Deep Learning and Approach for Tracking People's Movements in a Video

Jemai Bornia
Robotics, Computing and Complex Systems
National Engineering School of Tunis
University Tunis El Manar
Tunis, Tunisia
bornia.jemai@gmail.com

Ali Frihida
Robotics, Computing and Complex Systems
National Engineering School of Tunis
University Tunis El Manar
Tunis, Tunisia
ali.frihida@enit.utm.tn

Olivier Debauche
Faculty of Engineering - ILIA
Infortech Institut
University of Mons
Mons, Belgium
orcid:0000-0003-4711-2694

Sidi Ahmed Mahmoudi
Faculty of Engineering - ILIA
Infortech Institut
University of Mons
Mons, Belgium

Pierre Manneback
Faculty of Engineering - ILIA
Infortech Institut
University of Mons
Mons, Belgium

***Abstract*—With the advent of the digital age and more specifically videos, a huge amount of data is produced every day such as television archiving, video surveillance, etc. Faced with the need to keep control over this content, in terms of data analysis, classification, accurate AI (Artificial Intelligence) algorithms are required to perform this task efficiently and quickly. In this paper, we propose an approach for movement analysis from video sequences using deep learning technologies. The proposed approach splits video in set of images, detects objects/entities present in these images and stores their descriptions into a standard XML file. As result, we provide a Deep Learning algorithm using TensorFlow for tracking motion and animated entities in video sequences.**

***Index Terms*—object tacking, motion tracking, TensorFlow, deep learning, video**

I. Introduction

Lately, there has been an increase in video data that has been noticed [1] [2] because it generally gives more information in a simple way but the manners used to exploit them still human supervised. This limits their benefits as it is expected in many application domains. It also comes down to technological advancements such as cameras, storage devices and scanners. in this case the multimedia community has challenged to seek and find more effective methods of video analysis such as the classification concepts [3], [4], [5] and the recognition of actions [6], [7]. These methods are increasingly important to retrieve [8], [9], [10] or to summarize videos for efficient navigation [11]. Exploiting large video databases is becoming a very active field of research [12] [13]. The published works present algorithms or tools developed while Deep Learning technology and TensorFlow are considered to be a very effective tool for video processing.

Deep Learning [14] represents a set of learning methods that allow to model data with complex neural architectures combining different layers and non-linear transformations. The main components of a Deep Learning architecture are the perceptrons, which are combined to form several layers and therefore a deep neural networks (DNN). However, there are difficulties when looking for a representation of visual content that reflects the semantics of the video. Extracting semantic content directly from raw video data is difficult because video is a temporal sequence of images, in explicit relation to their semantic content. In this context, the Google's Open Source Machine Learning Framework for dataflow programming, called TensorFlow, across a range of tasks [15]. Nodes in the graph represent mathematical operations, while the graph edges represent the multi-dimensional data arrays (tensors) communicating between them. Tensors are just multidimensional arrays, an extension of two dimensional tables to data with a higher dimension. There are many features of Tensorflow, which makes it appropriate for Deep Learning [16].

In this paper, we present an approach that allows to detect objects in video images (frames). Our approach starts by data annotation that consists on providing label (type of object and coordinates) to each object in the image, the labelImg package [1]. For example, if an object and a person exist in one image, an XML file is attributed to each one. XML files are first converted to CSV format, then converted to TFRecord format in order to use TensorFlow packages. As result, two new files are generated "train.record" for training and "test.record" for test. These two files allow to train and evaluate the tracking model before moving the test phase to track real movements.

The remainder of the paper is organized as follows: the background of Deep Learning algorithms are presented in section 2. In section 3, we present some existing works in the field of motion detection tracking; we present our contribution and experimental results in section 4. Finally, the conclusion and future work are drawn in section 5.

