

CAPRI 2017

Workshop on

Computational Aspects of Pattern Recognition and Computer Vision with Neural Systems

May 18th 2017

Anchorage, Alaska, USA

Organized under

The International Joint Conference on
Neural Networks (IJCNN 2017)

AGH University of Science and Technology
Signal Processing Group
Al. Mickiewicza 30
30-059 Krakow, Poland

This work was supported by the National Science Centre, Poland, grant no.
2016/21/B/ST6/01461

Editorial layout and cover design by Paweł Ksieniewicz

© Copyright by Bogusław Cyganek

AGH University of Science and Technology
Signal Processing Group
Al. Mickiewicza 30, 30-059 Krakow, Poland

ISBN 978-83-948049-0-9

Preface

Computational requirements on information processing systems are nowadays enormous - not only huge amounts of data needs to be processed and classified but also the systems need to deal with massive data usually in the form of data streams and frequently real-time processing requirements. On the other hand, neural systems proved their great potential, especially in pattern recognition and computer vision. However, all of the above rely heavily on efficient algorithms and continuously improved implementations. Therefore computational aspects become a key issue in pattern recognition and computer vision.

The aim of this workshop to collect researchers and practitioners to share interesting research topics and ideas especially in the area of computational aspects of pattern recognition and computer vision processed on all types of neural systems, starting from algorithm design and up to implementations and applications, encountered in computer vision and pattern recognition computer vision for information mining, especially form from massive data streams and new neural architectures. Scope of the workshop includes, but is not limited, to the following topics:

- Parallel implementations of pattern recognition and computer vision neural systems;
- Deep learning techniques and new achievements in computer vision with special stress on image enhancement for pattern recognition;
- Real-time neural systems, their implementation and application;
- Rapid neural system development new directions and platforms;
- Graphic card (GPU) implementations of pattern recognition and computer vision systems;
- Hardware implementations (FPGA) of pattern recognition and computer vision systems;
- New algorithms for efficient computations on pattern recognition and computer vision neural systems;
- Tips and tricks in pattern recognition and computer vision algorithms;
- Industrial applications of pattern recognition and computer vision, especially with dedicated streaming data;
- Computational aspects in all kinds of massive and streaming data;
- Pattern recognition in computer vision, multimedia, and image processing;
- Multilinear and tensor approach to data representation and pattern recognition;
- Active learning for neural based pattern recognition and computer vision;
- Hyperspectral image processing;
- Pattern recognition in hyperspectral images;
- Visualization and sonification for high dimensional data;

May 2017

Boguslaw Cyganek
Michał Wozniak
Workshop Chairs
CAPRI 2017

Organization

Workshop Chairs

Boguslaw Cyganek AGH University of Science and Technology, PL
Michał Wozniak Wrocław University of Science and Technology, PL

Program Committee

Boguslaw Cyganek	AGH University of Science and Technology, PL - chair
Michał Wozniak	Wrocław University of Science and Technology, PL - chair
Tomasz Andrysiak	University of Technology and Life Sciences, PL
Colin Bellinger	University of Alberta, CA
Robert Burduk	Wrocław University of Science and Technology, PL
Alberto Cano	Virginia Commonwealth University, US
Alberto Fernandez	University of Granda, ES
Mikel Galar	Public University of Navarra, ES
Manuel Grana	University of the Basque Country, ES
Bartosz Krawczyk	Virginia Commonwealth University, Richmond, US
Pawel Ksieniewicz	Wrocław University of Science and Technology, PL
Piotr Porwik	University of Silesia, PL
Radek Silhavy	Tomas Bata University in Zlín, CZ
Isaac Triguero	University of Nottingham, UK
Richard Zurawski	ISA Group, US, US

Organizing Committee

Dariusz Jankowski	Wrocław University of Science and Technology, PL
Pawel Ksieniewicz	Wrocław University of Science and Technology, PL
Alex Savio	Klinikum rechts der Isar, TUM, Munich, DE
Michał Koziarski	Wrocław University of Science and Technology, PL
Mateusz Knapik	AGH University of Science and Technology, PL

Table of Contents

Automated Image Captioning Using Nearest-Neighbors Approach Driven by Top-Object Detections	1
<i>K. Sharma, A. CS Kumar, S. M. Bhandarkar</i>	
Speeding up Deep Neural Networks on the Jetson TX1	11
<i>M. Eisenbach, R. Stricker, D. Seichter, A. Vorndran, T. Wengefeld, and HM. Gross</i>	
Software Framework for Morphological Neural Network Computations ...	23
<i>B. Cyganek</i>	
Evaluation of background subtraction methods in thermal imaging	33
<i>M. Knapik, B. Cyganek</i>	
Author Index	41

Automated Image Captioning Using Nearest-Neighbors Approach Driven by Top-Object Detections

Karan Sharma

Arun CS Kumar

Suchendra M. Bhandarkar

Department of Computer Science, The University of Georgia

Athens, Georgia 30602–7404, USA

{karan@uga.edu, aruncs@uga.edu, suchi@cs.uga.edu}

Abstract. The significant performance gains in deep learning coupled with the exponential growth of image and video data on the Internet have resulted in the recent emergence of automated image captioning systems. Two broad paradigms have emerged in automated image captioning, i.e., generative model-based approaches and retrieval-based approaches. Although generative model-based approaches that use the recurrent neural network (RNN) and long short-term memory (LSTM) have seen tremendous success in recent years, there are situations in automated image captioning for which generative model-based approaches may not be suitable and retrieval-based approaches may be more appropriate. However, retrieval-based approaches are known to suffer from a computational bottleneck with increasing size of the image/video database. With an aim to address the computational bottleneck and speed up the retrieval process, we propose an automated image captioning scheme that is driven by top-object detections. We surmise that by detecting the top objects in an image, we can prune the search space significantly and thereby greatly reduce the time for caption retrieval. Our experimental results show that the time for image caption retrieval can be reduced without suffering any loss in accuracy.

Keywords: Automated image captioning, top-object detection, image retrieval, k -nearest-neighbor search

1 Introduction

Automated image captioning, i.e., the problem of describing in words the situation captured in an image, is known to be challenging for several reasons. The recent significant performance gains in deep learning coupled with the exponential growth of image and video data on the Internet have resulted in the emergence of *automated* image captioning systems. Two broad paradigms have emerged in the field of automated image captioning, i.e., generative model-based approaches [3], [5], [9], [11], [17] and retrieval-based approaches [1]. Although generative model-based approaches that use the recurrent neural network (RNN) and long short-term memory (LSTM) have seen tremendous success in recent years, there are situations for which retrieval-based approaches may be better suited. Examples of such situations include:

- (1) Situations wherein the training sets are dynamically changing. To keep up with the increasing pace of visual data being constantly uploaded on the Internet, computer

vision practitioners face a challenging task of training models that are capable of adapting to constantly changing datasets or reducing the size of the datasets. By reducing the size of the datasets, one runs the risk of discarding useful data resulting in the learning of simplistic models. Adaptive models have the added overhead of requiring constant training or retraining as the underlying datasets change over time. Moreover, adaptive models need to deal with the problem of *concept drift*, i.e., situations where the statistical properties of the target variable or concept, which the model is trying to predict, change over time in unforeseen ways, especially when the new data being uploaded is significantly different from previously observed data. In contrast, retrieval-based approaches, modeled on nearest-neighbor search, do not entail the overhead of constant retraining of models since one can store all the images in the dynamically changing dataset in a database.

(2) Situations wherein one needs to deploy an automated image captioning system with the goal of simultaneously reducing system development time and CPU execution time. Nearest-neighbor approaches lend themselves easily to rapid implementation and deployment since they have very few tunable hyperparameters compared to other approaches. Hence retrieval-based approaches based on nearest-neighbor search are naturally preferred in rapid prototyping situations. However, the potential downside of retrieval-based approaches is that nearest-neighbor search can be exhaustive if one has to perform all possible comparisons between the query image and the database entries. Traditionally, techniques such as locality sensitive hashing (LSH) have been used to speed up nearest-neighbor search. However, effective use of LSH requires the proper tuning of several hyperparameters in order to achieve accurate results. In contrast, the proposed approach has very few tunable hyperparameters and hence a much less computationally intensive hyperparameter tuning phase.

Although retrieval-based approaches to automated image captioning have not been as successful as generative model-based approaches, the performance of retrieval-based approaches has been observed to be not very far behind that of RNN- and LSTM-based approaches when addressing the *Microsoft Common Objects in Context* (MS COCO) challenge. Therefore the obvious question arises - in situations (such as the ones described previously) where retrieval-based approaches are called for, how does one speed up the nearest-neighbor search procedure? To this end, we propose a variant of the nearest-neighbor search procedure to speed up image caption retrieval using top-object detections. Specifically, we use the detection of the most significant objects in an image (i.e., the top objects) to speed up the k -nearest-neighbor (k -NN) search for retrieval-based automated image captioning. Although, as noted previously, approaches such as LSH can be used to accelerate the retrieval process, LSH entails a hyperparameter tuning procedure that is computationally complex and difficult to implement thereby calling for a significant expenditure of programmers' development time.

2 Related Work

Automated image captioning: Automated image captioning systems have grown in prominence owing, in large part, to the tremendous performance gains shown by deep

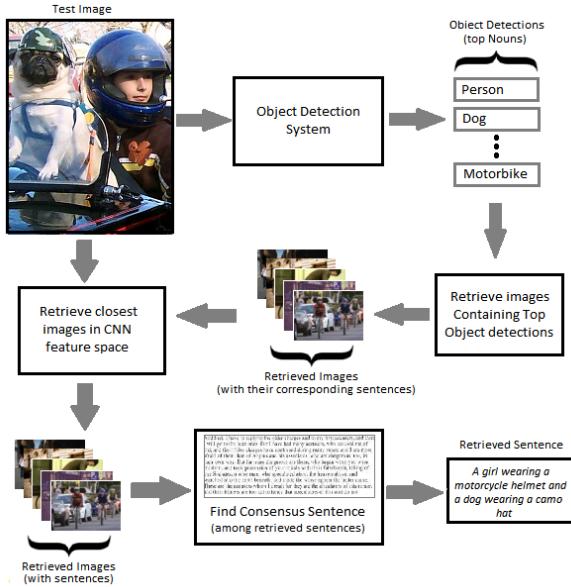


Fig. 1. Top-object detections used to drive k -NN search.

learning in recent times. Existing automated image captioning systems can be categorized as either generative model-based or retrieval-based. Generative model-based systems involve correctly identifying objects, verbs, adjectives, prepositions or visual phrases in an image and generating a caption from these or directly from the representation of the image [3], [5], [9], [11], [17]. Retrieval-based approaches [1], on the other hand, involve retrieving the most suitable caption from a database of captions and assigning it to an image. Currently, generative model-based approaches that use the RNN and LSTM have been shown to yield the best performance metrics in the context of automated image captioning; however retrieval-based approaches have also proved to be quite competitive in terms of performance. The generative model-based and retrieval-based paradigms are each suited for different kinds of situations and applications. However, in situations where retrieval-based approaches are more appropriate and successful, we propose a scheme to optimize and speed up the caption retrieval process by exploiting the top-object detections in the image.

3 Motivation

Although generative model-based approaches that use the RNN and LSTM are regarded as the state-of-the-art in automated image captioning, there are potential situations for which they may not be well suited and hence retrieval based captioning approaches may be called for. However, retrieval-based approaches to automated image captioning can be computationally intensive and slow especially when a query image is compared with all the images stored in the database. However, before we proceed to address the question of how to speed up retrieval-based approaches to automated image captioning,

we digress to answer an important related question, i.e., under what potential situations would retrieval-based approaches have an advantage over state-of-the-art generative model-based approaches that use the RNN and LSTM in the context of automated image captioning?

Concept Drift: Consider the problem of automated image captioning in situations where the underlying datasets are dynamically changing such as when visual data (in the form of images and videos) is being uploaded over the Internet at an extraordinary pace, both on popular online social media (OSM) platforms such as Facebook, Instagram, Snapchat, Google and Twitter, and on websites that contain more structured and specific information such as those dealing with news, sports, art, and technology. Many of the uploaded images and videos have some sort of textual information associated with them, typically in the form of tags, captions, and/or comments. Mining such a large data set is tremendously challenging for most computer vision practitioners. The constant pace of the dynamically changing dataset makes it incredibly difficult to learn reliable computer vision models. The standard assumption underlying most machine learning techniques is that the training data will be similar to the testing or querying data. However, in dynamic situations where the underlying data is continuously changing, it is especially hard to train reliable models. In this paper, we propose a retrieval-based model for automated image captioning for situations wherein the training datasets are very volatile and constantly changing.

