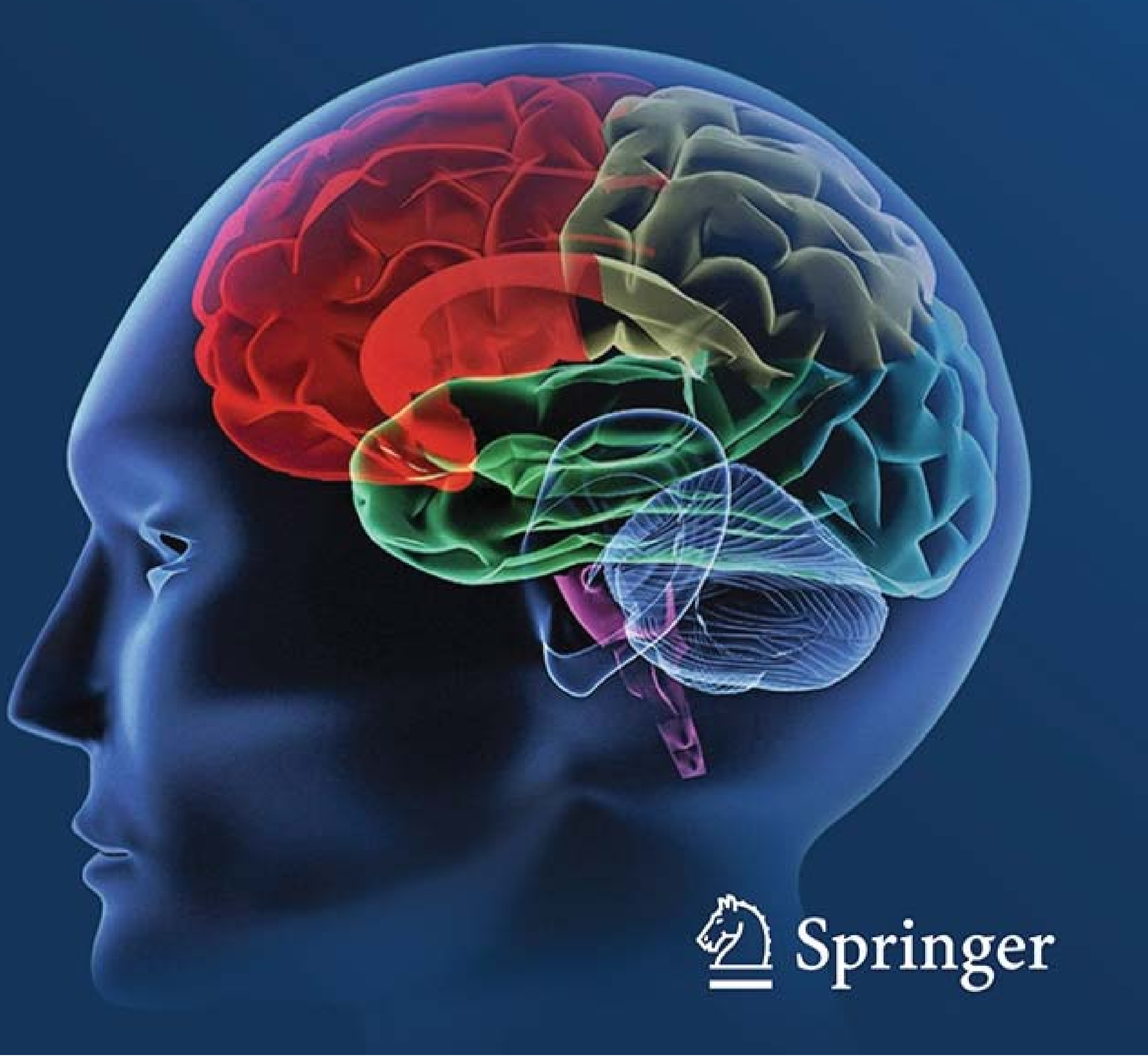
Joseph Awange
Bela Palancz
Lajos Völgyesi

