A Keystroke and Pointer Control Input Interface for Wearable Computers

Farooq Ahmad
University of Alberta
Department of Electrical
and Computer Engineering
Edmonton, Canada

Petr Musilek
University of Alberta
Department of Electrical
and Computer Engineering
Edmonton, Canada

Abstract

The widespread adoption of mobile electronic devices and the advent of wearable computing has encouraged the development of compact alternatives to the keyboard and mouse. These include one-handed keyboards, digitizing tablets, and glove-based devices. This paper describes a combination pointer position and non-chorded keystroke input device that relies on miniature wrist-worn wireless video cameras that track finger position. A Hidden Markov Model is used to correlate finger movements to keystrokes during a brief training phase, after which the user can type in the air or above a flat surface as if typing on a standard keyboard. Language statistics are used to help disambiguate keystrokes, allowing the assignment of multiple unique keys to each finger and obviating chorded input. In addition, the system can be trained to recognize certain finger positions for switching between input modes; for example, from typing mode to pointer movement mode. In the latter mode of operation, the position of the mouse pointer is controlled by hand movement. The camera motion is estimated by tracking environmental features and is used to control pointer position. This allows fast switching between keystroke mode and pointer control mode.

1 Introduction

The increasing popularity of small portable electronic devices has driven demand for new input technologies. For example, handwriting recognition using pressure sensitive screens and a stylus has been successfully incorporated on many handheld computers. Onscreen keyboards and small 'thumboards' are another popular input method. However, input speed achieved with these devices is slow relative to standard keyboards [9]. The surface area available for input is even less on cellular phones, where 12 digit numerical keypads usually do double duty for text entry. Generally, when mobile phone keypads are used for text input, each key is mapped to three or four letters and predictive text entry algorithms are used to disambiguate keypresses. The application of predictive text entry methods to mobile phone text entry has significantly lowered the average number of keystrokes required per character (KSPC), from 3 KSPC to close to 1 KSPC. Nevertheless, key sequences for shorter words are often ambiguous, and the limited keypad size makes input speed slow relative to full-size keyboards. See [9] and [12] for a review of keypad based mobile input technologies.

Mobile computing and wireless networking technology has advanced the integration of computing into everyday life. Globally, over 1 billion SMS messages are sent per day, and mobile phone use for messaging and internet access continues to grow, motivating the need for simple and efficient portable input devices. An alternative to keypad based input is based on virtual keyboards, that do not rely on an input surface or physical keys to detect input. For example, the keyboard is projected onto a flat surface and keypresses detected by infrared ranging sensors in [16]. The increasing sophistication of mobile technology has also supported the emergence of multimedia, entertainment, and augmented reality applications for mobile computers. Augmented reality and wearable computers in particular require input interfaces that are lightweight and unobtrusive.

One barrier to the adoption of wearable computing devices is the absence of an input interface that allows the user to interact with the system while standing or moving. Typically, portable keyboards must be set or held in place to be used effectively. One approach to solving this problem is one-handed (split) keyboards. However, wearing a wrist-mounted split keyboard interferes with the use of the hands to do other tasks. Glove-based devices incorporating flex sensors within the glove fingers are less cumbersome, allowing free use of the hands. Another input method analyzes the data collected from tiny accelerometers worn on each finger to determine user input [3]. An even less obtrusive approach is virtual glove based input devices, that use wrist mounted sensors to track hand and finger posi-

Proceedings of the Fourth Annual IEEE International Conference on Pervasive Computing and Communications (PERCOM'06)
0-7695-2518-0/06 $20.00 © 2006 IEEE

tion. For example, the lightglove [6] uses infrared LEDs and sensors worn on the arm to track finger position and allows the control of electronic systems. The concept of using wrist mounted cameras as input devices is introduced in [21], where the fingers are tracked and used as input. Virtual glove based devices enable the user to interact with a mobile computer while moving, and allow free use of the hands while not in use. However, the devices mentioned above rely on a chorded input scheme to interpret finger movements. Since there are only 10 fingers available to represent all alphanumeric characters, several fingers must be moved simultaneously for each input. This requires the user to learn a new method of input.

The input system described in this paper can interpret non-chorded input, allowing the user to type as if using a standard keyboard, as each finger can be used to type several different characters. For example, a user trained on a QWERTY keyboard moves the little finger of the left hand to type the Q, A, and Z keys. Probabilistic models of finger movement and language are learned and used to resolve ambiguous inputs. Using a learned observation model also allows the mapping of finger movements to commands to be adapted to each user. Thus, QWERTY, DVORAK, or any user-defined keymappings can be used. The use of cameras accommodates accurate tracking of the position of each finger at all times. In addition, environmental features can be tracked to determine the movement of the hand itself. The incorporation of vision-based ego-motion estimation provides for an intuitive method of pointer control by translating hand movements to pointer movements. As a wearable input device, the complete system consists of two wrist worn cameras, a wearable computer used to process the images, and a head mounted display.