One of the problems faced when dealing with dynamic datasets is the problem of *concept drift* where the function learned by a machine learning model is rendered not particularly useful for newly arriving data. For example, near Christmas, people tend to post more pictures or images of their activities around a Christmas-oriented theme on OSM sites. An existing machine learning model may not be in situation to automatically label or caption these images since it has not seen these images previously. One solution is to constantly retrain the existing model as the new data arrives or use models that are capable of adapting to new data. However, the constant retraining of models could pose significant and, in some cases, an impossibly high computational demand, especially in situations where images are being uploaded at a very rapid pace. Moreover, many adaptive models, in the interest of computational efficiency, subsample the data during retraining. The discarding of data could lead to the learning of overly simplistic models. The interested reader is referred to the work of Gama et al. [7] for a more detailed and comprehensive treatment of the concept drift problem.

For the reasons mentioned above, some of the most popular automated image captioning schemes, based on generative models that use the RNN and LSTM, are seriously disadvantaged in situations where a large proportion of the training data is in a state of constant flux. In such instances, the generative models will learn a classification or prediction function that could account for most cases, but may miss cases that occur only a few times. Moreover, the cases that occur infrequently may contain valuable information. For example, if the training set has millions of images, and only five instances of *Man is biting a dog*, the generative model may simply ignore this infrequent case during the training process, although the case may be of potential interest. Hence, for this reason and reasons described in previous paragraph, generative models are not well suited for image captioning under dynamically changing training datasets. However,

retrieval-based approaches, such as ones based on k -NN search do not suffer from such problems. It has been convincingly shown by Hays and Efros [8] that k -NN search is one of the most effective retrieval algorithms if one has a very large dataset. However, exhaustive k -NN search could be computationally very expensive. Although techniques such as LSH have been traditionally used to speed up k -NN search-based image retrieval [4], the hyperparameter tuning procedure needed to optimize the performance of LSH is non-trivial in terms of its computational complexity [4]. The situation is further complicated if we need to retune the LSH procedure in the face of constantly arriving new training data. Thus, retrieval-based automated image captioning techniques suffer from the same disadvantages as their generative model-based counterparts if the former use k -NN search optimized via LSH. In this paper, we propose a simple retrieval-based technique for automatic image captioning that is accurate, reliable and computationally efficient. The proposed technique is based on enhancing the k -NN search by exploiting the top-object detections in an image.

Rapid Prototyping: We use top-object detections to speed up the caption retrieval procedure during automated image captioning. Specifically, we use the detection of the most significant objects in an image (i.e., the top objects) to speed up the k -NN search for retrieval-based automated image captioning. We show top-object detection to be a preferable alternative to the more conventional retrieval-based automated image captioning methods that employ LSH to speed up the k -NN search. It is to be noted that although techniques such as LSH can be used to speed up k -NN search-based image retrieval, the hyperparameter tuning procedure needed to optimize the performance of LSH is non-trivial in terms of computational complexity [4], especially in the case of complex applications such as automated image captioning. Thus, complete automation of the LSH procedure for automated image captioning is a challenging task. Implementation and proper tuning of LSH also presents a significant expenditure of system development time, which is an important consideration in real-world situations where rapid prototyping is called for.

4 k -NN Search Driven by Top-Object Detections

Previously, Devlin et al. [1] have obtained good results for automated image captioning based on k -NN search-based image retrieval. Their approach determines the k -NN images by computing a measure of image similarity between the test/query image and each of the database images. The test/query image is then assigned the caption obtained by computing the consensus of the retrieved k -NN image captions. Performing an exhaustive search of the image database to retrieve the k -NN images using an image feature-based similarity metric is clearly not a scalable approach. We show that, in the context of automated image captioning, by detecting all objects in a test image, selecting the top- n objects (where n is a small number) and retrieving all images that contain at least one of these n objects, one can achieve results comparable to those of k -NN retrieval via exhaustive search while simultaneously obtaining a significant speedup. Fig. 2 summarizes the proposed approach. We demonstrate our approach on the MS COCO dataset as a proof of concept. We believe the experimental results on the MS COCO dataset are transferable and generalizable to real-world dynamic datasets.

Although the proposed approach involves tuning the parameters of a support vector machine (SVM)-based classifier for object detection/recognition, it is computationally much less expensive than the LSH hyperparameter tuning procedure used to optimize k -NN search and also yields readily to automation.

Although running various object (i.e., noun) detectors on the test/query image imposes a computational overhead, it is offset by the following considerations: (a) the space of objects (i.e., nouns) is bounded. Also, since objects are concrete entities, generating training sets for object detectors is not very difficult if one uses web-based data coupled with crowdsourcing, (b) sliding windows are not used during the object detection procedure, i.e., the entire test/query image is fed as input to the SVM-based object detector. The computational overhead of object detection in the test/query image is also offset by: (a) the resulting speedup over k -NN image retrieval via exhaustive search and, (b) savings in development time compared to the scenario wherein k -NN image retrieval is optimized using LSH. Additionally, the proposed approach also results in significant savings in CPU execution time as shown in Table 1.

Complexity Analysis: Given a set of objects $X = \{x_1, x_2, \dots, x_n\}$, and a set of images $I = \{I_1, I_2, \dots, I_m\}$, we make the following assumption regarding the dataset: Each object x_i does not occur in more than k images in the dataset where $k \ll m$. In real world datasets, especially in large datasets, it is expected that no single object category will dominate the images in the dataset. Even generic categories such as *person, car, ...*, would be expected to occur in a significantly small percentage of the total number of images in the dataset. Also, for a small subset $Y \subset X$ where no member of Y occurs in more than r images ($r \ll m$) in the dataset, the number of comparisons is bounded by $r \cdot |Y|$ resulting in a $O(r \cdot |Y|)$ time complexity. However, what if r is a large number? We argue that in datasets that are sufficiently representative of real world, this will not be the case. For example, consider an image whose top detections are *person, dog, road, and building*. Intuitively, in a large dataset representative of many nouns and concepts in the world, we can expect that all the images that contain at least one entity from the set $\{\textit{person, dog, road, building}\}$ are far fewer than all the images in the dataset thus resulting in an order of magnitude reduction in search complexity.

Retraining Event Analysis: Assume an image dataset (with associated captions for each image) of size N (i.e., N is the number of data points). Assume this dataset is being constantly augmented with new incoming image data (and the associated captions). Assume that after every w data points (i.e., images) are added to the dataset, there is a concept drift, that requires retraining of the model. In a traditional generative model-based system that uses an RNN, retraining will be needed in two situations after the addition of new data points to the existing dataset:

- (a) Changes in concepts, where a concept is any word, which includes nouns, verbs, adjectives and so on. Assume that the concepts change at an average rate of c concepts after w new data points are introduced. Clearly, the space of concepts is far greater than the space of objects (i.e., nouns). Let $tr(c)$ denote the average number of training events required to account for the concept changes after a collection of w new data points is added to the existing dataset.
- (b) Changes in concept dependencies. The dependency between two words is a measure of how much a given word depends on the other word. For example, the word *eating*

is dependent on the words *person* and *food*. We need to retrain the model to learn such dependencies after a collection of w new data points is introduced. Assume that the concept dependencies change an average rate of d concept dependencies upon introduction of a collection of w new data points. Again, based on our understanding of the real world, the space of these dependencies is significantly larger than the space of objects alone. Let $tr(d)$ denote the average number of training events required to account for the changes in concept dependencies after a collection of w new data points is added to the existing dataset.

In contrast to a traditional generative model-based system, in the proposed approach, the training events will be required only when new objects are introduced at an average rate of ob objects after w new data points are added to the existing dataset. Clearly, the training events are bounded by the number of objects under consideration. Let $tr(ob)$ denote the average number of training events required after w new data points are added to the existing dataset. Based on our knowledge of the real world and the above arguments, the training events in the proposed approach will be significantly fewer than the training events in a traditional generative model-based system (such as one that uses an RNN), i.e., $tr(ob) \ll tr(c) + tr(d)$.

5 Experimental Results

Training: For the purpose of training, we use 80 annotated object categories in the MS COCO dataset [10]. Binary SVM classifiers are trained for each of these 80 annotated categories using VGG-16 *fc-7* image features [13], and the SVMs are calibrated using Platt scaling. For the extraction of *fc-7* features, Matconvnet package [15] is employed.

In addition, we store each training image in the MS COCO dataset and its accompanying sentences (5 sentences per image) in our database. We treat these sentences as ground truth captions for the corresponding training image. For testing purposes, we consider the MS COCO validation set consisting of close to 40,000 images.

Testing: For each test image in the MS COCO validation set, we run all the 80 object detectors on the test image. We select the top- n objects from all the detected objects in the image. In our current implementation $n = 5$. The detected top objects are the ones that are deemed to possess the highest probability of occurrence in the image. The probability of occurrence of an object in the image is computed by mapping the classification confidence value generated by the SVM classifier for that object to a corresponding probability value using Platt scaling [12]. From the training dataset, we retrieve all images that contain at least one of the top- n objects detected in the previous step, using the corresponding ground truth captions, i.e., a training image is retrieved if at least one of its associated ground truth captions contains a noun describing the object under consideration. In addition, for the purposes of retrieval, all the synonyms for certain words such as *person* (synonyms are man, woman, boy, girl, people, etc.) are taken into consideration. Using the cosine distance between the *fc-7* features of each retrieved image and the test image, we select the k -NN images for further processing.

In the current implementation we have chosen $k = 90$ as recommended by [1]. Since each of the k -NN images has 5 associated sentences (captions), we have a total of $5k$ potential captions for the test image. We determine the centroid of the $5k$ potential

captions and deem it to represent the consensus caption for the test image. The consensus caption is then assigned to the test image in a manner similar to [1]. The BLEU measure is used to evaluate the similarity (or distance) between individual captions and to determine the centroid of the $5k$ potential captions. We have also implemented image retrieval using exhaustive k -NN search [1] and compared the CPU execution time of the proposed approach with that of image retrieval using exhaustive k -NN search for 2000 random images .

Results: As shown in Table 1, the proposed image retrieval, using k -NN search driven by top-object detections, and the standard image retrieval, that employs exhaustive k -NN search, yield very similar results when the BLEU and CIDEr [16] similarity metrics are used to compare the retrieved captions.

Table 1. Comparison of image captioning results obtained using the proposed approach for image retrieval based on k -NN search driven by top-object detections (Obj- k -NN) and those obtained using conventional image retrieval based on exhaustive k -NN search (Exh- k -NN).

	BLEU ₁	BLEU ₂	BLEU ₃	BLEU ₄	CIDEr	CPU time
Exh- k -NN	65.6%	47.4%	34%	24.7%	0.70%	2.5e+04s
Obj- k -NN	64.6%	46.2%	32.8%	23.6%	0.68%	1.17e+04s

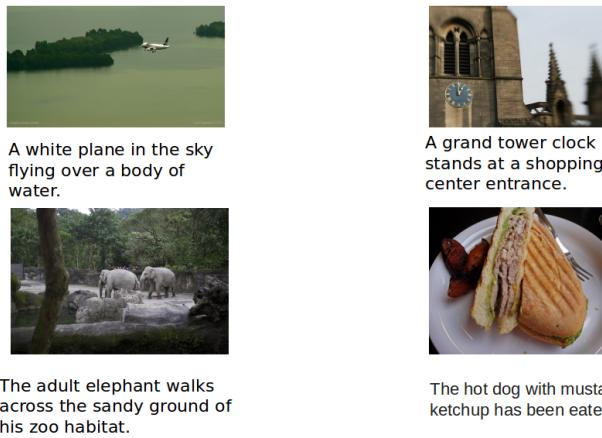


Fig. 2. Qualitative Results for nearest neighbor driven by top-object detections. Some captions retrieved accurately describe the image while others are partially correct.

The proposed approach is seen to yield significant gains in CPU execution time when compared to image retrieval using exhaustive k -NN search. Essentially, the proposed image retrieval technique based on k -NN search driven by top-object detections is observed to provide an attractive alternative to LSH for the purpose of speeding up

k -NN search-based image retrieval in the context of automated image captioning. As a proof of concept, the results of the proposed image retrieval technique based on k -NN search driven by top-object detections on the MS COCO dataset are fairly convincing. We believe that these results could be directly transferred to real-world datasets that are dynamically changing.

These results show that k -NN search driven by top-object detections, even though simple in concept, can provide significant gains in critical situations where the datasets are dynamically changing. This approach requires that we store all the training images along with their associated captions in the database. When dealing with real-world problems, we will store all the image instances in the database and retrieve the relevant images from the database using top-object driven k -NN search. There are three advantages to the proposed approach: We do not need to subsample the dataset by discarding any potentially useful information, we do not need to exhaustively search for the k -NN images, and we do not need to retrain the retrieval models in the face of changing information.

6 Conclusions

We have shown that retrieval-based approaches for automated image captioning could be made computationally more efficient if they are driven by top-object detections. The potential advantages of our approach are in situations where the underlying datasets are changing dynamically. In addition, the proposed approach needs much less parameter tuning when compared to the computationally intensive hyperparameter tuning associated with traditional LSH-based optimization of k -NN search. The proposed approach is a natural candidate for use in rapid prototyping conditions that also call for optimization of CPU time.

