Journal of open research software

VIFECO: An Open-Source Software for Counting Features on a Video

SOFTWARE METAPAPER

PHILIPPE APPARICIO
DAVID MAIGNAN
JÉRÉMY GELB
Author affiliations can be found in the back matter of this article

ubiquity press

ABSTRACT

The aim of this article is to describe an open-source application (Vifeco) that makes it possible to manually identify features on a video. Vifeco also allows to: manage the number of users, create a category (feature) and a collection of categories, read video and identify the features on it, and analyze the counting concordance between two users. Written in Java 11 with the JavaFX UI toolkit, Vifeco is a stand-alone, multiplatform (Windows, Mac and Linux) and multi-language (3 languages supported) application. The software is available under Apache Licence on GitHub (*https://github.com/LAEQ/vifeco*).

CORRESPONDING AUTHOR:
Philippe Apparicio
Environmental Equity Laboratory, Centre Urbanisation Culture Société, Institut national de la recherche scientifique, Montréal, Canada
Philippe.apparicio@ucs.inrs.ca

KEYWORDS:
Video; Digital Action Camera; Mobile Methods; Visual Anthropology; JavaFX

TO CITE THIS ARTICLE:
Apparicio P, Maignan D, Gelb J 2021 VIFECO: An Open-Source Software for Counting Features on a Video. *Journal of Open Research Software*, 9: 7. DOI: *https://doi.org/10.5334/jors.300*

(1) OVERVIEW

INTRODUCTION

Over the past ten years or so, due to their affordability, compactness, high-resolution imagery, and durability, digital action cameras (e.g. Garmin VIRB, GoPro, Campark, Akaso, VanTop Moment) have become an increasingly popular tool in various research fields. In social sciences, they are typically used in visual anthropology [1, 2] and geography [3, 4] for collecting a broad range of data in the field. Indeed, videos recorded by digital action cameras (DACs) can enhance analyses based on innovative methodologies such as mobile methods [5, 6], participatory visual research [7–9], or non-participatory observation studies [10, 11]. They are also many applications of DACs in transportation studies [12–15], environmental sciences [16, 17], and organizational research [18, 19].

Furthermore, recent introductions of automatic detection video tools in the field of computer vision include SegNet [20] based on Deep Convolutional Neural Networks, Vatic [21] or development tools associated with the OpenCV libraries [22], YOLO3, BGSLibrary [23], such as Simple Vehicle Counting [24] and Vehicle Detection with Haar Cascades [25]. Automatic detection video tools are more commonly developed and used in studies focused on road traffic where there is a need for tracking and counting vehicles in a fixed camera video recording [26, 27]. Consequently, the combined use of computer vision and video analysis becomes an interesting tool for comprehending the urban area and the detection of traffic conflicts [28, 29], the classification of users [30, 31], and lastly, for the analysis of user behavior from motor vehicles [32], pedestrians [33–35], as well as cyclists [36].

Since these applications are specifically used to identify features related to road traffic recorded by a fixed camera, automatic detection becomes more difficult when the camera is secured to a moving object. For example, when the camera is secured to the handle bars of a moving bike.

The application presented (Vifeco) in this article is not aligned with the above described work on automatic object detection in videos for two reasons. First, we want the user to be able to define their own features for identification. In consequence, the application can then be used in many different domains outside of transport. Second, we seek to develop an application that is not limited to fixed cameras.

The aim of this article is, therefore, to describe an open-source application (Vifeco) that makes it possible to manually identify features on a video.

IMPLEMENTATION AND ARCHITECTURE

From a technical point of view, Vifeco is an open-source software (Apache licence, version 2.0) written in Java 11 with the JavaFX UI toolkit. The application can be installed on any platform which has a Java Runtime Environment (JRE) and JavaFX runtime (please see the Dependencies section). The user interface currently supports three languages (English, French, Spanish). The application and its Java source code are available free of charge (under the Apache Licence 2.0) and can be downloaded at *https://github.com/LAEQ/vifeco/releases/*.

Vifeco is built with the Griffon framework. Inspired by Grails, this framework enforces concepts like convention over configuration and dependency injection (JSR 330) which facilitates building modular and independent components and services [37]. Also, the Griffon framework is based on the MVC pattern that allows a clear separation between the model, the controller, and the view. The communication between the modules is based on the observer pattern with the dispatch of events and the registration of listeners to allow the communication and the exchange of data throughout the application.

The application's features allow: 1) the management of several users for counting features on videos, 2) the creation of a category (i.e. feature) and a collection of categories, 3) reading video and identifying the features on it (e.g. moving car, moving truck, moving bicycle, etc.); 4) exporting and importing the results in *json* files; 5) analyzing the counting concordance between two sessions (e.g. users). All of these reported functions are described in detail in the following section.

The application is split into multiple MVC modules (***Figure 1***). The Main module loads the different sub-panels of the application. It displays the navigation bar which allows the user to load different sub-modules. Next, four modules implement a CRUD (create, read, update and delete) interface for managing videos, users, categories and collections. These four components receive an injected DAO service to persist data on an embedded H2 database (*https://www.h2database.com/html/main.html*), using the ORM (Object Relation Mapping) Hibernate library for managing transactions with the database. The module entitled *Editor* is dedicated to the manipulation of a video as follows:

- a *Player component* with all basic functionalities (play, pause, rewind, forward);
- a *Controls* component for adjusting video player settings (playback speed) and customize the display of category icons (size, opacity and duration of icons on the video)
- a *Timeline* component for displaying a panel where the category icons are plotted throughout the video.
- A *Summary* module for displaying a panel where the counts for each category are reported.

