University of Applied Sciences Mittweida and Chemnitz University of Technology at TRECVID 2018

Rico Thomanek[1], Christian Roschke[1], Robert Manthey[2], Benny Platte[1], Tony Rolletschke[1], Manuel Heinzig[1], Matthias Vodel[1], Danny Kowerko[2], Stefan Kahl[2], Frank Zimmer[1], Maximilian Eibl[2], and Marc Ritter[1]

[1]University of Applied Sciences Mittweida, D-09648 Mittweida, Germany
[2]Chemnitz University of Technology, D-09107 Chemnitz, Germany

Abstract. The analysis of video footage regarding the identification of persons at defined locations or the detection of complex activities is still a challenging process. Nowadays, various (semi-)automated systems can be used to overcome different parts of these challenges. Object detection and their classification reach ever higher detection rates when making use of latest cutting-edge convolutional neural network frameworks. In our contribution to the *Instance Search* task, we specifically discuss the design of a heterogeneous system which increases the identification performance for the detection and localization of persons at predefined places by heuristically combining multiple state-of-the-art object detection and places classification frameworks.
In our initial appearance to the task of *Activity of Extended Video* (ActEV) which is engaged in the detection of more complex activities of persons or objects, we also incorporate state-of-the-art neural network object detection and classification frameworks in order to extract bounding boxes of salient regions or objects that can be used for further processing. However, a basic tracking of objects detected by bounding boxes needs a special algorithmic or feature-driven treatment in order to include statistical correlations between the individual frames. Our approach describes a simple but yet powerful method to track objects across video frames.

1 Introduction to our Appearance in the Instance Search Task

The new approach for processing Instance Search Task 2018 (INS) Awad et al. (2018, 2017) involves a cumulative process consisting of several components for automated evaluation and interactive search. We retrieved 1.2 million keyframes from the 464 hours (in around 44 million frames) of video footage from the BBC series *EastEnders* by our dynamic reduction method (Ritter et al. (2015, 2014)) and stored them within a database. To process the task, we divided the overall problem into the individual tasks of person recognition and location recognition. Subsequently, we merged the solutions of the two subtasks in order to achieve better results.

In our work we focused mainly on the underlying architecture which is adaptable to other application domains. We developed a complex but yet extensible system for data preparation, data processing and data evaluation based on Docker containers and the persistent storage of metadata in relational databases. We extracted the metadata by making use of a variety of existing frameworks and their feature extraction capabilities. The underlying data set contains 300 GB of extracted data which represents the 464 hours of video footage from 244 TV series (*BBC EastEnders*) encoded in MPEG-4/H.264 format. (cf. to Cao et al. (2016))

Correspondence to: Marc Ritter
marc.ritter@hs-mittweida.de

2 System Architecture

Our system consists of three client units to perform the detection tasks. A database server is responsible for the persistent storage of all results, including the respective raw data. To access the stored raw data, an *Apache* web server was used, which provides complete access to any stored information via HTTP protocol. In consequence, it is not necessary to store raw data for processing tasks on desktop computers any further. Instead, the required source material is obtained from the web server via HTTP directly during the data processing. As a consequence, all results can also be stored in the respective database. In addition, we developed an API which provides the data in XML / JSON-based exchange formats. Thus, new desktop computers can be added quickly and easily for further processing, including the distribution & scaling of data processing tasks. All intermediate and final results generated throughout the working process are stored

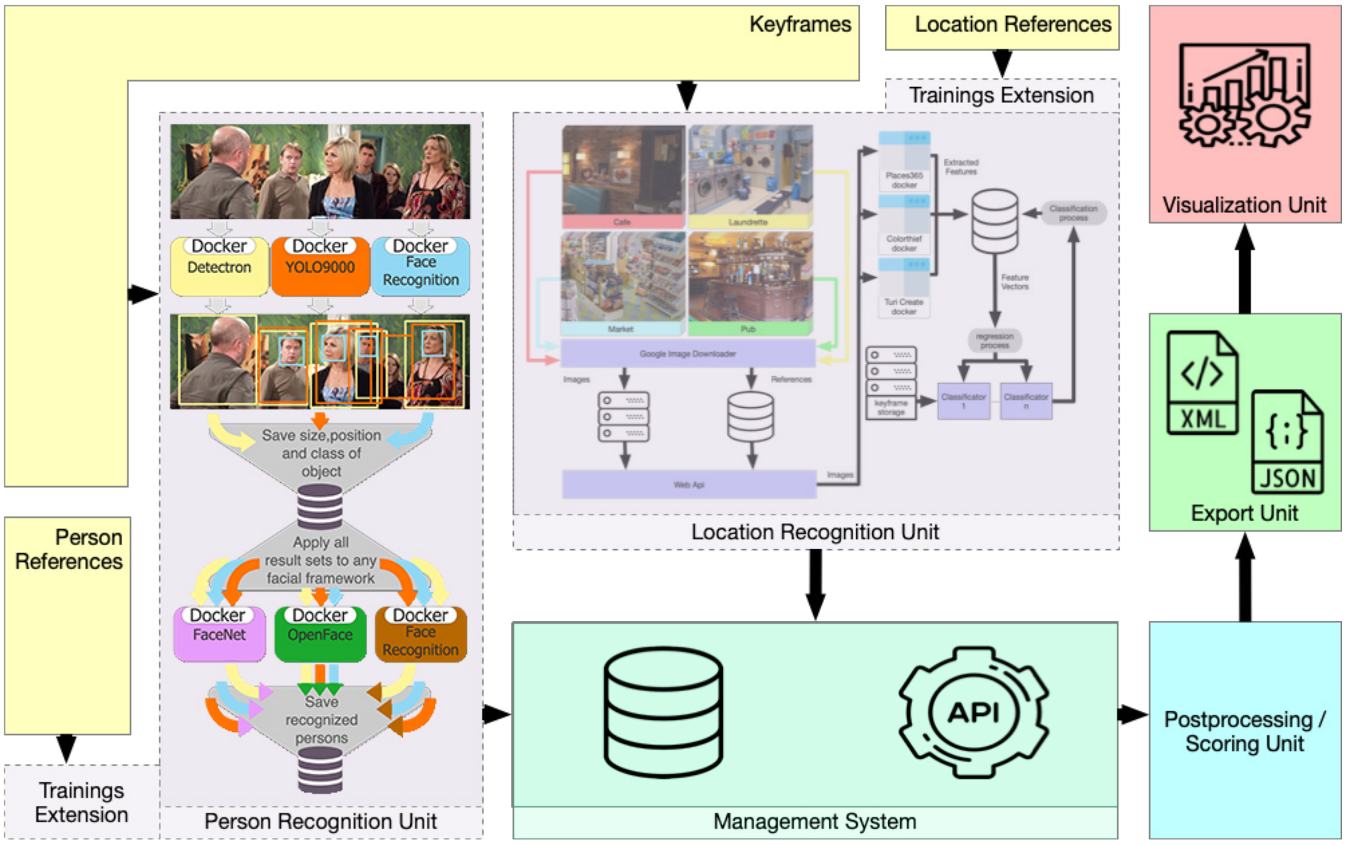


Figure 1: Our holistic system workflow for Instance Search.

in the database. Based on the respective API, any hardware node is capable of accessing this data directly.

Our approach to solve the Instance Search Task makes use of multiple feature extraction approaches based on open source frameworks. Identical detection tasks are processed by different frameworks and the respective results are transferred into the database. Since the different frameworke provide individual strengths and weaknesses, the detection results can be merged by suitable regression and scoring procedures allowing for a boost in the detection performance. The feature extraction processes for persons and locations are executed separately from each other in parallel. Identified persons and given locations are merged again within a dedicated storing procedure by the usage of appropriate SQL statements.

The feature extraction process can be assigned to three dedicated processing units. In order to increase performance and efficiency, all frameworks are executed in docker containers. Accordingly, no additional software installations are required on the clients site. In addition, the execution of multiple instances of the same framework on a single host reduces the processing time significantly. Each framework was integrated into a HTTP-based batch processing system. Hence, each single video shot is loaded from the web server by using an HTTP GET request referring either to a person or location recognition process. In a next step, the system loads the respective results and stores the retrieved data into the database.

The basic structure of the system is shown in Figure 1. Reference images of locations and people are used by training extensions of the Location Recognition Unit (LRU) and the Person Recognition Unit (PRU). The reference images are used for training and the creation of classifiers. Both the PRU and the LRU use these classifiers to perform recognitions in the provided keyframes. All knowledge and metadata gained from these processes are managed in the management system and become accessible via the API. To perform the INS task, the management system transmits all results obtained to the Processing / Scoring Unit. This contains the business logic for evaluation and generates a result object which is then transferred from the Export Unit to an XML or JSON container. The Visualization Unit uses these containers directly for visualizing the results. Thus, it is possible to create an interface for the intelligent annotation of the data for the INS Interactive Search Task and to measure and evaluate the quality of the results intellectually after the competition period for subsequent work or a more thorough analysis.

2.1 Frameworks

For the identification of places and people we use different state of the art frameworks. Most of these frameworks only allow a directory-based processing of images. For the integration of such frameworks into our system environment, we extended the source code of all frameworks with online processing functions. Hence, images are not stored locally on the host system, but are downloaded directly from the web server during the runtime. For this purpose, a specific API was developed.

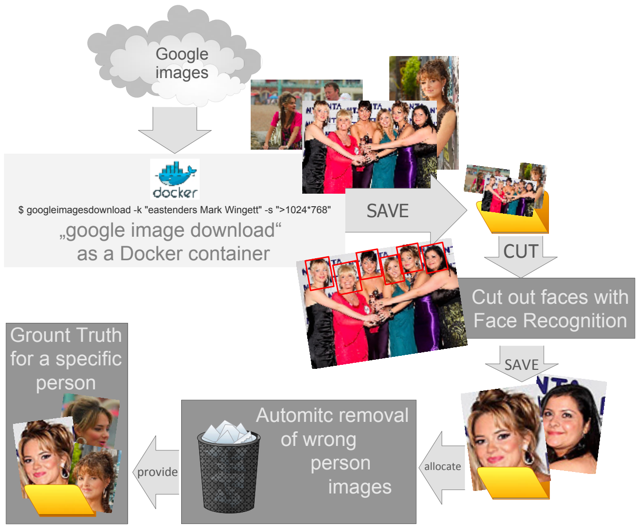
We separate the recognition of places and people from each other. Any results are merged into feature vectors. The following evaluation step processes the data with appropriate regression and clustering methods. Accordingly, the identification of a specific person at a defined location is achieved by using SQL queries. The relational linking of all results to the actual shot is performed via shot ID, which is implemented by using foreign keys. For a profound feature extraction, the following frameworks are used: *Places365*, *Turi Create*, *Color Thief*, *Detectron*, *Yolo9000*, *FaceNet*, *OpenFace* and *FaceRecognition*.