Acknowledgment The authors wish to thank Devi Parikh for her invaluable suggestions during this research.

References

1. Devlin, J. et al. (2015). Exploring nearest-neighbor approaches for image captioning. *arXiv preprint arXiv:1505.04467*.
2. Devlin, J. et al. (2015). Language models for image captioning: The quirks and what works. *Proc. ACL 2015*.
3. Donahue, J. et al. (2014). Long-term recurrent convolutional networks for visual recognition and description. *Proc. IEEE Conf. Comp. Vis. Patt. Recog. (CVPR 2014)*.
4. Dong, W. et al. (2008). Modeling lsh for performance tuning. *Proc. ACM Conf. Info. & Know. Mgmt.*, October, pp. 669-678.
5. Fang, H. et al. (2014). From captions to visual concepts and back. *arXiv preprint arXiv:1411.4952*.
6. Farhadi, A. et al. (2010). Every picture tells a story: Generating sentences from images. *Proc. Eur. Conf. Comp. Vis. (ECCV 2010)*, pp. 15-29.
7. Gama, J., Zliobaite, I., Bifet, A., Pechenizkiy, M., and Bouchachia, A. (2014). A survey on concept drift adaptation. *ACM Computing Surveys (CSUR)*, Vol. 46(4), pp. 44.

8. Hays, J., and Efros, A. A. (2007). Scene completion using millions of photographs. *ACM Transactions on Graphics* (TOG), Vol. 26(3), pp. 4, August.
9. Karpathy, A. and Fei-Fei, L. (2015). Deep visual-semantic alignments for generating image descriptions. *Proc. IEEE Conf. Comp. Vis. Patt. Recog.* (CVPR 2015).
10. Lin, T. Y. et al. (2014). Microsoft COCO: Common objects in context. *Proc. Eur. Conf. Comp. Vis.* (ECCV 2014), pp. 740-755.
11. Mao, J. et al. (2014). Explain images with multimodal recurrent neural networks. *Proc. NIPS 2014*.
12. Platt, J. (1999) Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in Large Margin Classifiers*, Vol.10(3), pp. 61-74.
13. Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *Proc. Intl. Conf. Learn. Rep.* (ICLR 2014).
14. Slaney, M. et al. (2012). Optimal parameters for locality-sensitive hashing. *Proc. IEEE*, Vol. 100(9), pp. 2604-2623.
15. Vedaldi, A. and Lenc, K. (2015). MatConvNet-convolutional neural networks for MATLAB. *Proc. ACM Conf. Multimedia Systems* (MMSys 2015).
16. Vedantam, R. et al. (2015). Cider: Consensus-based image description evaluation. *Proc. IEEE Conf. Comp. Vis. Patt. Recog.* (CVPR 2014).
17. Vinyals, O. et al. (2014). Show and tell: A neural image caption generator. *Proc. IEEE Conf. Comp. Vis. Patt. Recog.* (CVPR 2014).
18. Yang, Y. et al. (2011). Corpus-guided sentence generation of natural images. *Proc. Conf. EMNLP*, pp. 444-454.

Speeding up Deep Neural Networks on the Jetson TX1

Markus Eisenbach, Ronny Stricker, Daniel Seichter, Alexander Vorndran, Tim Wengefeld, and Horst-Michael Gross*

Neuroinformatics and Cognitive Robotics Lab,
Ilmenau University of Technology,
98694 Ilmenau, Germany.
markus.eisenbach@tu-ilmenau.de

Abstract. In recent years, Deep Learning (DL) showed new top performances in almost all computer vision tasks that are important for automotive and robotic applications. In these applications both space and power are limited resources. Therefore, there is a need to apply DL approaches on a small and power efficient device, like the NVIDIA Jetson TX1 with a powerful GPU onboard. In this paper, we analyze the Jetson’s suitability by benchmarking the run-time of DL operations in comparison to a high performance GPU. Exemplary, we port a top-performing DL-based person detector to this platform. We explain the steps necessary to significantly speed up this approach on the device.

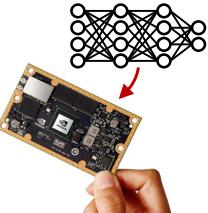
1 Introduction

In recent years, Deep Learning approaches have surpassed the performance of traditional computer vision methods by far for almost all image processing tasks that are essential for autonomous cars and mobile service robots. Examples are object recognition [27], scene understanding [8], person detection [6], and many more. All these DL approaches need extensive computational resources. Thus, usually they are processed on high performance GPUs. For neural network training, indeed, much computation power is indispensable to get the task done in a reasonable amount of time. Once trained, the deep networks can be applied on less powerful systems, but still need an adequate GPU to be fast.

Mobile robots have very tight power restrictions to guarantee an appropriate service accessibility time. Thus, high performance GPUs are typically unsuitable, since they are known for their high power consumption. Similar restrictions apply to the size and weight of components for autonomous cars, in order to increase the passenger compartment and carrying capacity as well as the car’s fuel-efficiency. Bulky high performance PCs with powerful GPUs contradict this principle.

Luckily, in 2015 NVIDIA presented the Jetson TX1, an embedded system of the size of a credit card with a GPU onboard, that consumes less than ten

* This work has received funding from the German Federal Ministry of Education and Research as part of the SYMPARTNER project under grant agreement no. 16SV7218.



CPU	4-core ARM Cortex-A57 @ 1.9 GHz
GPU	256-core Maxwell @ 1 GHz
RAM	4 GB LPDDR4 (shared CPU + GPU)
Processing speed	1024 GFLOPS in float16 precision
Communication	Gigabit Ethernet, WiFi, Bluetooth, USB
Power consumption	< 10 W (peak 15 W in worst case)
Size	50×87 mm

Fig. 1. Basic data of NVIDIA Jetson TX1 platform used for Deep Learning.

watts (see Fig. 1). Therefore, this platform is perfectly suited for application on autonomous cars or mobile robots. For comparison, on our mobile robotic rehabilitation assistant [11] and our robotic companion for domestic use [13] we need two PCs with Intel core-i7 CPUs (i7-4770R, 4 Cores @ 3.2 GHz) to provide all the robot’s services simultaneously, which together consume more than 170 watts.

In this paper, we show that computational expensive Deep Neural Network (DNN) computations can be outsourced to a Jetson TX1 with only very little communication overhead. Therefore, we show detailed run-time analyses of typical DL operations on this platform. Exemplary, we choose person detection as application, which is a central task for both autonomous cars (pedestrian recognition [4]) and mobile service robots (being aware of persons for keeping personal space [38], being polite in navigation [37], following a specific person [11], or identify the current user [7]).

In [6] we presented a DL-based person detector that surpassed the state of the art on the standard person detection benchmark dataset Caltech [4]. This neural net has relatively low memory requirements, which makes it a good candidate for porting it to the Jetson TX1. We will explain the steps necessary to speed up this approach on a Jetson TX1. Then we will show its application on a mobile robot, where it significantly outperforms traditional person detectors and other DL approaches.

2 Related Work

In recent years, low power consuming embedded devices with onboard GPU have gained increased attraction in a wide range of research fields. Deep neural networks heavily benefit from parallelization. Thus, they are perfectly suited for application on such devices.

2.1 Automotive Applications

The Jetson TX1 is widely spread among autonomous car research projects. It has been used for navigation tasks [32], sensor fusion and probabilistic tracking [19], semantic road scene segmentation based on dynamic programming [14], and

DL-based traffic sign recognition [25]. In [36] a DL-based semantic image segmentation is implemented on this embedded platform by applying the smaller SqueezeNet architecture instead of DL nets, that have many weights, to gain a speedup. The segmentation performance decreases due to the modifications but is still sufficient for self-driving cars. In [15] a related technique, called "distillation", is used for decreasing the size of a network in order to obtain computational savings.

2.2 Robotic Applications

Embedded platforms like Jetson TK1 and TX1 have been used on mobile robots for several tasks, such as the rectification of an omnidirectional image [33], particle-based monte carlo localization [29], SLAM [10], path planning [24], and speech processing [30]. In [16], sonar images of an underwater robot are classified by a DL approach on a TX1. They apply a retrained, but structurally unchanged YOLO detector [27] without further optimization.

In [21] a combination of a less powerful embedded platform on a mobile robot in combination with a cloud solution is proposed for Deep Learning. The onboard device processes just shallow neural networks. High accuracy can only be achieved by sending the computing job to a high performance server. In our opinion, this is not a feasible solution, since in many real-world environments WiFi might not be fast and reliable enough for unconstrained robot applications, as experienced e.g. in medical environments such as rehab clinics [11], in private apartments of elderly people [13], and in stores [12]. In these cases, the robot would not be able to provide service tasks that depend on DL-based modules. But a mobile service robot should be able to fulfill all of its tasks with high accuracy at any time. Therefore, DL operations should be performed onboard without the need for a cloud solution.

2.3 Person Detection

Also person detection has been done on a TK1 on a mobile robot. In [31] face detection is used to locate the user of a telepresence robot. In [2] persons are detected in 3D point clouds of a Kinect 1. These approaches, however, do not deploy Deep Learning.

In [1] classical computer vision based person detection approaches are ported to the Jetson TX1 achieving a huge speedup, up to 20 frames per second (HOG + LBP), due to the use of a GPU instead of a CPU. While the speed is very good, the accuracy is only mediocre.

Deep Learning approaches for person detection have also been ported to the Jetson TX1. In [40], an AlexNet [18] is retrained to detect skin color as indication for the presence of persons. This approach is not sufficient to detect persons robustly.

More often, general object detectors trained on ImageNet are applied. These include the class 'person', but are not fine-tuned on this specific task. In [20] a multi-scale wide residual inception network is applied for object detection on

the TX1. It can process 16 rescaled frames of size 300×300 per second, but the detection results for the class 'person' are only mediocre. The YOLO object detector [27] is ported to the Jetson TK1 in [26]. It does not fit into the memory of the TK1. Therefore several strategies to reduce the network size are evaluated. Since fully connected (FC) layers need more than 80% of network's memory, the lack of memory could be handled by splitting the FC layers and processing parts of them sequentially. Other techniques, like decreasing the network size, degraded the performance. On the Jetson TK1, they processed 4 rescaled frames of size 448×448 per second. In [42], the YOLO detector was ported to a Jetson TX1 without modifications. They were able to process 12 frames of size 448×448 per second. In [22] the Faster R-CNN [28] approach based on VGG networks was ported to the Jetson TX1 without modifications. It is reported, that the detection for 1280×720 images "is near real-time frame rates".

All these approaches are relatively good general object detectors. However, for person detection they perform relatively poor (see Sec. 4.3).

2.4 Our Contribution

Summarized, none of these studies has analyzed the run-time of DL operations on the Jetson TX1 in detail. Only few of these studies have adapted their baseline DL approach in terms of computational savings for processing on the embedded platform. Most approaches take the neural nets as they are. In comparison, we analyze the run-time of DL operations and show, which additional optimizations speed up the computation without decreasing accuracy. As baseline, we use a specialized person detector from our previous work [6].

3 Speed up a Deep Learning based Person Detector on the Jetson TX1

3.1 Baseline Multi-Scale CNN Person Detector

For detecting persons at different scales, we build on our previous work [6] that set a new top mark on the most popular Caltech pedestrian detection benchmark. It uses a resolution pyramid in combination with three Convolutional Neural Networks (CNNs). Due to the use of multiple CNNs at different scales, the learned features are specific for the respective resolutions the particular CNNs are applied to, which improves the performance significantly. The network topologies used in the different stages are similar with the exemplary stage displayed in Fig. 2. The networks take raw pixels as input and predict whether the image patch shows a person or not. The networks were designed with a real-time application and an embedded device in mind. Thus, the largest net (Fig. 2) has relatively few weights (4M) to fit into the memory of a Jetson TX1, even when applied to a full-size image. More modern architectures known for their good performance on ImageNet would not fulfill this requirement. Additionally, the net is not overly deep and wide to avoid needless computations (see Fig. 2). In [6], the design choices are described in detail.

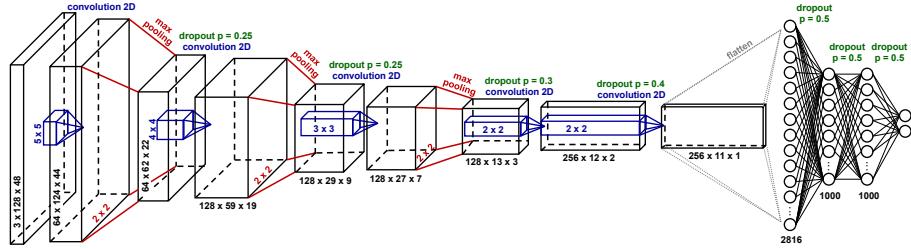


Fig. 2. CNN topology for detecting persons. The neural network consists of five convolutional layers with ReLU activation [23], three max-pooling layers, two fully connected layers with ReLU activation, and a softmax output layer. Dropout [34] is used for regularization.

