MOBILE-PHONE BASED GESTURE RECOGNITION

Barış Bahar [1], *Işıl Burcu Barla* [1], *Ögem Boymul* [1], *Çağlayan Dicle* [1], *Berna Erol* [2], *Murat Saraçlar* [1], *Tevfik Metin Sezgin* [3], *Miloš Železný* [4]

[1] Boğaziçi University, İstanbul, Turkey
[2] Ricoh California Research Center, CA, USA
[3] University of Cambridge, England
[4] University of West Bohemia in Pilsen, Czech Republic

ABSTRACT

Mobile phones are increasingly being used for applications beyond placing simple phone calls. However, the limited input capabilities of mobile phones make it difficult to interact with many applications. In this work, we present a mobile gesture recognizer where camera input of a mobile device is analyzed to determine the user's actions. Simple user gestures such as moving the camera from left to right and up and down are employed in two mobile applications: map navigation and a drawing program. In addition, more complex user gestures are converted into commands with an HMM based algorithm. These commands allow higher level interaction with the applications, such as moving to a specific location on the map or changing the drawing color. Index Terms mobile devices, computer vision, mobile phones, camera phones, motion estimation, camera-based interaction, HBMA, command recognition, gesture recognition

KEYWORDS

Mobile devices – Computer vision – Mobile phones – Camera phones – Motion estimation – Camera-based interaction – HBMA – Command recognition – Gesture recognition

1. INTRODUCTION

The simple keypad present in many mobile phones is adequate for placing voice calls but falls short when interacting with mobile applications such as web browsers, image and video browsers, and location services. Because mobile phones are handheld devices, user's gestures can be easily utilized as additional inputs (e.g. moving the phone to the right resulting in panning to the right on a map on the screen). User gestures can be recognized by sensors such as accelerometers and gyros on the device. However, currently mobile devices with such sensors is not common. Most mobile phones are equipped with video cameras which can be used to perform gesture recognition by analyzing the optical flow. Simple gestures such as moving the phone up and down can then be converted into commands that are interpreted by the application running on the phone.

In this paper, we present an implementation of mobile camera phone based gesture recognition and its applications. We recognize simple gestures such as leftright and up-down by analyzing optical flow and utilize these gestures for panning on map images, Figure 1, and in a drawing application. In addition, a sequence of gestures, such as up-down followed by left-right, are converted into high level commands that can be used for map navigation or for customizing the parameters of a paint applications (e.g., specify pen thickness).

Although other camera phone based gesture recognizers exists in the literature [1][2], our system is the first vision based gesture recognizer designed to run on a windows mobile phone and first to incorporate high-level command recognition from gestures. In the following sections, we first present an overview of our system, then give details of the camera phone based gesture recognizer in Section 2. Command recognizer is described in Section 4, and applications are presented in 5. Finally, conclusions and future work is presented in Section 6.

Figure 1: *Using gesture recognition users can easily navigate maps and images on mobile phones.*

2. SYSTEM OVERVIEW

Figure 2 given an overview of the camera phone based gesture recognizer, which is implemented in C++ using Windows Compact Framework and Pocket PC SDKs. A Windows mobile device, Palm Treo 700w [3] is used for deploying and running the gesture recognizer.

Direct Show filters are employed for communication and frame capture from the device camera. Frames are captured in YUV12 format. Only luminance (Y) component is used for analyzing the motion for real-time processing. Motion estimator uses hierarchical block matching algorithm to determine the global motion direction. We currently assume motion is limited to panning (left-right-up-down). Once the global motion vector $[\Delta x, \Delta y]$ is computed, it is directly used as input for applications, such as displacement on a map and moving the cursor in a paint application. In addition, it is sent to a **command recognizer** to determine the high level meaning of gestures.

