Pers Ubiquit Comput
DOI 10.1007/s00779-012-0617-z

ORIGINAL ARTICLE

Using kernels for a video-based mouse-replacement interface

Samuel Epstein · Eric Missimer · Margrit Betke

Received: 30 September 2011 / Accepted: 23 June 2012

Abstract Some people cannot use their hands to control a computer mouse due to conditions such as cerebral palsy or multiple sclerosis. For these individuals, there are various mouse-replacement solutions. One approach is to enable them to control the mouse pointer using head motions captured with a web camera. One such system, the Camera Mouse, uses an optical flow approach to track a manually-selected small patch of the subject's face, such as the nostril or the edge of the eyebrow. The optical flow tracker may lose the facial feature when the tracked image patch drifts away from the initially-selected feature or when a user makes a rapid head movement. To address the problem of feature loss, we developed and incorporated the Kernel-Subset-Tracker into the Camera Mouse. The Kernel-Subset-Tracker is an exemplar-based method that uses a training set of representative images to produce online templates for positional tracking. We designed the augmented Camera Mouse so that it can compute these templates in real time, employing kernel techniques traditionally used for classification. We propose three versions of the Kernel-Subset-Tracker, each using a different kernel, and compared their performance to the optical-flow tracker under five different experimental conditions. Our experiments with test subjects show that augmenting the Camera Mouse with the Kernel-Subset-Tracker improves communication bandwidth statistically significantly. Tracking of facial features was accurate, without feature drift, even during rapid head movements and extreme head orientations. We conclude by describing how the Camera Mouse augmented with the Kernel-Subset-Tracker enabled a stroke-victim with severe motion impairments to communicate via an on-screen keyboard.

Keywords Augmentative and alternative communication (AAC) · Human–computer interaction · Camera-based interfaces · Computer vision · Assistive technology

1 Introduction

Millions of people worldwide are affected by neurological disorders that cause communication barriers. If individuals with severe traumatic brain injuries, strokes, multiple sclerosis, or cerebral palsy are quadriplegic and nonverbal, they cannot use the computer with a standard keyboard and mouse, or a voice recognition system, as a communication tool.

Among individuals with these severe impairments, the Camera Mouse has been established as an assistive communication tool in recent years [4]. Individuals, who can control their head movement, even if the movement range is very small, use systems such as the Camera Mouse as a mouse-replacement interface. The Camera Mouse tracks head movements with a webcam and thereby enables a computer user to control the movement of the mouse pointer [5]. The Camera Mouse tracks a small feature on a user's face, such as a nostril or eyebrow corner. The location of the feature in the camera frame is transformed into the position of the mouse pointer on the screen (Fig. 1).

S. Epstein · E. Missimer · M. Betke (✉)
Department of Computer Science, Boston University, Boston, MA, USA
e-mail: betke@bu.edu; betke@cs.bu.edu

S. Epstein
e-mail: samepst@cs.bu.edu

E. Missimer
e-mail: missimer@cs.bu.edu

Published online: 06 November 2012

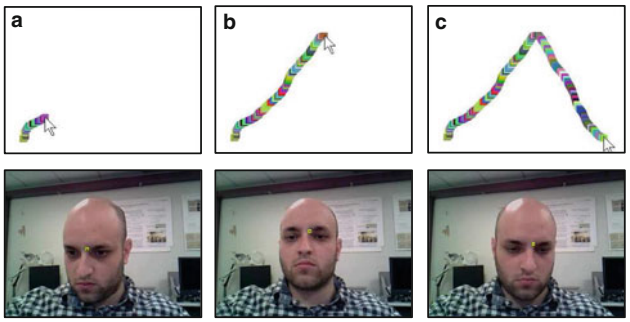

Fig. 1 Mouse replacement systems enable the user to control the mouse pointer using head movements captured by a webcam. Here, the user is drawing a line with a painting program by moving his head. The feature being tracked is a 10 × 10-pixel image patch on the subject's *left* eyebrow. The subject moved his head from his *lower left* (**a**), upward (**b**), and then to his *lower right* (**c**). The image coordinates of the feature were translated into screen coordinates for the mouse pointer by a linear mapping

The most recent version of the Camera Mouse uses an optical flow approach for tracking [23]. Optical-flow trackers estimate the location of a feature to be tracked by matching the image patch estimated to contain the feature in the previous image with the locally best-matching patch in the current image. Optical-flow trackers are known to incur "feature drift" [9]. The tracked location may slowly drift away from the initially-selected feature, for which no record is kept. Camera Mouse users may experience a slow drift of the tracked feature along the nose or eyebrow of the user. Feature loss can also occur when a spastic user makes a rapid involuntary head motion.

To address the problems of optical-flow tracking, we introduce the KERNEL-SUBSET-TRACKER. The KERNEL-SUBSET-TRACKER uses an exemplar-based approach to track the user's head. A training set of representative sample images of the user's face (or regions of the face) are collected at the beginning of the computer session. After the setup phase, these images are used to create template images for positional tracking. Our approach is based on kernel projections [10, 11], a technique from classification theory.

We here report a significant improvement of the communication bandwidth of test subjects when the Camera Mouse is augmented with the KERNEL-SUBSET-TRACKER. We refer to this system as the *Augmented Camera Mouse* to distinguish it from the standard Camera Mouse. The Augmented Camera Mouse tracks facial features accurately, without any notable drift, even when subjects move their heads quickly or through extreme orientations, and in the presence of background clutter. We also report that the Augmented Camera Mouse successfully tracked the eyebrow of a user with severe movement impairments. The user was thus able to generate mouse-click events by raising his eyebrow.

2 Related work

Assistive technology offers many hardware devices for people with motion impairments, but very few video-based mouse-replacement systems. A database of information about assistive technology, ABLEDATA [1] lists more than 36,000 products for users with disabilities. The database category "mouse emulation programs" has only 58 entries, and most of these describe education software to be used with physical switches. Only two systems listed offer camera-based mouse-pointer control: the Camera Mouse and the Quick Glance 3^{TM} mouse emulator system by EyeTech Digital Systems. Quick Glance 3 [33] illuminates the user's face with infrared lighting and tracks his or her pupils using infrared-sensitive cameras. Other infrared-based commercial mouse-replacement systems are the SmartNAV [36] system by NaturalPoint, which follows a reflective dots attached to the user's head, and the RED Eye Tracking System by SensoMotoric Instruments [34]. Another SensoMotoric product, the iView X HED [18], is a head-mounted system for eye tracking. The QualiEye program [32] by Qualilife is a camera-based mouse-replacement system that tracks a user's face using a webcam.

Unfortunately, commercial hardware solutions are often prohibitively expensive for many people with disabilities and their caregivers [22]. The most expensive commercial products are infrared-based eye-trackers that offer a high resolution in estimating gaze direction. Users, however, find it easier to control a mouse pointer with head motions than with their gaze [3] (in the latter case, users must look at the location of the mouse pointer while in the former case, they may look elsewhere, e.g. to plan their next move). Fortunately, there are a number of free mouse-emulation systems for users with motion impairments.

The Camera Mouse was the first camera-based mouse-replacement interface that was freely available to users with motion impairments [14], for example, to children with cerebral palsy. In the past decade, a number of other systems have been developed and tested successfully with people with motion impairments. The mouse-emulation system Nouse, for example, uses two web cameras to track the 3D position of the nose of the user and was tested with 15 users with motion impairments [15]. Another 3D approach was proposed by Tu et al. [38], which tracked one subject's face using a 3D model with 12 facial motion parameters. Based on the experiments with users with motion disabilities, Gorodnichy et al. [15] pointed out that the smoothness and range of the users' head movements are often overestimated by developers of camera-based interfaces.