Places365 is a subset of Places2, a dataset which contains about 10 million images in over 400 scene categories. The dataset features 5,000 to 30,000 training images per class. We use a training set of Places365-Standard with about 1.8 million images from 365 scene categories. Based on this dataset, we use the *ResNet152-places365* framework, which was trained from scratch using PyTorch. This is the original ResNet with 152 layers (Zhou et al., 2017). The framework allows ten classifications with decreasing accuracy to be determined for each image. In addition, it provides the first ten attributes found within the *SUN* attribute dataset (Patterson and Hays, 2012) for each individual image. The classifications results as well as the respective attributes and the prediction are saved as feature vectors for each individual image.

TuriCreate, an open source machine learning framework, was developed by Apple in 2017. TuriCreate focuses on developers with little or no experience in the context of machine learning. The framework offers several libraries for the implementation of various tasks of the domain of machine learning. E.g., this includes image classification, image similarity, clustering or regression (Sridhar et al., 2018). Our use case for this framework deals with image similarity. Here, comparable images to a particular reference image have to be found. Such images presumably contain similar optical

Table 1: Examples of object classes in *COCO-Dataset* (Lin et al., 2014)

object	**COCO object class**						
vehicles	things	→	outdoor	→	vehicle	→	car
						→	motorcycle
						→	truck
persons	things	→	outdoor	→	person	→	person


Google images
SAVE
CUT
Cut out faces with Face Recognition
SAVE
allocate
Automitc removal of wrong person images
provide
Grount Truth for a specific person
docker
„google image download“ as a Docker container

```
$ googleimagesdownload -k "eastenders Mark Wingett" -s ">1024*768"
```

Figure 2: Workflow preprocessing of images from specific persons.

features and characteristics compared to the reference image. In order to determine these characteristics, deep learning techniques are used whereas the model is generated by unsupervised learning. Thus, no annotated data is required. We used a pre-trained neural network from *ImageNet*, which is able to distinguish between 1,000 categories out of the box. However, we ignore the classification of the output layer and use the neural network only to extract the feature vectors of the images. Finally, the individual features of the images are combined by a nearest neighbors algorithm to determine the images with the highest similarity.

We use the Python module **Color Thief** for grabbing, extracting and saving color palette and dominant colors from an image. The generated vector consists of ten colors which in turn are composed of the values red, green and blue resulting in a total of 30 values for the complete image (Feng, 2016). We use this Python module to extract the color palette and the dominant color (again in RGB format).

A framework which contains object detection algorithms is **Detectron**. The framework was developed by Facebook, is based on the deep learning framework Caffe2 and offers a high-quality as well as a high-performance codebase for object detection. Various algorithms such as Mask R-CNN, RetinaNet, Faster R-CNN, RPN, Fast R-CNN and R-FCN are implemented. In addition, several backbone network architectures such as ResNet50, ResNet101, ResNet152, VGG16 or Feature Pyramid Networks can be used. Facebook also provides a large set of trained models for further use with the *Detectron Model Zoo dataset*. (Girshick et al., 2018)

We applied Detectron to detect objects and their bounding boxes. In addition to the x- and y-coordinates of the starting point, we also store the height and width of the box in addition to the object classification. Furthermore, we use

ResNet152 as a backbone net in combination with Mask R-CNN. The classes assigned to each image are obtained from the Coco dataset (Lin et al., 2014), shown in examples at Section 2.1.

"YOLO" (You Only Look Once) is a network for real-time classification and localization of objects within an image. In contrast to a stack-based and step-by-step approach to object detection, object recognition is processed as a single regression problem. (Redmon et al., 2016)

YOLO9000 represents a revision of the YOLO framework, since it made significant mistakes, especially in object localization. Furthermore *YOLO9000* with over 9,000 different object categories was trained. (Redmon and Farhadi, 2016) We use *YOLO9000* to detect objects and their bounding boxes. The position and size of the bounding box is described by *YOLO9000* using upper left corner coordinates and lower right corner coordinates. The detected object class and the bounding box values are stored in the database.

FaceNet is a Face Recognition Framework based on the paper "FaceNet: A Unified Embedding for Face Recognition and Clustering". *FaceNet* is a one-shot model that directly learns a mapping of historical images to a compact Euclidean space in which distances directly correspond to a measure of facial similarity (Schroff et al., 2015). There are currently several implementations of *FaceNet* available. We use the implementation of *David Sandberg*, which also employs the framework *Tensorflow*. For us, *FaceNet* is one of three frameworks for person recognition. Again, we store the retrieved results (classification and prediction) into our database. For this purpose, we extended the source code of the framework for online processing, so that the keyframes can be automatically retrieved by the web server, processed and the results stored in the database.

The framework **OpenFace** is used for face recognition and was developed by Brandon Amos, Bartosz Ludwiczuk and Mahadev Satyanarayanan. It is also based on the paper *FaceNet: A Unified Embedding for Face Recognition and Clustering*, but unlike the *FaceNet* framework, it was implemented using Torch. (Satyanarayanan et al.) We use OpenFace for personal identification and store the results in the database again.

FaceRecognition is a framework for finding and identifying faces. The framework uses face recognition from *dlib*, which has been enhanced with Deep Learning (Geitgey and Nazario, 2017). The source code of *Face Recognition* has been extended again for online processing and the results are also stored in the database.

2.2 Data handling and interface

For the persistent storage and delivery of data, we use a combined system of an object-relational database and a RESTful web API. This allows us to store data and final results together with the intermediate results that are necessary for later processing steps, and distributes the workload optimally across several systems.

Table 2: Number of ground truth images per character.

Person	Images retrieved
Max	297
Jack	296
Heather	243
Zainab	225
Darin	116
Mo	112
Garry	91
Minty	57
Jane	129
Chelsea	186

In order to provide a suitable data storage solution, we have to choose between relational, object-oriented and object-relational database systems. In the object-oriented database system, data is assigned to an object and stored in it. The database management system (DBMS) takes care of the organization and administration. A disadvantage of this model is that the mapping of very complex data records in objects can lead to a decrease in performance, which is why we omitted it. In the relational database system, data is stored in tables, which are retrieved using a defined query language. Links between the tables are realized by the use of foreign keys. Data can be represented adequately; however and due to the flat structure, one difficulty is to represent complex hierarchical relationships. The object-relational system is based on the relational system. Data is stored in tables and linked using foreign keys. Further, it is also possible to map object-oriented structures and thus hierarchical relationships. We selected *PostgreSQL* as a representative of the object-relational database systems for the persistent storage of metadata due to the listed advantages.

For the *Instance Search Task*, a vast amount of image data needs to be stored. We decided to come up with a file-system based solution. In order to still allow access to the individual files, we implemented an additional layer. In this case, we use an Apache Webserver which is capable to deliver the images via HTTP request. This allows us to store the references to all images directly as meta information in the database. Furthermore, we developed a web-based API using the model-view-controller (MVC) framework Laravel. This enables us to sort, filter, process and make the data records available in any form and on the fly. We use a chain of docker instances in order to process all data analysis tasks, and outsourced the necessary system parameters via command line parameters. Docker is an open source software for isolating applications with container virtualization. Docker simplifies the deployment of applications by making it easy to transport and install containers containing all the necessary packages as files.

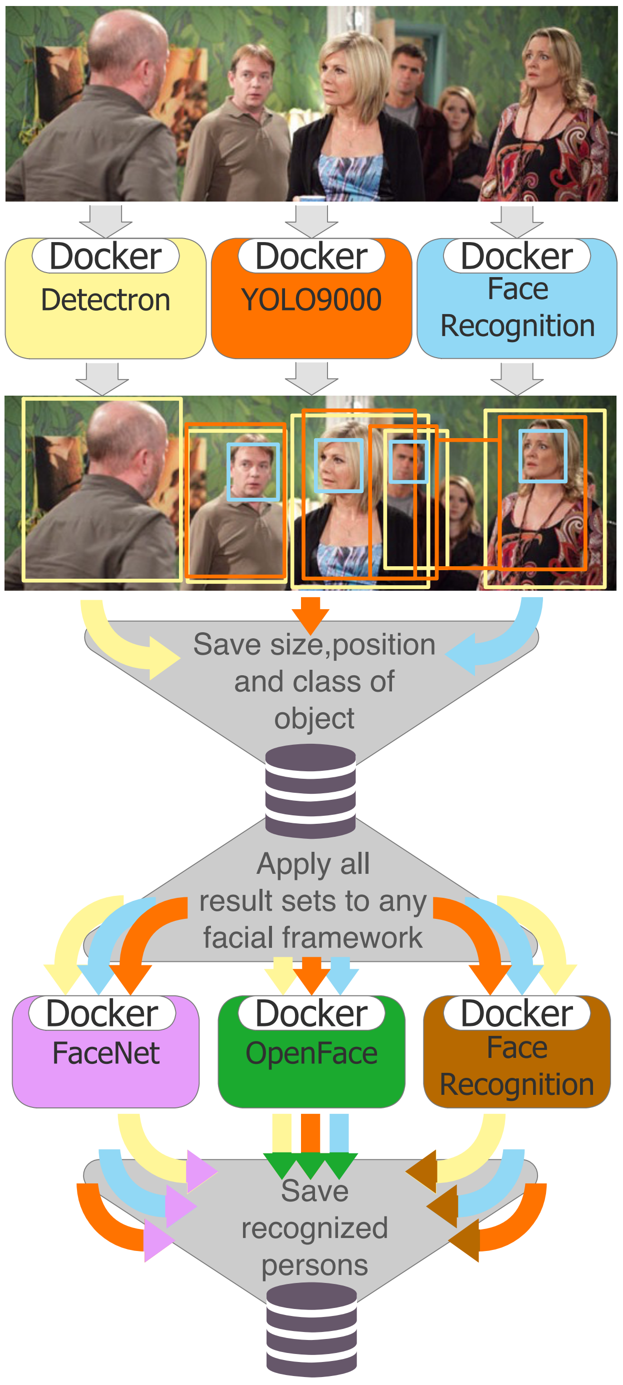


Figure 3: Workflow for the recognition of persons.