Finally, the last module is dedicated to calculating several concordance indicators —described in detail later in the next section— for evaluating the agreement between two users.

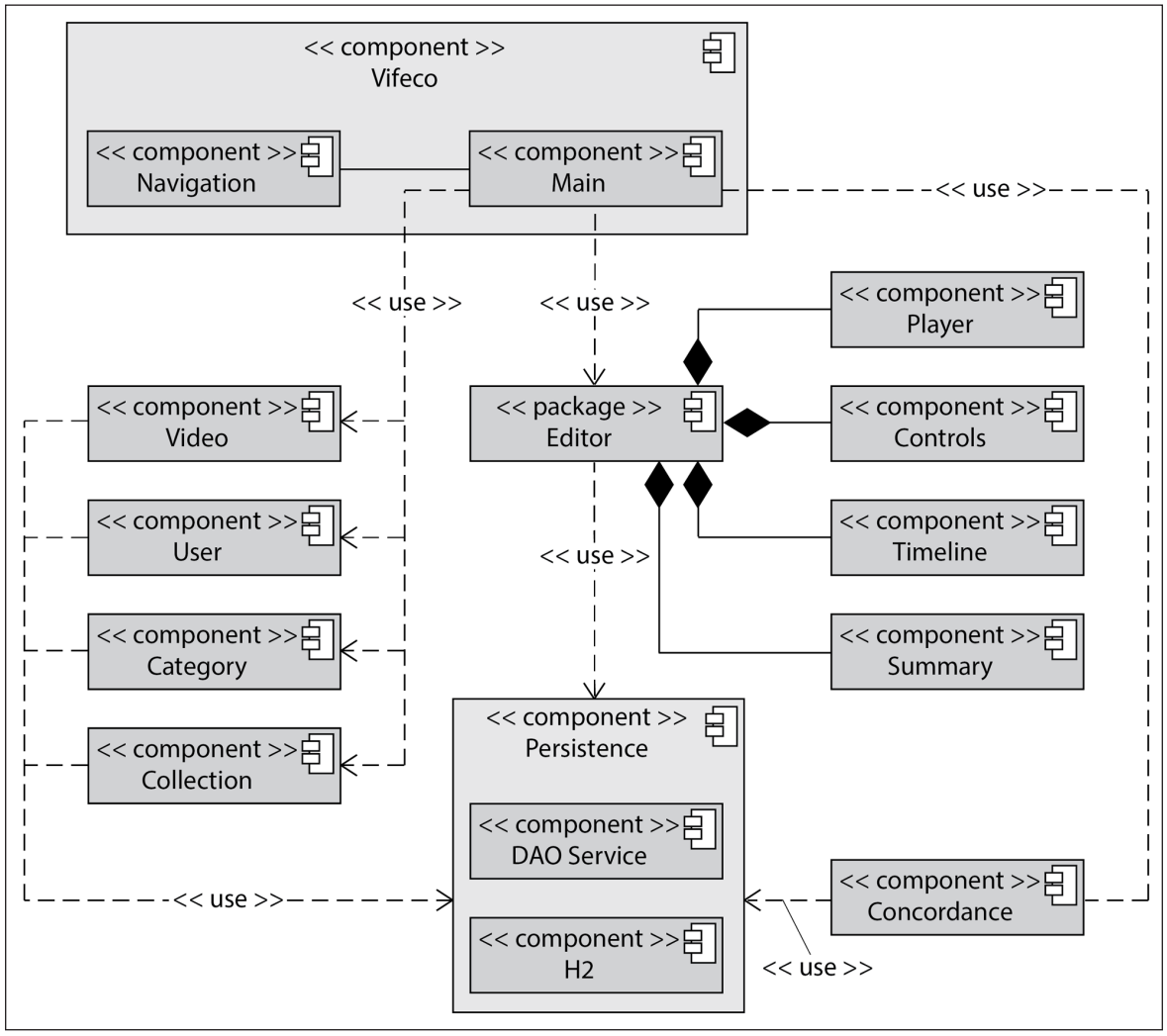

Figure 1 Architecture of VIFECO as reflected by the different modules.

Before describing the interface, it is worth noting that when the executable is first launched, the application creates a folder entitled *vifeco* at the user root level (e.g., ~/vifeco for Linux and macOS, and, c:\users\<username>\vifeco for Windows).

Vifeco Interface

The application's interface is easy to use and organized into different panels. First, four CRUD interfaces are designated to manage users, features, collections, and videos (***Figures 2*** to ***5*** in which the red square indicates the button that activates the panel).

It is important to remember that a category is a feature that the user wants to locate and count on a video, such as a moving car or a moving truck. Basically, the application already includes several categories (***Figure 3***). To create a new category, it is necessary to type its name and specify a unique keyboard shortcut. For the icon shape, the application only supports an SVG (*scalable vector graphics*) with a single path attribute (*https://developer.mozilla.org/en-US/docs/Web/SVG/Element/path*). Using this W3C vector standard makes it easy to change the color and size of icons that will be overlaid on the video (***Figure 3***). Once the icons are created, application users can then create a new collection of categories (***Figure 4***).

Another interface allows users to manage videos. First, the user can add video to Vifeco (by clicking on the icon surrounded by the blue square that activates an open dialog window for selecting a single file). Each new video is assigned with the default user and default collection. It can be changed by clicking on the user or collection name. A drop menu will appear with a list of user or collection to choose from. Selecting a video will display the counts for each category and will allow the user to start a session by clicking Start (***Figure 5***).