Kjeldsen [21] focused on the problem of non-smooth head movements. He created the HeadTracking Pointer, a

mouse-replacement system that converts head movement to pointer movement with a sigmoidal transfer function. The function adapts the transfer rate based on the predicted mouse pointer destination and thus yields smooth mouse pointer movement. A preliminary camera-based mouse-replacement system, using traditional template matching techniques, was created by Kim and Ryu [19]. Palleja et al. [30] described a mouse-replacement system that tracks the head and detects blinks and mouth movements. Kjeldsen [21] and Kim and Ryu [19] mentioned plans to test the proposed interfaces with users with motion impairments.

Manresa et al. [27] tested an interface developed by Varona et al. [39] with 10 users with movement disabilities. Interface tracks multiple features on a subject's face, such as the nose, eyes, and mouth. The tracker can recover from tracking failures of individual features through support from other features. Tracking was accomplished using intensity gradients in the video frames. Using the same interface, eight users with movement disabilities reportedly controlled the temperature and lighting of a room [31].

Another camera-based mouse-pointer manipulation system was designed by Loewenich and Maire [22]. This system uses a boosted cascade of classifiers to detect a user's face in the video. During tracking, a collection of features are tracked using optical flow. This system was tested with 10 volunteers without movement disabilities.

It will be exciting to see how the computer vision techniques discussed above will improve the accuracy of facial feature tracking so that camera-based mouse-replacement systems can be successful tools for the larger community of people with movement disabilities. At this time, unfortunately many individuals with severe movement disabilities, who use mouse-replacement systems, gain only limited control of the mouse pointer. This is due to the difficulties many users have in positioning the mouse over traditional target areas such as buttons or web links.

Research efforts have been made to adjust application software so that it can be used successfully with a mouse-replacement system. Examples are the WebMediator, a program that alters the display of a web page so that the fonts of links become larger [40] and the CameraCanvas, an image editing tool for users with severe motion impairments [20]. Another example is the Hierarchical Adaptive Interface Layout (HAIL) by Magee and Betke [26], which is a set of specifications for the design of user interface applications, such as a web browser and a Twitter client, that adapt to the user. In HAIL applications, all of the interactive components take place on configurable toolbars along the edge of the screen.

Hwang et al. [16] reported that some users with impairments pause the pointer more often and require up to five times more submovements to complete the same task than users without impairments. Wobbrock and Gajos [41] focused on the difficulty that people with motion impairments have in positioning the mouse pointer within a confined area to execute a click command. They introduced "goal posts" which are circular graphical boundaries that trigger application actions when crossed with the mouse pointer. Findlater et al. [12] used this idea to create "area cursors" that use goal-crossing and magnification to ease selection of closely positioned interface targets. Betke et al. [6] proposed to discretize user-defined pointer-movement gestures in order to extract "pivot points," i.e., screen regions that the pointer travels to and dwells in. Related mechanisms are "gravity wells" that draw the mouse pointer into a target on the screen once it is in proximity of the target [8] and "steady clicks," a tool that reduces button-selection errors by freezing the pointer during mouse clicks and by suppressing clicks made while the mouse is moving at a high speed [37].

The Camera Mouse system may be the most-used freely-available camera-based mouse-replacement system to date. It has been downloaded 500,000 times as of August 2011 and is popular with users. Our new tracker, the Kernel-Subset-Tracker, is designed to support current Camera Mouse users and also empower new users, who could not use the Camera Mouse previously due to frequent feature loss. We incorporated the proposed Kernel-Subset-Tracker into the original Camera Mouse software. The new tracker can be toggled on and off to suit the needs of the user.

3 The Kernel-Subset-Tracker

The Kernel-Subset-Tracker is an exemplar-based tracking algorithm that uses a representative training set to model the objects to be tracked. It requires a training phase at the beginning of the interaction session. In the training phase, a set of object images is collected as a training set. For face tracking, the training set consists of images of size 100×100 of the face at different orientations of the head relative to the camera. The training set is used to identify the object to be tracked in successive image frames during human-computer interaction. At time t, the Kernel-Subset-Tracker determines a dissimilarity score, distance d_i, of the current object at position p, to each training image q_i in the training set $Q = \{q_1, q_2, \ldots, q_n\}$. From such distances, a positional template is created and used to find the next position p' of the object in the video frame.

```
1:  function KERNEL-SUBSET-TRACKER(t, p)
2:      I = GetVideoFrame(t)
3:      q = GetRealTimeObs(I, p)
4:      for all n training images q_i in Q do
5:          d_i = f(q, q_i)
6:      a = CreateTemplate(Q, d)
7:      p' = PositionSearch(v, p, a)
8:      Output(qt, p', d, a)
9:      KERNEL-SUBSET-TRACKER(t + 1, p')
```

In the KERNEL-SUBSET-TRACKER, see pseudocode above, function GETVIDEOFRAME returns the complete image frame at the current time t. Function GETREALTIMEOBS crops a subimage located at the current position p from the current video frame I. This subimage is the real-time observation q. Function f returns a distance measure between the real-time observation and each training image q_i of the training set Q. For many distance measures, evaluating f exhaustively becomes untenable for current computers if the distance measure uses every pixel in the input images. In Sect. 4, we describe a method to approximate the distance measure with a kernel (see Sect. 4).

The positional template a is computed by function CREATETEMPLATE, which takes as inputs the distances d and the training set Q. Function POSITIONSEARCH computes the optimal local alignment p' of template a, given the current video frame I and the previous position p. Eight subimages are cropped from the current video frame I from windows centered at position p and each of its eight neighbors $p + (-1, -1)$, $p + (0, -1)$, $p + (1, -1)$, $p + (1, 0)$, $p + (1, 1)$, …. The first estimate $\hat{p}'$ of the position is equal to the center position of the subimage that best matches a. The same distance measure used by function f is also used in the POSITIONSEARCH method to evaluate the eight alignment candidates. This process is repeated by considering the eight neighbors of $\hat{p}'$. Hill climbing proceeds until none of the neighboring subimages can provide a better alignment or until a fixed number of iterations has occurred. POSITIONSEARCH then returns the locally best estimate p'. The OUTPUT of the KERNEL-SUBSET-TRACKER for each frame is the 2D position of the tracked object, the distances d_i of the training images and the positional template a of the tracked object.

4 Distance approximation with kernels

The most computationally intensive component of the KERNEL-SUBSET TRACKER is the repeated calls to the distance method $f(\cdot, \cdot)$ for each training image q_i in the training set Q. We describe how to use kernel methods from machine learning [35] to approximate the distance function quickly.

Distance functions such as $f(\cdot, \cdot)$ define metric spaces and likewise inner product functions $\langle \cdot, \cdot \rangle$ define vector spaces. The most common inner product is the one for Euclidean spaces,

$$\langle (x_1, \ldots, x_n), (y_1, \ldots, y_n) \rangle = \sum_{i=1}^{n} x_i y_i.$$

Another example of an inner products is

$$\langle (x_1, x_2), (y_1, y_2) \rangle = x_1 y_1 + x_2 y_2 + (x_1 + x_2)(y_1 + y_2). \quad (1)$$

These inner products are also known as kernels. We use the notation of $k(\cdot, \cdot)$ to describe the kernels. If $k(\cdot, \cdot)$ is semi-positive definite then it is a valid kernel [35].

The main benefit of using kernels is that they endow distance measures with notions of angles and length and so projections can be used. Given the distance function f, we can create a kernel function $k(\cdot, \cdot)$ whose induced distance is equal to the function f. Thus the function f can be isometrically embedded in the vector space implied by the kernel[1]. We define such a kernel function $k(\cdot, \cdot)$ by

$$k(q, q') = h(q) - \frac{1}{2}\left(f(q, q')\right)^2 + h(q'), \quad (2)$$

for any arbitrary function $h : \mathcal{Q} \to \mathbb{R}$. In practice, however, it is easer to define the kernel function directly.