This paper is organized as follows. Section 2 describes the keystroke detection and recognition process. Section 3 describes how to switch between keystroke and pointer control modes, and section 4 details the implementation of pointer control by vision-based motion estimation. In Section 5, experimental results obtained from the prototype system are discussed. Conclusions and possible extensions are explored in Section 6.

2 Keystroke Detection and Recognition Overview

The prototype system is shown in figure 1. A wireless color camera is worn on the underside of each wrist. The images are transmitted to a receiver connected to a computer that processes the video stream in real-time. As keystrokes are being input, the hands can be moved in any desired position including by the users sides, and do not need to be held stationary.

The fingertips are tracked continuously, and keystrokes

Figure 1. The prototype

are detected when one or more fingers deviates from the rest position above a distance and speed threshold. Keystroke movements are assigned to characters during a training stage, and a Hidden Markov Model (HMM) is used for character recognition. The three stages of detecting and identifying keystrokes are described below.

1. Hand Extraction and Fingertip Recognition
   - The hand contour is extracted from the images by skin color discrimination;
   - The fingertips are detected as the minima of the extracted hand contour
2. Keystroke Detection
   - The finger rest position is recorded in an initialization stage;
   - Fingertip position is tracked over time, and keystrokes are recognized as peaks in the deviation of the finger position from the observed rest position
3. Hidden Markov Model Based Character Recognition
   - The correlation between finger movement and character/command input is observed during a brief training phase;
   - During operation, keystrokes are interpreted as characters and language statistics are used to disambiguate keystrokes.

2.1 Hand Extraction and Fingertip Detection

The video camera worn on the underside of each wrist continuously records the hand and fingers. The contour corresponding to the edge of the hand is extracted from the image stream by means of color segmentation. Pixels whose value lies within a predefined range corresponding to skin color are extracted, exploiting the alignment of the fingers

Figure 2. Stages in image preprocessing. The figure shows 1)the image after skin color detection, 2)after erosion and dilation, 3) the extracted edge contour and 4) the smoothed contour and extracted interest points

relative to the camera to aid in extraction. Since the fingers are aligned nearly vertically in the camera images, skin color extraction proceeds column by column starting at the top left corner of each image. Columns of the image are searched for consecutive sequences of skin colored pixels longer than a predefined threshold, and the bottommost run of pixels in each column is marked as an edge contour of the finger.

The minimum run length threshold affords some robustness to noise caused by variations in illumination conditions and similarly colored objects in the environment. Further invariance to changes in illumination can be achieved using an adaptive skin color classifier as in [17]. After the first stage of segmentation, the image is converted into a binary image. All pixels that are within the derived hand contour are non-zero within the binary image, and all pixels not considered part of the hand are assigned a value of zero. An erosion filter is then applied to the binary image to remove any remaining noise by eliminating small patches incorrectly identified as part of the hand. Then, a dilation filter is applied to fill in spots within the hand and fingers that were incorrectly classified as non-hand regions. Finally, the extracted edge contour is smoothed with a gaussian filter, and the contour minima are identified as fingertips and the maxima as phalanges. The preprocessing stages are shown in figure 2.

Four fingers and three phalanges are visible within the field of view of the camera, giving seven interest points to describe the hand and finger position. Figure 3 shows the fingertip and phalange detection, as well as the centroid of the interest points.

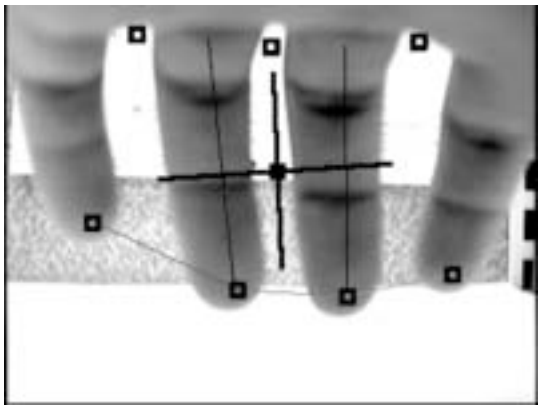

Figure 3. The seven interest points and detected orientation of the hand

2.2 Keystroke Detection

2.2.1 Initialization and Normalization

Observations of the system consist of the position and velocity of the four fingertips that are visible in the cameras field of view. Initialization of the system involves recording the rest position of each of the interest points. The coordinate system used for all measurements is defined by the centroid of the seven interest points and the alignment of the hand. A median filter is applied to the position measurement of the centroid so that it remains stationary during a keystroke. The centroid is used as the origin of the coordinate system, and the orientation of the axes is determined by the angle of the ring and middle fingers relative to the image plane. Normalizing the position of the fingertips relative to this coordinate system accounts for shifts in the position and alignment of the camera over time.