Once uploaded, the user can start a work session by selecting the video and then clicking the start button. A new window will open with the editor. The interface is divided into three panels indicated by the label numbers in the ***Figure 6***: 1) the video, 2) player functionalities, 3) a table which summarizes the counts for each category, and 4) a table which displays the full list of the features added by the user. While viewing the video, the user can place the mouse pointer on a video feature and press the keyboard shortcut associated with a category to

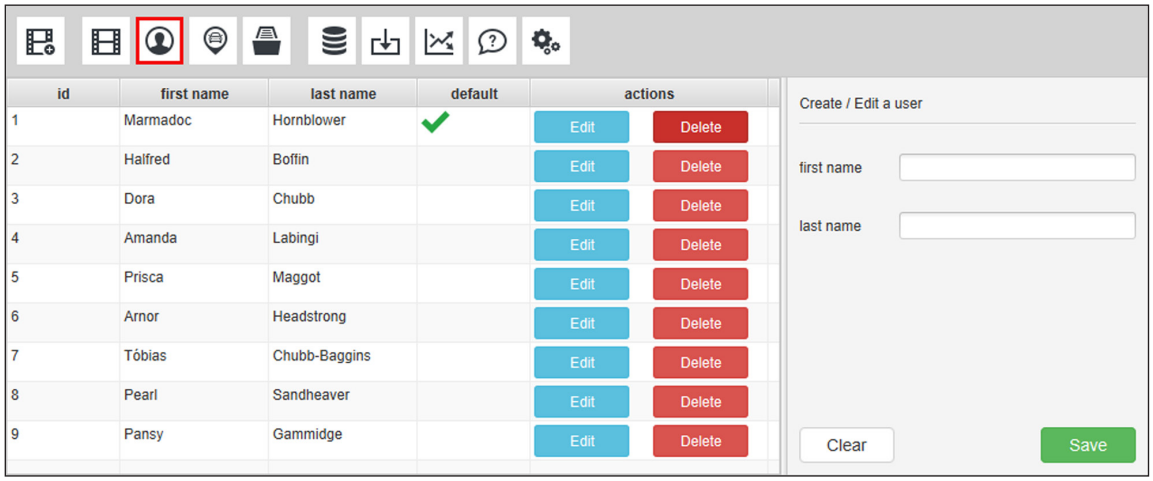

Figure 2 CRUD interface for managing users.

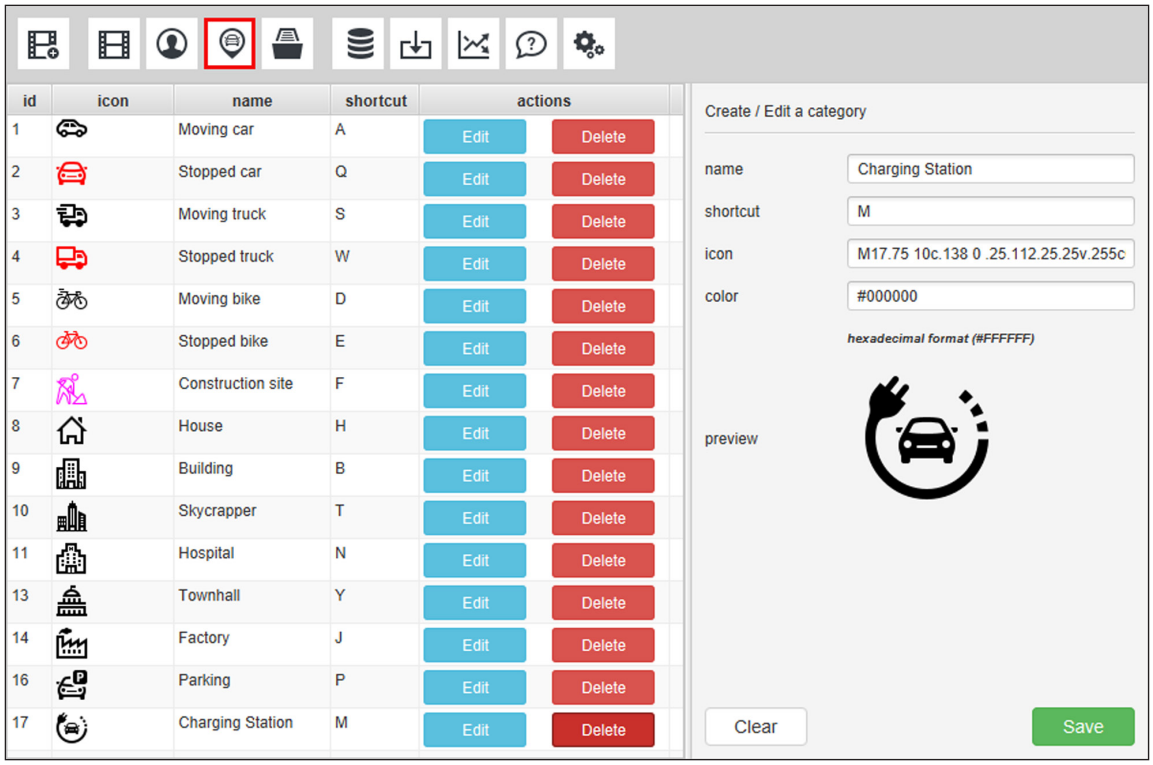

Figure 3 CRUD interface for managing categories.