Using the subset projection method described by [11], we do not need to compute the distance function f between the real-time observation q and every training image q_i. Instead, we can compute a kernel function $\hat{f}$ that represents the distance between a real-time observation q and a small subset of the training images $R \subset Q$, with $R = \{r_1, \ldots, r_m\}$. The results of these inner products can be used to approximate the distances d_i. The pseudocode of KERNEL-SUBSET-TRACKER can be modified to accommodate this subset projection method by replacing lines

```
4:      for all n training images q_i in Q do
5:          d_i = f(q, q_i)
```

by the subset projection functionality:

```
4:      R = RANDOMSUBSET(Q, d^prev)
5:      for all m training images r_j in R do
6:          v_j = k(q, r_j)
7:      for all n training images q_i in Q do
8:          d_i = f̂(q, q_i, v)
```

The RANDOMSUBSET method returns a random subset R of the training images Q. The probability that a training

[1] This is assuming the distance function is Hilbertian.

image q_i will be chosen for a subset R is inversely proportional to its distance to the previous real-time observation d_i^{prev}. Thus, training images that are similar to the real-time observation of the previous frame have a higher probability to be in subset R. In practice, the distances to a training set Q of size 25 can be approximated using the subset projection method and a small set R of size 5.

5 Three kernels for the kernel-subset-tracker

In this section, we define three image-based kernels used in our experiments. An image-based kernel is a function of two grayscale images that returns a real number representing their inner product. A simple example of an image-based kernel function is one which returns the sum of the pairwise product of the intensity values of the images. On input images q and q' of size 100×100, this kernel returns

$$k(q, q') = \sum_{x=1}^{100} \sum_{y=1}^{100} q(x, y) * q'(x, y), \tag{3}$$

with $q(x, y)$ representing the brightness of image q at position (x, y).

5.1 Threshold kernel

The threshold kernel is the main kernel we used in our experiments (Fig. 2). This kernel first performs thresholding of a pair of grayscale images according to threshold τ to produce two processed binary observations. It computes the size of the intersection of the "1" pixels of these two processed observations. For simplicity, this number is divided by the number of pixels of the input images to yield an output between 0 and 1 (the division operation has no effect on the performance of the kernel).

As we show below, the threshold kernel results in excellent tracking in certain imaging scenarios; however, it is not robust to changes in brightness, contrast, or object scale. This is due to the fixed nature of τ, the thresholding parameter.

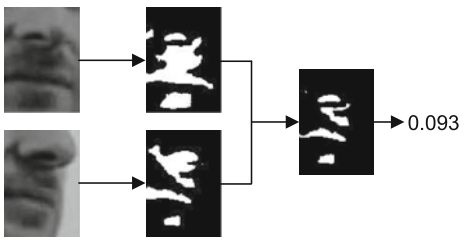

Fig. 2 An example of the threshold kernel. Two *grayscale* images are converted to binary images using a set threshold and then combined to a single binary image using the intersection operation. The final output is the percentage of "set" pixels in this combined image

5.2 Normalized threshold kernel

We designed the NORMALIZED THRESHOLD KERNEL to provide a tracking mechanism that is robust to changes in brightness and contrast. This kernel takes as input two grayscale images q and q' and outputs a real number between 0 and 1 (see pseudocode). Each input is converted to a binary image using its mean as the threshold. The size of the intersection of the two binary images is computed. This value is normalized by the number of pixels and returned. This final normalization is a convenience step, having no effect on the performance of the kernel.

The function NORMALIZED THRESHOLD KERNEL is semi-positive definite, and thus a valid kernel. It is invariant to uniform changes in brightness and contrast (Fig. 3).

```
1: function NormalizedThresholdKernel q, q'
2:     m = ComputeMean(q)
3:     m' = ComputeMean(q')
4:     c = 0
5:     for x = 1 to width of training images do
6:         for y = 1 to height of training images do
7:             If q(x,y) ≥ m and q'(x,y) ≥ m' then
8:                 c = c + 1
9:     return c/NumPixels(q)
```

5.3 Normalized radial intensity kernel

We introduce the NORMALIZED RADIAL INTENSITY KERNEL (NRI) to provide a tracking mechanism that is robust to changes in object scale. The NRI-Kernel computes an inner product on two grayscale images q and q' in the following two part process.

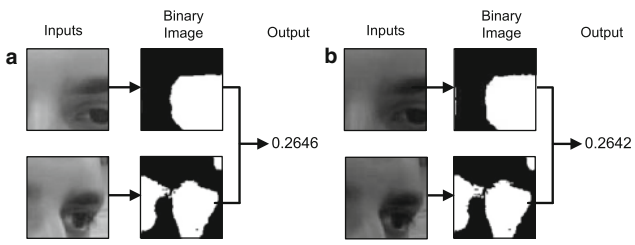

Fig. 3 NORMALIZED THRESHOLD KERNEL. The images (**a**) were subjected to the lowering of brightness and contrast (**b**). Thresholding based on the means of the images results in similar binary images (**a** and **b**) and kernel outputs. This is an example of the invariance of the NORMALIZED THRESHOLD KERNEL to uniform changes in brightness and contrast

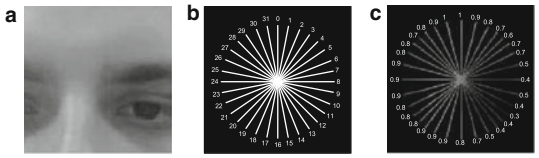

Fig. 4 A *grayscale* image (**a**) is converted into the intermediate feature vector (**c**) used by the NORMALIZED RADIAL INTENSITY KERNEL. Each number in (**c**) is an entry of the feature vector, which is created by summing up the intensity values from the center point in the directions shown in image **b**. The result is a feature vector of 32 positive numbers representing the relative intensity of each radial direction, normalized to be between 0 and 1, as shown in (**c**), rounded to one significant digit

The first part converts each grayscale image to an intermediate feature vector, which is a small array of positive real numbers between 0 and 1. Each value of the array represents the summation of intensity values of the image, along a ray from the center of the image proceeding in a specified direction. The array is normalized such that its largest entry is 1.0. An example conversion can be seen in Fig. 4. We tried a number of different array sizes, including 8 and 16 rays. We found the best performance of the KERNEL-SUBSET-TRACKER when we used 32 directions.

The second part of the NRI-Kernel computes an inner product between the two radial feature vectors v and v' derived from two images. We tried several methods, including the standard sum of pairwise multiplication of the values of the two vectors. However we found the intersection operation resulted in the best tracking results. Thus the NRI-Kernel returns the sum of the pairwise minimum of every pair of values in vectors v and v'. The sum is normalized (divided by 32) so that the output of the NRI-Kernel is between 0 and 1. This normalization is done for ease of comparison, and has no effect on the performance of the kernel.

The NRI-Kernel is invariant to small changes in scale of the object being tracked, since they would not affect the relative intensity values along the radial directions. The normalization operation makes the kernel also invariant to changes of brightness and contrast. Some sample inputs demonstrating this invariance can be seen in Fig. 5.

