Activity Recognition Using Optical Sensors on Mobile Phones

Michael Wittke, Uwe Jänen, Aret Duraslan,
Emre Cakar, Monika Steinberg, and Jürgen Brehm
Leibniz Universität Hannover, Institut für Systems Engineering, SRA
{wittke,jaenen,duraslan,cakar,steinberg,brehm}@sra.uni-hannover.de

Abstract: Each mobile phone with a built-in CMOS sensor can inherently be seen as sophisticated optical sensor being able to analyze its environment in terms of visual events and its own mobility. Due to mass production their price decreases steadily, although their processing capacity increases. Mobile phones are usually attached to people, who are driven by mobility. We define activities arising from this mobility as *internal* activities in contrast to *external* activities, that are caused by visual events. Both activities can be recognized by measuring the sensor's optical flow. We present a method to identify internal activities based on optical flow measurements and probabilistic reasoning. We implement a lifelogging application, running on a Linux-based mobile phone, that can detect internal activities such as moving left-hand, right-hand or walking with a recognition rate of 80%. While standing still external activities are recognized using object detection.

1 Introduction

Recent advances in visual sensor technology, digital communications, and networking have enabled the deployment of a growing number of camera networks for varied surveillance applications [QKR+07, VR06]. Approaches from the realm of computer vision and artificial intelligence have been applied to these networks for modeling and recognition of human activities. For instance, smart environments for assisted living use video analysis algorithms to replace human operators to detect residents, which are at risk to fall, see [FS08]. Detection of human activities in crowds is used to prevent crowd disasters [HJAA07] and analyze the crowd's behavior with respect to salient events or dominant motions, see [CR07]. This is motivated by surveillance and security aspects. Nevertheless, the underlying hardware for these vision networks - being able to fulfill the task of human activity detection - is either supplied by custom hardware solutions to support high-processing capacities for image processing algorithms [VSC+06] or standard personal computers accessing high-resolution IP cameras [HWHMS08]. Therefore, these approaches are not suitable for embedded devices with constrained resources e.g. mobile devices.

In this paper, we introduce a methodology, which is adapted to the needs of constrained embedded devices, being able to perform human activity detection on mobile devices. We use open-source software (e.g. Intel's OpenCV image processing library) to perform

sophisticated image processing algorithms (e.g. optical flow calculations). This allows for internal activity recognition on these devices using feature points. These are points in the camera's field of view, which can be differentiated from their neighboring image points easily. Internal activities are activities, being performed by the mobile phone's owner (e.g. a person carrying the mobile phone whilst walking or turning left-hand and right-hand respectively). In addition to this, we use OpenCV's built-in face detection algorithm to detect people passing by. Thus, we make way for future interactive systems such as lifelogging using internal and external activity detection for visual logging.

Our focus is on lifelogging applications, where mobile phones are used to log the owner's life by analyzing the pictures captured by its built-in camera. The goal of lifelogging is to record and archive all information in one's life. This includes all text, all audio, all media activity, all biological data from sensors on one's body as well as all visual information. We focus on recording and analyzing all visual information that are captured by the mobile phone's built-in camera. The mobile phone's camera could be head-mounted or attached to the owner's clothes. In such a setup, our methodology can be used to detect and report internal activities (e.g. the owner is moving or standing still) for statistical applications (e.g., capturing and characterizing the owner's movement tracks by analyzing its speed, degree of backward movements, stop duration, turning rate, meander and maximum distance), and for external activities (e.g., talking to another person by detecting a face in the video stream).

Realizing this application requires a number of basic functionalities. First of all, the mobile device must *capture and analyze* a stream of images from the built-in camera to detect internal and external activities. In many cases, the stream has to be segmented into activities stemming from the owner's (i.e. *internal activities*) movement or from external visual events (i.e. *external activities*). Afterward, the recognized internal activity has to be *analyzed* in terms of its motion patterns, e.g. left-hand or right-hand movement. Analyzing the single movements over time enables us to interpret the activity in terms of *motion sequences*. Answering these questions requires services for *video segmentation* and *pattern recognition* on mobile devices. To recognize external activities computer vision algorithms from the realm of *object detection* are required. Our methodology provides portable implementations of these services as discussed in Sects. 3 and 4.

This paper is organized as follows: Sec. 2 discusses related work in the field of activity detection based on optical flow and presents a brief history of lifelogging. Sec. 3 explains our methodology to detect and analyze internal activities and Sec. 4 focuses the exemplary implementation on a Linux-based mobile phone. Three internal activities can be detected as well as external activities. Results in terms of accuracy, robustness and performance of our system are presented. Finally, we conclude with a summary and the future work in Sec. 5.

