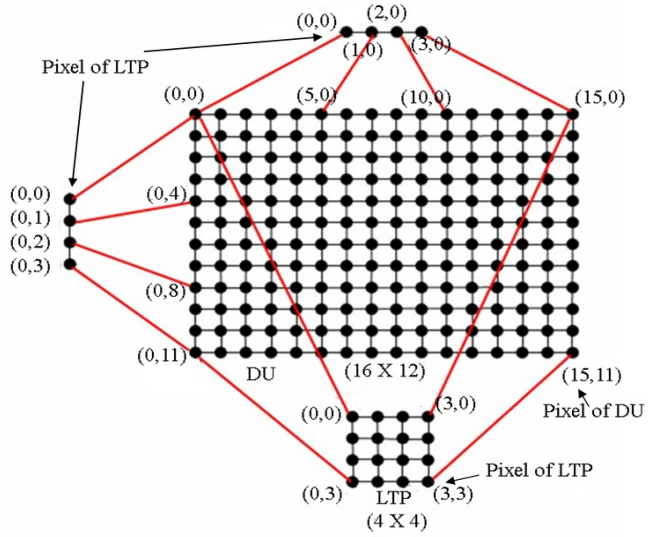


Fig. 6 LTP to DU pixel wise mapping.

In the above section we have discussed about the Basic LTP to DU Mapping Algorithm. But, this algorithm has a severe limitation as one pixel movement of the laser beam on LTP along the width & the height causes a movement of (1/ RI_W) & (1/ RI_H) pixels of the mouse curser on the DU respectively. That is, if the laser beam moves from (0,0) to (0,1) along the width on the LTP, the mouse cursor jumps from (0,0) to (0,5) on the DU. That is the intermediate pixels viz. (0,1), (0,2), (0,3) & (0,4) on the DU could not be reached by the movement of the laser beam on LTP.

To solve this problem in both horizontal and vertical directions, we have introduced two new speed factors (SF_W & SF_H) in the Intermediate Pixel Reachability Algorithm that will control the mouse cursor movement to reach the intermediate pixels.

2. Intermediate Pixel Reachability Algorithm

Let, "s" be the speed at which 1 pixel movement of the laser beam on the LTP causes maximum of (IW_{DU} /IW_{LTP}) = (1/ RI_W) (say N) pixel movement on the DU --- From Eqn. (1).

Now, we can understand that to reach the intermediate pixels on the DU, we have to move the laser beam at a speed fractional to "s" (lesser than "s", different for different intermediate pixels).

Experimental results are given in the following Table-III **where N = 5**. Here, the dots in the 4th column of the table represent the pixels on the DU and the connecting line represents the number of pixels covered.

Table-III
PIXELS MOVEMENT ON DU

Movement of beam	SF_W (Speed)	# Intervals moved on DU	Pixels moved on DU
1 unit	$\frac{(N-0)}{N} * s$	N - 0	
1 unit	$\frac{(N-1)}{N} * s$	N - 1	
1 unit	$\frac{(N-2)}{N} * s$	N - 2	
1 unit	$\frac{(N-3)}{N} * s$	N - 3	
1 unit	$\frac{(N-4)}{N} * s$	N - 4	

Therefore, If the Laser beam moves 1 pixel horizontally on the LTP with a speed factor $SF_W = \frac{(N-i)}{N} * s$ then, the $(N-i)^{th}$ intermediate pixel is reached.

Here, $N = (1/ RI_W)$ and i = 0, 1, 2, 3, ……., (N-1).

Using the same approach we can reach the intermediate vertical pixels (along the height) with $SF_H = \frac{(M-j)}{M} * s$ where $M = (1/ RI_H)$ and j = 0, 1, 2, 3, ……., (M-1).

VI. IMPLEMENTATION

We have implemented the application of the LTP in Aforge.Net [7] framework. Aforge.Net is basically a library for image processing routines and filters [8], [9]. To track the laser light, Aforge.Net is provided with a number of standard methods, which can deal with image processing. We have used few such methods to process the video stream captured by the web camera.

Considering video as a continuous stream of images, for real time image processing we need to process each and every image captured by the web camera. We are taking an average of 7 frames per second (fps) within our application. More the fps, more smoothly the program will run.

The program searches for the brightest red blob [7] in web camera's field of views. It may be done by using two of the filter defined in AForge.NET Framework (blob counter and color filter).

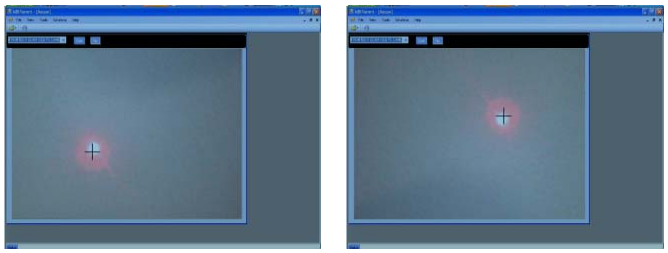


Fig. 7 Application output

In this application we have already implemented the portion to identify the specific intensity laser light and the co-ordinate of that laser light on the LTP. And we have mapped the LTP with the DU. We have also successfully implemented the 'mouse left click' and the 'mouse double click' operation through LTP. And we are working on the rest part of our proposed model. Fig. 7 shows the movement of the mouse cursor according to the movement of the red laser light.

VII. CONCLUSION

The implementation of LTP shows that it is a very effective pointing device and it significantly reduces the head/body part movement of the user, which is really helpful for the physically challenged people. In our implementation the LTP tracks and separates all the lights within a specific intensity range (the intensity range of the red Laser light). So, there is a possibility of some other lights within that intensity range to be unnecessarily tracked by the LTP. These unwanted lights may cause the identification of the Laser incident point erroneous. As future works it can be possible to implement the infrared version of the LTP. Still in the series of Camera mouse, Laser Track Pad (LTP) is a useful successor.

REFERENCES

[1] Aygun, R.S.; Zhang, A.; *"Global motion estimation from semi-dynamic video using motion sensors*", Volume: 2, On page(s): II-273 - II-276 vol.2 ISSN: 1522-4880, Print ISBN: 0-7803-7622-6.

[2] M. Betke, J. Gips, and P. Fleming, *"The Camera Mouse: Visual Tracking of Body Features to Provide Computer Access for People with Severe Disabilities"*, IEEE Trans. on NSRE, vol. 10, pp.1-10, 2002.

[3] Yun Fu and Thomas S. Huang, *"hmouse: Head Tracking Driven Virtual Computer Mouse"*, WACV'07.

[4] T. Hutchinson, K. P. White Jr., W. N. Martin, K. C. Reichert, and L. A. Frey, *"Human-Computer Interaction Using Eye-Gaze Input"*, IEEE Trans. Syst., Man, Cybern., vol. 19, pp. 1527–1533, 1989.

[5] Rajesh Kumar, Anupam Kumar*," Black Pearl: An Alternative for Mouse and Keyboard"*, ICGST-GVIP, ISSN 1687-398X, Volume (8), Issue (III), October 2008

[6] Perez-Arancibia, N.O.; Gibson, J.S.; Tsao,T.-c*, "Observer-Based Intensity-Feedback Control for Laser beam Pointing and Tracking"*, IEEE control system society, vol. 6, pp.1-17, 2011.

[7] http://www.codeproject.com/KB/GDIplus/Image_Processing_Lab.asp

[8] RafaelC.Gonzalez, Rechard E.Woods, *"Digital Image processing"*, 3 Ed.

[9] Zhou, Zhong-liang; Ying, Jia-ju;"Study on Image Processing Technology in Imaging Laser Detection System", 19-21 June 2010

[10] Pranam Paul, S.Dutta, A. K. Bhattacharyaa,"*An Approach to Ensure security Through Bit-level encryption with possible loss less Compression";*IJCSNS vol-8,No-2,feb 2008,pp 291-299

[11] Gil-Whoan Chu, Won Phil Yu, Myung Jin Chung, "*A simple method of finding a visible region for the optimal camera position",*ICAR '97 Proceedings, 6th Aug 2002, pp 583-588

[12] Matulka, Donald D.*"Application of LASERS to Digital Communication",* Aerospace and Navigational Electronics, IRE Transactions, vol. ANE-9, Issue 2, May 2009, pp 104-109

[13] Jilin Tu, Thomas Huang Hai Tao*," Face as Mouse Through Visual Face Tracking",* Proceedings of the Second Canadian Conference on Computer and Robot Vision (CRV'05) 0-7695-2319-6/05

[14] W. J. Perkins and B. F. Stenning, "*Control units for operation of computers by severely physically handicapped persons," J. Med. Eng. Technol.*, vol. 10, no. 1, pp. 21–23, 1986.

[15] O. Takami, N. Irie, C. Kang, T. Ishimatsu, and T. Ochiai, "*Computer interface to use head movement for handicapped people,*" in *Proc. IEEE TENCON'96, Digital Signal Processing Applications*, vol. 1, 1996, pp. 468–472.

[16] D. G. Evans, R. Drew, and P. Blenkhorn, "*Controlling mouse pointer position using an infrared head-operated joystick*," *IEEE Trans. Rehab. Eng.*, vol. 8, no. 1, pp. 107–117, 2000.

US 20120287284A1

(19) **United States**

(12) **Patent Application Publication**

(10) **Pub. No.: US 2012/0287284 A1**

(43) **Pub. Date: Nov. 15, 2012**

Jacobsen et al.

(54) **HEADSET COMPUTER THAT USES MOTION AND VOICE COMMANDS TO CONTROL INFORMATION DISPLAY AND REMOTE DEVICES**

(75) Inventors: **Jeffrey J. Jacobsen**, Hollister, CA (US); **Christopher Parkinson**, Richland, WA (US); **Stephen A. Pombo**, Campbell, CA (US)

(73) Assignee: **Kopin Corporation**, Taunton, MA (US)

(21) Appl. No.: **13/468,207**

(22) Filed: **May 10, 2012**

Related U.S. Application Data

(60) Provisional application No. 61/484,464, filed on May 10, 2011.

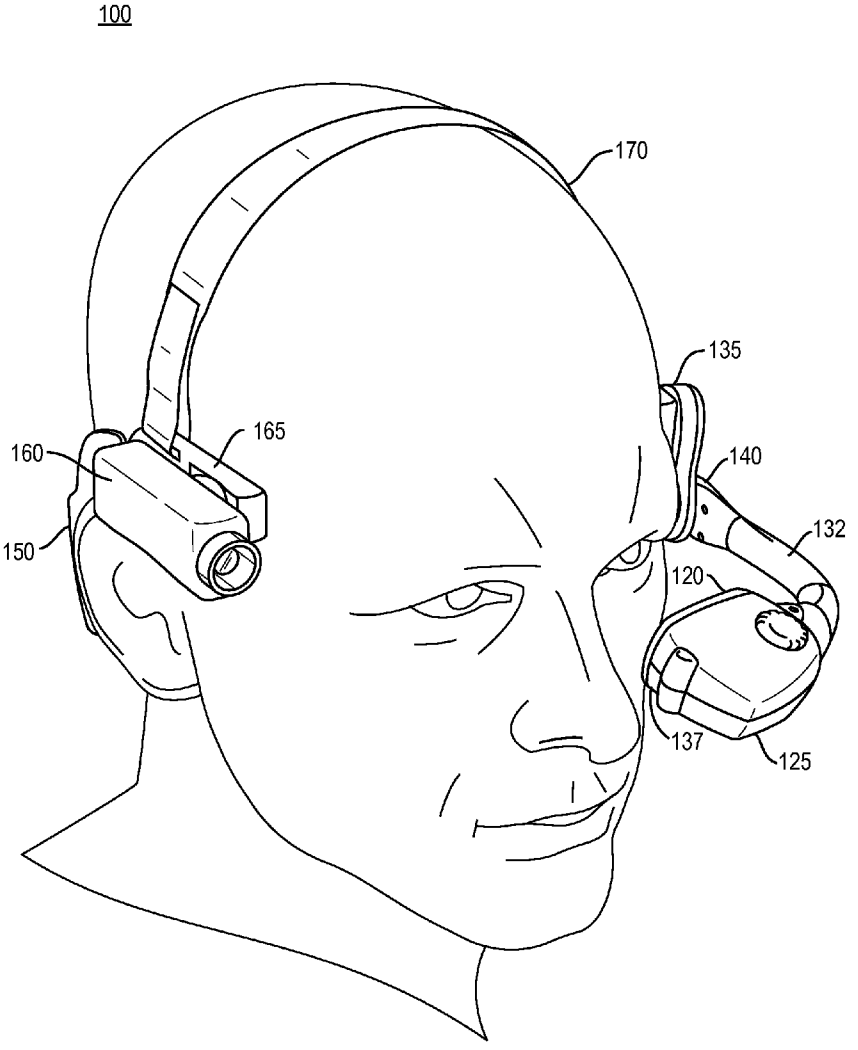
Publication Classification

(51) **Int. Cl.**
H04N 7/18 (2006.01)
G06T 15/00 (2011.01)

(52) **U.S. Cl.** **348/158**; 345/419; 348/E07.085

(57) **ABSTRACT**

A wireless hands-free portable headset computer with a micro display arranged near but below a wearer's eye in a peripheral vision area not blocking the wearer's main line of sight. The headset computer can display an image or portions of an image, wherein the portions can be enlarged. The headset computer also can be equipped with peripheral devices, such as light sources and cameras that can emit and detect, respectively, visible light and invisible radiation, such as infrared radiation and ultraviolet radiation. The peripheral devices are controllable by the wearer by voice command or by gesture. The headset computer also can be broken down into component parts that are attachable to another article worn by an individual, such as a helmet or respirator mask.