[1]labelimg. https://github.com/tzutalin/labelImg

II. Background

A. General presentation of TensorFlow

TensorFlow [17] is a software library or framework, designed by the Google team for developing and training machine learning and deep learning models in an easiest manner. It combines the computational algebra of optimization techniques for easy calculation of many mathematical expressions. The most important features of TensorFlow are: (1) it allows to define, optimize and calculate mathematical expressions easily with the help of multi-dimensional arrays called tensors ; (2) it provides a programming support of deep neural networks and machine learning techniques ; (3) it offers the possibility to use the high computation power of GPU [18] [19] ; (4) it provides a optimization and efficient management of memory and the data used. Moreover, TensorFlow is well-documented and includes plenty of machine learning libraries. It offers a few important functionalities, methods and includes a variety of machine learning and deep learning algorithms. TensorFlow can train and run deep neural networks for handwritten digit classification, image recognition, word embedding, and be used for the creation of various sequence models.

B. General presentation of deep learning

Deep learning is a set of learning methods attempting to model data with complex architectures combining different non-linear transformations [20]. The elementary bricks of deep learning are the neural networks that are combined to form the deep neural networks (DNN). These techniques have enabled significant progress in the fields of sound and image processing, including facial recognition, speech recognition, computer vision, automated language processing, text classification (for example spam recognition). Potential applications are very numerous. A spectacularly example is the AlphaGo program, which learned to play the go game by the deep learning method, and beat the world champion in 2016. There exist several types of architectures for neural networks (NN) [21]:

- The Multi-layer Perceptron, that are the oldest and simplest ones
- The Convolutional Neural Networks (CNN), were inspired by the work of Hubel and Wiesel on the visual cortex. Like MLPs (Multi-layer Perceptron, Perceptron Multi layers), these ConvNets consist of several hidden layers. They are mainly used for image classification and extraction of visual features.
- The recurrent neural networks (RNN), used for sequential data such as text or times series

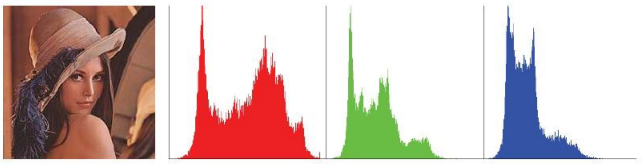
III. Related Work

Video analysis presents a fundamental computer vision problem, which has received a lot of attention. We briefly present the main existing motion and object extraction algorithms in video sequences.

A. Object and person detection from video

The detection of objects in a video first passes through the detection within an image. Several approaches have been introduced over last years and will be detailed in this section. Some methods rely on the color contained in the images by means of histograms. Others use the shape of the objects or the textures contained in the image. More recently, Neural networks have proven their ability to perform detection. Convolutional Neural networks (CNNs) are particularly suitable for image processing. In order to be able to find an object in an image, it is first necessary to obtain the descriptors (feature) of this object. Then the objective is to compare the descriptors of a given image and to conclude if the resemblance is sufficient to affirm that the object in question is in the image. The set of algorithms described below makes it possible to extract the characteristics of an image. The descriptions below focus on feature extraction except for deep neural networks that have a an additional classification step in their architecture.

1) Color histogram: Most images used today are recorded in RGB. They have three components for each pixel representing the intensity in Red Green and Blue color. Color histograms consist of an inventory of colors in the image [22]. The three compound diagrams are the descriptors of the image in input and can be used to perform object detection. They represent the number of pixels contained in the image for a precise intensity of color. Several works have been done to study the similarity of images [22].



(a) Lenna couleur (b) Histogramme couleur

Fig. 1. RGB color histograms for an image

2) R-CNN: Going through an entire image by moving the convolution kernel was far too resource intensive. This is why an algorithm based on the reduction of the number of pixels to be analyzed has been introduced: the R-CNN (Region-based Convolutional Neural Networks) [23]. It consists of the selection of about 2000 regions with color, texture and intensity as the criteria in order to estimate a higher probability of presence of a desired object. This algorithm is broken down into three main steps:

1) The selection of regions of interest.
2) The application of the convolution layers on the proposed regions.
3) The optimization of the position thanks to a regression algorithm on the boxes delimiting the detected objects.