2 Related Work

This paper focuses on internal and external activity recognition using optical flow on off-the-shelf mobile devices. Therefore, we present (1) the state-of-the-art in activity recognition on existing frameworks for mobile devices and (2) approaches using optical flow for activity recognition. Afterward, we give a brief overview about lifelogging, since this is our application scenario. We emphasize that in contrast to our work, using optical flow for internal activity recognition is not considered in the following approaches. Furthermore, many of these approaches do not use off-the-shelf hardware and do not focus on the portability of their software.

In [BDM+06] a Smart Camera framework is presented, which is based upon COTS hardware components with several DSPs. A multi-camera object-tracking application is implemented to test and evaluate the framework. The tracking agent is based on a Lucas-Kanade-Tomasi feature tracker [Luc84]. In [HA06] a framework is introduced by Aghajan et al., which is based upon an ARM7 micro-controller. It is optimized to low-power consumption and real-time object detection. Wolf et al. [WOL03] present a multiple-camera architecture for real-time video analysis. The algorithms for object detection and activity recognition run on special video signal processors being able to perform all low-level video operation. In contrast to our work, their approaches are limited to special underlying hardware and cannot be ported to off-the-shelf mobile devices. Römer et al. [BKR07] propose an approach being the most similar one to ours. They implement a framework running on off-the-shelf Java mobile phones. However, as they use the Java API as programming environment, their framework suffers from performance issues and is limited to only perform *simple* computer vision algorithms. In contrast to this, our system is able to perform sophisticated computer vision algorithms such as optical flow calculations and algorithms such as probabilistic reasoning (e.g. artificial neural networks). In addition to optical sensors, accelerometers can be used for activity recognition, too. For instance, modern mobile devices such as the Nokia N95 [Web95], the Blackberry Storm [Webrm], Apple's iPhone [Webml] or the HTC Touch Diamond [Webnd] contain a built-in 3-axis accelerometer. The Wii game console [Webom] uses the same technology. Previous researchers used accelerometers on mobile phones for gesture recognition or interaction across distinct devices [HMS+01], [DB08]. Additionally, user activity is recognized from accelerometer data [RDML05], [BGC09]. In contrast to our work, internal but not external activities can be recognized using accelerometers.

Since we use the optical flow for activity recognition, we will briefly present the state-of-the-art in the field of optical flow used for activity recognition. In [ABF06] Fisher et al. present a technique for activity recognition in crowds based upon optical flow and Hidden Markov Models (HMM). Hidden Markov Models, cf. [RN03], are used as motion model to cope with the variable number of motion samples. Optical flow helps to observe and characterize the crowd behavior. Radke et al. [CR07] introduce an approach for detecting dominant motions in crowded scenes. This approach uses optical flow for motion calculations, too. In [FBYJ00] the use of parameterized motion models of optical flow is introduced. Affine models are used for image motion analysis. In contrast to the approaches described before, we use optical flow for *internal* activity recognition on mobile

devices instead of using it for dominant motion computations.

By 2007 several people have experimented with lifelogs. In the 1990s Steve Mann at MIT outfitted himself with a head-mounted camera and recorded his daily life on video tape, see [Man97]. Beginning in 2000 Gordon Bell at Microsoft Research has been documenting every aspect of his work life. His experiment is called MyLifeBits, see [Webll]. Bell wears a special camera around his neck, the SenseCam, which will notice a nearby person's body heat and take a photo of him, or snap a picture of a new place if it detected a change in light. From 2006 to 2008 Daniel Ellis, an associate professor of electrical engineering at Columbia University, has audio recorded - or audio lifelogged - his life, see [Webwe]. Alex Pentland at the MIT Media Lab built a "socioscope" which continuously tracked the location of 81 volunteers for 9 months, as well as tracked the pattern of their conversations, see [Webdu]. Nevertheless, the tools for lifelogging today are still scarce. E.g., the SenseCam used by Gordon Bell and produced by Microsoft is not available for the mass market yet.

3 Methodology

In this section we introduce our methodology for internal and external activity detection, which involves three phases and can be summarized as follows:

1. Video segmentation (for internal and external activities)
2. Determining motion sequences over time
3. Activity recognition

In Sec. 3.1 we explain how to distinguish between internal and external activities. An internal activity is detected, if the loss of feature points in the video data stream is above a special threshold. Sec. 3.2 contains the optical flow computation and clustering of the calculated optical flow angles. To analyze the internal activities, Artificial Neural Networks (ANN) are used, which are explained in Sec. 3.3.