2.3 Person detection and recognition

In order to create ground truth data for classification and person recognition tasks, we make use of the Google image search tool. For the partial automation of the download process, we use the MIT license based software "google-image-download". This software allows the command line based download of images by specifying parameters to define the search phrases and the desired image size. In order to obtain as many images as possible, the command line tool is executed several times with different parameter values for each target person. Depending on the frequency of appearance of an individual actor, an average of 200-300 images can be collected using this method. Since only the faces of the actors are relevant for the creation of the classifiers, it is necessary to extract them from the downloaded images. We utilize the tool *Face Recognition* (under MIT license) for the task of face detection and classification. The *Face Recognition* detection mechanism is capable to extract and deliver the bounding boxes of all the faces detected. In order to retrieve the detected region of interest, we implemented our own method as well as some functionality that extends the boundary of returned bounding box in such a way that it covers also the entire surrounding head region including hair and neck. In the interactive part of the task, the resulting image set may then be forwarded to an intellectual annotation step which aims to primarily remove inappropriate images. The output of this processing pipeline defines the ground truth data set for an individual person or respective actor, and finally consists of about 50–300 images in average for each individual person. An schematic overview of this processing pipeline is shown in Figure 2.

The resulting number of images for each person is shown in Table 2. According to the requirements documentation of the framework *OpenFace* (Satyanarayanan et al.), a minimum number of 10 images per person is required to create a classifier (numImagesThreshold=10). Accordingly, the frameworks *Face Recognition* and *FaceNet* require a minimum number of images somewhere in the single-digit range. With an average number of 150 images per person, we expect our ground truth data to surpass those requirements during the application runtime including some tolerance range.

The generated person images are then transferred to the web server while being indexed into the database. As shown in Figure 3, the person recognition consists of in three major components. In the first step, all persons and their absolute positions are determined using the frameworks *YOLO9000*, *Detectron* and *Face Recognition*. For each framework, the detection results are stored in a separate database table with each framework detecting a different number of persons. *Detectron* achieved the highest detection rate with 2,515,332 retrieved faces. The framework *Face Recognition* detected 1,384,747 faces and *YOLO9000* yields 1,013,007 persons. In the second step, the frameworks *Face Recognition*, *FaceNet* and *OpenFace* were used for person recognition. For each framework, a classifier is generated based on the previously created ground truth. In order to classify people, an image section is created for each person entering in step one as a mere base for person recognition. Since each classifier is applied to the three results of step one, the result is a 3 constellation. The resulting nine tables are merged by means of our heuristic scoring schema shown in formula (1). In our approach, the detection performance of each framework is considered equally. This leads to the fact that the prediction value (2) for a recognized person can be evaluated by each framework with a maximum of 100 points. After the summation of all single prediction values, a maximum of 900 points (4) can be achieved per person.

$$predPerson = \sum_{h=0}^{r} (x_h \in K), \tag{1}$$

$$K = \{x \mid 0 \leq x \leq 100\}, \tag{2}$$

$$predPerson \in L, \tag{3}$$

$$L = \{y \mid 0 \leq y \leq 900\}. \tag{4}$$

In order to calculate the scoring value, we create a new holistic table with the merged values from all nine tables. All keyframes in which an individual person was found are retrieved. The prediction values of the same keyframe are then added up. Since some frameworks provide several bounding boxes for one person, we always use the higher prediction value. The discussed prediction values are then stored in the new table. In addition, for each entry we store the person ID, the URL on the keyframe in relation to the frameworks involved for the person identification.

For the subjective evaluation of the results, we have developed a web service for visual representation. Figure 4 illustrates that the number of false detections decreases with increasing score value. At the same time, however, the number of images is also reduced. In this regard, it should be noted that good detection results are achieved around 300 points. According to the rules of the evaluation campaign, knowledge gained from visual processing has not been included into the automatic evaluation.

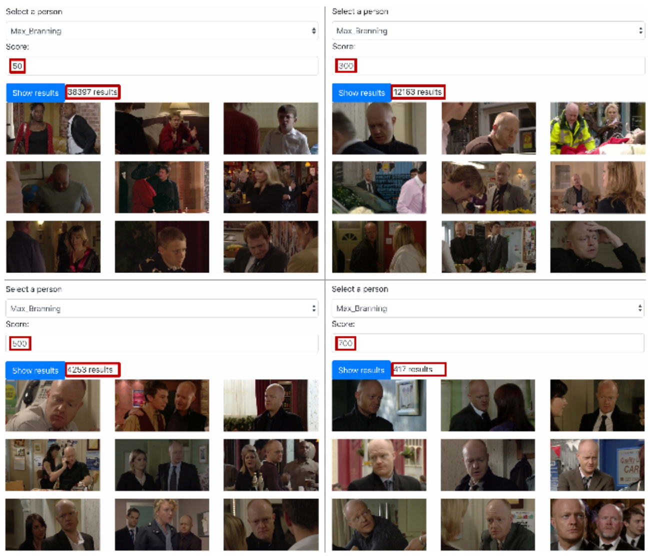


Figure 4: With a higher score value, the probability of person recognition increases.

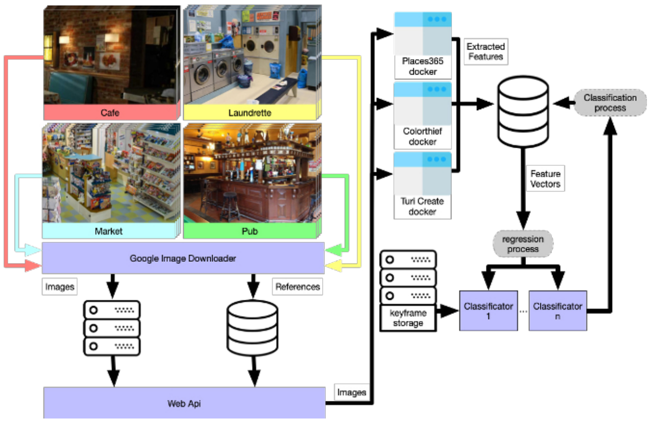


Figure 5: Workflow for recognizing locations

2.4 Location recognition

The basis for the location detection are a data set of 14,423 reference images of the locations *cafe*, *laundrette*, *market* and *pub*. Therefore, we developed the workflow shown in Figure 5. In this context, we automatically search for the sample images shown in Table 3 using the procedure described in Section 2.3 in an analogous manner. The images are made accessible via an API and the references are stored in the database. As with the processing of the person images, the underlying database architecture allows us to put all meta information directly into relation. At the beginning, the reference data retrieves features by using the frameworks *Places365*, *Places365-Attributes*, *Color Thief dominant color* and *Color Thief color palette*. *Places 365* contains 365 classes, whereby for each image of the ten most probable classifications with the corresponding prediction values are stored for further processing.

In addition, we store the ten most frequently recognized attribute values (*eating people*, *people*, *tables*, *outdoor*, ...) that the framework has recognized and utilized for classification. By using *Color Thief* we also determined the dominant color for each reference image and stored the corresponding r-, g- and b-values separately. In addition, the framework provides us with a color palette of ten colors which we also transfer to the database according to the same scheme as the dominant color. In addition, we use *TuriCreate* to determine the ten most similar images using the integrated image similarity procedure in order to create a similarity classifier. By applying *Yolo* and *Detectron*, we continue to extract all ob-

Table 3: Reference images of the places.

Places	**Images retrieved**
cafe	7,529
laundrette	1,007
market	3,347
pub	2,540

jects found in the images. For each extracted feature, we apply the regression methods *Boosted Tree*, *Decision Tree* and *Random Forest* in order to get three classifiers per feature (feature classifiers).

The created classifiers (all feature classifiers and the similarity classifier) are then used to classify the keyframes extracted from shot boundaries provided by NIST. In this respect, a scoring procedure was developed which summarizes all classification results (5) and calculates a score. Each classification result can reach a maximum of 100 points, whereby a total of 2,100 points can be achieved over all features. The score of a feature is multiplied by the probability determined by the classifier for each possible class and the values obtained are summed up in correlation to the respective class. The variable m stands for the corresponding class and x_i for the prediction value of the regression method of the respective feature i. If only one class with a prediction is determined as the result of a classification, the prediction values for all other classes are set to 0.

$$predLocation_m = \sum_{i=0}^{p} (x_i \in M) \cdot 100, \qquad (5)$$

$$M = \{ x \,|\, 0 \leq x \leq 1 \} \qquad (6)$$

While incorporating all scoring values, result tables can be created and analyzed for each keyframe. Table 4 shows an example of the structure of such a table. As shown in (5), the displayed points are then summed up and divided by the maximum total score of 2,100 in order to create normalized values.

$$scoreLocation_m = (\sum_{j=0}^{q} x_j \in N) \,/\, 2100, \qquad (7)$$

$$N = \{ x \,|\, 0 \leq x \leq 100 \}. \qquad (8)$$

The evaluation of the results in Table 4 illustrates an example of the scores achieved in Table 5. The result with the highest relative score is selected while the other values are discarded. The provided example indicates that a corresponding keyframe is the location *Market* with a confidence of 0.05

Table 4: Example of a results table of the scoring procedure for a keyframe

Features	**Cafe**	**Laundrette**	**Market**	**Pub**
F1 + BoostedTree	0	0	44	0
F1 + RandomForest	0	22	0	0
F1 + DecisionTree	0	0	34	0
...	...	...	...	...
Fn + BoostedTree	0	0	0	10
Fn + RandomForest	0	0	20	0
Fn + DecisionTree	0	0	12	0

Table 5: Example evaluation of the scoring results of a keyframe

Results	**Cafe**	**Laundrette**	**Market**	**Pub**
total score	0	22	110	0
relative score	🟥 0.00	🟥 0.01	🟩 0.05	🟥 0.00

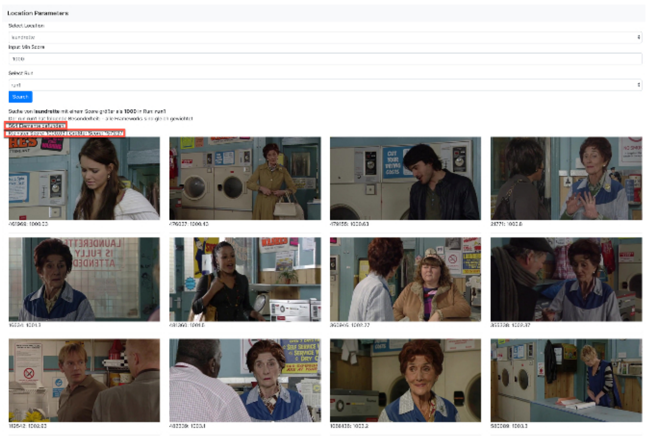
whereas the provided keyframe appears as the same as the requested location (*Market*).