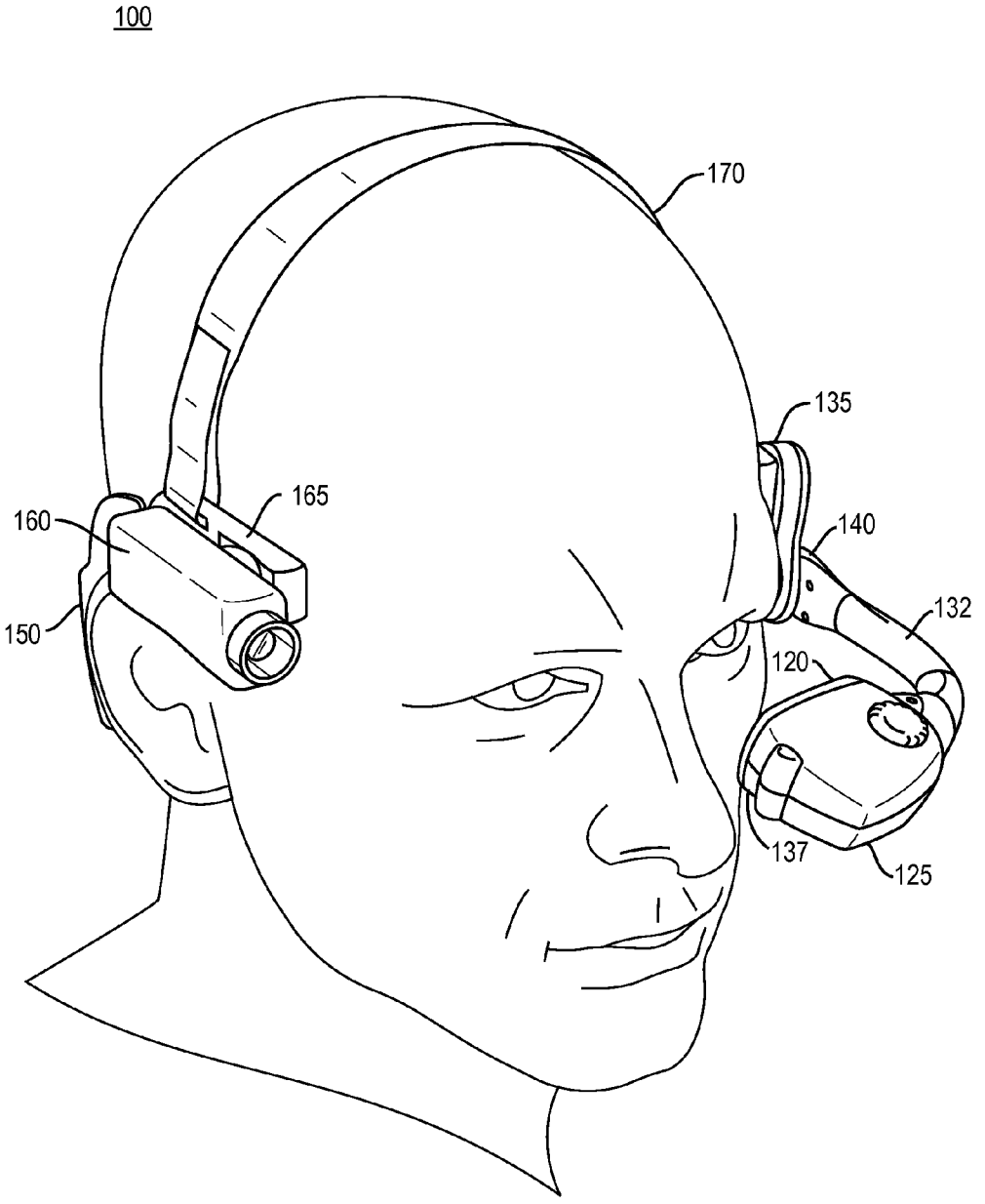


FIG. 1

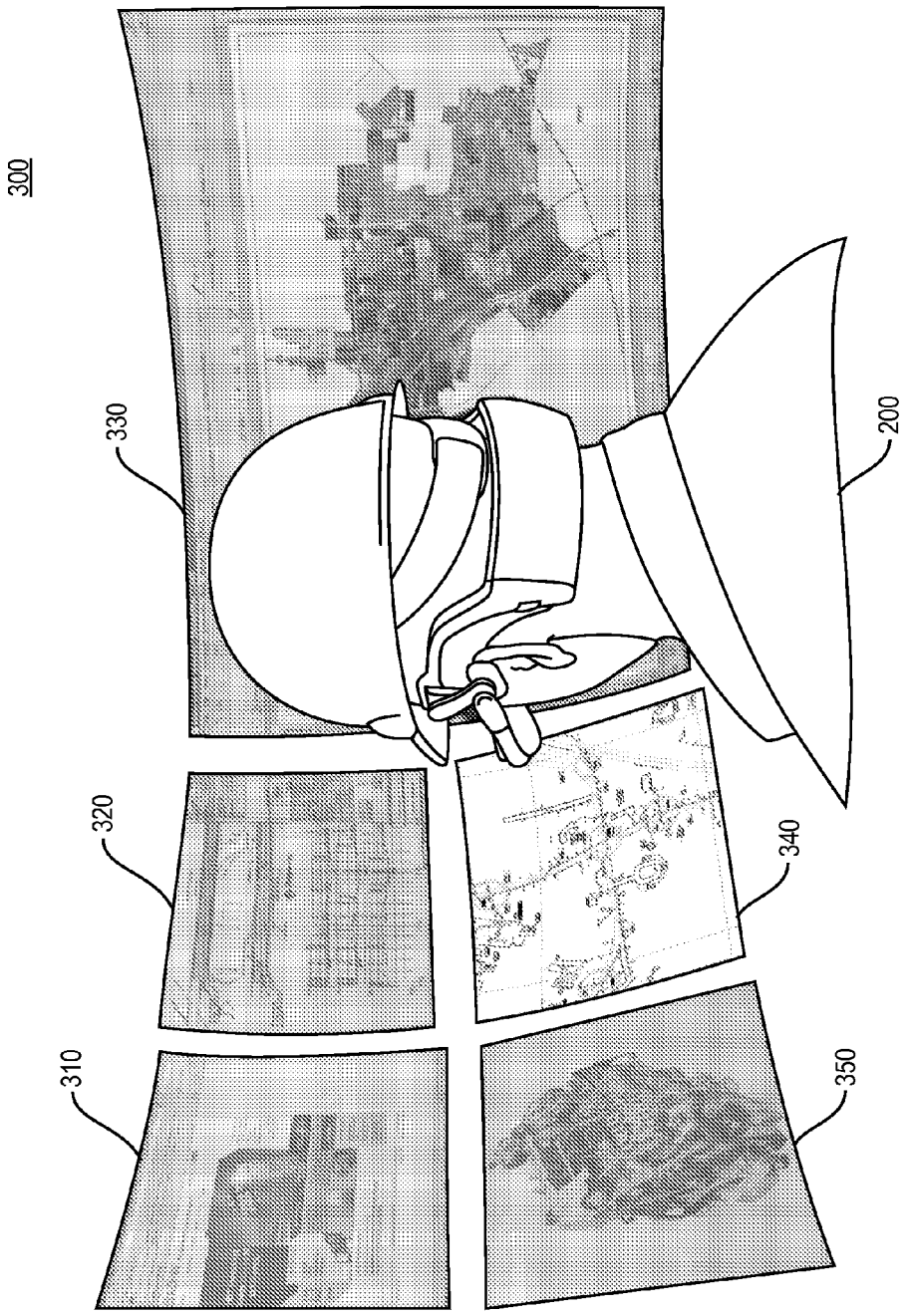


FIG. 2

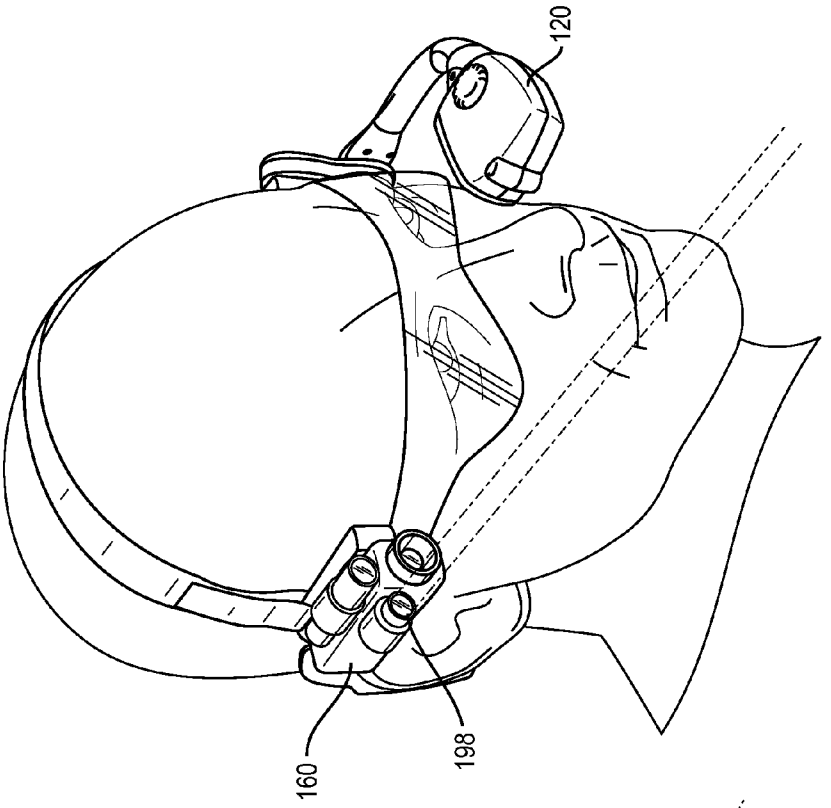


FIG. 3B

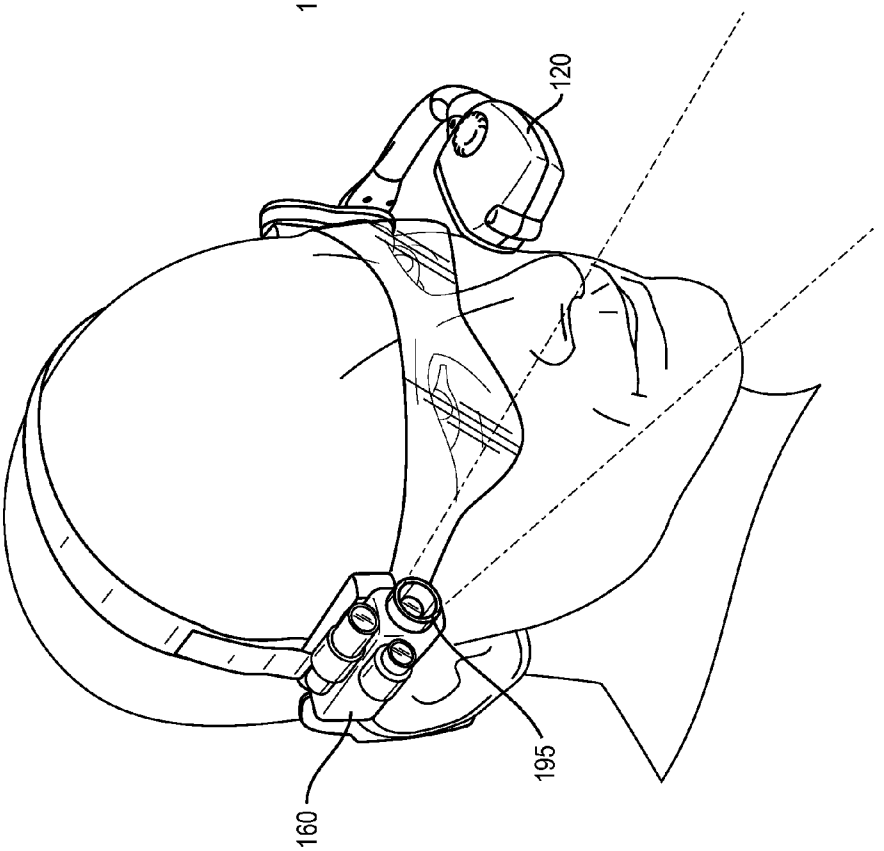


FIG. 3A

FIG. 4B

FIG. 4A

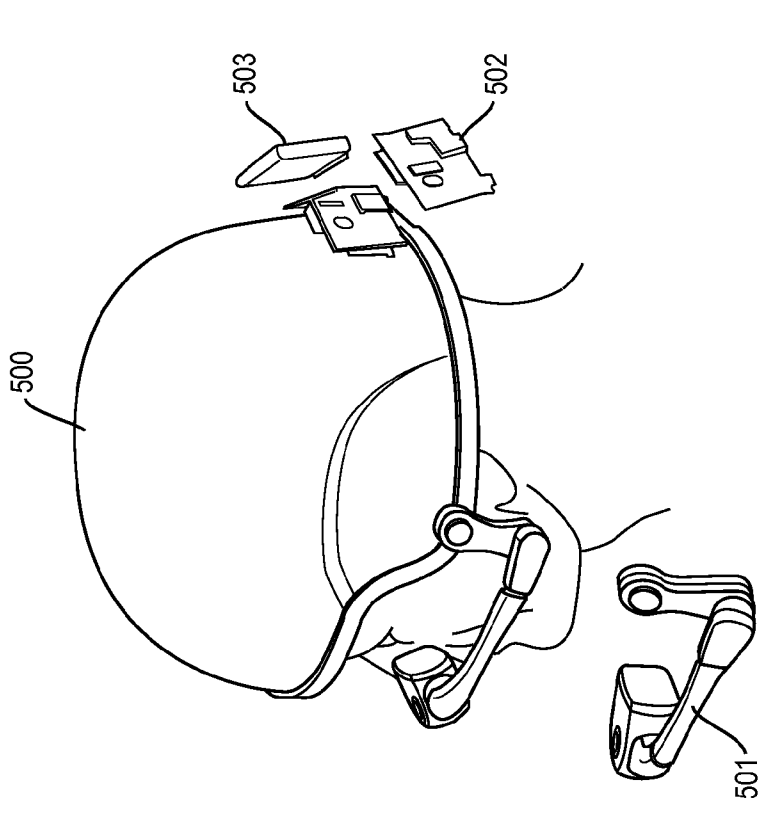


FIG. 5B

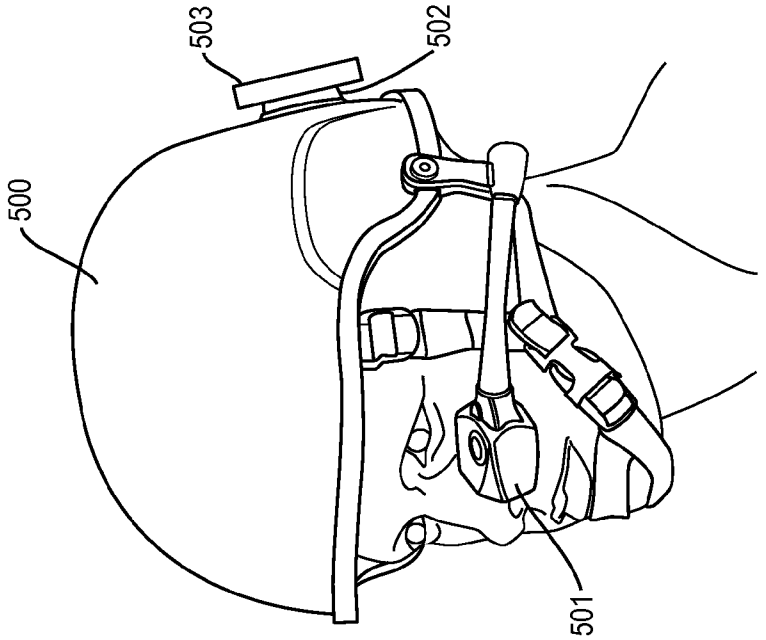


FIG. 5A

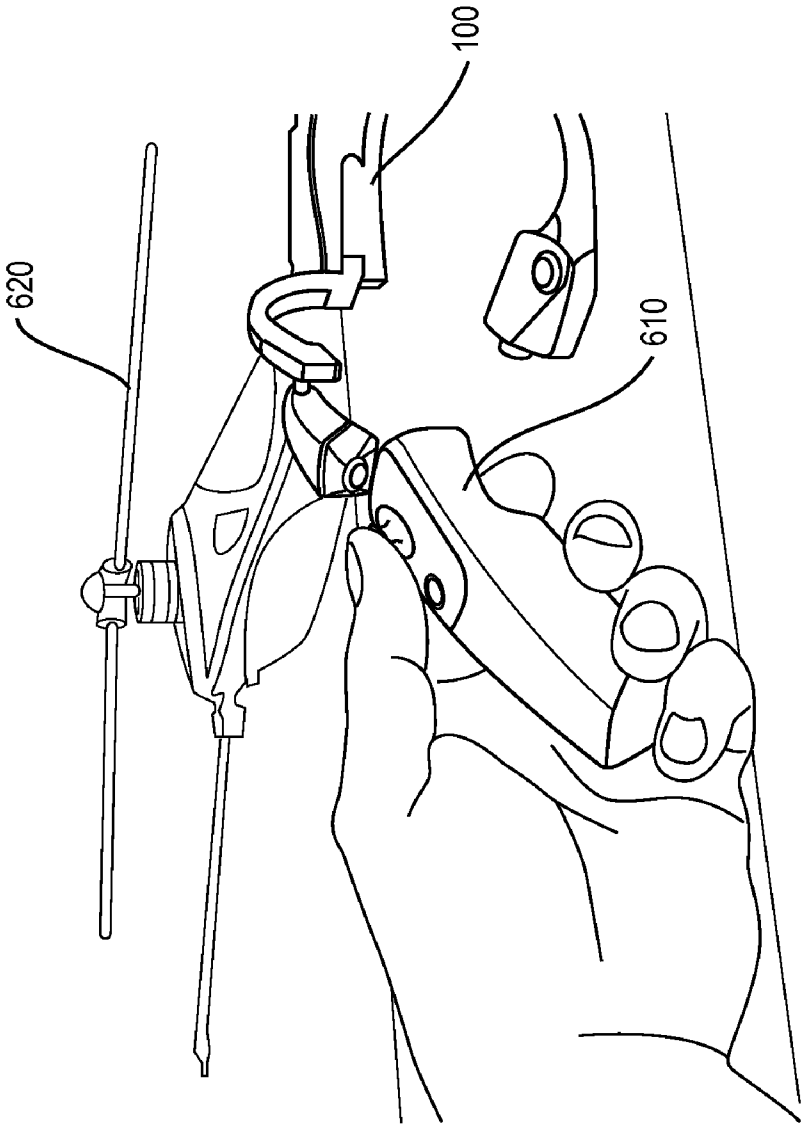


FIG. 6

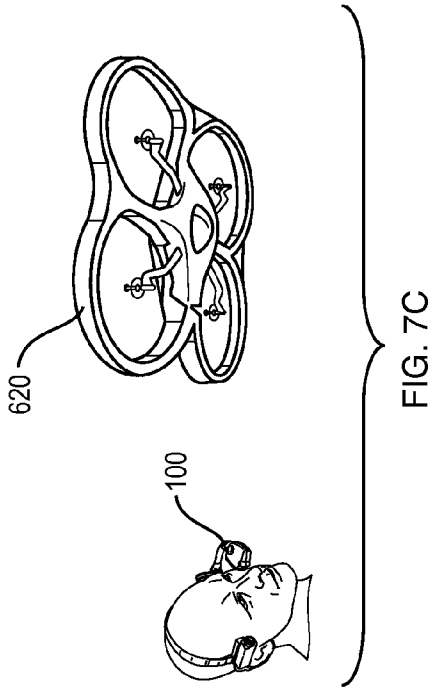


FIG. 7C

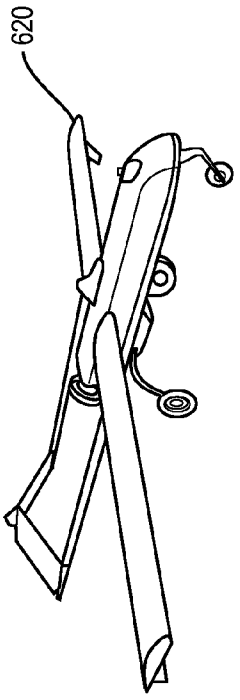


FIG. 7A

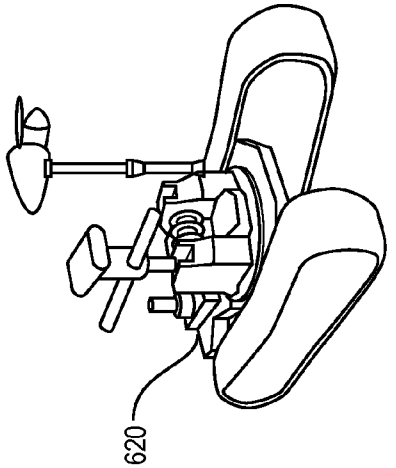


FIG. 7B

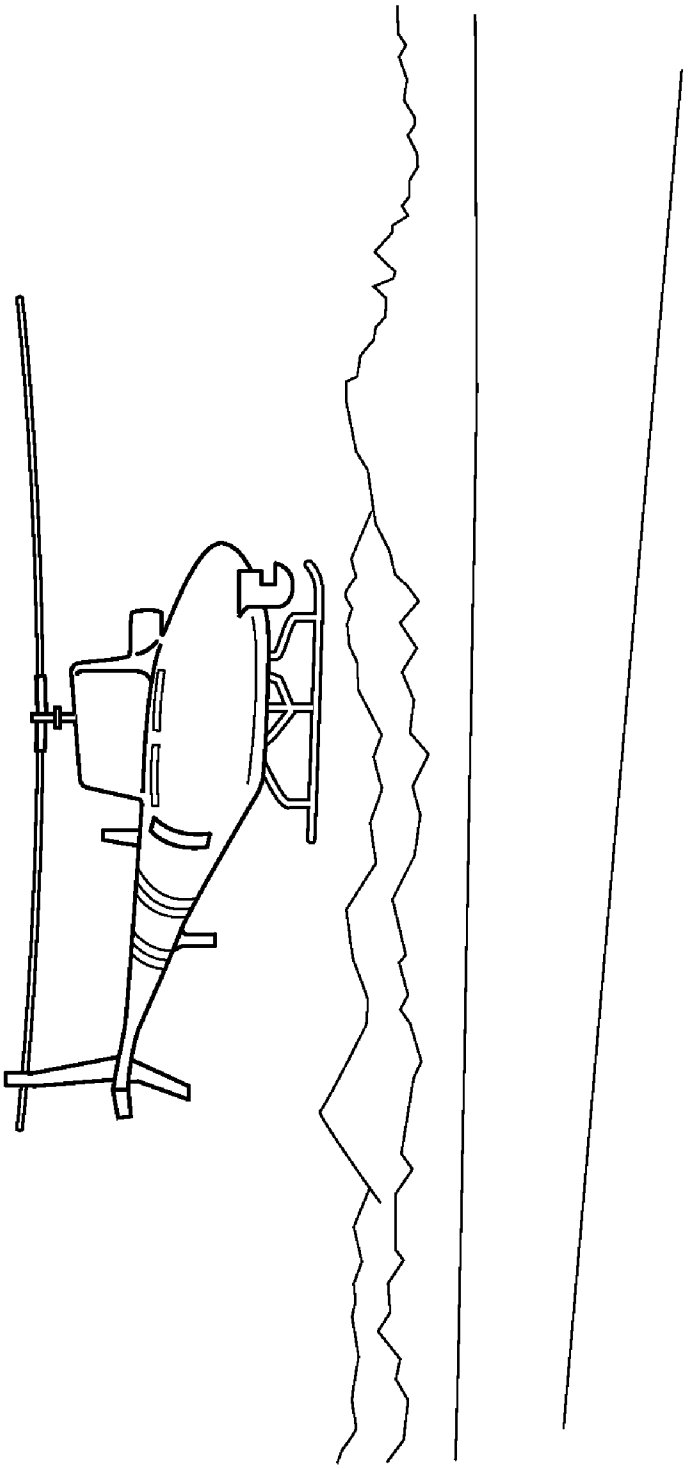


FIG. 8

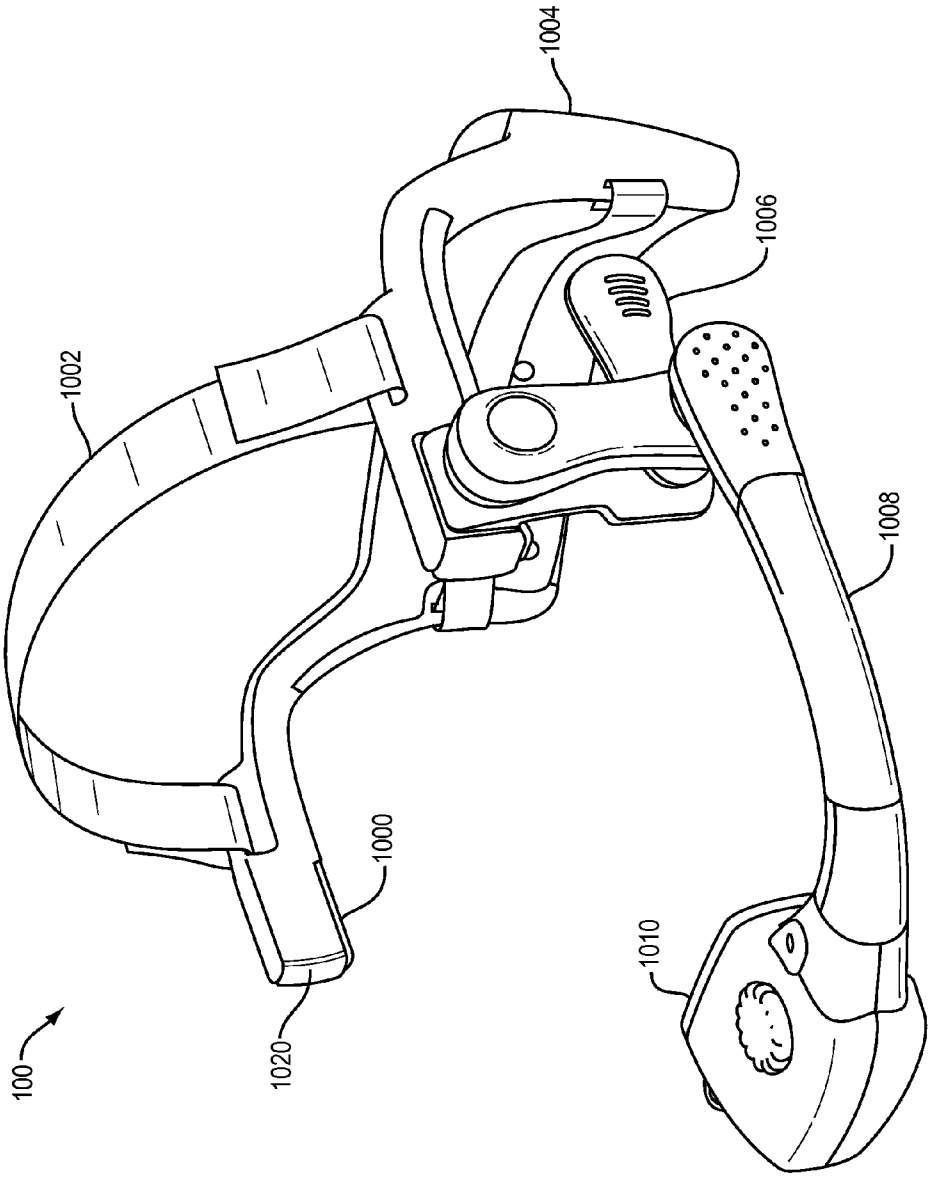


FIG. 9

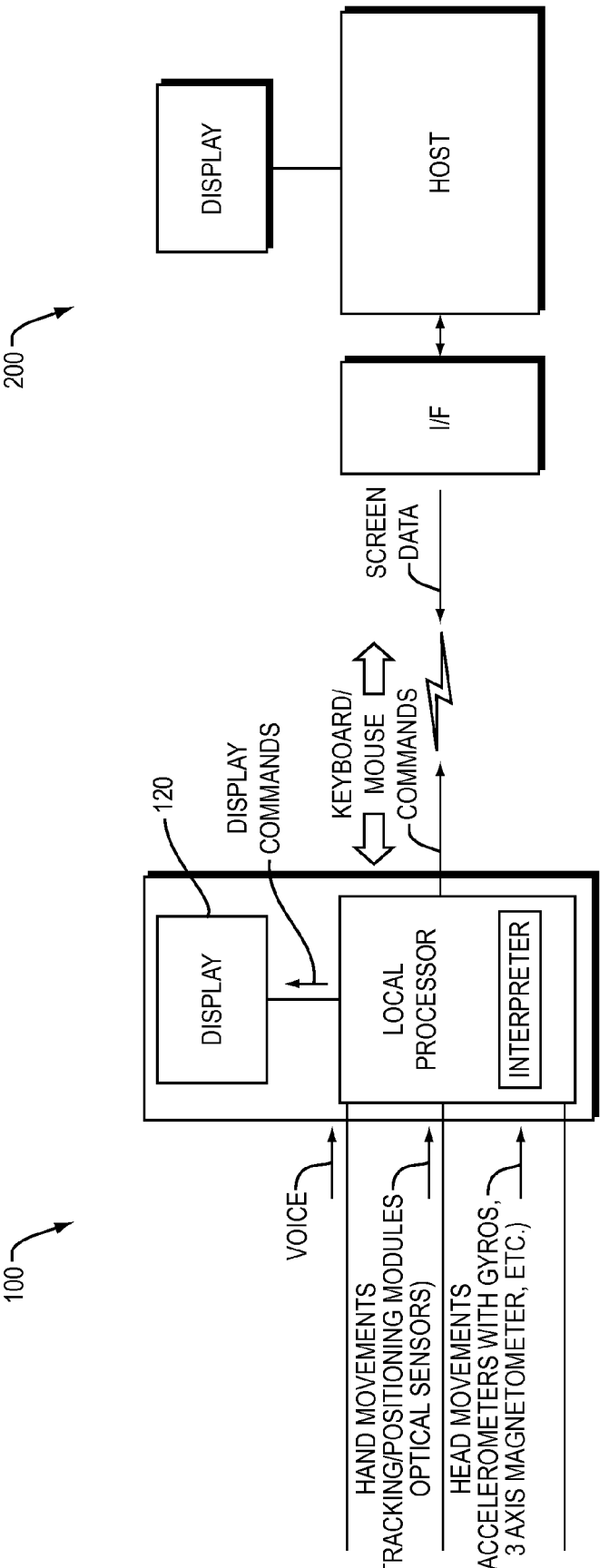

200
DISPLAY
HOST
I/F
SCREEN DATA
KEYBOARD/ MOUSE COMMANDS
DISPLAY COMMANDS
120
100
DISPLAY
LOCAL PROCESSOR
INTERPRETER
VOICE
HAND MOVEMENTS (TRACKING/POSITIONING MODULES OPTICAL SENSORS)
HEAD MOVEMENTS (ACCELEROMETERS WITH GYROS, 3 AXIS MAGNETOMETER, ETC.)