3.1 Video Segmentation

We segment the video data stream into blocks of internal and external activities. Therefore, we first introduce the terms *internal activity* and *external activity*. We assume that the mobility of a mobile device (MD) is driven by an external object. For instance, we assume that the MD is a mobile Internet tablet (e.g. a *Nokia N 810* device) called $N810$ and the external object is a person called $Michael$. Further, we assume that our MD possesses a free field of view and is attached to a table. $\vec{v}_{N810}$ is the velocity vector of the $N810$. If $\vec{v}_{N810} = 0$, all activities being observed by the device are defined as **external activities**. For instance, this could be another person passing by. If $\vec{v}_{N810} \neq 0$, this is

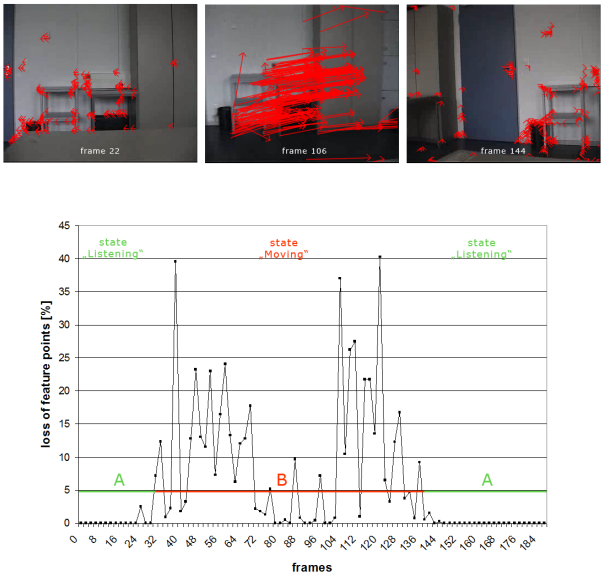

Figure 1: Movement detection and segmentation in distinct states: State *Listening* is marked with (A) and the state *Moving* is marked with (B)

due to an **internal activity**, e.g. *Michael* moving the device. Our goal is to identify these internal and external activities. As described in Sec. 1 this information can be used by sensing applications to compute motion tracks or profiles and by surveillance applications to analyze the environment with respect to salient events. We will use the image sensor's optical flow to compute the velocity vector. This supports us with information about the movement's absolute value and angle.

To distinguish between internal and external events, we introduce two states: *Moving* and *Listening*. If a MD is in the state *Moving*, it waits for internal events. We recognize this state by calculating the loss of feature points in the camera's field of view. The optical flow measurement supports data to infer the movement's velocity and change of the direction, see Fig. 1. The y-axis in Fig. 1 displays the loss of feature points from one frame to the subsequent one and the x-axis the frames captured. To calculate the optical flow vectors we use the Lucas-Kanade-Tomasi algorithm (parameters: 400 feature points, 3x3 window), see [Luc84]. If the loss is constantly below 5% (e.g. for more than 2 seconds: 20 frames assuming a frame rate of 10 fps), we assume that the camera's movement has stopped and change to the state *Listening*. This state is marked with A in Fig. 1. In this state the MD tries to detect external, salient events. Our recent work, see [WHHMS08], uses this state to detect salient events. The state *Moving* is marked with B in Fig. 1. If a large number of detected feature points cannot be retrieved in the subsequent frame, the MD's environment has changed due to a movement. In this state the loss of feature points is above a pre-defined threshold (e.g. $> 5\%$, marked by the bold line).

Thereby, we are able to segment the captured images into video sequences of internal (i.e.

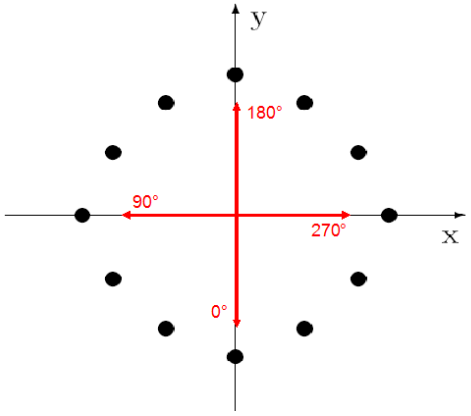

Figure 2: Internal activity recognition with the angle definitions for motion sequences: 90° means a movement to the left and 270° a movement to the right.