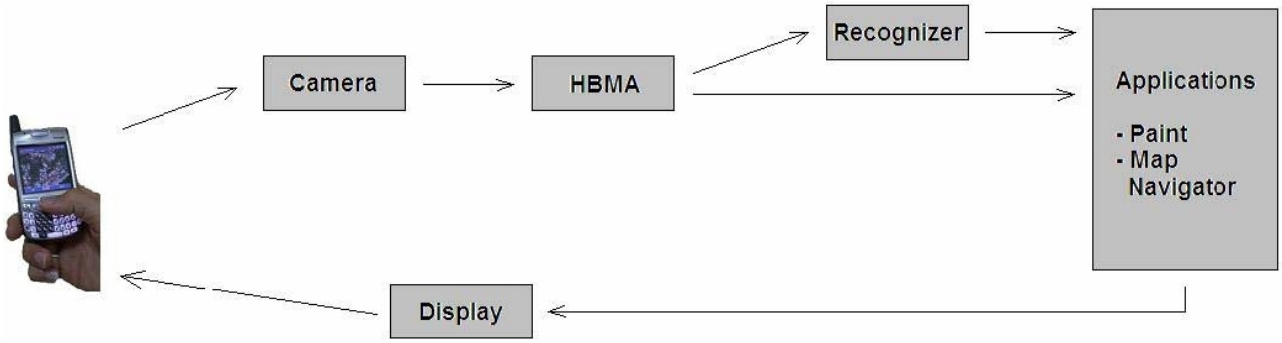


Figure 2: *An overview of the gesture recognition system and its interaction with the applications.*

3. MOTION RECOGNITION

In order to estimate the motion, the mobile phone's onboard camera is employed as the source of input. The direction and the magnitude of movement in consecutive video frames are used to infer the physical movement of the device by the user. Illustrative screen-shots captured while performing various gestures are presented in Figure 3. As seen in the figure, our algorithm is aimed to work with a variety of indoor and outdoor backgrounds.

Proper estimation of the observed motion from camera's onboard camera requires fast and accurate tracking. In order to fulfil the speed and accuracy criteria, hierarchical block matching algorithm (HBMA) is used to determine the motion direction and magnitude. The difference between two adjacent frames is used to estimate motion direction and magnitude, where direction values represent the x-y plane and magnitude is scaled according to dimension of the video frame.

Since colour image processing is computationally more expensive, we work with gray level images. For each consecutive image, we estimate motion matrices for the x and y directions. The means of these matrices give the direction and magnitude of the motion in x-y plane. In general, HBMA tries to align an anchor image to a target image while minimizing their difference, as illustrated in Figure 4. In order to achieve this, frames are compared at different scales. First, the frames are down sampled and smaller images are compared with little computation. Then, at each level, images are enlarged in order to make accurate estimates based on the ones made in the previous level. This low computational complexity algorithm results in accurate motion estimation.

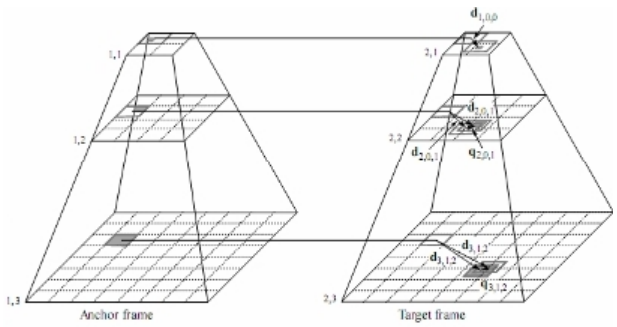


Figure 4: *3-D illustration of HBMA process.*

At every stage of the above mentioned computations, the anchor frame is divided into blocks according to the level, and then each block is compared with the corresponding ones in the target frame as shown in Figure 5. The correspondence is computed within a fixed search radius with the assumption that a target block exists in this specified area. Although this limits the maximum interframe device movement to fall below this specified radius, it does not cause any deficiency in practice. When matching two adjacent blocks, sum of Minimum Absolute Difference method is employed and block mean values are subtracted from this sum to reduce the lightning disturbance on matching. The target block is chosen to be the one that has the minimum difference. The distance and the position of the computed target block with respect to the anchor block give the magnitude and direction of the motion. By applying this technique for each block in the anchor frame, motion matrices of the images in x and y directions are computed. At the subsequent level, larger blocks are compared in a wider search area using a larger radius where the previous motion matrices are used as a starting point in order to reach a better estimation. Searching at multiple levels and reusing estimates from coarser levels results in an efficient and accurate matching. Motion vectors estimated using HBMA for consecutive levels is presented in Figure 6.

4. COMMAND RECOGNITION

Command gesture recognition system processes the motion vectors coming from the motion tracking part and decides on the most probable command. This command is then sent to and realized by the GUI. All possible commands, which are seven letters namely "b, r, g, t, e, w and d" in this project, are defined to the system. These commands and their corresponding meeting in our two applications, paint and map navigation, are presented in There is a database consisting of short movies, which are recordings with the mobile device, where each subject has imitated the letter by moving the device on the air. For building the database all movies are processed by the motion tracker, then vector quantization is applied for feature extraction and the system is trained using this data.