6 Positional template creation

In this section we describe the positional template function CREATEBINARYTEMPLATE we used in the KERNEL-SUB-SET-TRACKER in conjunction with both the Threshold Kernel and the Normalized Threshold Kernel. In the CREATEBINARYTEMPLATE function, the positional template a is constructed from the observation set Q, where the contribution of each individual q_i to the output is inversely proportional to its distance d_i to the real-time observation q. Given are

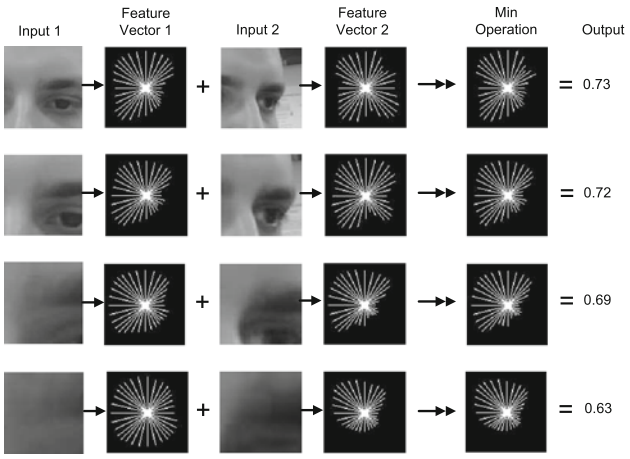

Fig. 5 The operations of the NORMALIZED RADIAL INTENSITY KERNEL. Each row shows the two *grayscale* input images at increasing scales. The NRI Kernel converts each image into an feature vector of size 32, where each value represents the sum of the pixels in a particular direction, starting from the center position. The feature vectors are shown with the lengths of rays representing the magnitude of each value. The arrays are combined into a third feature vector using the minimum operation. The output is the magnitude of this feature vector normalized to be between 0 and 1. The similarity of the outputs exemplifies how the NRI Kernel successfully handles local changes in scale

the distances d_i of the current frame subimage and the threshold of the Threshold Kernel τ.

The binary image template output a is created by iterating through every pixel position of the training images and setting a temporary value δ to 0. If the grayscale value of training image q_i is greater than threshold τ at the current position index $(x_{\text{pos}}, y_{\text{pos}})$, then it will "vote" for a 1 pixel by adding weight $1/d_i$ to δ. Similarly $1/d_i$ will be subtracted from δ if its intensity is below threshold τ. The contribution of each training sample q_i to the construction of a is proportional to $1/d_i$. After all training images have voted, the output a at position $(x_{\text{pos}}, y_{\text{pos}})$ will have intensity 1 if $\delta \geq 0$, otherwise 0.

```
1:  function CreateBinaryTemplate Q, τ, d
2:    for x_{pos} = 1 to width of training images do
3:      for y_{pos} = 1 to height of training images do
4:        δ = 0
5:        for i = 1 to n do
6:          If q_i(x_pos, y_pos) ≥ τ then
7:            δ = δ + 1/d_i
8:          else
9:        δ = δ − 1/d_i
10:         if δ ≥ 0 then a(x_pos,y_pos) = 1 else 0
11:   return a
```

This binary image is then used by the KERNEL-SUBSET-TRACKER algorithm in a local search to find the new

position of the object in the frame. This action is performed in the PositionSearch function of the Kernel-Subset-Tracker. At each position in the local search, a grayscale image is cropped from the current video frame. This image is thresholded into a binary image using the threshold τ of the kernel. All of the binary images of the neighboring positions are compared against the template and the current tracking position is changed to that of the closest matching neighboring binary image. This process is repeated until a local maximum is reached.

7 Augmenting the camera mouse with the KERNEL-SUBSET-TRACKER

In the Augmented Camera Mouse, the user can configure the Kernel-Subset-Tracker by selecting a kernel to use, the size of the training set, and the size of the subset projection. During the training phase, Augmented Camera Mouse populates the training set by obtaining a series of pictures of the user's head in different positions. To guide the user in making head movements that yield effective training images, the Augmented Camera Mouse asks the user to perform a simple target-reaching task. In this training phase, the user's motion is tracked with optical flow for bootstrapping. The target-reaching task requires users to move the mouse pointer with their head over a set of blocks on the screen as shown in Fig. 6. When the pointer enters a block, a subimage of the user's face, which is a 100×100 window around the currently tracked position, is stored as a training image. The number of blocks n^2 (e.g., $n = 2$, 3, or 4), and the size of blocks are configurable. The training phase lasts only a few seconds—as long as it takes the user to move his or her head into the n^2 positions. Retraining is required if the conditions during the computer session change significantly (e.g., the lighting changes or the user starts wearing glasses).

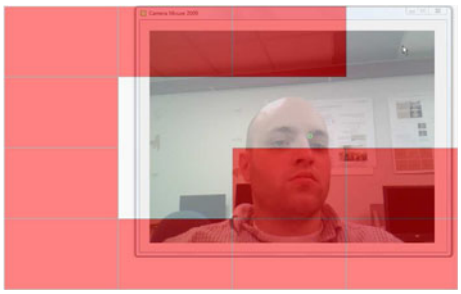

Fig. 6 Target-reaching task during the real-time image-collecting training phase of the Augmented Camera Mouse. Optical flow is used for tracking as a bootstrapping technique. The screen initially shows the overlay of n^2 *red blocks* (here 16) that the user is asked to reach with the mouse pointer. When the pointer enters a screen block, the Augmented Camera Mouse obtains a 100×100 subimage of the user's head (centered around the tracked feature) and adds it to the training set. The *red* overlay disappears to indicate that the screen region has been reached successfully (here, *five blocks* have been reached and five training images have been obtained) (color figure online)

The Augmented Camera Mouse uses both the original optical flow tracking algorithm and the Kernel-Subset-Tracker. At each frame, the old position of the facial feature is updated. The optical flow algorithm first computes an estimate of the position using a 10×10 square patch around the previous position. The Kernel-Subset-Tracker then crops a square window of length 100 pixels around this estimate. The Kernel-Subset-Tracker refines the estimate of the position for the next frame, using the hill climbing PositionSearch algorithm (Sect. 3).

8 Experiments with subjects without motor impairments

8.1 Participants

We worked with 19 subjects (16 males, 3 females, 20–40 years of age). The subjects did not have motion disabilities.

8.2 Apparatus

A Logitech Quickcam Pro 4000, which captures images at a frame rate of 30 Hz, was used as the video capture device. The Kernel-Subset-Tracker software package was implemented in C++. The experiments were conducted with a laptop with 4 GB of RAM and Intel Core Duo 2.1 GHz processors.

8.3 Test software

We developed test software that encourages subjects to move their head significantly while interacting with the Augmented Camera Mouse interface. Similar to HCI experiments in the past [2, 17], our test software displays a series of circles that the user targets with the mouse pointer. Each circle appears individually and disappears when the subject moves the mouse pointer to the current circle, triggering the next circle to become visible (Fig. 7). To induce different types of user motions, we designed three target arrangements that differ in placement, ordering, and sizes of circles.

8.4 Test procedure and setting

We tested the accuracy of the Augmented Camera Mouse with regard to tracking a subject's facial feature during varied head movements (Fig. 8). The subjects used our

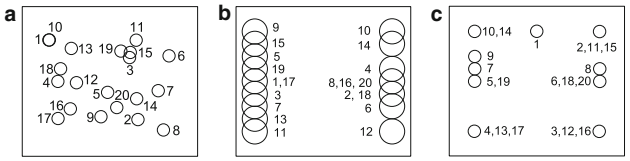

Fig. 7 The placement, size and ordering of the targets in our experiments. Numbers correspond to the time steps in the experiment

testing software in 10 sessions for about 30∼min, on average. The subjects sat in front of a cluttered background and faced the external monitor that contained the test software that we developed. The test supervisor faced the laptop monitor that contained the Augmented Camera Mouse interface. This interface showed the current tracking positions overlaid on the webcam image (Fig. 1, bottom). If the Augmented Camera Mouse lost the selected feature, the supervisor would record the event as a tracking failure and reinitialize the mouse pointer by manually resetting the tracking position to the appropriate image feature.