state *Moving*) and external (i.e. state *Listening*) activities. For instance, we use a mobile device to capture 100 images, the video sequence V, consisting of 45 frames of inactivity (i.e. the mobile phone lies on a table), followed by 10 frames of an internal activity (e.g. the mobile device is moved to the right by its owner) and ends with 45 frames of inactivity (e.g. lying on the table as before). The segmentation of this video sequence V results in three partitions: $V_{seg} = (v_{e_1}, v_{i_1}, v_{e_2})$. Segments, comprising external activities such as v_{e_1} and v_{e_2}, are analyzed by computer vision algorithms with respect to objects. Segments, that describe internal activities such as v_{i_1}, are analyzed with respect to their optical flow values. As mentioned before, the internal activity v_{i_1} comprises 10 frames (i.e. 10 images): $v_{i_1} = (I_1, I_2, \ldots, I_{10})$. Each image is analyzed in terms of its feature points to compute the optical flow vectors to the successive one. E.g., if each image is analyzed with respect to its most significant 400 feature points, $I_{1..10}$ are characterized by a 400-dimensional optical flow vector each, e.g. $I_1 = (fp_1, \ldots, fp_{400})$.

3.2 Determining Motion Sequences over Time

External activities are analyzed with respect to objects that are contained within the image. Therefore, it is not necessary to determine motion sequences for them. However, if we want to analyze internal activities with respect to particular motions, we have to form motion sequences over time.

Based on the optical flow vectors of an internal activity v_i consisting of j frames/images ($v_i = I_{1..j}$ with 400 feature points each), the type of the motion is determined considering the positions of corresponding feature points in subsequent frames. According to these positions, we calculate the angle as depicted in Fig. 2, i.e. 90° describes a left-hand motion and 270° a right-hand motion respectively. Thereby, $I_{1..j}$ are encoded by a 400-dimensional angle vector each, e.g. $I_{1,2} = (\angle fp_1, \ldots, \angle fp_{400})$. Afterward, a mean angle

is computed for each image and stored in the j-1 dimensional motion vector m: $m = (\angle I_{1,2}, \ldots, \angle I_{j-1,j})$. Hence, we are able to determine motion sequences over time.

3.3 Activity Recognition

Standard computer vision algorithms for object detection for external activity recognition are used. These algorithms usually use Haar-like features, since a recognition process can be much more efficient if it is based on the detection of features that encode some information about the class to be detected, see [VJ01]. This is the case of Haar-like features that encode the existence of oriented contrasts between regions within the image. A set of these features can be used to encode the contrasts exhibited by a human face and their spacial relationships. Haar-like features are computed similar to the coefficients in Haar wavelet transforms. E.g., to detect people passing by, we used *OpenCV's*, see [Webcv], face detection, which is based upon Haar-like features.

For internal activity recognition ANN are used, see [Lip88]. ANNs apply the principle of function approximation by example, meaning that they learn a function by looking at examples of this function. To be able to train the ANN, we define a function with a set of input and output variables supported by examples of how this function works. To recognize activities we define a function with one input variable (our motion vector) and an output vector consisting of floating-point values, whereas each value represents the probability that the motion vector represents a pre-defined activity.

Nevertheless, before using this function - encoded by the ANN - we have to approximate it by training. We aim at recognizing activities. Each activity is determined by a motion sequence, which is defined by the sequence of angles received from the optical flow calculation. Therefore, we can determine a pre-defined activity by examining the frequency of angles occurring in the motion sequence. To train our ANN with an internal activity, we capture a number of training videos containing the activity. Afterward, we determine the motion sequences and examine the frequency of angles occurring in these sequences. This means that the temporal ordering of the angles is not considered, however, complexity is reduced. To reduce the computational effort, we cluster the angles into different groups. For instance, to build 36 groups each group has to cover a range of 10 degrees, whereas the first group ranges from 0 to 10 degrees. Thus, the input vector's size is defined by the number of groups and the output vector's size is defined by the number of activities.

The process of this method can be explained using a short example: We train the ANN with two different internal activities. Each of them requires four videos to determine the characteristic motion sequences and frequencies of angles according to the positions of feature points in subsequent frames. Therefore we capture four videos of each activity. Then their motion sequences are determined and the characteristic frequencies examined. These are clustered in 36 groups, i.e. each group covers a range of 10 degrees. The resulting ANN has an input neuron for each of the 36 angle groups, and a floating-point output neuron for each of the internal activities. After the training the ANN can be used to analyze motion sequences in terms of the pre-defined activities. An example will be given

in the following section.