As with person recognition, we use a web service to visually display the results in order to subjectively evaluate the resulting results. Figure 6 shows an example of the developed interface. With a score of 1,000 good detection results could already be achieved. According to the rules, knowledge from the visual overview was not included in the automatic evaluation.

2.5 Fusion of the determined scoring values

Starting from the scoring result tables for the location and person recognition process, an overall score can be calculated. The two input tables contain the locations and persons with the highest score value, and will now be correlated to each other and normalized to obtain a final score value. In order to submit these results to the Instance Search Task review board, a number of 1,000 scoring results for each person in respective locations needs to be provided, and the result set must be transformed into a XML document. The necessary transformation steps and calculation formulas are shown below. It is important to note that all tasks after the initial step 1 are optional and need to be executed only if the required image quantity has not yet been achieved.

1. At first, the score values of the recognized persons and locations are recorded for each keyframe (formulas (9), (10)). A vector product is then calculated over the set of keyframes whose person and location score is greater than or equal to 50% of the maximum score (12)–(14).



Suche von laundrette mit einem Score größer als 1000 in Run run1

Figure 6: Screenshot of the subjective location score view.

The result set for a given search topic T finally contains the set LP of all keyframes for the set Q of all defined search queries, containing all vectors of the requested persons and places (15).

2. If it was not possible to retrieve 1,000 results in the first step, further results are determined based on the keyframes from step 1. All neighbour shots are used whose person/place combination matches the search criterion, regardless of the scoring values.

3. If the previous step 2 did not provide a sufficient result set, all neighbouring shots containing the searched person are used, based on the results of step 1. During this step, the assigned location is ignored in addition to the scoring values.

4. The procedure described in step 1 is repeated to replenish the required result set, whereby the range of accepted scores is relaxed to a value between 25% and 50% of the maximum score.

5. Still missing results are finally replenished by merging the personal identifications whose score value exceeds 50% of the maximum score. The determined location is ignored.

$$P_{Keyframe\_x} = \{\text{Score}_{Person\_1}, \ldots, \text{Score}_{Person\_n}\}, \tag{9}$$
$$L_{Keyframe\_x} = \{\text{Score}_{Location\_1}, \ldots, \text{Score}_{Location\_m}\}, \tag{10}$$
$$LP_{Keyframe\_x} = P_{Keyframe\_x} \times L_{Keyframe\_x}\,. \tag{11}$$

$$P_{Keyframe\_x} \times L_{Keyframe\_x} := \left\{ (y,z) \;\middle|\; \begin{array}{c} y \in P_{Keyframe\_x} \\ y \geq \frac{PS_{max}}{2} \\ z \in L_{Keyframe\_x} \\ z \geq \frac{LS_{max}}{2} \end{array} \right\}, \tag{12}$$
$$PS_{max} = 900, \tag{13}$$
$$LS_{max} = 2{,}100, \tag{14}$$
$$T = LP \cap Q\,. \tag{15}$$

2.6 Interactive Evaluation

For the interactive evaluation of the TRECVid INS task, we use our optimized web-based interactive platform for evaluation (WIPE) from 2016. WIPE is a stable, scalable and reusable platform to integrate services and evaluate the system using TRECVid.

We select a web-based architecture as the basic system because it offers several advantages over native applications. On the client side, a browser installed on most operating systems can be used and there is no need to install additional tools. Another advantage is the automatic and persistent backup of data in a central database. Each transaction of a user is time-exactly related to the current data and is stored time-exactly to these. In the domain of web development there are various frameworks to support the development process. As a basis we chose the MVC Framework Laravel, because it offers the following advantages: Chen et al. (2017)

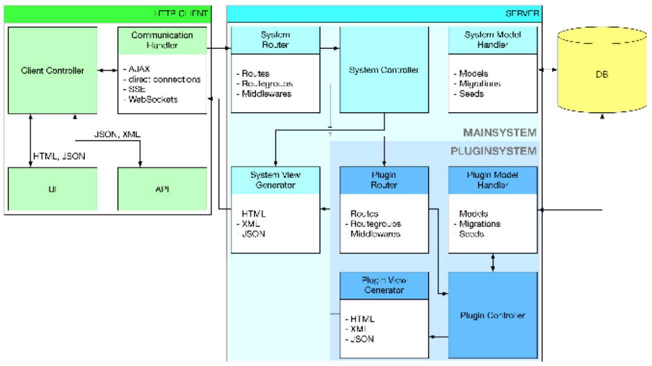


Figure 7: Management architecture

1. Logic can be distributed as required so that calculations can be performed on the server, client and database side allowing for simple load distribution.

2. All developed systems can be easily extended and maintained due to a clear separation of concerns. Any changes made to the code can be interpreted and displayed at runtime.

3. Created modules can be easily reused, parameterized and updated.

4. The framework already provides a lot of commonly required functionality, which facilitates and supports the progress within the development process.

The open source framework Laravel follows the development pattern Model View Controller (MVC) and is licensed under MIT. Laravel offers multiple components in the standard installation, which can be used by developers for larger and more complex developments. In addition to the integration of Laravel modules, Symfony components can also be integrated.

Figure 8 shows our Laravel-based system architecture. A plugin manager has been added to the classic Laravel components. This allows the administration and integration of plugins and thus the expansion of the framework by additional functionalities. This allowed us to develop components for iBeacon communication and to add personalized content to the standard views.

The user interface for annotation developed in our appearance at TRECVid 2016 still allows to select a run and to visualize the current annotation status of the registered user.

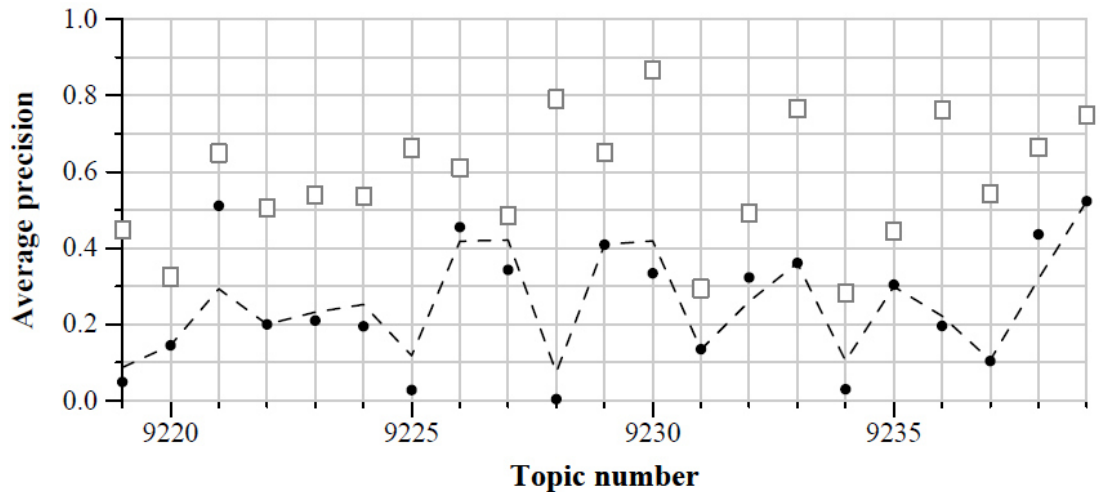


Figure 8: Results of our interactive submission (Run 4) provided by the organizers. Dots are our scores, the dashed line indicates the median score and boxes represent the best results of other participants.

After the start of an annotation process a timer is generated and set to 5 minutes. The start time is transferred to the server and stored there to prevent fraud or manipulation of the timer. Several visualizations are generated for the user, each containing 2×3 result images of the corresponding query. The user can change the status of an image using keyboard shortcuts. Changing the status of an image changes the color to green and vice versa. After the timer has expired, the user interface is deactivated while the transfer is being uploaded to the server. Kahl et al. (2016)

In this period, we keep the basic interface and the underlying architecture whereas the business logic and the relations of objects has been integrated into the database with respect to performance issues. Furthermore, we add automated mechanisms to transfer the results of the automatic processing directly into the system. Thus we can significantly increase the speed of data preparation for intellectual annotation.

3 Results and Future Work in Instance Search

In this years edition of the TRECVid evaluation campaign we only achieve mediocre results. In most topics and categories, our system performs sub-par when compared to the work of other teams. Furthermore, we are unable to continue our series of minor but steady improvements in result quality, that we obtained over the last years. The main reason for this is our fully reconstructed system, which features a whole new software architecture pattern: a diverse set of algorithms which are distributed over various machines for efficient calculation and finally merge their results into a central database. The implementation took us a lot of time and efforts depriving us of the opportunity to fully flesh out some undoubtedly necessary fine tuning of basic parameters and weight vectors in the software. It is yet to be noted, that due of the new structure we were now capable to precalculate all necessary data from the given video corpus. This enables us to solve the retrievement step within (milli)seconds once the targeted topics were announced.

3.1 Run 1: Full Set Emphasizing Dominant Color

With our first automatic run, we were able to return 3,478 of 11,717 relevant shots at a mean average precision (MAP) of 0.105. The run scored a precision@100 of 0.264. It features the full set of frameworks we assumed to be useful to fulfill the given task the best possible way (FaceNet, FaceRecognition, Openface for person detection as well as Places365, Turic Create Similarity, Dominant Color, Yolo, Detectron for place recognition). Originating from our experience of the last years, we incorporate the dominant color framework with a weight of 1.2 over all others that are taken into account at an equal factor of 1.0. Consequently, our best results are topics 9242 and 9239 which both feature a noticeable color characteristic.

3.2 Run 2: Full Set of Equal Weights

With nearly the same setup as Run 1, this automatic run features all frameworks that are present in our system. This time though, all of them are weighted equally without any presumptions and corresponding preferences on an analytic aspect. This leads to 3,488 returned relevant shots out of 11,717. This - in comparison to the previous run - slightly higher return count manifests itself in a somewhat better MAP (0.109) while maintaining the precision@100 (0.264). The fact that we still see topics 9242 and 9239 with the best results leads to the conjecture, that our system is intrinsically focused on color information in general, even if we do not encourage this via parameter settings.

