DOI: 10.15849/IJASCA.220720.05

Int. J. Advance Soft Compu. Appl, Vol. 14, No. 2, July 2022
Print ISSN: 2710-1274, Online ISSN: 2074-8523

Novel Approach for Augmented Reality using Convolutional Neural Networks

Zainab Oufqir, Abdellatif EL ABDERRAHMANI and Khalid Satori

LISAC, Department of Mathematics and informatic University Sidi Mohamed Ben Abdellah, Faculty of Sciences Fes, Morocco
zainab.oufkir@usmba.ac.ma, abdellatif.elabderrahmani@usmba.ac.ma, khalid.satori@usmba.ac.ma

Abstract

In this paper, we will exploit the potential of convolutional neural networks (CNN) in augmented reality. Our work combines existing approaches and produces a new method for aligning a virtual object in the real world and in real time. Our method consists in detecting 2D objects of one or more classes present in the real world with CNN algorithms, then using the output of the network to calculate the position of the camera with the PnP algorithm in order to augment the detected object with additional information. The lightness of the MobileNet convolutional neural network combined with the speed of the Single Shot multibox Detector (SSD) framework allows to analyze the acquired images in real time and to use devices with limited resources and performance. We use a trained model that detects 20 different classes, the network receives as input an image sequence acquired in real time. The output of the network provides the set of detected classes as well as the coordinates of the corners of the surrounded rectangle on the object of this class. The coplanar coordinates of this rectangle are used to calculate the camera position and to align a 3D virtual object in the middle of the bounding box surrounding the detected object. The results obtained in the experimental part show the importance and the robustness of the method.

Keywords: *augmented reality; camera pose estimation; convolution neural network; MobileNet-SSD*

1 Introduction

Object detection is the fundamental problem and most studied in computer vision, it describes and locates any object in a scene. This detection is performed at the level of a digital image which represents the 2D projection of a 3D scene [1] [2]. Computer vision relies heavily on machine learning, especially when it comes to real-time detection, which is the case for augmented reality [3]. The machine must be trained with labelled data in the form of several images on classes of objects, the goal of the training is to optimize the precision, the more models there are the more precise detection, the

objective of this training is to identify and locate in real time objects by specifying the name of their class through a supervised training. Supervised learning is a system that provides both input data and expected output data [4], input and output data are labelled for classification, to provide a learning base for further processing of the data. Let's take the example of a supervised learning system for image processing in which photos of vehicles belonging to the categories cars and trucks are introduced. After sufficient observation time, the system must be able to distinguish between several unlabelled images and to categorise them, once this objective has been achieved, the learning process can be considered complete.

From a training set containing n examples and their labels, given a data set D, described by a set of characteristics X, a supervised training algorithm will find a mapping function between the predictive input variables X and the variable to be predicted Y. The mapping function describing the relationship between X and Y is called a prediction model. When the variable to be predicted takes a discrete value, we talk about a classification problem. Among the classification algorithms, we can find: Support Vector Machine (SVM), Neural Networks, Naïve Bayes, Logistic Regression, etc [5]. The robustness of the algorithm will depend on the precision of its training. An algorithm that learns from supervised content produces an internal map that can be used to classify new amounts of data. Each of these algorithms has its own mathematical and statistical properties, depending on the training set.

In augmented reality, object detection is the initial step in the process of camera pose calculation and the most expensive in calculation time [6]. The progress and power of the new technology allows the execution of robust detection algorithms in real time. Our aim in this work is to detect objects of a certain class and to augment them with additional information. In the following sections we detail the different approaches and techniques based on supervised learning to detect an object in real time as well as how we exploit this detection in the camera pose calculation process to align a 3D virtual object in the real world.

2 Related Work

2.1 Classical algorithmic approach

In order to identify objects in real world, it is necessary to use image processing algorithms through classical algorithmic approach (edge, shape, marker or pattern detection) or by a neural system (Deep Learning). For classical approaches, two kinds of detection are used in augmented reality, the first one is marker-based, it consists in adding in the scene a marker with a known geometry, this method has been popularized by the ARToolKit [7], the most commonly used markers are black and white images that have a unique identifier in the center of a square, the process consists of detecting the four corners of the square (example Harris [8]) and identify the pattern in the center of the square to correctly align a virtual object on the marker [9]. The marker-based method is used in several areas and facilitates the camera pose estimation process [10].The second represents markerless methods , these methods detect characteristic points present in the scene without any prior knowledge of the environment and use descriptor to match them [11], there are several algorithms to detect them such as SIFT [12], SURF [13], BRIEF [14], the process consists of detecting key points and matching them in a sequence of images taken from different points of view [15] [16]

[17] [18]. First, we used the markerless detection approach to detect different objects present in a scene, but this approach is limited to objects of simple shape (square, rectangle etc...).