2.2.2 Peak Detection

After normalization, the position in polar coordinates, r_i and θ_i, and speed v_i of each fingertip relative to its rest position is tracked. Thus at any particular moment of time, each of the four finger tips $\mathbf{x_i}(i = 1..4)$ is defined by a 3 dimensional vector $< r_i, \theta_i, v_i >$

Keystrokes are detected as maxima in fingertip distances relative to their rest positions. Generally, one or more fingers will move for each keypress and the distance reading for all fingers will peak at approximately the same time. Peaks that occur within a five frame window, corresponding to a time frame of 0.17 seconds at 30 frames per second, are clustered and interpreted as a single keystroke.

There is considerable noise in the position readings, which would lead to spurious keystroke detections without smoothing the positions over time. A low-pass filter is ap-

plied to the fingertip positions to remove noise. For the system to be usable, keystrokes must be detected immediately and thus a causal filter with minimum lag is required. In addition, peak attenuation needs to be minimized in order to maintain accurate position measurements. An first order low-pass filter was used to attain the best compromise among noise reduction, peak attenuation, and delay characteristics.

2.3 Hidden Markov Model Based Character Recognition

Hidden markov models are commonly used in speech recognition and have found growing application in vision research as well. HMMs are particularly well suited to tasks involving the interpretation of noisy, continuous observations into small symbol vocabularies, such as in handwriting and gesture recognition. For example, HMMs have been used to recognize playback commands for a portable DVD player using embedded accelerometers [13]; head and hand gestures such as pointing, hand waving, and nodding [15]; and American Sign Language gestures using a fixed camera [18]. Typically, in machine vision based gesture recognition, a static camera is placed in front of the user and records movement of the hands, arms and head. The recognition system then interprets the sequence of movements as a set of words or commands.

The input to a handwriting or gesture recognition system consists of a continuous stream of data that must be segmented into meaningful units and interpreted as a sequence of discrete states. States can represent gestures, letters, words, or commands. In the case that the states are not directly observable and must be deduced, a hidden markov model can be used to determine the most probable underlying state sequence from a series of observations.

The states in an HMM are inferred on the basis of the observation model and transition model that are associated with each state. The observation model describes the relationship between system outputs and states, allowing the determination of the most probable state for any given output. The transition model consists of a table of probabilities of moving from one state to any other state.

To summarize, HMMs require the following elements

1. A finite number of states Q;
2. A set of output probabilities B, which may be discrete or continuous;
3. A set of state transition probabilities A

For keystroke recognition, each state in the HMM corresponds to a single alphanumeric character. At each detected keystroke, the most likely character is determined using the observation and transition models. As described above, the observation model relates system observables (fingertip positions and speeds) with hidden states (the characters that the finger movements represent). The observation model is learned during training by correlating each character in the training set with the output vector observed during its corresponding keypress. Often, two or more keys, such as the 'a' and 'z' keys, have similar observation vectors. The state transition model is used to disambiguate keystrokes. It is based on a language model that is derived from the analysis of a large corpus of text. The observation and transition models are described in detail below.

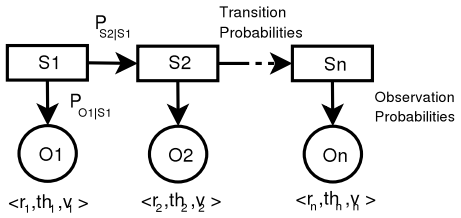

Figure 4. An HMM. For keystroke recognition, each state S is a character, the transition model is based on letter bigram probabilities, and the observation model contains finger position and speed vectors

2.3.1 Observation Model

Observations consist of a set of vectors that describe the position and speed of each fingertip during the instant of a keypress. Each observation is made up of four vectors, one for each finger, each containing three elements; the position of the finger relative to its rest position in polar coordinates, and the speed of the fingertip. In the current implementation, the thumb is not consistently present in the cameras field of view, and is ignored.

Finger tip positions and speeds constitute a continuous observation space, so that the observation model must be expressed in terms of a probabilistic function. In general, the observation symbol probability distribution B = $\{b_j(k)\}$ is denoted by

$$b_j(k) = P\left(o_t = \mathbf{k} | q_t = j\right) \quad (1)$$

$b_j(\mathbf{k})$ is the probability of observation $\mathbf{k}$ given that the current state is j. In the case of keystroke recognition, each observation $\mathbf{k}$ consists of 4 vectors, each containing the position and speed of a fingertip during a keypress. $\mathbf{k} = \langle \mathbf{k_1}, ..., \mathbf{k_4} \rangle$ where each $\mathbf{k_i} =< r_i, \theta_i, v_i >$, the fingertip position and speed relative to its rest position. Each state j is an alphanumeric character.

