BioFoV - An Open Platform for Forensic Video Analysis and Biometric Data Extraction

Miguel Reis Moitinho de Almeida

May, 2015

Abstract

This paper proposes an open platform, Biometric Forensic Video analyzer (BioFoV), for forensic video analysis and biometric data extraction. The platform's architecture and implemented modules are described, sample results are shown, and a list of possible enhancements is included. BioFoV is implemented with open software, can be run in multiple software platforms, and is designed to be easily extended. An implementation is publicly available, supporting camera calibration, detection of events and allowing the computation of several soft biometrics, such as bodily measurements or joint angles, taking as input surveillance video footage. Also face detection is available to illustrate the ability to deal with different biometric traits.

1 Introduction

Forensic experts have discussed over the last decades the need for moving the foundation of forensic case work from identification of perpetrators based on expert knowledge to expressing evidence based uncertainties. Barriers to this include both limited time and lack of knowledge to be engaged in research and development for many forensic experts.

Biometric researchers, on the other hand, have done much research often without access to data from cases found in real forensic scenarios, and which they generally need to further research. The European Cooperation in Science and Technology (COST) Action 1106 "Integrating Biometrics and Forensics for the Digital Age" was established to bring these two communities together to foster cooperation and new research.

Techniques, which can enhance forensic casework, have already been developed such as [1], [2] and [3]. However, there is still the challenge to transform the algorithms, e.g. developed in the context of PhD theses, into a set of tools that can be used by forensic experts.

Currently forensic scientists often use commercially available general image/video processing software packages, not specifically developed for forensic case work. An example is the comparison of measurable silhouettes of a perpetrator and a suspect, illustrated in Figure 1. The creation of these silhouettes is the result of quite laborious processes, first involving commercial photogrammetric software, primarily developed for 3D scans of objects using photos taken with the same high resolution camera from different angles, to construct a 3D model wherein silhouettes as those shown in Figure 1 can be created. Another step is to incorporate in the 3D model frames from a low resolution surveillance camera often subjected to significant distortions. This is a challenge for the photogrammetric software, with the internal and external calibration of the surveillance camera often being a problematic and laborious process. Next challenge is that the software is not build to handle several photos from the same location in the 3D-model, as taken from a stationary surveillance camera. At last, the created silhouettes have to be superimposed and compared, as shown in the right image of Figure 1, using another piece of software.

Hence, from a forensic expert practitioners point
of view, there is the need for forensic casework tailor-
made software, based on state of art knowledge in
fields such as soft biometrics, gait recognition, pho-
togrammetry, etc. Forensic experts, police officers
and others working with forensic video analysis can
also benefit from more ”intelligent” video handling
tools, such as automatic event/face detection, to help
search the relevant video footage prior to the actual
case work.

Other software packages aimed at forensic video
and image analysis have been developed, such as
Forevid [4] or the tools developed by Ocean Systems
[5], although more directed either for video capture
and processing or enhancement of the material, not
fully addressing the above mentioned challenges.

A tool for forensic gait analysis has recently been
developed [6] and automatic facial algorithms are
valuable tools for the forensic expert [7]. Including
such tools into a common and extensible software
platform would allow narrowing down the software
tools a forensic video analysers needs to master, and,
more importantly, would support the combination of
results from different algorithms to enhance overall
forensic analysis results.

This paper proposes such a software platform, Bio-
FoV, resulting from the cooperation between a bio-
metrics research group and an experienced group of
forensic practitioners. In BioFoV it is possible to add
new modules or update the existing ones when new
relevant research emerges making new developments
available to the community, instead of remaining in
a hard-disk once a research project is completed.

2 Software Platform Proposal

The proposed BioFoV software platform objective is
to provide an extensible tool, useful to forensic inves-
tigators by fitting their needs and thereby minimising
the required workload. By automating a set of
time consuming and laborious tasks, it is expected
to enable a deeper case analysis, requiring less effort
from the investigator, while allowing to focus the ef-
fort on those case aspects that require expert insight, thus contributing to an overall quality
increase.

Two of the key characteristics of this proposal are:

1. The usage of open software, to avoid complex
   licensing; and

2. Its extensibility, to allow the inclusion of new
   modules adding functionality, as well as the poss-
   ability to include alternative or improved ver-
   sions for existing modules.

The proposed BioFoV software platform is thus struc-
tured around a set of functional modules, also includ-
ing a Graphical User Interface (GUI) to be usable by
forensic analysers of different backgrounds. The fol-
lowing subsections describe the modular architecture
proposed.

2.1 Data Classes

BioFoV is dedicated to the analysis of videos, typi-
cally captured by surveillance cameras, available as
forensic evidence. Therefore, the main data classes
used as building blocks for the platform development
are:

- Video – Used to read and perform operations
  on a video file, mainly seeking, grabbing frames,
and checking properties such as number of frames, frame rate, among others.