4 Experimental Results

To study the feasibility of our approach, we implemented a lifelogging application on a mobile phone. Three internal activities can be recognized: (1) turning left-hand, (2) turning right-hand, (3) and moving. In case of external activities, human faces can be detected within the video stream. If an internal or external activity is recognized by our system, an event is sent to all applications being registered to this event.

Initially, we investigated the accuracy of our implementation, i.e. how accurate our approach can distinguish between the internal and external activities. To measure the robustness of our methodology to recognize internal activities, we conducted experiments to analyze our optical flow based method using distinct backgrounds, since textures of the background may influence the recognition of feature points. To answer the questions related to performance issues, we analyzed the CPU and memory utilization of our implementation on a real mobile phone.

4.1 Experimental Setup

We used the mobile Internet tablet *Nokia N810* as a platform for the implementation of our system [Webes]. This is a mobile device running Linux with built-in WLAN. The device is equipped with a TI OMAP 2420 processor with 400 MHz being ARMv6-compatible, which has 128 MByte of DDR RAM and 256 MByte of hard disk available. Pictures are obtained via a USB webcam (QCIF resolution: 176x144). As image processing library we choose *Intel's OpenCV (Open Source Computer Vision)*, see [Webcv], since this is a widely-used open source project with a high reputation in the computer vision community. To implement ANNs we use *FANN (Fast Artificial Neural Network Library)*, see [Webnn]. *FANN* is a free open source Neural Network library, which implements multilayer ANNs in C with support for both fully connected and sparsely connected networks. For all cross-compilation tasks we use an ARM GNU/Linux target cross-compiler from CodeSourcery [Webom] (e.g. OpenCV without X-Server, libz, libpng and libtiff). Events are sent via *libcurl*, see [Webor].

For activity recognition of the three internal activities, we capture four training videos each with a frame rate of 10 frames per second. The training videos have a mean length of approx. 50 frames and are captured in front of a TFT display to support enough background textures for the optical flow. We set up an ANN consisting of three layers (one input, one hidden and one output). The input layer has 36 neurons (the number of groups as explained in Sec. 3.3), the output layer has three neurons (one for each activity, i.e. left-hand, right-hand and walking) and the hidden layer has 18 neurons. The number of layers and neurons in the hidden layer has been selected heuristically, as there is no trivial way to determine these values. Fig. 3 shows the average frequencies of arising angles in

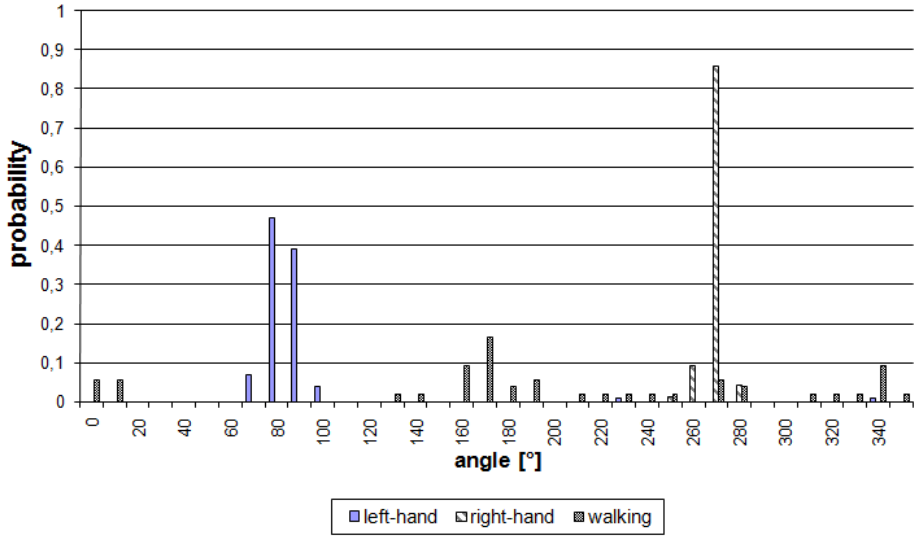

Figure 3: Average frequencies of the angles of the three internal activities

Accuracy of internal activity recognition		
	success	failure
left-hand	70%	30%
right-hand	70%	30%
walking	100%	0%

Table 1: Accuracy of recognition of left-hand, right-hand and walking motion (based on 10 test videos each)

the three internal activities. Obviously, the right-hand and left-hand motion possess clusters of angles around $270°$ and $90°$ respectively. Angles of the walking motion are almost equally distributed.