Hybrid Imaging and Visualization

Employing Machine Learning with Mathematica – Python



Springer

Hybrid Imaging and Visualization

Joseph Awange • Béla Paláncz
Lajos Völgyesi

Hybrid Imaging and Visualization

Employing Machine Learning with *Mathematica - Python*

Springer

Joseph Awange
Spatial Sciences
Curtin University
Perth, WA, Australia

Béla Paláncz
Department of Photogrammetry & Geoinformatics
Budapest University of Technology & Economics
Budapest, Hungary

Lajos Völgyesi
Department of Geodesy & Surveying
Budapest University of Technology & Economics
Budapest, Hungary

ISBN 978-3-030-26152-8 ISBN 978-3-030-26153-5 (eBook)
https://doi.org/10.1007/978-3-030-26153-5

This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Preface

Computer vision is a subfield of artificial intelligence that enables the understanding of the content of digital images, such as photographs. Currently, machine learning is making impressive inroads in tackling challenges posed by computer vision related tasks, promising further impressive advances.

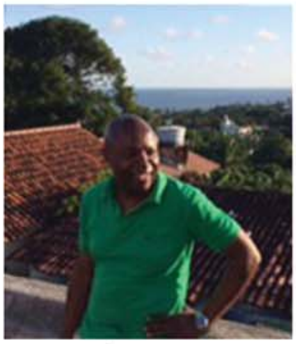
Speaking of computer vision, two modes of books frequently appear (i) reference-based textbooks written by experts, who often are academics, targeting students and practitioners, and (ii), programming oriented books (i.e., play books) written by experts, who often are developers and engineers, and designed to be used as references by practitioners. Whereas the former mainly focus on general methods and theory (Maths) and not on the practical aspects of the problems and the applications of methods (code), the latter focuses mainly on the techniques and practical concerns of the problem solving, where the focus is placed on examples of codes and standard libraries.

Although programme-based books briefly describe techniques with relevant theory (Maths), they probably do not predispose themselves for use as primary reference. In this regard, Dr. Jason Brownlee, a machine learning specialist who teaches developers how to obtain results from modern machine learning methods via hands-on tutorials recommends the work of Richard Szeliski (2010; Computer Vision: Algorithms and Applications) since it provides a short, focused, and readable introduction to computer vision complete with relevant theory, without getting too bogged down. For programmers, he suggests Jan Erik Solem's (2012; Programming Computer Vision with *Python*) since it focuses on real computer vision techniques with standard (or close enough) *Python* libraries. It is an excellent starting point for those who want to get their hands dirty with computer vision.

Our contribution, therefore, intends to be a go between these two types of books. On the one hand, it is like a programmer's book presenting many different techniques illustrated by a large number of examples, accompanied by detailed discussions on the Mathematics behind the different methods. The codes of the algorithms are given in *Python* as well as in *Mathematica* form. In the book the version of *Mathematica* 11 integrates *Python*, most of the codes are blended as hybrid codes, however in the electronic supplement the recent version of Mathematica 12 is employed and uploaded to Researchgate. *Mathematica* is an incredibly powerful platform with a fun and intellectually pleasing language, but is expensive and closed source. *Python* is a convenient, powerful language with a lot of support from the developer community. For as long as the two have existed people have been trying to tie them together, so that one can utilize the integrated advantages of both languages.

This book is divided into five chapters. The first one deals with dimension reduction techniques of visual objects where besides the standard methods; it includes Independent Component Analysis, AutoEncoding and Fractal Compression. The second chapter discusses classification methods that include Support Vector Classification. In the third chapter, different clustering techniques are demonstrated, like Hierarchical Clustering, Density-Based Spatial Clustering of Applications with Noise and Spectral Clustering. The fourth chapter presents different regression techniques, where different robust regression models such as Expectation Maximization, RANSAC and Symbolic Regression are also discussed. The last chapter provides a deep insight into applications of neural networks in computer vision. Besides the standard network types, Deep Learning and Convolutional Networks are also discussed. At the end of every chapter, the considered methods are compared and qualified from different practical points of views.