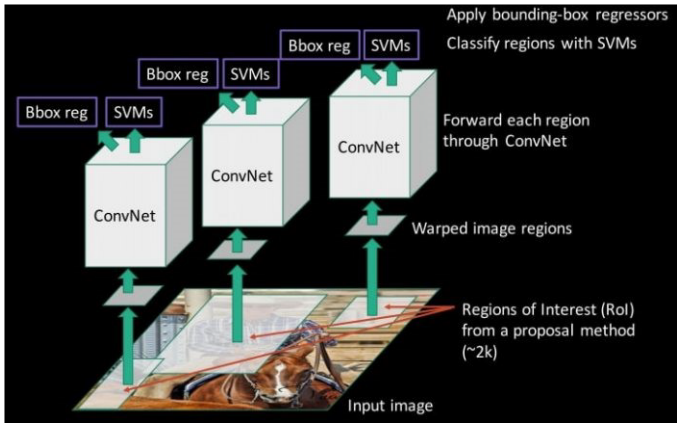


Fig. 2. Application Convolutional Neural Networks to selected images areas [23]

3) Yolo (You Only Look Once): It is an image classification and object detection application based on a totally different approach [24]. This application uses a network of neurons to the complete image. This network divides the image into regions and predicts the boundaries (segments) and probabilities for each region. These segments are weighted by the predicted probabilities [2]. The goal is to provide the membership class of each segment detected in the image. Yolo has several advantages over existing classification work in the literature.

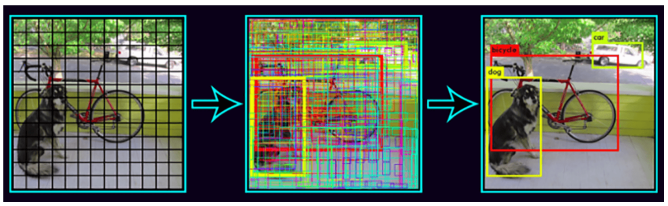


Fig. 3. Selection boxes in YOLO. source : https://pjreddie.com/darknet/yolo/

B. Algorithms of motion detection

Being able to determine the movement of person or object in a video is a challenge that requires the use several algorithms of computer vision and deep learning. For example, to determine the direction of movement of a vehicle, to detect a car ghost (moving against the direction) or to detect the fall of certain objects. In this context, object detection algorithms are now performing well. However, they require access to important resources.

In literature, motion detection algorithms consist of three main steps: features detection, movement features tracking (optical flow computation) and noise elimination (static features removing).

1) Feature detection: The first step of this method is to detect features that are good to track, i.e. corners. This is a very efficient method thanks to its invariance to rotation, scaling, brightness and noise [25]. It is based on five steps: spatial derivatives computation, eigenvalues computation, maximum eigenvalue selection, small eigenvalues removing and eigenvalues selection, all of them briefly described below:

(a) **Spatial derivatives computation**: this step consists of computing the matrix G of spatial derivatives for each pixel. Using the Eq. (2). This 4-element matrix (2×2) is calculated with the spatial derivatives Ix, Iy computed using the Eq. (1).

$$\begin{cases} I_x(x,y) &= \frac{I(x+1,y)-I(x-1,y)}{2} \\ I_y(x,y) &= \frac{I(x,y+1)-I(x,y-1)}{2} \end{cases} \quad (1)$$

$$G = \begin{pmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{pmatrix} \quad (2)$$

(b) **Eigenvalues computation**: based on the matrix G, for each pixel we calculate two eigenvalues of which only the greater is kept.

(c) **Maximum eigenvalue selection**: once all the eigenvalues are calculated, the algorithm retrieves the maximum value.

(d) **Small eigenvalues removing**: the search for small eigenvalues is performed by comparing the eigenvalue of each pixel with the maximum eigenvalue. If this absolute value is lower that 5% of the maximum value, the pixel is Maximum eigenvalue selection: once all the eigenvalues are calculated, the algorithm retrieves the maximum value excluded.

(e) **Corner selection**: For each image area (of predefined size), the algorithm extracts one pixel having the largest eigenvalue. The selected pixels represent points of interest (the final corners).