By using the command recognizer in the demo applications, the system will work real-time. In these applications, while one certain button is pressed, the mobile device captures the video. Then the motion tracker is activated. For each processed frame, the motion vectors are sent to the command recognizer. By releasing the button, the system finishes getting the observation. After that, the command recognizer decides on the command and sends this information to the running application GUI in order to be realized.

In the following parts, first some general information about the recognition theory will be given. Then, the implementation of this theory will be mentioned, and lastly some test results will be shown.

Figure 3: *Actual captured image sequences for various gestures.*

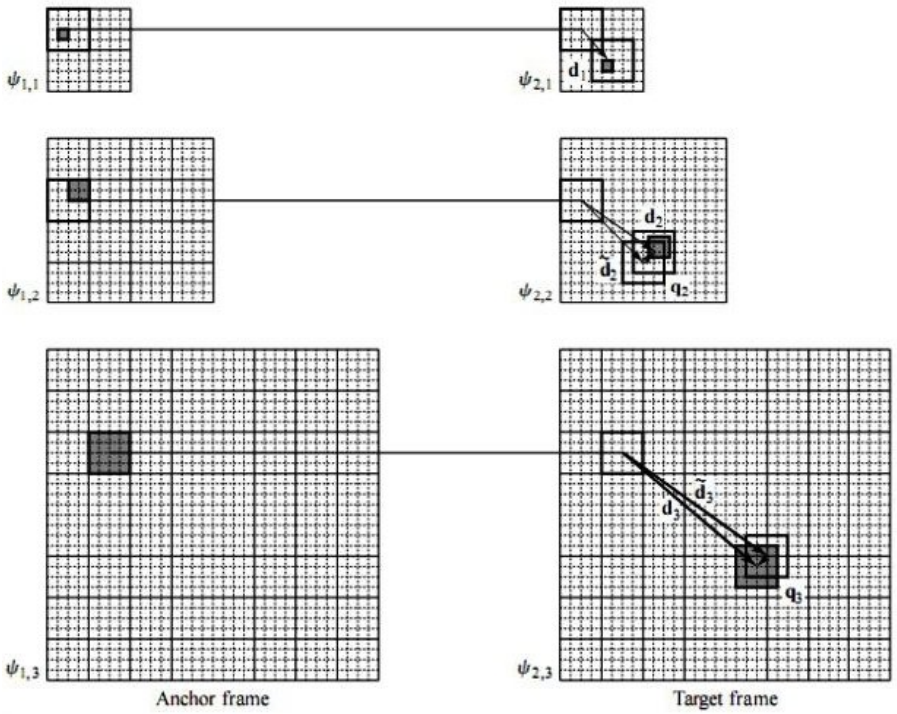


Figure 5: *In the first level of the pyramid, corresponding target frame block is estimated. In the subsequent levels, frame is up sampled, previous motion vectors are corrected.*

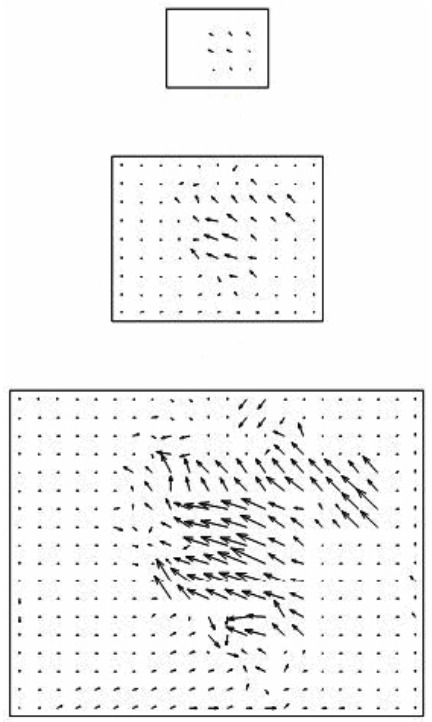


Figure 6: *Motion vectors corresponding to the estimates of HBMA for consecutive levels. The final output of HBMA is the image at the bottom from which the final motion is computed.*

Command s	Meaning	
	In Paint Application	In Map Navigation Application
b	Set the pen color to blue	Go to BUMED
r	Set the pen color to red	Go to Revir
g	Set the pen color to green	Go to Green Court
t	Make the pen thinner	Go to Teras Canteen
e	Enable eraser	Go to ETA Building
w	Make the pen thicker (wider)	Go to Washburn Hall
d	Set the pen color to black (default)	Go to Dorm

Table 1: *High level commands and their interpretations in two different applications.*

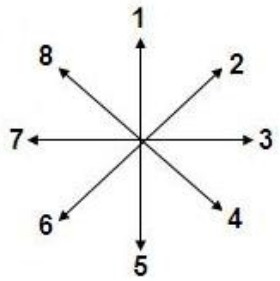


Figure 7: *The enumeration of the directions.*

4.1. Theory

All recognition systems are based on similar theory. This type of command recognition, which is described above briefly, can be used in many applications, where the user wants to control a mobile device without being limited with the keyboard.

