Wearable Eye Tracking for Multisensor Physical Activity Recognition

Peter Hevesi[*], Jamie A. Ward[†], Orkhan Amiraslanov[‡], Gerald Pirkl[§] and Paul Lukowicz[¶]
[*†‡¶]German Research Center for Artificial Intelligence, Kaiserslautern, Germany
[†]University College London
[¶]University of Kaiserslautern
[§]Ostbayerische Technische Hochschule Amberg-Weiden
[*] peter.hevesi@dfki.de, [†] jamie@jamieward.net,
[‡] orkhan.amiraslanov@dfki.de, [§] g.pirkl@oth-aw.de,
[¶] paul.lukowicz@dfki.de

***Abstract*—**

This paper explores the use of wearable eye-tracking to detect physical activities and location information during assembly and construction tasks involving small groups of up to four people. Large physical activities, like carrying heavy items and walking, are analysed alongside more precise, hand-tool activities, like using a drill, or a screwdriver. In a first analysis, gaze-invariant features from the eye-tracker are classified (using Naive Bayes) alongside features obtained from wrist-worn accelerometers and microphones. An evaluation is presented using data from an 8-person dataset containing over 600 physical activity events, performed under real-world (noisy) conditions. Despite the challenges of working with complex, and sometimes unreliable, data we show that event-based precision and recall of 0.66 and 0.81 respectively can be achieved by combining all three sensing modalities (using experiment-independent training, and temporal smoothing). In a further analysis, we apply state-of-the-art computer vision methods like object recognition, scene recognition, and face detection, to generate features from the eye-trackers' egocentric videos. Activity recognition trained on the output of an object recognition model (e.g., VGG16 trained on ImageNet) could predict Precise activities with an (overall average) f-measure of 0.45. Location of participants was similarly obtained using visual scene recognition, with average precision and recall of 0.58 and 0.56.

Keywords–Wearable sensors; Machine learning; Activity recognition; Feature extraction; Computer vision.

I. Introduction

Self organizing teamwork – when a task is not assigned to individuals but to a group of people – is a common phenomena in the professional sector. Examples of this occur in many situations, for example on construction sites, in rescue teams, or in hospitals. The current work is a development of our earlier contribution on eyetracking for activity recognition [1], and forms part of a wider project that aims to study teamwork in industrial settings, and how the interactions and collaborations of individuals can characterize such teams.

One possible approach for understanding a group's processes and structures is to recognize the activity of each person and detect signs of interactions between the team members. This low level information can be fed into higher level recognition modules e.g., for semantic analysis [2] or collaboration recognition. Obtaining accurate information about an individuals' activities in a real world environment is a challenging task, one which most likely requires a variety of different wearable sensors and sensing modalities. These sensors have to be unobtrusive, accurate and scalable (both in price and availability). As mobile eyetrackers become cheaper and more widespread, they might be used for purposes other than simply gaze tracking. In this work we aim to expand their usage towards detecting physical activities.

Eyetracking provides a useful insight into a person's attention. Attention in turn can provide useful indications of that person's activity. Previous work has explored the idea of using mobile eyetrackers for activity recognition [3]. To date the vast majority of this research has concentrated on activities directly related to visual attention and cognition (reading, watching TV, etc.) [4][5]. Our current investigation uses eye movement information, alongside camera images, to recognize physical activities, specifically activities related to an assembly and construction task. In doing so, we don't intend to develop highly specialized methods, optimized to perform well only in the selected scenario, but rather showcase the possibilities of the approach by using a collection of "off the shelf" methods.

In our earlier work we described how wearable eye tracking devices can be used to provide useful informations about physical activities that are outside the mainstream eye-tracking application domain [1]. Here we add to this in two important ways. Firstly, we deepen our evaluation methodology by using a wider range of features, a range of different window durations, and add an additional, post-classification, temporal smoothing filter. Secondly, we apply state-of-the-art computer vision methods to the task of classifying activity and location using only egocentric camera images. This adds an extra layer of context to the activity classification, and points towards a promising sensor-fusion based approach for future work in this area.

In Section II, we provide an overview of recent articles and methods in our research field. In Section III, we describe the performed experiments, sensor setup and the datasets including the ground truth labels used during the evaluation. Section IV contains the overview and in-depth description of our methodology. The results for activity recognition using eye data are presented in Section V. In Section VI, we discuss possible use-cases of the egocentric videos recorded by the eyetracker devices. In Section VII, we highlight open questions for future work and conclude our findings.

II. Related work

As sensing technology continues to shrink in size and cost, an increasing number of researchers are turning their focus towards sensor-based activity recognition. One prominent research direction is using sensors embedded in the environment. A popular application of this approach is towards assisted living and smart homes. Methods for detecting activities of daily living are presented for example in [6], [7] and [8].

In workplace scenarios, like the one examined in the current paper, the sometimes dynamic and mobile nature of a task makes wearable sensing the best option for capturing it. Many studies deploy distributed body-worn or mobile inertial sensors to recognize a wide-range of physical activities (see [9] for an overview).

Sound is another common sensing modality. In [10], Lu et al. introduce a mobile-phone based system for classifying ambient sound, voices and music. Previous works use multiple streams of audio to recognize social situations [11][12], or to infer collocation and social network information [13].

Combined sound and movement data obtained from the smartphones of groups was used to analyses pedestrian congestion at busy thoroughfares, making use of changes in people's step-intervals and ambient audio [14]. Wrist-worn microphones and accelerometers were first used together to detect hand-tool activities in a wood workshop scenario [15]. More recently, these sensors were used to recognize physical collocation and collaboration of co-workers performing a group task [16].

A. Eye-based activity recognition

Eye tracking is a widely used technique in human computer interaction (HCI). It can be used for example in assistive technologies for people with limited motor skills (e.g., by Barea et al. [17]). Typically, researchers are interested in the object of a user's gaze – what it is that the user is looking at – e.g., in areas such as marketing research [18], or user interface design [19], [20], [21]. A different approach is to analyze the patterns created by eye movement in various situations. To detect reading activities while walking, Bulling et al. used patterns of eye fixation and saccadic movement recorded from changes in the eye's electrical activity (electrooculography, or EOG) [3]. Later, this work was extended to detect activities such as writing, reading, watching a video, etc. [22]. An advantage of a pattern-based approach is that no calibration is needed with a worldview video.

Changes to the blink rate can indicate different mental loads corresponding to different types of tasks. In [23], the authors describe methods for eyelid position and blink detection. Platforms like Google Glass include the ability to record blink rate, which when combined with head movement can be an effective method for recognizing activities [5].

Vidal et al. introduced a calibration-free, gaze interaction method based on tracking the smooth pursuit movements that occur when the eye follows a moving target [24]. And in [25] a commercial, wearable EOG system, the Jiins Meme, was used as a novel gestural input device based on a similar approach.

Shiga et al. proposed a system based on eyetracker and first person videos to recognize daily activities [26]. This work is the closest to our research. However, the authors focus on activities directly related to gaze (e.g., reading, video watching), whereas our work is more concerned with using eye-patterns to recognize physical activities that do not necessarily involve direct gaze (e.g., using a screwdriver).

B. Computer Vision on First Person Videos

Advances in computer vision, especially in the area of deep neural networks, have opened up a wide-range of possibilities. Frameworks like Keras [27], or Detectron [28], make it relatively easy for researchers to apply state-of-the-art vision algorithms quickly. Deep learning architectures optimized for recognizing images (e.g., [29] or [30]) are often available with pre-trained weights, and can be fine-tuning using only a few samples. Amos et al. presented a general purpose face detection framework [31] with an accuracy of ca. 93%. These methods usually support GPU-acceleration and can be used for real time video processing too.

Based on a day-long, first-person video footage of trips to an amusement park, Fathi et al. analyzed social structures of groups and the interactions within them [32]. Ryoo et al. use first-person perspective cameras to try and understanding how different people interact with an observer, and to be able to differentiate between friendly and hostile actions [33]. And in [34], the authors argue that to detect activities of daily living, the objects seen and interacted with could play a major role. Motivated by these results, we investigate the usefulness of egocentric videos to detect physical activities and to recognize locations in our dataset.

III. Experiment

To evaluate our different sensors and algorithms we designed a construction-work-inspired, data-collection experiment, that was initially described in [16]. In the experiment, groups of up to four people work together on a demanding physical task (build a large TV wall), but crucially are free to choose how they go about this - and whether or not to collaborate with one another. Some of the sub-tasks can be performed in parallel, while others must be done in sequence. This led to a complex, highly variable, and noisy, dataset that closely-mirrors real-world construction scenarios.

A. Scenario

Four participants collaborate to build a 2.5 meter high TV wall consisting of 8 large LCD screens, 3 base panels, 18 screen spacers, and more than 50 screws. The parts are stored in containers at a storage area, which is separated by a ca. 25 meter long hallway from the assembly area.