The three CNNs were trained on a large and versatile dataset composed of 22 datasets from pedestrian detection and person re-identification domains. It contains cropped images showing persons (100,107 samples) and 628,636 non-person samples. The negative class includes random crops from non-person objects, typical false detections, and (on purpose) badly aligned crops showing only parts of persons. In this paper, we use the trained weights of [6]. Since the training dataset is versatile and generic, we do not need to retrain on a scenario specific dataset.

After the networks were trained, fully connected layers were converted to convolutional layers to be able to process images of any size without the need to shift a sliding window to several locations. Fig. 3 shows the processing chain of the application phase. Each of the CNNs calculates output maps for multiple scales of the resolution pyramid. When these classifications have been done, the

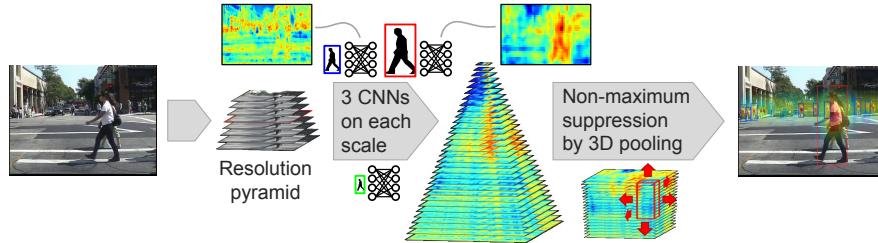


Fig. 3. Baseline approach [6]. Processing chain in the application phase. To create the resolution pyramid, the input image is scaled such that it is exactly halved in size after seven scales. Thus, each of the three CNNs has to process seven scales. The near scale CNN (red box) additionally processes smaller scales to detect larger persons. Exemplary network outputs are shown. High neural activations (shown in red) suggest that persons are present in that region of the image. The classification results are stacked and non-maximum suppression implemented as 3D pooling is applied to find the best fitting positions and scales for all persons in the scene.

full output pyramid can be constructed. Then, a 3D non-maximum suppression (NMS) is applied to find persons in the scene and at the best fitting scale. For NMS, we implemented an approximation of the mean-shift algorithm as 3D pooling. For algorithmic details we refer to [6].

For implementation, we used Keras [3] and Theano [35]. The network training was performed on a single NVIDIA GTX Titan X GPU in float32 precision.

3.2 Performance Analysis

The application phase on a NVIDIA GTX Titan X GPU took 0.231 seconds on average per image of size 640×480 if persons of a height of at least 80 pixels should be detected (reported in [6]). Persons, that are partly out of the image were not considered so far.

Hence, we extended this approach to also detect persons in front of the robot or pedestrians crossing the street in front of a car, where only the upper body is visible. This was achieved by zero padding the image below its bottom. The image size and, thus, also the computation time doubled. We also increased the maximum distance at which persons are detected. Therefore, the detector now searches for persons of height 75 – 927 pixels on 26 scales.

When applied on the Jetson TX1, the run-time increased by a factor of 18 to 8.4 seconds. In this paper, we explain how to significantly speedup this DL-based detection approach by optimizing it for application on the NVIDIA Jetson TX1 without a loss in accuracy.

To figure out, what caused the significant slowdown, we first analyzed the run-time of DL operations on the Jetson TX1 in comparison to the high performance Titan X GPU (see Sec. 4.1). Convolutions, that account for 90% of the computations, are up to 11 times slower on the Jetson platform. Although, this explains most of the observed slowdown, the overall slowdown factor is still a lot higher. Therefore, we searched for additional slowdown factors that should be eliminated.

3.3 Optimizing the Detector for Processing on a Jetson TX1

Our analyzes have shown, that copy operations between CPU and GPU created a large overhead. Although, the Jetson’s CPU and GPU share the same memory, making copy operations dispensable, the DL frameworks were not able to make use of this fact. To avoid unnecessary copy operations, we made sure, that everything is processed on the GPU, including non-DL operations such as construction of the image pyramid and postprocessing. Then, we optimized the computation graph in the Theano framework [35] by removing redundancy and by specifying exact shapes, which means, that all image and succeeding tensor sizes are set beforehand of processing. We can do this since the camera frame size will not change. These optimizations, that do not change any results, reduce the run-time by approximately 66%.

The run-time benchmarks of DL operations (see Fig. 4, Sec. 4.1) confirm, that float16 precision instead of float32 precision speeds up the computation

significantly. We therefore checked, if using float16 precision instead of float32 precision changes the results significantly. The only part of the CNN that is extremely sensitive to a lower floating point precision is the gradient during backprobagation. During training float16 precision would significantly worsen the results and, thus, cannot be applied. In the application phase we observed, that the accuracy of the classifier does not change. None of the linear operations (Fig. 4 (a) – (c), (e) – (l)) are sensitive to precision. However, the outputs after the softmax operation loose floating point precision due to the exponential operation. That means, that non-maximum suppression (NMS) becomes hard, if not impossible, since positions near the optimal location of a detected person get equal scores. Therefore, operations after softmax need special treatment. We used the network’s linear output before the softmax exponential operation for NMS. Using this trick, we observed only minor differences to the float32 version,

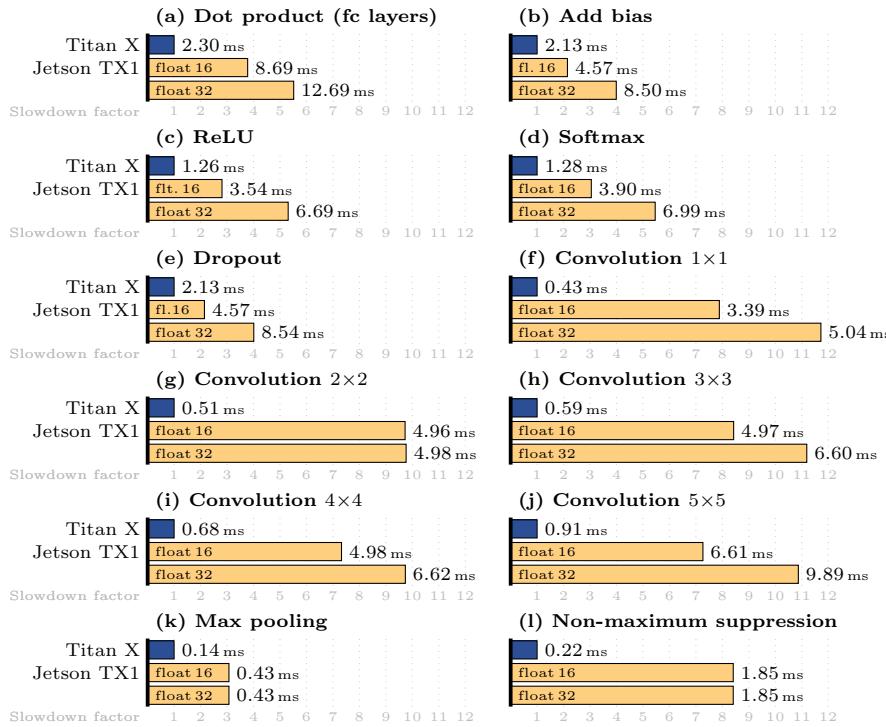


Fig. 4. Slowdown of typical Deep Learning operations on Jetson TX1 in comparison to GTX Titan X GPU. Absolute time measured in milliseconds is shown behind each bar. Times are averaged over 50 runs. (a)–(e) operations on 1000×1000 matrices, (f)–(j) valid convolutions on a 3-channel 320×240 image using 200 filters and a batch size of 1, (k) 2×2 max pooling on the same image size using stride 2×2 , no padding and a batch size of 1, (l) $3 \times 7 \times 3$ 3D max pooling for non-maximum suppression on the same image size using stride $1 \times 1 \times 1$, no padding and batch size 1.

while the accuracy did not change. Using float16 precision instead of float32 precision reduced the remaining run-time by approximately 25%.

To further speed up the computation, we introduced a ground plane constraint. By assuming that all persons must stand on the ground and by utilizing the extrinsic camera parameters, the image pyramid can be significantly reduced [17]. Since the relative position of the camera to the robot does not change over time, we can set all image sizes and succeeding tensor sizes beforehand. Again, when done properly, the accuracy does not decrease. Using this ground plane assumption, the remaining run-time was reduced by approximately 75%.

4 Experiments

4.1 Benchmarking Deep Learning on the Jetson TX1

To figure out, which DL operations mainly caused the significant slowdown on the Jetson TX1 in comparison to the high performance Titan X GPU, we first analyzed their run-time (Fig. 4).

Most critical are convolutions and rarely used and thus less optimized operations such as 3D pooling. Convolutions are significantly slower on the Jetson platform with a slowdown factor of up to 11. In our network, as in most other modern CNNs too, more than 90% of the computations are convolutions. This explains most of the observed slowdown. Fig. 4 also shows, that for large matrix and tensor sizes in common neural networks, all operations using float16 precision are faster than equivalent float32 precision operations.

4.2 Gained speedup

Tab. 1 shows the run-time for different stages of optimization. By optimizing the detector for processing on the Jetson TX1, we were able to speed up the run-time by a factor of 15 considering both preprocessing and DNN output computation (619.7 ms vs. 9,321.3 ms). At the same time, the accuracy did not decrease.

Table 1. Runtime of optimized detector on Jetson TX1

Detector	FloatX	Inp. scaling [ms]	Detection [ms]
Reference [6]	float32	930.1 ± 2.8	$8,391.2 \pm 15.3$
Optimizations			
No overhead & float16	float16	46.1 ± 3.2	$2,103.2 \pm 5.9$
+ Ground plane	float16	32.1 ± 0.3	587.6 ± 1.8

The steps described in Sec. 3.3 to reduce the run-time on an embedded platform are not specific for this detector, but can be applied to other Deep Learning approaches, too. Thus, we recommend to always check if the run-time can be reduced by:

- processing as much operations as possible on the GPU instead of the CPU to avoid overhead of copy operations,
- removing redundancies from the computation graph,
- specifying exact shapes to make all tensor sizes static,
- using float16 instead of float32 precision if accuracy does not drop,
- using assumptions to avoid needless computations.

4.3 Benchmarking the Person Detector on a Mobile Robot

We evaluated the person detector ported to this low power consuming small board on our mobile robot. Therefore, the Jetson TX1 was coupled to the main computer using its network interface, while data were exchanged using the robotics middleware MIRA [5]. In sum, data conversion, synchronization and communication for data transfer took only 3.36 ms per frame.

Tab. 2 shows the detection results on a recorded dataset [39].

Table 2. Detection performance of Computer Vision (CV) and Deep Learning (DL) approaches in a robotic application measured by miss rate (MR) for different false positives per image (FPPI). Additionally, the average number of frames that could be processed per second on a Jetson TX1 is reported.

Approach	Type	MR @ 0.1 FPPI	MR @ 0.01 FPPI	Frame rate
YOLO [27]	DL	0.386	0.918	12 [42]
Faster R-CNN [28]	DL	0.321	0.734	2–3 [22]
Part-based HOG [9]	CV	0.273	0.603	1.5–2
Proposed [6]	DL	0.115	0.287	1.6

The proposed multi-scale detector based on [6] clearly outperforms the other approaches in accuracy. The other Deep Learning detectors (YOLO, Faster R-CNN) perform relatively poor in comparison. Their performance is in the proximity of the best classical approaches. This was also observed by Zhang *et al.* [41] on the Caltech dataset. In their extensive analysis, they found, that a Region Proposal Network "specially tailored for pedestrian detection achieves competitive results as a stand-alone pedestrian detector." But surprisingly, the accuracy is degraded after feeding these proposals into the Fast R-CNN classifier." One reason is, that small objects are not detected appropriately. We observed the same in our experiments. The second reason is the presence of unseen hard negative examples that trick most detectors to false detections. This issue can only be solved by training on a dataset including these difficulties, as we did in our previous work for the baseline CNN [6].

If a higher processing speed is desired, the detection can be divided and distributed onto multiple Jetson TX1. Four of these devices would still require less power than a single PC with a powerful CPU and would also require less space.

5 Conclusion

The small and low power consuming NVIDIA Jetson TX1 platform with a powerful GPU onboard makes it possible to apply top performing Deep Learning approaches on an autonomous car or a mobile robot. Thus, all DL solutions can be processed onboard. Exemplary, we showed how to port a DL-based person detector to this platform. Furthermore, we showed how to speed up the detector's run-time by factor 15. The result is a top performing DL-based person detector fast enough to replace the currently used CPU-based classical computer-vision-based detector on our robot. Our benchmark of typical DL operations will help other researches to estimate the run-time of Deep Learning approaches when applied on a Jetson TX1. Additionally, we have presented a list of generally applicable optimizations to speedup the computation on that device. In the qualitative evaluation on a robotic person detection benchmark dataset, the proposed detector clearly outperforms the state of the art including the popular DL-based object detectors YOLO and Faster R-CNN.