The experiments involved five sessions:

- **Normal session**. The subject was instructed to move the mouse pointer to a series of 20 randomly placed targets. This session represents the typical motions and orientations which a Camera Mouse user would encounter in day-to-day operations (Fig. 7a).
- **Hastened session**. A total of 20 targets were placed alternatively on the left and right side of the screen. The subject was instructed to move the mouse as quick as possible. This session was designed to induce large horizontal motions (Fig. 7b). We chose not to use vertical motions to decrease neck strain in the users.
- **Boundary session**. This session was designed to have the subject occlude large portions of his or her face due to moving the head in extreme positions. A total of 20 targets were placed along the boundary of the screen (Fig. 7c).
- **Changed lighting session**. The subject was instructed to move to the same target arrangement as those of the normal session. The overhead lights in the room were turned off to create darker lighting condition than that during the setup phase (Fig. 7a).
- **Changed scale session**. This session used the same arrangement as the normal session, but with the camera moved two feet away from the subject. This resulted in smaller scaled features (Fig. 7a).

For consistency, the order of the sessions and trackers was fixed for all subjects. We first worked with the Standard Camera Mouse and then the Augmented Camera Mouse. We tested the perfomance of a given tracker in only the sessions that were appropriate for it. We tested the Augmented Camera Mouse with the Threshold Kernel, the

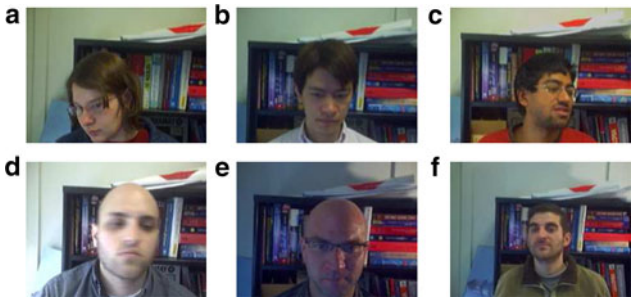

Fig. 8 Sample images captured by the webcam during the testing phase. The images show different head orientations (**a**, **b**, **c**), rapid motions (**d**), changed lighting (**e**), and changed scale (**f**). All subjects were tested in front of the cluttered bookcase shown in the images

Normalized Threshold Kernel, and the Normalized Radial Intensity Kernel, defined in Sect. 5 Using the Augmented Camera Mouse with the Normalized Threshold Kernel in the normal and changed-lighting sessions, we tested the invariance of the kernel to differences in feature illumination. In the changed-scale sessions, we tested the invariance of the Normalized Radial Intensity Kernel to changes in size of the tracked feature. We also compared the performance of the Standard Camera Mouse and Augmented Camera Mouse with the threshold kernel during the normal, hastened, and boundary sessions.

During each session the Augmented Camera Mouse used 25 training images and subset projections of size 5. The limit for the number of steps of the hill climbing algorithm for any video frame was set to 10.

The facial feature tracked was the inner left eyebrow corner. We selected it since it is centered in the face, and not likely to be occluded. From our experience, when the eyebrow was the feature tracked, subjects required less cognitive processing in converting head motions to mouser pointer motions.

8.5 Analysis procedure

To evaluate the tracking accuracy of the Augmented Camera Mouse, we compared computed feature positions against manual "ground-truth" of feature locations. For each session, an image of the webcam was saved once per second. After the session was over, an independent observer used a custom program to mark the location of the facial feature in each image. For each session, the average Euclidean distance between the target locations and the manually marked locations was computed. We use this distance to represent the error of the tracker with regard to the hand-marked "ground truth."

We also evaluated the potential of "feature drift," in which a tracked point diverges away from the initially selected feature. The issue of feature drift particularly

arises when trackers are used for extended periods of time. The drift measure can be approximated by the increase of the error of a tracker over time. For each subject session, feature drift is determined by the slope of the best linear fit of the error, as computed above, versus time into the session. Feature drift is measured in units of pixels per second.

Between 18 and 64 images were saved per session, with an average of 34 images. The average time to manually mark the ground truth for each subject was 45 min.

We evaluated the benefit of the Augmented Camera Mouse with an HCI theoretic performance measure known as the *Index of Performance* [24]. This measure describes the performance of one or many users with a particular device. The Index of Performance is also known as the bandwidth of the device, with units in bits per second. The measure is similar to the performance indices of the electronic communication devices, with larger values signifying better performance.

The Index of Performance can be approximated using Fitts' law [13]. Fitts' law says that for pointing devices, the average time it takes a user to use a device to point to a target is linearly related to the level of difficulty of the task. It can be stated succinctly as

$$MT = c_1 + c_2 \times ID, \tag{4}$$

where MT represents the (mean) time to reach a target, ID is the index of difficulty of reaching the target, and c_1 and c_2 are constants dependent on the device and the user. Of the many variants of the index of difficulty, we use an information theoretic formulation [24, 25],

$$ID = \log\left(\frac{D}{W} + 1\right), \tag{5}$$

where D is the distance to the target and W is the diameter of the target. The Index of Performance (IP) for a particular user and device is

$$IP = 1/c_2, \tag{6}$$

with units of bits per second. We found the Index of Performance experimentally by collecting the behavior of our group of subjects performing a number of *actions* with a particular device. For our purposes the device is the Standard or Augmented Camera Mouse with different kernels. An action represents the task of moving the mouse pointer to a target. A user performing the mouse tracking experiment with one of the target arrangements shown in Fig. 7 produces 19 actions. Each action is represented by a (*Movement Time*, *Index of Difficulty*) pair, which contains the time to move the mouse from the previous to the new target position and the Index of Difficulty of the task, as described in Eq. 5. The terms W and D are the width and distance between the targets in screen pixels, with ranges of [100, 200] and [128, 976], respectively.

8.6 Results

Using the KERNEL-SUBSET-TRACKER with the threshold kernel, the Augmented Camera Mouse achieved a frame rate of 30 fps. The other kernels defined in Sect. 5 are more computationally expensive, but still achieved a frame rate of 30 fps.

We evaluated the tracking accuracy of the Augmented Camera Mouse (Table 1). The Augmented Camera Mouse with the threshold kernel during the normal, hastened, and boundary sessions performed with an average Euclidean error distance of 6.1, 7.9, and 7.7 pixel widths, respectively. On average, the width of the eyebrow of the subjects was 63 pixels. The error in localizing the eyebrow corner was therefore only about 1/10 the length of the eyebrow, implying the Augmented Camera Mouse tracked the left eyebrow with a high degree of accuracy.

The pairwise difference in accuracy of the Augmented Camera Mouse with the threshold kernel versus the Standard Camera Mouse was statistically significant. In the random, hastened and boundary sessions, the (p, $t(18)$) results were (0.004, 0.002), (0.006, 0.003), and (0, 0) respectively, based on a t test with 18 degrees of freedom.

The Augmented Camera Mouse was empirically shown to be very resilient to feature drift (Table 2). The average feature drift for all configurations used by the Augmented Camera Mouse was very close to zero, except for the hastened session with a modest drift of 0.1 pixels per second. The pairwise difference of feature drift of the Augmented Camera Mouse with the threshold kernel versus the Standard Camera Mouse was statistically significant, with $p = 0.0$, $t(18) = 0.0$ in the random and boundary sessions. For the hastened sessions a weak statistical significance was found, with $p = 0.22$ and $t(18) = 0.11$.

We empirically tested the invariance of the specialized kernels to changes in lighting and scale. The average error of the normalized threshold kernel was comparable to average error of the regular threshold in the normal sessions, in terms of average error. The normalized threshold kernel was shown to be generally invariant to changes in lighting conditions. The average error in the changed lighting session increased by 58 % to 9.2 $\pm$ 7.1 pixels. The average and variance of the feature drift are equal in the normal and changed lighting sessions with the normalized threshold kernel. Their pairwise difference had $p = 0.82$ and $t(18) = 0.41$, indicating no statistical significance.