In this project by the command recognition, Hidden Markov Models (HMMs) are used as the underlying technology. Generally, modeling and recognition process requires four steps [4], namely, feature extraction, determining the model parameters, computing the probabilities and determining the most probable state sequence.

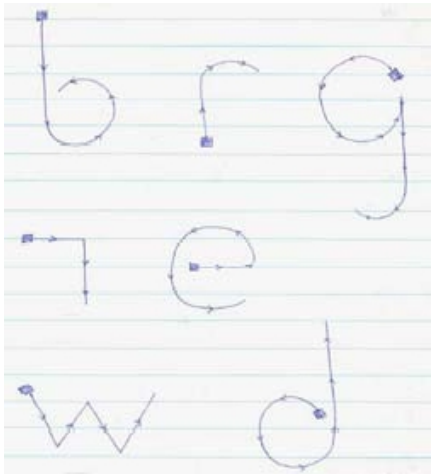
Let's examine the above mentioned steps in more detail. In our project, by feature analysis vector quantization is used. A vector quantizer maps k-dimensional vectors in the vector space R_k into a finite set of vectors. The motion vector pairs are grouped into three and these small groups are mapped in the observation space to some centroids and according to the database of all these observations a codebook is determined and defined to the system. The incoming data is then quantized according to this codebook, which contains the values of the 32 centroids, and each data group is mapped to the centroids.

The other three steps are the main problems in recognition. First of all the model parameters should be determined according to the nature of the commands, which will be recognized. The model parameters should be chosen such that the probability of the observation sequence given the model is locally maximized and then the iterative Baum-Welch algorithm or the Viterbi algorithm will be used by computing the reestimated model parameters. In this project the first approach was using the eight directions as the base of the model parameters. The advantage of this choice is that these model parameters are used by all of the letters, so the need on huge amount of database is decreased because by this method more samples for each model parameter could be observed. The definitions of the letters and the model parameters are shown in Figure 7. The model parameters were these eight directions, long 1, long 5 (these were needed according to the used letters) and the stabile situation. Long 1 was labeled as 1+, long 5 as 5+ and the stabile standing as 9. So, the letters coded with these labels are shown in Figure 8.

The test results with these model parameters were not satisfactory. One reason for that was that the samples were very different than each other. So, sharing the model parameters between the letters caused a large error rate. Hence, it was decided to define 7 different parameters for each command. By running the training algorithm, the system will learn the probability values corresponding to these parameters automatically.

There are two estimation algorithm candidates: Baum-Welch and Viterbi algorithms. The difference of these two algorithms is that the first one computes probabilities of all paths for all times, where in Viterbi only the most probable path survives to the next time instant. Hence, using Viterbi in this step decreases the needed memory and improves the speed of the system, which is very important in this project, since the applications will be run on a mobile device, which has a low CPU and limited memory capacity. Therefore Viterbi algorithm is chosen as the used algorithm in estimation part. Moreover, the number of the iterations should be chosen such that an optimum value for the model is determined but one should take care not to cause the model to completely depend on the training data. That will cause the model to give false results by a data outside the training set.

By computing the probabilities and building the file contain-



(a)

b	: 9 5+ 4 3 2 1 8 7 6 9
r	: 9 1+ 2 3 4 9
g	: 9 8 7 6 5 4 3 2 1 5+ 6 7 8 9
t	: 9 3 5+ 9
e	: 9 3 2 1 8 7 6 5 4 3 9
w	: 9 4 2 4 2 9
d	: 9 8 7 6 5 4 3 2 1+ 9

(b)

Figure 8: *Letter and model parameter definitions.*

ing the probability mass functions of the model parameters the training algorithm is used. As stated above the Viterbi algorithm is used by determining the most probable state sequence in each iteration. According to the state sequence and corresponding observation in the data, the probability values are updated and used in the next iteration.