2) Movement features tracking (optical flow computation): Once the corners are selected, we track them within the next frame using the optical flow technique [26] [27]. We exploit the Lucas-Kanade [25] algorithm for the optical flow estimation. As pointed out in the Background section, this method is well-known for its high efficiency, accuracy and robustness. The algorithm consists of seven steps that.

(a) Pyramid construction
(b) Pixels matching over levels
(c) Local gradient computation
(d) Iterative loop launch and temporal derivative computation
(e) Optical flow computation
(f) Estimation correction and end of the iterative loop
(g) Result propagation and end of the pyramid loop

IV. PROPOSED ALGORITHM FOR OBJECT DETECTION AND MOTION TRACING

A. Description of algorithm

The main goal of our approach is to create a method capable of extracting the movement from a video. The proposed

[2]https://pjreddie.com/darknet/yolo/

approach permits to extract the key frames of a video then. Before training models, we start by annotating data and video frames using labelImg package. If an object or person exist, an XML file is generated to each one. XML files that include the objects coordinates and labels are converted to CSV format than to TFRecord format in order to have an annotation compatible with Tensorflow. Once the annotation of video frames completed, two files are generated "train.record" for model training and "test.record" for model evaluation. The model is trained and evaluated at the end of each epoch within validation dataset (10% from the training dataset). The generated model is then evaluated within the test dataset in order to validate our results. In our case, the model is used for tracking the movements of persons or objects, present in the video, via python script that allows to plot the results a in a curve. What we have to do at rudimentary level is shown in Fig. 4.

B. Technological choices

In our approach, we use two main libraries: OpenCV and Pandas.

- **OpenCV:** This library is well adapted for image processing. The functions of reading and annotating images are very useful. In addition, this library offers an interesting API for tracking algorithms [28].

- **Pandas:** To ensure easy processing of the produced data (box coordinates ...), the pandas framework in python provides access to very practical analysis tools. It introduces the concept of DataFrame a structure designed to facilitate treatments of data [29].

C. Description of steps

the following figure illustrates the main steps of our algorithm

Our algorithms is composed of four steps: video segmentation and annotation: video segmentation and annotation, database division and format conversion, model training, test and evaluation.

1) video segmentation and annotation: the first step is video segmentation that provides a descriptive file including the present features of each video frame. In our case, each video frame is annotated using the labelImg package. This annotation consists of listing the coordinates of all the boxes containing an object or a person with the corresponding class number (Fig 6 and fig 8). Once all the video images annotated, a set of new XML files, one for each image is generated (fig 7 and fig 9).

2) Database division and format conversion: After generating of our annotations, the file is split into three subsets: training, validation and test datasets. These files are then converted to csv format firstly and to TFRecords format secondly in order to have a compatible format of annotations with Tensorflow and Yolo.

3) Model training and evaluation: Once our records files are ready, we are almost ready to train the model so we need to launch the script tarin.py and we should see a series of print outs. After training the model, we can follow it under a web page based on Jupyter notebook. This will create a local web page on the local computer at the address YourPCName: 6006, which can be displayed via a web browser. The TensorBoard page provides information and charts showing the progress of the training. One of the most important graphs is the loss graph, which shows the overall loss of the classifier over time.

4) Test and evaluation: The Test dataset provides the gold standard used to evaluate the model. It is only used once a model is completely trained (using the train and validation sets). The test set is generally well curated. It contains carefully sampled data that spans the various classes that the model would face, when used in the real world.

V. EXPERIMENTATAL RESULTS

Given an image or a video stream, an object detection model can identify which of a known set of objects might be present and provide information about their positions within the image. The algorithm allows to browse one video after another; each video is divided into images to facilitate the detection of the people appearing on each image, then the movements of each. to apply algorithm we followed the following steps:

(a) Organize our workspace and training files.
(b) Prepare and annotate image datasets.
(c) Generate *tf* records from such datasets.
(d) Configure a simple training pipeline.
(e) Train a model and track its progress.
(f) Export the resulting model and use it to detect people.