The Normalized Radial Intensity Kernel (NRI Kernel) proved to be very effective in tracking the eyebrow at different distances from the camera. The average tracker error of the NRI Kernel for the normal and changed scale sessions decreased from 6.5 pixel widths to 5.6 pixel widths. The increased distance of the users to the cameras,

Table 1 Tracking error

Tracker error (pixels)					
Tracker	Sessions				
	Normal	Hastened	Boundary	Changed lighting	Changed scale
Standard Camera Mouse	9.7 ± 6.2	13 ± 5.8	13 ± 5.9	×	×
Threshold kernel	6.1 ± 2.7	7.9 ± 2.6	7.7 ± 2.7	×	×
Normalized threshold kernel	5.8 ± 2.5	×	×	9.2 ± 7.1	×
Normal radial intensity kernel	6.5 ± 1.7	×	×	×	5.6 ± 2.2

Average and standard deviation of the Euclidean distance in pixels widths between the feature position estimated by the tracker and the ground truth marking. Each session had 19 subjects and on average 34 images. The field of view of the webcam is 640 by 480 pixels. The × symbol marks sessions that were not tested

Table 2 Drift error

Drift error (pixels per second)					
Tracker	Sessions				
	Normal	Hastened	Boundary	Changed lighting	Changed scale
Standard Camera Mouse	0.22 ± 0.29	0.25 ± 0.38	0.37 ± 0.31	×	×
Threshold kernel	0.0 ± 0.07	0.1 ± 0.24	0.03 ± 0.1	×	×
Normalized threshold kernel	−0.02 ± 0.16	×	×	0 ± 0.1	×
Normalized radial intensity kernel	−0.02 ± 0.11	×	×	×	−0.01 ± 0.1

The drift metric represents the rate of error increase of a tracker over time. For each subject session, the feature drift is determined by the slope of the best linear fit of the error versus time into the session. The values below represent this feature drift, averaged over 19 subjects. They are in units of pixels per second. The error is determined as for Table 1

which results in smaller faces in the captured image, is a likely reason for the decrease. Similar results were achieved for the feature drift of both sessions with the NRI Kernel. A pairwise comparison resulted in $p = 0.69$ and $t(18) = 0.34$. indicating no statistical significance in the difference of drift.

Both the Augmented Camera Mouse and the Standard Camera Mouse had occasional tracking failures. In particular when the subject had extreme motions, we measured the same number of tracking failure losses in the Standard Camera Mouse and with the Augmented Camera Mouse using the threshold kernel. The Standard Camera Mouse had three tracking failures in the hastened sessions. The Augmented Camera Mouse with the threshold kernel had one tracking failure in the normal sessions and two in the hastened sessions. The tests for lighting and scale invariance resulted in a single extra tracking loss in one of the changed lighting sessions.

The Index of Performance of the Augmented Camera Mouse was derived from the inverse slope of the best linear fit of the actions (Fig. 9). The Index of Performance of the Augmented Camera Mouse was higher than the Standard Camera Mouse in the Normal and Boundary Sessions, e.g., 2.9 bits/s versus 1.4 bits/s (Table 3). In both sessions, users were instructed to move naturally. This indicates when the users did not rush with the devices, they performed the tasks quicker with the Augmented Camera Mouse than the Standard Camera Mouse. In the hastened sessions, we instructed users to move as quick as possible, and devices had equal Indices of Performance, due to the rushed motions of the users. Sessions using the Normalized Threshold and Normalized Radial Intensity Kernels had performance measurements lower than the Threshold Kernel, but higher than the Standard Camera Mouse. The changed lighting and scale sessions resulted in slightly lower performance of the Augmented Camera Mouse.

We did not randomize the order of the experiments. During the experiments, increased familiarity of the users with the Camera Mouse may cause them to naturally move the mouse quicker in the sessions at the end of their time with the trackers. This results in a potential source of bias for Table 3. To address this issue, we examined the average acceleration of mouse pointer movements. This measure is the increase in speed of the movement of the pointer controlled by subjects within a particular session and it indicates the rate of learning of the users. The average acceleration can be approximated by the slope of the best linear fit of actions in a session. Each action is plotted by

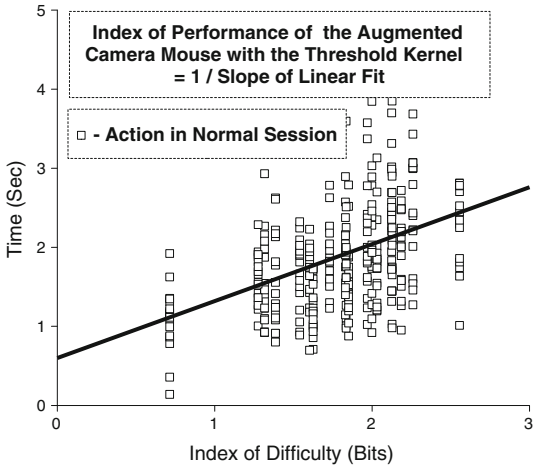

Fig. 9 Index of Performance of the Augmented Camera Mouse with the threshold kernel in normal sessions. Each point represents the *action* of a user in the Normal Sessions, who directs the mouse to a target, with 400 actions total. The Index of Difficulty (ID) of each action, corresponding to the size and the distance of the target. For each action, a higher ID is correlated to more time to reach the target. The Index of Performance represents the bandwidth of the device, and is the reciprocal of the slope of the best linear fit of the actions

the mouse speed of the pointer during the action (in units of pixels per second) versus the occurrence time of the action in the session (in units of seconds). The average increase of speed across all users is in units of pixels per second squared.

The average acceleration of the users was heavily correlated to the session type and not the tracker used (Table 4). Users had average accelerations close to zero when instructed to move naturally (in the normal and changed lighting and scale sessions), so the bias can be discounted for those sessions. Users had the same average acceleration for the boundary sessions with both systems, indicating no relative bias. Users had high average accelerations for the hastened sessions with respective rates of 6.9 and 11 pixels/s^2 for the Standard and Augmented Camera Mouse, indicating the possibility of a comparative bias between the hastened sessions. From results of Table 4, we showed that learning was not a significant factor for bias in Table 3.

9 Experiment with subject with severe motion impairments

We worked with a quadriplegic subject whose voluntary motion is severely limited due to a massive stroke, which had occurred four years earlier. The subject communicates with friends and family members through eye and eyebrow motions. In our experiments, we used a blink detection method [28] to automatically find the eyes of the subject and then tracked the subject's eyebrow motion with the Augmented Camera Mouse. Since the eyebrow motion was mostly vertical, see Fig. 10, the conversion of this motion into mouse pointer coordinates would only enable up- and down cursor movements. We needed to adjust our experiment to the subject's movement abilities. We therefore simplified the interaction mechanism and worked with test programs that only required mouse clicks and not mouse pointer positions as inputs. Our system automatically interpreted raised eyebrows as mouse clicks. Click events were sent to a text-entry program called CUSTOMIZABLE KEYBOARD [29].

CUSTOMIZABLE KEYBOARD is a scan-based on-screen keyboard that can be adapted to the user's motion abilities. It is similar to virtual scanning keyboards analyzed by [7]. Using the Augmented Camera Mouse with the CUSTOMIZABLE KEYBOARD, the subject was able to spell out words by raising his eyebrows and thereby selecting highlighted letters during a scan of the alphabet.

The eyebrow was tracked using the Augmented Camera Mouse, in the same configuration as described in Sect. 8.4 The KERNEL-SUBSET-TRACKER was used with the threshold kernel. A training set of size 25 was used with a real-time subset of size 5. The training set consisted of images of size 100×100 centered at the subject's eyebrow.

The user task during the training phase, as described in Sect. 7 had to be adjusted for our subject due to his limited movement abilities. To enable the Augmented Camera Mouse to collect training images, we asked the user to look at the camera, blink a few times, and then raise his eyebrows. The central location of the eyebrow was detected using an automatic feature locator that is based on a blink detection method [28]. A representative set of images of the subject's eyebrow in the raised and normal states was collected every second for 25 s while the subject moved his eyebrows up and down.