When the system is trained according to the database, the final step is determining the most probable state sequence where any observation sequence and the determined model are given. By this decoding task the Viterbi algorithm is used. Given the state transitions, the first thing to do is building the trellis. Each cell of the Viterbi trellis, $v_t(j)$ represents the probability that the HMM is in state j after seeing the first t observations and passing through the most likely state sequence $q_1 \cdots q_{t-1}$, given the model λ. The value of each cell $v_t(j)$ is computed by recursively taking the most probable path that could lead to this cell. Formally, each cell expresses the following probability [5]:

$$v_t(j) = P(q_0, q_1 \cdots q_{t-1}, o_1, o_2 \cdots o_t, q_t) = j|\lambda) \quad (1)$$

Like other dynamic programming algorithms, Viterbi fills each cell recursively. Given that the probability of being in every state at time $t-1$ is already computed, the Viterbi probability is computed by taking the most probable of the extensions of the paths that lead to the current cell. For a given state q_j at time t, the value $v_t(j)$ is computed as [5]:

$$v_t(j) = \max_{1 \le i \le N-1} v_{t-1}(i) a_{ij} b_j(o_t) \quad (2)$$

In the above equation, a_{ij} gives the transition probability from state i to state j and $b_j(o_t)$ is the probability that at time t in state j observation ot occurs.

Explaining shortly the algorithm, it can be stated that for time $t = 1$, the metrics values in the trellis corresponding to the states to which a transition from the beginning state is allowed, are computed. After that according to the transitions the metrics are computed at each time slot. By these calculations only the path with the lowest cost can survive, others are eliminated, which is the key point in the Viterbi algorithm. Moreover, at each time slot the most probable path for each state from the previous time slot is memorized in a back pointer. At the end, the back pointer is read from end to beginning starting by the state for $t = T$ with highest probability. The recognition system used in this project is designed in the light of this theory using the programming languages C/C++.

4.2. Implementation

4.2.1. Search and Decoding

This part is done by using the Viterbi algorithm as explained above. The main function is called `Recognizer()` and is designed as an API, which should be called by the application. While a certain button is pressed, the movie is captured, processed and given as input to the recognizer after the features are extracted. Feature extraction is done by quantizing the incoming vectors according to the predefined codebook. This program needs four input files, namely, `pmf.txt`, `durations.txt`, `labels.txt` and `transitions.txt`. The first file contains the probabilities of the model parameters. The second one gives the probability of staying in the same state for each model parameter. In this project seven commands are used and 49 model parameters are defined. The third file contains the labels of these parameters and the last one determines the model, where each possible transition and its probability are defined.

4.2.2. Model parameter estimation

It is used for estimating the pmf values and the duration probabilities. As output it gives the minus likelihoods and the updated versions of `pmf.txt` and `durations.txt` at each iteration. The `Count()` function counts the occurrence rate of each model parameter for the complete command space. Using this information and the result of the Viterbi algorithm at each step, new probabilities are calculated. It takes as input the initial probabilities and the transition files corresponding to each separate command.

5. APPLICATIONS

Possible applications of the gesture recognition include: browsing maps, images, videos, web; playing games; handwriting... For demonstration of gesture recognition we implemented two applications: a map navigation application and a drawing application.

5.1. Map Navigation

A map navigation application is implemented utilizing gesture recognition as illustrated in Figure 9. In the map navigation application three control modes are assumed: panning the map, zooming the map and navigating to a particular place. To be able to process these three control modes, we designed main variables for the control: X and Y coordinates of the center of the map and zoom factor. Map control can be seen as watching bigger map (generally bitmap image) through a small window. X and Y coordinates control the position of the center of this window and zoom factor controls the magnification of the viewable part.

To control the map the algorithm is as follows:

- to navigate the map to certain position means set X and Y to that position (in image coordinates)