FIG. 10

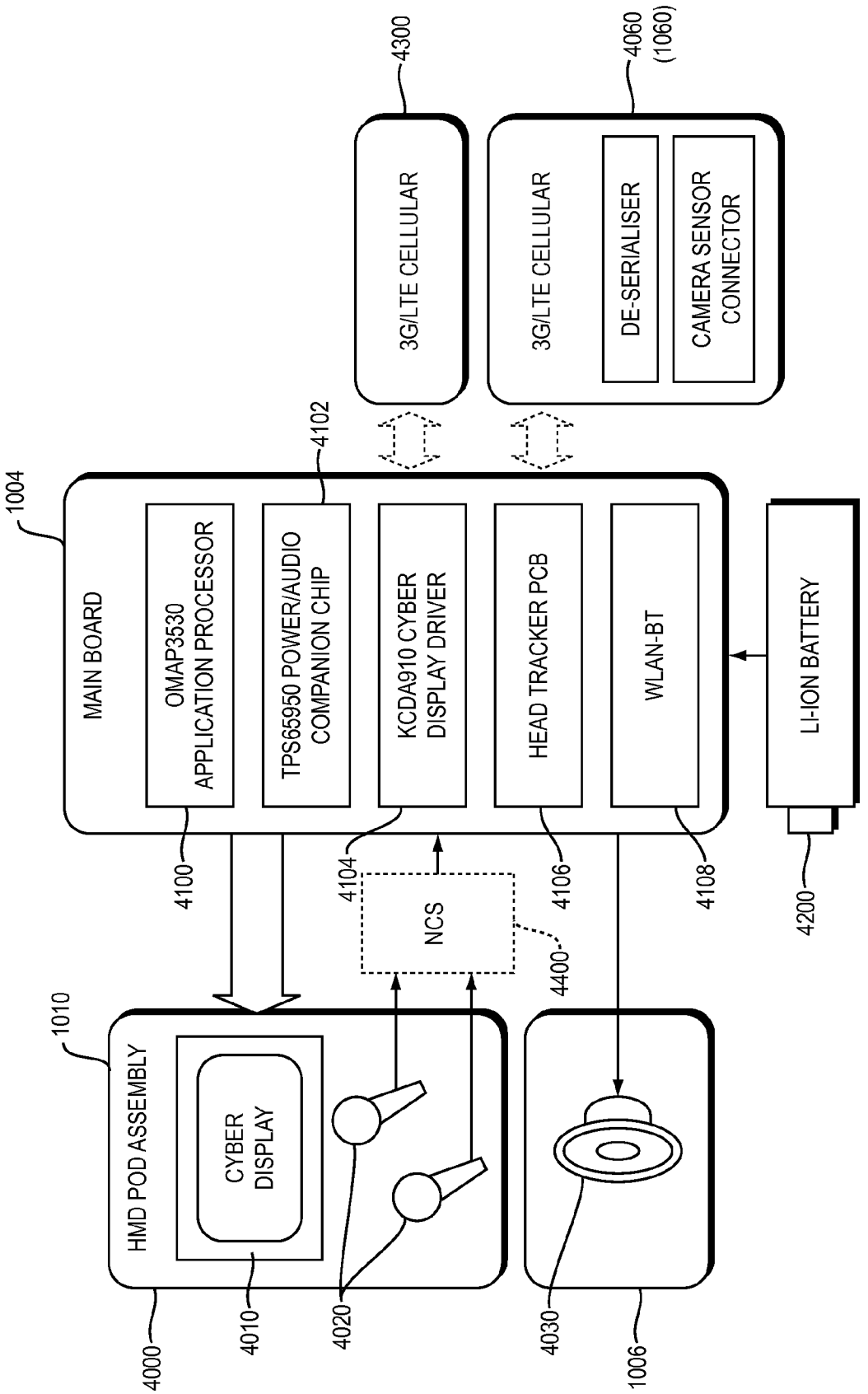

4300
3G/LTE CELLULAR
4060
(1060)
3G/LTE CELLULAR
DE-SERIALISER
CAMERA SENSOR CONNECTOR
1004
MAIN BOARD
4102
OMAP3530 APPLICATION PROCESSOR
TPS65950 POWER/AUDIO COMPANION CHIP
KCDA910 CYBER DISPLAY DRIVER
HEAD TRACKER PCB
WLAN-BT
LI-ION BATTERY
4100
4104
4106
4108
4200
NCS
4400
1010
HMD POD ASSEMBLY
CYBER DISPLAY
4000
4010
4020
4030
1006

FIG. 11

HEADSET COMPUTER THAT USES MOTION AND VOICE COMMANDS TO CONTROL INFORMATION DISPLAY AND REMOTE DEVICES

CROSS REFERENCE TO RELATED APPLICATION

[0001] This application claims the benefit of U.S. Provisional Patent Application Ser. No. 61/484,464, entitled "Hands-Free Enhanced Digital Eyewear" filed May 10, 2011. The entire contents of which are hereby incorporated by reference.

BACKGROUND

[0002] The present application relates to human/computer interfaces and more particularly to a wearable headset computer that accepts voice commands, tracks hand gestures and/or detects head movements to provide inputs to control software running within the headset computer and/or peripheral devices.

[0003] Small, portable electronic devices capable of storing and displaying large amounts of high resolution computer graphic information and video content continue to be increasingly popular. Devices such as the Apple iPhone™, Google Android™ and other smartphones represent a significant trend in convergence among mobile telephones, portable computers and digital media players (iPhone is a trademark of Apple Computer, Inc. and Android is a trademark of Google, Inc.). Although these smartphones typically include a display screen, the visual experience of a high-resolution, large format display cannot easily be replicated of because physical size limitations in the handheld form factor.

[0004] Other devices which provide improved functionality over smart phones are known by various names such as headset computers, video eyewear, head mounted displays with embedded computer processors, and the like. These devices include a frame or other support mechanism that is worn about the face and/or head, similar to a pair of eyeglasses and/or headphones. The frame houses a small, high-resolution microdisplay, optical lenses and other components needed to present an electronic image to the wearer. Circuitry in the headset computer can include display drivers, wireless interface(s), and fully functional personal computer systems. For further information regarding such headset computers, refer to corresponding patent application entitled "Handheld Wireless Display Devices Having High-Resolution Display Suitable for Use as a Mobile Internet Device", PCT International Application Number PCT/US09/38601 filed Mar. 27, 2009, the entire contents of which is hereby incorporated by reference.

[0005] Such devices are also further described in U.S. Application Nos. 61/300,611, filed on Feb. 2, 2010, titled "Head Mounted Video Eyewear With Accessory Mount;" 12/774,179, filed on May 5, 2010, titled "Remote Control Of Host Application Using Motion And Voice Commands;" 61/176,662, filed on May 8, 2009, titled "Remote Control Of Host Application Using Tracking And Voice Commands;" 61/237,884, filed on Aug. 28, 2009, titled "Remote Control Of Host Application Using Motion And Voice Commands;" 12/008,114, filed on Jan. 8, 2008, titled "Monocular Display Device;" and 12/008,104, filed on Jan. 8, 2008, titled "Monocular Display Device;" the contents of each of which are incorporated by reference in their entirety.

SUMMARY

[0006] A headset computer includes a microdisplay, multiple input devices such as a head tracking accelerometer and/or camera to detect movements such as head movements, hand motions and/or gestures, and audio processing circuits to detect voice commands. These inputs provide control over and operation of an application program running within the headset computer itself and/or peripherals associated with the headset computer.

[0007] In one implementation, voice, head motion, and/or hand gesture inputs are received from sensors located within the headset computer. A field of view into a 3-D virtual space is then determined from the voice, head motion and/or hand gesture inputs. Data representing the 3-D virtual space may be maintained either by a processor local to the headset computer device and/or a remote processor. The 3-D virtual space contains data representing one or more graphical objects. The graphical objects may include various elements such as computer desktops, application windows, digital images, photographs, 3-D models or other image data. The voice, head motion, and/or hand gesture inputs may determine both the field of view and scale factor to determine a viewpoint into the 3-D virtual space. Image data selected from the 3-D virtual space determined based on the field of view and scale factor, and then presented on the microdisplay.

[0008] The hand gesture, head motion and/or voice commands can be used not only to set the field of view and scale factor but also to select which of several graphic objects within the 3-D virtual space are selected for presentation on the microdisplay, in whole or in part. Thus by using these input commands the wearer of the headset may navigate through a large format 3-D space and completely control which portions of the 3-D space are seen on the microdisplay.

[0009] The head set computer may also include a light source and camera. The spectral properties of the light source and camera can be manipulated by the input commands to provide a synthetic vision function. In particular, the scene illuminated by the light source is detected by the camera, and in turn presented on the microdisplay. The light source and camera preferably operate in invisible electromagnetic portion of the spectrum including infrared, near infrared, ultraviolet, shortwave infrared or other invisible wavelengths. In this manner, the wearer of the headset computer has the ability to view scenes in an invisible portion of the spectrum.

[0010] The voice, head motion and/or hand gestures can control an operating wavelength for the light source and the camera, intensity of emissions from the light source, sensitivity of the camera, or other aspects of presentation of the synthetic vision function on the micro display.

[0011] The light source also can be used to determine range information. In particular, the light source may emit a high precision light, such as a laser light. A reflection of the light by an object can then be detected by a camera or other sensor capable of determining a round-trip time delay for the light. The headset computer can then determine a range to one or more points in physical space. This can provide not only a range to a given object but also, for example, a distance between two objects. In a case where the distances to more than two points are determined, a volume of space can be estimated.

[0012] The headset computer can be packaged to fit headgear such as a helmet. In one implementation, the processors and other electronic components can be disposed in a first housing, a second housing may carry the microdisplay on a boom and a third housing may include a power supply. One or more signal and/or power connectors are then provided between the various housings. The housings are separately attachable and detachable from the headgear. This permits retrofitting of a headset computer to helmets or other headgear that safety, security, and military personnel are accustomed to wearing. This then eliminates the need for them to fit completely assembled headsets inside of or on top of their headgear. In certain embodiments the housing for the micro display and/or processor may include noise cancellation circuits that may assist with operations in a noisy environment such as with a rebreather apparatus.

[0013] The headset computer may also control a remote vehicle and receive and display images returned from a camera on the remote vehicle. Control commands derived from voice, head motion, and hand gesture inputs can be used as a remote control for an associated vehicle. In other embodiments, the control command can alter position and/or other operating characteristics of a camera located on the vehicle. The image information returned from the camera on the remote vehicle may be displayed on the micro display.

[0014] A wearer of the headset computer can therefore experience using the remote vehicle as if he were a miniature pilot traveling on the vehicle, operating the vehicle while having a view of the scene around the vehicle displayed on the micro display. A wireless interface between the headset computer and the vehicle can be used to detect control commands for the vehicle.

[0015] The control commands may also control apparatus that are mounted on the vehicle such as payloads and other devices to be delivered by vehicle. In certain aspects, the voice command can generate complex commands for the vehicle such as "return to base", "circle at a specific altitude", and so forth.

BRIEF DESCRIPTION OF THE DRAWINGS

[0016] The foregoing will be apparent from the following more particular description of example embodiments of the disclosure, as illustrated in the accompanying drawings in which like reference characters refer to the same parts throughout the different views. The drawings are not necessarily to scale, emphasis instead being placed upon illustrating various embodiments.

[0017] FIG. **1** shows a headset computer that implements several functions described herein.

[0018] FIG. **2** depicts a viewport into a 3-D virtual space that may be manipulated with voice, head motions, or hand gestures.

[0019] FIGS. **3**A and **3**B depict a synthetic enhanced vision mode and range finding mode.

[0020] FIGS. **4**A and **4**B illustrate the synthetic vision mode in more detail, with FIG. **4**A being a black screen seen with the light source off, and FIG. **4**B a scene that is in view when the light source is enabled.

[0021] FIGS. **5**A and **5**B illustrate an implementation where components of the headset computer are individually attached to a helmet.

[0022] FIG. **6** illustrates a wireless joystick and mouse controller that can be used with the headset computer to control another device such as a vehicle.

[0023] FIGS. **7**A, **7**B and **7**C illustrate more examples of vehicles that can be controlled by the headset computer operating components contained thereon.

[0024] FIG. **8** shows a wireless joystick used with the headset computer.

[0025] FIG. **9** is a more detailed view of the headset computer.

[0026] FIG. **10** is a high-level block diagram of the functions performed by the headset computer.

[0027] FIG. **11** is a more detailed block diagram of the components of the headset computer.

DETAILED DESCRIPTION

[0028] FIG. **1** illustrates a headset computer **100** that is worn on the head of a person. The headset computer **100** is also known as video eyewear, head mounted display (HMD) with embedded computer, and by various other names. In this example, the headset computer **100** consists of apparatus, processor(s), and software that control the presentation of objects on microdisplay **120** and peripheral devices. The microdisplay **120** may be enclosed in a display housing **125** supported by a boom **132** that attaches to a frame **140**. The frame **140** may additionally contain a housing **150** to enclose further electronic components; in the example shown here the housing **150** is positioned on the back of the wearer's head. One or more speakers **135** deliver audio signals to the wearer's ears and similarly one or more microphones **137** either contained in the housing **125** and/or in other areas detect audio signals. A peripheral such as a camera **160** may be attached to a peripheral port **165**. The headset computer **100** may be further supported, for example, by a strap **170** on the wearer's head.

[0029] As will be explained in detail below in connection with FIGS. **9**, **10** and **11**, the headset computer **100** is a completely contained personal computer system including one or more data processor(s) for generating images on the microdisplay **120** and performing other functions. The processor(s) located in the headset computer **100** are capable of interpreting voice commands, detecting hand movements of the wearer (such as through the camera) **160** and/or detecting the wearer's head movements through accelerometers or other motion sensors. These inputs are then interpreted as commands to either the headset computer **100** and/or remote devices with which the headset computer can communicate, such as over wireless interfaces.

Viewport into 3-D Virtual Space

[0030] One function performed by the headset computer **100** is to provide a graphical viewport and/or window into a 3-D virtual space. The graphical viewpoint determines which information is presented on the microdisplay **120**. In this mode, for example, a movement by the wearer's head can bring a different section of that 3-D virtual space into view on the microdisplay **120**.

[0031] FIG. **2** illustrates this in some detail. Here the wearer **200** is shown relative to the virtual 3-D space **300**. The virtual 3-D space **300** has a number of windows **310**, **320**, **330**, **340**, **350** located within it and maintained by the processor(s) within the headset computer **100** or the external host. However, only a single one of the windows **340** is shown without shading, to indicate that it is an active window. The other windows **310**, **320**, **330** and **350** are shown grayed out. Thus while these other windows exist in the 3-D virtual space, the wearer **200** does not see the entire 3-D virtual space—rather the user **200** sees only the window **340** (or more typically, a

selected portion thereof) on the microdisplay **120**. The window **340** may be simply a viewport into for example, a graphic window in a windowed operating system. In the example shown here, the graphical object is a digital map image and the wearer is viewing only a portion of the map and not the whole map.

[0032] It will be understood that the 3-D virtual space may include various elements such as computer desktops, application windows, photographs, 3-D object models or any other type of digital image objects. It should be further understood that these image objects can be positioned next to, overlaid on or behind or beside one another in the 3-D virtual space.

[0033] The user can manipulate the various image objects by giving commands using the headset computer **100**. In one example, the user can ask for a level of enlargement of a particular area of interest within one of the objects. The location and size of the window area may be selected by the tracking of head motions, voice commands and/or hand gestures. For example, the user may specify a position and magnification and/or zoom level to be applied to a particular application software window. The result is similar to using a magnifying glass to look at something seamlessly over a large area, but by using the head tracker/gesture detector/voice input detection to zoom into an area being seen on the microdisplay **120** and at what level of magnification. Thus, using this feature the user can move his head left, right, up or down and then select a particular one of the image objects **300**, **310**, **320** through **340** to be active. In one example, the user **200** might from the position shown in FIG. **2**, turn his head to the right. This motion would then cause a new window **330** to then become the active window, with window **340** then becoming deactivated.

[0034] The user **200** can also issue commands to retain a piece of a large image that he wishes to magnify, freezing that portion on the screen and setting it aside and then going back and looking at another area of that image or even requesting another level of magnification for that other area. In this way, the user can view the same portions of an image at different levels of magnification and/or view different bits or pieces of a larger image at different levels of magnification and then switch between them by merely moving his head left or right, up or down

[0035] In yet another example, the wearer may issue voice commands to manipulate the position of the various image objects in the 3-D virtual space. For example, he may select an image object such as by moving his head, but then issue a voice command such as to "move object up" or "move object A behind object B". This causes the head tracker to then control the relative position of the selected image object(s) within the 3-D virtual space, rather than allowing him to navigate among a given single object within the 3-D space.

[0036] It will be understood that the wearer **200** thus has access to a virtual desktop that is in any form factor that can be represented in a 3-D virtual space, i.e. he may be working in a 360° surface that wraps around his head or may be given the impression that he is working in a 3-D space with a long depth of field.

[0037] In another example, the user **200** may turn his head to the lower left causing the window **350** to become active. This window may be a 3-D model of an object such as an engine. The user may then proceed to manipulate this 3-D model using voice, head tracking and/or hand gesture commands to manipulate the viewpoint in 3-D space. The wearer may also issue a command to manipulate the model itself, such as to say, "rotate object 90° horizontal" causing the representation of the motor to rotate in 3-D space.

[0038] The view of the displayed image on the microdisplay **120** does not require the user to be physically oriented as if he were looking in any particular direction. For example, the user may remotely view any image being virtually generated in a sitting or standing position as might be projected on a wall in a room, but yet that wearer may be himself physically oriented in other positions such as laying down.

Hands-Free Synthetic Vision

[0039] FIGS. **3**A and **3**B illustrate other modes and functions provided by the headset computer **100**. In this example, an emitter such as a light source **195** is located within the headset computer **100** typically within the same small housing as the camera **160**. If the wavelength of the camera **1600** and light source **195** are coordinated such that the camera is sensitive at the same wavelength emitted by the light source (s) then a synthetic vision function can result. For example, the light source may be infrared or ultraviolet, and if the camera is similarly sensitive in these wavelength regions, the resulting image from the camera can be seen on to the microdisplay **120**. The processor in headset computer **100** can also cause not just the image to be captured by the camera and directly viewed on the microdisplay **120**, but to also be stored in memory and/or be sent to a remote system or display.

[0040] Using the headset computer **100**, the wearer can thus experience hands-free synthetic vision that combines a synthetic view that is, for example, a far infrared view showing heat signatures of individuals or objects on the other side of a wall or other obstruction. An example of the same is shown in FIGS. **4**A and **4**B. With the light source **195** off, the image on the microdisplay is completely blank. However, with the light source **195** on, the infrared camera picks up the emission of the infrared and see an image that would not otherwise be visible. Using the headset computer **100**, with an integrated infrared light source and camera, the wearer is then able to see what is in the dark environment on the microdisplay **120**, but himself remain unseen by those relying only on natural non-enhanced sight. The individual **350** in the scene would not be able to detect the wearer's presence, because the infrared illumination from the light source would not be visible to the unaided eye.