2.2 Neural algorithmic approach

For the neural approach, convolutional neural networks (CNNs) are widely used in pattern and object recognition problems because they offer a number of advantages over other techniques [19], they are the best performers in image processing applications, in some cases surpassing human performance. The main advantage of the deep learning approach with CNN is the simplification of image data extraction, which was previously done by standard image processing algorithms (thresholding, contouring, segmentation) [20]. The standard artificial neural network differs from convolutional networks mainly by the structure of the hidden layers of neurons and a significantly higher number of neurons per layer [21]. Detectors such as Look Once (YOLO) [22], Faster- Recurrent Convolutional Neural Networks (RCNN) [23], Single-Shot Detectors (SSD) [24] use deep convolutional neural networks to detect and classify objects and are dedicated especially for mobile devices such as smartphones or tablets. YOLO needs a powerful graphical processing unit to perform the classification process, RCNNs are quite slow (on the order of seven frames per second), which will affect the entire classification process. For SSD, it can detect several objects in an image using a single deep neural network and is characterized by its speed and suitable for real-time image processing [25]. It is combined with MobileNets [26], a lightweight convolutional neural network dedicated specifically to resource-limited devices [27].

In augmented reality, there are different libraries such as ARCore by Google [28] and ARKit by Apple [29] which detects markers added to the application or flat surfaces in an unknown environment. The camera of a device provides a sequence of images acquired in real time where the detection process can be applied. Both libraries use computer vision approaches and data from the Inertial Measurement Unit sensor to calculate the camera position in order to correctly align a virtual object in the real world. These two technologies can exploit the power of Machine Learning through other modules such as CoreML from Apple and MLKit or TensorFlow Lite from Google, they integrate trained models of Machine Learning on a mobile application and to exploit these models in object detection.

3 Methodology

3.1 Convolutional neural networks

Artificial neural networks play a very important role in image understanding, character and shape recognition [30], they analyse each image in successive stages: the first layer of "neurons" analyses the information contained in the image, then the second layer carries out an even finer and more precise scan thanks to the result obtained previously, and so on, until the final visual recognition. A large number of objects of the same category are presented to the neural network, the computer learns to recognize these objects in new images by analyzing the recurring patterns within the example images. For image processing, convolutional neural networks (CNNs) are used. They define a sub-category of neural networks and are currently the most powerful models for

classifying images [31]. Convolution is a mathematical tool that is very widely used for image processing, which explains why convolutional neural networks are particularly suitable for image recognition and are able to categorize information from the simplest to the most complex. This type of network is inspired by the biological functioning of the visual cortex. Their architecture is composed of two main blocks, the first block is the particularity of this type of neural networks, it contains several layers, the convolution layer is the key component of convolutional neural networks, its purpose is to identify the presence of a set of features in the images received as input.

3.2 Feature Learning

The convolution acts as a filter, the filters analyze the images zone by zone. A color image is generally composed of 3 channels: red, green and blue, each pixel of the image is composed by an integer value between 0 and 255 on each of the channels. The filter moves over the entire input image and specializes in recognizing patterns. For example, one filter may specialize in edge detection, while another will recognize certain shapes.