A 3D gaussian distribution describes the probability of observing $\mathbf{k}$ for each character j.

$$P(o_t = \mathbf{k}|q_t = j) = N(\mathbf{k} - \mu_\mathbf{j}, \sigma_\mathbf{j}) \quad (2)$$

where $\mu_\mathbf{j} =< \mu_r, \mu_\theta, \mu_v >_j$, is the mean finger position and speed for character j;
$\sigma_\mathbf{j} =< \sigma_r, \sigma_\theta, \sigma_v >_j$ is the variance; and
N is the gaussian function $\frac{1}{\sigma_j\sqrt{2\pi}} e^{-(\mathbf{k}-\mu_\mathbf{j})^2/2\sigma_\mathbf{j}^2}$

Thus given an observation vector $\mathbf{k}$, the probability that it represents character j is inversely proportional to the distance between $\mathbf{k}$ and the vector corresponding to the character, taking into account the variance of each vector element. The means and variances of the vectors associated with each character are learned during a short training stage, and continuously improved during operation. To shorten training time, the variances are set to uniform initial values for each finger.

During training, the user is asked to type a small paragraph of text. As the user types, keypress events are detected, and the current observation vector is associated with the next character in the training text. At the end of the training stage, each character has been typed several times and has several observations associated with it. On completion of training, each character is defined by the mean and variance of the set of observation vectors that was assigned to it.

Finger movements vectors that have a strong association with a particular character exhibit low variance, and are therefore most heavily weighted in the observation model described above. Similarly, finger movements that are are not consistently observed for a specific character have a higher variance, and do not factor as strongly towards identification of the character. For example, if the index finger consistently moves to the same position for a certain character, but the position of the ring finger varies widely, the observation model for that character relies more heavily on the index finger observation. Thus the system is robust to variations in finger movements during a keystroke, as long as at least one of the fingers moves in a consistent way for each letter. Figure 5 shows the observation vectors for the middle and index fingers for a subset of keys.

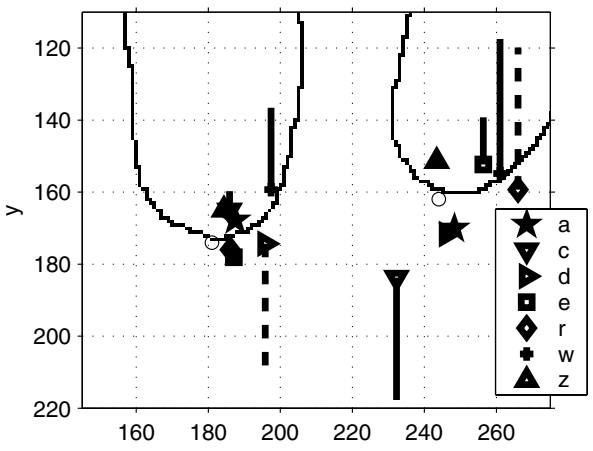

Figure 5. The mean position and speed vectors for the middle and index fingers of the left hand. The outline of the middle finger is on the left, and the index finger is on the right. Circles indicate the rest position, and other symbols represent letters. The length of the line attached to each symbol corresponds to the finger speed magnitude during a keystroke.

2.3.2 Character Disambiguation

The statistical regularities of language have been exploited for many purposes including spelling correction in word processors, automatic speech recognition, and predictive text entry in mobile phones. A language model can greatly aid in interpreting noisy or ambiguous text. For example, mobile phones are commonly used to send text messages and retrieve information from the internet. However, the limited number of keys on numerical keypads requires that several characters be mapped to one key, leading to ambiguous inputs. There are two main approaches to determining the desired text; dictionary-based approaches and statistical approaches. Dictionary-based predictive text entry involves searching a dictionary for words that are consistent with an observed sequence of key presses. Often several allowable words are found, and the user must make additional keypresses to select the correct one. In addition, the desired keys are not known until the entire word is typed. Statistical approaches to predictive text entry have attempted to address these issues by taking advantage of the statistics of word and letter sequences. Certain combinations of letters and words are more probable than others. Thus a probabilistic language model can be created by extracting the relative frequencies of letter and word combinations from a large body of text. With the aid of a probabilistic language model, ambiguous inputs can be resolved.

In this paper, the most probable word corresponding to a sequence of keystroke observations is determined in a two stage process using both word frequency and letter pair (bigram) frequency. While a word is being typed, individual keystrokes are disambiguated based on the previous characters using a letter bigram model (LBM). This model increases the accuracy of keystroke recognition substantially relative to a naive system that relies only on observations. However, the recognition rate can be even further

improved by incorporating word frequencies into the probability model. This is accomplished by considering the most likely character sequences determined by the letter bigram model, and rescoring them using a word frequency table. The two components of the language model are described below.