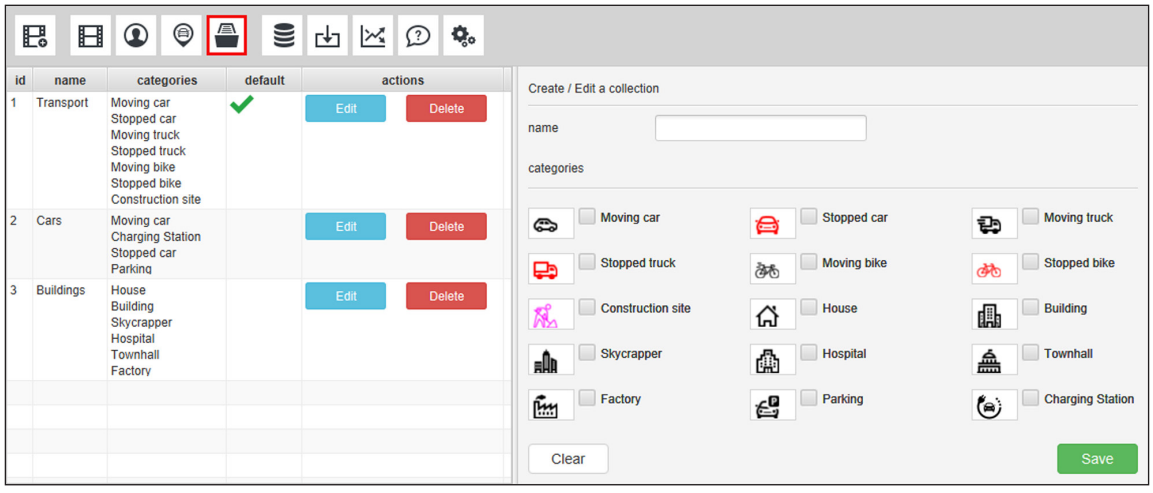

Figure 4 CRUD interface for managing collections.

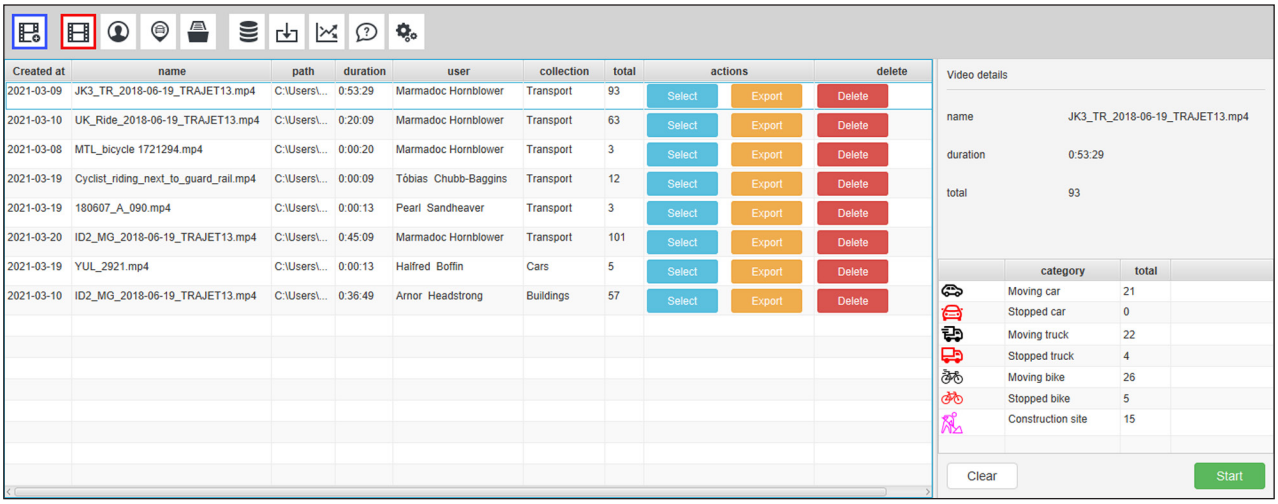

Figure 5 CRUD interface for managing videos.