The building phase included the following main steps: 1.) Unload screens (each screen weights 8 kg.) and other TV parts from the containers, 2.) Carry items to the assembly area, 3.) Assemble and place base items, 4.) Lift screens onto the wall, 5.) Fix screens on the wall by tightening the screws. After the build phase and a short break the participants perform the process in reverse: 6.) removing the screws, 7.) taking down the screens and other parts carefully, 8.) carrying back to the storage area, 9.) put them back into the containers.

Generally, the participants had the freedom to organize and execute the tasks as they thought it's best. Since the TV screens were quite heavy, they divided themselves almost every time into groups of two to carry and lift the screens. After the first components were delivered to the assembly area, they had the option to start with mounting and fixing the screws parallel to the transportation task. The overall task takes usually from 40 minutes up to 1 hour.

B. Wearable sensors

While performing the tasks, the participants were equipped with a with a mobile eyetracker, a sound recording device with two separate microphones and three inertial measurement units. This setup is shown in Figure 1.

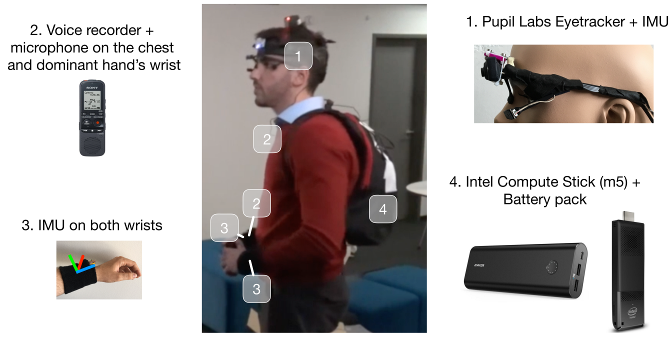
2. Voice recorder + microphone on the chest and dominant hand's wrist
3. IMU on both wrists
1
2
2
3
3
4
1. Pupil Labs Eyetracker + IMU
4. Intel Compute Stick (m5) + Battery pack

Figure 1. Recording setup for each participant includes an eyetracker connected to a small recording computer. Additional sensors: IMU on both arms and head, microphone on the wrist and at the chest.

a) Inertial measurement unit (IMU): For tracking movements of the participants, they wore IMUs on both wrists and one on the head. The IMU devices record 3-axis acceleration, gyroscope, and magnetic field as well as 3D orientation at approximately 40 Hz.

b) Sound recorder: Each participant wore two microphones: one on the dominant hand's wrist and a second one attached on the chest. The microphones were connected to a voice recorder capable of recording stereo sound and were saved as the two channels of an audio file.

c) Mobile eyetracker: In our experiment, we used Pupil Labs eyetracker devices (as described in [35]) with the 100 degrees field of view lenses for the world camera to cover more of the world's scene. The eyetracker setup additionally consists of an Intel Compute Stick with an m5 1.6 GHz processor (running Ubuntu 16.10) as a recording device for each person. They were powered by a portable 20100 mAh battery. This setup is able for mobile recording for ca. 1 hour 30 minutes, before the battery have to be recharged. The recording itself was done using Pupil Capture (v0.82) software. We implemented scripts to remotely control and monitor the recordings. The overall cost of this eyetracker setup is around 1600 Euros, which is significantly lower then many other commercially available mobile eyetracker solutions.

During our experiments, we observed a lot of issues with the eyetracker calibration. The main reason for these was the displacement of the eyetracker's frame on the head due to sudden movements and sweating as the participants performed heavy physical work. In an attempt to overcome these issues, we integrated the eyetracker's components into ski-goggles, which usually sit really tight and fixed on the face. However, this concept showed improved calibration stability, they were also more bulky and therefore not suitable for real world experiments. Also for reproducibility reasons, we later decided to use the unmodified version.

C. Datasets and Labels

For our testing and evaluation, we used two recordings of the above described experiment performed by two different groups of participants. In each dataset, four stationary cameras recorded the scene additionally to the above mentioned wearable sensors (eyetracker, IMU data, sound recordings). Two of the cameras were recording the assembly area, one the storage area and one the hallway. The main purpose of these cameras is to help the annotation process. An important step was the synchronization of the signal sources and videos. This was done in a post processing step with the help of predefined synchronization gestures at the beginning of the experiment.

The data for each participant was annotated into 6 different activity events (adjust, screwdriver, drill, carry, screen placement, walk) and no activity (NA). These activities were then sub-divided for the 2-class (Large vs. Precise activities), and 1-class (Precise) analyses (details below). To evaluate how well the eyetracker can detect location and co-location of the subjects, we additionally labeled each participants positions throughout the datasets.

The degree of freedom to organize and perform the experiment resulted often in unexpected event flows with many short interruptions and activity changes. This proved to be a challenge to label, making low-level event annotation nearly impossible. On the other hand this makes the data realistic. By keeping this in mind, we consider each ground truth label as a rough description of what a participant is mainly doing within a given time interval (from a few seconds up to a minute). Short interruptions (e.g., person taking additional screw from the desk or interacting with other participants) are not represented in this ground truth. In total, we labeled 606 activity events with an overall length of ca. 260 minutes. An additional 'No Activity (NA)' class was annotated to cover all the instances where a person is not doing any of the defined activities.

a) Six class problem: The detailed label set includes six classes (alongside the NA class):

1) Adjust: during these activities the subject is interacting (placing, taking or adjusting) with screws without any tool.
2) Screwdriver: subject tightens or loosens screws using screwdriver.
3) Drill: events when a participant tightens or loosens screws using a power drill with screwdriver attachment.
4) Carry: the times when one or two participants carry the heavy TV screens to or from the assembly area.
5) Screen placement: segments where screens are taken out of or placed back into the container or put on or taken off the TV wall.
6) Walk: person moves between assembly area and storage area (without carrying heavy objects).

b) Two class problem: With this label set, we want to investigate the performance of the system for separating actions involving large, location changing motions and subtle action requiring subtle movements. Thus, we defined two classes (in addition to NA):

1) Large: all events containing the above categories carry, screen placement and walk.
2) Precise: a combined set of the above defined adjust, screwdriver and drill categories.

c) One class problem: The third set looks only at the single class of Precise activities, as defined above, against a catch-all class comprised of everything else.

With the current dataset, it wouldn't be viable to annotate accurate 2D or 3D position of the participants. Instead, we created contextually meaningful location labels such as the person is at the storage area, on the hallway, on the corridor or at the assembly area. The layout of the experiment area including the location labels is showed in figure 2. Based on these location labels, we derived the co-location of participants. Co-location is defined here as a pair of participants is in the same region (has the same location label).

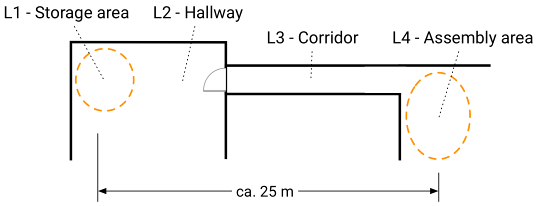

Figure 2. Floor plan of the experiment area with the highlighted location labels.

d) Location labels:

IV. Activity Recognition with Eye Tracking Data

This section describes our workflow and methodology used during the study. All code for processing the data is implemented in Python with Numpy.

A. Overview

The exploration space to find good parameter combinations is large. For example, we wanted to test different window sizes for the feature extraction on different label sets. We generate over 3000 feature samples per person per signal source for each dataset. To calculate one sample, the method selects raw values inside a window and performs the feature computations. Depending on the window size and the sampling rate of the data source, this window can contain more than thousand raw samples. So, recalculating a feature set can take longer time periods even on modern processors (usually between 5-20 minutes depending on the signal source). To be able to perform tests more efficiently and quicker, we divided our process into two main modules: 1) feature generation, 2) activity recognition. After a general description of these modules, we describe each important step in more detail.

1) Feature extraction module: As a first step, this module can find and parse the different data sources such as eyetracking data, measurements of the inertial measurement units (IMU) or the sound recordings. It can also load the files containing the ground truth information.

The feature extraction part takes the appropriate input signals and calculates the selected features (as described below) in the sliding windows. As a result, we obtain a table for each person where each row contains the sample's time, the current activity label and the feature values.

An overview of the module's data flow is illustrated on Figure 3. We run this process for each parameter set (e.g., window size 5s, 6 classes, eye data only) and store the resulting feature matrices. Because of the high amount of data and number of combinations, this process usually has a longer runtime (for all combination it can take up to 7 hours). However, it needs only to be run once as long we don't introduce new features.

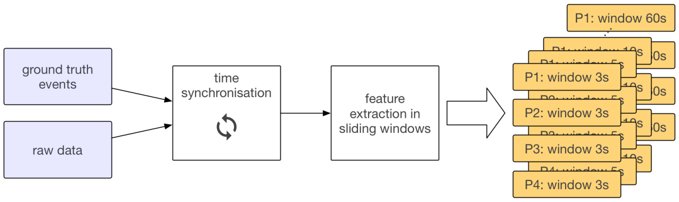

Figure 3. We first extract and synchronise the raw data, and then implement a parallelized process to evaluate different parameters (such as window size) for the selected features.