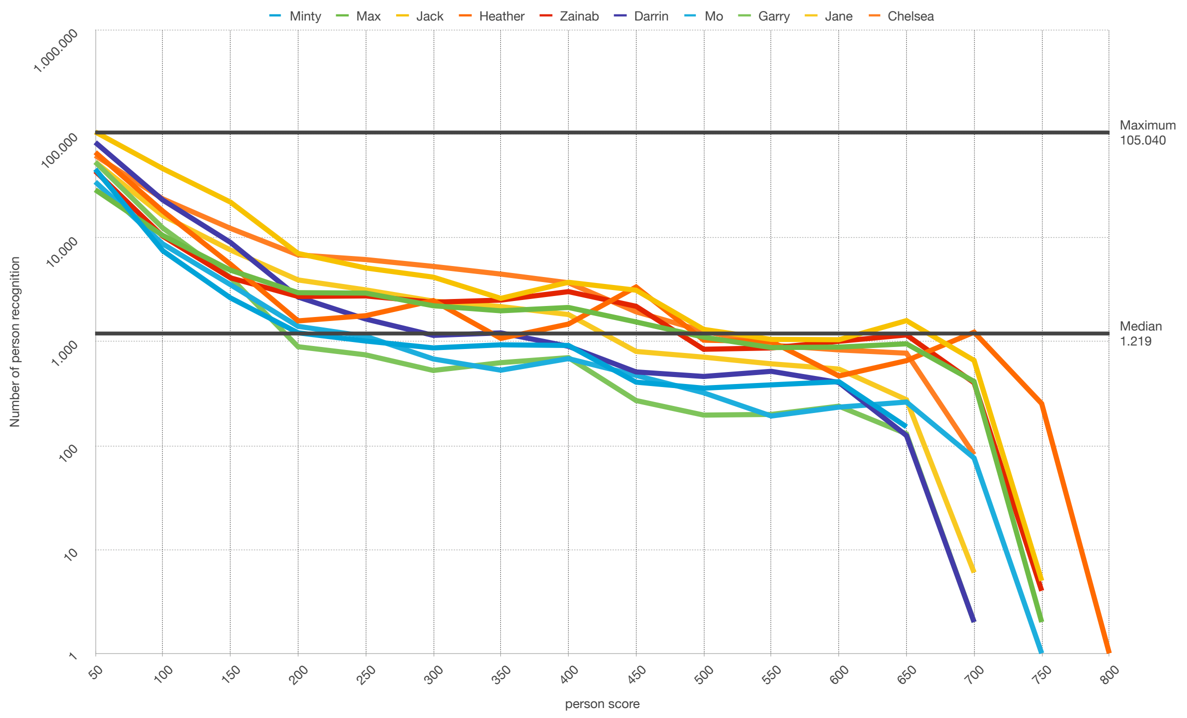


Figure 9: Number of recognized persons per score

3.3 Run 3: Full Set emphasizing similarity

For our final automatic run, all available frameworks are used once more. This time, we emphasize the similarity framework, which focuses on the structural and textural similarities of an image. To be consistent with our other runs, we increase the weight of the corresponding framework with a factor of 1.2, while all other frameworks remain at an equal coefficient of 1.0. Using this setup, we end up with 3,550 retrieved relevant shots out of 11,717. With the resulting Mean Average Precision of 0.111 and a slightly higher precision@100 of 0.268 the run emerges as our best automated attempt for this years TRECVid iteration. As a consequence from the embedded prioritization of structural information, topics which rely on such characteristics show could benefit from small improvements in system performance.

3.4 Run 4: Interactive Full Set emph. dom. color

As we have a bit of a track record when it comes to interactive runs a TRECVid (one might even denote as ”tradition”), we decided to prioritize a run with such setup over yet another automatic variation. Using the results of our full system with an 1.2 weighting of the dominant color module, we improve the results using our self coded evaluation tool from two years ago (Kahl et al. (2016)). This enabled us to significantly increase the quality of our results at that time. This year it repeatedly proved itself a valuable addition to our toolchain. With the help of the tool, we achieve a MAP of 0.252 and a precision@100 of 0.454, which outperforms the results of our fully automated runs by far. The fact that we only retrieved 3,036 out of 11,717 relevant shots (less than the automated runs) does not lessen the fact that the interactive run is by far our best entry to this years TRECVid evaluation period.

3.5 Conclusion

Figure 9 shows the number of persons recognized per score. Based on these statistics, it can be stated that an average of approximately 1,200 correct identifications were determined for a person. Furthermore, a visual inspection also confirms that good results can be already obtained by using a score of 300. However, we found that at least two different frameworks are required for a reliable person recognition. Moreover, it can be stated that only the framework *OpenFace* could identify persons with full scores for the data sets of *YOLO9000* and *Detectron* for only four provided keyframes. For a score of 300 points or more, at least three result tables consisting of at least two different frameworks for the identification of persons need to be involved.

Furthermore, we were able to determine that each framework delivers different prediction values even with a small change in the image region (a few pixels). The targeted multiple classification of the same object using slightly different image sections could thus possibly contribute to an increase in the identification performance.

For location recognition, we noticed that the total score of the individual locations often differs only slightly. One reason could be the determined parameters for the regression

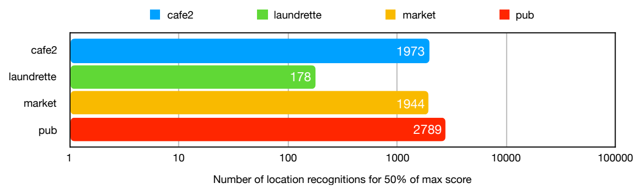


Figure 10: Number of recognized locations for 50% of max score

methods used, which were not chosen optimally. In the future, we expect machine learning methods (e.g. boosting) to deliver seriously increased scorings. Figure 10 shows that at 50% of the maximum score for *cafe2*, *market* and *pub* an average of 2,235 locations were detected. An exception appears to be the *laundrette* with only 178 detections.

The reference pictures and shots often contain actors in the foreground covering the relevant background. The methods *Color Thief* and *Image-Similarity* could also be used to consider the actors in their evaluation, which increases the probability of false detection. A future approach could also incorporate the detection of persons and a reconstruction of the background while combining the structure and texture of such regions with synthetic methods.

4 Introduction to our Appearance at Activitiy in Extended Video

Over the last few years, the number of surveillance cameras has increased worldwide. Closed circuit television (CCTV) cameras record a steadily increasing amount of image data. Usually, such data is only reviewed after an specific event being thoroughly investigated for indications of noteworthy actions. In the domain of traffic safety (e.g. monitoring of intersections) or other sensitive areas, there is an increasing desire to evaluate the videos in a sensible and resource-saving way. In the context of ”predictive policing”, movements of objects including persons are to be detected. Monitoring movements for several hours is commonly known as a monotonous task that requires a high level of concentration and quickly tires people. A further effect is *unaware blindness*: With high concentration on an object or an activity, unexpected objects slip through unrecognized—even when the actions are taking place in the center of the field of vision (Simons and Chabris, 1999). An intellectual evaluation is mainly limited by personnel resources and the human error rate through monotonous work. Previous work in this area builds on features vectors based on reference data in order to execute tracking algorithms Lillo et al. (2017); Zhuang et al. (2017); Jalal et al. (2017). In the area of big data, using several thousand hours of video footage from a wide variety of situations, it is not feasible to use those techniques to a bigger extend.

This paper focuses on our approach to solve the TRECVid Activity in Extended Video task (ActEv, Awad et al. (2018)) and in this case the automatic detection of object activities within surveillance areas. The goal of ActEv is to develop robust automatic activity detection algorithms for multi-camera streaming video environments. ActEV is a subtask of TRECVid as well as an extension of the Surveillance Event Detection Task (SED) and aims to enable real-time detection. The data set to be analyzed is VIRAT V1 (Oh et al., 2011), a data set consisting of approximately 450 different video sequences from the area of video surveillance. ActEV includes the challenges “Activity detection”(AD) and “Activity and object detecion”(AOD). In “AD” activities are to be detected and the frame areas of the detections are to be determined. AOD extends this by the challenge of additionally finding the actors involved and visually marking them with bounding boxes.

In the area of surveillance, the underlying video footage suffers from problems related to the complexity of real world environments. For example, perspectives of the recording cameras, distances to relevant objects and the quality of individual video recordings vary strongly. Furthermore, a lack of information about the world coordinate system due to varying cameras and their positioning complicates the localization of moving objects in multi-camera systems. In the remainder of this paper, we discuss a method for detecting and capturing essential activities in videos. The aim is to design and implement a heterogeneous system for use in a future real-time environment. For this purpose, all components can act autonomously and make their data available via a central entity. Communication between the instances is realized via standardized interfaces (JSON, XML).

5 Workflow of Our Method

Our approach to the task was to detect the requested objects (vehicle, person) in the video footage by incorporating the bounding box coordinates for activity detection. The heuristic system environment described in Section 1 is used to determine partial and final results of that task. For Activity Detection the framework *Detectron* described under Section 2.3 was repeatedly used. The provided video material was segregated into individual frames. Those frames were generated using *OpenCV* being stored on a web servers file system, while maintaining references and their locations in a database.

Subsequently, the bounding boxes of all relevant objects in the individual keyframes were determined by means of *Detectron*. For performance reasons, only every 12th frame was used for this purpose. In order to derive activities from the objects determined with *Detectron*, it was necessary to link them statistically across the individual frames. By using the bounding box coordinates, the retrieved objects (car, bicycle, person, motorcycle, truck and bus) were collected by employing a self-developed tracking algorithm. In order to determine a possible temporal dependency, we used the center of individual bounding boxes of all objects and determined their change in location in relation to the previous and subsequent frames. In order to track those objects appropriately across multiple subsequent frames, we assigned an interrims marker. The object type can be passed to the algorithm as a parameter, whereafter we calculated the Euclidean Distance between the frames accordingly. The number of previous frames is another relevant parameter that represents the *history depth* and needs to be adjusted to slow or fast moving objects. The objects captured by this method are then assigned with a unique tracking ID in the database.

In a subsequent step, the bounding boxes of all objects involved in a scene are retrieved in chronological order via database queries. In addition, motion paths (tracklets) are determined for all objects. The angle changes of the tracklet components are accumulated and the aggregated movement patterns are derived (*left turn, right turn, u turn*). Simple heuristics based on the temporal sequence and the overlapping of objects are utilized to determine further activities. All detected activities are transferred to the database while preserving any object relations. The final results are then extracted by using SQL queries and converted to the JSON export format. Due to the complete database-driven interprocess communication of several individual frameworks, it should be prospectively possible to determine activities in real-time.