References

1. Campmany, V., Silva, S., Espinosa, A., Moure, J.C., Vázquez, D., López, A.M.: Gpu-based pedestrian detection for autonomous driving. In: Int. Conf. on Computational Science (ICCS). pp. 2377–2381. Elsevier (2016)
2. Carraro, M., Munaro, M., Menegatti, E.: A powerful and cost-efficient human perception system for camera networks and mobile robotics. In: Int. Conf. on Intelligent Autonomous Systems (IAS). pp. 485–497. Springer (2016)
3. Chollet, F.: Keras. <https://github.com/fchollet/keras> (2015)
4. Dollar, P., Wojek, C., Schiele, B., Perona, P.: Pedestrian detection: An evaluation of the state of the art. Pattern Analysis and Machine Intelligence, IEEE Transactions on 34(4), 743–761 (2012)
5. Einhorn, E., Langner, T., Stricker, R., Martin, Ch., Gross, H.M.: MIRA – middleware for robotic applications. In: Int. Conf. on Intelligent Robots and Systems (IROS). pp. 2591–2598. IEEE (2012)
6. Eisenbach, M., Seichter, D., Wengfeld, T., Gross, H.M.: Cooperative multi-scale convolutional neural networks for person detection. In: World Congress on Computational Intelligence (WCCI). pp. 267–276. IEEE (2016)
7. Eisenbach, M., Vorndran, A., Sorge, S., Gross, H.M.: User recognition for guiding and following people with a mobile robot in a clinical environment. In: Int. Conf. on Intelligent Robots and Systems (IROS). pp. 3600–3607. IEEE (2015)
8. Farabet, C., Couprie, C., Najman, L., LeCun, Y.: Learning hierarchical features for scene labeling. Transactions on Pattern Analysis and Machine Intelligence (TPAMI) 35(8), 1915–1929 (2013)
9. Felzenszwalb, P.F., Girshick, R.B., McAllester, D., Ramanan, D.: Object detection with discriminatively trained part-based models. Trans. on Pattern Analysis and Machine Intelligence (TPAMI) 32(9), 1627–1645 (2010)
10. Gong, Z., Li, J., Li, W.: A low cost indoor mapping robot based on tinyslam algorithm. In: Int. Geoscience and Remote Sensing Symposium (IGARSS). pp. 4549–4552. IEEE (2016)

11. Gross, H.M., Meyer, S., Scheidig, A., Eisenbach, M., Mueller, S., Trinh, T.Q., Wengfeld, T., Bley, A., Martin, C., Fricke, C.: Mobile robot companion for walking training of stroke patients in clinical post-stroke rehabilitation. In: Int. Conf. on Robotics and Automation (ICRA). IEEE (2017)
12. Gross, H.M., Boehme, H.J., Schroeter, Ch., Mueller, St., Koenig, A., Einhorn, E., Martin, Ch., Merten, M., Bley, A.: TOOMAS: Interactive shopping guide robots in everyday use – final implementation and experiences from long-term field trials. In: Int. Conf. on Intelligent Robots and Systems (IROS). pp. 2005–2012. IEEE (2009)
13. Gross, H.M., Mueller, St., Schroeter, Ch., Volkhardt, M., Scheidig, A., Debes, K., et al.: Robot companion for domestic health assistance: Implementation, test and case study under everyday conditions in private apartments. In: Int. Conf. on Intelligent Robots and Systems (IROS). pp. 5992–5999. IEEE (2015)
14. Hernandez-Juarez, D., Espinosa, A., Vázquez, D., López, A.M., Moura, J.C.: Gpu-accelerated real-time stixel computation. arXiv preprint arXiv:1610.04124 (2016)
15. Hinton, G., Vinyals, O., Dean, J.: Distilling the knowledge in a neural network. In: Neural Information Processing Systems (NIPS) Workshop: Deep Learning and Representation Learning (2014)
16. Kim, J., Yu, S.C.: Convolutional neural network-based real-time rov detection using forward-looking sonar image. In: Autonomous Underwater Vehicles (AUV). pp. 396–400. IEEE/OES (2016)
17. Kolarow, A., Schenk, K., Eisenbach, M., Dose, M., Brauckmann, M., Debes, K., Gross, H.M.: APFel: The intelligent video analysis and surveillance system for assisting human operators. In: Int. Conf. on Advanced Video and Signal-Based Surveillance (AVSS). pp. 195–201. IEEE (2013)
18. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems (NIPS). pp. 1097–1105 (2012)
19. Laugier, C., Chartre, J., Perrollaz, M., Stein, S., et al.: Intelligent perception and situation awareness for automated vehicles. In: Conf. GTC Europe (2016)
20. Lee, Y., Kim, H., Park, E., Cui, X., Kim, H.: Wide-residual-inception networks for real-time object detection. arXiv preprint arXiv:1702.01243 (2017)
21. Leroux, S., Bohez, S., Verbelen, T., Vankeirsbilck, B., Simoens, P., Dhoedt, B.: Resource-constrained classification using a cascade of neural network layers. In: Neural Networks (IJCNN), 2015 International Joint Conference on. pp. 1–7. IEEE (2015)
22. Mhalla, A., Gazzah, S., Ben Amara, N.E., et al.: A faster r-cnn multi-object detector on a nvidia jetson tx1 embedded system: Demo. In: Int. Conf. on Distributed Smart Camera (ICDSC). pp. 208–209. ACM (2016)
23. Nair, V., Hinton, G.E.: Rectified linear units improve restricted boltzmann machines. In: Int. Conf. on Machine Learning (ICML). pp. 807–814 (2010)
24. Orozco-Rosas, U., Montiel, O., Sepúlveda, R.: Embedded implementation of a parallel path planning algorithm based on eapf for mobile robotics. In: Congreso Internacional en Ciencias Computacionales (CiComp). pp. 178–184 (2016)
25. Otterness, N., Yang, M., Rust, S., Park, E., Anderson, J.H., Smith, F.D., Berg, A., Wang, S.: An evaluation of the nvidia tx1 for supporting real-time computer-vision workloads. In: Real-Time and Embedded Technology and Applications Symposium (RTAS). IEEE (2017)
26. Rallapalli, S., Qiu, H., Bency, A.J., Karthikeyan, S., Govindan, R.: Are very deep neural networks feasible on mobile devices? Tech. rep., University of Southern California, TR 16-965 (2016)

27. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: Unified, real-time object detection. In: Conf. on Computer Vision and Pattern Recognition (CVPR). pp. 779–788. IEEE (2016)
28. Ren, S., He, K., Girshick, R., Sun, J.: Faster R-CNN: Towards real-time object detection with region proposal networks. In: Advances in Neural Information Processing Systems (NIPS). pp. 91–99 (2015)
29. Rud, M.N., Pantiykchin, A.R.: Development of gpu-accelerated localization system for autonomous mobile robot. In: Int. Conf. on Mechanical Engineering, Automation and Control Systems (MEACS). pp. 1–4. IEEE (2014)
30. Sansen, H., Torres, M.I., Chollet, G., Glackin, C., Petrovska-Delacretaz, D., Boudy, J., Badii, A., Schlägl, S.: The roberta ironside project: A dialog capable humanoid personal assistant in a wheelchair for dependent persons. In: Int. Conf. on Advanced Technologies for Signal and Image Processing (ATSIP). pp. 381–386. IEEE (2016)
31. Satria, M.T., Gurumani, S., Zheng, W., Tee, K.P., Koh, A., Yu, P., Rupnow, K., Chen, D.: Real-time system-level implementation of a telepresence robot using an embedded gpu platform. In: Design, Automation & Test in Europe Conf. & Exhibition (DATE). pp. 1445–1448. IEEE (2016)
32. Shimchik, I., Sagitov, A., Afanasyev, I., Matsuno, F., Magid, E.: Golf cart prototype development and navigation simulation using ros and gazebo. In: Int. Conf. on Measurement Instrumentation and Electronics (ICMIE). vol. 75. EDP Sciences (2016)
33. Silva, G., Reátegui, J., Rodriguez, P.: Fast omni-image unwarping on the jetson tk1. In: GPU Technical Conf. (GTC) (2015)
34. Srivastava, N., Hinton, G.E., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. Journal of Machine Learning Research 15(1), 1929–1958 (2014)
35. Theano Development Team: Theano: A Python framework for fast computation of mathematical expressions. arXiv preprint arXiv:1605.02688 (2016)
36. Treml, M., Arjona-Medina, J., Unterthiner, T., Durgesh, R., Friedmann, F., Schubert, P., Mayr, A., Heusel, M., Hofmarcher, M., Widrich, M., Bodenhofer, U., Nessler, B., Hochreiter, S.: Speeding up semantic segmentation for autonomous driving. In: Neural Information Processing Systems (NIPS) (2016)
37. Trinh, Th.Q., Schroeter, Ch., Gross, H.M.: "go ahead, please": Recognition and resolution of conflict situations in narrow passages for polite mobile robot navigation. In: Int. Conf. on Social Robotics (ICSR). pp. 643–653. Springer (2015)
38. Weinrich, Ch., Volkhardt, M., Einhorn, E., Gross, H.M.: Prediction of human collision avoidance behavior by lifelong learning for socially compliant robot navigation. In: Int. Conf. on Robotics and Automation (ICRA). pp. 376–381. IEEE (2013)
39. Wengfeld, T., Eisenbach, M., Trinh, Th.Q., Gross, H.M.: May i be your personal coach? bringing together person tracking and visual re-identification on a mobile robot. In: Int. Symposium on Robotics (ISR). pp. 141–148. VDE (2016)
40. Yu, C.H., Lee, T., Chen, C.W., Hsieh, M.R., Fuh, C.S.: Human segmentation with nvidia-tx1. In: Conf. on Computer Vision, Graphics, and Image Processing (CVGIP). pp. 1–8. IPPR (2016)
41. Zhang, L., Lin, L., Liang, X., He, K.: Is faster r-cnn doing well for pedestrian detection? In: European Conf. on Computer Vision (ECCV). pp. 443–457. Springer (2016)
42. Zhang, W., Zhao, D., Xu, L., Li, Z., Gong, W., Zhou, J.: Distributed embedded deep learning based real-time video processing. In: Int. Conf. on Systems, Man, and Cybernetics (SMC). pp. 1945–1950. IEEE (2016)

Software Framework for Morphological Neural Network Computations

Bogusław Cyganek

AGH University of Science and Technology
Al. Mickiewicza 30, 30-059 Kraków, Poland
cyganek@agh.edu.pl

Abstract. In the paper we present architecture and functions of the software framework for morphological neural network computations. This type of networks shows very usable features especially if real-time training and response are required. In the paper the basic information on the C++ software for morphological neural networks is presented, whereas the whole software framework has been made accessible from the Internet.

Keywords: Morphological neural-network, computer vision, real-time computations

1 Introduction

Neural computations for artificial intelligence belong to the fastest growing areas of research, as well as industrial and social applications [1][3][4][5]. A real breakthrough was development of deep neural architectures which outperform other classifiers in terms of accuracy [8][1]. However, despite their excellent accuracy they show many drawbacks from which the most severe is very high computational demands [9]. On the other hand, there are many artificial intelligence applications which are aimed to operate on embedded or robot platforms. In such cases, the morphological neural networks exhibit many benefits, especially in terms of learning and run-time performance. Also interesting is to consider other types of deep architectures – for example, deep architectures with the morphological neural networks. For this purpose we developed a software framework for easy processing of the morphological neural networks (MNN). In this paper we present and discuss the basic properties of this framework, as well as present its main benefits.

2 Introduction to the Morphological Neural Architectures

The most outstanding characteristic feature of the morphological neural networks is utilization of exclusively summation and min and max operators in the basic form of the morphological neuron. When compared with the perceptron, in which there are multiplications, summations, as well as nonlinear functions, the former offer much faster computations. As shown by Ritter, these operations – based on the lattice theory

– are sufficient to define neural networks that offer many interesting properties [13]. Although, MNN now are in somehow recession due to enthusiasm associated with deep architectures, as already mentioned, these can be considered as building blocks of deep architectures as well. This subject belongs to one of our research directions.

After their proposition by Ritter, MNN found interest among other researchers showing some useful properties. They were shown to operate well in associative and auto-associative memories [11], for image denoising [12], as well as useful and very fast classifiers [10][14][15]. For example Villaverde *et al.* propose MNN in the problem of simultaneous localization and mapping (SLAM), which is a key task in robotics [16]. They compute the indoor non-metric SLAM problem with help of the visual information obtained from the morphologically independent images. It appears that these correspond to the vertices of the convex hull encompassing data points in a high dimensional space. MNN were also applied by Cyganek to the classification of binary pictograms for the road signs recognition [6].