2) Activity recognition module: This module is designed to quickly evaluate feature sets and different parameter combinations. Figure 4 shows the flowchart of the module.

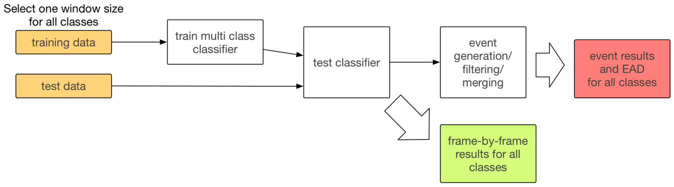
Select one window size for all classes
training data
test data
train multi class classifier
test classifier
event generation/ filtering/ merging
event results and EAD for all classes
frame-by-frame results for all classes

Figure 4. For each selected parameter set (window size, train-test split method, label set), we apply this evaluation process. As a result, we obtain frame-by-frame as well as event based performance metrics.

At first, it loads the previously generated feature tables for a selected window size. These are then split up into training and test sets based on the selected evaluation strategy. The training set is then used to train a classifier. After that, we test the classifier's frame-based performance by comparing the for the test set's features predicted labels with their ground truth labels.

Additionally, the frame by frame prediction results are then transformed into events. The event generation method can apply temporal smoothing for merging events close to each other and removing events that are too short. In the last step, we calculate event level metrics. The module can produce visualizations for both frame-based and event-based results.

B. Synchronization

During the recording, due to technical reasons, each device recorded the measurement values with timestamps based on their system time. The inertial measurement units and the eyetrackers use POSIX timestamps, however there can be offsets between the current system times. Drift during the experiment is negligible, since all of the systems use quartz oscillators internally. In case of the stationary cameras and voice recorders, each frame's or sample's time is registered as the elapsed time from the beginning of the recording.

In the synchronization step, the goal is to bring the different signal sources to a common time base. For that purpose, as previously mentioned in III-C, each participant performed a synchronization gesture at the beginning of the experiment. The gesture was defined as jumping and clapping with the hands in front of the recording camera. Patterns of this gesture can be easily pinpointed on all of the raw signals. For the eyetrackers (first person video) as well as for the stationary cameras, we looked up the corresponding frames' timestamps.

Based on the known times of the gestures for each source, we could calculate the offset with the following simple equation:

$$\Delta t^{A-B}_{offset} = t^{A}_{sync} - t^{B}_{sync} \tag{1}$$

where A and B are references to signal sources (e.g., P4's eyetracker and main video). t^{A}_{sync} and t^{B}_{sync} refer to the same synchronization event's time on respectively A or B's system time. With the known offset, we can convert times from one signal's time to the others using the following formula:

$$t^{A}_{i} = t^{B}_{i} + \Delta t^{A-B}_{offset} \tag{2}$$

By applying this method, we converted each source channel's timebase to the main video's time for easier referencing.

C. Feature Extraction

All of the below described features are calculated with the sliding window method over the evaluation time. This means that, we take the raw signal(s) between the start and end of the current window and calculate the source specific features. The window is then shifted forward until it reaches the end of the experiment. In this analysis, we use the center of the window to look up the activity label of the participants and also to combine feature rows from different sources (IMU, eye data or sound).

1) Eye features (eye): In each window, we calculated 14 statistical features on different eye related events and properties. These are:

- mean (μ) and standard deviation (σ) of the fixation length during the window
- μ and σ of the gap's duration between fixations
- μ, σ and zero-crossings (ZC) of the eye's spheric coordinates (θ and ϕ)
- μ and σ of the pupil size
- μ and ZC of eye position estimator's confidence value.

ZC, a simple measure of dominant signal frequency, is calculated by counting the zero-crossings on each window after subtracting μ.

Based on our initial findings, we expect the length of the fixations and the gap between two fixations to be a good indicator for increased attention. Information about the 3D orientation of the eyeball can help us to distinguish between different type of eye activities. The pupil size, could help to distinguish dark and bright environments. Accommodation (change of viewing distance) can also cause changes of pupil size. Changes in the confidence of the eye position estimator correlate usually (if no displacement, loss of calibration or other problems) with the blink rate of the person. Blink rate can vary very much from person to person, but can also be related to specific type of activities (e.g., when focusing, blink rate drops).

We calculated eye features using window sizes of 3, 5, 10, 15, 30, 45 and 60 seconds.

2) Accelerometer features (ACC): Only the accelerometer signals (ACC) from each person's right-wrist IMU are used in this study. These 3-axis accelerometer signals (x,y,z) are combined to give a single orientation-invariant reading by the formula:

$$a = \sqrt{x^2 + y^2 + z^2} \tag{3}$$

For each of these readings five standard features are calculated across a 1 second rolling window, these are: mean (μ), standard deviation (σ), short-term energy (E), zero-crossing rate (ZC) and skewness (γ).

3) Sound features (snd): Sound signals from each participant's dominant wrist, s_r (all were right-handed), and head, s_h, are downsampled from the recording rate of $44.1kHz$ to $8kHz$ (16 bit). In the first step, two features are extracted for each of these across a rolling window of 40 milliseconds: short-term energy, E, and zero-crossing rate, ZC. These features were chosen because of their widespread use in low-cost speech and audio analysis [36]. In a second step, we re-sampled these features using a window of 1 second for smoothing.

4) Fusion of features (acc+snd and all): One of our goals is to evaluate how well a combination of different data sources performs on the task of detecting physical activities compared to the baseline performance. For this purpose, we use the approach of combining feature matrices and training a new classifier with the merged matrix.

When merging feature matrices, we look up the corresponding row of the new matrix for each row of the original matrix by selecting the one with the closest sample time. If no feature row can be found with a smaller sample time difference than a threshold of one second, we skip this row (new feature cells have then non-valid values).

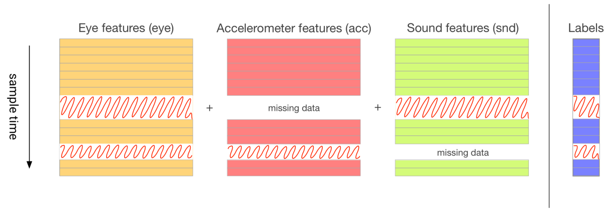

Figure 5. Merging of feature matrices. To eliminate the issue of missing data blocks in different sources, we simply remove all rows from the feature matrix and label vector, which consist non-valid values.

One of the challenges here is that in many cases data is not available or corrupt for a time period on one of the sources. Figure 5 shows the selected approach to handle missing data. When any row of the merged feature matrix includes any corrupt or non-existing values, we simply remove this line from the matrix. This also means that these rows are not considered in the evaluation.

In this work, we used combinations of:

1) ACC and sound features for comparing eye features with other modalities (acc+snd)
2) eye, ACC and sound features to see, if the eyetracker data can improve the overall recognition rate (all)

D. Classifier training

For all evaluations, we used a Naive Bayes classifier with a One vs. Rest training strategy. This means training one classifier for each class, where positive samples are taken from training instances of the current class, and negative samples from all other classes. This strategy typically achieved 1-2 percent better results than the One vs. One strategy that we used in our earlier work [1]. We also experimented with different classifier methods, but found the Naive Bayes to be sufficient for the purposes of the current work. All methods for training and testing are implemented on Python using the scikit-learn toolkit [37].

E. Event generation

In the prediction phase, the classifier produces one label per sample (row of the feature matrix). We refer to these as frame-based or frame-by-frame predictions. When the same label is returned in a sequence, we can combine them into an event by saving the time of the first occurrence as the event's start and the last as the event's end. If there is only one single frame in the sequence detecting a class, we use the sample distance as the event's length to avoid zero length events, which is 1 second in our case.

We implement a temporal smoothing on event level to reduce the number of false detections. In a first step, if an event follows a previous event of the same class within a specified time, they will be merged into one event. We used a threshold of 10 seconds for the event merging. Finally, events with a duration smaller than a threshold of 1.5 seconds are removed. Figure 6 shows the event generation and filtering concept.

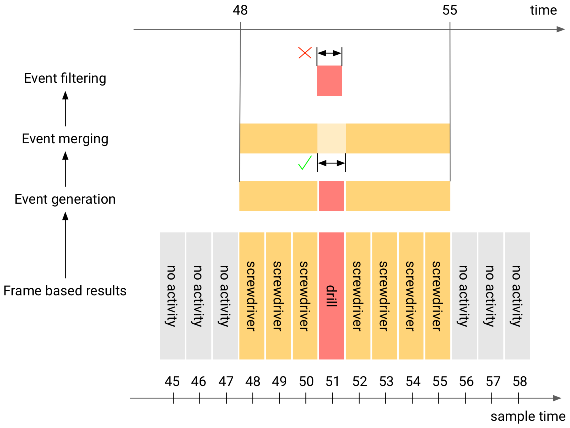

Figure 6. Event generation process. Frame-based results are transformed into events, events following each other in a close distance can be merged and very short events can be removed.