4.2 Accuracy

For external activity recognition, we use standard computer vision algorithms based on Haar-like features. These algorithms usually achieve high recognition rates. In [VJ01] experiments demonstrate that a frontal face classifier constructed from 200 features yields a detection rate of 95% with a false positive rate of 1 in 14084. Since the accuracy of this class of algorithms is broadly investigated, we focus on investigating the accuracy for internal activity recognition.

To study the accuracy of internal activity recognition we captured 10 videos for each in-

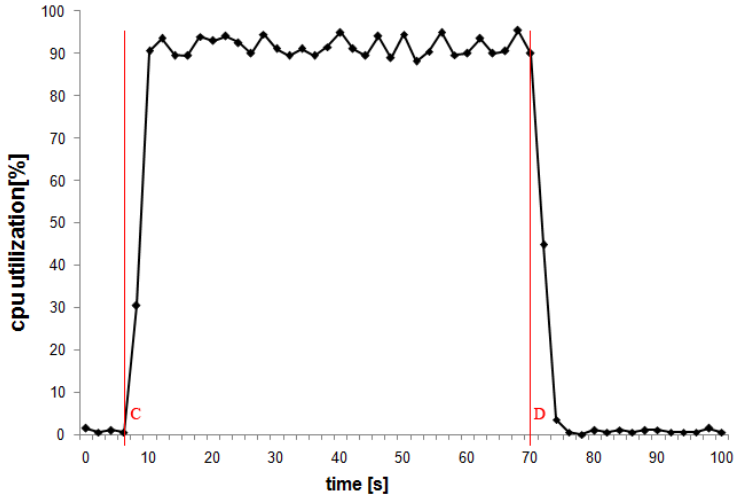

Figure 4: CPU utilization: From start to position C and from position D the device is in idle mode, between it captures pictures continuously performing the optical flow computations and the ANN-based activity recognition

ternal activity and tested the recognition rate. The results are presented in Tab. 1. The left-hand and right-hand motion are recognized with a mean accuracy of 70%. Reasons for failing recognitions for both activities are variations in the dominant clusters of the training videos. Since both activities have strong clusters around one angle (e.g. 90 degrees for the left-hand motion and 270 degrees for the right-hand motion), its accuracy is reduced. Due to these restrictions, the walking motion is recognized with an accuracy of 100%. Since it is almost equally distributed over a large number of clusters, it is very robust toward variations in the cluster groups. Nevertheless, since our activities possess different characteristics toward their dominating clusters, we achieve good results in terms of recognition.

4.3 Performance

We measured the CPU and memory utilization on the mobile phone. As depicted in Fig. 4 there are two (C and D) state changes. Up to the position C in Fig. 4 the mobile phone is in idle mode (mean CPU utilization: below 2%). Afterward, the activity recognition is started. That means, pictures are continuously captured from the webcam and analyzed with respect of the loss of feature points to distinguish between internal and external activities. The CPU utilization increases up to 95% during execution of our system due to the image processing tasks. For instance, the Lucas-Kanada-Tomasi algorithm for analyzing a picture with respect to its most significant 400 feature points, has a mean run time of

Robustness of activity recognition		
	success	failure
wall - low frequency background texture	50%	50%
PC monitor - high frequency background texture	80%	20%

Table 2: Robustness of recognition in terms of different background textures

$1004.91ms$. In contrast to the image processing algorithms, the ANN needs $13.1ms$ to recognize our internal activities. OpenCV's face detect algorithm has a mean run time of $608.01ms$. Since application scenarios for lifelogging are dominated by image processing tasks (i.e. capturing pictures and analyzing them in terms of feature points or objects) constantly, this will always account for a basic CPU utilization of approx. 70 to 80%. Although the CPU utilization is very high, future ARM based mobile phones will be equipped with more powerful processors (e.g. 1 GHz XScale ARM processor, see [Webor]). Therefore, the CPU utilization will be reduced on future systems. Thus, the CPU utilization of our method carries less weight as an indicator of the system performance than depicted in Fig. 4.

The memory utilization is about 17.6% of the device's memory. This is approximately 22.5 MByte. As the Nokia device is equipped with 128 MByte of RAM and future systems will possess more RAM, 17.6% of memory utilization is a reasonable value.