The authors:



Joseph L. Awange
Curtin University
Australia

Béla Paláncz
Budapest University of Technology and Economics
Hungary

Lajos Völgyesi
Budapest University of Technology and Economics
Hungary

Budapest – Perth – Karlsruhe
May 2019

Acknowledgements

This work was partially funded by National Research, Development and Innovation Office – NKFIH No. 124286. The first author also wishes to express his sincere thanks to *Prof. B. Heck* and *Prof. Hansjörg Kutterer* (Karlsruhe Institute of Technology (KIT), Germany) for hosting him during the period of his Alexander von Humboldt Fellowship (July–September 2019). Parts of this book were written during this period. He is also grateful to *Prof. I. Fitzsimons* (Head of School, Earth and Planetary Sciences, Curtin University, Australia) for the support and motivation that enabled the preparation of this edition. He also wishes to acknowledge the support of Alexander von Humboldt that facilitated his stay at KIT. To all, he says, "ahsante sana" (Swahili for thank you very much). Special thanks go to *Dr Cecilia Xia* for her inputs during the preparation of this book, especially on the *Python* code.

The authors highly appreaciate are greatly thankful for the time and efforts taken by *Dr. Daniel Lichtblau*, Wolfram Research in reading the draft version of the book and providing valuable comments and advice.

Contents

Introduction

1 Computer Vision and Machine Learning

Computer vision (also known as machine vision; Jain et al., 1995), a multi-disciplinary field that is broadly a subfield of artificial intelligence and machine learning has as one of its goals the extraction of useful information from images. A basic problem in computer vision, therefore, is to try to understand, i.e., "*see*" the structure of the real world from a given set of images through use of specialized methods and general learning algorithms (e.g., Hartley and Zisserman 2003; see Fig. 1). Its applications are well documented in Jähne and Haußecker (2000), where it finds use e.g., in human motion capture (Moeslund and Granum 2001). With the plethora of unmanned aircraft vehicles (UAVs) or drones (see Awange 2018; Awange and Kiema 2019), computer vision is stamping its authority in the UAV field owing to its intelligent capability (Al-Kaff et al., 2018). Several publications abound on computer vision, e.g., on algorithms for image processing (e.g., Parker 2011, Al-Kaff et al., 2018), pattern recognition/languages in computer vision (e.g., Chen 2015), feature extraction (Nixon and Aguado 2012) and among others.

On its part, *Machine Learning* (ML) is the employment of statistical techniques by computers to learn specific and complex tasks from given data that are discriminated into learnt and defined classes (Anantrasirichai et al., 2018, 2019). They have widely been used, e.g., for landslides studies (Yilmaz, 2010), vegetations (Brown et al., 2008), earthquakes (Adeli and Panakkat, 2009), land surface classificatios (Li et al., 2014) and for classification of volcanic deformation (Anantrasirichai et al., 2018, 2019). Lary et al., (2016) provides a good exposition of its application.

Traditionally, computer vision's contributions are largely grouped into two categories; *textbook-based* that focus on <u>methods and theory</u> rather than on the practicality, and *programming-based* that focus on the techniques and the practicality of solving the problems. There is hardly any book that tries to bring the two together; i.e., methods/theory on the one hand, and techniques/practicality (i.e., codes) on the other hand. This present book attempts to fill this missing gap by treating computer vision as a machine-learning problem (Fig. 1), and disregarding everything we know about the creation of an image. For example, it does not exploit our understanding of perspective projection.

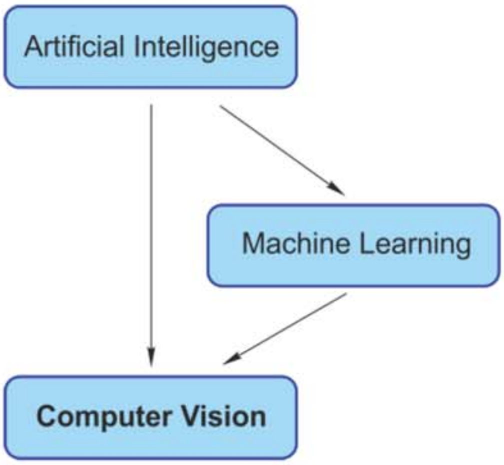


Fig. 1 Relationship between Computer Vision, Artificial Intelligent and Machine Learning.