Fig. 1 shows a model of a morphological neuron – as in majority of the neural networks, there is a number of L input signals and one output. However, the main difference lies in the performed operations. That is, the input signals are summed together with the weights, and the maximum value of this is then computed.

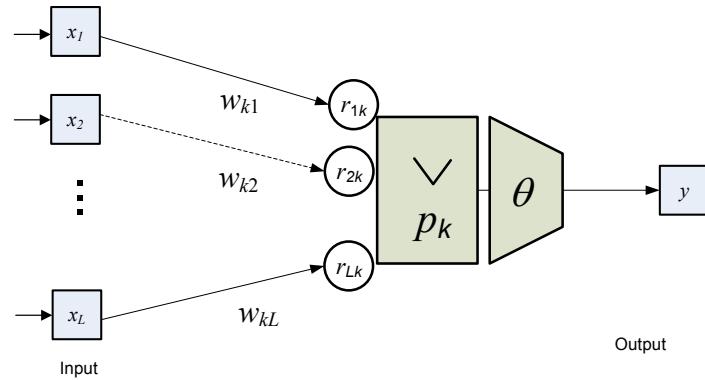


Fig. 1. View of a morphological neuron.

More specifically, the morphological neuron operates in accordance with the following expression

$$y_k = \theta \left(p_k \bigvee_{i=1}^L r_{ik} (x_i + w_{ik}) \right), \quad (1)$$

where r_{ik} denote a pre-synaptic response which transfers excitatory ($r_{ik}=+1$), or inhibitory ($r_{ik}=-1$), incitation of an i -th neuron, p_k is the post-synaptic response of a k -

th neuron to the total input signal, whereas the symbol \vee denotes a *max* product, then θ stands for a saturation function. However, the above equation is then further simplified. For example, values of r_{ik} and p_k are positive.

Let us take a closer look at the most important operations of the MNN. The *max* product \vee for two matrices \mathbf{A}_{pq} and \mathbf{B}_{qr} is a matrix \mathbf{C}_{pr} , with elements c_{ij} is defined as

$$c_{ij} = \bigvee_{k=1}^q (a_{ik} + b_{kj}). \quad (2)$$

Analogously, the *min* operator \wedge is defined as follows

$$c_{ij} = \bigwedge_{k=1}^q (a_{ik} + b_{kj}). \quad (3)$$

Thus, comparing the well-known model of perceptron with the morphological neuron we easily notice the exchange of multiplication of inputs x_i with the synaptic weights into their summation, as well as summation of these products into their max value.

Fig. 2 shows the MNN proposed in [6] for classification of binary pictograms. This is a type of a morphological associative memory in which a set of i input/output pairs is given in the form: $(\mathbf{x}_1, \mathbf{y}_1), \dots, (\mathbf{x}_N, \mathbf{y}_N)$. The pattern \mathbf{x}_i of dimensions 256x256, is a linear version of an image of a sign, in this case, and \mathbf{y} is binary version of a pattern's class. All the inputs are gathered in the input layer \mathbf{X} .

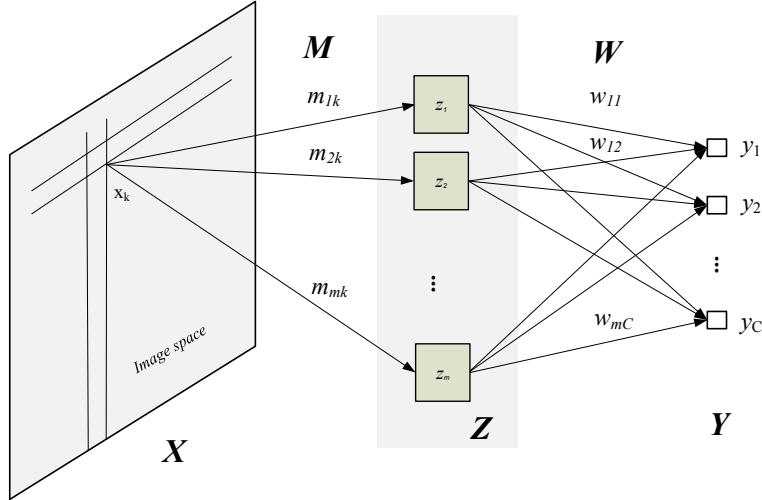


Fig. 2. Exemplary architecture of an associative morphological neural network for visual patterns recognition.

On the other hand, vectors \mathbf{y} decode classes of the pictograms in the one-of-N code. Then, from the pairs $(\mathbf{x}_i, \mathbf{y}_i)$, the matrices $\mathbf{X}=(\mathbf{x}_1, \dots, \mathbf{x}_N)$ and $\mathbf{Y}=(\mathbf{y}_1, \dots, \mathbf{y}_N)$ are constructed. In the case of one layer, a matrix \mathbf{W} or \mathbf{M} of weights is determined from \mathbf{X} and \mathbf{Y} as follows [13][6][3]

$$\mathbf{W}_{XY} = \bigwedge_{i=1}^N \left(\mathbf{y}_i \times (-\mathbf{x}_i)^T \right), \quad (4)$$

and

$$\mathbf{M}_{XY} = \bigvee_{i=1}^N \left(\mathbf{y}_i \times (-\mathbf{x}_i)^T \right). \quad (5)$$

In the above equations, T denotes transposition of a vector, the symbol \times denotes the so called morphological outer product of vectors, which is defined as follows [13][14]

$$\mathbf{y} \times \mathbf{x}^T = \begin{bmatrix} y_1 + x_1 & \cdots & y_1 + x_n \\ \vdots & \ddots & \vdots \\ y_m + x_1 & \cdots & y_m + x_n \end{bmatrix}. \quad (6)$$

It is interesting to observe that the above is analogous to the outer product of vectors, in which addition is substituted for multiplication, however. Now, for any real number a its additive conjugate is given as follows

$$a^* = -a. \quad (7)$$

Thus, let us also observe that for all $a, b \in \Re$, the following holds

$$a \wedge b = (a^* \vee b^*)^*. \quad (8)$$

With \mathbf{W} and \mathbf{M} defined in (4)-(5) the following hold

$$\mathbf{W}_{XY} \vee \mathbf{x}_i = \mathbf{y}_i, \quad (9)$$

$$\mathbf{M}_{XY} \wedge \mathbf{x}_i = \mathbf{y}_i, \quad (10)$$

which constitute the basic operation of associative neural memories. As was shown, the above hold even for erosively or dilatively distorted versions $\tilde{\mathbf{x}}_i$ of \mathbf{x}_i . That is, a perfect recall is guaranteed if there are some pixels in a prototype image \mathbf{x} which values are greater for \mathbf{W} (or lower for \mathbf{M}) in this image from the maximum of all the remaining patterns \mathbf{x}_i stored in the network. For binary images this means a unique pixel with

value ‘1’ in each of the prototypes \mathbf{x}_i . This feature, in turn, is connected with the concept of the morphological independence, as well as the strong independence [11-13].

Further, to make MNN robust to random, i.e. dilative *and* erosive noise, a kernel version of MNN was proposed by Ritter [11]. The idea is to replace the associative memory \mathbf{W} or \mathbf{M} with *a series* of two memories \mathbf{M}' and \mathbf{W}' which are connected by the intermediate pattern \mathbf{Z} , which is called a kernel. Such a computation scheme is also assumed in the network shown in Fig. 2. It operates as follows: input- \mathbf{M}'_{ZZ} - \mathbf{W}'_{ZY} -output. This pattern of operation can be written as follows

$$\mathbf{W}'_{ZY} \vee \underbrace{\left(\mathbf{M}'_{ZZ} \wedge \tilde{\mathbf{x}}_i \right)}_{\mathbf{z}_i} = \mathbf{y}_i . \quad (11)$$

In the above, $\tilde{\mathbf{x}}_i$ denotes a randomly corrupted input pattern. The matrices \mathbf{M}'_{ZZ} and \mathbf{W}'_{ZY} in (11) are found in accordance with the previously described mechanisms, however for different matrices and after finding a kernel \mathbf{Z} . Construction of the kernel \mathbf{Z} is outlined by the theorems provided in the papers [11][14].

3 Software Framework for Morphological Neural Computations

Fig. 3 depicts class hierarchy of our software component. Shown classes *TMorphoOperationsFor* and *TMorphoNet*, which implement basic operations of lattice algebra and the morphological associative network, respectively. Visible is also the *MorphMatrix* class which is a special version of the common *TImageFor* class. All these were written into the more general DeRecLib, accessible from the Internet [10].

The methods were implemented in C++. The experiments were run on a laptop computer equipped with the Intel® Xeon® E-1545 CPU @2.9GHz, 64GB RAM, and OS 64-bit Windows 10.

Majority of operations of the *TMorphoOperationsFor* class are used in implementation of the MNN, implemented by the *TMorphoNeuralNet* class. These follow operations expressed in equations (9)-(11). For this purpose the object of the *TMorphoNeuralNet* class stores the matrices \mathbf{W} , \mathbf{M} , and the \mathbf{Z} respectively, as shown in Fig. 3.

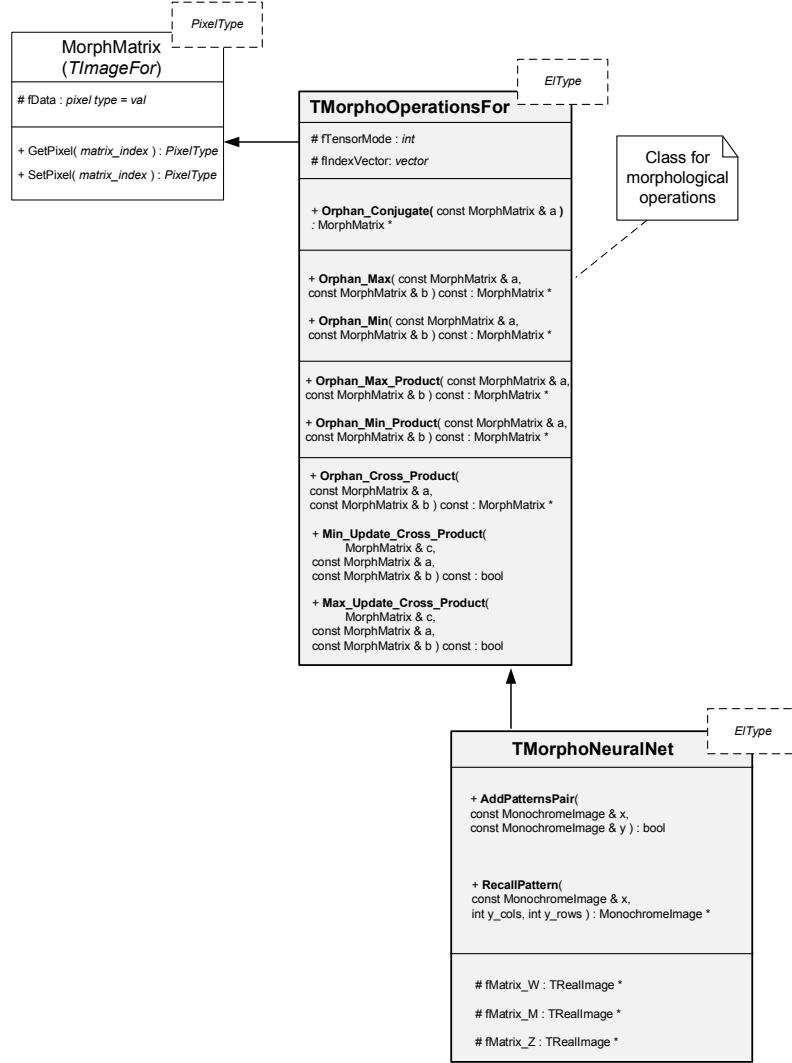


Fig. 3. Class hierarchy of the *TMorphoOperationsFor* and *TMorphoNet* classes which implement basic operations of lattice algebra and the morphological associative network, respectively.

The principal members of the *TMorphoOperationsFor* class are the *AddPatternsPair* and *RecallPattern* functions. The former is responsible for network training and enters a new associative pair of patterns *x* and *y* into the network. The latter, retrieves an associated pattern to the shown one. The auxiliary parameters for the

RecallPattern function are dimensions of the image which will be reconstructed from its vector-like version.