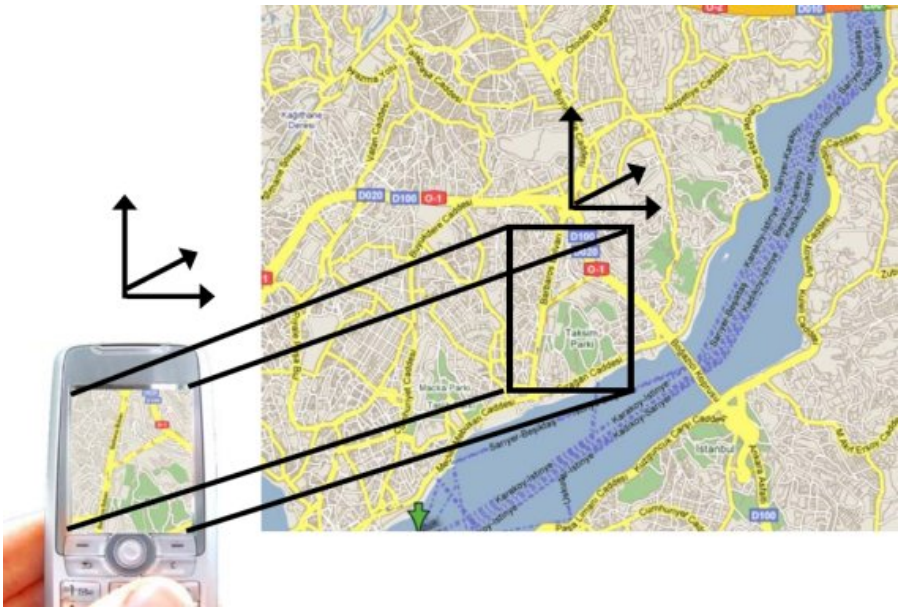


Figure 9: *Map application that interprets hand gestures for moving the viewing window on a larger map image in the memory.*

- to pan the map means modify X and/or Y according to desired pan motion
- to zoom the map means to increase or decrease the zoom ratio by amount specified by desired zoom change

It is supposed to control these three functionalities by device gestures. Navigating the map by letter gesture command (or speech), panning the map by the movement of the device in front of a stable view, and zooming the map by the movement in the direction towards the user or farther from the user.

To simulate the pan and zoom functionality before gesture input could be connected, mouse (stylus) input was implemented. The Palm application had two main modes, the pan and zoom mode. In the pan mode, the touch and movement of the stylus caused panning the map in the corresponding direction. In the zoom mode, the touch and movement of the stylus caused the zooming (movement upwards - zoom in, movement downwards - zoom out).

To demonstrate the navigate functionality there also exist several modes which zoom into the predefined places of the map. They are activated by drawing the first letter of the place in a predefined manner with the device in the air, or by choosing the name from the menu with the stylus. In the latter case the functionality could be debugged before connecting the real gesture input.

5.2. Paint Application

Paint application assumes simple drawing functions: Starting to draw a stroke, drawing a stroke by device movements, finishing to draw a stroke, changing the colour and thickness of the pen and erase. Then user may want to change position and start to draw another stroke. To accomplish this functionality, we must retain the list of stroke point coordinates and draw a line to the screen between points given by these coordinates. Dynamic array of coordinates is a good structure for this purpose, since the length of a stroke is not known in advance and is dependent on how long the user draws the stroke. To indicate any change of colour or thickness, again dynamic arrays are used.

Functionality that has to be implemented is thus

- start to draw
- draw
- finish drawing
- add color and width options (as eraser is imitated with a considerably thick white pen.)

Again, this will be controlled by gestures of the device, starting and finishing by pushing the button on the device, drawing by movements of the device. Another button is reserved for identifying the color and thickness options which are specified by drawing the first letter of the command in the air with the device in a predefined manner. For simulation, stylus events were processed and used as an alternative control.

5.2.1. *Mouse/stylus input and windows messaging*

To process mouse/stylus input in MS Windows environment, windows messaging system is used. In the main processing function of an application, we must process the windows messages. For mouse input, the main three messages that were used for simulation functionality are `WM_LBUTTONDOWN`, `WM_LBUTTONUP`, and `WM_MOUSEMOVE`. The first message is sent whenever user presses the left mouse button (or touches the touch-screen with the stylus), the second one is sent once the button is released (the stylus leaves the touch-screen), the third when the mouse coordinates have changed (in the case of stylus, we can of course follow the coordinates only when the stylus touches the screen).

All three messages are sent together with screen coordinates of the mouse/stylus. Thus, we have to process these coordinates to control the application. In the case of map panning functionality, on `WM_LBUTTONDOWN` we set `panning=TRUE`, and remember the coordinate, then on each `WM_MOUSEMOVE` we check if `panning==TRUE`, then get coordinates, count difference from last stored coordinates and control panning by this difference. Then we store new last coordinate. On `WM_LBUTTONUP` we finish panning by setting `panning=FALSE`. Similar approach is used for other functionalities. In zooming mode we also retain the last coordinates, but by the coordinate difference we control the zoom factor. In paint functionality, on `WM_LBUTTONDOWN` we start to draw (`drawing=TRUE`) and retain coordinates of the first point of a stroke, on `WM_MOUSEMO`

VE we check whether `drawing==TRUE`, then get coordinates and store them in memory. On `WM_LBUTTONUP` we finish the stroke and set `drawing=FALSE`.