F. Evaluation

1) Train-test split: By using dataset A and B, we defined four evaluation runs, two for dataset dependent and two for dataset independent. Dependent 1 strategy uses only data from experiment A for training and testing. Dependent 2 does the same on dataset B. For independent 1 strategy, we use dataset A for training and dataset B for testing. In case of independent 2, the classifier is trained on data from B and tested on data from A.

a) Experiment dependent evaluation: For experiment dependent evaluation, data from one dataset (dataset A or B) were split into training and test sets by the leave one person out method. For example, data of P1, P2 and P3 participants were used to train the classifier and data of P4 to test it. The method is therefore person independent. The method was applied for all four combinations of an experiment independently and the results were averaged. With this approach, the test data is always unseen for the classifier and at the same time we get an estimation about the generalization error.

b) Experiment independent evaluation: A second approach for the evaluation is to use one dataset for training and the other one for testing. Both for training and test, we include data from all participants of the corresponding dataset. These tests usually indicate how well the system can generalize the results and handle later datasets without any additional training effort.

2) Performance Metrics:

a) Frame based evaluation: Each item of the classifier's prediction output is compared to the corresponding ground truth labels for the frame based evaluation. For that, we use the standard convention, where each predicted label is considered as true positive (TP) if its equal to the samples ground truth label or as false positive (FP) otherwise. A ground truth label is a false negative (FN) if the predicted label for the same sample is different.

We calculate then precision and recall values as defined by the standard definition. That is for precision:

$$P = \frac{TP}{TP + FP} \tag{4}$$

and for recall:

$$R = \frac{TP}{TP + FN} \tag{5}$$

Accuracy score is calculated by:

$$A = \frac{TP + TN}{TP + TN + FP + FN} \tag{6}$$

with TN for the true negatives.

b) Event based evaluation: In the event based evaluation, we compare detection events with the ground truth. A detected event is considered as a true positive (TP_{det}) if it has an overlap with a ground truth event of the same activity (for the same participant) or as a false positive (FP_{det}) otherwise. Similarly, ground truth events are labeled as true positives (TP_{gt}) if they are detected at least once otherwise as false negatives (FN_{gt}).

We calculate the event-based precision (P) and recall (R) analogous to the standard frame-based definitions, with the addition of the events' durations as weight factor:

$$P_e = \frac{TP_{det} \cdot \Delta T_{det}^{TP}}{TP_{det} \cdot \Delta T_{det}^{TP} + FP_{det} \cdot \Delta T_{det}^{FP}} \tag{7}$$

and,

$$R_e = \frac{TP_{gt} \cdot \Delta T_{gt}^{TP}}{TP_{gt} \cdot \Delta T_{gt}^{TP} + FN_{gt} \cdot \Delta T_{gt}^{FN}} \tag{8}$$

Additionally, we calculate scores specially designed for event based evaluation [38]. The approach is designed to acknowledge the fact that event-based recognition is not as clear-cut as true/false, positive/negative-based evaluations imply. Real-world recognition can be fragmented in time, or multiple events can be merged together. The evaluation methodology used works by labeling both ground truth events and prediction events into separate error categories.

Error categories that are applied to the ground truth include:

- Deletions (D): when a ground truth event is not detected at all, practically the same as a false negative
- Fragmented (F): a ground truth event is detected by multiple (at least 2) true positive fragments
- Fragmented and Merged (FM): the ground truth event is detected in multiple (at least two) fragments, and at least one of the detections also overlaps with another ground truth event
- Merged (M): a detection event covers more than one (true positive) ground truth events

Error categories that are applied to the detected events include:

- Insertions (I'): equivalent of false positives, when a detection event has no overlap with a ground truth event at all
- Merging (M'): detection events that overlap with more than one ground truth events
- Fragmenting and Merging (FM'): a detection event, which overlaps with at least two ground truth events, where one of them at least is also covered by another detection event
- Fragmenting (F'): when an event is detected during a ground truth event that is also covered by other detection events

Finally, one category can be applied to both ground truth and detected event output:

- Correct (C): a ground truth event corresponds to one true positive detection event

In our tests, we used the open source Python implementation of this scoring mechanism and associated metrics [39].

V. Results

This section contains the summary of our most important results regarding the usage of eye movement features and it's combination with acceleration and sound features for physical activity recognition.

A. Window size for eye features

In a first step, we evaluated the influence of the eye features' window size on the classification results. Features for each person in both datasets were generated using eight different window sizes ranging from 3 seconds to 60 seconds. We performed tests with these features on all three label sets (one-, two-, six-class problem) and using all four evaluation strategies.

Results for the window size evaluations for the different label sets are presented on Figure 7. The y axis on these figures represents the accuracy value as defined in Equation 6. The six class problem's results are displayed on Figure 7-A. In this case, the overall accuracy seems to be very similar across different evaluation strategies (dependent, independent), especially at small window sizes. With an increasing window size, two of the strategies perform worse. For the poor performance in independent 1 and dependent 2, probably the missing data points in dataset B are responsible.

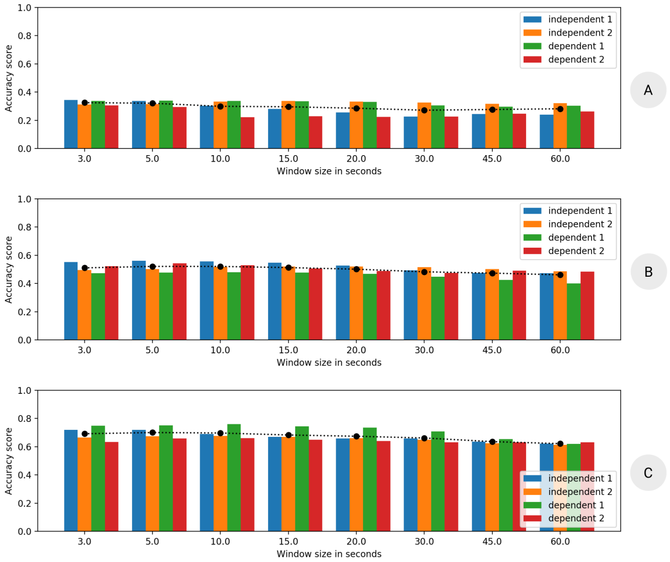

Figure 7. Activity recognition accuracy scores (frame-based) for eye features calculated with different window sizes on the six class problem (A), on the two class problem (B) and on the one class problem (C)

For the two class problem (Figure 7-B) and the one class problem (Figure 7-C), the different strategies perform very similarly to each other. As going from the six class problem towards the one class problem, as expected, the overall accuracy of the classifier improves significantly and is always better than random guessing. The average accuracy across strategies for the ten second window is for the six class problem 0.37, 0.56 for the two class and 0.71 for the one class problem.

In the further analysis, we use the ten second window size to extract eye features, because it performed fairly good on each label set. Larger window sizes tend to be more sensitive about the distribution of the training and test data as we could observe on Figure 7-A.

B. Results for six classes

The confusion matrix on Figure 8-A shows the frame based results using the eye features extracted in ten seconds windows. The classifier was evaluated using the independent 1 strategy. However, single classes not recognized very well (but still better than random guessing), they point out two major groups in the data. This two groups are defined as Large and Precise class in the two class problem.

Figure 8 show the results using the same strategy, but with the combination of all features into a single feature matrix. The results are still not perfect, but the improvement is visible and the two categories can still be observed.

The results on Figure 7 show us that different evaluation strategies perform similarly. However, it varies from label set to label set, which one is better, the other results are typically in a range of ± 5 percent. That's why and for a better overview, we decided to provide in the following the performance values only for the independent 2 strategy.

Table I shows all frame-based results of the six class problem. For each feature set (eye: using only eye based

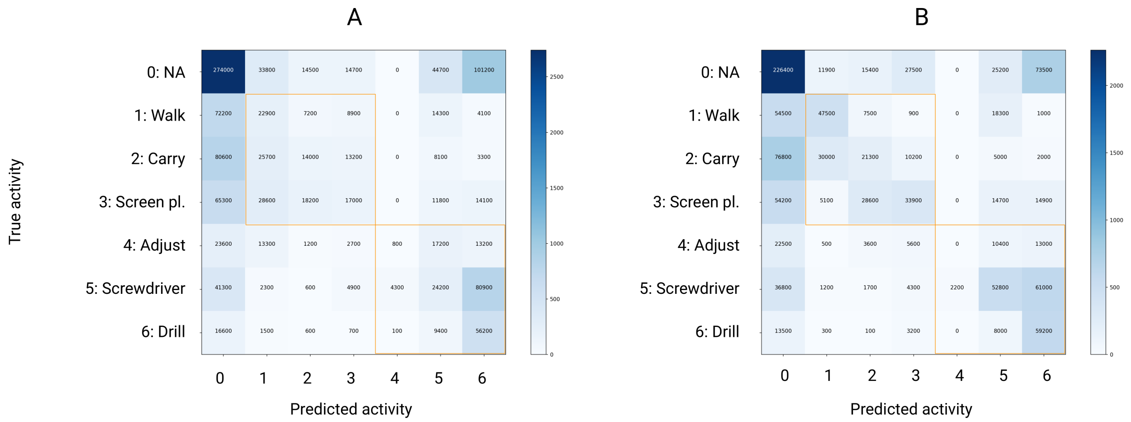

Figure 8. Confusion matrices for six the class problem using eye-based features only (A) and using all features (B) with 10 seconds window for dataset independent training 1. These frame-based results implicate that some classes should be merged together (see two class problem).