Algorithm 1 shows test code for morphological neural network. These simple tests were performed to check basic properties of the MNN. In this case, for training we used test images shown in Fig. 4.

```
void MorphoTest( void )
{
    TMorphoNeuralNet MNN;

    int num_of_patterns = 3;

    TIFF_Converters TIFF_Wizard;
    MonochromeImage * image_x_1 =
        TIFF_Wizard.OrphanMonochromeImageFrom_TIFF_File
        ( _T("Lenna_part_1.tiff") );

    MonochromeImage * image_x_2 =
        TIFF_Wizard.OrphanMonochromeImageFrom_TIFF_File
        ( _T("Lenna_part_2.tiff") );

    MonochromeImage * image_x_3 =
        TIFF_Wizard.OrphanMonochromeImageFrom_TIFF_File
        ( _T("Lenna_part_3.tiff") );

    CreateNewViewWindowAndSetImage( * image_x_1, _T("Input pattern 1") );
    CreateNewViewWindowAndSetImage( * image_x_2, _T("Input pattern 2") );
    CreateNewViewWindowAndSetImage( * image_x_3, _T("Input pattern 3") );

    int x_cols = image_x_1->GetCol();
    int x_rows = image_x_1->GetRow();

    // This memory works fine for autoassociative memories
    MonochromeImage image_y_1( * image_x_1 );
    MonochromeImage image_y_2( * image_x_2 );
    MonochromeImage image_y_3( * image_x_3 );

    bool add_ret_val = false;

    add_ret_val = MNN.AddPatternsPair( * image_x_1, image_y_1 );
    add_ret_val = MNN.AddPatternsPair( * image_x_2, image_y_2 );
    add_ret_val = MNN.AddPatternsPair( * image_x_3, image_y_3 );

    // Ok, let's see what we recall...

    MonochromeImage * y_recalled;

    /////////////////////////////////
    // Here we warp an image and test network response
}
```

```

AffinelyWarp( * image_x_1, -2, 0.0, 0.0, 1.0, 1.0 );
CreateNewViewWindowAndSetImage( * image_x_1, _T("Warped pattern 1") );

y_recalled = MNN.RecallPattern( * image_x_1, x_cols, x_rows );

if( y_recalled != 0 )
    CreateNewViewWindowAndSetImage( * y_recalled, _T("1 of MNN") );
delete y_recalled;

///////////////////////////////
AffinelyWarp( * image_x_2, 2, 0.0, 0.0, 1.0, 1.0 );
CreateNewViewWindowAndSetImage( * image_x_2, _T("Warped pattern 2") );

y_recalled = MNN.RecallPattern( * image_x_2, x_cols, x_rows );

if( y_recalled != 0 )
    CreateNewViewWindowAndSetImage( * y_recalled, _T("2 of MNN") );
delete y_recalled;

///////////////////////////////
AffinelyWarp( * image_x_3, 5, 0.0, 0.0, 1.0, 1.0 );
CreateNewViewWindowAndSetImage( * image_x_3, _T("Warped pattern 3") );

y_recalled = MNN.RecallPattern( * image_x_3, x_cols, x_rows );

if( y_recalled != 0 )
    CreateNewViewWindowAndSetImage( * y_recalled, _T("3 of MNN") );
delete y_recalled;
///////////////////////////////

delete image_x_1;
delete image_x_2;
delete image_x_3;

}

```

Algorithm 1. Test code for morphological neural network.

Fig. 4 shows images used in tests of the associative morphological neural network in our software framework. These are parts of the famous *Lena* image, frequently used for testing of different image processing procedures. On the other hand, testing was done with affinely deformed versions of the input images. The network always responded with a proper class.



Fig. 4. Test images used in tests of the associative morphological neural network in our software framework.

Both, training *and* response of the network are very fast and allow real-time operation even with image patterns in HD resolution. In our implementation, these were in order of few milliseconds. Hence, in many practical cases the morphological neural networks can pose a real substitute for more demanding neural algorithms when run on embedded or robotic platforms. For instance, MNN can be considered in embedded AI systems. Also, its operations can further benefit from parallel implementations [19].

4 Conclusions

In the paper, the software framework for real-time training and run-time of the morphological neural networks is presented. We show that despite a tremendous impact of deep neural architectures they still show a number of drawbacks, from which the computational requirements can be prohibitive for small factor embedded or robotic platforms. As an alternative, the morphological neural networks can be used. In this paper we present an object-oriented software platform to operate this type of networks. The software is also available from the Internet [7].

Acknowledgement

This work was supported by the National Science Centre, Poland, under the grant no. 2016/21/B/ST6/01461.

References

1. Bengio, Y., Goodfellow, I. J., Courville, A.: Deep learning. An MIT Press, 2015.
2. Bovik, A. C.: Handbook of image and video processing. Academic Press, 2010.
3. Cyganek, B., Object Detection and Recognition in Digital Images: Theory and Practice, Wiley, 2013.
4. Cyganek B., Gruszczyński S., Hybrid Computer Vision System for Drivers' Eye Recognition and Fatigue Monitoring, Neurocomputing, Vol. 126, pp. 78–94, 2014.
5. Cyganek B.: An Analysis of the Road Signs Classification Based on the Higher-Order Singular Value Decomposition of the Deformable Pattern Tensors. Advanced Concepts for Intelligent Vision Systems Acivs 2010, LNCS 6475 Springer, pp. 191–202, 2010.
6. Cyganek B.: Neuro-Fuzzy System for Road Signs Recognition. Lecture Notes in Computer Science LNCS 5163, Springer, pp. 503-512, 2008
7. <http://home.agh.edu.pl/~cyganek/Projects.zip>
8. Krizhevsky, A., Sutskever, I., & Hinton, G. E.: Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems, pp. 1097-1105, 2012.
9. Kozierski M., Cyganek B.: Deep Neural Image Denoising, International Conference on Computer Vision and Graphics ICCVG 2016: Computer Vision and Graphics, September 19-21, pp. 163-173, 2016.
10. Raducanu B., Graña M., Albizuri F.X.: Morphological Scale Spaces and Associative Morphological Memories: Results on Robustness and Practical Applications. Journal of Mathematical Imaging and Vision, Vol. 19, pp. 113-131, 2003.
11. Ritter, G., X., Sussner, P., Diaz, J., L.: Morphological Associative Memories, IEEE Transactions on Neural Networks, Vol. 9, No. 2, pp. 281-293, 1998.
12. Ritter, G.,X., Urcid, G., Iancu, L.: Reconstruction of Patterns from Noisy Inputs Using Morphological Associative Memories. Journal of Mathematical Imaging and Vision 19, pp. 95–111, 2003.
13. Ritter, G.,X., Iancu, L.: A Lattice Algebraic Approach to Neural Computation, in Handbook of Geometric Computing. Applications in Pattern Recognition, Computer Vision, Neuralcomputing, and Robotics, edited by Corrochano E.B., Springer, pp. 97-127, 2005.
14. Sussner P.: Observations on Morphological Associative Memories and the Kernel Method. Neurocomputing , Vol. 31, Elsevier Science, pp. 167-183, 2000.
15. Sussner P., Valle M.E.: Gray-Scale Morphological Associative Memories. IEEE Transactions on Neural Networks, Vol. 17, No. 3, pp. 559-570, 2006.
16. Villaverde I., Graña M., d'Anjou A.: Morphological neural networks and vision based simultaneous localization and mapping. Integrated Computer-Aided Engineering, Vol. 14, No. 4, pp. 355-363, 2007.
17. Woźniak M.: A hybrid decision tree training method using data streams. Knowl. Inf. Syst. 29(2), pp. 335–347, 2011.
18. Woźniak, M., Grana, M., Corchado, E.: A survey of multiple classifier systems as hybrid systems. Information Fusion 16(1), pp. 3–17, 2014.
19. <http://www.openmp.org/>
20. Zadeh L.A., Kacprzyk J.: Fuzzy logic for the management of uncertainty. John Wiley & Sons, Inc. New York, NY, USA, 1992.

Evaluation of background subtraction methods in thermal imaging

Mateusz Knapik¹, Bogusław Cyganek²

¹ AGH University of Science and Technology
Al. Mickiewicza 30, 30-059 Kraków, Poland
mateuszknapi@gmail.com

² AGH University of Science and Technology
Al. Mickiewicza 30, 30-059 Kraków, Poland
cyganek@agh.edu.pl

Abstract. The study evaluates possibility of use of real-time background detection based on the PCA with fast eigen-decomposition method [1] when used in thermal imagery. It shows how algorithms designed for classic vision systems perform when applied to long-infrared range of the electromagnetic spectrum.

Keywords: thermal imaging, computer vision, background subtraction.

1 Introduction

Thermal imaging in recent years gains popularity, both in industrial solutions and research projects. Thanks to technological developments low-cost thermal imaging temperature sensors starts to become available for scientists and automation engineers [4].

Using long-infrared thermal imaging devices over classic cameras allows to produce usable image even in total darkness and/or extreme lightning conditions. This allows to simplify further processing, because input data stays relatively coherent over wide range of situations.

In this paper we try to evaluate if algorithm designed for background subtraction (BS) in visual light images can be applied to thermal images without major modifications. For this purpose real-time background detection algorithm based on the PCA with fast eigen-decomposition method proposed by Cyganek and Woźniak [1].

2 Related work

Earlier works in this area [2] exploited image characteristics of thermal halos that are visible on images produced by ferroelectric BST sensors to find regions-of-interests (ROIs) and amplify salient gradient information. Approach used in this paper allows usage of uncooled microbolometer thermal sensors, which do not produce the haloing effect [6].

3 Test equipment and methodology

For the purpose of this study five thermal video sequences were recorded, both indoor (3 sequences) and outdoor (2 sequences).

Recording was carried out using a FLIR A35 thermal imaging camera [5]. This device streams 320 x 256 thermal images at up to 60 frames per second over Gigabit Ethernet connection. It allows to measure object temperature ranging from -25 to +135 degrees Celsius, with accuracy of 5% of reading. Custom written software was used to automate initial camera configuration and to allow saving 14-bit raw images to 16-bit TIFF files.

From every sequence 100 frames is used as a training data for the BS algorithm. Then 1 frame was chosen as a test frame for algorithm, to compute it's background. Images are processed with real-time background detection algorithm based on the PCA with fast eigen-decomposition method and then compared with manually labeled ground-truth image for that frame to provide quantitative comparison.

4 Experimental results

The code for background subtraction method is the same as used in [1], only minor changes were made to allow usage of 14-bit raw images. The experiments were run on a laptop computer equipped with the Intel® Core i7® 6700HQ CPU @2.6GHz, 16GB RAM, and OS 64-bit Windows 10.

The results for the *Hands*, *Notebook*, *Office*, *Car* and *BirdInTheSky* are shown in Figure 1, Figure 2, Figure 3, Figure 4 and Figure 5 respectively. Quantitative result is the F number, which is computed based on true-positive (TP), true-negative(TN), false-positive (FP) and false-negative (FN) using procedures described in [3].

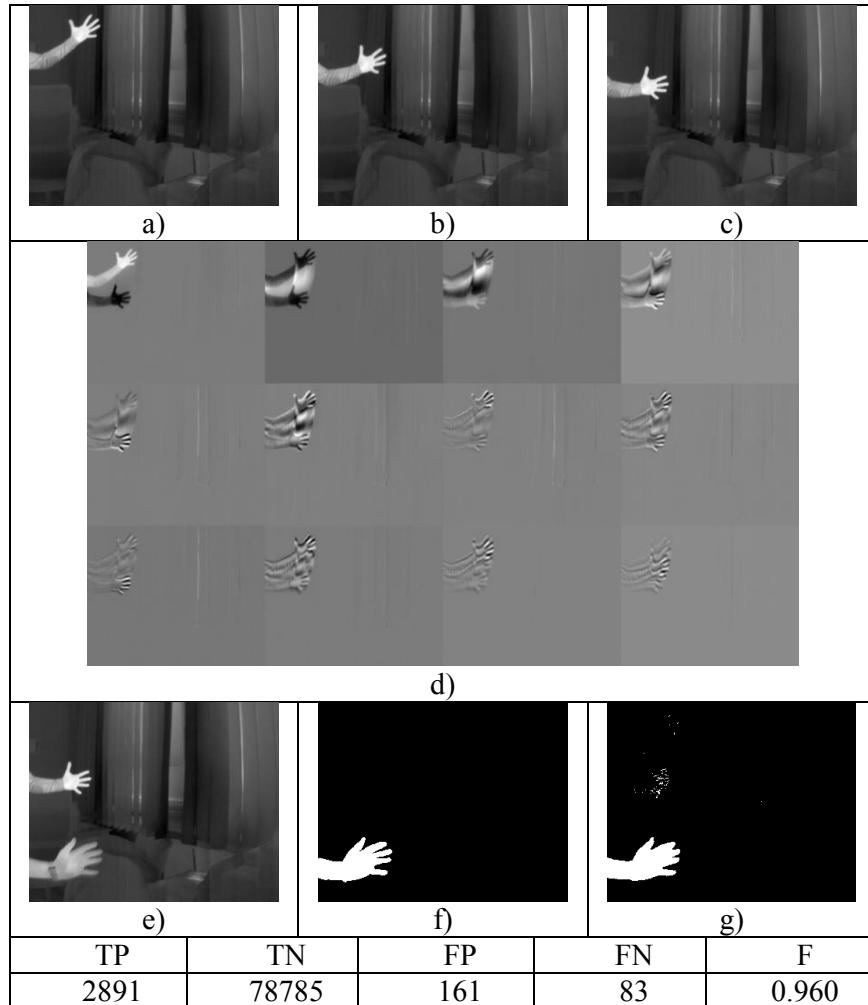


Fig. 1. Results obtained from *Hands* test sequence. Examples from the training data (a)-(c). Eigenimages corresponding to the largest eigenvalues (d). Test frame (e), background ground-truth for the test frame (f), obtained background (g). Below there are results True-Positive/True-Negative/False-Positive/False-Negative and the F score.