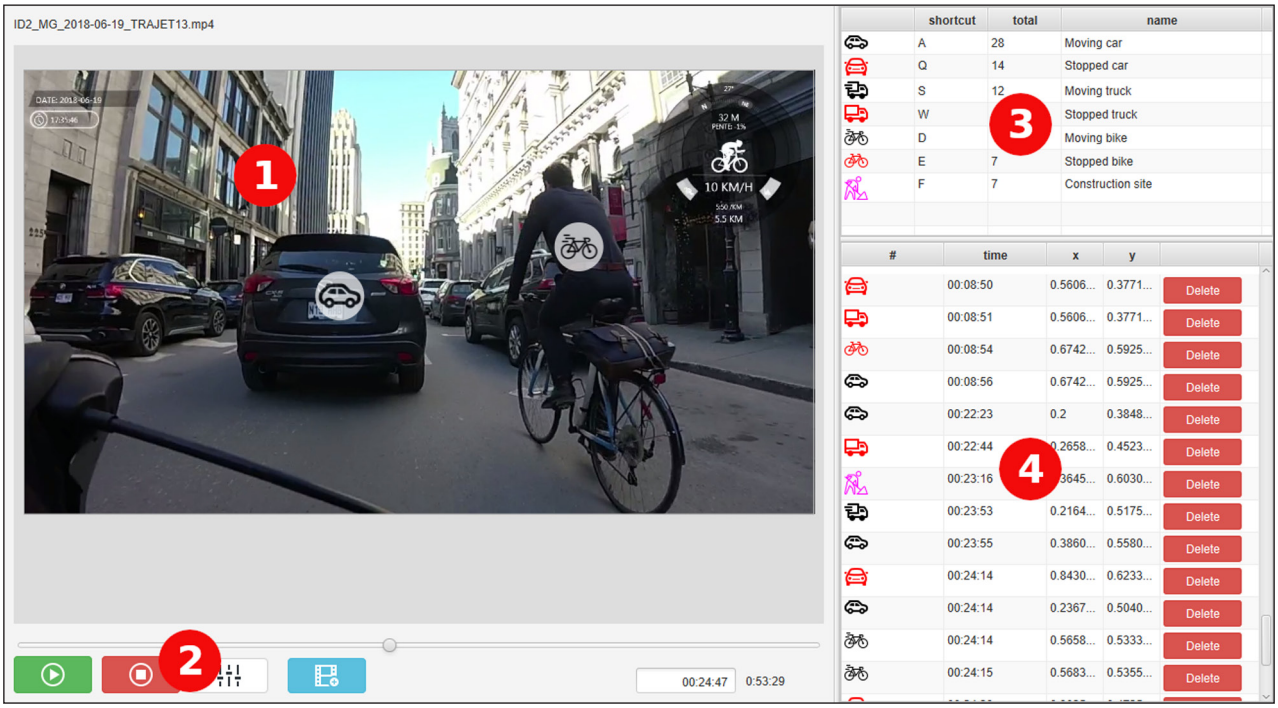

Figure 6 Interface for editing video.

precisely add an icon above it. In case of an error, it can be removed by either clicking the icon or with the delete button in the panel 4.

Clicking on the row of the table list will bring the video directly to the timestamp associated with the point. The user can also manually edit the text field of the elapsed time using the format HH:MM:SS.

The player functionalities offer a play button, pause button and a slider to scroll throughout the video. The controls button will open a separate window with sliders to adjust the playback speed, and different settings for the display of the icons (size, opacity, duration of display). The blue button allows selection of a second video file to be played in a separate window. It stays synchronized with the main player. This can be useful for watching second camera recording with a different angle.

Evaluating the agreement between two users

VFC allows cross validation between counts done by two users. This is a crucial feature to validate the counting process and ensure the robustness of the final measure.

This is achieved by pair-wise comparisons between counts coming from two users for the same video. Tabular and graphical outputs are provided to describe the level of agreement (***Figure 7***).

In the table, three columns are shown for each user: the number of times each category has been counted (Total), the numbers of matched (Concordant) and unmatched (Discordant) counts. Finally, the last column provides the global concordance index (***Figure 8***), defined as the percentage of all the recorded features that have been counted by the two counters.

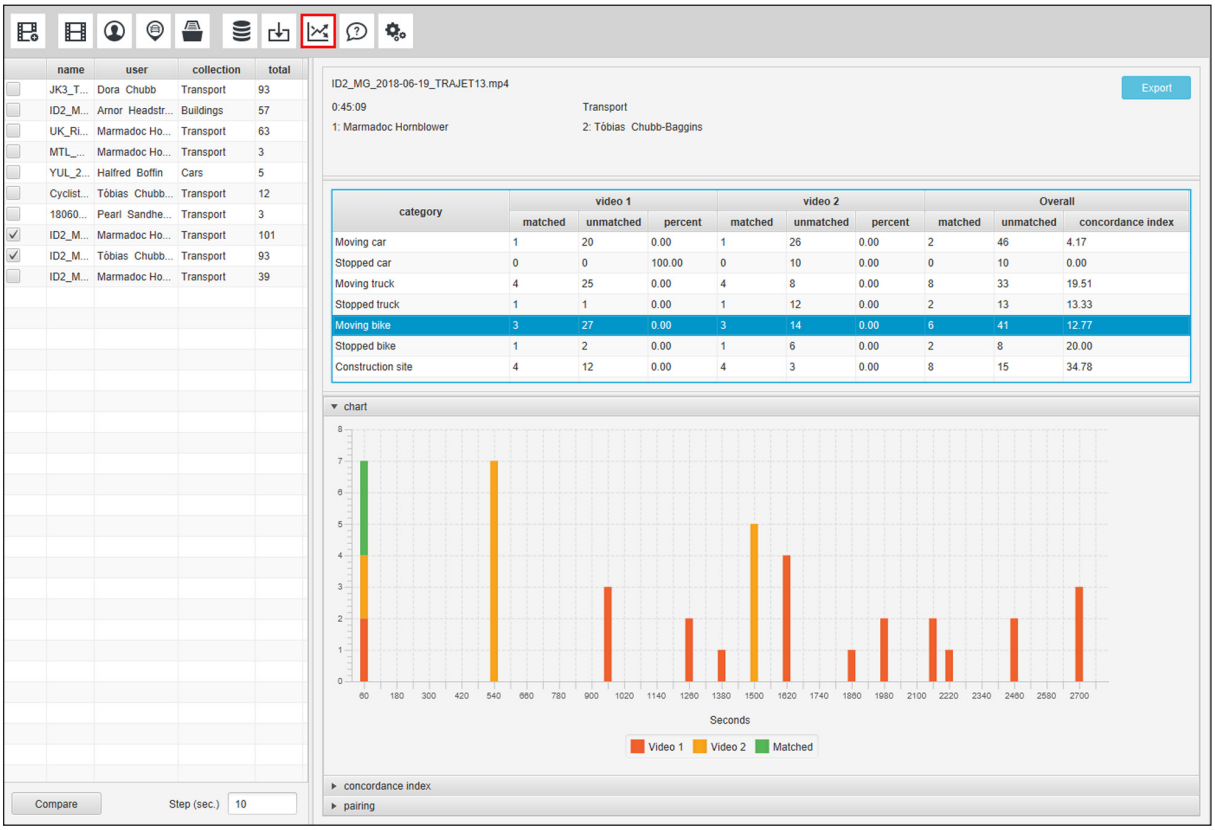

Figure 7 Concordance between two users.