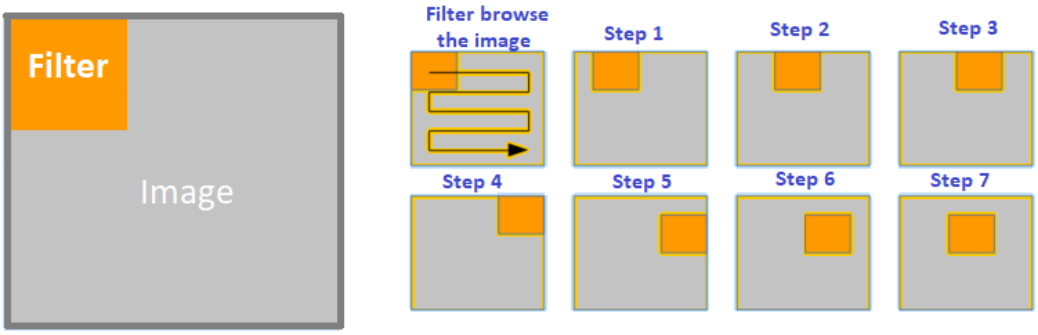

Fig1: Filtering step in convolutional neural networks

The ReLU correction layer is a mathematical function that cancels everything below a certain value and keeps everything higher:

$$relu(x) = \max(0, x) \qquad (1)$$

The pooling layer consists in reducing the size of the images, while preserving their important characteristics, it consists on a small region (a few pixels in squares), to keep only the highest value.
Flatten: this is the flattening of the image from a rectangular (or square) matrix to a long vector. To do this, we need to concatenate all the rows of pixels of the image (or the columns) in succession.

3.3 Classification

The second block is not characteristic of a CNN, it is in fact at the end of all neural networks used for classification. It takes as input the image completely modified by convolutions and other vector operations and returns probabilities for each prediction class. The following figure summarizes this process:

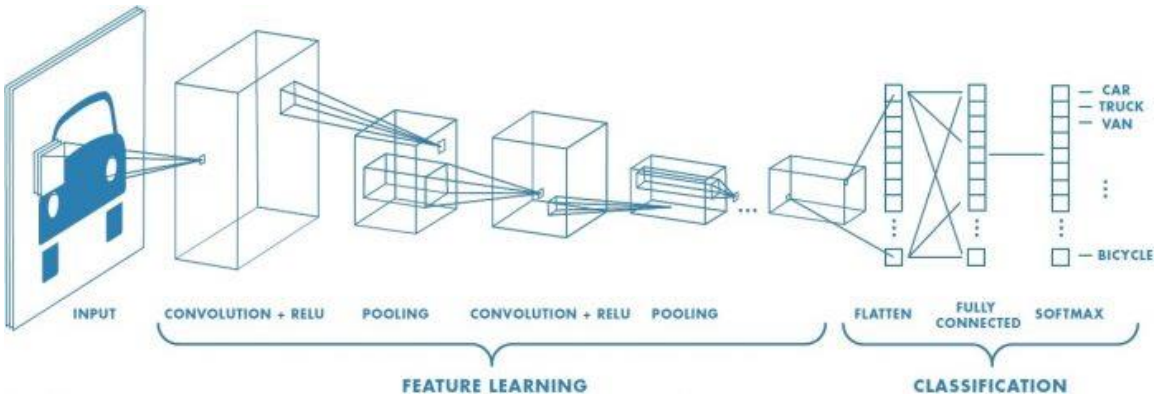

Fig2: Feature Learning + Classification

3.4 MobileNet & SSD (Single Shot multibox Detector)

Neural networks and more specifically CNN - Convolution Neural Network are particularly appreciated for image classification and object detection. However, these neural networks perform convolutions and operations that are very costly in terms of computation and memory. Image classification in embedded systems therefore represents a major challenge due to hardware constraints. A new algorithm appeared MobileNet [32] has optimized convolution by making it faster, less memory-intensive and less power-consuming while maintaining good prediction, and is designed specifically for mobile applications. In MobileNet, convolution is replaced by a Depthwise Separable Convolution to build deep and lightweight neural networks. This is a two-step process, the first step is called Depthwise Convolution, which applies a filter to each channel, as opposed to traditional convolution, which applies a filter to all channels. The second step is the Pointwise Convolution, which combines the outputs of the Depthwise Convolution [26].

To detect an object in real time, the MobileNet classification algorithm has been combined with the Single Shot multibox Detector (SSD) framework, it is a detector that sees the entire image at the input of its network once without going through a mechanism of proposal and then classification of regions, which explains its speed, it is an approach that can detect the occurrences of a certain category of objects in an image, essential in real-time applications [24], SSD has achieved new records in terms of performance and accuracy for object detection tasks, with a score of more than 74% mAP (mean accuracy average) at 59 frames per second on standard data sets such as PascalVOC and COCO.