To launch our algorithm, we are testing with two videos from the "UCF101 Train TestSplit-Recognition Task" database as Table 1 shows. Training the model is as simple as executing the following code. We just need to give it:

(a) model_main.py which runs the training process
(b) pipeline_config_path=Path/to/config/file/model.config
(c) model_dir= Path/to/training/

TABLE I
VIDEO USED

Video	Size	Number of images	Number of key frames
1	69.3 MB	74	12
2	73.2 MB	81	15

A. Creating Tensorflow records

Tensorflow accepts the data as TFRecords data.record, which represent a binary file that runs fast with low memory usage. It contains all the images and labels in one file. In our case, we will have two TFRecords; one for testing and

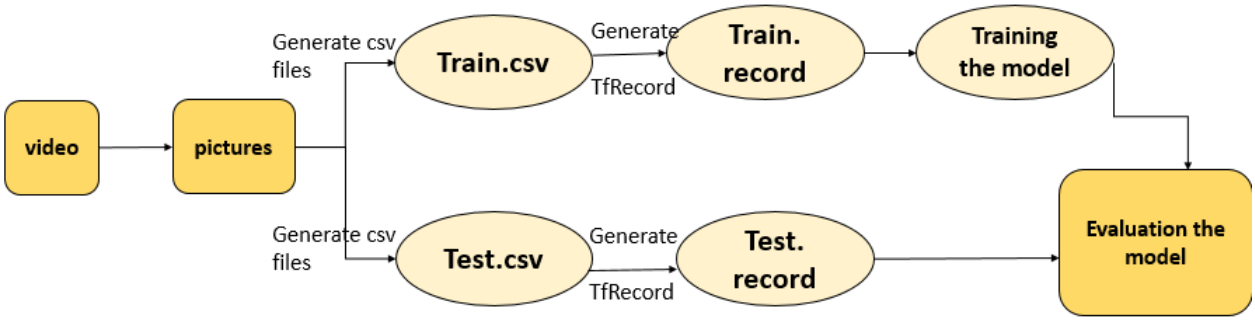


Fig. 4. Basic Flow Diagram

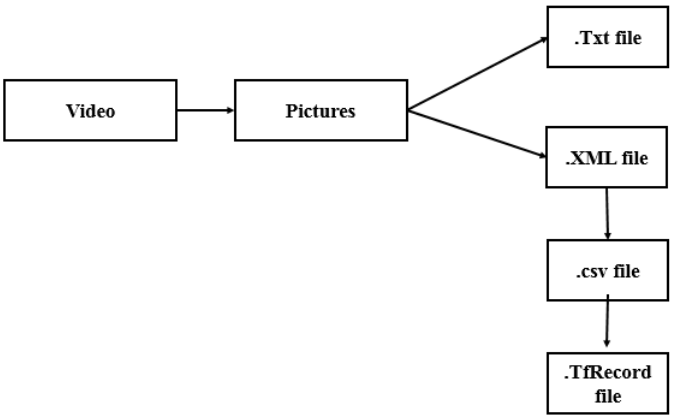


Fig. 5. basic steps diagram

Fig. 6. Example of output of person detector

another for training. To make this work, we need to make sure that:

(a) The CSVs file names is matched:train_labels.csv and test_labels.csv
(b) Current directory is object_detection/models/research
(c) Check if the path to data/ directory is the same as data_base_url below