5.1 Data Preparation and Data Processing

To generate a homogeneous subset, we use *OpenCV*, an open source library with algorithms for image processing, to extract the individual frames from the entire video dataset. Each image is provided with the original video title and an incremental ID. Then, in accordance with our developed architecture, it is stored in a memory accessible from the web server. Access is granted via our self-developed API while the meta information and references are efficiently stored in the database. We searched for objects in each individual frame using *Detectron*. During detection, we save the bounding box found in each keyframe defined by the (x,y)-coordinates of their upper left corner as well as its width and height. Since the detections are limited to single images and no tracking of each retrieved object over several images is inherently possible, we developed our own tracking algorithm.

5.1.1 Retrieval of Object Data for Tracking

Using the *Detectron* framework, all frames are examined for the presence of objects. Those are classified based on the COCO dataset (Lin et al., 2014), shown in examples at Section 2.1.

After the extraction of frames, the bounding boxes of several object types are determined using *Detectron* and stored in the database for each frame individually. Figure 11 shows an exemplary frame. All objects contained in this frame are stored by the coordinates of their bounding boxes. For further tracking, the center of the enclosed area is calculated for each bounding box.

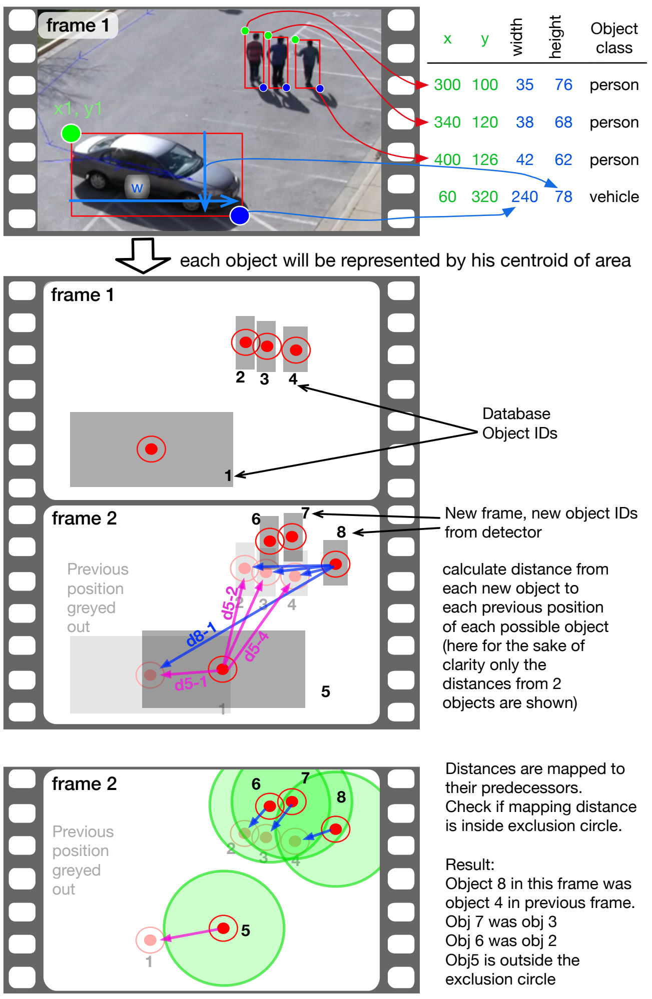

frame 1
x y width height Object class
300 100 35 76 person
340 120 38 68 person
400 126 42 62 person
60 320 240 78 vehicle
x1, y1
w
each object will be represented by his centroid of area
frame 1
2 3 4
1
Database Object IDs
frame 2
6 7 8
New frame, new object IDs from detector
Previous position greyed out
d5-2
d8-1
d5-4
d5-1
2 3 4
1
5
calculate distance from each new object to each previous position of each possible object (here for the sake of clarity only the distances from 2 objects are shown)
frame 2
6 7 8
Previous position greyed out
2 3 4
5
1
Distances are mapped to their predecessors. Check if mapping distance is inside exclusion circle.
Result: Object 8 in this frame was object 4 in previous frame. Obj 7 was obj 3 Obj 6 was obj 2 Obj5 is outside the exclusion circle

Figure 11: Schema to retrieve the bounding box of objects and to assign subsequent temporal positions computed by our tracking method.

We create quasistatic images, by splitting the videos into frames. Retrieved bounding boxes are linked to their related frames whereas the object coordinates within one frame are not related to those of adjacent frames. The bounding box identification algorithm treats each frame individually without any reference to the previous one. This means that any information about the coordinates of the objects retrieved from *Detectron* are purely static. Consequently, the object IDs in the database table only represent an object in the list of objects of a single frame and are not transferable to another frame. Another identifier is assigned to the same object on the next frame. Coordinates naturally change differently depending on the object movement. Since the content of the frames of a video may change quiet fast (usually at rates of 25 to 30 frames per second), there is only a slight difference in the bounding box coordinates of adjacent frames. Thus, our assumption is that the reduction of frames to every twelfth still appears sufficient enough to properly reflect the movements of the objects.

The underlying concept is to buffer all coordinates of each object and compare them with the corresponding coordinates of other frames in near timeline history. The parameters required have been determined experimentally.

5.1.2 Computation of Temporal Connections

Currently, an object is only represented by its coordinates and type excluding any further semantic features. The object detector returns a set of predictions $\mathbb{V}_f = \{v_i\}$ for each frame and as a consequence ($\mathbb{V} = \{v_{f,j}\}$) for all frames. Each individual prediction v of a specific point consists of:

$$v_{f,j,\beta} = (f, x_1 j, \beta, y_{1\,j,\beta}, w_{j,\beta}, h_{j,\beta}) \tag{16}$$

where: j = within frame-object-index,
f = frame index in given context,
(x_1, y_1, w, h) = coordinates for boundingbox β.

All predictions in a frame are given by the set of all $\mathbb{Q}_j$ with m as number of objects in the frame:

$$\mathbb{R} = \bigcup_{j=1..m} \mathbb{Q}_j. \tag{17}$$

All predictions in frame 1 are represented by $\mathbb{R}_1$; all predictions in frame 2 by $\mathbb{R}_2$, and so on.

For an object represented by the prediction set $\mathbb{Q}$, the most suitable object in the previous frame is searched for. The Euclidean distances in the 2D space of the video from the coordinates of an object to the related coordinates of all object detections in the previous frame $(f-1)$ are calculated as follows:

$$d_i = \sqrt{\left(x_{f,i} - x_{f-1,i}\right)^2 + \left(y_{f,i} - y_{f-1,i}\right)^2} \tag{18}$$

where: x, y = center of area coordinates of the bounding box,
i = object index from previous frame, equivalent to $i = j_{f-1}$.

All distances from one specific object to all objects of a previous frame form the distance set $\mathbb{E}_i = \{d_1, d_2, ..., d_m,\}$ with m as a number of objects in the previous frame. The resulting set $\mathbb{E}$ now contains all distances *of one object* to any objects in the previous frame.

We further determine the minimum $M = min(\mathbb{E})$ within the set $\mathbb{E}$. These minima of all eligible objects are compared to a fixed threshold ϵ (*"exclusion circle"*). If the current minimum deceeds the threshold, the object is considered to be found again. This results in the parameter ϵ forming a robust *feedback exclusion circle* around the coordinates. If an object disappears and another object appears in another position in the following frame, the exclusion circle prevents the new object from being assigned to the already existing object that has just disappeared.

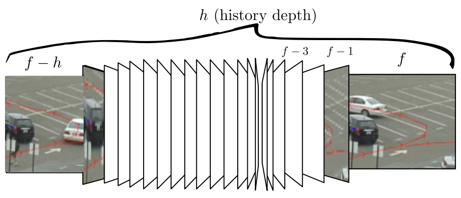


Figure 12: Search for matching objects in previous frames.

The described procedure is iterated *for each object* in each frame. Furthermore, the distance sets $\mathbb{E}$ are calculated for frames even further into the past being specified by a parameter h (*history depth*) as shown in Figure 12.

In order to cope with temporal hidden or occluded objects, we assume the following prerequisites for an appropriate retrieval:

- The occlusion duration remains within the historical search range h: the object is detected again before the *history depth* parameter limit is reached.
- The covert object does not move out of the exclusion circle during the occlusion, defined by the parameter *exclusion circle* ϵ.

5.1.3 Tracking Results and storage

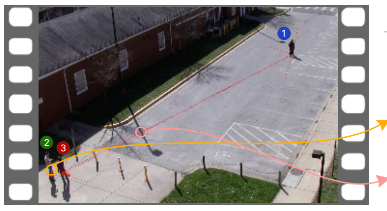
The objects in the individual frames are tagged with a frame-spanning identifier and are stored together with a unique ID in the database.

5.2 Data analysis

Regarding the ActEV task, we participated in the "AOD 1.A evaluation" and "AOD reference segmentation task" subtasks. Therein, the following activities needed to detected:

- "Closing"
- "Closing_trunk"
- "Entering"
- "Exiting"
- "Loading"
- "Open_Trunk"
- "Opening"
- "Transport_HeavyCarry"
- "Unloading"
- "Vehicle_turning_left"
- "Vehicle_turning_right"
- "Vehicle_u_turn"

Since we did not obtain detailed definitions regarding the required activities at calculation time, we did not impose any time-constraints for determining the activities. For the detection of the activities *Closing*, *Closing_trunk*, *Entering*, *Exiting*, *Loading*, *Open_Trunk* and *Opening*, we applied simple heuristics based on the results of the tracking algorithm. Here, we take a spatial resize of the extracted bounding boxes



Object ID	Length Vector 1	Length Vector 2	Cluster Vector 1	Cluster Vector 2
1	20.62	25.46	core	core
1	27.31	26	core	core
1	20	25.5	core	core
1	25.32	33.54	core	core
1	26.4	20	core	core
2	1	4.47	core	core
2	13.15	3.16	core	core
2	31.02	38.64	core	boundary
2	4	5.1	core	core
2	1	3.61	core	core
3	2.24	4.24	core	core
3	10.82	11.4	core	core
3	1216.44	1235.73	noise	noise
3	5.83	5	core	core
3	6.08	6.08	core	core

Figure 13: Filtering of outliers.

or an overlap with other objects into account. For example, if a person's bounding box is located near the bounding box of a car, an enlargement of the bounding box of the car within this constellation can be inferred to as the activity *Open_Trunk*. To determine such scenarios, we create database tables where the size, center, and overlaps of the bounding box are stored for each keyframe. With reference to the increasing keyframe ID, the activities mentioned can then be determined and stored subsequently.