In general the image processing chain contains five different tasks: reprocessing, data reduction, segmentation, object recognition and image understanding. Optimisation techniques are used as a set of auxiliary tools that are available in all steps of the image processing chain, see Fig. 2.

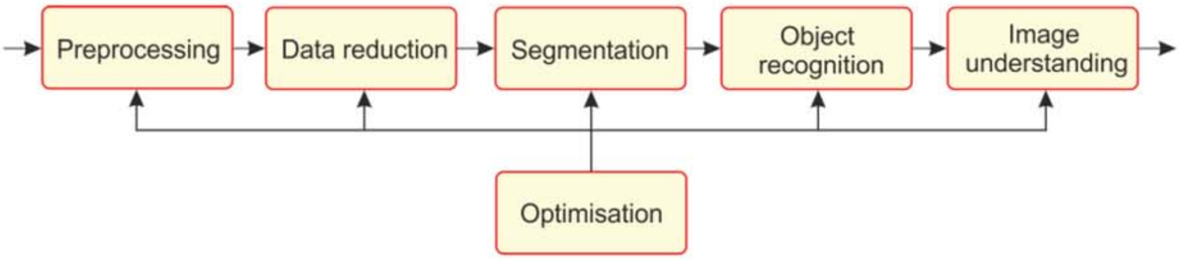


Fig. 2 The image processing chain containing the five different tasks.

Many popular computer vision applications involve trying to recognize things in photographs; for example:

1. Object Classification: What broad category of objects are in this photograph?
2. Object Identification: Which type of a given object is in this photograph?
3. Object Verification: Is the object in the photograph?
4. Object Detection: Where are the objects in the photograph?
5. Object Landmark Detection: What are the key points for the object in the photograph?
6. Object Segmentation: What pixels belong to the object in the image? and
7. Object Recognition: What objects are in this photograph and where are they?

Let us consider some examples, where Machine Learning techniques are applied to solving these computer vision problems.

Example 1 (*Segmentation as Clustering*)

Segmentation is any operation that partitions an image into regions that are coherent with respect to some criterion. One example is the segregation of different textures. The following is an image that highlights bacteria. Now, we would like to remove the background in order to get clear information about the size and form of the bacteria.

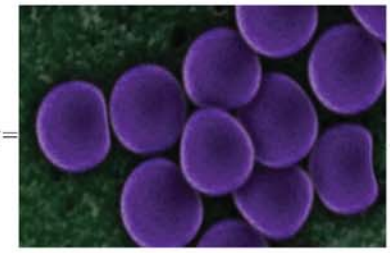
⇒ `img=`  `;`

Fig. 3. Original image.

```
⇒ RemoveBackground[img, {"Foreground", "Uniform"}]
```

⇐

Fig. 4. Image of bacterias after background removal.

Example 2 (*Object Recognition as Classification*)

Object detection and recognition determine the position and, possibly, also the orientation and scale of specific objects in an image, and classify them. In the image below, we can get information on the type of the image object (car) and the possible subclasses probability.

⇒ `img=`  `;`

Fig. 5. Original image.

```
⇒ data=ImageIdentifyn[img, car(WORD), 10, "Probability"]
⇐ <|convertible → 0.725076,saloon → 0.150854,coupe → 0.0760313,
   station wagon → 0.0414005,hatchback → 0.00342106,
   limousine → 0.00153371,automobile → 1.|>
```

Example 3 (*Image Understanding as Landmark Detection*)

Key points of an image can characterize the main feature locations of an image. This information can be employed for further image processing operations like image transform, classification and clustering. Consider the image in Fig. 6 below.

⇒ img=  ;

Fig. 6. Original image of a cathedral.

Let us find the first thirty most important keypoints of the image.

```
⇒ HighlightImage[img, ImageKeypoints[img, "MaxFeatures" → 30]]
```

⇐ 

Fig. 7. The first thirty most important keypoints.

2 *Python* and *Mathematica*

In this book, we employ *Python* (see e.g., Lutz 2001; Oliphant 2007) and *Mathematica* (e.g., Maeder 1991) as well as their blending, since *Python* code can be run from *Mathematica* directly. It is therefore appropriate to provide a brief discussion on them. This section is thus dedicated to their exposition.