TABLE I. FRAME BASED PRECISION AND RECALL VALUES FOR THE SIX CLASS PROBLEM, DATASET INDEPENDENT TRAINING

class	*eye*		*ACC+snd*		*all*	
	P	R	P	R	P	R
no activity	0.45	**0.63**	0.44	0.44	**0.47**	0.60
Walk	0.32	0.04	0.48	**0.60**	**0.49**	0.42
Carry	**0.32**	0.15	0.23	0.04	0.27	**0.19**
Screen pl.	0.14	0.09	0.27	0.01	**0.40**	**0.29**
Adjust	0.00	0.00	0.00	0.00	0.00	0.00
Screwdriver	0.26	0.29	0.35	0.23	**0.39**	**0.33**
Drill	**0.32**	0.14	0.12	0.62	0.26	**0.70**

features, ACC+snd: combination of accelerometer and sound features, all: combination of all three feature sources), it presents precision (P) and recall values (R).

One important observation is that for almost every class, the combined feature set (all) outperforms the other two baselines. In some cases (Screwdriver, Screen placement), the combination is clearly better for both recall and precision value. In contrast, the Walk activity's recall is higher (0.60) using ACC+snd features and is reduced in combination (0.42). Recognition of the Adjust activity doesn't work, most probably, because it is not well represented in the training samples (due to missing data, the activity's sample instances were cut out by chance). In some other training-test split strategies, it can be detected (as seen on Figure 8-A).

TABLE II. EVENT BASED PRECISION AND RECALL VALUES FOR THE SIX CLASS PROBLEM, DATASET INDEPENDENT TRAINING

class	*eye*		*ACC+snd*		*all*		*filter*	
	P_e	R_e	P_e	R_e	P_e	R_e	P_e	R_e
Walk	0.54	0.45	0.46	**0.92**	0.53	0.67	**0.67**	0.54
Carry	0.52	0.48	0.19	0.40	0.39	**0.66**	**0.68**	0.51
Screen pl.	0.42	0.52	0.22	0.09	0.50	**0.61**	**0.51**	0.41
Adjust	0.10	0.02	0.00	0.00	0.00	0.00	0.00	0.00
Screwdriver	0.19	0.53	0.40	0.88	0.42	**0.90**	**0.49**	0.78
Drill	0.22	0.61	0.14	**1.00**	0.26	0.61	**0.30**	0.53

Results of the event-based analysis for the six class problem are presented in Table II. For the evaluation of the first three columns (eye, ACC+snd, all) no event filter or merging is applied. The fourth column named "filter" merges events that are closer than 10 seconds and removes events shorter than 1.5 seconds based on the combined (all) features' event detections.

Most important observations in the event-based results are:

- for some classes (carry, screen placement) eye features work best, while for other events (e.g., walk, screwdriver) the accelerometer and sound combination work best
- the merged features (all) achieves overall better results
- as expected, by using the event merging and filtering, the precision of the event classifiers could be increased (at the cost of a lower recall)

To understand the nature of the errors, we analyzed the results of the combined feature set and event filtering (filter) with the event analysis diagrams proposed by [38]. Figure 9 illustrates the detailed event score distribution for different classes for the dataset independent training. According to this analysis, 60% of Walk events are successfully detected. However, detected events for Screwdriver and Drill classes are mainly dominated by insertions (I'). These activities are easily mistaken for one another (as can also be seen in Figure 8), which can help account for the low precision values obtained, too.

C. *Results for two classes*

We expect the classifier to perform significantly better for the two class problem, as many of the errors reported above are due to confusions between similar activities (again, see Figures 8 A and B).

TABLE III. FRAME BASED PRECISION AND RECALL VALUES FOR THE TWO CLASS PROBLEM, DATASET INDEPENDENT TRAINING

class	*eye*		*ACC+snd*		*all*	
	P	R	P	R	P	R
no activity	**0.50**	0.43	0.43	**0.57**	0.48	0.52
Large	0.62	0.45	0.61	0.36	**0.69**	**0.48**
Precise	0.47	**0.74**	0.38	0.45	**0.55**	0.73

Frame by frame results of the two class problem are in Table III. The values in this table confirm our expectations. Using only data from the eyetracker (10 second window, independent training) Large class can be detected with a precision and recall of 0.62 and 0.45, meanwhile the classifier achieves 0.47 precision and 0.74 recall for Precise class. Similarly to the

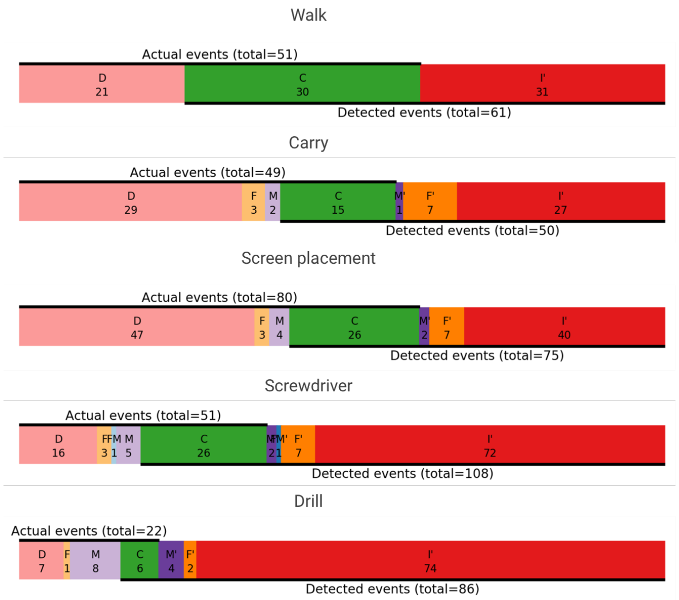

Figure 9. Event analysis diagram for the six class problem (class Adjust had zero detections), dataset independent training. (C denotes correct events, D deletions, I' insertions)

TABLE IV. Event based precision and recall values for the two class problem, dataset independent training

class	*eye*		*acc+snd*		*all*		*filter*	
	P_e	R_e	P_e	R_e	P_e	R_e	P_e	R_e
Large	0.74	0.83	0.54	**0.86**	0.80	0.72	**0.80**	0.61
Precise	0.57	0.96	0.39	**0.99**	0.62	0.98	**0.63**	0.76

six class problem, the combined feature set has an improved overall performance compared to the baselines.

Figure 10 presents a perfect example while the pure frame-based evaluation is not sufficient in our application. The temporal distribution of the detections is also very important. The figure shows the ground truth and detected events for person 2 over the dataset A. It can be observed that the majority of the detections for a class is around the real activity. Using event filters as described above, we also can filter out some of the wrong detections.

Table IV shows the actual event length weighted precision and recall values for all four categories (eye features only, using ACC+snd, using all features, using event filter as well). For Precise events, the classifier has a precision of 0.62 with a recall of 0.98 when using the combined feature set.

The event analysis diagram (EAD) of the two class problem is shown in Figure 11. For both Large and Precise classes many of the correctly detected events are merged with other ground truth events. This merging together of correct events might not be strictly incorrect for some applications, but it does indicate ill-defined event borders. The Precise class events perform slightly better in terms of number of correctly discovered events. However, this result is dominated by insertion errors, which are likely responsible for the lower precision value.

D. *Results for one class*

Instance of the Precise class are important for the high level analysis, since this activity category is typically performed alone. It is characterized by, small precise movements over a longer time period. In this evaluation, we test the classifier to spot these events against the empty class ('no activity').

TABLE V. Frame based precision and recall values for the one class problem, dataset independent training

class	*eye*		*ACC+snd*		*all*	
	P	R	P	R	P	R
no activity	0.89	0.64	0.75	**0.74**	**0.90**	0.70
Precise	0.43	0.78	0.35	0.23	**0.50**	**0.80**

Results of the frame-based analyses are in Table V. As expected, using only eye based features results in better performance then using the accelerometer and sound combination. The best performance is produced again by the all combined feature set with a precision of 0.5 and recall of 0.8 percent.