[0041] As shown in FIG. **3**B, the light source may also include a laser range finder **198**. The laser **198** can be aimed by the user either by moving his head and/or by using voice commands. The range finder can be used for various functions such as finding the distance to an object in physical space, determining relative distance between two things. In the latter example, the user may aim his head at one object and measure a distance to that first object, and then move his head to otherwise aim the laser at second object. The user can then ask the headset computer **100** to solve the triangulation equation between his position and the two objects, thereby estimating a distance between the two objects.

[0042] In a further example, a volume of space can be estimated by the wearer aiming the laser at three or more points and asking the headset computer to figure out the distances between them. These functions can be useful in uses such as surveying or material estimating necessary. This can now be accomplished without the wearer actually moving

about or by using measuring implements other than the laser range finder as built into the headset computer **100**.

Components Retrofittable to Helmet

[0043] FIGS. **5**A and **5**B illustrate another example of the headset computer **100** packaged in a particular way to be retrofit onto existing headgear. In this implementation, headset computer electronics (including peripheral devices such as a camera and battery power source), the main processor and so forth may be packaged into one component housing **502**, and the boom with an integrated optical/visual/audio part packaged as another component housing **501**. The components **501**, **502** are individually attachable to an existing headgear familiar to the user. In this example shown, a helmet **500** may have two Velcro™ pads, with the electronics component **502** attached to one pad Velcro™, and a battery **503** separately packaged and attached to the other Velcro™ pad (Velcro is a trademark of the Velcro Corporation). The boom element **501** is also attached to the helmet, such as via a mechanical clip or fastener; the boom of course including the integrated microdisplay and microphones. A cabling system can connect the microdisplay and boom **501** to the electronics **502** and battery **503**. The boom **501** may be fixed and/or bolted to the left side or right side of the helmet, depending on user preference and also depending upon other equipment that the user may be operating.

[0044] Camera(s), laser(s), and other peripherals can also be mounted to the helmet **500**. Instead of requiring the wearer to wear a dedicated headset under the helmet, this packaging approach can implement a headset computer functionality without the user having to become comfortable with new headgear. In addition, operation with certain types of headgear (such as a rebreather) is not affected. This particular end use may be improved if the on board electronics also provide for noise cancellation. For example, if the wearer is using a rebreather, the rebreather tends to make a lot of background noise that would otherwise interfere with voice inputs or sound recording. The on-board electronics may include noise cancellation circuits or programming that eliminate the background noise of the rebreather. A similar approach can be used to cancel out other background noises to allow for clearer recording of voices or other sounds.

Headset Computer Controls Remote Vehicle, Receives and Displays Images from and to the Remote Vehicle

[0045] In yet another implementation, the voice, head motion and/or hand gesture inputs received from the sensors located within the headset computer **100** can be used to derive a remote control command. That control command can then be sent over a wireless interface to control a remote vehicle robot, or other object. In this end use, the input device may also further include a wireless joystick and/or mouse to provide further inputs to control the vehicle.

[0046] In one example, a voice input to the headset computer can generate a control command to control the path of the vehicle. Voice commands, such as "turn right", "turn left", "move forward", "move backward", "stop" and so forth can be included in the processing capabilities of the headset computer **100**. Similarly, head tracking inputs can generate a control command to control the path of the vehicle, or more commonly the direction of the camera on the vehicle. In this way, the user can obtain an experience that he is physically located on the vehicle. This is accomplished by having the camera on the vehicle transmitting video preferably wirelessly back to the headset computer. The video received at the remote vehicle can then be displayed on the display within the headset computer.

[0047] In yet another example, a wireless handheld controller **610** such as that shown in FIG. **6** can be used with the headset computer **100** to control the path position, attitude and/or direction of the vehicle **620** more naturally.

[0048] Using this arrangement, a person can control a vehicle such as an unmanned aerial vehicle (FIG. **7**A), unmanned ground vehicle (FIG. **7**B) or a toy (FIG. **7**C) and so forth. This eliminates problems with prior art that simply operates a remote vehicle **620** with a videogame type of controller that requires almost total user attention and both hands to operate. By utilizing wireless joystick controller **610** in combination with the wearable headset computer **100** that can obtain head motion, voice and hand tracking commands, the control and electronic processing capabilities of the headset computer can give anyone control such as a soldier, policeman, fire or industrial worker control over one or more remote systems or vehicles **620** in simple and natural way.

[0049] In the absence of a separate user input device, the camera on the headset computer **100** may detect the user's hand gestures as control inputs. The wearer can also give speech commands to give the vehicle certain commands. For example, if the wearer says "freeze", that can be detected by the headset computer which then translates the spoken command into one or more commands to control the flight path of the unmanned aerial vehicle, to stop doing everything else and simply hover or follow a circular flight path around a current point of interest.

[0050] In other examples a voice command such as "return to base" can cause the vehicle to follow a complex programmed flight path. Another example can be "circle at a specific altitude" which can cause the vehicle to generally follow a geo-stable circle around its present location. This can alleviate the user from tediously having to continuously provide commands via the handheld controller.

[0051] Other voice commands and hand held commands can be used to control other aspects of the vehicle's capabilities, performance and/or path of travel.

[0052] In one embodiment, the vehicle **620** may itself contain a camera that transmits its video output wirelessly back to the headset computer **100**. Video carried back to the headset computer **100** is then displayed on the microdisplay **120**. The wearer's head movements and/or gestures may then be used in a natural way to control the position, attitude, pan, zoom, magnification, light spectral sensitivities or other capabilities of the camera on the remote vehicle. The user's head movements can then be tracked by the on board electronics of the headset computer **100** and translated by the headset computer into commands that are sent back to aim the camera of the unmanned vehicle. As an example, if the wearer looks to the left, that motion is detected by the head tracker in the headset computer, translated into a camera "move left" command. That "move left" command is then sent wirelessly to the remote vehicle, causing the camera on the remote vehicle to pan to the left.

[0053] By returning the video stream back from the vehicle and displaying it on the microdisplay gives the wearer a visual experience as if he were, for example, a miniature pilot inside an unmanned aerial vehicle.

[0054] In yet another function, the user can, for example, use speech commands to control other peripherals that the vehicle itself might contain. An unmanned aerial vehicle such

as shown in FIG. **8**, may carry a payload such as a camera or other sensor to be dropped at a remote location. These payloads, weapons or other objects that the vehicle is capable of delivering can be controlled by the user of the headset computer **100**. Control over these payloads can be implemented regardless of what the vehicle itself is being commanded to do.

System Description

[0055] FIG. **9** shows a wireless headset computer **100** (also referred to as a video eyewear device **100**) that incorporates a high resolution (VGA or better) microdisplay element and other features described below. Headset computer **100** typically includes many different types of integrated circuits including a microprocessor (single or multi-core), one or more wireless interfaces, associated memory or other storage devices, one or more cameras (optical sensors) and/or various sensors. These sensors may include audio input and/or output devices, such as one or more microphone(s) input and output speaker(s) the sensors may include geo-positional sensing, 3 axis to 9 axis degrees of freedom orientational sensors (such as a digital magnetometer), atmospheric sensors, health condition sensors, GPS, digital compass, pressure sensors, environmental sensors, energy sensors, acceleration, position, attitude, motion, velocity or optical sensors, and cameras (visible, infrared, etc.). Further circuits such as additional wireless radios, auxiliary lighting, range finders, or the like, and/or an array of sensors may be embedded in and/or attached to the device. Also typically located within the device **100** are a peripheral mount or mounts such as a "hot shoe" (not shown in FIG. **9**) for mounting optional accessories such as cameras or additional sensors. The camera(s), motion sensor(s) and/or sensor(s) are used to track the motion and/or position of the user's head, hands and/or body in at least a first axis **111** (horizontal), but preferably also a second (vertical), a third (depth), a fourth (pitch), a fifth (roll) and a sixth (yaw).

[0056] The headset computer device **100** can be used in various ways. It can be used as a completely contained, head-mounted fully functional portable personal computer/smart phone with full connectivity to external computers and networks through a short and/or long-range wireless links such as Bluetooth, WiFi, cellular, LTE, WiMax or other wireless radios.

[0057] Device **100** can be also used as a remote display for a streaming video signal provided by a remote host computer. The host may be, for example, a laptop, cell phone, Blackberry, iPhone™, or other computing device having lesser or greater computational complexity than the device **100** itself. The host then provides information to the device **100** to be displayed. The device **100** and host are connected via one or more suitable wireless connections such as provided by the Bluetooth WiFi, cellular, LTE, WiMax or other wireless radio link. The host may itself be further connected to other networks such as through a wired or wireless connection to the Internet.

[0058] While what is shown in FIG. **9** is a monocular microdisplay presenting a single fixed display element supported on the face of the user with a cantilevered boom, it should be understood that other mechanical configurations for various video eyewear devices **100** are possible.

[0059] In the FIG. **9** implementation, headset computer **100** includes generally a frame **1000**, a strap **1002**, a back section **1004**, speaker **1006**, cantilever or arm **1008**, and microdisplay subassembly **1010**. On one side of the device **100** opposite the cantilever arm **1008** is a peripheral port **1020**. The peripheral port **1020** provides corresponding connections to one or more peripheral devices, so a user can removably attach various accessories to the device **100**. As an example port **1020** provides a mechanical and electrical accessory mount such as a hot shoe. Wiring carries electrical signals from port **1020** through, for example, the back portion **1004** to circuitry disposed therein. Hot shoe **1020** can operate much like the hot shoe on a camera, automatically providing connections to power the accessory and carry signals to and from the rest of device **100**. Various types of accessories can thus be used with port **1020** to provide the hand movements, head movements, and or vocal inputs to the system, such as but not limited to microphones, positional, orientation and other previously described sensors, cameras, and the like.

[0060] FIG. **1** was a view of the headset computer **100** as worn on the head of a user where an accessory **1060** has been placed in the hot shoe port **1020**. This accessory **1060** is a self-contained camera (or other motion sensor) assembly. The camera **1060** can include both audio and video sensing, recording, and light emission capabilities in a package similar to a "bullet cam". It can be connected to the remaining components in device **100** via built in wiring in the back section **1004** (as in the case of the speaker previously described) or can be wirelessly connected via a Bluetooth™ or WiFi™ connection. The camera **1060** may not necessarily be a video camera, but may also detect infrared, ultraviolet, or other wavelengths. The camera **1060** can also include a user adjustable auxiliary light source. With the light source, the camera **1060** can also be used as a light source as desired without activating the camera portion.

[0061] The camera, motion tracking and audio inputs to the device **100** are interpreted as user commands in various ways to control operation of the local processor, the microdisplay, or the external host.

[0062] Head movement tracking and/or vocal commands can also be provided by the user **1050** to manipulate the settings of camera **1060**. For example, a user vocal command, such as "zoom" or "pan", can be recognized by the local processor and cause the camera **1060** to zoom in or telephoto out.

[0063] FIG. **10** is a block diagram showing more detail of the device **100**, optional host **200** and the data that travels between them. The device **100** receives audio signals input via the microphone, hand movements or body gestures via positional and orientation sensors, the camera or optical sensor(s), and head movement inputs via the head tracking circuitry such 3 axis to 9 axis degree of freedom orientation sensors. These are translated by software in a processor local to the device **100** into commands. These commands may then be interpreted by a processor internal to the device **100** to control aspects of the presentation of information on the microdisplay or other objects such as a peripheral or remote vehicle. The commands may also be sent over the Bluetooth or other wireless interface **150** to the host **200**. The host **200** then interprets these translated commands in accordance with its own operating system/application software to perform various functions, and or returns information to the device **100**. In one example, the device **100** and/or host **200** maintain a 3-D virtual space into what graphic objects are stored.

[0064] Among the commands that can be carried out on the local processor and/or the remote host **200** is one to select a field of view **300** within the virtual display. Thus, it should be

understood that a very large format virtual display area might be associated with operating system or application software running on the device **100** or on the host **200**. However, only a portion of that large virtual display area within the field of view is returned to and actually displayed by the remote control display device **120** as selected by the voice, hand gestures, or head motion commands.

[0065] FIG. **11** is a simplified high level block diagram of a non-limiting example embodiment of the headset computer device **100**. The system electronics can be placed on or in the frame in an appropriate location (such as back section **1004**) and include an Open Media Application Platform (OMAP) as the local processor **4110**, a power/audio companion chip **4102**, a display driver **4104**, a head tracker circuit board **4106**, and wireless LAN/Bluetooth interface **4108**. Also located in the housing is a power source, such as a lithium ion battery **4200**.

[0066] The device **100** may also include an eye pod assembly **4000** that includes the aforementioned microdisplay **4010** (e.g. the microdisplay **1010** and boom **1008** of FIG. **2**A), and one or more microphones **4020**. One or more speakers **4030** are positioned in the HMD housing earpiece near the user's ear (item **1006** in FIG. **9**). The head tracker circuitry **4106** may include circuits to determine head movements and gestures detected by sensors in the device **100**, such as lateral movements along and rotation gestures around the X, Y and Z axes using Hall effect sensors, MIM diodes, accelerometers, gyros and/or transducers or other sensors as mentioned above.

[0067] Device system **100** may also receive inputs from external input devices such as a wireless mouse, track ball, or keyboard that may be wirelessly connected through the Bluetooth interface **4108**.

[0068] Software in the WLAN/BT front end **4108**, the OMAP **4100** and/or host **200** may be used to interpret hand gestures detected by the camera or other sensors. A camera board **4060** may optionally provide video input, as well.

[0069] The OMAP processor **4100** may include a central processing unit, and on-chip memory such as Random Access Memory (RAM) that may include non volatile memory and/or Read Only Memory (ROM). The OMAP may be a Texas Instruments model OMAP **3530** processor or newer version sold by Texas Instruments, Inc. and using a multimedia processor. The OMAP **4100** may typically execute an embedded system such as operating a particular version of MicroSoft Windows®. The OMAP **4100** is generally a more powerful, and more power consuming processor than the WLAN/BT interface **4108**.

[0070] In this example, a TPS **65950** power/audio companion chip, also available from Texas Instruments, provides audio, USB, keypad control and battery charging functions to the system.

[0071] The WLAN/BT interface **4108** may be a model LBEE 1W8 NEC-interface circuit, a Bluetooth circuit such as available from CSR, Ltd. of Cambridge, United Kingdom or other radio module with similar or greater capabilities.

[0072] The display driver may be a model KCD-A **910** display driver available from Kopin Corporation of Westborough, Mass.

[0073] The microdisplay **4010**, also available from Kopin, can include models CyberDisplay 230K, WQVGA, VGA, WVGA, SVGA or other manufactures' acceptable microdisplays.

[0074] An NCS module **4400** takes raw microphone signal data as input, and outputs audio data with background noise removed. It produces an audio signal to the audio companion chip **4102** and from there to the OMAP processor **4100**. Voice recognition is performed in software on the OMAP processor **4100**, using the cleaned up microphone signals as fed in by the NCS **4400**.

[0075] The teachings of all patents, published applications and references cited herein are incorporated by reference in their entirety.

[0076] While this disclosure has described several example embodiments, it will be understood by those skilled in the art that various changes in form and details may be made therein without departing from the scope of the invention encompassed by the appended claims.

What is claimed is:

1. A method performed by one or more processors disposed within a headset computer device, comprising:
- receiving image data from a remote processor;
- receiving voice, head motion, and/or hand gesture input(s) from sensors located within the headset computer;
- determining a viewport into a three-dimensional (3D) virtual space from the voice, head motion and/or hand gesture inputs;
- selecting an image portion from the image data depending on the field of view and scale factor; and
- presenting the image portion on a microdisplay disposed within the headset computer.

2. The method of claim **1** further comprising:
- maintaining a representation of multiple graphical objects within the 3D virtual space.

3. The method of claim **1** further comprising:
- communicating with a host processor disposed outside of the headset computer device to maintain a representation of multiple graphical objects within the 3D virtual space.

4. The method of claim **2** further comprising:
- detecting a head motion to move the field of view within the 3D virtual space.

5. The method of claim **2** further comprising:
- detecting a voice input to move the field of view within the 3D virtual space.

6. The method of claim **2** wherein the graphic objects are selected from a group consisting of an application window, digital image, etc.

7. A method performed by one or more processors disposed within a headset computer device, comprising:
- operating a light source located within the headset computer according to voice, head motion, and/or hand gesture input(s) detected by sensors located within the headset computer;
- operating a camera located within the headset computer according to voice, head motion, and/or hand gesture input(s) detected by sensors located within the headset computer;
- the light source and camera operating at a non-visible wavelength; and
- displaying the image data on a microdisplay disposed within the headset computer.

8. The method of claim **7** wherein the light source and camera operate at an infrared (IR) or ultraviolet (UV) wavelength.

9. The method of claim **7** additionally comprising:
- determining a field of view and scale factor from the voice, head motion and/or hand gesture inputs; and

selecting an image portion to be presenting on the microdisplay from the image data depending on the field of view and scale factor.

10. The method of claim **7** additionally comprising:

receiving voice, head motion and/or hand gesture inputs and in response thereto, controlling an operating wavelength for the light source and the camera.

11. The method of claim **7** additionally comprising:

receiving voice, head motion and/or hand gesture inputs and in response thereto, controlling an intensity of emissions from the light source.

12. The method of claim **7** additionally comprising:

receiving voice, head motion, and/or hand gesture inputs, and in response thereto, controlling a sensitivity of the camera.

13. The method of claim **7** wherein the light source is a laser range finder, and the response includes range information.

14. The method of claim **7** further comprising:

determining range information for at least two points;

determining distance information between the two points from the range information; and

displaying the distance information on the microdisplay.

15. The method of claim **7** wherein the range information is determined for multiple points and further comprising:

determining volume information for a space defined by the multiple points; and

displaying the volume information on the microdisplay.

16. A headset computer system comprising:

a first housing for a processor;

a second housing for a microdisplay;

a third housing for a power supply;

one or more signal and power connections between the first, second and third housings; and

fasteners for separately attaching and detaching the first, second and third housing to and from headgear.

17. The system of claim **16** wherein the headgear is a helmet.

18. The system of claim **16** wherein the fasteners additionally comprise hook and loop fasteners.

19. The system of claim **16** wherein the first housing further encloses noise cancellation circuits.

20. The system of claim **16** wherein the second housing further comprises a boom for supporting the microdisplay.

21. A method performed by one or more processors disposed within a headset computer device, comprising:

receiving image data from a remote vehicle;

receiving voice, head motion, and/or hand gesture input(s) from sensors located within the headset computer;

determining a control command from the voice, head motion, and/or hand gesture inputs; and

sending the control command to a remote vehicle.

22. The method of claim **21** wherein the control command controls a camera located on the vehicle.

23. The method of claim **22** additionally comprising:

in response to sending the control command,

receiving image data from the camera located on the remote vehicle;

presenting the image data on a microdisplay disposed within the headset computer.

24. The method of claim **21** wherein the control command controls operation of the vehicle.

25. The method of claim **24** wherein a voice input generates a control command to control a path of the vehicle.

26. The method of claim **25** wherein a head motion input generates a control command to control the camera on the vehicle.

27. The method of claim **26** wherein the control command controls position, attitude and/or direction of the camera.

28. The method of claim **27** additionally comprising:

receiving an input from a handheld controller; and

determining a command to control a path of the vehicle from the hand held controller input.

29. The method of claim **28** wherein the voice input generates a hover command to cause the vehicle and camera to remain fixed in position.

* * * * *

MarkerMouse: Mouse Cursor Control Using a Head-Mounted Marker

Rados Javanovic and Ian Scott MacKenzie

Dept. of Computer Science and Engineering
York Univeristy
Toronto Ontario Canada M3J 1P3
{rados,mack}@cse.yorku.ca

Abstract. We propose *MarkerMouse*, an inexpensive method for controlling the mouse cursor using a web cam and a marker placed on the user's forehead. Two modes of cursor control were compared: position-control and velocity-control. In position-control mode the cursor is positioned where the user's head is pointing. In velocity-control mode the mouse cursor moves in a constant speed in the direction the user's head is pointing. In an experiment designed according to ISO 9241-9, we found a mean throughput 1.61 bps in position-control mode. Throughput was 34% less, or 1.07 bps, in velocity-control mode. We explain how from the marker image we control the mouse cursor position and reduce noise in our computations.

Keywords: User interfaces, cursor control, web cam, marker tracking, head position tracking, head-operated mouse, mouse emulation, ISO 9241-9.

1 Introduction

Interacting with desktop computers typically engages both hands of the user. This is not a problem for most people, but for people with certain disabilities this scenario may be untenable. Fortunately, there is considerable research on providing efficient access to computers for disabled users. Several interesting technologies and solutions have emerged such as blink-based text entry [7] and voice-based mouse cursor control [4].

People with severe disabilities who cannot use, or have limited use of, their hands and who cannot speak might not be able to control an on-screen cursor. Sometimes the only option is to use the motion of the head to control the cursor. However, commercial solutions are typically expensive. Origin Instruments' *HeadMouse Extreme*, for example, costs about one thousand U.S. dollars (http://www.orin.com). A solution that costs less would be highly desirable. A new method for head cursor control is proposed in this paper.

Our proposed method does not require expensive hardware. It uses a common web camera, a printout of a marker (Fig. 1a) mounted on the user's forehead, and the software to process the marker orientation and move the cursor. Notable, as well, is that the user-to-computer interface is wireless, making it comfortable to use. We refer to this head-mounted marker system for mouse cursor control as *MarkerMouse*.