Python is now undoubtedly the most popular language for data science projects, while the *Wolfram Language* is rather a niche language in this concern. Consequently, *Python* is probably well-known to the reader compared to *Mathematica*. Given that *Wolfram Language,* widely used in academia (especially in physics, mathematics and financial analytics) has been around for over 30 years, it is actually older than both *R* and *Python*.

The general principle of the *Wolfram Language* is that each function is very high level and automated as much as possible. For example the Classify[ ] function chooses the method automatically for the user. However, the user can also set it manually to something like Method → "RandomForest". The neural network function employed in *Mathematica* uses MxNet as a backend and is similar in its use to *Keras* in *Python*, although nicer to view. In general, the *Machine Learning* (ML) functions in *Wolfram* have a black box feeling to them, although there are lower level functions as well. One should therefore not blindly trust that the automatic solutions provided by the Predict and Classify functions are the only optimal solutions. They are often far from that and at best give baseline solutions on which to rely upon. One can then always use lower level functions to build one's own customized ML solution with *Wolfram* or *Python*. However this ability of *Mathematica* has been improved considerably in the last version released in 2019.

Mathematica has a very good system for documentation with all built-in functions. Also, the documentation itself is in notebooks so that one can quickly try something directly inside the documentation. The documentation in *Mathematica* is really good, but *Python* has a much bigger community with a widened network of support such that it is very likely that one finds an answer to a given problem. Also, one can learn a lot through sites like *Kaggle*. The *Wolfram Mathematica* community in comparison is small and therefore it is harder to find relevant information, although the *Mathematica* community (https://mathematica.stackexchange.com) on stack exchange is really helpful.

So let us talk about the elephant in the room: the price. *Mathematica* is not free, it is actually quite expensive. Since *Mathematica* comes with all functions from the start, there is no need to buy additional "Toolboxes" like in *Matlab*. Now, we present some bullet points for both languages in no particular order.

Wolfram Mathematica

- natural language interpretation
- pattern matching is powerful and prominent, for example in function declaration
- interactive and very good documentation
- consistent

- symbolic, one can pass everything into a function (has a lot of advantages but also makes it harder to debug)
- more advanced notebooks
- no virtual environments and dependencies
- works the same in every OS
- most of the time there is only one obvious way to do things, for example plotting
- Dynamic and manipulates functions for more interactivity
- built-in knowledge
- indices start at 1
- instant Application Programming Interface (API) (although only in the *Wolfram Cloud* or one's own *Wolfram Enterprise Cloud*)
- hard to find a job / hard to recruit people who know *Wolfram*

Python

- "There is a package for that"
- closer to state of the art
- codes are easier to read and to maintain
- debug messages are usually more helpful
- free
- learn from *Kaggle*
- lots of possibilities to deploy a trained model
- a lot of online courses, podcasts and other resources
- use of google-colab or *Kaggle* for learning ML without a local GPU
- pandas is easier to use than the "Dataset" in *Mathematica*
- bigger community, hence easier support.

Learning another language is usually beneficial for one's overall understanding of programming. So learning *Wolfram* might be a nice addition. We use the *Wolfram* language for quick prototyping of ideas and often come up with interesting combinations of data or feature engineering with the built-in knowledge of the *Wolfram* language. In addition, a quick Manipulate is fun and can help a lot in understanding the problem and data better.

In *Mathematica,* with just one line, one can deploy our model as an Application Programming Interface (API) or web-app, although only in the *Wolfram* infrastructure, which might not fit inside one's infrastructure or policy. Also, the high level functions Classify and Predict are too much of a black box and even standard scikit learn algorithms outperform them.

Overall, we hope that both languages inspire each other, as the *jupyter notebook* was certainly inspired by *Mathematica*. On the other hand *Wolfram* will have a difficult future if they continue to try to do everything on their own and lock users into their infrastructure. Therefore, a combination of the two languages will be more and more fruitful in the future. For more details see: *Wolf-*

ram Language (*Mathematica*) vs. *Python* for Data Science Projects, 2019, ATSEDA AB (https://atseda.com/en/blog/2019/02/12/mathematica - and - python)

References

Adeli, H., and A. Panakkat (2009), A probabilistic neural network for earthquake magnitude prediction, Neural Networks, 22(7): 1018-1024, doi:https://doi.org/10.1016/j.neunet.2009.05.003.