TABLE VI. Event based precision and recall values for the one class problem, dataset independent training

class	*eye*		*ACC+snd*		*all*		*filter*	
	P_e	R_e	P_e	R_e	P_e	R_e	P_e	R_e
Precise	0.59	0.96	0.35	0.97	0.64	**0.98**	**0.66**	0.81

The event based results in Table VI show the same improvement characteristics as observed for the other label sets. However, the accelerometer and sound based recognition has a high recall, but a poor precision value for a single class problem. Using only the eye features results in a fairly good 0.59 and 0.96 precision and recall. These values are further improved when using the combined feature set with 0.64 precision and 0.98 recall. When applying event filters, we get higher precision (0.66) on cost of the recall value (0.81).

The event analysis diagram (EAD) (see Figure 12) of the one class problem shows less deletions and insertions for the Precise class compared to the two class problem's figure (see Figure 11). The majority of the ground truth events are detected merged with other events. The reason for this is probably, as seen on Figure 10, that, many occurrences of the Precise class (ground truth) occur close to each other.

VI. Computer vision on egocentric videos

In addition to the eye-feature based activity recognition, we also explored the possibility of using the world camera images as an information source. Specifically, we wanted to find out how "off the shelf" machine vision tools might be used to automatically process the camera data. In this section, we describe our findings on the use of automatic face detection, object recognition around the gaze point, and scene recognition.

A. *Objects of interest*

The initial idea is based on the assumption that a person looks at specific objects more often if these are related to his or her current task. One such example is the activity of fixing screws with a screwdriver. We would expect that it is more likely to find a screwdriver in the egocentric video of the person doing this task, than it would be in the video of someone who was doing something else. If we can detect the objects that the person looked at during specific activities, we can train a classifier using this information and use it to later recognize the same activity again.

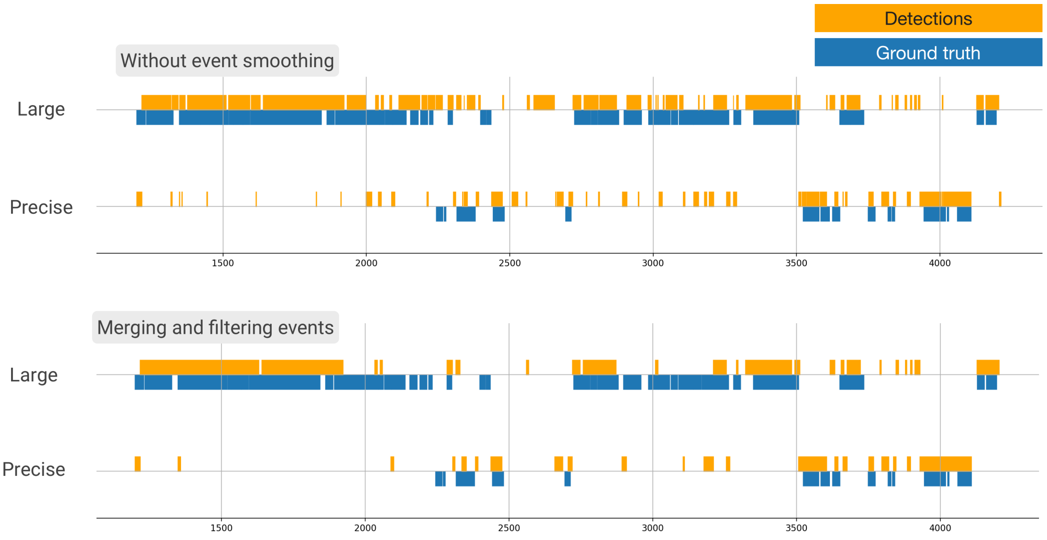

Figure 10. Ground truth events and detections for the two class problem for person two in dataset A without (top) and with (bottom) event filtering and merging. The classifier was trained using all features combined and only with data from dataset B.

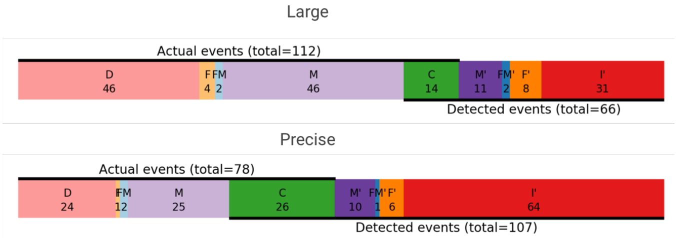

Figure 11. Event analysis diagram for the two class problem, dataset independent training

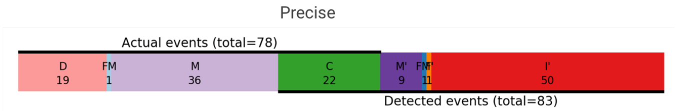

Figure 12. Event analysis diagram for the one class problem, dataset independent training

To test the feasibility of this idea, we implement the following workflow:

1) Crop each frame of the eyetracker's world video around the gaze point with a radius of ± 200 pixels horizontally and vertically to obtain the region of interest (ROI).
2) Use deep learning models pretrained on 1000 classes of the ImageNet database to obtain prediction weights on the ROI. We tested the Keras implementation of the following models: InceptionV3, ResNet50, VGG16, VGG19 and Xception.
3) We calculated the average of the weights in one second windows over the dataset and stored as a feature matrix.
4) Using the activity ground truth label and the feature matrix generated from the prediction weights, we trained a Naive Bayes classifier with the same methods and training strategies as for the eye based features.

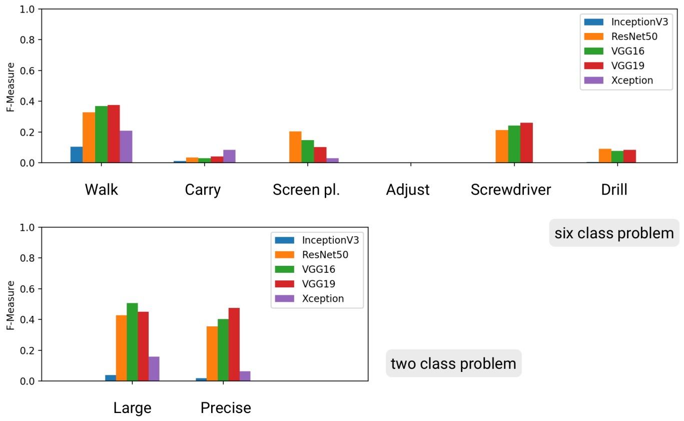

Figure 13. F-measure of activity classifiers trained on prediction weight features of different models with dataset independent 2 training for six class problem (top) and two class problem (bottom)

Results for the activity recognition using the prediction weight based features are presented on Figure 13 for the six class problem (top) and for the two class problem (bottom). The figure displays the f-measure, that can be interpreted as a weighted harmonic mean of the precision and recall, for each class. The values represented on the figures are from a dataset independent training. The dataset identical trainings showed nearly identical results without much improvement.

On Figure 13, we can see that using the ResNet50, the VGG16 or the VGG19 models, the classification result for Walk and Screwdriver are comparable with eye features or the all feature combination from above. InceptionV3 and Xception models are not working well on these datasets. Detection results for the two class problem (Figure 13) are worse than the performance of the eye features on the same two classes, but still better then random guessing of the class though.

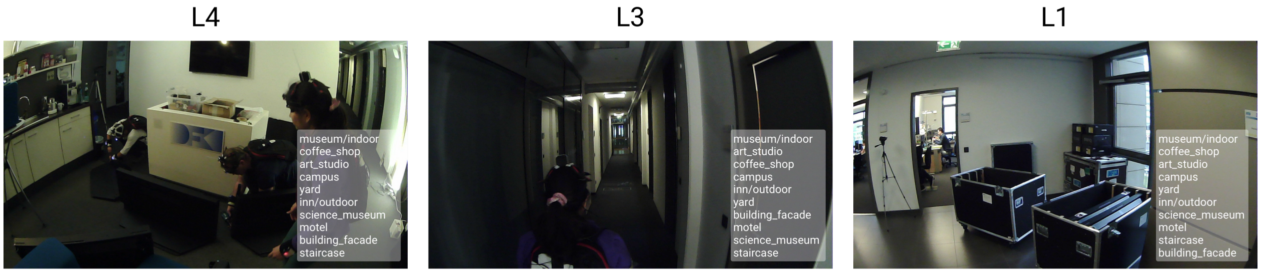

Figure 14. Top ten classes for scene recognition on first person camera images at different locations using a VGG16 model trained on the Places365 dataset. As expected, the predictions are very similar, but the weights' distributions are slightly different.

Motivated by these results, we merges the object recognitions features with the eye, accelerometer and sound features from earlier in the paper using the same merging method as before. The merged features are then evaluated using the same strategies and concepts as above.

Table VII and VIII contains frame- and event-based evaluation values for two feature combinations:

1) eye features extracted with a 10 second window together with object recognition features (eye+obj)
2) eye, ACC and sound features combined with the object recognition features (all+obj)

In both cases, we use the VGG19 pretrained model to obtain the object recognition features and use independent evaluation strategy 2. By comparing Table VII with the above results in Tables I and II, the values are slightly worse than before. For the two class problem however, this approach brings a minor improvement on frame-by-frame as well as on event level for both feature sets (see Table VIII vs. Tables III and IV).