For drawing to the device screen, another windows message is used: `WM_PAINT`. We have to process this message and paint the screen according to given control variables. It means that all the painting (showing the desired map portion or drawing a stroke) is done in this part of code whenever the Windows ask for it. In processing of the mouse messages (or the gesture input) we only modify control variables and tell the Windows that there is a change that needs repainting the window (calling `InvalidateRect(...);`)

Appendix 9.1 contains part of the code that illustrates the windows messaging.

6. CONCLUSIONS AND OUTLOOK

In this paper we presented our work on mobile phone based gesture recognition and its applications. The video input of a camera phone is analyzed with HBMA in order to determine simple user actions. These simple actions then analyzed further with command recognizer to generate high level commands.

Improvements to the motion recognizer is possible by incorporating tracking of image features, such as SIFT features, and prediction of motion with Kalman filtering. We presented two sample applications, map navigator and a drawing program. Many other applications, such as video games, web browsing, and handwriting recognition are possible using mobile gesture recognizer.

7. ACKNOWLEDGEMENTS

Authors thank Ricoh Innovations California Research Center for donating the mobile device for development.

This report, as well as the source code for the software developed during the project, is available online from the eNTERFACE'07 web site: http://www.enterface.net

8. REFERENCES

[1] J. Wang, S. Zhai, and J. Canny, "Camera Phone Based Motion Sensing: Interaction Techniques, Applications and Performance Study", in *ACM UIST*, 2006. 139

[2] A. Haro, K. Mori, V. Setlur, and T. Capin, "Mobile Camera-based Adaptive Viewing", in *4th International Conference on Mobile Ubiquitous Multimedia (MUM)*, 2005. 139

[3] *Palm Treo 700 w mobile device*. http://www.palm.com/us/products/smartphones/treo700w/. 139

[4] L. R. Rabiner, "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition", in *Proceedings of the IEEE*, vol. 77, feb 1989. 142

[5] D. Jurafsky and J. H. Martin, *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition*. Prentice-Hall, 2000. 143

9. APPENDIX: SOFTWARE NOTES

9.1. Applications

Below is a part of the code that illustrates the windows messaging:

```
LRESULT CALLBACK WndProc (
  HWND hWnd, UINT message,
  WPARAM wParam, LPARAM lParam)
{
 int wmId, wmEvent;
 PAINTSTRUCT ps;
 HDC hdc;
// RECT rc_inv;
 int xPos;
 int yPos;
 double half_x;
 double half_y;
(...)
 switch (message)
 {
(...)
  case WM_PAINT:
   hdc = BeginPaint(hWnd, &ps);
   DrawImage(hdc);
   EndPaint(hWnd, &ps);
  break;
(...)
  case WM_LBUTTONDOWN:
//WM_LBUTTONDOWN fwKeys = wParam;
   last_mouse_x = LOWORD(lParam);
   last_mouse_y = HIWORD(lParam);
   drawing = TRUE;
  break;
  case WM_LBUTTONUP:
   drawing = FALSE;
  break;
  case WM_MOUSEMOVE:
   if (drawing==TRUE)
   {
    xPos = LOWORD(lParam);
    yPos = HIWORD(lParam);
    shift_x = xPos - last_mouse_x;
    shift_y = yPos - last_mouse_y;
(...)
    InvalidateRect(hWnd, &rc_wnd, FALSE);
    UpdateWindow(hWnd);
   }
  break;
(...)
 }
 return 0;
}
```

9.2. Pseudo code of HBMA and EBMA

```
Input:
A = Anchor Image
T = Target Image
B = Block Size
R = Radius
N = Number of Pyramid Levels

Output:
Mv = Motion Vectors

Initialize motion vectors Mv to zero.
For n = N to 1
 Downsample anchor image A and target image T by 2^(n
      -1)
 Pad zero pixels to fit the block size
 Set the search radius r = R/2^(n-1)
 Set the block size b = B
 Apply EBMA to update the motion vectors Mv
 Upsample Mv by 2
End
```

```
Input:
A = Anchor Image
T = Target Image
B = Block Size
R = Radius
Mv = Initial Motion Vectors

Ouput:
Mv = Updated Motion Vectors

For each block in the anchor frame
 Take the next bxb anchor block Ba
 Shift to the previous estimate Mv
 For each block Bt of target frame in the radius r
  Find the block with minimum MAD error
```

```
    Update motion vectors Mv
  End
End
```