The tasks of locating and classifying objects are performed in a single pass through the network and classifies the detected objects. SSD performs a convolutional network on the input image only once and calculates a characteristic map, a small convolutional kernel is used on this feature map to predict the classification probability and a bounding box in the form of a rectangle containing the object of this class, This rectangle is described by cx(x coordinate of center), cy(y coordinate of center), h(height of object), w(width of object), it can detect several objects at the same time in an image using a single deep neural network.

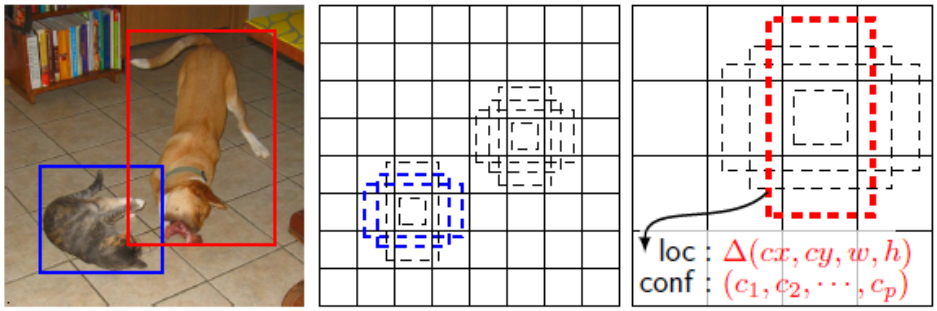

Fig3: Detecting and classifying objects with SSD

By scanning the image, the SSD is able to identify all areas that have an item to classify, it is able to predict whether an object is present in many areas of the image. These areas (bounding boxes), of different scales and ratios, are distributed on a grid applied to the image. The neural network is also trained to predict the correction of these boxes so that they adapt more finely to the detected object.
SSD's bounding box detection technique is inspired by the work of Szegedy [33] [34], Suppose we want to use m feature maps for prediction. The default bounding box scale for each feature map is calculated as follows:

$$s_k = s_{\min} + \frac{s_{\max} - s_{\min}}{m-1}(k-1),\ \mathrm{k} \in [1, m] \tag{2}$$

$S_{\min}$ equal to 0,2 et $S_{\max}$ equal to 0,9 which means that the lowest layer has a scale of 0.2 and the highest layer has a scale of 0.9 and all intermediate layers are evenly spaced. For an aspect ratio $a_r \in \left\{1, 2, 3, \frac{1}{2}, \frac{1}{3}\right\}$, the width and height of the rectangle is calculated as follows:

$$\text{Width } (w_k^a = s_k\sqrt{a_r}) \text{ and height } (h_k^a = s_k / \sqrt{a_r}) \tag{3}$$

3.5 Camera pose estimation

Now that we have our rectangle around the detected object, we go through the augmentation step to insert and align a 3D virtual object on the image of the detected object. The calculation of the camera position is the most studied problem in augmented reality, it's about calculating the user's point of view in a real world reference. In general, a camera allows to switch from a 3D real world to its 2D representation in an image plane. The perspective projection from the pinhole model best represents reality and most smartphone cameras are configured according to this model. It is represented by the following figure:

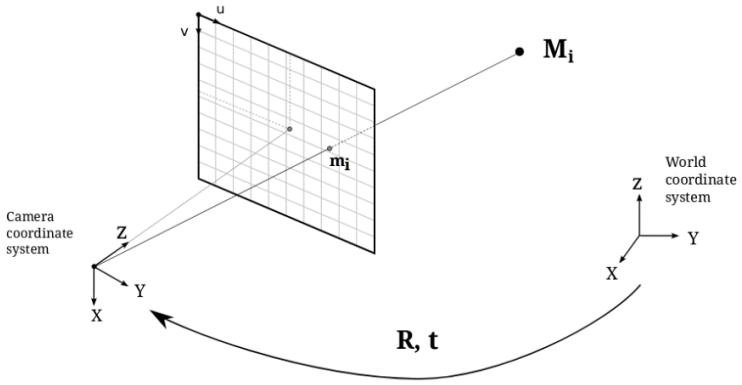

Fig4: Representation of pinhole model

$$s\,m = K\left[R|t\right]M \tag{4}$$

M: coordinates of a 3D point in a real-world reference
m: coordinates of the projection of point M in the image frame
K: matrix of internal camera parameters
$\left[R|t\right]$: rotation matrix R and translation t of the camera

Calculating the position of the camera means finding its rotation R and translation t in the real world [6]. To solve this equation, we must have the position of four 3D points of the scene in the real-world reference and the 2D position of their projection in the image sequence acquired in real time. To do this, we use the position of the four corners of the resulting rectangle at the output of the network, this rectangle surrounds and locates the detected object in the image sequence acquired in real time. The organigram in figure 5 shows the different steps implemented to apply our method.