TABLE VII. Evaluation results for combinations with the object recognition features using the pretrained VGG19 for the six class problem, dataset independent training

	Frame-based				*Event based*			
	eye+obj		*all+obj*		*eye+obj*		*all+obj*	
class	P	R	P	R	P	R		
no activity	**0.53**	0.33	0.52	**0.45**	-	-	-	-
Walk	0.25	**0.74**	**0.43**	0.44	0.48	**0.86**	**0.55**	0.77
Carry	0.25	0.02	**0.34**	**0.15**	0.37	0.07	**0.59**	**0.61**
Screen Pl.	0.21	0.2	**0.34**	**0.33**	0.33	**0.59**	**0.51**	0.56
Adjust 0.00	0.00	0.06	0.01	0.00	0.00	0.00	0.21	0.02
Screwdriver	0.24	**0.33**	**0.32**	0.27	0.42	0.77	**0.46**	**0.82**
Drill	0.08	0.12	**0.21**	**0.76**	0.14	0.47	**0.21**	**0.56**

TABLE VIII. Evaluation results for combinations with the object recognition features using the pretrained VGG19 for the two class problem, dataset independent training

	Frame-based				*Event based*			
	eye+obj		*all+obj*		*eye+obj*		*all+obj*	
class	P	R	P	R	P	R		
no activity	**0.53**	0.36	0.52	**0.39**	-	-	-	-
Large	0.66	**0.67**	**0.69**	0.55	0.66	0.70	**0.83**	**0.80**
Precise	**0.53**	0.73	0.52	**0.84**	**0.63**	**0.97**	0.60	0.74

B. *Face detection*

For analyzing interactions in the group, an important information source could be if we are able to detect if the person looks on another participant's face. Additionally, co-location information can be important to recognize collaboration events. If we can detect specific person's face on the egocentric video, this can be a clear sign of co-location. As an initial exploration towards these goals, we tested the face detection library OpenFace [31]. The authors report an accuracy of ca. 0.93 with their default model on the LFW benchmark [40].

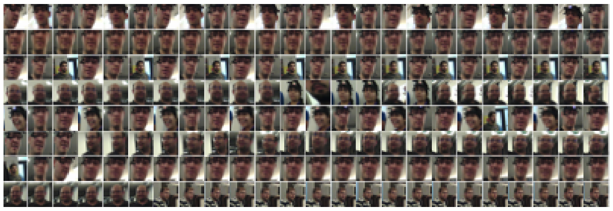

Figure 15. Examples for face detection using the OpenFace library on our dataset. It spots faces robustly throughout the experiment. (Image quality artificially reduced here)

We run the face detection on each frame of the eyetracker videos an saved position, bounding box size and face features together with the corresponding timestamp into a feature file for later processing. Figure 15 illustrates some examples of the detected faces in a sequence.

We performed a statistical analysis of the temporal distribution of the face detections compared with activity labels. We found that during activities of the Precise category (using screwdriver, drill or adjust) the average number of face detections per second is much lower (0.05) than during Large activities (0.26). The reason behind this statistical difference is, that these activities are typically performed independently from other participants, meanwhile the persons are physically connected for many Large activities especially during the Carry event. Also they used the time e.g., during carry event for social interactions. Another interesting observation we made, was that two of the participants look at each other much more often than other ones. We found that meanwhile these two knew each other well from before the experiment, they had no or little prior relations to the other ones, suggesting that this kind of data could be used for analyzing social structures in the group, as also done by [32]. But further analysis of this topic is outside the scope of the current work.

Lastly, we compared temporal distribution of the face detections against co-location of the participants. Figure 16 shows a scatter plot with the number of faces detected in a 15 seconds window against the ratio if the person is co-located with one or more other participants during this window. On the x-axis, zero means that during the window the person is completely alone, 100 percent means that during the whole time he or she is co-located with exactly one other participant.

The distribution of the sample points suggests a correlation between a high number of faces detected and co-location. However, this feature alone is not sufficient to predict co-location. Even if all four participants are at the same location, they do not necessarily look in each other's direction. This explains the sample points on the figure that have zero or very few faces detected even during higher values of co-location.

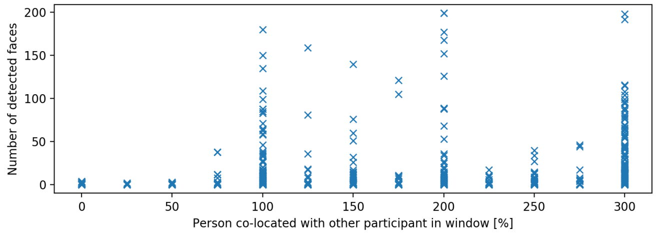

Figure 16. Number of faces detected in a window of 15 seconds versus the time spent co-located with another participants during this window. Values above 100% mean that the subject is co-located with more than one participant at the same time.

C. Location detection with scene recognition

Finally, we investigate location detection using egocentric video. Our assumption is that deep learning models trained to recognize different scenes can extract useful features for distinguishing locations in our dataset.

To test this assumption, we used the Keras implementation [41] of an VGG16 model trained on the Places dataset [42]. This model tries to predict a scene description for the input image. Figure 14 shows example predictions (top 10 for each) for three different locations in our dataset. As expected, the predictions are similar (are parts of the same building), but the order (and also the weight distribution) is different for each.

We calculated the output weights (predictions) of the network for each video frame from the eyetrackers and saved them into feature matrices. We also experimented to cut the network at an earlier stage (not the final class predictions), but that didn't improved our results and introduced higher computational needs. On the generated feature set and location labels, we evaluated the concept with a nearest neighbor classifier using only one person's data for training and every other participant's data for testing. Precision and recall values were computed and then averaged for all four combinations of dataset A. For the four location classes, the classifier achieved and averaged precision and recall of 0.5774 and 0.558, well above random values. Considering the difficulty of the task (image quality, similarity of the places, in many cases bad lighting as visible on Figure 14), it's a promising result and could be an useful input for a high level decision making system.

VII. Conclusion and Future work

We evaluated the use of data from wearable eye trackers as part of a multi-sensor system for recognizing real-world physical activities. In the two datasets we used, participants perform (sometimes heavy) physical construction tasks in an unrestricted order over a duration of around 90 minutes. Due to the realistic data collection setup, data is in many cases unavailable or unreliable.

Despite the challenging conditions, we show in this work that it is still possible to obtain useful recognition results using eye tracking. Although the gaze point may not be correct, due to factors such as loss of calibration during the experiment, we can nonetheless successfully recognize key activity categories by utilizing calibration free eye features (based on non-gaze eye properties and movements). The results additionally show that recognition can be achieved in both a person and dataset independent way. That means that we can expect the system to work on future datasets without any additional training effort.

The combination of eye features with simple acceleration (measured on the right wrist), and sound features, produces even more reliable results especially on an event level. Additional event filtering can further reduce the number of false positive outputs. For detecting the broader classes of Large activities (involving full body movement) and Precise activities (involving smaller hand-tool manipulations), we achieved event-based (duration weighted) precision and recall values of 0.74 and 0.83 using only eye features, and 0.57 and 0.96 using the combination of eye, acceleration and sound features.

Exploration of the image processing methods on egocentric videos showed promising results despite the challenging dataset (bad lighting, blurry images caused by sudden movements, etc.). Although the initial results for location, co-location detection, or activity recognition using recognized objects are not perfect, they point towards potentially valuable areas upon which future research on this topic might be focused.

We aim to further explore the combination possibilities of eye features with visual object detection and scene analysis. Additional topics for investigation are the combination of sound analysis with face detection to recognize interactions within the group, and hand detection on egocentric videos for object manipulation detection.

Acknowledgment

The work is funded by the German Federal Ministry of Education and Research (BMBF) through the project iGroups.

References

[1] P. Hevesi, J. A. Ward, O. Amiraslanov, G. Pirkl, and P. Lukowicz, "Analysis of the usefulness of mobile eyetracker for the recognition of physical activities," in *UBICOMM 2017 The Eleventh International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies*. IARIA XPS Press, November 2017, pp. 5–10.

[2] A. Gruenerbl, G. Bahle, and P. Lukowicz, "Detecting spontaneous collaboration in dynamic group activities from noisy individual activity data," in *Pervasive Computing and Communications Workshops (PerCom Workshops), 2017 IEEE International Conference on*. IEEE, 2017, pp. 279–284.

[3] A. Bulling, J. A. Ward, H.-W. Gellersen, and G. Tröster, "Robust recognition of reading activity in transit using wearable electrooculography," in *PERVASIVE*, May 2008, pp. 19–37.