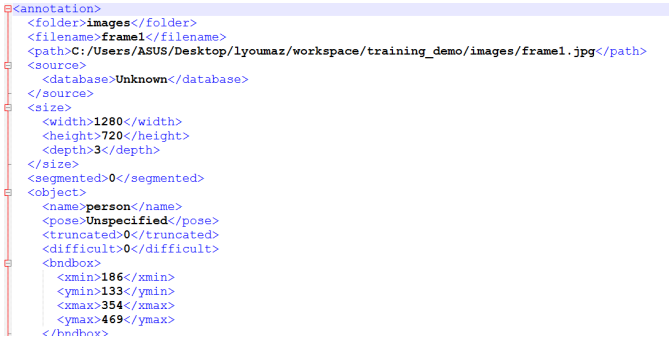


Fig. 7. XML file generated for the previous image

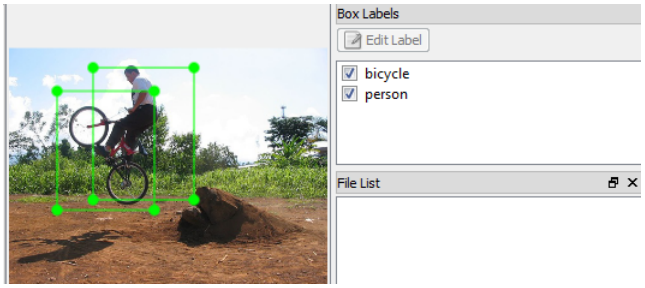


Fig. 8. Example of output of object detector

B. Training the model and evaluation

Once our records files are ready, we are almost ready to train the model so we need to launch the script train.py and we should see a series of print outs similar to the one below.

In the Fig. 10, each step of the training reports the loss. It will start high and decrease as training progresses. For our training, it started around 2-3 and quickly dropped below the 0.5 mark. I recommend allowing your model to train until the loss consistently falls below 0.05, which can take about 30,000 steps or hours (depending on the power of your processor or graphics processor). The number of losses can be different if a different model is used. In addition, it depends on the objects you want to detect.

```
</size>
<segmented>0</segmented>
<object>
    <name>person</name>
    <pose>Left</pose>
    <truncated>0</truncated>
    <difficult>0</difficult>
    <bndbox>
        <xmin>135</xmin>
        <ymin>25</ymin>
        <xmax>236</xmax>
        <ymax>188</ymax>
    </bndbox>
</object>
<object>
    <name>bicycle</name>
    <pose>Left</pose>
    <truncated>0</truncated>
    <difficult>0</difficult>
    <bndbox>
        <xmin>95</xmin>
        <ymin>85</ymin>
        <xmax>232</xmax>
        <ymax>253</ymax>
    </bndbox>
```

Fig. 9. XML file generated for the previous image

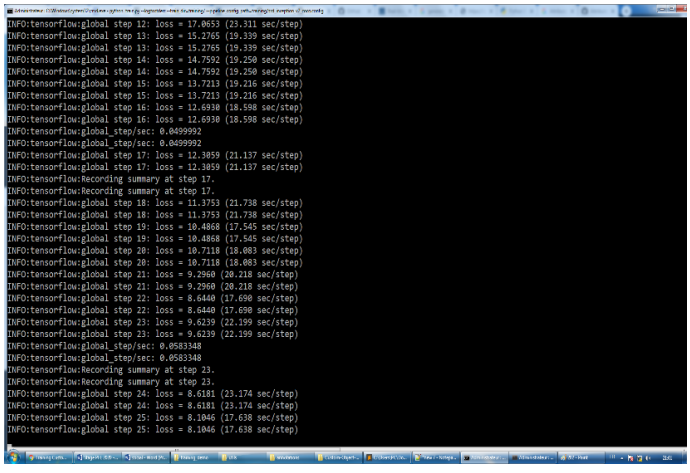


Fig. 10. Output of training the model

C. Testing and motion detection

The test dataset provides the gold standard used to evaluate the model. It is only used once a model is completely trained (using the train and validation sets). The test set is generally well curated. It contains carefully sampled data that spans the various classes that the model would face, when used in the real world. The following figures show the people detected by the model and the location of a bounding box that contains each person.

Fig. 11. person detection in the test part

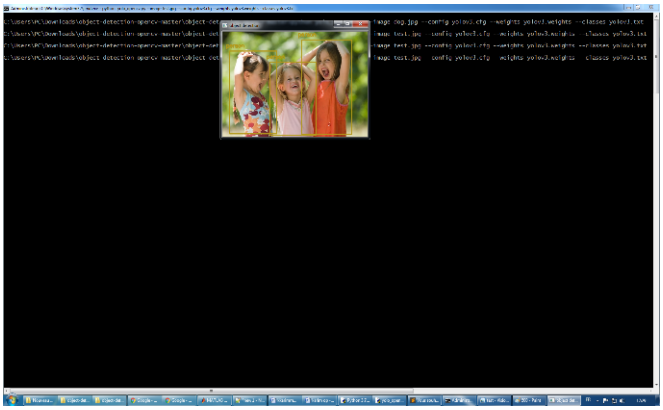


Fig. 12. Second example for person detection

The following diagram shows the movement of two objects selected randomly by the algorithm during the video. The motion is detected according to the change of the pixel positions of an object or a person from one image to another (Fig. 13).