During the test phase of the experiment, the subject generated click events by raising and lowering his eyebrows. Upward motions of the tracked feature on the eyebrow would trigger a click event (Fig. 11). In every frame, the system determines the vertical difference Y between the position of the eyebrow in current frame and in the previous frame. The "raw Y movement" is smoothed using a moving average of period 20 with exponentially decreasing weights.

Before the subject could use the Augmented Camera Mouse as an interface, we needed to specify a threshold for the range of motion that was comfortable for him and that could be mapped accurately to a click command. We set the click threshold manually using the pop-up window shown in Fig. 12.

The subject used the Augmented Camera Mouse in two test sessions. The first session lasted 4.7 min and the

Table 3 Index of performance

Index of performance (bits per second)					
Tracker	Sessions				
	Normal	Hastened	Boundary	Changed lighting	Changed scale
Standard Camera Mouse	1.4	0.94	2.1	×	×
Threshold kernel	2.9	0.87	2.4	×	×
Normalized threshold kernel	1.9	×	×	1.7	×
Normal. radial intensity kernel	1.7	×	×	×	1.6

The rows represent the type of system used for tracking. The columns represent the tracking session performed. The values represent the Index of Performance or the bandwidth of the device, in units of bits per second. The Index of Performance is roughly the ratio of the difficulty of reaching a target to the time to reach it, with larger numbers signifying better performance

Table 4 Learning bias in index of performance measurements

Average acceleration of mouse pointer (pixels/s^2)					
Tracker	Sessions				
	Normal	Hastened	Boundary	Changed lighting	Changed scale
Standard Camera Mouse	−0.06	6.9	5.4	×	×
Threshold kernel	−0.23	11	5.4	×	×
Normalized threshold kernel	0.14	×	×	0.64	×
Normal. radial intensity kernel	0.82	×	×	×	0.35

The rows represent the type of Kernel-Subset-Kernel used for tracking. The columns represent the tracking session performed. The values represent the average acceleration of mouse pointer speed of the subjects, which is used to approximate the average rate of learning of the subjects. The results show that the learning of the subjects had minimal impact on the bias of the Indices of Performance measurements (Table 3)

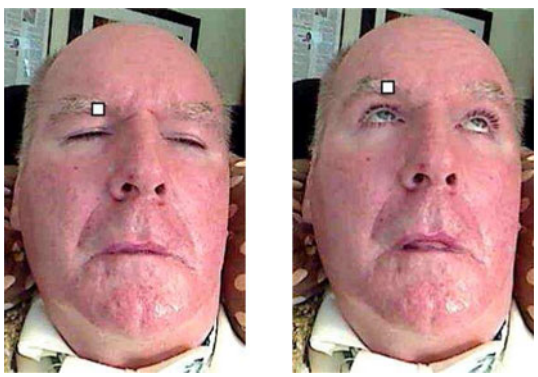

Fig. 10 The Augmented Camera Mouse was used to track the eyebrow of a subject with movement disabilities. The vertical motions of the subject's eyebrow were translated into mouse click events

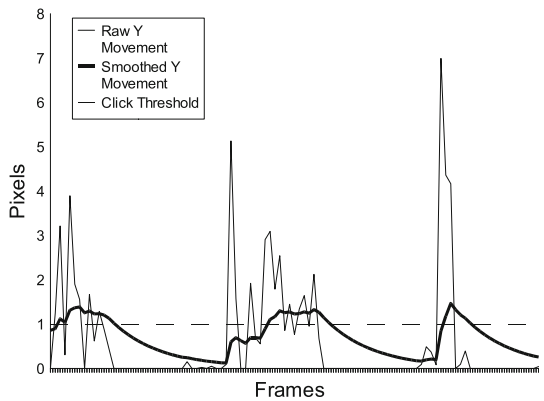

Fig. 11 The difference in Y positions of the feature between frames is represented by "Raw Y Movement". This value is smoothed using an exponential average, as represented by the "Smooth Y Movement". Click events are generated when the smoothed Y movement first transitions from under the click threshold to over it. In the example above, three clicks were generated

second session lasted 6.9 min. The Augmented Camera Mouse successfully tracked the user's eyebrow. The user was able to communicate by raising his eyebrow and selecting letters, spelling out words, and creating sentences.

To evaluate the tracking accuracy of the Augmented Camera Mouse, we compared computed feature positions against manual "ground-truth" markings of feature locations. For each session, an image from the webcam was saved once per second. After the session was over, an independent observer used a custom program to mark the location of the facial feature in each image. For both sessions, the average Euclidean distance between the target locations and the manually marked locations was

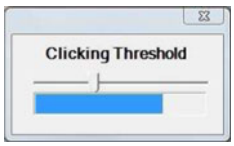

Fig. 12 A pop-up window is used to determine the user-specific threshold for click events. The *blue bar* represents the vertical distance of the tracked eyebrow from its neutral position. The slider knob pointing down represents the position of the click threshold. When the blue motion bar transitions from the *left* to the *right* of this position, a mouse click is generated. At the beginning of the experiment, we asked the subject to make eyebrow movements as if he intended to generate mouse clicks. By observing the range of the *blue bar* during his movements, we could determine a click-threshold position, as shown here, that was comfortable and effective for him (color figure online)

Table 5 Results of eyebrow clicking experiment with user with severe motion disabilities

	Eyebrow tracking sessions	
	Session 1	Session 2
Duration	4.7 min	6.9 min
Number of manually marked images	270	403
Average tracker error	4.6 pixels	4.2 pixels
Feature drift	0.0 pixels per second	0.0 pixels per second

computed. We also computed the feature drift, as defined in Sect. 8.4.

Our results (Table 5) show that the subject's eyebrow was tracked accurately by the Augmented Camera Mouse for the duration of the two test sessions. The average pixel error was very small and the feature drift was minimal.

10 Conclusions

We introduced the KERNEL-SUBSET-TRACKER, an exemplar tracker that uses kernel methods traditionally associated with classification. We showed that the KERNEL-SUBSET-TRACKER can maintain a sufficiently reliable tracking performance with a subset size of 5, given 25 training observations. The setup phase of the KERNEL-SUBSET-TRACKER is efficient and can be accomplished in real time.

We showed how the standard threshold kernel can be "normalized" to provide invariance to linear changes in brightness and contrast. As shown experimentally, the Normalized Radial Intensity Kernel is invariant to changes in scale. The NRI Kernel is computationally more expensive than the other two kernels, but it still maintains the same frame rate as the other kernels when used by the Augmented Camera Mouse. The use of the NRI Kernel is recommended in interaction scenarios where the user may move significantly towards or away from the camera. Additional kernels may be developed in the future that enable to the KERNEL-SUBSET-TRACKER to achieve invariance to other object transformations that represent user movement.

Our experimental results show that the Augmented Camera Mouse had no significant feature drift, and therefore was anchored to a particular feature, regardless of fast movement or extreme head positions. This is an improvement to the Standard Camera Mouse, which was subject to feature drift, even in the "normal" test sessions.

We tested the Augmented Camera Mouse with a user with severe movement disabilities. The Augmented Camera Mouse was shown to track the subject's eyebrow accurately, enabling him to communicate via mouse click events.

Acknowledgments The authors thank the test subjects for their assistance and Diane Hirsh Theriault for her feedback on the manuscript. The authors acknowledge funding from the National Science Foundation, HCC grants IIS-0713229 and IIS-0910908.