[4] M. Vidal, D. H. Nguyen, and K. Lyons, "Looking at or through?: Using eye tracking to infer attention location for wearable transparent displays," in *Proceedings of the 2014 ACM International Symposium on Wearable Computers*, ser. ISWC '14. New York, NY, USA: ACM, 2014, pp. 87–90. [Online]. Available: http://doi.acm.org/10.1145/2634317.2634344

[5] S. Ishimaru, K. Kunze, K. Kise, J. Weppner, A. Dengel, P. Lukowicz, and A. Bulling, "In the blink of an eye: Combining head motion and eye blink frequency for activity recognition with google glass," in *Proceedings of the 5th Augmented Human International Conference*, ser. AH '14. New York, NY, USA: ACM, 2014, pp. 15:1–15:4. [Online]. Available: http://doi.acm.org/10.1145/2582051.2582066

[6] B. Zhou, J. Cheng, M. Sundholm, A. Reiss, W. Huang, O. Amft, and P. Lukowicz, "Smart table surface: A novel approach to pervasive dining monitoring," in *Pervasive Computing and Communications (PerCom), 2015 IEEE International Conference on*. IEEE, 2015, pp. 155–162.

[7] C. Debes, A. Merentitis, S. Sukhanov, M. Niessen, N. Frangiadakis, and A. Bauer, "Monitoring activities of daily living in smart homes: Understanding human behavior," *IEEE Signal Processing Magazine*, vol. 33, no. 2, pp. 81–94, 2016.

[8] P. Hevesi, S. Wille, G. Pirkl, N. Wehn, and P. Lukowicz, "Monitoring household activities and user location with a cheap, unobtrusive thermal sensor array," in *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, ser. UbiComp '14. New York, NY, USA: ACM, 2014, pp. 141–145. [Online]. Available: http://doi.acm.org/10.1145/2632048.2636084

[9] A. Bulling, U. Blanke, and B. Schiele, "A tutorial on human activity recognition using body-worn inertial sensors," *ACM Computing Surveys (CSUR)*, vol. 46, no. 3, p. 33, 2014.

[10] H. Lu, W. Pan, N. D. Lane, T. Choudhury, and A. T. Campbell, "Soundsense: Scalable sound sensing for people-centric applications on mobile phones," in *Proceedings of the 7th International Conference on Mobile Systems, Applications, and Services*, ser. MobiSys '09. New York, NY, USA: ACM, 2009, pp. 165–178. [Online]. Available: http://doi.acm.org/10.1145/1555816.1555834

[11] Y. Yang, B. Guo, Z. Yu, and H. He, "Social activity recognition and recommendation based on mobile sound sensing," in *2013 IEEE 10th International Conference on Ubiquitous Intelligence and Computing and 2013 IEEE 10th International Conference on Autonomic and Trusted Computing*, Dec 2013, pp. 103–110.

[12] N. Eagle and A. S. Pentland, "Social network computing," in *International Conference on Ubiquitous Computing*. Springer, 2003, pp. 289–296.

[13] D. Wyatt, T. Choudhury, J. Bilmes, and J. A. Kitts, "Inferring colocation and conversation networks from privacy-sensitive audio with implications for computational social science," *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 1, pp. 7:1–7:41, Jan. 2011.

[14] T. Nishimura, T. Higuchi, H. Yamaguchi, and T. Higashino, "Detecting smoothness of pedestrian flows by participatory sensing with mobile phones," in *Proceedings of the 2014 ACM International Symposium on Wearable Computers*, ser. ISWC '14. New York, NY, USA: ACM, 2014, pp. 15–18. [Online]. Available: http://doi.acm.org/10.1145/2634317.2642869

[15] J. A. Ward, P. Lukowicz, G. Tröster, and T. E. Starner, "Activity recognition of assembly tasks using body-worn microphones and accelerometers," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 28, no. 10, pp. 1553–1567, October 2006b.

[16] J. A. Ward, P. Lukowicz, G. Pirkl, and P. Hevesi, "Detecting physical collaborations in a group task using Body-Worn microphones and accelerometers," in *13th Workshop on Context and Activity Modeling and Recognition (CoMoRea'17)*, Big Island, USA, Mar. 2017, pp. 268–273.

[17] R. Barea, L. Boquete, M. Mazo, and E. Lopez, "System for assisted mobility using eye movements based on electrooculography," *Trans. on Rehabilitation Engineering*, vol. 10, no. 4, pp. 209–218, December 2002.

[18] M. Wedel, R. Pieters *et al.*, "Eye tracking for visual marketing," *Foundations and Trends® in Marketing*, vol. 1, no. 4, pp. 231–320, 2008.

[19] R. J. Jacob, "Eye tracking in advanced interface design," *Virtual environments and advanced interface design*, pp. 258–288, 1995.

[20] L. A. Granka, T. Joachims, and G. Gay, "Eye-tracking analysis of user behavior in www search," in *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR '04. New York, NY, USA: ACM, 2004, pp. 478–479. [Online]. Available: http://doi.acm.org/10.1145/1008992.1009079

[21] A. Poole and L. J. Ball, "Eye tracking in hci and usability research," *Encyclopedia of human computer interaction*, vol. 1, pp. 211–219, 2006.

[22] A. Bulling, J. A. Ward, H. Gellersen, and G. Tröster, "Eye movement analysis for activity recognition," in *(Ubicomp) Proceedings of the 11th international conference on Ubiquitous computing*. New York, NY, USA: ACM, 2009, pp. 41–50.

[23] J. Nagel, P. Hevesi, C. Beck, U. Gengenbach, and G. Bretthauer, "Eyelid detection in eye-tracking experiments," *BIOMEDICAL ENGINEERING-BIOMEDIZINISCHE TECHNIK*, vol. 59, pp. S496–S499, 2014.

[24] M. Vidal, A. Bulling, and H. Gellersen, "Pursuits: spontaneous interaction with displays based on smooth pursuit eye movement and moving targets," in *Proc. UbiComp*. ACM, 2013, pp. 439–448.

[25] M. Dhuliawala, J. Lee, J. Shimizu, A. Bulling, K. Kunze, T. Starner, and W. Woo, "Smooth eye movement interaction using eog glasses," in *Proceedings of the 18th ACM International Conference on Multimodal Interaction*. ACM, 2016, pp. 307–311.

[26] Y. Shiga, T. Toyama, Y. Utsumi, K. Kise, and A. Dengel, "Daily activity recognition combining gaze motion and visual features," in *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct Publication*. ACM, 2014, pp. 1103–1111.

[27] F. Chollet *et al.*, "Keras," https://github.com/keras-team/keras, 2015, accessed February 15, 2018.

[28] R. Girshick, I. Radosavovic, G. Gkioxari, P. Dollár, and K. He, "Detectron," https://github.com/facebookresearch/detectron, 2018, accessed February 15, 2018.

[29] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," *CoRR*, vol. abs/1512.00567, 2015. [Online]. Available: http://arxiv.org/abs/1512.00567

[30] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[31] B. Amos, B. Ludwiczuk, and M. Satyanarayanan, "Openface: A general-purpose face recognition library with mobile applications," CMU-CS-16-118, CMU School of Computer Science, Tech. Rep., 2016.

[32] A. Fathi, J. K. Hodgins, and J. M. Rehg, "Social interactions: A first-person perspective," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, 2012, pp. 1226–1233.

[33] M. S. Ryoo and L. Matthies, "First-person activity recognition: What are they doing to me?" in *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*. IEEE, 2013, pp. 2730–2737.

[34] H. Pirsiavash and D. Ramanan, "Detecting activities of daily living in first-person camera views," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, 2012, pp. 2847–2854.

[35] M. Kassner, W. Patera, and A. Bulling, "Pupil: An open source platform for pervasive eye tracking and mobile gaze-based interaction," in *Adjunct Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, ser. UbiComp '14 Adjunct. New York, NY, USA: ACM, 2014, pp. 1151–1160. [Online]. Available: http://doi.acm.org/10.1145/2638728.2641695

[36] R. Bachu, S. Kopparthi, B. Adapa, and B. Barkana, "Separation of voiced and unvoiced using zero crossing rate and energy of the speech signal," in *American Society for Engineering Education (ASEE) Zone Conference Proceedings*, 2008, pp. 1–7.

[37] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[38] J. A. Ward, P. Lukowicz, and H. W. Gellersen, "Performance metrics for activity recognition," *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 1, pp. 6:1–6:23, Jan. 2011. [Online]. Available: http://doi.acm.org/10.1145/1889681.1889687

[39] P. Hevesi, "Ward metrics for event based activity recognition evaluation," https://github.com/phev8/ward-metrics, 2017, accessed February 15, 2018.

[40] G. B. H. E. Learned-Miller, "Labeled faces in the wild: Updates and new reporting procedures," University of Massachusetts, Amherst, Tech. Rep. UM-CS-2014-003, May 2014.

[41] G. Kalliatakis, “Keras-places,” https://github.com/GKalliatakis/Keras-VGG16-places365, 2017, accessed February 15, 2018.

[42] B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, and A. Torralba, “Places: A 10 million image database for scene recognition,” *IEEE transactions on pattern analysis and machine intelligence*, 2017.