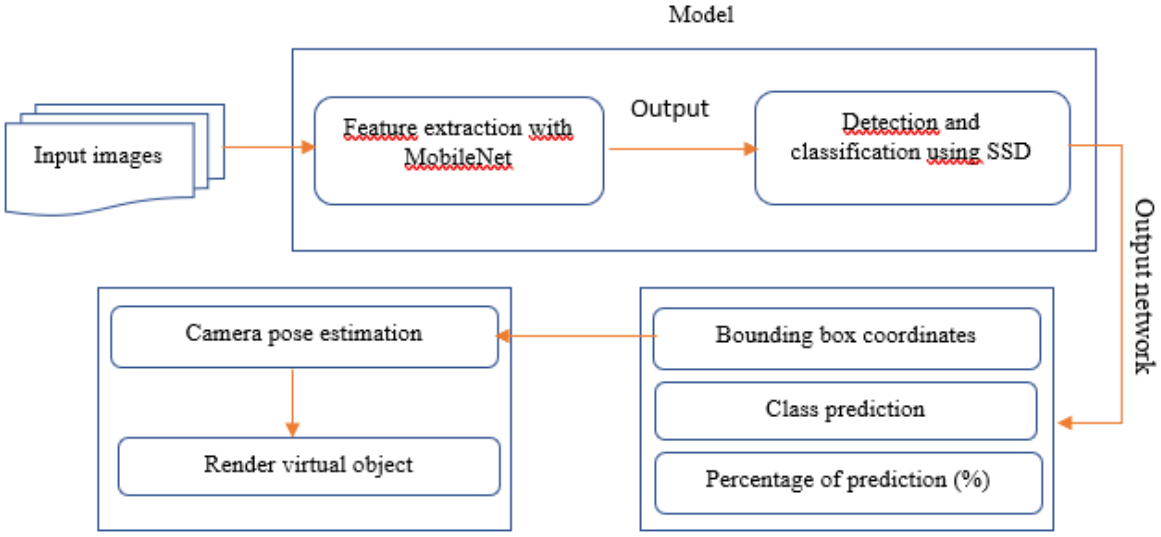

Fig5: Organigramme of our method

The model receives as input the images acquired in real time, each image undergoes feature extraction with MobileNet, SSD receives these features to detect the categories of objects present in the images. This network provides as output the class of the detected object, the percentage of this prediction and the coordinates of the rectangle surrounding the object. These coordinates are used to calculate the camera position in order to align a virtual object with the detected object in the real world.

4 Experimentation

To obtain the coordinates (x, y) of the rectangle in an image, we have to apply object detection, this detection allows us to know which objects are present in an image and where they are located. Single Shot Detectors and MobileNets are used for real-time and ultra-fast object detection on devices with limited resources and performance.

We used OpenCV, an open source library dedicated to real time image processing, we used its dnn module [35] to load a trained object detection network. This will allow us to pass input images in real time through the network and obtain the output coordinates (x, y) of the rectangle surrounded by each object in the image.

We chose a pre-trained model that can detect 20 objects in an image (airplanes, bicycles, birds, boats, bottles, nozzles, cars, cats, chairs, cows, dining tables, dogs, horses, motorbikes, people, potted plants, sheep, sofas, trains, and tv monitors), this model is a Caffe version [36] [37] of the original implementation of TensorFlow by Howard and was trained by chuanqi305 [38]. This model uses MobileNet-SSD to train a COCO (Common Objects in Context) data set [39] and achieves an accuracy of 72.7% mAP (medium accuracy).