5.2.1 Motion Pattern Detection

The cross-frame object identifier allows the combination of single coordinates into vectors. The analysis of vectors over time enable us to draw conclusions about the directions and actions of the movements of an object. Hence, we generate the corresponding direction vectors and magnitudes for all traceable objects from all successive bounding box centers. Invoked errors due to the processing constraint of every twelfth frame are removed by an outlier detection resulting from the temporal traces of the bounding boxes by using the DBSCAN (Density-Based Spatial Clustering of Applications) clustering method in the *Turi Create* software. The resulting vector consists of the several values: *Core*, *Boundary* and *Noise*. Salient points are inferred to if there exists a large set of vectors with a similar magnitude. Boundary points are located within a distance radius around a core point, but with a lack of related neighbors. Vectors with magnitudes larger than the defined distance radius are classified as noise. The subdivision introduced by DBSCAN is stored in the database for each traced object. Figure 13 shows three tracked persons, whereby the object with ID 3 shows two false vector magnitudes (two long red arrows) computed by the tracking procedure. With the use of DBSCAN, these entries could be classified as noise points. As for object ID 2, a vector magnitude has been defined as boundary point. Such boundary points regularly occur when the bounding box and the correlating center point alters due to movements or changes in the shape of an object (e.g. arm lifting). In the end, we only make use the vector magnitudes classified as *Core* and *Boundary* for activity detection. Consequently, entries categorized with *noise* are ignored.

To determine the activities "Vehicle_turning_left", "Vehicle_turning_right" and "Vehicle_u_turn" the direction vectors were accumulated over consecutive frames starting from a

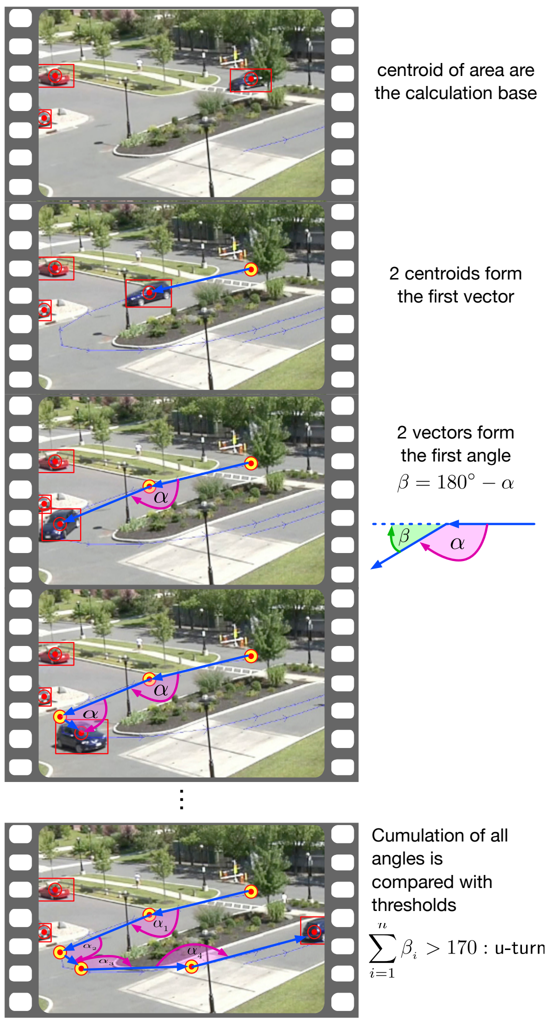

centroid of area are the calculation base
2 centroids form the first vector
2 vectors form the first angle
$\beta = 180° - \alpha$
α
β
α
α_1
α_2
α_3
α_4
Cumulation of all angles is compared with thresholds
$\sum_{i=1}^{n} \beta_i > 170$: u-turn

Figure 14: Cumulating the vector angles to derive motion paths.

determined starting point. If a defined threshold value is exceeded, we infer that a corresponding motion pattern is present. Thus a U-turn is assumed if the vectors accumulate over time to form an angle of 170 degree. A motion pattern is considered as completed as soon as the direction of the vector moves contrary to the current direction or the last tracked frame of an object has been reached. The minimum angles we defined for left and right motion are $\pm 40°$. To detect a left or right turn, at least two consecutive left or right vectors must be present. For a turning movement, the minimum cumulative angular value was set to $\pm 170°$. A "vehicle_u_turn" can only be triggered if this minimum sum of angles is exceeded and at least two left- or right-oriented vectors have occurred before. Figure 14 depicts the summation of the angle vectors and the resulting motion detection.

5.2.2 Analysis Using Simple Heuristics

In order to detect the activities *Closing Trunk* and *Opening Trunk*, we simply observed the area of each bounding box.

We proceed according to the heuristic that, if a person appears next to a vehicle and the bounding box enlarges or reduces in size over time, it is either *Closing Trunk* or *Opening Trunk*. A similar heuristic is used for *Closing* and *Opening*. Here, we examine the change of the area analogously, but only invoke a detection, if a person is situated closely to the front of the vehicle. To compute the activities *Entering* and *Exiting*, we take the pure existence of objects in the image over time into account. We assume that, if a person is found in front of a vehicle and suddenly disappears over time, the person has probably entered another object. Consequently, we assess that the activity *Exiting* corresponds to the sudden appearance of an object in the area of another object within the scenery.

In the context of *Transport HeavyCarry*, we focus our attention to non-person objects. If the non-person objects happen to occur in front of a person and move in harmony with that person, we trigger the detection. With regard to *Loading* and *Unloading*, we assume that a person's bounding box changes within the area if something is picked up or put down. Thus, the sudden occurrence of an enlargement (*Loading*) and a reduction (*Unloading*) is treated as a strong indicator of the event. For each heuristic, we wrote our own SQL queries and integrated the necessary business logic into the database. Thus, we are capable to generate all queries and the result files within milliseconds.

5.2.3 Reference Temporal Segmentation

"AOD reference temporal segmentation" is a reverse task aiming to determine the corresponding activity for predefined time sequences in the video footage. This allows us to focus solely on activity classification rather than performing a full segmentation and instance classification workflow. Due to our system architecture, all activities have been detected beforehand being already stored in the database. Therefor, to solve this task just requires an appropriate querying of the database with filters containing the respective periods of time followed by a conversion to the requested submission format.

6 Results and Future Work in Activity Event Detection

The results of our approach are shown in Figure 16. The yellow bars indicate the performance of the (missed) detected objects at 0.5 false alarms per minute. The blue bars indicate the performance of the (missed) detection actions within the AD task for sole activities; the red bars for activity and objects, both at an false alarm rate of 1 per minute.

The distribution of the yellow bars shows that the detection of the objects involved into activities gives much room for improvements. However, we already can rely on the detection of vehicles and their tracking at specific activities like *Vehicle_left_turn, Vehicle_right_turn, Vehicle_u_turn*.

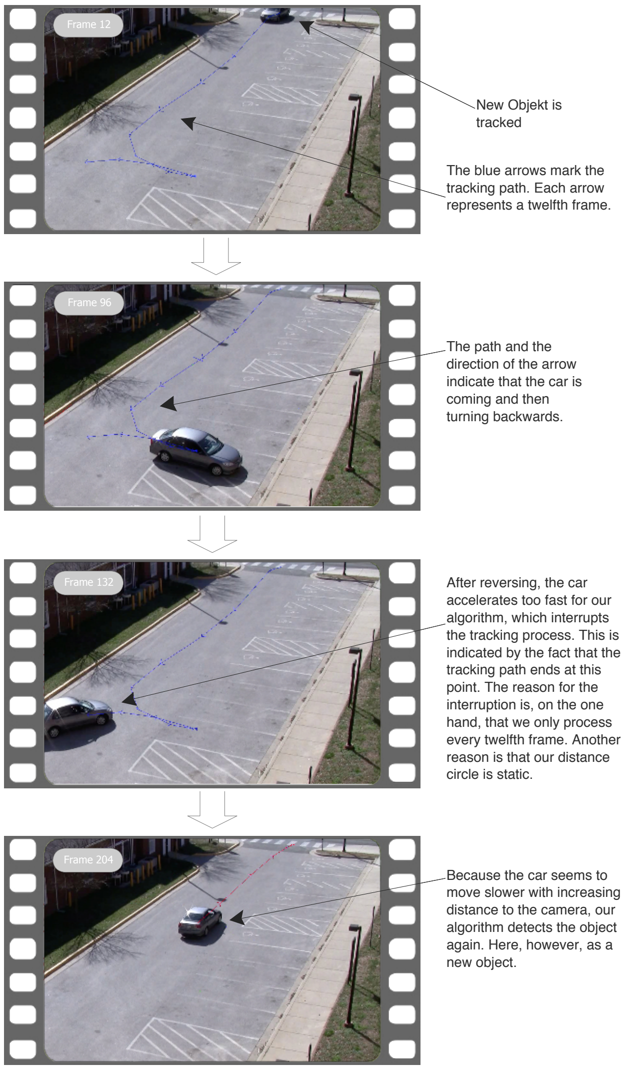

Frame 12
New Objekt is tracked
The blue arrows mark the tracking path. Each arrow represents a twelfth frame.
Frame 96
The path and the direction of the arrow indicate that the car is coming and then turning backwards.
Frame 132
After reversing, the car accelerates too fast for our algorithm, which interrupts the tracking process. This is indicated by the fact that the tracking path ends at this point. The reason for the interruption is, on the one hand, that we only process every twelfth frame. Another reason is that our distance circle is static.
Frame 204
Because the car seems to move slower with increasing distance to the camera, our algorithm detects the object again. Here, however, as a new object.

Figure 15: Tracking the loss of an object.

However, we alreay completely failed to detect the events of *Closing, Opening, Transport_HeavyCarry*. Here, we refer to some unknown activity constraints in the definition of the events during the submission periods in the first subtasks that has already been cleared afterwards in the next evaluation phases. Unfortunately, our current implementation lacks a detection of the class door or the constraint that heavy carry employs the detection of object larger than 1.5 times of the human body.

However, in all other events—ranging at best at around 0.62 miss rate at 1 false alarm per minute at the activity *Unloading* raises over *Open Trunk* (0.72) and others to *Entering* (0.95) and *Exiting* (0.98)—our simple engineered heuristics already show mediocre but reasonable results that can be extended in the future with optimized parameters and more sophisticated machine learning methods.