K. Miesenberger et al. (Eds.): ICCHP 2010, Part II, LNCS 6180, pp. 49–56, 2010.

(a)  (b) 

Fig. 1. *MarkerMouse* (a) printout (b) participant using *MarkerMouse*

The system is intended for people with restricted motor-control in the arms since only head motion is required to move the mouse cursor. The rest of this paper is organized as follows. In the next section, we review related work. Following this, we describe our system and provide details on the interface and noise reduction algorithm. Finally, we present an evaluation of our system, designed according to the ISO 9241-9 standard for evaluating non-keyboard input devices [5].

2 Related Work

Most uses of head-mounted markers are in virtual worlds to connect the physical position and movements of a real person with the virtual objects they interact with. For example, Knoeflein et al. [6] used a head-mounted marker in a virtual world gaming application. An IR marker was attached to a head-mounted camera and tracked to determine the user's head pose while playing ping-pong. Applications for cursor control tend to use face and feature tracking, rather than makers [2, 11]. In these applications, the focus is on the algorithms rather than the interaction. The use of a head-mounted marker greatly simplifies the software, since the geometric features in the marker are both precise and user-independent.

3 MarkerMouse

In this section, we describe the *MarkerMouse* interface, the position-control and velocity-control modes of cursor control, button clicks, and noise reduction.

3.1 Interface Details

The *MarkerMouse* head-mounted marker system is simple to assemble. The user simply prints the marker, positions it in some manner on her forehead, installs the software, and performs a simple calibration procedure. The marker can be attached to the user's glasses, a hat, a bandana, or anything that might be comfortable.

For *MarkerMouse* to work, the user must sit in front of a web camera so that the marker on the user's forehead is clearly visible (Fig. 1b). The orientation of the marker with respect to the camera when the user is looking straight at the monitor is the "neutral orientation". The neutral orientation is sampled and used in computing the movement of the marker and, hence, the motion of the mouse cursor.

3.2 Image Processing

The software linking the marker orientation to mouse cursor movement processes the marker image as input by the web camera. The first task is to compute the marker's normal vector in 3D space. Since the marker has a distinct, non-symmetric pattern on a square background it is possible to take an image captured by the web camera and retrieve the 3D plane on which the marker is sitting. By finding this plane, we calculate the normal vector of the marker. This is the neutral orientation. When the marker undergoes movement, we take the angle between the new normal vector and the neutral orientation and use this to determine where to move the cursor. To compute the marker's normal vector we used a software library called *NyARToolkit*[1].

3.3 Button Clicks

Our primary focus is on emulating mouse cursor control, apart from the functionality of clicking. To emulate a mouse button click, our system uses any key press on the system's keyboard. For accessible computing, depending on the disability, users may have enough manual facility to press a switch, for example using their hand. For users with severe disabilities who are not able to use their arms, the mouse button clicks can be emulated in different ways, for example, using blinks [7, 10], intentional muscle contractions [1, 3] or a sip-and-puff straw [12].

Position-Control and Velocity-Control

We evaluated two different modes of cursor control: position-control and velocity-control. In position-control mode, the marker orientation on the user's head determines the final position of the cursor on the screen. For example, if the user looks straight at the monitor the cursor will be in the center of the screen. If the user looks left the cursor will be on the left side of the screen, and if he then looks straight ahead again the cursor will be in the center again. This is similar to the operation of a stylus on a tablet or a finger on a touchscreen.

In velocity-control mode, the head orientation determines the direction to move the cursor. If the user looks straight at the monitor, the cursor is motionless. If the user looks left the cursor starts moving left and continues moving left while the user continues to look left. Movement is initiated when the computed displacement is equivalent to 200 pixels of mouse displacement. The rate of cursor movement is constant at 150 pixels per second. On the test system, this amounts to 10% horizontally and 16% vertically of the display extent per second. In velocity-control mode, we constrained the mouse movement to eight directions: up, down, left, right, and along the four diagonals.

[1] Available from Virtual Worldlets (http://www.virtualworldlets.net/).

3.4 Noise Reduction

Having the mouse cursor position directly dependent on the orientation of the marker may cause the cursor to jump around due to camera noise and inaccuracies in computing the marker's transformation. We used two methods to minimize noise.

The first method uses dampening. Instead of using the marker's normal vector to position the cursor directly, we compute the normal vector as a tentative cursor position. For each step the cursor moves accordingly, but with added dampening, as follows. Before the cursor is moved, a vector is computed in the direction of movement. The length of the vector is the distance between the two positions. This vector is multiplied by a dampening constant, D ($0 < D < 1$), and then the cursor takes a step accordingly:

$$NewMousePos = MousePos + (TargetPos - MousePos) \times D \quad (1)$$

In our application, we set D to 0.5. For a large head movement, the cursor moves relatively fast. For a small head movement, the cursor moves somewhat more slowly. This helps reduce jitter in the cursor movement due to small variations in the computation of the marker's normal vector.

The second method uses a history of the mouse cursor position. Previous cursor positions are recorded and used in computing the new position. Each new cursor position was determined using the computed, new position averaged with the last 10 mouse positions. This sliding-average approach helps minimize big spikes in the change of the marker's normal vector due to noise.

Applying these two methods together, the mouse cursor behaviour is more stable.

4 Method

While the design and initial testing of *MarkerMouse* system were promising, a proper evaluation with a pool of participants is required to validate the design. Our experimental evaluation tested how well participants could execute point-select tasks with our *MarkerMouse* in each mode of cursor control. The evaluation also included a generic mouse as a baseline condition.

Participants. Nine able-bodied participants were recruited from the local university campus. Five were male, four female, Ages ranged from 22 to 27 years. All participants used a computer on a regular basis.

Apparatus (Hardware and Software). Participants worked on a Hewlett Packard *Pavilion dv7* laptop computer. The system includes an embedded web camera on the top of the display. The resolution of the display was 1440×900 pixels. The operating system was Microsoft *Vista* (Home Edition).

Two separate software utilities were used. For demonstrating *MarkerMouse* and to provide participants the opportunity to practice, a simple program presenting an array of circular targets was created. See Fig. 2a. The targets were of random diameters in the range 30-80 pixels and were randomly scattered about the display. The targets could be selected in any order. When a target was clicked on, the colour changed to green to give visual feedback of a successful selection.

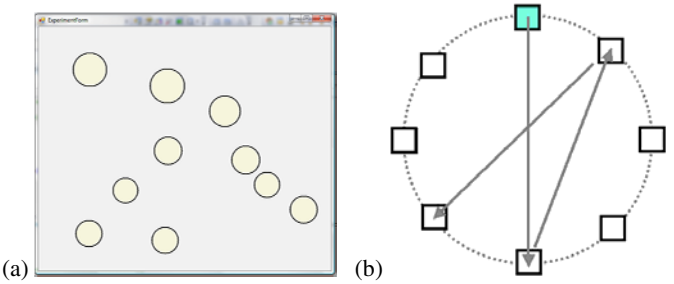


Fig. 2. Software (a) practice (b) ISO 9241-9 task #2

The experiment software was an implementation of task #2 in ISO 9241-9. See Fig. 2b. Eight square targets were uniformly positioned around a layout circle with a diameter of 600 pixels. Each target was 80 × 80 pixels. The participant first clicked on the top target, then on the target directly opposite, then on the target next to the first target, and so on around the layout circle. At any time, the target to select was highlighted. The target colour changed during selection. Selecting all targets was a "sequence". This was repeated five times per mode of control.

Procedure. Participants sat in front of the laptop computer. The marker was put on the participant's forehead clearly visible to the web-cam. The participant was asked to look straight into the screen and with a press of a button the current orientation of the marker was saved. This orientation was the neutral orientation and was used for system calibration. The calibration was done only once before each condition. Before each condition, participants spent 10 minutes getting familiar with the head-mounted marker using the practice software. For each condition, the participant was instructed on how the mode of control worked and on what was expected. Testing started after any button was pressed.

After the completion of all tests a questionnaire was handed to the participants to solicit their impressions on the input modes. The questionnaire was based on response items in ISO 9241-9 [5]. Each item had a rating from 1 to 5, as follows:

Smoothness during operation (1: very smooth – 5: very rough)
Mental effort required for operation 1: too low – 5: too high)
Accurate pointing (1: easy – 5: difficult)
Neck fatigue (1: none – 5: very high)
General comfort (1: very uncomfortable – 5: very comfortable)
Overall the input device was (1: very difficult to use – 5: very easy to use)

Experiment design. The experiment was a 3 × 5 within-subjects design, with the following independent variables and levels:

Mode of control: Position-control, Velocity-control, Generic Mouse
Sequence: 1, 2, 3, 4, 5

Participants were divided into three groups with the mode of control conditions administered using a Latin square.

The experiment included throughput as the dependent variable, as specified in ISO 9241-9. Throughput, in bits per second (bps), is a composite measure combining the speed and accuracy of participant responses. After each sequence of trials, throughput was calculated using

$$Throughput = ID_e / MT \quad .(2)$$

MT is the mean movement time per trial over all trials in a sequence. ID_e is the effective index of difficulty, computed as

$$ID_e = \log_2(D / W_e + 1) \quad (3)$$

where D is the distance in pixels between the two targets (600 pixels) and W_e is the width of the distribution of clicks. W_e, is computed using

$$W_e = 4.133 \times SD_x \quad (4)$$

where SD_x is the standard deviation in selection coordinates along the axis of approach. Details are provided in other papers [e.g., 9].

5 Results and Discussion

Throughput. As clearly seen in Fig. 3a, there was a significant effect of mode of control on throughput ($F_{3,8}$ = 555.7, p < .0001). Not surprisingly, the mouse performed best. Its throughput of 4.42 bps is consistent with mouse results in other ISO 9241-9 evaluations [9], thus validating our methodology.

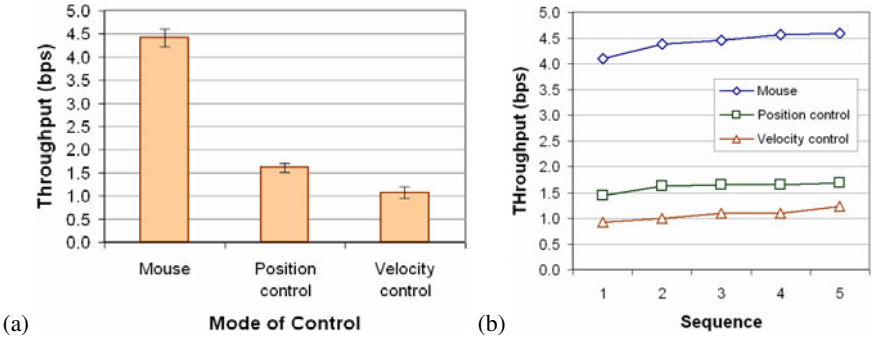


Fig. 3. Throughput (a) by mode of control (b) by sequence

The mean throughput for position-control movement of 1.61 bps was about 34% higher than the throughput of 1.07 bps observed for velocity-control movement. While substantially less than the throughput for the mouse, these figures are consistent with throughput for other pointing devices, such as a joystick (1.8 bps; see [8]).

As the trials continued, the throughput for both the position-control and velocity-control modes improved (Fig. 3b). After the first sequence, the position-control mode did not improve much, but the throughput for the velocity-control mode kept increasing.

Position-control mode produced better results because the mouse cursor went to the target position determined by the marker orientation almost instantaneously. In velocity-control mode, the mouse cursor had a constant speed (150 pixels/second). This slowed velocity-control mode, as multiple iterations of move and adjust were needed to position the cursor on the target.

Participant feedback. As *MarkerMouse* is intended for accessible computing and testing was done with able-bodied participants, there was an expected bias favoring the mouse. In comparing the two modes of control for *MarkerMouse*, participants indicated an overall preference for the position-control mode, rating it better than velocity-control mode for both neck fatigue and accurate pointing. Participants, however, gave slightly better scores to velocity-control mode for mental effort and smoothness.

6 Conclusions

In this paper we introduced a new method of controlling a mouse cursor using a marker placed on the user's forehead. Two methods of cursor control, position-control and velocity-control, were compared along with a desktop mouse. The mean throughputs for the position- and velocity-control modes were 1.61 bps and 1.07 bps, respectively, compared to the desktop mouse mean throughput of 4.42 bps.

Noise in marker detection was a limitation for the position-control mode. Velocity-control mode was less influenced by noise. However, because velocity-control is less intuitive, velocity-control mode performed worse than position-control mode.

An improvement for position-control movement would be a better way to deal with the marker detection noise, to improve precision and reduce jitter. This problem can be approached from two directions: to have a more robust method for marker detection and marker orientation calculation, and to have a better smoothing algorithm that minimizes noise after the orientation was computed.

References

1. Barreto, A.B., Scargle, S.D., Adjouadi, M.: A practical EMG-based human-computer interface for users with motor disabilities. Journal of Rehabilitation Research 37, 53 (2000)
2. Chauhan, V., Morris, T.: Face and feature tracking for cursor control. In: Proceedings of the Scandinavian Conference on Image Analysis (Unpublished, 2001); retrieved from `http://www.cs.man.ac.uk/~tmorris/SCIA630.pdf` (December 23, 2009)
3. Felzer, T., Nordmann, R., Rinderknecht, S.: Scanning-based human-computer interaction using intentional muscle contractions. In: Smith, M.J., Salvendy, G. (eds.) Universal Access in Human-Computer Interaction, Part II, HCII 2009. LNCS, vol. 5618, pp. 309–318. Springer, Heidelberg (2009)
4. Harada, S., Landay, J.A., Malkin, J., Li, X., Bilmes, J.A.: The vocal joystick: Evaluation of voice-based cursor control techniques. In: Proceedings of the ACM Conference on Computers and Accessibility - ACCESS 2006, pp. 187–204. ACM, New York (2006)
5. ISO, Ergonomic requirements for office work with visual display terminals (VDTs) - Part 9: Requirements for non-keyboard input devices (ISO 9241-9), International Organisation for Standardisation. Report Number ISO/TC 159/SC4/WG3 N147, February 15 (2000)

6. Knoerlein, B., Szekely, G., Harders, M.: Visuo-haptic collaborative augmented reality ping-pong. In: Proceedings of the International Conference on Advances in Computer Entertainment Technology - ACE 2007, pp. 91–94. ACM, New York (2007)
7. MacKenzie, I.S., Ashtiani, B.: BlinkWrite: Efficient text entry using eye blinks. In: Universal Access in the Information Society (UAIS) (in press)
8. MacKenzie, I.S., Kauppinen, T., Silfverberg, M.: Accuracy measures for evaluating computer pointing devices. In: Proceedings of the ACM Conference on Human Factors in Computing Systems - CHI 2001, pp. 119–126. ACM, New York (2001)
9. Soukoreff, R.W., MacKenzie, I.S.: Towards a standard for pointing device evaluation: Perspectives on 27 years of Fitts' law research in HCI. International Journal of Human-Computer Studies 61, 751–789 (2004)
10. Su, N.M., Mark, G.: Communication chains and multitasking. In: Proceedings of the ACM Conference on Human Factors in Computing Systems - CHI 2008, pp. 83–92. ACM, New York (2008)
11. Tu, J., Tao, H., Huang, T.: Face as mouse through visual face tracking. Computer Vision and Image Understanding 108, 35–40 (2007)
12. Yanco, H.A.: Wheelesley: A robotic wheelchair system: Indoor navigation and user interface. In: Mittal, V.O., Yanco, H.A., Aronis, J., Simpson, R. (eds.) Assistive Technology and Artificial Intelligence, pp. 256–268. Springer, Berlin (1998)

The Wiimote with SAPI: Creating an Accessible Low-Cost, Human Computer Interface for the Physically Disabled

Aqil Azmi†, Nawaf M. Alsabhan†† and Majed S. AlDosari††

†King Saud University, Riyadh, Saudi Arabia
††King Abdulaziz City for Science and Technology, Riyadh, Saudi Arabia

Summary
In this paper we report on an inexpensive device that helps a person with physical disability to interact with the Web. The human computer interface is made through a head tracking pointer. The head tracking pointer consists of the Wii controller (Wiimote), a low cost and readily available game controller, and infrared (IR) LED. The Wiimote is used to detect the position of the user's head. The Windows API (Application Programming Interface) has been used to move the pointer to the proper position according to the position given by the Wiimote. Microsoft SAPI (Speech Application Programming Interface) has been used to input commands via speech. The browser has been developed to provide a simple easy-to-use graphical user interface (GUI) to be used by the physically disabled. The total cost of the system is less than $50, much less than any commercial product with the same functionality. The system received high marks in the field test.

Key words:
Accessibility, physical disability, input and interaction technology, object tracker, Wii Remote.

1. Introduction

In less than ten years, the Internet has become indispensable. Consequently, physically handicapped people require an efficient, user-friendly Web browser application with innovative technologies, and most importantly an inexpensive software and hardware interface that ease the process of surfing the Web and which is accessible to general users. New user interfaces have been developed that allow direct control of pointers using body gestures [1]-[4]. Previously available technologies, for users with physical disabilities, are developed with expensive technologies, making them inaccessible to general users. In this paper, a new innovative head tracking pointer system for people with major defects, such as people with spinal injuries, was introduced and coupled with simple user-friendly Web browser software.

Gaze or head pose is a potentially powerful and intuitive pointing cue if it can be obtained accurately and unobtrusively. Eye movement is an important topic of study in evaluating user interface performance and potentially as an input cue [5]. However, the dynamics of involuntary human eye movements limit its accuracy for independent fine pointing control, although it can provide a coarse reference frame [6]. In this paper, we focus on head tracking, which has been a topic of active interest to many people in the computer vision community [7], and which appears to be less affected by involuntary control movements than eye gaze. Several authors have built prototypes which incorporate head pose tracking into user interface controls [8]-[11].

The objective of this paper is to propose simple user-friendly head pointer system and Web browser software for people with physical disability. The goal of the head pointer system and Web browser is to ease the process of surfing the Internet. To validate the viability of the proposed new head pointer system and Web browser, the Web browser software and the head tracking pointer system will be built and tested.

In the following section we discuss related work. Some assistive technologies and adaptive strategies are discussed in Section III. In Section IV, we describe both hardware and software components and how the system is put together or designed. Then, we go on to discuss the technologies used in the implementation and describe how the system is used in Section V. The paper concludes with the present status of the project and a description of further work.

2. Related Work

The proposed head pointer system and Web browser has a foundation in previous works. The MBROLA, BrailleSurf, Access Gateway and NEDO projects were designed for handicapped people, such as blind people and people with speech impairment [12].

- The MBROLA Project
  The aim of the MBROLA project, initiated by the Circuit Theory and Signal Processing (TCTS) Lab, is to obtain a set of speech synthesizers for as many languages as possible. A speech synthesizer takes a list of phonemes as input, together with prosodic information, to produce a speech sample. (It converts text to speech "TTS.")

Manuscript received December 5, 2009
Manuscript revised December 20, 2009

- BrailleSurf
  An Internet browser for users with vision impairment, which allows a simplified reading of the information available on the Web. BrailleSurf shows this information in a text form [13], [14]. This information then can be displayed on a Braille bar, or it can be spoken out by a speech synthesizer. The text also can be presented on the screen for people with low vision, and can be used to provide a fast review of the accessibility level of a Web site for visually impaired people.
- Access Gateway
  An online "browser within a browser" providing more control over how Web sites are displayed (regardless of which browser is being used). The Gateway's main purpose is to make the Web easier to access for people with a print disability (such as users with weak vision or dyslexia) [13]. It also translates Web pages into other languages and displays them as an image, so they can be displayed when the encodings are not supported by the user's browser.
- NEDO (New Energy Development Organization) IT Barrier-Free Project of disabled persons and other users. This project promotes the development of verification and evaluation experiments on a navigation system having portable user terminals that can be shared among all people, including the disabled, and which are particularly easy to operate [15].

3. Assistive Technologies and Adaptive Strategies

The proposed head pointer system and Web browser is intended to make use of assistive technology and adaptive strategies. An understanding of what assistive technologies and adaptive strategies are and what kinds are available will assist in the comprehension of the system and browser design.

Assistive technologies are products used by people with disabilities to help accomplish tasks that they cannot accomplish otherwise or could not do easily otherwise [16]. When used with computers, assistive technologies are sometimes referred to as adaptive software or hardware.

Adaptive strategies are techniques that people with disabilities use to assist in using computers or other devices [16]. For instance, someone who cannot see a Web page may tab through the links on a page as one strategy for helping skim the content.

Following is a *non-comprehensive* list of assistive technologies and adaptive strategies available [16].

- Alternative keyboards or switches
- Scanning software
- Screen reads
- Speech recognition
- Speech synthesis
- Tabbing through structural elements
- Text browsers
- Visual notification
- Voice browsers

4. System Synthesis

In designing the head pointer system prototype, Commercial-Off-The-Shelf (COTS) hardware and software were used. This helped in focusing on the functionality of the system. The head tracking pointer system is designed to be comfortable, simple, user friendly and most importantly cost efficient. In addition, it is designed to require a minimum effort from the user to interact with computer and the browser. To validate the viability of this project, a model system was built, implemented and tested.

An in-depth look at the hardware and software of the system is provided in the following sections, as is a discussion of the system as a whole.

4.1 Hardware Components

An extensive search for a hardware that can be used to sense the movement of a user keeping cost and reliability of system in mind. We decided to integrate the following two hardware components for the head-tracking pointer system: (1) Wiimote; and (2) IR Glasses. Below is further detail of both of these devices.