After this model is loaded by the application, for each incoming image in real time we apply the detection process to extract the objects defined by the model, this process starts by formatting the image for the network and pass it as input using the dnn module of OpenCV deep learning [35]. The output of this network provides us the information on each detected object, the first one is the identifier of its class, the second one is the percentage of correspondence with the class in question, the rest are the coordinates of the corners of the delimitation box which represents the rectangle surrounded on the detected object.

To find the position of the camera, we use the method introduced by F.Moreno-Noguer, V.Lepetit and P.Fua [40], It is a fast and more precise method, it’s able to find translation and rotation of the object using its 3D points in a local coordinate system and their 2D projections on the image. The application has been developed with Unity software [41] using the programming language C#.

For the tests, we chose to detect different classes, for each object detected and identified, a virtual object of the same class is inserted on it specifying the percentage of similarity with this class. The following figures show the result of our tests for the different objects in an image. For each test, the first image represents the object to be augmented, the second image represents a scene augmented by a virtual object of the same class of the detected object.


Test 1


Test 2


Test 3


Test 4


Test 5


Test 6


Test 7


Test 8

Fig6: Scenes before and after augmentation

Since each detection and augmentation is performed in real time, it was interesting to test the execution time as well as the similarity score of the detected class for each test performed. The following table summarizes the performance of our approach:

Table 1: Execution time and similarity score for each test

	First Detection & augmentation (seconde)	**Similarity score**
Test 1	0.259655	0.9999994
Test 2	0.335411	0.9997669
Test 3	0.268999	0.9998919
Test 4	0.259655	0.9999981
Test 5	0.328886	0.9999148
Test 6	0.228465	0.9995827
Test 7	0.356975	0.9913841
Test 8	0.288651	0.9983723
Test 9	0.359294	0.7530333
Test 10	0.299655	0.7306817
Test 11	0.228496	0.9988723
Test 12	0.266421	0.857177

As we can see in the tests we have performed, the virtual objects are correctly aligned with the images of their classes in the real world, which shows the robustness and efficiency of our method. A rectangle surrounds the detected object, its four corners represent the output of the network and their coordinates have been used to calculate the position of the camera in the real world.

5 Conclusion

In this article, we presented a new method to insert and align a 3D virtual object in a real scene. We have exploited the richness of convolutional neural networks to classify and detect objects present in a scene. The lightness and speed of MobileNet-SSD allowed us to use the power of this network on limited power devices. This network

receives as input the image sequence acquired in real time and sends as output the identifier of the detected class and the coordinates of the delimiter box representing the surrounded rectangle on the object of this class. The coordinates of these corners allowed us to calculate the position of the camera using the PnP algorithm to align a 3D virtual object on the detected object.

References

[1] R. T. Azuma, « A Survey of Augmented Reality », p. 48, 1997.
[2] Z. Oufqir, A. E. El Abderrahmani, et K. Satori, « Important Method for Detecting and Tracking Based on Color », vol. 8, no 5, p. 5, 2019.
[3] D. Chatzopoulos, C. Bermejo, Z. Huang, et P. Hui, « Mobile Augmented Reality Survey: From Where We Are to Where We Go », IEEE Access, vol. 5, p. 6917-6950, 2017, doi: 10.1109/ACCESS.2017.2698164.
[4] E. G. Learned-Miller, « Introduction to Supervised Learning », p. 5.
[5] R. Caruana et A. Niculescu-Mizil, « An empirical comparison of supervised learning algorithms », in Proceedings of the 23rd international conference on Machine learning - ICML '06, Pittsburgh, Pennsylvania, 2006, p. 161-168. doi: 10.1145/1143844.1143865.
[6] E. Marchand, H. Uchiyama, et F. Spindler, « Pose Estimation for Augmented Reality: A Hands-On Survey », IEEE Transactions on Visualization and Computer Graphics, vol. 22, no 12, p. 2633-2651, déc. 2016, doi: 10.1109/TVCG.2015.2513408.
[7] H. Kato et M. Billinghurst, « Marker tracking and HMD calibration for a video-based augmented reality conferencing system », in Proceedings 2nd IEEE and ACM International Workshop on Augmented Reality (IWAR'99), oct. 1999, p. 85-94. doi: 10.1109/IWAR.1999.803809.
[8] C. Harris et M. Stephens, « A Combined Corner and Edge Detector », in Procedings of the Alvey Vision Conference 1988, Manchester, 1988, p. 23.1-23.6. doi: 10.5244/C.2.23.
[9] M. Hirzer, « Marker Detection for Augmented Reality Applications », p. 28.
[10] S. Vogt, A. Khamene, et F. Sauer, « Reality Augmentation for Medical Procedures: System Architecture, Single Camera Marker Tracking, and System Evaluation », Int J Comput Vision, vol. 70, no 2, p. 179-190, nov. 2006, doi: 10.1007/s11263-006-7938-1.
[11] P. Moreels et P. Perona, « Evaluation of Features Detectors and Descriptors based on 3D Objects », Int J Comput Vision, vol. 73, no 3, p. 263-284, juill. 2007, doi: 10.1007/s11263-006-9967-1.
[12] D. G. Lowe, « Object recognition from local scale-invariant features », in Proceedings of the Seventh IEEE International Conference on Computer Vision, sept. 1999, vol. 2, p. 1150-1157 vol.2. doi: 10.1109/ICCV.1999.790410.
[13] H. Bay, T. Tuytelaars, et L. Van Gool, « SURF: Speeded Up Robust Features », in Computer Vision – ECCV 2006, vol. 3951, A. Leonardis, H. Bischof, et A. Pinz, Éd. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, p. 404-417. doi: 10.1007/11744023_32.
[14] D. Hutchison et al., « BRIEF: Binary Robust Independent Elementary Features », in Computer Vision – ECCV 2010, vol. 6314, K. Daniilidis, P. Maragos, et N. Paragios, Éd. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, p. 778-792. doi: 10.1007/978-3-642-15561-1_56.