10. BIOGRAPHIES

Barış Bahar finished high school in Bursa in 2003. He is now an undergraduate student of the Faculty of Computer Engineering at Boğaziçi University, İstanbul.
Email: barisbahar86@yahoo.com

Işıl Burcu Barla graduated from the Faculty of Electrical and Electronic Engineering of Boğaziçi University, Turkey in 2007. She is now continuing her study at Technical University Munich, Germany. She has worked on speech and gesture recognition systems.
Email: burcubarla@gmail.com

Ögem Boymul finished high school in Adana in 2003. She is currently an undergraduate student of the Faculty of Electrical and Electronic Engineering at Boğaziçi University, İstanbul.
Email: ogem.boymul@boun.edu.tr

Çağlayan Dicle received his B.Sc. degree from Computer Science Department at Yeditepe University in 2004. He is now an M.S. student at System and Control Engineering at Boğaziçi University. His main research interests are computer vision and machine learning applications specifically on deformable object tracking and non-parametric classification.
Email: caglayan.dicle@boun.edu.tr

Miloš Železný was born in Plzen, Czech Republic, in 1971. He received his Ing. (=M.S.) and Ph.D. degrees in cybernetics from the University of West Bohemia, Plzen, Czech Republic in 1994 and in 2002 respectively. He is currently a lecturer at the University of West Bohemia, Plzen, Czech Republic. He has been delivering Digital Image Processing, Structural Pattern Recognition and Remote Sensing lectures since 1996 at the University of West Bohemia. He is working in projects on multi-modal speech interfaces (audio-visual speech, gestures, emotions, sign language). He publishes regularly and he is a reviewer of the INTERSPEECH conference series.
Email: zelezny@kky.zcu.cz

Berna Erol received her B.Sc. degree in Control and Computer Engineering at İstanbul Technical University and M.Sc. and PhD. Degrees in Electrical and Computer Engineering at the University of British Columbia, in 1998 and 2002, respectively. Since September 2001 she has been a senior research scientist at Ricoh California Research Center, USA. Dr. Erol has authored or co-authored more than 35 journal and conference papers, two book chapters, and more than 40 patent applications in the area of multimedia signal processing. Her main contributions to research and development consist of content-based video and image retrieval, image and video coding and representation, E-learning and E-meeting systems, text retrieval, new era multimedia systems, and applications, and multimedia processing for mobile devices. She has served in the program and the technical committees of leading ACM and IEEE conferences such as ACM Multimedia and IEEE ICME. She is an associated editor of the IEEE Signal Processing Magazine and a co-chair in the SPIE Electronic Imaging organizing committee. She had been an active participant in the video coding standardization activities such as ITU-T H.263 and MPEG-7.
Email: berna_erol@rii.ricoh.com

Murat Saraçlar received his B.S. degree from Bilkent University, Ankara, Turkey in 1994. He earned both his M.S.E. and Ph.D. degrees from Johns Hopkins University, Baltimore, MD, USA in 1997 and 2001 respectively. He worked on automatic speech recognition for multimedia analysis systems from 2000 to 2005 at the AT&T Labs Research. In 2005, he joined the Department of Electrical and Electronic Engineering at Boğaziçi University as an assistant professor. His main research interests include all aspects of speech recognition, its applications, as well as related fields such as speech and language processing, human-computer interaction and machine learning. He currently leads a TUBITAK funded project on Turkish Broadcast News Transcription and Retrieval. He authored and co-authored more than two dozen papers in refereed journals and conference proceedings. He has filed four patents, both internationally and in the US. He has served as a reviewer and program committee member for various speech and language processing conferences and all the major speech processing journals. He was the Area Chair for the Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP) in 2005. He is currently a member of IEEE and ISCA. He was recently elected to serve as member of the IEEE Signal Processing Society Speech and Language Technical Committee (2007-2009).
Email: murat.saraclar@boun.edu.tr

Tevfik Metin Sezgin graduated summa cum laude with Honors from Syracuse University in 1999. He received his MS in 2001 and his PhD in 2006, both from Massachusetts Institute of Technology. He is currently a Postdoctoral Research Associate in the Rainbow group at the University of Cambridge Computer Laboratory. His research interests include intelligent human-computer interfaces, multimodal sensor fusion, and HCI applications of machine learning.
Email: metin.sezgin@cl.cam.ac.uk