1) Wii Remote (Wiimote as it is commonly known*):* The dimensions of the Wiimote body are 148 mm long, 36.2 mm wide and 30.8 mm thick [17] (Fig. 1). It is considered to be a low-cost controller, costing less than $35.

The controller communicates wirelessly with the console (in our case a PC) via short-range Bluetooth radio, with which it is possible to operate up to four controllers as far as 10 meters (approx. 30 ft) away from the console. However, to utilize pointer functionality, the Wiimote must be used within five meters (approx. 16 ft) of the Sensor Bar [17], [18].

A main feature of the Wiimote is its motion sensing capability, which is particularly relevant to the proposed head pointer system because all such systems have some movement as a basis for their operation. The Wiimote

has the ability to sense acceleration along three axes through the use of an ADXL330 accelerometer [17]-[20]. The Wiimote also features a PixArt optical sensor, allowing it to determine where the Wiimote is pointing [17], [18], [21]. Unlike a light gun that senses light from a television screen, the Wiimote senses light from the console's Sensor Bar, which allows consistent usage regardless of a screen's type or size. The Wiimote's image sensor [18], [21] is used to locate the Sensor Bar's points of light in the Wiimote's field of view. The Sensor Bar is about 20 cm (8 in) long and features ten infrared (IR) light-emitting-diodes (LEDs), five at each end of the bar [18], [21] (Fig. 2). Innovative users have used other sources of IR light as Sensor Bar substitutes, such as a pair of flashlights and a pair of candles [22]. Such substitutes for the Sensor Bar illustrate the fact that a pair of non-moving lights provides continuous calibration of the direction that the Wiimote is pointing and its physical location relative to the light sources.

Another feature of the Wiimote is its expandability through the use of attachments. In the future, attachments for the Wiimote could be designed that would expand or improve the capabilities of the proposed head pointer system.

Fig. 1: Wii Remote.

Fig. 2: Sensor Bar IR LEDs.

2) IR Glasses: The proposed head pointer system was designed to use a pair of LED Light Vision V2 Safety Glasses which come with two LEDs that emit bright white light (Fig. 3). These LEDs were replaced with IR LEDs, since the Wiimote can track sources of IR light (IR LED). With the LEDs replaced, the location of the user's head can be tracked accurately and the cursor can be made to respond to the user's head movements. This will result in full control of the curser. The only consideration left is how the user would click (see *4.2. Software Components).*

Fig. 3: LED Glasses.

4.2. Software Components

The system uses three software components. The first component is the Windows API (or formerly known as Win32 API), which is used to create and manage the user interface. For instance, in our case managing the position of the mouse pointer and its functions. The second component is the WiiLib, which is the library used to control and communicate with the Wiimote. The last software component is the SAPI (Speech Application Programming Interface) (Fig. 4).

SAPI, an interface designed by Microsoft, supports dynamic speech input and output, and is integrated in Microsoft's current operating system. With the application programming interface (API), it is possible to develop speech enabled applications without caring about the details of synthesis and recognition.

In general, all versions of APIs have been designed such that a software developer can write an application to perform speech recognition and synthesis by using a standard set of interfaces, accessible from a variety of programming languages [23]. In addition, it's possible for a third party company to produce their own Speech Recognition and Text–To–Speech engines, or to adapt existing engines to work with SAPI.

Typically, SAPI is freely redistributable component which can be shipped with the Windows application that is intended to use Speech technology. Many versions (although not all) of the Speech Recognition and Synthesis engines are also freely redistributable [23]. Since Windows is the most popular operating system throughout the world, this API was chosen to give the user of the proposed head pointer system the ability to click, right click and double click, using a predefined vocabulary.

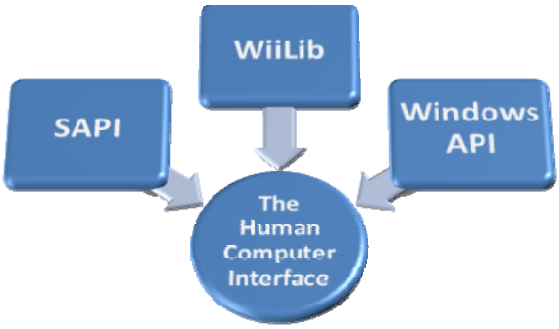


Fig. 4: Software Integration.

5. Implementation

Implementation of the proposed head pointer system was possible after completion of the system design. The technologies used in the implementation, as well as component integration and system use, are discussed in this section.

5.1 Used Technologies

For software writing we decided to use C# to write the desired classes, interfaces and the business logic, which must use .NET environment. Our system was built using some of Microsoft's development products (e.g., Visual Studio 2005).

1) C# (C Sharp): C# is a multi-paradigm object oriented Programming language developed from C and C++. It was created to run on the Microsoft .NET platform. C# was chosen based on our experience and preference. Other programming languages can be used. However, since both SAPI and WiimoteLib work either under or with the .NET framework, it is required to use a programming language supported by the .Net framework.

2) WebBrowser Class: Microsoft introduced the WebBrowser class (System.Windows.Forms), which enables applications to incorporate Web browsing capabilities [24]. It can be used, for example, to provide integrated HTML based user assistance or Web browsing capabilities in an application. Additionally, it can be used to add existing Web based controls to Windows Forms client applications. This control has several properties, methods, and events related to navigation. The following members can be used to navigate the control to a specific URL, move backward and forward through the navigation history list, and load the home page and search page of the current user [25].

- URL
- Navigate
- GoBack
- GoForward
- GoHome
- GoSearch

5.2 Initialization Set

For the set-up, a subject (or user) sits in front of the presentation monitor. The user must wear the IR glasses and make sure that the IR LEDs are working. The Wiimote IR camera next to the presentation monitor tracks the user's head activity.

A few seconds are taken initially to calibrate the particular user and their posture in this head tracking system. Then, the speech recognition functionality from SAPI is enabled to recognize a few words that we predefined to allow the user to click, right click, and double click. The Wiimote IR camera locates the position of the IR LEDs in 3D coordinates and sends them to the head tracking system Control Unit System (the application on the PC) via Bluetooth. The system Control Unit processes the information it receives from the Wiimote connected to it, and calculates head position in both vertical and horizontal in 3D coordinates. This information can be used in real time to convert 3D coordinates received from the IR camera to 2D coordinates of the screen. The result of this conversion is used in real time by the system to move the mouse pointer according to the given 2D coordinates.

The application was prototyped with the following properties:

- The Bluetooth is used to communicate with the Wiimote IR camera.
- Information is read continuously from the Wiimote IR camera.
- The information coming from the Wiimote is decoded to get the head coordinates in real time.
- The coordinates of the data stream coming from the Wiimote continuously are compared.
- If the subject says voice command "click," for instance, then a mouse click is made.

5.3 How to Use

As soon as the program runs, a small GUI form will pop up. This form consists of the following (Fig. 5):

- Bar indicator:
  Indicates the battery level of the Wiimote, which is retrieved from it using its API.
- Simple button:
  This button can be used to start the calibration, or calibration can be started just by clicking on the A button on the Wiimote. When the

calibration starts, a code segment will be executed to predefine the screen borders.

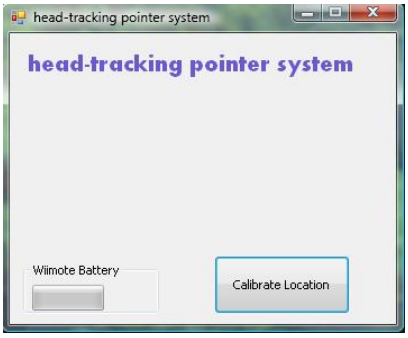


Figure 5: First Form.

Now, the user can interact freely with the computer via voice and head movements. With this ability to interact, the user freely can choose the browser that was developed with the HCI application to ease the process of browsing Web sites with the use of a virtual keyboard. This virtual keyboard considers the most frequently accessed Web pages and most frequently inputted terms (Fig. 6).

In order to test our system usability and get some feedback, users from different backgrounds, were invited to try it. All users found the system acceptable and most of them learned to use it in a minute. It is worth noting that most users were very pleased with the entire system.

Figure 6: Browser Form.

6. Conclusions and Future Works

One of the main ideas about this project was to develop a system that costs much less than any commercial products with the same functionality. Consequently, the total cost of this project results in less than $50, which is the price of the Wiimote (Fig. 1) and the IR glasses (Fig. 3).

An innovative system that combines head tracking and mouse input to allow a user to control a conventional GUI has been presented for people with major defects, such as people with spinal injuries. Both latency and obtrusiveness were found to be minimal in this system. Head tracking can be accurate, robust, automatically initialized and fast with this system, and the system can work efficiently on any current computer system. The developed browser 1) provides a virtual keyboard with large keys and font to make system use easier, and 2) provides a certain features, such as shortcut keys for the most frequently entered terms and most frequently visited Web sites.

To augment this system, new techniques and technologies for future work are being explored. Integrating some of the previously discussed assistive technologies and adaptive strategies may result in a better result or application. In future work, we will enhance this prototype and investigate how we can proceed towards mobile and robust use of this innovative technology.

Acknowledgment

The authors of this work would like to express gratitude and thanks to Dr. J.C. Lee, a HCI researcher at Carnegie Mellon University, for all his Wiimote projects. They were really the inspiration for this project. The authors are also grateful to B. Peek for providing the Wiimote library (WiimoteLib) on the Web, which was used for reading data from the Wiimote.

References

[1] A. Pentland, "Perceptual Intelligence," *Comm ACM*, Vol. 43, No.3, pp. 35-44, Mar. 2000.

[2] J. Crowley, J. Coutaz and F. Berard, “Things That See: Machine Perception for Human Computer Interaction,” *Comm ACM*, Vol. 43, No.3, pp. 54-64, Mar. 2000.

[3] IBM Almaden Research Center, "Creating computers that know how you feel," *Int. Business Machines (IBM) Corp.*, 2006.[Online]. Available: http://www.almaden.ibm.com/cs/BlueEyes/index.html. [Accessed: Oct. 10, 2009]

[4] M. Turk and G. Robertson, "Perceptual User Interfaces," *Comm ACM*, Vol. 43, No.3, pp. 32-34, Mar. 2000.

[5] R. Jacob, “What you look at is what you get: Eye movement-based interaction techniques,” *Proc. CHI '90* (1990), pp. 11-18.

[6] S. Zhai, C. Morimoto and S. Ihde, “Manual and gaze input cascaded (MAGIC) pointing,” *Proc. ACM CHI*, 1999, pp. 246-253.

[7] K. Schwerdt and J. L. Crowley, “Robust face tracking using color,” in 4th *IEEE Int. Conf. on Automatic Face and Gesture Recognition*, Grenoble, France, Mar. 2000, pp. 90-95.

[8] F. Berard, “The Perceptual Window: Head Motion as a new Input Stream,” in 7th *IFIP Conf. Human-Comput Interaction (INTERACT)*, 1999, pp. 238-244.
[9] M. Jones and T. Poggio, “Multidimensional Morphable Models,” *Proc. ICCV98*, 1998, pp. 683-688.
[10] R. Kjeldsen, "Head Gestures for Computer Control", *Proc. 2nd Int. Workshop on Recognition, Analysis Tracking Faces Gestures Real-time Systems, ICCV*, IEEE Press, 2001, pp. 62-67.
[11] ABiSee Inc., [Online]. Available: www.abisee.com. [Accessed: Oct. 20, 2009]
[12] T. Dutoit, X. Ricco, "Towards a Free Multilingual Speech Synthesis Software for the Vocally Handicapped," *Österreichische Gesellschaft für Artificial Intelligence*, vol. 20, Jul. 2001, pp. 36-38.
[13] “Software for people with disability,” [Online]. Available: http://www.ebility.com/links/software.php. Jul. 13, 2009. [Accessed: Sept. 18, 2009].
[14] Web Alternative Initiative. “Alternative Web Browsing.” [Online]. Available: http://www.w3.org/WAI/References/Browsing#1. Oct. 18, 2005. [Accessed: Sep. 10, 2009].
[15] New Energy and Industrial Technology Development Organization (NEDO). “IT Barrier-Free Project for Disabled Persons and Other Users.” [Online]. Available: http://www.nedo.go.jp/english/activities/1_sangyo/1/p04006e.html. [Accessed: Sept. 12, 2009]
[16] W3C, “How People with Disabilities Use the Web.” [Online]. Available: http://www.w3.org/WAI/EO/Drafts/PWD-Use-Web/#tools. May 5, 2005. [Accessed: Sept. 11, 2009]
[17] H. O. Wiggins, "Gesture Recognition of Nintendo Wiimote Input Using an Artificial Neural Network," Henry Owen Wiggins webpage, pp. 2, Apr. 17, 2008. [Online]. Available: http://sites.google.com/site/kindjie/GestureRecognitionofNintendoWiimoteI.pdf. [Accessed: Sept. 14, 2009].
[18] H. Filippi, “Wireless Teleoperation of Robotic Arms,” M.S. Thesis, Dept. Space Science, Luleå Univ. of Technol., Kiruna Helsinki, Finland, 2007, http://epubl.ltu.se/1653-0187/2007/079/LTU-PB-EX-07079-SE.pdf.
[19] C. Guo and E. Sharlin, “Exploring the use of tangible user interfaces for human-robot interaction: a comparative study,” in *Proc. 26th Annual SIGCHI Conf. on Human Factors in Computing Systems*, 2008, pp. 121-130, Florence, Italy.
[20] E. L. Wong, W. Y. F. Yuen and C. S. T. Choy, “Designing Wii controller: A powerful musical instrument in an interactive music performance system,” in *Proc Int Conf Advances in Mobile Computing and Multimedia*, New York: ACM, Linz, Austria, 2008,pp. 82-87.
[21] T. Schou , H.J. Gardner, “A Wii remote, a game engine, five sensor bars and a virtual reality theatre,” in *Proc. of the 19th Australasian Conf. on Computer-Human Interaction: Entertaining User Interfaces*, Adelaide, Australia, 2007, pp. 231–234.
[22] I. A. Suhardi, “Large Group Games With A Motion And Orientation-Sensing Game,” M.S. Thesis, Dept. Mathematics and Computer Science, Univ. of Bremen, Germany, 2008. [Online]. Available: http://www.abiamy.com/abiyasa/thesis/thesis-suhardi.pdf. [Accessed: Sept. 18, 2009]
[23] H. Shi and A. Maier, “Speech-enabled Windows application using Microsoft SAPI,” *International Journal of Computer Science and Network Security*, Vol. 6, No. 9A, pp. 33-37, Sep. 2006.
[24] H. Deitel and P. Deitel, *Visual Basic: How to Program*, Prentice Hall, 2005.
[25] MSDN. “WebBrowser Class.” [Online]. Available: http://msdn.microsoft.com/en-us/library/system.windows.forms.webbrowser.aspx. 2009. [Accessed: Sep. 12, 2009].

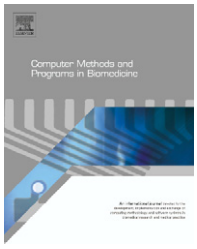
Aqil M. Azmi received the B.S.E degree from Univ. of Michigan, Ann Arbor and the M.Sc and Ph.D from Univ. of Colorado, Boulder. Currently he is an assistant professor at the Dept of Computer Science, King Saud University. His research interests include application of computers in religious studies, Arabic language processing, HCI and algorithms in general.

Nawaf M. AlSabhan received the B.Sc degree in Computer Science from King Saud University in February, 2009. Currently, he is an academic researcher at King Abdulaziz City for Science and Technology (KACST). His research interests include artificial intelligence and pattern recognition, ubiquitous computing, HCI and robotics. He is a member of ACM.

Majed S. AlDosari received the B.Sc in Computer Science from King Saud University, in February, 2009. He has been a software developer at King Abdulaziz City for Science and Technology (KACST) since June, 2009. His research interests include artificial intelligence, algorithm design, HCI and robotics.

ELSEVIER

journal homepage: www.intl.elsevierhealth.com/journals/cmpb



Robust real time eye tracking for computer interface for disabled people

*Alberto De Santis, Daniela Iacoviello**

Dept. Informatica e Sistemistica "A. Ruberti", Sapienza University of Rome, Via Ariosto 25, 00185 Rome, Italy

ARTICLE INFO

Article history:
Received 18 July 2008
Received in revised form
23 March 2009
Accepted 24 March 2009

Keywords:
Human computer interface
Eye tracking
Image segmentation
Level set
Recursive estimation

ABSTRACT

Gaze is a natural input for a Human Computer Interface (HCI) for disabled people, who have of course an acute need for a communication system. An efficient eye tracking procedure is presented providing a non-invasive method for real time detection of a subject eyes in a sequence of frames captured by low cost equipment. The procedure can be easily adapted to any subject and is adequately insensitive to changing of the illumination. The eye identification is performed on a piece-wise constant approximation of the frames. It is based on a discrete level set formulation of the variational approach to the optimal segmentation problem. This yields a simplified version of the original data retaining all the information relevant to the application. Tracking is obtained by a fast update of the optimal segmentation between successive frames. No eye movement model is required being the procedure fast enough to obtain the current frame segmentation as one step update from the previous frame segmentation.

1. Introduction

An HCI facility for a disabled person is a complex system composed of physical devices (such as video cameras, brain activity sensors, head movement sensors, special contact lenses, etc.) and signal analysis algorithms. The purpose consists in determining with sufficient accuracy the point of gaze of a subject exploring a pc screen, to provide a smart tool improving the subject autonomy both in terms of communication and environment interaction [1,2]. The various solutions proposed have different degree of invasiveness, depending on the technology involved: generally speaking the more sophisticated the devices the more invasive and higher cost [3–5] the HCI; correspondently signal analysis is quite simple. The use of non-invasive low cost technology demands complex algorithms to deal with the real time constraint. We consider indeed the latter situation: the subject is positioned in front of the screen of a general purpose pc endowed with a low cost video camera. A general plot of an eye tracking system is represented in Fig. 1.

Three main reference systems can be identified [6]: the eye reference system, the camera reference system and the screen reference system. The tracking system analyzes the image captured by the camera, determines the line of gaze of the subject watching the screen and identifies the point of the screen the subject is looking at. The position of the pupil center and the pupil shape on the image plane depend on the point on the screen the pupil is directed to, on the location of the user head, on the parameters of the eye model, on the position of the screen with the respect to the camera and on the camera intrinsic parameters.

Eye detection procedures usually exploit the pupil reflectance power to perform image zoning to separate the subject head from the environment and therefore identify the pupil shape and center by template matching. As suggested in [7], there are mainly three kinds of methods: template,

* *Corresponding author*. Tel.: +39 0677274061; fax: +39 0677274033.
E-mail addresses: desantis@dis.uniroma1.it (A. De Santis), iacoviel@dis.uniroma1.it (D. Iacoviello).
0169-2607/$ – see front matter © 2009 Elsevier Ireland Ltd. All rights reserved.
doi:10.1016/j.cmpb.2009.03.010

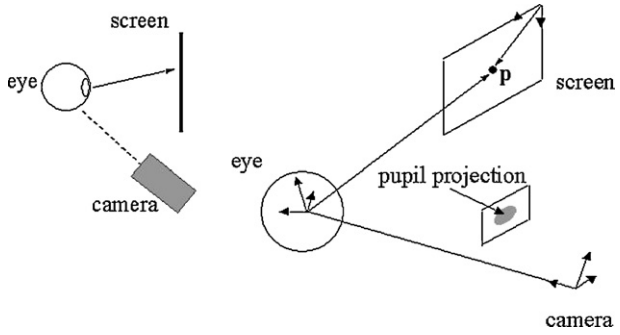


Fig. 1 – General eye tracking set-up and coordinate references.

appearance and feature-based methods. In the first one, a deformable template is designed to fit at best the eye shape in the image; the eye identification is usually accurate but needs a proper initialization, is computationally expensive and requires good image contrast [8,9]. The appearance methods require large amounts of data to train a classifier with the eye photometric appearance in different subjects under different face orientations [10,11]; a good performance can be obtained provided that the classifier has a good generalization property, which depends on the completeness of the training set. Feature-based methods rely on characteristics such as the pupil darkness, the sclera lightness, the iris circular shape, the shape of the eye corners, etc., to distinguish the human eye from the context. In [12] for instance, a picture zoning is performed by the Turk method [13] and eye blinking detection is accomplished to initialize the location of the eyes providing the position of the feature points for the eye tracking by means of the Lucas Kanade's feature tracker [14]. All these steps require a high image contrast to detect and track the eye corners. In [15], after a picture zoning is performed, the iris lower semi circle is detected as the curve that maximizes the normalized flow of the luminance gradient. In [7] eye detection and tracking is based on the active remote IR illumination approach, combined along with an appearance-based method. IR illumination approaches are simple and generate alternate bright/dark pupil images so that pupil detection is obtained after thresholding the difference image [16]; non-perfect illumination, occlusion and eye closure may limit the success of the eye tracking process. These drawbacks are avoided in [7] by using a support vector machine; consequently eye tracking is performed by a Kalman filtering approach reinforced by a mean shift tracking algorithm. In [17] eye corners, eyelids and irises are tracked and their motion is analyzed to determine gaze direction and blinking: here template matching is exploited to identify the eye elements whereas their motion is determined by the optical flow associated with their edge segments; head motion is estimated to detect head-independent eye movements such as saccades or smooth pursuit. In [18], the *between-the-eyes* pattern is considered for feature tracking, and head movement cancellation is performed for easier eyelids movement detection.