As conclusion, detecting objects from images and videos is a bit easier than in real- time. When we have a video or an image that we want to detect an object from, we don't care much about the inference time the model might take to detect the object. In real-time object detection, we might want to sacrifice some precision over a faster inference time.

VI. Conclusion

The purpose of this work was to create a method that combines detection and tracking algorithms to track objects

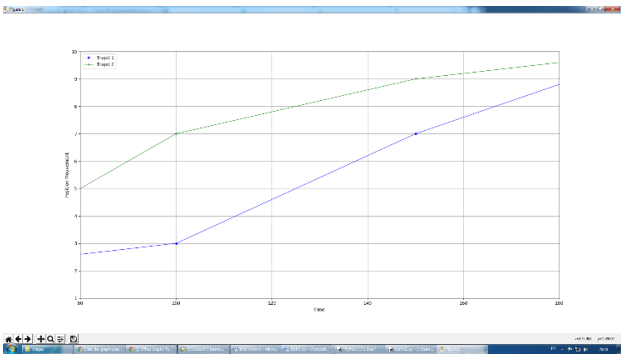


Fig. 13. Motion tracking of two randomly selected objects

or people in a video. More precisely, we used Deep Learning and TensorFlow for this goal. Our future work is to enhance this algorithm in order to extract semantic features from the video and to record them as ontological entities with formal relationships. We will improve the resulting ontology by integrating the moves and actions rules e.g. injecting the dynamic dimension of the video streams into the ontology structure.

REFERENCES

[1] S. Abburu and R. Anandhi, "Concept ontology construction for sports video," in *Proceedings of the 1st Amrita ACM-W Celebration on Women in Computing in India*, 2010, pp. 1–6.

[2] S. A. Mahmoudi, M. Ammar, G. L. Joris, and A. Abbou, "Real time gpu-based segmentation and tracking of the left ventricle on 2d echocardiography," in *Bioinformatics and Biomedical Engineering*, F. Ortuño and I. Rojas, Eds. Cham: Springer International Publishing, 2016, pp. 602–614.

[3] T. Zhang and C.-C. J. Kuo, "Audio content analysis for online audiovisual data segmentation and classification," *IEEE Transactions on speech and audio processing*, vol. 9, no. 4, pp. 441–457, 2001.

[4] K. Lee and D. P. Ellis, "Audio-based semantic concept classification for consumer video," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, no. 6, pp. 1406–1416, 2009.

[5] J. Liang, Q. Jin, X. He, G. Yang, J. Xu, and X. Li, "Detecting semantic concepts in consumer videos using audio," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2015, pp. 2279–2283.

[6] Q. Wu, Z. Wang, F. Deng, Z. Chi, and D. D. Feng, "Realistic human action recognition with multimodal feature selection and fusion," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 43, no. 4, pp. 875–885, 2013.

[7] D. Oneata, J. Verbeek, and C. Schmid, "Action and event recognition with fisher vectors on a compact feature set," in *Proceedings of the IEEE international conference on computer vision*, 2013, pp. 1817–1824.

[8] H. Naphide and T. S. Huang, "A probabilistic framework for semantic video indexing, filtering, and retrieval," *IEEE Transactions on Multimedia*, vol. 3, no. 1, pp. 141–151, 2001.

[9] W. Hu, N. Xie, L. Li, X. Zeng, and S. Maybank, "A survey on visual content-based video indexing and retrieval," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 41, no. 6, pp. 797–819, 2011.