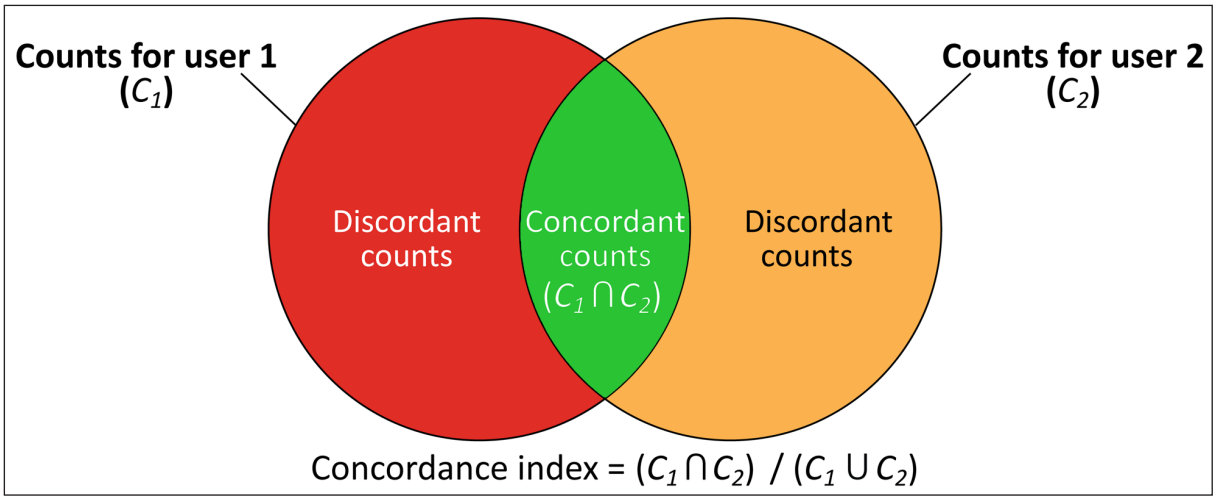

Figure 8 Concordance index between two users.

The Overall Concordance Index is calculated for the full video (last row of the table, ***Figure 5***). Note that if the difference in the number of features between categories is wide, the overall index will hide strong disagreement for categories with lower number of features. This is why a concordance index per category is also provided.

Finally, the chart provides temporally detailed information for a temporal window selected by the user (1 minute in the figure). Thus, during the video, the analyst can quickly find where the agreements and disagreements are strong. The bar chart shows in green the matched counts (agreements) and in red and orange the unmatched features from the two users (disagreements). A second chart represents the Concordance Index (%) calculated for each temporal window.

VFC can also export the counts and the matching of the counts as json files to allow a more in-depth analysis with other statistical software (e.g. R) if needed.

QUALITY CONTROL SECTION

All the entities and core services are fully tested with the Spock framework. The validation and relations between entities is done using annotation with Hibernate ORM. To guarantee data integrity, the configuration does not

allow any cascading deletion, which could accidentally result in corrupted data. The data access object layer (DAO) which execute the CRUD operations are tested against an in-memory H2 database. Serialization and deserialization of data is also tested to validate the import and export of json files. The statistic service is tested against different sets of data to check the accuracy of the Tarjan algorithm and the creation of graphs for matching features between two sessions.

Due to the constant changes in the UI, we did not automate UI testing and was done manually during development of the application. Any bugs can be reported on *github.com* by creating an issue.

(2) AVAILABILITY

OPERATING SYSTEM

Windows, Mac OS X and Linux operating systems.

PROGRAMMING LANGUAGE

Java with the Griffon framework 2.15.1 (*http://griffon-framework.org/*), an embedded H2 database (*https://www.h2database.com/html/main.html*), and the Bootstrap toolkit (*https://getbootstrap.com/*). The free icons used for Vifeco software are distributed by *http://iconmonstr.com*.

ADDITIONAL SYSTEM REQUIREMENTS

JavaFx media player supports a limited number of audio and video encoding types. Please refer to the JavaFx documentation at *https://openjfx.io/javadoc/11/javafx.media/javafx/scene/media/package-summary.html*.

DEPENDENCIES

Starting with Java 9, JavaFX is not bundled with Oracle nor Open JDK/JRE. JavaFX has been broken out into its own separate modules. Therefore, it must be downloaded separately and configured at compile and runtime. You can find more information on how to compile and run Java 11 programs with JavaFX at *http://openjfx.io*.

A better alternative is to use Liberica JDK from Bellsoft (*https://bell-sw.com/pages/downloads/*). Based on OpenJDK, it comes bundled and configured with JavaFX. Available for Java 8, 11 and 15, Windows, MacOS and Linux. This is the configuration we used for the development and is our recommendation to be used with Vifeco. It is also the runtime bundle with the Vifeco installer.

Another alternative is ZuluJDK from Azul (*https://www.azul.com/downloads/zulu-community/?package=jdk*). We do not guarantee compatibility with your application.

LIST OF CONTRIBUTORS

Project management: Philippe Apparicio (Full professor, INRS, Montreal, Canada). Software development: David Maignan (Undergraduate student in computer science and software engineering, UQAM, Montreal, Canada). Collaborator for the concordance module: Jérémy Gelb (PhD student in urban studies, INRS, Montreal, Canada). Contributors for translation: Jean-Marie Buregeya and Victoria Jepson (English), Dominique Mathon (Spanish).