In this paper we present a robust and efficient procedure of image analysis to identify the eyes in a video sequence. This is the first block in the functional flow chart shown in Fig. 2; the other blocks are just standard and a possible implementation can be found in [6].

The new method is feature-based and relies on an image segmentation technique developed in [19] and applied to the identification of pupil shape parameters in [20]. In these papers the image segmentation process is accomplished by a region-growing algorithm based on a discrete level set formulation of an optimal approximation problem. It exploits a very general convex cost function that takes into account a fit error term, that evaluates the approximation error between the image and its segmentation, and regularity terms penalizing a segmentation with ragged boundary and small area sub domains. A further term penalizes large values of the level set function, and makes the cost function convex. The unique optimal solution provides a piece-wise constant segmentation of the original image with a lower number of gray levels and therefore with clear-cut edges, preserving all the information relevant to the considered application. The numerical efficiency and the quality of the segmentation of the discrete level set approach versus competing segmentation algorithms, in the class of edge detection, global thresholding and continuum level set, were tested in [19,20].

Region-growing algorithms are known to be robust with respect to various signal degradations such as additive noise, blurring, illumination changes. On the other hand they may be time consuming and therefore their use in an eye tracking application needs some adaptation. This issue is the main motivation of this work. The method in [19] is modified by considering a simpler cost function: only the fit error and the convexity terms are retained. Indeed, the segmentation boundary regularity term are computational expensive and their suppression in the considered application does not affect significantly the quality of the segmentation. The interconnection between frames is obtained by updating the current frame segmentation starting from the previous frame optimal segmentation. This also significantly contributes to the algorithm speed increase, and would not be possible with simple thresholding methods. An efficient picture zoning is performed to quickly identify the user head position, aiming to reduce the size of each frame to be analyzed and therefore to decrease the processing time.

The paper is organized as follows. In Section 2 design issues and the general set-up are presented. Then the optimal segmentation procedure is described in Section 2.1, whereas the feature extraction and pupil identification are presented in Section 2.2. The automatic calibration procedure is explained in Section 2.3 and the eye tracking procedure is found in Section 2.4. In the experimental Section 3 the eye tracking procedure is evaluated in terms of accuracy and robustness,

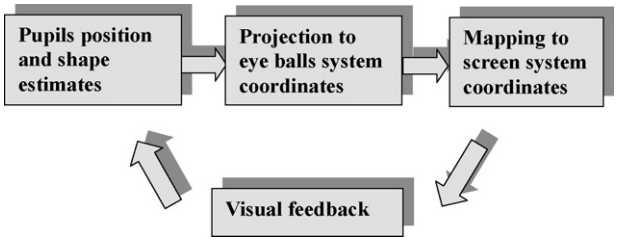


Fig. 2 – Flow chart of the eye tracking system.

and results of the application to real video sequences are discussed. Conclusions and further developments can be found in Section 4.

2. Design

The standard set-up we refer to is a regular pc facility: the user is sat in front of a screen endowed with a single low cost video camera, pointed on the subject face, Fig. 3. The videotaping occurs in a regular room illumination, no additional light sources are considered, like IR lamps.

To test the algorithm, described in the next sections, in critical conditions, we did not produce any ad hoc video sequence but rather used two sequences taken from the web, http://www.ecse.rpi.edu/homepages/cvrl/database/database.html, courtesy of Prof. Qiang Ji. The sequences are taken at 15 frames per second, each frame is an 8-bit gray level image with resolution of 320×240; each sequence presents moderate facial expression changes. In these sequences the subjects perform eye movements far more complex than those expected in an application for disabled people, like wide head rotation and eye occlusions.

2.1. *The optimal segmentation procedure*

Consider a simple 2D monochromatic image I with just one object over the background. The object boundary can be represented by the *boundary set* ϕ_0 of a function ϕ: $D \to R$, where $D \subset R^2$ is a grid of points (pixels) representing the image domain. The boundary set is defined as follows

$$\phi_0 = \{(i,j) : sign(\phi_{h,k}) \neq sign(\phi_{i,j}), \quad \text{for at least one } (h,k) \in [i \pm 1, j \pm 1]\} \tag{1}$$

Let us assume that the region $\{(i,j)$: $\phi_{i,j} \geq 0\}$ coincides with the object; the pixels not belonging to ϕ_0 are either in the interior of the object or in the background. In this case it is easy to obtain a binary representation I_s of the original picture

$$I_s = c_1 \chi_{(\phi \geq 0)} + c_2 \chi_{(\phi < 0)} \tag{2}$$

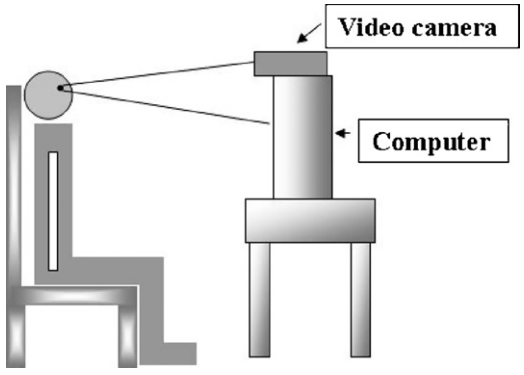


Fig. 3 – Experimental set-up.

where

$$\chi_{(\phi \geq 0)} = \begin{cases} 1 & \phi_{i,j} \geq 0 \\ 0 & \text{elsewhere} \end{cases}, \quad \chi_{(\phi < 0)} = \begin{cases} 1 & \phi_{i,j} < 0 \\ 0 & \text{elsewhere} \end{cases} \tag{3}$$

and c_1, c_2 are two different constant positive gray level values. Image I_s provides a piece-wise constant segmentation of the original picture I: it is a simpler representation of the data with a clear-cut between the object and the background; the information about the object shape is preserved. Function ϕ is called *level set function* and, according to (2), operates the image segmentation. Segmentation (2) can be obtained by solving an optimal approximation problem defined as follows:

$$\begin{aligned} \min_{(c_1,c_2,\phi)} E(c_1, c_2, \phi) &= \min_{(c_1,c_2,\phi)} \left[\lambda ||I - I_s||^2 + \alpha ||\phi||^2\right] \\ &= \min_{(c_1,c_2,\phi)} \left[\lambda \sum_{i,j} (I_{i,j} - c_1)^2 H(\phi_{i,j}) \right. \\ &\quad \left. + \lambda \sum_{i,j} (I_{i,j} - c_2)^2 (1 - H(\phi_{i,j})) + \alpha \sum_{i,j} \phi_{i,j}^2 \right] \end{aligned} \tag{4}$$

where H is the Heaviside function, λ and α are two positive parameters. The first two terms represents the fit error between the original data and the piece-wise constant approximation; the third term is a regularity term that makes the cost function convex [19]. Function E is a particular form of the energy functional used in [19,20]; it does not contain terms weighting the area and the boundaries length of the segmentation sub-regions, as the contribution of these terms is marginal to the current application and their computation is time consuming. Following the argument in [19], a smooth version of the cost function is advisable and it is obtained by substituting in (4) the Heaviside function by a smooth approximant

$$H_\varepsilon(\phi) = \frac{1}{1 + \exp(-\phi/\varepsilon)} \tag{5}$$

It can be proved that the smooth version of problem (4) admits necessary and sufficient conditions for a unique global minimum that, by standard calculus, are obtained as follows

$$c_1 = \frac{\sum_{i,j} H_\varepsilon(\phi_{i,j}) I_{i,j}}{\sum_{i,j} H_\varepsilon(\phi_{i,j})}, \quad c_2 = \frac{\sum_{i,j} [1 - H_\varepsilon(\phi_{i,j})] I_{i,j}}{\sum_{i,j} [1 - H_\varepsilon(\phi_{i,j})]} \tag{6}$$

$$\alpha \phi_{i,j} + \lambda [(I_{i,j} - c_1)^2 - (I_{i,j} - c_2)^2] \delta_\varepsilon(\phi_{i,j}) = 0 \tag{7}$$

where δ_ε is the derivative of function H_ε. Different kind of approximant can be considered, as for instance indicated in [21], but expression (5) is more convenient from the computational point of view. From (6) we note that c_1 is the image gray level average over the region in which $\phi \geq 0$, whereas c_2 is the average over the complementary region. Eq. (7) provides an implicit equation for the level set function in every pixel (i,

Fig. 4 – Image segmentation by the discrete level set procedure; (a) original; (b) first binarization; (c) four levels segmentation.

j); it can be solved by the following iterative scheme

$$\alpha\phi_{i,j}^{n+1} + \lambda[(I_{i,j} - c_1^n)^2 - (I_{i,j} - c_2^n)^2]\delta_\varepsilon(\phi_{i,j}^n) = 0 \tag{8}$$

that has a fast convergence to the unique solution of (7), starting from any initial condition ϕ^0. The computational cost of any update of Eq. (8) can be easily obtained in terms of the image size $N \times M$; once constants c_1 and c_2 are known, the level set function can be updated according to (8) with $3(N \times M)$ sums and $3(N \times M)$ products. According to (6), constant c_1 requires $2(N \times M)$ sums, $N \times M$ products and one division; constant c_2 requires $3(N \times M)$ sums, $N \times M$ products and one division. Therefore one run of the algorithm (6)–(8) is linear in the image size $N \times M$ in terms of sums and products.

The level set method has the amenable property that during the level set evolution (8) the boundary set ϕ_0^n, starting from *any* initial shape ϕ_0^0, can *merge* and *split* in order to easily deal with the complex topology of the real world images.

Should we need to preserve other object appearance details, the number of gray levels can be increased to four by further segmenting the regions in (3) by the use of two more level set functions ϕ_+, ϕ_-

$$I_s = c_{11}\chi_{(\phi\geq 0)}\chi_{(\phi_+\geq 0)} + c_{12}\chi_{(\phi\geq 0)}\chi_{(\phi_+<0)} + c_{21}\chi_{(\phi<0)}\chi_{(\phi_-\geq 0)} + c_{22}\chi_{(\phi<0)}\chi_{(\phi_-<0)} \tag{9}$$

where c_{ij}, $i, j = 1, 2$, are the constant gray level values within each of the four sub-regions individuated by the two level set functions. Segmentation (9) is therefore accomplished by a hierarchical procedure obtained by successive binarizations. First the image domain is partitioned in two regions $\{(i, j): \phi_{i,j} \geq 0\}$ and $\{(i, j): \phi_{i,j} < 0\}$. Then each of these sub domains is further partitioned in two sub domains with the help of the additional level set functions ϕ_+, ϕ_-:

$$\{(i,j) : \phi_{i,j} \geq 0\} = \left(\{(i,j) : \phi_{i,j} \geq 0\} \bigcap \{(i,j) : \phi_{+i,j} \geq 0\}\right) \bigcup \left(\{(i,j) : \phi_{i,j} \geq 0\} \bigcap \{(i,j) : \phi_{+i,j} < 0\}\right)$$

and

$$\{(i,j) : \phi_{i,j} < 0\} = \left(\{(i,j) : \phi_{i,j} < 0\} \bigcap \{(i,j) : \phi_{-i,j} \geq 0\}\right) \bigcup \left(\{(i,j) : \phi_{i,j} < 0\} \bigcap \{(i,j) : \phi_{-i,j} < 0\}\right)$$

By the same argument the number of gray levels can be further increased, but for the eye tracking problem the four levels segmentation (9) proved to be sufficient to represent all the information relevant to the features extraction process. Fig. 4 shows the result of the hierarchical procedure.

The binarization of Fig. 4b will be referred to as *first binarization*. The four levels segmentation provides a *cartoon image* of the original data where the eyes always belong to the darkest part, whatever the colour of the eyes and the race of the subject.

2.2. *Features extraction and pupils detection*

The frame four levels segmentation, as obtained in Fig. 4, is not suitable for the identification of the subject pupils. We need a more detailed segmentation to distinguish the pupils from the other eye elements. To this aim a binary image can

Fig. 5 – Pupils identification. (a) Image mask selecting the darkest elements in the early four levels segmentation in Fig. 4c; (b) final four levels segmentation where the pupils are separated from the sclera; (c) darkest elements of (b).

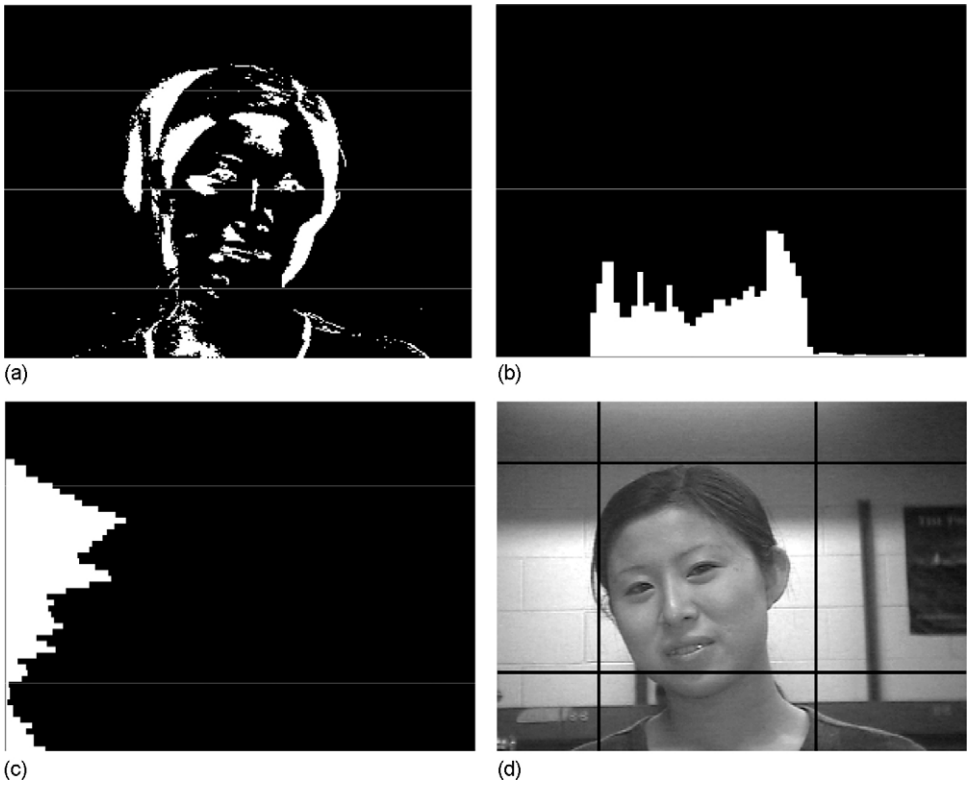


Fig. 6 – Frame zoning. (a) Thresholded difference picture between a frame and the average background; (b) histogram of the horizontal distribution of the white pixels in (a); (c) histogram of the vertical distribution of the white pixels in (a); (d) original frame with the zoning boundaries obtained by the analysis of histograms (b) and (c).

be obtained from the previous four levels segmentation, by a simple boolean operation, thus highlighting the dark elements (and therefore the eyes) and leaving the rest of the picture in the black background, see Fig. 5a.

The obtained binary image is a mask that is used to select two regions on the original image to be segmented, obtaining again a four levels segmentation where, this time, the pupils are well separated from the sclera. Indeed, a simple boolean operation allows now to identify the pupils as the darkest elements of Fig. 5b, thus obtaining the picture on Fig. 5c, that will be referred to as *final binarization*.

The Matlab©instruction *bwlabel* can now label all the white objects over the black background and the instruction *regionprops* evaluates the morphological characteristics of any labeled object. Among the others, the shape parameters of interest are: the area, the centroid coordinates, the major and minor axes length, the orientation. The area is just the number of pixels that build up the object. The centroid coordinates are computed as the coordinates of the object center of mass. The major and minor axes belong to the ellipse that has the same second order moment of the object, computed according to [22]. The orientation measures the angle formed by the major axis and the horizontal axis. All these quantities can be used to identify the subject pupils, among the other objects on the final binarization, as will be explained in the next subsection.

2.3. *Automatic calibration*

The main problem in an eye tracking procedure is a trade-off between accuracy and processing time. In this respect

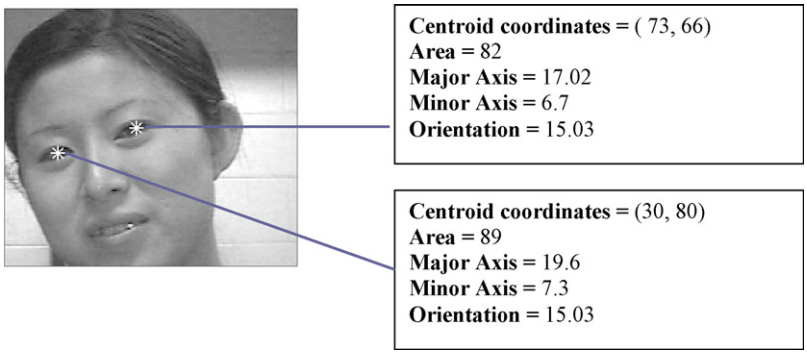


Fig. 7 – Estimation of pupils position and some shape parameters.

the data size is an issue and therefore it may be necessary to reduce it for further speed increase. This can be obtained by a frame zoning procedure aimed at identifying the frame portion containing the subject head. Consider that between adjacent frames (within 1/15-th of a second) the picture background practically remains the same whereas the subject position changes only slightly. Before the tracking starts, few seconds of the video acquisition can be used to estimate the pixel-to-pixel average and standard deviation of the gray level. Therefore the pixel-to-pixel comparison of one frame to the average clearly indicates the frame portion where the head movement has occurred, see Fig. 6a. In this picture, in the pixels marked as white, the difference between the gray level and the average was, in absolute value, greater than the standard deviation.

The spatial distribution of the white pixels in Fig. 6a along the horizontal and the vertical directions can be easily obtained. The horizontal spatial distribution, for instance, is determined by dividing the picture into vertical strips, that represent the histogram bins, and counting the number of white pixels into each strip. The distribution concentrates in the picture zone where the head movement occurred, Fig. 6b (for the vertical distribution see Fig. 6c). The average and the standard deviation of the two distributions are finally used to build a box that contains the subject head with a suitable tolerance, Fig. 6d. This zoning need not to be updated, since any subject working while sitting in front of a pc does not drastically move his body. Should this be the case the zoning could be easily repeated, at a rate slower than the frame acquisition rate of course, and used as a feedback to the tracking algorithm. Once the zoning has been performed, the calibration is completed by a reliable subject pupils detection. This is obtained by determining the final binarization, according to the procedure of Section 2.2, on the last frame acquired in the early short acquisition. Now the pupils can be identified since they are a couple of objects with very similar size and shape characteristics, Fig. 7.

These quantities can be stacked in two features vectors that represent the *eyes signature*. If needed, to avoid false

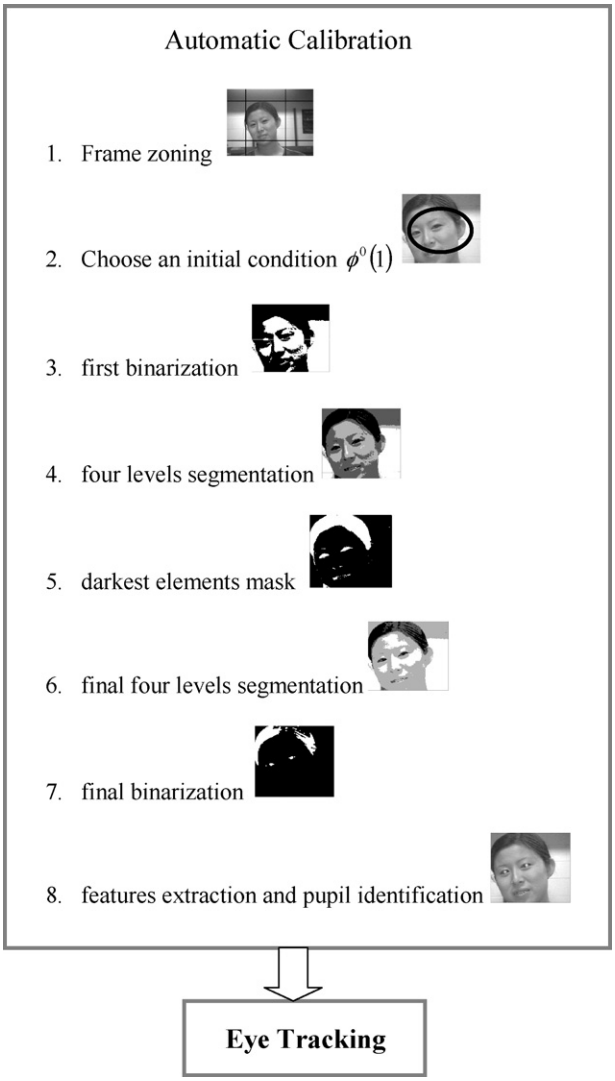


Fig. 8 – Flow chart of the initial automatic calibration procedure. The computed eye signature is passed to the next block to start the eye tracking, along with the final configuration $\phi(1)$ of the level set function that has performed the first binarization at step 3.