[10] Y. Wang, S. Rawat, and F. Metze, "Exploring audio semantic concepts for event-based video retrieval," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2014, pp. 1360–1364.

[11] M. Gygli, H. Grabner, and L. Van Gool, "Video summarization by learning submodular mixtures of objectives," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3090–3098.

[12] S. A. Mahmoudi, E. Ozkan, P. Manneback, and S. Tosun, "Taking advantage of heterogeneous platforms in image and video processing," in *High-Performance Computing on Complex Environments*, 2014, pp. 429–449.

[13] S. A. Mahmoudi and P. Manneback, "Multi-gpu based event detection and localization using high definition videos," in *2014 International Conference on Multimedia Computing and Systems (ICMCS)*, 2014, pp. 81–86.

[14] M. Y. K. Tani, A. Lablack, A. Ghomari, and I. M. Bilasco, "Events detection using a video-surveillance ontology and a rule-based approach," in *European Conference on Computer Vision*. Springer, 2014, pp. 299–308.

[15] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: A system for large-scale machine learning," in *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, 2016, pp. 265–283.

[16] K. Ashangani, K. Wickramasinghe, D. De Silva, V. Gamwara, A. Nugaliyadde, and Y. Mallawarachchi, "Semantic video search by automatic video annotation using tensorflow," in *2016 Manufacturing & Industrial Engineering Symposium (MIES)*. IEEE, 2016, pp. 1–4.

[17] D. G. Murray, F. McSherry, R. Isaacs, M. Isard, P. Barham, and M. Abadi, "Naiad: a timely dataflow system," in *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles*, 2013, pp. 439–455.

[18] S. A. Mahmoudi, M. A. Belarbi, S. Mahmoudi, G. Belalem, and P. Manneback, "Multimedia processing using deep learning technologies, high-performance computing cloud resources, and big data volumes," *Concurrency and Computation: Practice and Experience*, vol. n/a, no. n/a, p. e5699, 2020. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/cpe.5699

[19] S. A. Mahmoudi, M. A. Belarbi, and P. Manneback, "Towards a smart exploitation of gpus for low energy motion estimation using full hd and 4k videos," in *Cloud Computing and Big Data: Technologies, Applications and Security*, M. Zbakh, M. Essaaidi, P. Manneback, and C. Rong, Eds. Cham: Springer International Publishing, 2019, pp. 284–300.

[20] Z. Qiu, T. Yao, and T. Mei, "Learning deep spatio-temporal dependence for semantic video segmentation," *IEEE Transactions on Multimedia*, vol. 20, no. 4, pp. 939–949, 2017.

[21] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural networks*, vol. 61, pp. 85–117, 2015.

[22] K. Roy and J. Mukherjee, "Image similarity measure using color histogram, color coherence vector, and sobel method," *International Journal of Science and Research (IJSR)*, vol. 2, no. 1, pp. 538–543, 2013.

[23] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587.

[24] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[25] J.-Y. Bouguet *et al.*, "Pyramidal implementation of the affine lucas kanade feature tracker description of the algorithm," *Intel Corporation*, vol. 5, no. 1-10, p. 4, 2001.

[26] S. A. Mahmoudi and P. Manneback, "Multi-gpu based event detection and localization using high definition videos," in *2014 International Conference on Multimedia Computing and Systems (ICMCS)*. IEEE, 2014, pp. 81–86.

[27] S. A. Mahmoudi, M. Kierzynka, and P. Manneback, "Real-time gpu-based motion detection and tracking using full hd videos," in *Intelligent Technologies for Interactive Entertainment*, M. Mancas, N. d' Alessandro, X. Siebert, B. Gosselin, C. Valderrama, and T. Dutoit, Eds. Cham: Springer International Publishing, 2013, pp. 12–21.

[28] G. Bradski and A. Kaehler, *Learning OpenCV: Computer vision with the OpenCV library*. " O'Reilly Media, Inc.", 2008.

[29] W. McKinney *et al.*, "pandas: a foundational python library for data analysis and statistics," *Python for High Performance and Scientific Computing*, vol. 14, no. 9, 2011.