SOFTWARE LOCATION

Archive

Name: Zenodo
Persistent identifier: DOI: *10.5281/zenodo.3417052*
Licence: Apache Licence 2.0
Publisher: David Maignan
Version published: 3.0
Date published: 09/04/2021

Code Repository

Name: GitHub
Identifier: *https://github.com/LAEQ/vifeco*
Licence: Apache Licence 2.0
Date published: 09/04/2021

LANGUAGE

English, French, Spanish

(3) REUSE POTENTIAL

The Vifeco has been designed to be a very versatile application and the functionalities developed have made it usable in a broad set of research fields.

Our urban studies laboratory uses the application for counting different types of encountered vehicles during bicycle trips (moving or stopped car, heavy vehicle, bicycles). Additionally, it can be used to record and categorize conflicts with other road users and pedestrians. In sum, every research project in need of a human count of objects appearing in recorded videos could make a great use of Vifeco. For example, Vifeco can be useful for studies analyzing recorded discourses, audiences, fauna, or interactions in public space. That is why we publish it as an open-source software to make it accessible and eventually modifiable for more specific research needs.

We strongly believe that this software could have a direct use in many research fields. The development of algorithms to automatically detect features in video is really promising; however, two problems limit this potential. First, most machine learning algorithms require a large number of pre-classified examples before being able to perform well, and thus are not suitable for many research projects. Second, the features to detect on videos risk being too subtle to be caught by a computer, such as in studies about peoples' interactions. In the same vein, when counting vehicles during bicycle trips, stationary vehicles with running engines must be distinguished from parked vehicles. Therefore, the human judgement remains and will remain a necessity in many projects despite the development of computer vision.

FUNDING INFORMATION

The publication of the paper was financially supported by the Canada Research Chair in Environmental Equity (950-230813) and the Social Sciences and Humanities Research Council of Canada (Insight Grant # 435-2019-0796).

COMPETING INTERESTS

The authors have no competing interests to declare.

AUTHOR AFFILIATIONS

Philippe Apparicio *orcid.org/0000-0001-6466-9342*
Environmental Equity Laboratory, Centre Urbanisation Culture Société, Institut national de la recherche scientifique, Montréal, Canada

David Maignan
Département d'informatique, Université du Québec à Montréal, Canada; Environmental Equity Laboratory, Centre Urbanisation Culture Société, Institut national de la recherche scientifique, Montréal, Canada

Jérémy Gelb *orcid.org/0000-0002-7114-2714*
Environmental Equity Laboratory, Centre Urbanisation Culture Société, Institut national de la recherche scientifique, Montréal, Canada

REFERENCES

1. **Pink S.** *Doing visual ethnography*. 2013. Sage.
2. **Luvaas B.** The Camera and the Anthropologist: Reflections on Photographic Agency. *Visual Anthropology*, 2019; 32(1): 76–96. DOI: *https://doi.org/10.1080/08949468.2019.1568115*
3. **Kindon S.** Participatory video in geographic research: a feminist practice of looking? *Area*, 2003; 35(2): 142–153. DOI: *https://doi.org/10.1111/1475-4762.00236*
4. **Garrett BL.** Videographic geographies: Using digital video for geographic research. *Progress in Human Geography*, 2011; 35(4): 521–541. DOI: *https://doi.org/10.1177/0309132510388337*
5. **Büscher M, Urry J.** Mobile methods and the empirical. *European Journal of Social Theory*, 2009; 12(1): 99–116. DOI: *https://doi.org/10.1177/1368431008099642*
6. **Hein JR, Evans J, Jones P.** Mobile methodologies: Theory, technology and practice. *Geography Compass*, 2008; 2(5): 1266–1285. DOI: *https://doi.org/10.1111/j.1749-8198.2008.00139.x*
7. **Tutenel P, Ramaekers S, Heylighen A.** Conversations between procedural and situated ethics: Learning from video research with children in a cancer care ward. *The Design Journal*, 2019; 22(sup1): 641–654. DOI: *https://doi.org/10.1080/14606925.2019.1595444*
8. **Mengis J, Nicolini D, Gorli M.** The video production of space: How different recording practices matter. *Organizational Research Methods*, 2018; 21(2): 288–315. DOI: *https://doi.org/10.1177/1094428116669819*
9. **Lynch H, Stanley M.** Beyond words: Using qualitative video methods for researching occupation with young children. *OTJR: occupation, participation and health*, 2018; 38(1): 56–66. DOI: *https://doi.org/10.1177/1539449217718504*
10. **Caldwell K, Atwal A.** Non-participant observation: using video tapes to collect data in nursing research. *Nurse researcher*, 2005; 13(2). DOI: *https://doi.org/10.7748/nr.13.2.42.s6*
11. **Chapman BJ, MacLAURIN T, Powell DA.** Video Observation and Data Coding Methods. *Food Protection Trends*, 2013; 33(3): 146–156.
12. **Bélisle F,** et al. Optimized video tracking for automated vehicle turning movement counts. *Transportation Research Record*, 2017; 2645(1): 104–112. DOI: *https://doi.org/10.3141/2645-12*
13. **Saunier N, Sayed T.** Automated analysis of road safety with video data. *Transportation Research Record*, 2007; 2019(1): 57–64. DOI: *https://doi.org/10.3141/2019-08*
14. **Jackson S,** et al. Flexible, mobile video camera system and open source video analysis software for road safety and behavioral analysis. *Transportation research record*, 2013; 2365(1): 90–98. DOI: *https://doi.org/10.3141/2365-12*
15. **Ismail K,** et al. Automated analysis of pedestrian–vehicle conflicts using video data. *Transportation research record*, 2009; 2140(1): 44–54. DOI: *https://doi.org/10.3141/2140-05*
16. **Struthers DP,** et al. Action cameras: bringing aquatic and fisheries research into view. *Fisheries*, 2015; 40(10): 502–512. DOI: *https://doi.org/10.1080/03632415.2015.1082472*
17. **de la Rosa CA.** An inexpensive and open-source method to study large terrestrial animal diet and behaviour using time-lapse video and GPS. *Methods in Ecology and Evolution*, 2019; 10(5): 615–625. DOI: *https://doi.org/10.1111/2041-210X.13146*
18. **LeBaron C,** et al. An introduction to video methods in organizational research. 2018; Los Angeles, CA: SAGE Publications.
19. **Jarrett M, Liu F.** "Zooming with" a participatory approach to the use of video ethnography in organizational studies. *Organizational Research Methods*, 2018; 21(2): 366–385. DOI: *https://doi.org/10.1177/1094428116656238*
20. **Badrinarayanan V, Kendall A, Cipolla R.** Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 2017; 39(12): 2481–2495. DOI: *https://doi.org/10.1109/TPAMI.2016.2644615*
21. **Vondrick C, Patterson D, Ramanan D.** Efficiently scaling up crowdsourced video annotation. *International Journal of Computer Vision*, 2013; 101(1): 184–204. DOI: *https://doi.org/10.1007/s11263-012-0564-1*