The letter bigram transition model estimates the most probable current input character l_n based on the previously observed character l_{n-1}:

$$P(l_n|l_{n-1}) = \frac{C(l_{n-1}l_n)}{C(l_{n-1})}; \quad (3)$$

where $C(l_{n-1}l_n)$ is the number of times that the bigram $l_{n-1}l_n$ is observed in a large corpus of written text. The generated model consists of a table $A = a_{ij}$ of letter pairs and their probability of occurring. Each table entry a_{ij} contains the probability the letter j follows letter i.

After each detected keystroke, the generalized Viterbi algorithm [5] is used to determine the most likely character according to the observation and transition models. The algorithm calculates the probability of each character by combining information from the two models, and orders the characters by probability. The most likely character is displayed on the screen.

The possible letter sequences can be considered as a trellis with weighted links between the candidate characters for each keystroke. Part of a possible trellis structure generated after observing four keystrokes is shown in figure 6. The strength of the observation probabilities are represented by circle thickness, while the transition probabilities correspond to line thickness. While characters in the top row of the trellis have the highest observation probabilities, the best path through the trellis relies on both the observation and transition models. In this case the best path is d-r-a-w. As each keystroke is detected, the Viterbi algorithm identifies the most probable character by considering:

1. The probability of each previous character i
2. The transition probability a_{ij} from each previous character to each current character j
3. $P(\mathbf{o}|j)$, the probability of the finger movement observation vector $\mathbf{o}$, given that the current character is j

The above probabilities are combined to find the likelihood of each character. For the very first letter in a word, only the observation probability is considered. Finally, when the space character is observed, the sequence is terminated. The set of the m-most likely letter sequences, $\{w_m\}$, only some of which are valid English words, is saved. The Viterbi algorithm is explained in depth in [14].

The letter bigram model provides a good statistical model of letter sequences within words. However, by incorporating a higher level model of word probabilities the accuracy of the system can be further improved. So, each letter sequence $w_m =< l_1...l_n >$ generated by the generalized Viterbi algorithm is rescored according to a word unigram model (WUM). The word model is a table of word frequencies derived from analyzing a large section of text. The probability of word w_m is defined as the number of times the word has been observed divided by the total number of tokens in a body of text.

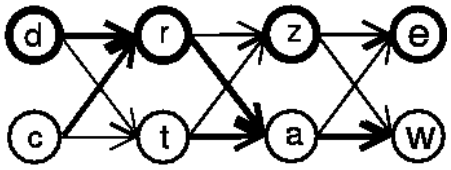

Figure 6. A portion of a character trellis. Circle boundary thickness represents observation probability and line thickness represents transition probability

$$P(w_m) = \frac{C(w_m)}{N} \quad (4)$$

where $C(w_m)$ is the count of word w_m, and N is the total number of tokens. The language models described above are created using the British National Corpus, a 100 million word sample of written and spoken English derived from a wide variety of sources. In [10] the BNC is tokenized and composed into a list of words ordered by frequency. The comprehensive word frequency list contains 939,028 unique words, including many proper names and colloquialisms. However, the rapid evolution of language on the internet continuously introduces new words and names that are often not included in large compiled lexica. One method of keeping an up-to-date lexicon is to integrate search query logs into the word list. Live updates of search queries are readily accessible online[1] and can be logged to generate a continuously evolving vocabulary.

The information provided by the letter bigram model LBM and word unigram model WUM is combined to determine the most probable word. The logarithm of the probabilities derived from each model are added to determine the final score, and the word with the maximum score is displayed.

$$W^* = argmax_n[P(w_n|LBM) + \alpha P(w_n|WUM)] \quad (5)$$

α is a weighting applied to the word model to determine its relative bearing on the score. W^* is the most likely word as determined by combining the results of the Viterbi algorithm and word model.

[1] www.metaspy.com

3 Mode Selection

During training, finger positions are used to select between the different modes of operation. Modes include alphabetical text entry, pointer control mode, and sleep mode. The system enters pointer control mode upon recognition of a specific finger position, and returns to text entry mode when the fingers return to the rest position. Figure 7 shows the open-handed finger position that has been trained to initiate pointer control mode. In sleep mode, finger movement and motion tracking are not used for input. Instead, the fingers are tracked passively until the keystroke assigned to resume input is detected. The pointer control mode is described below.

4 Pointer Control

Several pointer input devices have been developed to perform the task of pointer control on graphical displays. Pointer input devices designed for mobile computers aim to emulate the speed, accuracy, and intuitiveness of the standard computer mouse but in a more compact form. On notebook computers, trackballs, trackpads and micro-joysticks have enjoyed the most success, while touch sensitive screens and styluses are mainly used with handheld computers. The limited screen sizes on cellular phones has allowed pointer keys to function reasonably well for pointer control, but the convergence of handheld and mobile phone technology has driven the integration of large graphical displays. Consequently, accurate pointer control has become more significant.