[15] A. I. Comport, E. Marchand, M. Pressigout, et F. Chaumette, « Real-time markerless tracking for augmented reality: the virtual visual servoing framework », IEEE Transactions on Visualization and Computer Graphics, vol. 12, no 4, p. 615-628, juill. 2006, doi: 10.1109/TVCG.2006.78.
[16] A. I. Comport, E. Marchand, et F. Chaumette, « A real-time tracker for markerless augmented reality », in The Second IEEE and ACM International Symposium on Mixed and Augmented Reality, 2003. Proceedings., Tokyo, Japan, 2003, p. 36-45. doi: 10.1109/ISMAR.2003.1240686.
[17] Z. Oufqir, A. E. El Abderrahmani, et K. Satori, « From marker to markerless in augmented reality », p. 14.
[18] W.-Y. Lin, L.-F. Cheong, P. Tan, G. Dong, et S. Liu, « Simultaneous Camera Pose and Correspondence Estimation with Motion Coherence », Int J Comput Vis, vol. 96, no 2, p. 145-161, janv. 2012, doi: 10.1007/s11263-011-0456-9.
[19] S. Hijazi, R. Kumar, et C. Rowen, « Using Convolutional Neural Networks for Image Recognition », p. 12, 2015.
[20] K. Židek, J. Piteľ, et A. Hošovský, « Machine learning algorithms implementation into embedded systems with web application user interface », in 2017 IEEE 21st International Conference on Intelligent Engineering Systems (INES), oct. 2017, p. 000077-000082. doi: 10.1109/INES.2017.8118532.
[21] I. Pavlenko, V. Simonovskiy, V. Ivanov, J. Zajac, et J. Pitel, « Application of Artificial Neural Network for Identification of Bearing Stiffness Characteristics in Rotor Dynamics Analysis », in Advances in Design, Simulation and Manufacturing, 2019, p. 325-335.
[22] J. Redmon, S. Divvala, R. Girshick, et A. Farhadi, « You Only Look Once: Unified, Real-Time Object Detection », arXiv:1506.02640 [cs], juin 2015, [En ligne]. Disponible sur: http://arxiv.org/abs/1506.02640
[23] S. Ren, K. He, R. Girshick, et J. Sun, « Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks », arXiv:1506.01497 [cs], juin 2015, [En ligne]. Disponible sur: http://arxiv.org/abs/1506.01497
[24] W. Liu et al., « SSD: Single Shot MultiBox Detector », arXiv:1512.02325 [cs], vol. 9905, p. 21-37, 2016, doi: 10.1007/978-3-319-46448-0_2.
[25] Z. Oufqir, L. Binan, A. El Abderrahmani, et K. Satori, « Deep Learning for the Improvement of Object Detection in Augmented Reality », IJASCA, vol. 13, no 3, p. 130-143, déc. 2021, doi: 10.15849/IJASCA.211128.10.
[26] A. G. Howard et al., « MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications », arXiv:1704.04861 [cs], avr. 2017, [En ligne]. Disponible sur: http://arxiv.org/abs/1704.04861
[27] O. Younis, W. Al-Nuaimy, F. Rowe, et M. Alomari, « A Smart Context-Aware Hazard Attention System to Help People with Peripheral Vision Loss », Sensors, vol. 19, no 7, p. 1630, avr. 2019, doi: 10.3390/s19071630.
[28] « ARCore - Google Developer | ARCore », Google Developers. https://developers.google.com/ar/
[29] « ARKit - Apple Developer ». https://developer.apple.com/arkit/
[30] L. Liu et al., « Deep Learning for Generic Object Detection: A Survey », Int J Comput Vis, vol. 128, no 2, p. 261-318, févr. 2020, doi: 10.1007/s11263-019-01247-4.
[31] K. Židek, P. Lazorík, J. Piteľ, et A. Hošovský, « An Automated Training of Deep Learning Networks by 3D Virtual Models for Object Recognition », Symmetry, vol. 11, no 4, p. 496, avr. 2019, doi: 10.3390/sym11040496.
[32] « MobileNets: Open-Source Models for Efficient On-Device Vision », Google AI Blog, 2017. http://ai.googleblog.com/2017/06/mobilenets-open-source-models-for.html