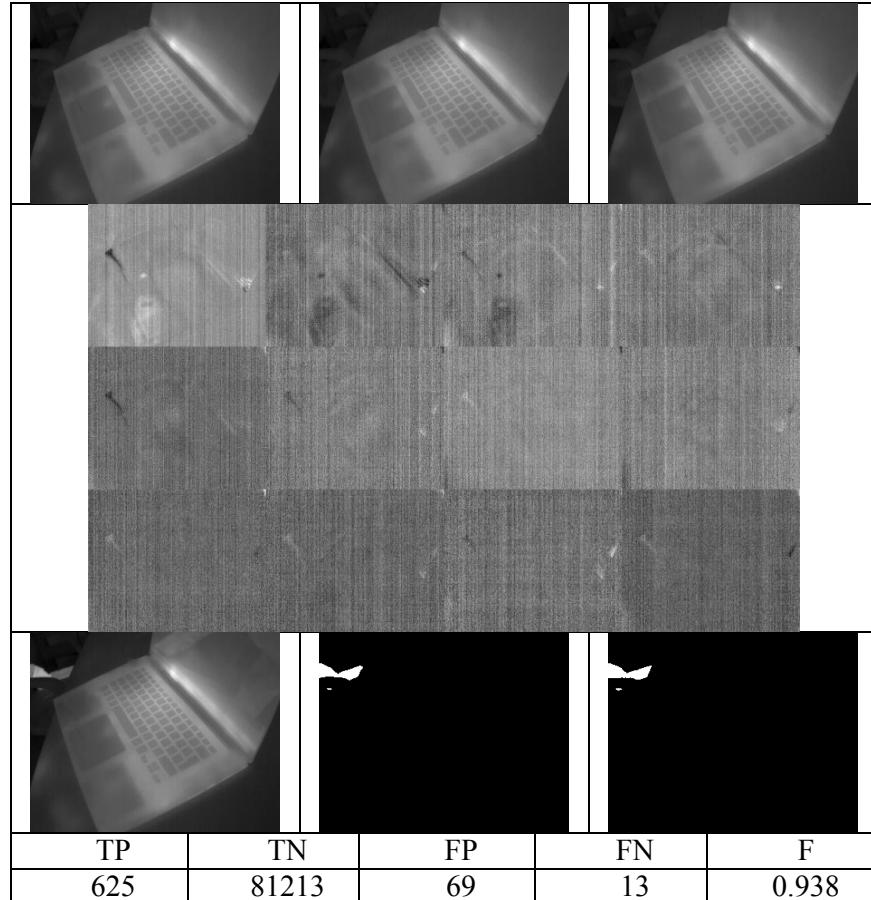
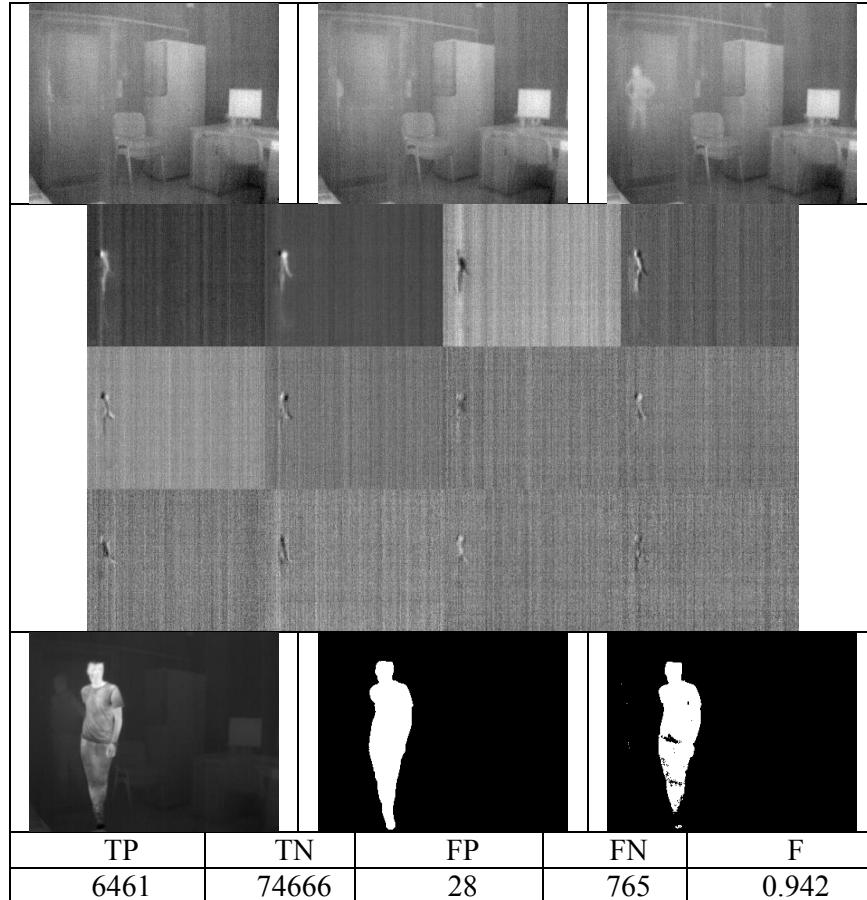


Fig. 2. Results obtained from *Notebook* test sequence.

**Fig. 3.** Results obtained from *Office* test sequence.

Tests performed indoor shows that separation of elements significantly warmer than surrounding background, like people or electronic devices is performed very well, without exhibiting too much irregularities. Additionally, as shows test sequence named Hands, background subtraction is undisturbed by introducing background model with moving object.



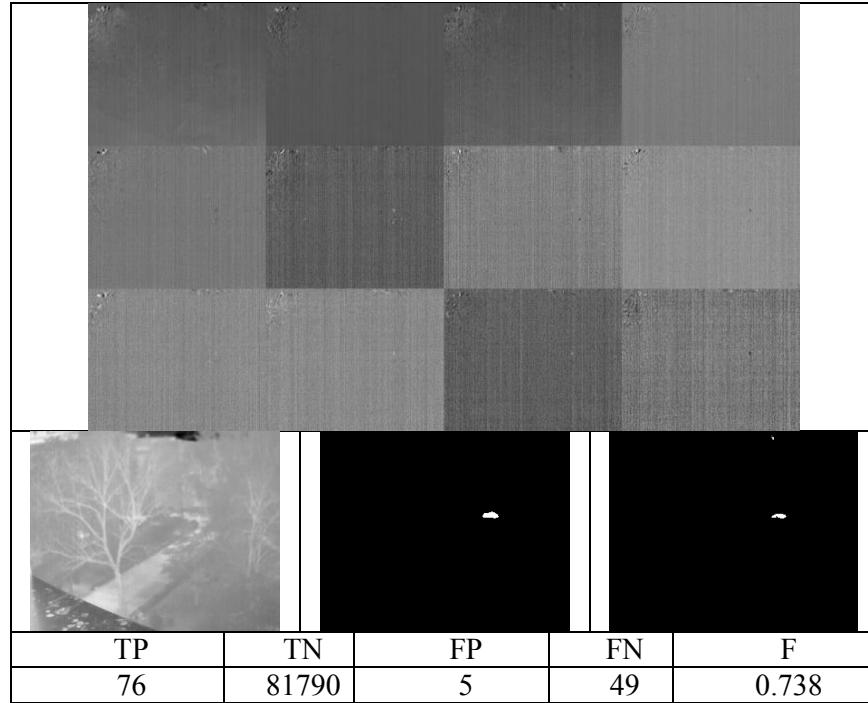


Fig. 4. Results obtained from *Car* test sequence.

It can be seen that outdoor sequences are performing worse than shown earlier, indoor images. This is caused by wrong lens position during recording, therefore images are slightly out of focus. Nevertheless, resulting background removal is still very good.



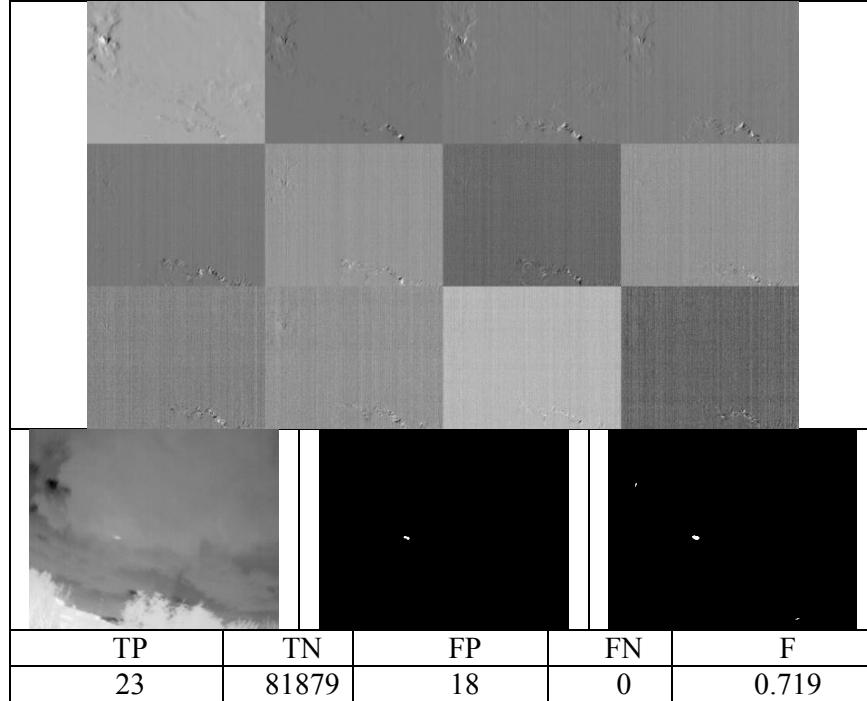


Fig. 5. Results obtained from *BirdOnTheSky* test sequence.

5 Conclusions

The ability to effectively detect and remove background in thermal video is crucial for applications like surveillance, autonomous vehicles or industrial inspection. Being able to employ the same methods for both images captured in visible and long-infrared light spectrum can greatly simplify processing applications, especially for embedded systems. Moreover, thermal images can be easily preprocessed to eliminate areas with too high or too low temperature to additionally improve background removal process.

As this study shows, methods based on PCA subspace decomposition can be successfully applied to IR images and perform as well as other methods designed specifically for thermal imaging. Algorithm also benefits from higher dynamic range provided by 14-bit signal, instead of usual 8-bit for classic video systems. Further investigation of this subject could include comparison with more BS methods.

Acknowledgement

This work was supported by the National Science Centre, Poland, under the grant no. 2016/21/B/ST6/01461.

References

1. Bogusław Cyganek, Michał Woźniak. Efficient Real-Time Background Detection Based on the PCA Subspace Decomposition . ICAISC, 2017.
2. James W. Davis, Vinay Sharma. Background-Subtraction in Thermal Imagery Using Contour Saliency. International Journal of Computer Vision, 2006.
3. Bogusław Cyganek, Object Detection and Recognition in Digital Images: Theory and Practice, Wiley, 2013.
4. FLIR THERMAL IMAGING FOR MACHINE VISION AND INDUSTRIAL SAFETY APPLICATIONS,http://www.flirmedia.com/MMC/THG/Brochures/AUT_021/AUT_021_EN.pdf, last accessed 2017/04/20.
5. FLIR A35SC datasheet, http://www.flirmedia.com/MMC/THG/Brochures/AUT_045/AUT_045_US.pdf, last accessed 2017/04/20.
6. FLIR Uncooled detectors for thermal imaging cameras,[http://www.flirmedia.com/MMC/CSV/App_Stories/AS_0015_EN.pdf](http://www.flirmedia.com/MMC/CSV/App_ Stories/AS_0015_EN.pdf), last accessed 2017/04/20.

Author Index

- | | |
|----------------------|-------------------|
| Bhandarkar, S. M., 1 | Kumar, A. CS, 1 |
| Cyganek, B., 23, 33 | Seichter, D., 11 |
| Eisenbach, M., 11 | Sharma, K., 1 |
| Gross, HM., 11 | Stricker, R., 11 |
| Knapik, M., 33 | Vorndran, A., 11 |
| | Wengefeld, T., 11 |