In addition, wearable computer systems often employ head-mounted displays (HMD) that project an image directly in front of the eye, rather than on a large surface. In such a setup, touch sensitive screens can not be used. For wearable systems, pointer control by hand movement is a more convenient approach. The movement of the wrist-worn camera itself, rather than the fingers, is used to control the pointer.

4.1 Description of Operation

Pointer control is accomplished by means of vision-based ego-motion estimation using the video from one of wrist-worn cameras. In pointer control mode, the user's hand movements are estimated in order to generate a proportional movement in the position of the pointer on the screen. Pointer control is initiated upon recognition of a specific finger position that the system has been trained to recognize in the observation model training stage described above. As long as the finger position corresponding to pointer control mode is maintained, the motion of the hand is used to control the speed and direction of the pointer. The motion of the wrist-worn camera is determined by selecting salient features from each frame of video, and tracking each feature over time. Pointer movement control is thus accomplished in four stages: feature extraction; feature tracking; camera motion estimation using tracked feature points; and scaling the camera movement to move the onscreen pointer. Button presses are identified by finger movements in the same way that keystrokes are recognized.

4.2 Feature Extraction and Tracking

To determine the motion of the wrist worn camera, environmental features that can easily be recognized across frames must be detected and tracked. Since the position of the user's hand is fixed relative to the camera, the first step of feature extraction is to remove the hand from each frame of video, leaving only the background. Since the hand has already been detected in the fingertip detection stage, the hand shape simply needs to be subtracted from each frame.

The next step involves extracting features from the image sequence that are easy to track from frame to frame. Some candidate features include lines, corners, and more complex geometric features. The more unique the feature, the more reliably it can be tracked. However, detecting and tracking complex features requires increased computation time. In this paper, corners are detected using the Harris corner detector [4], providing a balance between detection speed and tracking robustness.

The Harris detector is used to detect features in the image that have strong gradients in two orthogonal directions. The best corners in each frame, along with the surrounding block of pixels are selected as good features and saved.

4.3 Motion Estimation

In each successive frame, each corner feature is compared to corners within a local neighborhood of the initial feature position. The similarity of the feature template to each candidate corner is determined using a sum of squares difference (SSD) error measure. The candidate feature that is most similar to the template is selected, and the feature position is updated. Features for which no match are found are considered lost, and new features are acquired to replace them. Generally, 70% of features are kept from one frame to another. The movement of each tracked feature provides information about the cameras direction of movement. However, some of the feature matches will be incorrect. The Random Sample Consensus (RANSAC) algorithm [2] is used to determine the camera motion that is consistent with a large number of tracked features.

In the current implementation, the image motion is fit to a two dimensional movement model; camera motion perpendicular to the image plane is ignored. The RANSAC

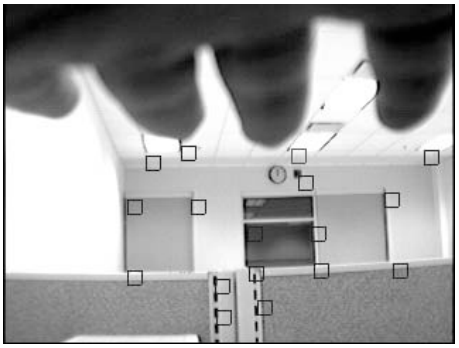

Figure 7. Corner Tracking in Pointer Control Mode

algorithm determines the most likely movement as follows:

1. Select one of the feature movement vectors at random and use it as the movement model.
2. Count K, the number of movement vectors that fit the model within a given tolerance.
3. If K is bigger than the minimum desired consensus K_{min}, exit successfully
4. Otherwise, repeat the above steps I times or until a correct model is found

In practice, the parameters K_{min} and I are set to attain a balance between accuracy and computation time. The higher K_{min} and I are set, the more likely the RANSAC algorithm will determine the correct motion.

4.4 Pointer Movement and Button Presses

After determining the most probable magnitude and direction of motion, the movement vector is scaled and applied to the pointer position. During training, the finger position corresponding to a mouse button click is recorded. As long as this finger position is maintained, the virtual button is held in the depressed position.

5 Experimental Results

A prototype system was implemented on a single wrist-worn color wireless camera. The camera captures images at a frame rate of 30 fps at a resolution of 320x240 pixels. The receiver was connected to a PC equipped with a video capture card, and processing was done on a 1533 MHz AMD Athlon system. The prototype system is shown in figure 1. During the evaluation, the camera was worn on the left hand and the training data was displayed on a monitor. Testing of the keystroke input and pointer control modes was done in a well-lit indoor environment.