Al-Kaff A, Martín D, García F, de la Escalera A, Armingol JM (2018) Survey of computer vision algorithms and applications for unmanned aerial vehicles. Expert Systems with Applications 92: 447-463, doi: 10.1016/j.eswa.2017.09.033

Anantrasirichai N, Biggs J, Albino F, Hill P, Bull D (2018) Application of machine learning to classification of volcanic deformation in routinely generated insar data, Journal of Geophysical Research: Solid Earth 123 (8) : 6592-6606. doi:10.1029/2018JB015911.

Anantrasirichai N, Biggs J, Albino F, Bull D (2019) A deep learning approach to detecting volcano deformation from satellite imagery using synthetic datasets. Remote Sensing of Environment, doi: 10.1016/j.rse.2019.04.032.

Awange JL and Kiema JBK (2019) Environmental Geoinformatics. Extreme Hydro-Climatic and Food Security Challenges: Exploiting the Big Data. Springer Nature, Heidelberg.

Awange JL (2018) GNSS Environmental Sensing. Revolutionizing Environmental Monitoring. Springer International Publishing AG.

Brown, M., D. Lary, A. Vrieling, D. Stathakis, and H. Mussa (2008), Neural networks as a tool for constructing continuous ndvi time series from AVHRR and MODIS, In ternational Journal of Remote Sensing, 29(24), 7141–7158, doi:https://doi.org/10.1080/01431160802238435.

Chen CH (2016) A handbook of pattern recognition and computer vision. 5th Edition. World Scientific, Dartmouth, USA.

Han J, Shao L, Xu D and Shotton J (2013) Enhanced computer vision with microsoft Kinect sensor: A review. IEEE Transactions on Cybernetics 43:1318-1334, doi: 10.1109/TCYB.2013.2265378.

Hartley R and Zisserman A (2003) Multiple view geometry in computer vision. 2nd Edition, Cambridge University Press, Cambridge, UK.

Jähne B and Haußecker H (2000) Computer Vision and Applications. A guide for students and practitioners. Academic Press. Elsevier.

Jain R, Kasturi R, Schunck BG (1995) Machine vision. McGraw-Hill.

Lary, D. J., A. H. Alavi, A. H. Gandomi, and A. L. Walker (2016), Machine learning in geosciences and remote sensing, Geoscience Frontiers, 7(1), 3 –10, doi: 10.1016/j.gsf.2015.07.003, special Issue: Progress of Machine Learning in Geosciences.

Li, C., J. Wang, L. Wang, L. Hu, and P. Gong (2014), Comparison of classification algorithms and training sample sizes in urban land classification with landsat thematic mapper imagery, Remote Sensing, 6(2), 964–983, doi: 10.3390/rs6020964

Lutz M (2001) Programming Python. 2nd Edition. O'Reilly & Associates.

Maeder RE (1991) Programming in Mathematica. 2nd Edition. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA

Moeslund TB and Granum E (2001) A Survey of Computer Vision-Based Human Motion Capture. Computer Vision and Image Understanding 81:231-268, doi: 10.1006/cviu.2000.0897

Nixon M and Aguado A (2012) Feature extraction and image processing for computer vision. 3rd Edition. Academic Press, Elsevier, London.

Oliphant TE (2007) Python for Scientific Computing. Computing in Science & Engineering 9(3): 10-20, doi: 10.1109/MCSE.2007.58

Parker JR (2011) Algorithms for image processing and computer vision. 2nd Edition. Wiley Publishing, Indianapolis, Indiana.

Yilmaz, I. (2010), Comparison of landslide susceptibility mapping methodologies for koyulhisar, turkey: conditional probability, logistic regression, artificial neural networks, and support vector machine, Environmental Earth Sciences, 61(4), 821–836, doi: 10.1007/s12665-009-0394-9.