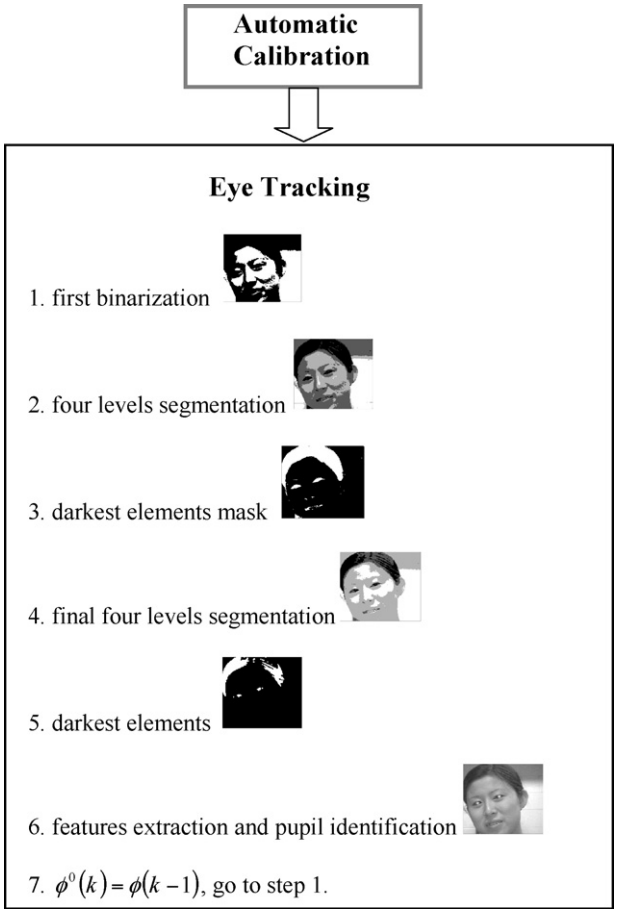


Fig. 9 – Flow chart of the eye tracking procedure. Step one is executed for the first frame after calibration by initializing the level set function at $\phi(1)$ coming from the previous block.

detections due to artefacts, anthropometric measure can be used to take into account, for instance, the between-the-eyes span.

This procedure is applied only in the calibration step and, for the available sequences, requires no more than 5 runs of algorithm (6)–(8) for each segmentation leading to the final binarization. During the eye tracking, the pupils can be directly identified by a fast procedure explained in the next subsection. The flow chart of Fig. 8 summarizes the initial automatic calibration procedure.

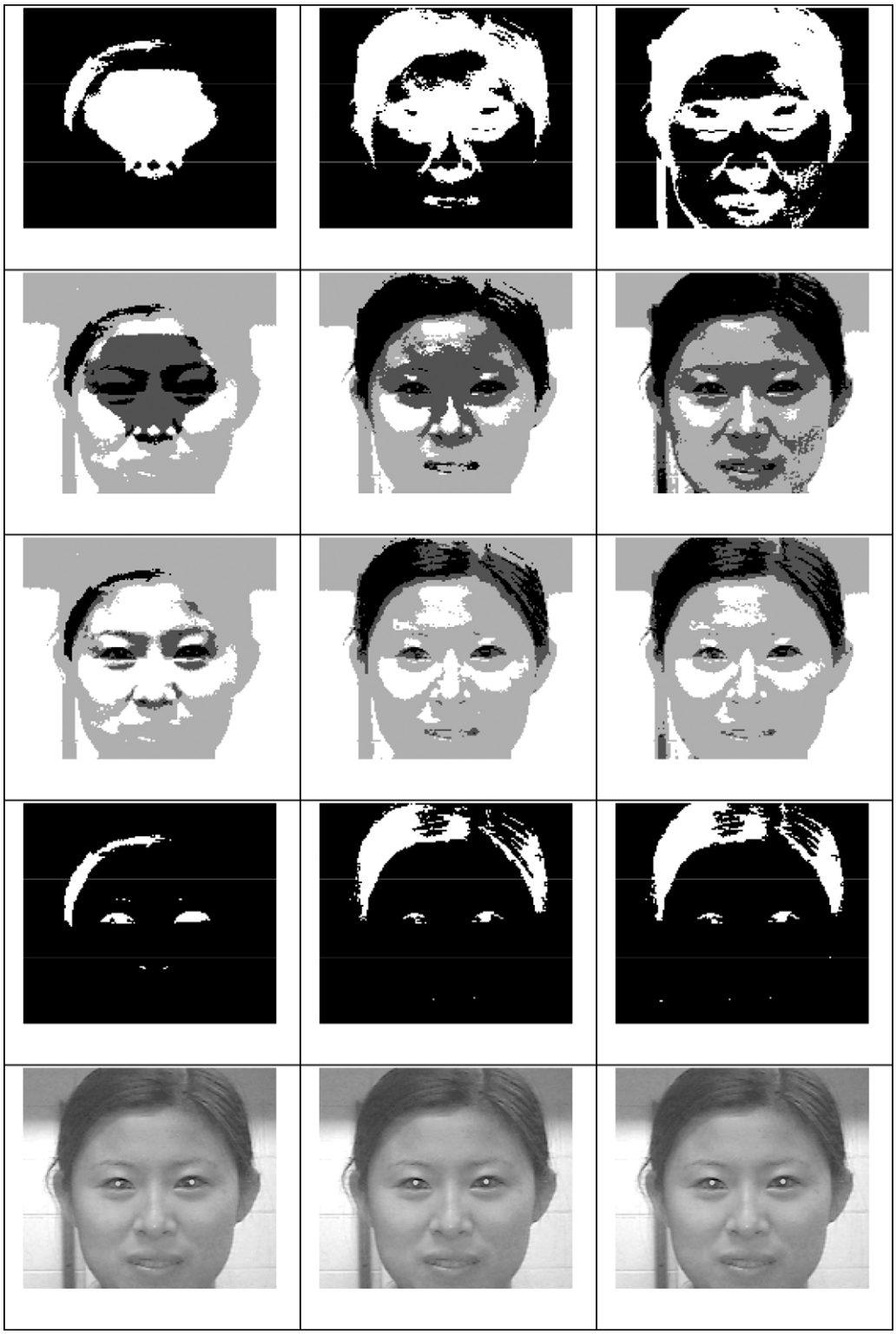


Fig. 10 – Segmentation results with $\lambda = 200$ (first column), $\lambda = 500$ (second column), $\lambda = 10^3$ (third column). In the first row the first binarizations are displayed. The second and third rows show the first and second four levels segmentations, respectively. In the fourth row the final binarizations are shown and in the last row the estimated pupils position are marked on the original frame.

Fig. 11 – Pupils tracking, some significant shots with gamma correction equal to 0.5.

2.4. *Eye tracking*

The eye tracking algorithm requires that the current frame be segmented quickly so to obtain in real time the eye features vectors. Let us suppose that the frame $(k-1)$ has been processed: this means that the hierarchical procedure of Section 2.1 has been completed and the level set function $\phi(k-1)$, that accomplished the first binarization, has reached its final configuration that solves (8). To apply the segmentation to the next frame k we need to choose an initial configuration $\phi^0(k)$ of the level set function to be evolved according to (8). It can be chosen equal to $\phi(k-1)$ to improve the rate of convergence of Eq. (8) since, due to the similarity of adjacent frames, the initial condition $\phi^0(k) = \phi(k-1)$ is very close to the solution of (8) for frame k. Indeed, each binarization step to obtain the final binarization of the k-th frame requires just one run of the algorithm (6)–(8).

On the final binarization for the current frame k all the objects are labeled and their shape parameters evaluated. Now the eyes signature computed at the step $(k-1)$ are exploited to quickly identify the pupils among all the labeled objects. This is obtained by simply matching the features vectors of these objects to the eyes signature of the previous frame. In this way false detections can be easily avoided. This matching procedure allows also to resolve eye occlusions, for instance due to blinking; indeed the eyes position is not updated until matching occurs. The matching is obtained when the normalized euclidean distance of the eyes signature between adjacent frames is within a chosen percentage. Normalization is taken with respect to the norm of the previous frame eyes signature vectors. The eye tracking procedure is outlined in the flow chart of Fig. 9.

3. Experiments

Before testing the tracking algorithm we want to support the choice of simplifying the segmentation cost function, as in (4), with respect to the general expression given in [19] where, besides the fit error, the segmentation boundary regularity was taken into account. In the first case the segmentation algorithm, denoted by A1, depends on parameters α and λ, while in the general case algorithm, denoted by A2, two more parameters have to be chosen: μ that weighs the area of the segmentation sub domains, ν that weighs the length of their boundary. Parameter ε only defines the degree of approximation of the generalized functions H and δ, with their approximants H_ε and δ_ε, and is usually chosen equal to 1. The

cost function expression in both cases can be normalized to the value of α, that is definitely set equal to 1. The reciprocal importance of the parameters for A2 was already assessed in [19], showing that the segmentation is more sensitive to λ; therefore also parameters μ and ν are set equal to 1. The choice of λ depends on the image contrast, but it is independent of its brightness: the lower the contrast the higher λ; values ranging from 10^2 to 10^5 have proven to give satisfactory results in general applications. For all these reasons, A1 is expected to be numerically more efficient than A2, with acceptable degradation of the accuracy. Indeed, we performed a four levels segmentation for 50 frames of the available sequence, and measured each time the fit error. In all the binarizations of the hierarchical segmentation procedure, 10 runs of the level set function update were considered with $\lambda = 10^3$, both for A1 and A2. For A2 a fit error with mean value of -6.8×10^{-4} and standard deviation of 0.06 was obtained within a cputime of 2.9 s, whereas for A1 the fit error mean value was equal to 1.3×10^{-4} with a standard deviation of 0.08, and a shorter cputime of 1.01 s. The algorithms have been implemented in Matlab© on a 2.6 Ghz pc. This result clearly indicates that the simplified cost function is the best choice when real time is an issue in the chosen application. This early experiment also shows the need for a frame resizing to reduce considerably the processing time.

Next we justify the choice of the value of λ. In Fig. 10, the results of every step of the hierarchical segmentation procedure are displayed in columns for values of $\lambda = 200$, 500, 10^3, respectively.

It can be seen that the segmentation results improve as λ increases, and are already good for $\lambda = 500$. Nevertheless on frames when the subject is looking aside the screen, the eye zone may suffer a contrast decrease, so that the higher value of λ is anyway preferred.

Now we perform the eye tracking on two sequences obtained from the web at http://www.ecse.rpi.edu/homepages/cvrl/database/database.html. For better signal conditioning a gamma correction equal to 0.5 was performed on the sequences frames, and $\lambda = 10^3$ was chosen. In Fig. 11, nine snapshots of the first video sequence are displayed; the subject performs a wide head movement so that various eyes orientations are present. Even though eye occlusion does not really happen, a very critical situation occurs when the subject is looking up. This indeed shows the robustness

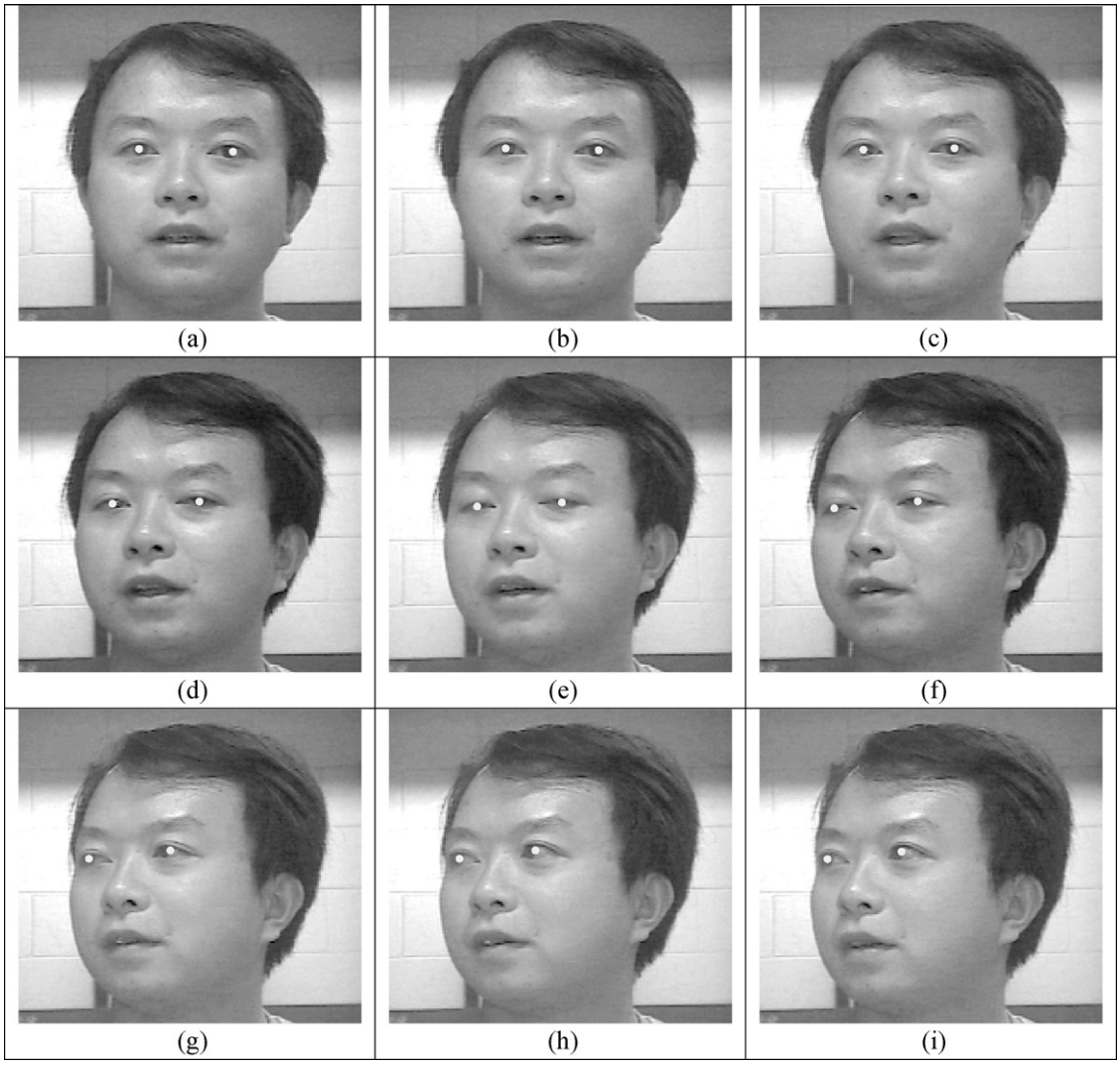


Fig. 12 – Pupils tracking, some significant shots with gamma correction equal to 1.5.

of the procedure with respect to different face orientation. This is mainly due to the tracking mechanism based on the segmentation update between successive frames and the matching of the eyes signature. The algorithm was able to track the pupils in all the 300 frames and the error on the pupils position, compared to the positions estimate provided by IR illumination as described in [7], was characterized by an average of 1 pixel with a standard deviation of 0.8. The used ground truth data was available on the database.

The experiment of Fig. 12 shows the robustness of the method with respect to eye occlusions. In this case, when one pupil, or even both, is lost because of the occlusion, the algorithm keeps memory of the last position correctly identified; this is updated as soon as the occlusion is removed and matching with the eyes signature occurs. Should the loss of matching of the eyes signature not recovered in few frames (for example, after occlusion), the procedure should be restarted from the beginning with the calibration procedure. Also in this case the error on the pupils position, compared to the positions estimate provided by IR illumination, was evaluated: it was characterized by an average of 1.45 pixel with a standard deviation of 0.806.

As final experiment the robustness of the tracking procedure with respect to changes of illumination was assessed. To this aim, the first sequence analyzed was artificially darkened/lightened by subtracting/adding a value of 0.3 to the gray level of the frames. Then we applied the eye tracking procedure to the new sequences obtained and evaluated the pupils position error with respect to the ground truth data. For the darkened sequence the error mean value was equal to 1.62 with standard deviation of 0.83, whereas for the lightened sequence the error had a mean value of 1.62 with standard deviation of 0.84. There is an increase in the average of the position error due to the signal variation but the accuracy remains practically the same, this denoting a good degree of robustness with respect to uniform changes of illumination.

The algorithm featured encouraging performances both in terms of accuracy of the pupils identification and speed, the latter being fundamental for real time applications. Indeed, the frame-to-frame processing for eye tracking takes 5 runs of the segmentation algorithm (6)–(8); each run requires a number of products and sums linear in the image size. The whole eye tracking procedure of Section 2.4 took 0.3 s on a pc endowed with a processor Intel Core Duo—2.6 Ghz and 4 Gbyte of RAM memory. Since the acquisition rate was 15 frames per seconds, we were able to process one frame over 5. This fact did not cause any tracking problem in these experiments, but of course it could be an important issue on other applications.

4. Conclusions

Eye tracking systems for disabled people should have some key characteristics: non-invasive low cost equipment, extremely simple calibration and robustness with respect to changes in illumination conditions. A tracking procedure has been developed able to track the subject eyes with no additional IR illumination. The method relies on a segmentation algorithm based on a discrete level set formulation of the optimal segmentation problem. The key feature of the method consists in a fast update of the optimal segmentation between successive frames. The tracking procedure shows a satisfactory degree of robustness with respect to various face orientations, eye occlusions, uniform change of illumination. Real time processing is attainable thanks also to a proper picture zoning reducing the data size. Further developments may include peculiar situations where, for example, the subject is wearing eye glasses, or where a changing in the background takes place.

Conflict of interest

The submission of the present work does not cause any conflict of interest of the Authors with any public Institution or private Company.

Acknowledgments

The authors wish to acknowledge the courtesy of Prof. Qiang Ji at Renssealer Institute for granting us the permission to use data from the database of his lab. They are also grateful to Dr. Jixu Chen at Renssealer Institute for all his collaboration in making the sequences available to them.

REFERENCES

[1] P. Majaranta, K.J. Raiha, Twenty years of eye typing: system and design issues, in: Eye tracking Research & Applications: Proceedings of the Symposium on ETRA, New York, 2002, pp. 15–22.
[2] D. Heckenberg, Performance evaluation of vision-based high DOF human movement tracking: a survey and human computer interaction perspective, in: Proceedings of the Conference on Computer Vision and Pattern Recognition Workshop, 2006, pp. 156–164.
[3] L.H. Yu, M. Eizenman, A new methodology for determining point-of gaze in head-mounted eye tracking systems, IEEE Transactions on Biomedical Engineering 10 (51) (2004) 1765–1773.
[4] D. Ishima, Y. Ebisawa, Eye tracking based on ultrasonic position measurement in head free video-based eye-gaze detection, in: Proceedings of IEEE EMBS Asian-Pacific Conference on Biomedical Engineering, 2003, pp. 258–259.
[5] A. Glenstrup, T. Angell-Nielsen, Eye Controlled Media, Present and Future State, Technical report, University of Copenaghen (1995), http://www.diku.dk/users/panic/eyegaze/.
[6] A. Villanueva, R. Cabeza, S. Porta, Eye tracking: pupil orientation geometrical modeling, Image and Vision Computing 24 (2006) 663–679.
[7] Z. Zhu, Q. Ji, Robust real-time eye detection and tracking under variable lighting conditions and various face orientations, Computer Vision and Image Understanding 98 (2005) 124–154.
[8] A. Yuille, P. Hallinan, D. Cohen, Feature extection from faces using deformable templates, International Journal of Computer Vision 8 (2) (1992) 99–111.
[9] K.M. Lam, H. Yan, Locating and extracting the eye in human face images, Pattern Recognition 29 (1996) 771–779.
[10] J. Huang, H. Wechsler, Eye detection using optimal wavelet packets and radial basis functions, International Journal of Pattern Recognition and Artificial Intelligence 13 (7) (1999) 1009–1025.

[11] W.M. Huang, R. Mariani, Face detection and precise eyes location, in: Proceedings of the International Conference on Pattern Recognition, 2000.
[12] T. Morris, P. Blenkhorn, F. Zaidi, Blink detection for real-time eye tracking, Journal of Network and Computer Applications 25 (2002) 129–143.
[13] M.A. Turk, Interactive-time vision: face recognition as a visual behaviour, MIT, Doctoral Thesis.
[14] X. Xie, R. Sudhakar, H. Zhuang, On improving eye feature extraction using deformable templates, Pattern Recognition 27 (1994) 791–799.
[15] Z. Hammal, C. Massot, G. Bedoya, A. Caplier, Eyes segmentation applied to gaze direction and vigilance estimation, in: International Workshop on Pattern Recognition for Crime Prevention, Security and Surveillance, 2005, pp. 236–246.
[16] C. Morimoto, D. Coons, A. Amir, M. Flickner, Pupil detection and tracking using multiple light sources, Image and Vision Computing, Special Issue on Advances in Facial Image Analysis and Recognition Technology 18 (4) (2000) 331–335.
[17] S. Sirohey, A. Rosenfeld, Z. Duric, A method of detecting and tracking irises and eyelids in video, Pattern Recognition 35 (2002) 1389–1401.
[18] S. Kawato, N. Tetsutani, Detection and tracking of eyes for gaze-camera control, Image and Vision Computing 22 (2004) 1031–1038.
[19] A. De Santis, D. Iacoviello, Discrete level set approach to image segmentation, Signal, Image and Video Processing 1 (2007) 303–320.
[20] A. De Santis, D. Iacoviello, Optimal segmentation of pupillometric images for estimating pupil shape parameters, Computer Methods and Programs in Biomedicine, Special Issue on Medical Image Segmentation 84 (2006) 174–187.
[21] T. Chan, L. Vese, Active contours without edges, IEEE Transactions on Image Processing 10 (2) (2001) 266–277.
[22] Haralick-Shapiro, Computer and Robot Vision, Vol. I, Addison-Wesley, 1992.