Table 1. Character Recognition Accuracy

Language Model	No Language Model	Letter Bigram	Letter & Word Model
Accuracy [%]	67	84	90

5.1 Typing

The letter bigram and word unigram models were derived from the British National Corpus as described in section 2.3.2. For the purposes of training and testing one-handed input, a selection of text that can be typed using only the left hand was extracted by selecting words consisting exclusively of characters from the following set

$$Q, W, E, R, A, S, D, F, Z, X, C, V$$

A touch-typist was trained on a 300 word test set over a period of one hour. Each character was typed approximately 25 times, and the mean and variance of each finger movement vector was calculated after the completion of the training stage.

After training, a 200 word test file was used to measure recognition accuracy and speed. In the testing stage, after each keystroke is detected, the most likely character was displayed on the screen. Finger movements incorrectly recognized as keystrokes (false positives) were classified as errors, while undetected keystrokes were retyped. The right hand was used to click a mousebutton to signal the completion of each word, at which time the most likely word was displayed on the screen.

5.1.1 Character Recognition Accuracy

Table 1 describes the character recognition accuracy with and without the help of the language models. As the table shows, the observation model alone provides a character recognition rate of 67%. Character recognition errors are a result of the ambiguity and high variance of keystroke movements. In general, sets of characters that are typed with the same finger on a qwerty keyboard, such as {e,d,c} exhibit similar observation vectors and are easily confused. Of the incorrectly recognized words, 16 were due to character substitution errors, and 6 were due to spurious keystrokes. Several methods could be used to improve the accuracy of the observation model. For example, including angular direction rather than just speed v in the observation model provides more information to help discern keystrokes. Nevertheless, while the observation model alone does not always recognize the correct character, the letter bigram transition model improves the recognition rate

substantially to 84%. The word model further improves the character recognition rate. Thus in some cases, the character is misidentified immediately after a keystroke, but is corrected as soon as the word is completed. In the current implementation, the end of each word is indicated with a mouse button press. Another alternative is to assign a chorded finger movement to represent a space character, or a thumb movement if a camera with a wider field of view is used. In practice, usability can be improved by displaying the three most probable words and allowing the user to choose the correct one in very ambiguous cases. In addition, explicitly including the probability of misdetected keystrokes in the observation model is expected to further improve the accuracy of the system.

5.1.2 Input Speed

A typing speed of 14 words per minute was achieved on the prototype system. The smoothing filter described in section 2.2.2 is one source of lag, introducing a delay of approximately three frames (0.1s) between a keystroke and its detection. Missed keystrokes, which need to be retyped, are another source of delay. In the current implementation, a detected keypress is indicated by displaying the most likely character on the screen. The absence of the tactile feedback is another possible limitation to input speed on virtual keyboard and glove-based devices. Nevertheless, the input speed for the reduced character set tested here is comparable to other input methods for portable devices. Furthermore, the addition of acoustic feedback after each detected keypress is expected to improve input speed. In [22], the input speed and accuracy for several mobile phone text entry methods is compared.

5.2 Pointer control

The speed and control of pointer movement were tested qualitatively. The finger position assigned to pointer control mode was an open hand as shown in figure 7 and was detected consistently. In each frame, 20 corner features were selected to be tracked. On average, 70% of the selected features are matched from one frame to another. Accurate control can be achieved with smooth hand movements. However, rapid hand movements hinder accurate feature tracking, resulting in pointer position lag. In addition, feature correspondence errors sometimes result in erratic pointer motion. Several improvements can be made to improve performance. For example, using more unique feature descriptors will improve feature matching between frames. In addition, while corner features are appropriate for tracking in indoor environments, other feature descriptors are more suitable for less structured outdoor environments. Inertial measurements can also be used for ego-motion estimation, as in [13]. Integrating vision and inertial sensor based methods can be used to further improve accuracy. An overview of sensor fusion techniques for motion estimation is given in [19].

6 Conclusions and Future Work

This paper describes a keystroke and pointer control input device for mobile and wearable computers. Wrist mounted cameras are portable and unobtrusive, and allow rapid input using all fingers. Probabilistic character disambiguation allows non-chorded input, providing touch-typists with a smooth transition from a traditional keyboard. While input speed lags that possible with a standard keyboard, performance is comparable to other portable input methods. In addition, integrated pointer control allows for fast switching between typing and pointer control that is appropriate for augmented reality systems. Moreover, the hands do not have to be held in a specific position during typing, so that input can be entered while the user is standing or walking with their hands at their sides.

Several improvements can be made to each element of the system. For example, hand and finger recognition can be made more robust by implementing adaptive skin color classification. Work has been done to accurately detect varying ranges of skin color and perform accurate skin color extraction in all illumination conditions. Furthermore, contour tracking algorithms using deformable templates or dynamic contours [7] could be applied to more accurately track the edges of each finger. The Condensation (Conditional Density Propagation) algorithm [8] is a probabilistic algorithm that can be used to track contours in cluttered environments. Alternately, in [20], the circular Hough transform is used to track fingertip position for a virtual drawing application. Finally, fitting the observed hand contour to a 3D physical model of the hand would allow finger position to be tracked in three dimensions.