Detection and tracking are currently performed on the 2D projection plane of the video. However, typical surveil-

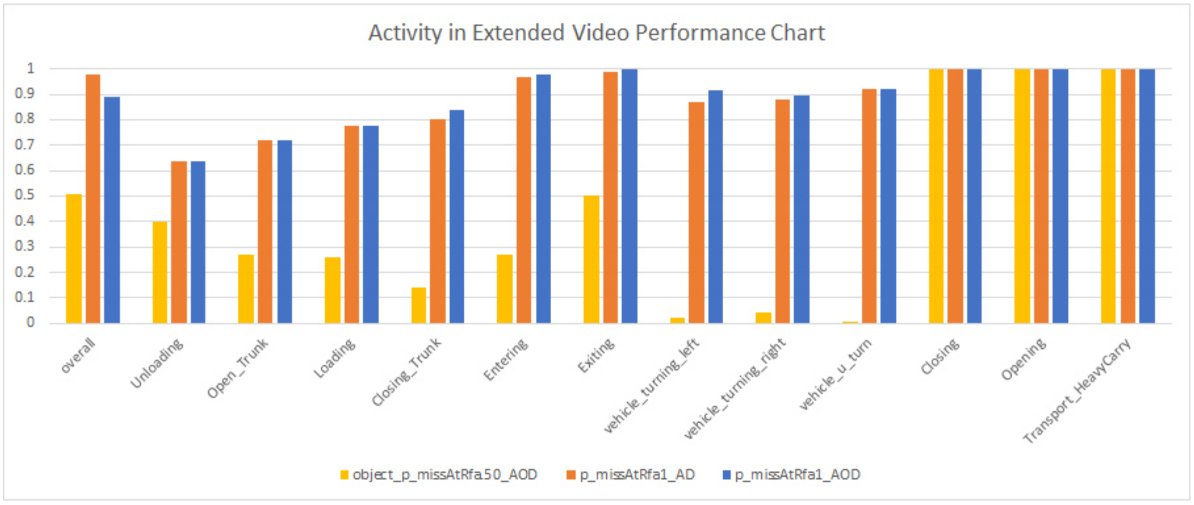


Figure 16: Results of our approach to ActEV subtasks AD (Activity Detection) and AOD (Activity Object Detection).

lance videos are generally captured from a raised position and show a slightly distorted view of the real world. This also effects every motion present in the footage. Objects that appear further away from the camera, seem to move more slowly than objects closer to the camera even when moving at a constant speed. Our algorithm is currently not taking any perspective corrections into account. If an object moves, the algorithm detects the minimum distance object in a *constant exclusion circle* ϵ area surrounding the coordinates of the last position. When the object retains a constant speed, a perspective-dependent should vary the size the near environment that is searched for specific items in the foreground.

The parameter ϵ additionally depends on the resolution of the underlying video data. For higher-resolution videos, motion includes larger pixel distances. The parameter must also be adjusted accordingly. For performance reasons, we only used every twelfth frame for object classification. In this respect, only the information of every twelfth frame is available for object tracking. Tracking losses or superimposition to other objects occur Figure 15, especially when objects are moving quite fast or when many objects are involved within a scene.

Another problem appears in video recordings with camera movements invoked by outer forces like weather conditions resulting from strong wind or storms. This results in unexpected movements of detected objects which might also infer errors in object tracking. Future approaches could benefit from a hardware or software-based stabilization of the video sequences as well as an incorporating of object-specific features within the motion and tracking analysis.

Activity detection is currently based on engineering methods. These are limited to the evaluation of relatively simple spatial object movements. Frameworks like *OpenPose* (Cao et al., 2016) enable to analyze the posture of persons and limbs more accurately. In the future, we incorporate and feed temporal characteristics of extracted *OpenPose* features into an activity classifier by making use of own deep learning models using synthetically generated ground truth.

Acknowledgments

The European Union and the European Social Fund for Germany partially funded this research. This work was also partially funded by the German Federal Ministry of Education and Research in the program of Entrepreneurial Regions InnoProfileTransfer in the project group localizeIT (funding code 03IPT608X). Program material in sections 1–3 is copyrighted by BBC. We want to thank all the organizers of these tasks, especially George Awad and Afzal Godil, for the hard work they put into the annotation, evaluation and organization of these challenges.

References

Awad, G., Kraaij, W., Over, P., and Satoh, S.: Instance search retrospective with focus on TRECVID, International Journal of Multimedia Information Retrieval, 6, 1–29, 2017.

Awad, G., Butt, A., Curtis, K., Lee, Y. L., Fiscus, J., Godil, A., Delgado, A., Smeaton, A. F., Graham, Y., Kraaij, W., Quénot, G., Magalhaes, J., Semedo, D., and Blasi, S.: TRECVID 2018: Benchmarking Video Activity Detection, Video Captioning and Matching, Video Storytelling Linking and Video Search, in: Proceedings of TRECVID 2018, NIST, USA, 2018.

Cao, Z., Simon, T., Wei, S., and Sheikh, Y.: Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields, CoRR, abs/1611.08050, http://arxiv.org/abs/1611.08050, 2016.

Chen, X., Ji, Z., Fan, Y., and Zhan, Y.: Restful API Architecture Based on Laravel Framework, Journal of Physics: Conference Series, 910, 012 016, 2017.

Feng, S.: Color Thief, https://github.com/fengsp/color-thief-py, viewed: 2018-10-14, 2016.

Geitgey, A. and Nazario, J.: Face Recognition, https://github.com/ageitgey/face_recognition, 2017.

Girshick, R., Radosavovic, I., Gkioxari, G., Dollár, P., and He, K.: Detectron, https://github.com/facebookresearch/detectron, 2018.

Jalal, A., Kim, Y.-H., Kim, Y.-J., Kamal, S., and Kim, D.: Robust human activity recognition from depth video using spatiotemporal multi-fused features, Pattern Recognition, 61, 295 – 308, doi:https://doi.org/10.1016/j.patcog.2016.08.003, http://www.sciencedirect.com/science/article/pii/S0031320316302126, 2017.

Kahl, S., Roschke, C., Rickert, M., Hussein, H., Manthey, R., Heinzig, M., and D. Kowerko, M. R.: Technische Universität Chemnitz at TRECVID Instance Search 2016, in: Proceedings of TRECVID Workshop, 2016.

Lillo, I., Niebles, J. C., and Soto, A.: Sparse composition of body poses and atomic actions for human activity recognition in RGB-D videos, Image and Vision Computing, 59, 63 – 75, doi:https://doi.org/10.1016/j.imavis.2016.11.004, http://www.sciencedirect.com/science/article/pii/S0262885616301949, 2017.

Lin, T., Maire, M., Belongie, S. J., Bourdev, L. D., Girshick, R. B., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L.: Microsoft COCO: Common Objects in Context, CoRR, abs/1405.0312, http://arxiv.org/abs/1405.0312, 2014.

Lin, T.-Y., Maire, M., Belongie, S., Bourdev, L., Girshick, R., Hays, J., Perona, P., Ramanan, D., Zitnick, C. L., and Dollár, P.: Microsoft COCO: Common Objects in Context, ArXiv e-prints, 2014.

Oh, S., Hoogs, A., Perera, A., Cuntoor, N., Chen, C.-C., Lee, J. T., Mukherjee, S., Aggarwal, J. K., Lee, H., Davis, L., Swears, E., Wang, X., Ji, Q., Reddy, K., Shah, M., Vondrick, C., Pirsiavash, H., Ramanan, D., Yuen, J., Torralba, A., Song, B., Fong, A., Roy-Chowdhury, A., and Desai, M.: A large-scale benchmark dataset for event recognition in surveillance video, in: CVPR 2011, IEEE, doi:10.1109/cvpr.2011.5995586, https://doi.org/10.1109/cvpr.2011.5995586, 2011.

Patterson, G. and Hays, J.: SUN Attribute Database: Discovering, Annotating, and Recognizing Scene Attributes, in: Proceeding of the 25th Conference on Computer Vision and Pattern Recognition (CVPR), 2012.

Redmon, J. and Farhadi, A.: YOLO9000: Better, Faster, Stronger, arXiv.org, p. arXiv:1612.08242, http://arxiv.org/abs/1612.08242v1, 2016.

Redmon, J., Divvala, S. K., Girshick, R. B., and Farhadi, A.: You Only Look Once - Unified, Real-Time Object Detection., CVPR, pp. 779–788, doi:10.1109/CVPR.2016.91, http://ieeexplore.ieee.org/document/7780460/, 2016.

Ritter, M., Heinzig, M., Herms, R., Kahl, S., Richter, D., Manthey, R., and Eibl, M.: Technische Universität Chemnitz at TRECVID Instance Search 2014, in: Proceedings of TRECVID Workshop, Orlando, Florida, USA, 2014.

Ritter, M., Rickert, M., Juturu-Chenchu, L., Kahl, S., Herms, R., Hussein, H., Heinzig, M., Manthey, R., Richter, D., Bahr, G. S., and Eibl, M.: Technische Universität Chemnitz at TRECVID Instance Search 2015, in: Proceedings of TRECVID Workshop, Gaithersburg, Maryland, USA, 2015.

Satyanarayanan, M., Ludwiczuk, B., and Amos, B.: OpenFace: A general-purpose face recognition library with mobile applications, https://cmusatyalab.github.io/openface/.

Schroff, F., Kalenichenko, D., and Philbin, J.: FaceNet: A Unified Embedding for Face Recognition and Clustering, ArXiv e-prints, 2015.

Simons, D. J. and Chabris, C. F.: Gorillas in Our Midst: Sustained Inattentional Blindness for Dynamic Events, Perception, 28, 1059–1074, doi:10.1068/p281059, 1999.

Sridhar, K., Larsson, G., Nation, Z., Roseman, T., Chhabra, S., Giloh, I., de Oliveira Carvalho, E. F., Joshi, S., Jong, N., Idrissi, M., and Gnanachandran, A.: Turi Create, https://github.com/apple/turicreate, viewed: 2018-10-12, 2018.

Zhou, B., Lapedriza, A., Khosla, A., Oliva, A., and Torralba, A.: Places: A 10 million Image Database for Scene Recognition, IEEE Transactions on Pattern Analysis and Machine Intelligence, 2017.

Zhuang, N., Yusufu, T., Ye, J., and Hua, K. A.: Group Activity Recognition with Differential Recurrent Convolutional Neural Networks, in: 2017 12th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2017), IEEE, doi:10.1109/fg.2017.70, https://doi.org/10.1109/fg.2017.70, 2017.