[33] D. Erhan, C. Szegedy, A. Toshev, et D. Anguelov, « Scalable Object Detection using Deep Neural Networks », arXiv:1312.2249 [cs, stat], déc. 2013, [En ligne]. Disponible sur: http://arxiv.org/abs/1312.2249
[34] C. Szegedy, S. Reed, D. Erhan, D. Anguelov, et S. Ioffe, « Scalable, High-Quality Object Detection », arXiv:1412.1441 [cs], déc. 2014, [En ligne]. Disponible sur: http://arxiv.org/abs/1412.1441
[35] « OpenCV: Deep Neural Networks (dnn module) ». https://docs.opencv.org/3.4/d2/d58/tutorial_table_of_content_dnn.html
[36] Y. Jia et al., « Caffe: Convolutional Architecture for Fast Feature Embedding », arXiv:1408.5093 [cs], juin 2014, [En ligne]. Disponible sur: http://arxiv.org/abs/1408.5093
[37] W. Liu, Caffe: a fast open framework for deep learning. Contribute to weiliu89/caffe development by creating an account on GitHub. 2019. [En ligne]. Disponible sur: https://github.com/weiliu89/caffe
[38] chuanqi305, Caffe implementation of Google MobileNet SSD detection network, with pretrained weights on VOC0712 and mAP=0.727.: chuanqi305/MobileNet-SSD. 2019. [En ligne]. Disponible sur: https://github.com/chuanqi305/MobileNet-SSD
[39] « COCO - Common Objects in Context ». http://cocodataset.org/#home
[40] V. Lepetit, F. Moreno-Noguer, et P. Fua, « EPnP: An Accurate O(n) Solution to the PnP Problem », International Journal of Computer Vision, vol. 81, no 2, p. 155-166, févr. 2009, doi: 10.1007/s11263-008-0152-6.
[41] U. Technologies, « Unity », Unity. https://unity.com/frontpage

Notes on contributors

Zainab OUFQIR PhD student at university sidi mohamed ben abdellah, LISAC Department of Mathematics and informatic Faculty of Sciences DharMahraz P.O.Box 1796 Atlas Fes, 30000, Morocco. Email: zainab.oufkir@usmba.ac.ma

Abdellatif EL ABDERRAHMANI¨ Professor at university sidi mohamed ben abdellah, LISAC Department of Mathematics and informatic Faculty of Sciences Dhar-Mahraz P.O.Box 1796 Atlas Fes, 30000, Morocco.

Email: abdellatif.elabderrahmani@usmba.ac.ma

Khalid SATORI ¨Professor at university sidi mohamed ben abdellah, LISAC Department of Mathematics and informatic Faculty of Sciences DharMahraz P.O.Box 1796 Atlas Fes, 30000, Morocco.
Email: khalid.satori@usmba.ac.ma