22. **Bradski G.** The opencv library. *Dr Dobb's J. Software Tools*, 2000; 25: 120–125.
23. **Sobral A, Bouwmans T.** Bgs library: A library framework for algorithm's evaluation in foreground/background segmentation. In T. Bouwmans, et al., (Eds.), *Background Modeling and Foreground Detection for Video Surveillance*. 2014, CRC Press, Taylor and Francis Group.
24. **Sobral A.** Vehicle Detection, Tracking and Counting. 2014; Available from: *https://github.com/andrewssobral/simple_vehicle_counting*.
25. **Sobral A.** Vehicle Detection by Haar Cascades with OpenCV. Available from: *https://github.com/andrewssobral/vehicle_detection_haarcascades*.
26. **Bouvie C,** et al. Tracking and counting vehicles in traffic video sequences using particle filtering. In *2013 IEEE international instrumentation and measurement technology conference (I2MTC)*. 2013. IEEE. DOI: *https://doi.org/10.1109/I2MTC.2013.6555527*
27. **Mithun NC, Rashid NU, Rahman SM,** Detection and classification of vehicles from video using multiple time-spatial images. *IEEE Transactions on Intelligent Transportation Systems*, 2012; 13(3): 1215–1225. DOI: *https://doi.org/10.1109/TITS.2012.2186128*
28. **Ismail K, Sayed T, Saunier N.** Automated analysis of pedestrian-vehicle: Conflicts context for before-and-after studies. *Transportation Research Record: Journal of the Transportation Research Board*, 2010; 2198: 52–64. DOI: *https://doi.org/10.3141/2198-07*
29. **Saunier N, Sayed T, Ismail K.** Large-scale automated analysis of vehicle interactions and collisions. *Transportation Research Record: Journal of the Transportation Research Board*, 2010; 2147: 42–50. DOI: *https://doi.org/10.3141/2147-06*
30. **Buch N, Orwell J, Velastin SA.** 3D extended histogram of oriented gradients (3DHOG) for classification of road users in urban scenes. 2009. DOI: *https://doi.org/10.5244/C.23.15*
31. **Messelodi S, Modena CM, Zanin M.** A computer vision system for the detection and classification of vehicles at urban road intersections. *Pattern analysis and applications*, 2005; 8(1–2): 17–31. DOI: *https://doi.org/10.1007/s10044-004-0239-9*
32. **Buch N, Velastin SA, Orwell J.** A review of computer vision techniques for the analysis of urban traffic. *IEEE Transactions on Intelligent Transportation Systems*, 2011; 12(3): 920–939. DOI: *https://doi.org/10.1109/TITS.2011.2119372*
33. **Yang CD, Najm WG.** Examining driver behavior using data gathered from red light photo enforcement cameras. *Journal of safety research*, 2007; 38(3): 311–321. DOI: *https://doi.org/10.1016/j.jsr.2007.01.008*
34. **Hediyeh H,** et al. Pedestrian gait analysis using automated computer vision techniques. *Transportmetrica A: Transport Science*, 2014; 10(3): 214–232. DOI: *https://doi.org/10.1080/18128602.2012.727498*
35. **Ismail K, Sayed T, Saunier N.** Camera calibration for urban traffic scenes: Practical issues and a robust approach. In *89th Annual Meeting of the Transportation Research Board*, Washington, DC. 2010.
36. **Zangenehpour S, Miranda-Moreno LF, Saunier N.** Automated classification based on video data at intersections with heavy pedestrian and bicycle traffic: Methodology and application. *Transportation research part C: emerging technologies*, 2015; 56: 161–176. DOI: *https://doi.org/10.1016/j.trc.2015.04.003*
37. **Layka V,** et al. Introduction to Griffon. In *Beginning Groovy, Grails and Griffon*. 2013, Springer. 305–331. DOI: *https://doi.org/10.1007/978-1-4302-4807-1_13*

TO CITE THIS ARTICLE:
Apparicio P, Maignan D, Gelb J 2021 VIFECO: An Open-Source Software for Counting Features on a Video. *Journal of Open Research Software*, 9: 7. DOI: *https://doi.org/10.5334/jors.300*

Submitted: 20 September 2019 Accepted: 29 April 2021 Published: 07 May 2021