4.4 Robustness

To study the robustness of the internal activity recognition with respect to distinct backgrounds we captured 10 videos for each background and tested the recognition rate for the left-hand motion. We choose the left-hand motion, since it possesses the same accuracy like the right-hand motion and a lower accuracy than the walking motion, however, having a higher relevance for our application scenario. We suppose that for a lifelogging application it is more relevant to know how the direction of the movement has changed than to know that it is still in process. The results are presented in Tab. 2.

The two backgrounds are different in terms of their textures. The background *wall* contains a high number of low frequency textures. We captured videos of a wall being situated in four meters distance. The color of the wall is white without any textures or outlines serving as feature points for the optical flow algorithm. That is why the recognition rate decreases to 50%. Using a background with a large number of high frequencies (i.e. with outlines like edges and lines such as a PC monitor) the robustness increases to 80%. We captured videos of a PC monitor displaying the content of a website possessing enough outlines from a distance of 30 centimeters. The recognition rate increases up to 80% and is higher than the accuracy achieved in the experiment investigating the recognition rate in terms of different movements, see Sec. 4.2. This is due to the fact that the background, which

we used in Sec. 4.2, contains less high frequencies. This means that the robustness and recognition rate of our system depend highly on the characterizations of the particular background requiring it to possess an enough number of outlines and textures.

5 Conclusion and Future Work

We have presented a methodology to identify external and internal activities on mobile phones. Our method is based on optical flow measurements and probabilistic reasoning. We discussed our implementation consisting of a real mobile phone running a Linux-based operating system. Experimental results are presented, in which a lifelogging application is implemented. This application can recognize three internal activities and external activities by using standard computer vision algorithms for object detection. We achieve a recognition rate of 80% for simple internal activities and can detect faces using Haar-like features. In this context, the robustness of our methodology for internal activity recognition and the performance on an off-the-shelf mobile phones is investigated. Further work is needed to completely implement and evaluate our methodology for internal activity recognition. Our future work will add more complex internal activities such as letters of the PALM alphabet. This makes way for complex pattern recognition such as gestures to interact with the system. More experiments are required to investigate our methodology under a number of constraints to show its feasibility for real-world application. The recognition rate gives promising results but the performance needs to be improved, since the processing capacity is still a limited resource on mobile phones. Further work is needed to enhance the system performance and reduce the time required to correctly detect internal and external activities and to determine the particular activity.

References

[ABF06] Ernesto Andrade, Scott Blunsden, and Robert Fisher. Performance Analysis of Event Detection Models in Crowded Scenes. In *Proc. Workshop on "Towards Robust Visual Surveillance Techniques and Systems" at Visual Information Engineering 2006*, pages 427–432, Sep 2006.

[BDM+06] Michael Bramberger, Andreas Doblander, Arnold Maier, Bernhard Rinner, and Helmut Schwabach. Distributed Embedded Smart Cameras for Surveillance Applications. *Computer*, 39(2):68, 2006.

[BGC09] Tomas Brezmes, Juan-Luis Gorricho, and Josep Cotrina. Activity Recognition from Accelerometer Data on a Mobile Phone. In *IWANN (2)*, pages 796–799, 2009.

[BKR07] Philipp Bolliger, Moritz Köhler, and Kay Römer. Facet: Towards a Smart Camera Network of Mobile Phones. In *Proceedings of Autonomics 2007 (ACM First International Conference on Autonomic Computing and Communication Systems)*, Rome, Italy, October 2007.

[CR07] A. Cheriyadat and R.J. Radke. Automatically Determining Dominant Motions in Crowded Scenes by Clustering Partial Feature Trajectories. *Proceedings of the First ACM/IEEE International Conference on Distributed Smart Cameras (ICDSC)*, 2007.

[DB08] Raimund Dachselt and Robert Buchholz. Throw and Tilt â Seamless Interaction across Devices Using Mobile Phone Gestures. In *2nd Workshop on Mobile and Embedded Interactive Systems (MEIS'08)*, September 2008.

[FBYJ00] David J. Fleet, Michael J. Black, Yaser Yacoob, and Allan D. Jepson. Design and Use of Linear Models for Image Motion Analysis. *International Journal of Computer Vision*, 36(3):171–193, 2000.

[FS08] Sven Fleck and Wolfgang Strasser. Smart Camera Based Monitoring System and Its Application to Assisted Living. *Proceedings of the IEEE*, 96(10):1698–1714, Oct. 2008.