Several improvements to pointer control are also possible. The feature tracking approach described in this paper uses a two dimensional model of camera movement. Structure-from-motion techniques can be used to extend motion tracking to three dimensions. In [1], real-time structure-from-motion is achieved by tracking corner features and determining 3D position using a Kalman filter. Using more distinctive features rather than corners, as in [11], could enable more accurate tracking, and smoother pointer control. Inertial sensors such as accelerometers could also be used to aid motion estimation.

7 Acknowledgements

Support for this project was provided by the Natural Sciences and Engineering Research Council of Canada

(NSERC).

References

[1] A. Davison. Real-time simultaneous localisation and mapping with a single camera. In *Proceedings of the International Conference on Computer Vision*, volume 2, page 1403, 2003.

[2] M. Fischler and R. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. In *Communications of the ACM*, pages 381–395, 1981.

[3] M. Fukumoto and Y. Suenaga. FingeRing: A full-time wearable interface. In *Conference on Human Factors in Computing Systems*, pages 81–82, 1994.

[4] C. Harris and M. Stephens. A combined corner and edge detector. In *Fourth Alvey Vision Conference*, volume 15, pages 147–151, 1988.

[5] T. Hashimoto. A list-type reduced constraint generalization of the viterbi algorithm. In *IEEE Transactions on Information Theory*, volume 33, pages 866–876, 1987.

[6] B. Howard and S. Howard. Lightglove: Wrist-worn virtual typing and pointing. In *Fifth International Symposium on Wearable Computers*, page 172, October 2001.

[7] M. Isard and A. Blake. *Active Contours*. Springer, 1998.

[8] M. Isard and A. Blake. Condensation conditional density propogation for visual tracking. In *International Journal of Computer Vision*, volume 29, pages 5–28, 1998.

[9] M. Kalsch and M. Turk. Keyboards without keyboards: A survey of virtual keyboards. Technical report, University of California, Santa Barbara, 2002.

[10] A. Kilgarriff. Putting frequencies into the dictionary. In *International Journal of Lexicography*, 1996. Word lists available at `http://www.itri.brighton.ac.uk/~Adam.Kilgarriff/bnc-readme.html`.

[11] D. Lowe. Distinctive image features from scale-invariant keypoints. In *International Journal of Computer Vision*, volume 60, pages 91–110, 2004.

[12] I. MacKenzie and R. Soukoreff. Text entry for mobile computing: Models and methods, theory and practice. In *Human Computer Interaction*, volume 17, pages 147–198, 2002.

[13] J. Mntyjrvi, J. Kela, P. Korpip, and S. Kallio. Enabling fast and effortless customisation in accelerometer based gesture interaction. In *International Conference on Mobile and Ubiquitous Multimedia*, volume 83, pages 25–31, 2004.

[14] L. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. In *Proceedings of the IEEE*, volume 77, pages 257–286, 1989.

[15] G. Rigoll, A. Kosmala, and S. Elckeler. High performance real time gesture recognition using hidden markov models. In *Proceedings of the International Gesture Workshop on Gesture and Sign Language in Human Computer-Interaction*, volume 1371, pages 69–80, 1997.

[16] H. Roeber, J. Bacus, and C. Tomasi. Typing in thin air: the canesta projection keyboard - a new method of interaction with electronic devices. In *CHI 2003 Extended Abstracts on Human Factors in Computing Systems*, pages 712–713, 2003.

[17] M. Soriano, B. Martinkauppi, and S. Huovinen. Skin detection in video under changing illumination conditions. In *15th International Conference on Pattern Recognition*, page 1839, 2000.

[18] T. Starner and A. Pentland. Visual recognition of american sign language using hidden markov models. In *Proceedings of the International Workshop on Face and Gesture Recognition*, pages 189–194, 1995.

[19] D. Strelow and S. Singh. Online motion estimation from image and inertial measurements. In *Proceedings of the 11th International Conference on Advanced Robotics*, 2003.

[20] N. Ukita and M. Kidode. Wearable virtual tablet: Fingertip drawing on a portable plane-object using an active infrared camera. In *Proceedings of the 8th International Conference on Intelligent User Interfaces*, pages 169–176, 2004.

[21] A. Vardy, J. Robinson, and L.-T. Cheng. The wristcam as an input device. In *Proceedings of the Third International Symposium on Wearable Computers*, pages 199–202, 1999.

[22] D. Wigdor and R. Balakrishnan. A comparison of consecutive and concurrent input text entry techniques for mobile phones. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 81–88, 2004.