References

1. ABLEDATA: online database that provides information on assistive technology, US Department of Education (2010) http://www.abledata.org
2. Akram W, Tiberii L, Betke M (2006) A customizable camera-based human computer interaction system allowing people with disabilities autonomous hands free navigation of multiple computing tasks. In: Stephanidis C, Pieper M (eds) Universal access in ambient intelligence environments—9th international ERCIM workshop "User Interfaces For All" UI4ALL 2006, Königswinter, Germany, September 2006, revised papers. LNCS 4397. Springer, Berlin, pp 28–42
3. Bates R, Istance HO (2003) Why are eye mice unpopular? A detailed comparison of head and eye controlled assistive technology pointing devices. Univ Access Inf Soc 2(3):265–279
4. Betke M (2009) Intelligent interfaces to empower people with disabilities. In: Nakashima H, Augusto JC, Aghajan H (eds) Handbook of ambient intelligence and smart environments. Springer, New York, pp 409–432
5. Betke M, Gips J, Fleming P (2002) The Camera Mouse: visual tracking of body features to provide computer access for people with severe disabilities. IEEE Trans Neural Syst Rehabil Eng 10(1):1–10
6. Betke M, Gusyatin O, Urinson M (2006) SymbolDesign: a user-centered method to design pen-based interfaces and extend the functionality of pointer input devices. Univ Access Inf Soc 4(3):223–236
7. Bhattacharya S, Samanta D, Basu A (2008) Performance models for automatic evaluation of virtual scanning keyboards. IEEE Trans Neural Syst Rehabil Eng 16(5):510–519

8. Biswas P, Samanta D (2007) Designing computer interface for physically challenged persons. In: Proceedings of the international information technology conference (ICIT 2007). IEEE, New York, pp 161–166
9. Bourel F, Chibelushi CC, Lowe AA (2000) Robust facial feature tracking. In: Proceedings of the British machine vision conference (BMVC), vol 1. BMVA Press, UK, pp 232–241
10. Epstein S, Betke M (2009) Active hidden models for tracking with kernel projections. Technical Report BU-CS-TR-2009–006, Department of Computer Science, Boston University
11. Epstein S, Betke M (2013) The kernel semi-least squares method for sparse distance approximation. Neural Comput (in press)
12. Findlater L, Jansen A, Shinohara K, Dixon M, Kamb P, Rakita J, Wobbrock JO (2010) Enhanced area cursors: reducing fine pointing demands for people with motor impairments. In: Proceedings of the 23rd annual ACM symposium on user interface software and technology (UIST'10). ACM, New York, pp 153–162
13. Fitts PM (1954) The information capacity of the human motor system in controlling the amplitude of movement. J Exp Psychol Gen 47(6):381–391
14. Gips J, Betke M, Fleming P (2000) The Camera Mouse: preliminary investigation of automated visual tracking for computer access. In: Proceedings of the rehabilitation engineering and assistive technology society of North America 2000 annual conference (RESNA 2000). RESNA, Arlington, pp 98–100
15. Gorodnichy D, Dubrofsky E, Ali M (2007) Working with computer hands-free using Nouse perceptual vision interface. In: Proceedings of the international CRV workshop on video processing and recognition (VideoRec'07), Montreal, Candada, NRC, Canada. pp 1–12
16. Hwang F, Keates S, Langdon P, Clarkson J (2004) Mouse movements of motion-impaired users: a submovement analysis. In: Proceedings of the 6th international ACM SIGACCESS conference on computers and accessibility (Assets'04). ACM, New York, pp 102–109
17. International Organization for Standardization (2007) Ergonomics of human-system interaction—part 400: principles and requirements for physical input devices
18. iView X HED: SensoMotoric Instruments (2010) http://www.smivision.com
19. Kim H, Ryu D (2006) Computer control by tracking head movements for the disabled. In: 10th International conference on computers helping people with special needs (ICCHP), Linz, Austria, LNCS 4061. Springer, Berlin, pp 709–715
20. Kim WB, Kwan C, Fedyuk I, Betke M (2008) Camera canvas: image editor for people with severe disabilities. Technical Report BU-CS-TR-2008-010, Computer Science Department, Boston University
21. Kjeldsen R (2006) Improvements in vision-based pointer control. In: Proceedings of the 8th international ACM SIGACCESS conference on computers and accessibility. ACM, New York, pp 189–196
22. Loewenich F, Maire F (2007) Hands-free mouse-pointer manipulation using motion-tracking and speech recognition. In: Proceedings of the 19th Australasian conference on computer–human interaction (OZCHI). ACM, New York, pp 295–302
23. Lucas BD, Kanade T (1981) An iterative image registration technique with an application to stereo vision. In: Proceedings of the 7th international joint conference on artifical intelligence. IJCAI, California, pp 674–679
24. MacKenzie IS (1989) A note on the information-theoretic basis for Fitts' law. J Mot Behav 21:323–330
25. MacKenzie IS (1992) Fitts' law as a research and design tool in human–computer interaction. Hum Comput Interact 7(1):91–139
26. Magee J, Betke M (2010) HAIL: hierarchical adaptive interface layout. In: Miesenberger K et al (eds) 12th International conference on computers helping people with special needs (ICCHP 2010), Vienna University of Technology, Austria, Part 1, LNCS 6179. Springer, Berlin, pp 139–146
27. Manresa-Yee C, Varona J, Perales FJ, Negre F, Muntaner JJ (2008) Experiences using a hands-free interface. In: Proceedings of the 10th international ACM SIGACCESS conference on computers and accessibility. ACM, New York, pp 261–262
28. Missimer E (2010) Betke M Blink and wink detection for mouse pointer control. In: The 3rd ACM international conference on pervasive technologies related to assistive environments (PETRA 2010), Pythagorion, Samos, Greece. UTA, Arlington, Texas, p 8
29. Missimer E, Epstein S, Magee JJ, Betke M (2010) Customizable keyboard. In: The 12th international ACM SIGACCESS conference on computers and accessibility (ASSETS 2010), Orlando, Florida, USA. ACM, New York
30. Palleja T, Rubion E, Teixido M, Tresanchez M, del Viso A, Rebate C, Palacin J (2008) Simple and robust implementation of a relative virtual mouse controlled by head movements. In: Proceedings of the conference on human system interactions. IEEE, Piscataway, pp 221–224
31. Ponsa P, Manresa-Yee C, Batlle D, Varona J (2009) Assessment of the use of a human–computer vision interaction framework. In: Proceedings of the 2nd conference on human system interactions, IEEE, Piscataway, pp 431–438
32. QualiEye: Qualilife (2010) http://www.qualilife.com/en/accessibility-download.html
33. Quick Glance 3: EyeTech Digital Systems (2010) http://www.eyetechds.com
34. RED Eye Tracking System: SensoMotoric Instruments (2010) http://www.smivision.com
35. Schölkopf B, Smola A (2001) Learning with kernels: support vector machines, regularization, optimization, and beyond. The MIT Press, Cambridge
36. SmartNAV: NaturalPoint (2010) http://www.naturalpoint.com/smartnav
37. Trewin S, Keates S, Moffatt K (2006) Developing steady clicks: a method of cursor assistance for people with motor impairments. In: Proceedings of the 8th international ACM SIGACCESS conference on computers and accessibility (Assets '06). ACM, New York, pp 26–33
38. Tu J, Tao H, Huang T (2007) Face as mouse through visual face tracking. Comput Vis Image Underst 108(1–2):35–40
39. Varona J, Manresa-Yee C, Perales FJ (2008) Hands-free vision-based interface for computer accessibility. J Netw Comput Appl 31(4):357–374
40. Waber B, Magee JJ, Betke M (2006) Web mediators for accessible browsing. In: Stephanidis C, Pieper M (eds) Universal access in ambient intelligence environments—9th international ERCIM workshop "User Interfaces For All" UI4ALL 2006, Königswinter, Germany, September 2006, revised papers. LNCS 4397. Springer, Berlin, pp 447–466
41. Wobbrock J, Gajos K (2008) Goal crossing with mice and trackballs for people with motor impairments: performance, submovements, and design directions. ACM Trans Access Comput 1(1):1–37