[HA06] Stephan Hengstler and Hamid Aghajan. A Smart Camera Mote Architecture for Distributed Intelligent Surveillance. In *In ACM SenSys Workshop on Distributed Smart Cameras (DSC*, USA, 2006. ACM.

[HJAA07] Dirk Helbing, Anders Johansson, and Habib Z. Al-Abideen. The Dynamics of Crowd Disasters: An Empirical Study. Feb 2007.

[HMS+01] Lars Erik Holmquist, Friedemann Mattern, Bernt Schiele, Petteri Alahuhta, Michael Beigl, and Hans-W. Gellersen. Smart-Its Friends: A Technique for Users to Easily Establish Connections between Smart Artefacts. pages 116–122. Springer-Verlag, 2001.

[HWHMS08] M. Hoffmann, M. Wittke, J. Hähner, and C. Müller-Schloer. Spatial Partitioning in Self-Organizing Smart Camera Systems. *Selected Topics in Signal Processing, IEEE Journal of*, 2(4):480–492, Aug. 2008.

[Lip88] Richard P. Lippmann. *An introduction to computing with neural nets*. IEEE Computer Society Press, Los Alamitos, CA, USA, 1988.

[Luc84] Bruce D. Lucas. *Generalized Image Matching by the Method of Differences*. PhD thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, July 1984.

[Man97] Steve Mann. Smart Clothing: The Wearable Computer and WearCam. *Personal Technologies*, March 1997. Volume 1, Issue 1.

[QKR+07] Markus Quaritsch, Markus Kreuzthaler, Bernhard Rinner, Horst Bischof, and Bernhard Strobl. Autonomous Multi-Camera Tracking on Embedded Smart Cameras. *EURASIP Journal on Embedded Systems*, 2007.

[RDML05] Nishkam Ravi, Nikhil Dandekar, Prreetham Mysore, and Michael L. Littman. Activity Recognition from Accelerometer Data. *American Association for Artificial Intelligence*, 2005.

[RN03] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice-Hall, Englewood Cliffs, NJ, 2nd edition, 2003.

[VJ01] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. volume 1, pages I–511–I–518 vol.1, 2001.

[VR06] Sergio Velastin and Paolo Remagnino. *Intelligent Distributed Video Surveillance Systems (Professional Applications of Computing) (Professional Applications of Computing)*. Institution of Engineering and Technology, 2006.

[VSC+06] Senem Velipasalar, Jason Schlessman, Cheng-Yao Chen, Wayne Wolf, and Jaswinder Singh. SCCS: A Scalable Clustered Camera System for Multiple Object Tracking Communicating Via Message Passing Interface. pages 277–280, July 2006.

[Webll] Web. Gordon Bell's Homepage. http://research.microsoft.com/en-us/um/people/gbell/.

[Webwe] Web. Dann Elli's Homepage. http://www.ee.columbia.edu/ dpwe/.

[Webcv] Web. OpenCV. http://www.intel.com/technology/computing/opencv/.

[Webnn] Web. Fast Artificial Neural Network. http://leenissen.dk/fann/.

[Webml] Web. Apple iPhone. http://www.apple.com/iphone/features/accelerometer.html.

[Webom] Web. WII Game Console. http://www.wii.com/.

[Webnd] Web. HTC Touch Diamond. http://www.htc.com/www/product/touchdiamond/.

[Webes] Web. Nokia N810 Technical Specification. http://www.nseries.com/nseries/.

[Web95] Web. NOKIA N95. http://www.forum.nokia.com/devices/N95.

[Webor] Web. Marvell unleashes 1GhZ XScale ARM processor. http://tamspalm.tamoggemon.com/2009/01/15/marvell-unleashes-1ghz-xscale-arm-processor/.

[Webdu] Web. MIT Media Lab. http://www.media.mit.edu/.

[Webrm] Web. Blackberry Storm. http://www.blackberry.com/blackberrystorm/.

[Webom] Web. CodeSourcery. http://www.codesourcery.com.

[WHHMS08] Michael Wittke, Martin Hoffmann, Jörg Hahner, and Christian Müller-Schloer. MIDSCA: Towards a Smart Camera Architecture of Mobile Internet Devices. In *ICDSC08*, pages 1–10, 2008.

[WOL03] Wayne Wolf, Burak Ozer, and Tiehan Lv. Architectures for distributed smart cameras. In *ICME '03: Proceedings of the 2003 International Conference on Multimedia and Expo*, pages 5–8, Washington, DC, USA, 2003. IEEE Computer Society.