- **Frame** – Used to describe a single frame. Its constructor saves an image on disk so that a large number of frames can be grabbed without filling up the computer’s memory.

- **Event** – Describes a set of Frames (contiguous in time). It holds the output of the event detector module, described later on.

- **Mask** – Specifies a part of a Frame.

### 2.2 Modules

The motivation of the BioFoV software platform development is the functionality to be offered to forensic researchers. Therefore the platform should be modular and extensible.

Each of the modules is a piece of code that implements a certain functionality. An initial set of modules has been implemented targeting camera calibration, automatic video analysis to detect relevant motion activity, and also biometric feature extraction, notably soft biometrics, such as bodily measurements or joint angles.

### 2.3 User Interface

The GUI allows the non-programming user to take advantage of the implemented functionalities. It supports the selection of input video contents and respective display, as well as information about the current user selections and, of course, to execute the required module functionalities. A screenshot of the main window is shown in Figure 2.

![Figure 2: User interface](image)

### 3 Implementation

This section describes the tools used to create the BioFoV software platform and some of the implemented modules.

BioFoV is being developed in C++ using Qt [8] for the GUI, and the Open Source Computer Vision (OpenCV) [9] library for supporting computer vision and image processing functionalities.

Qt and OpenCV were chosen with interoperability in mind. Their usage enables the same software to be compiled for any of the most commonly used platforms. So far it has been successfully built and tested on GNU/Linux systems (Debian and Ubuntu) and Microsoft Windows. An OS X build should also be easily compiled on the target platform.

All the developed software is Free and Open Source, therefore the final product can be shipped without any restrictive licenses or extra fees, allowing it to be freely shared amongst those wanting to use and/or improve it.

### 3.1 User Interface

User interface development is implemented using Qt, which provides the necessary GUI building blocks to create a cross platform tool. Since the objective of BioFoV is to bring tools to users, it was imperative...
to have an easy to use GUI.

As shown in Figure 2, the GUI main window is organized into four main areas: The top menu, where all the functionalities are listed in independent sub-menus; The input and result tabs, where inputs and outputs are listed; The details area, where extra information about what is selected or being played back can be shown to the user; The playback area, supporting user interaction with the video or still image, via mouse input, and supporting video exploration actions like seeking, zooming, pausing or taking snapshots.

The present structure of the GUI can still be improved, as further feedback from practitioners is gathered on the best layout for user interaction.

3.2 Implemented Modules

The current BioFoV version includes some generic modules, expected to be used by most practitioners, such as camera calibration or event detection. Also a couple of more specific modules have been included, notably for the computation of soft biometrics such as bodily measurements, or facial detection, as examples of possible feature extraction modules.

3.2.1 Camera Calibration

Since even the best video cameras distort the captured images, it is necessary to calibrate the camera with which images are taken, to compensate for distortions and be able to support accurate measurements.

By recording a video of a non symmetrical chess pattern with the same camera, the calibration parameters can be calculated and saved to a file. Later it can be imported for usage with the surveillance video being analysed, or with any other video captured by that camera.

In the present implementation, this module is, in its essence, a wrapper for two OpenCV functions, findChessboardCorners and calibrateCamera, being the latter based on [10] and [11].

The software extracts a set of frames containing the pattern, whose size is known. Each frame is converted to grayscale and passed to findChessboardCorners, which tries to find the location of the pattern in the image. If it is found, the coordinates of each inner corner of the pattern are stacked so that they can later be used by calibrateCamera, to compute the camera calibration parameters.

3.2.2 Event Detection

When a surveillance video has to be analysed thoroughly, a lot of man time needs to be spent to visualize it, even if nothing relevant is happening. This module allows a faster analysis of long videos, automatically looking for the occurrence of events. In the present implementation, this module trims the parts that have no movement, so that the user does not have to watch the entire video, which in some cases can be several days long.

Using an adaptive background subtraction model [12], the algorithm checks each frame of the video for significant changes, allowing the user to set a threshold, above which changes are considered significant enough to be analysed.

3.2.3 Re-projected Image Plane Measurements

This module enables the user to measure distances in pixels in a re-projected image plane. Prior to any kind of measurement, the camera has to be calibrated as mentioned before.

An image transformation tool has been included in this module, allowing an image to be re-projected to a vertical plane captured in a frame. This allows, for example, the comparison of heights or widths between known static planar objects and those of items of interest or to estimate bodily measurements of individuals in the re-projected plane.

The implemented tools allow measurement of: Linear distance between two points; Vertical distance between two points; Horizontal distance between two points; Angles.

It should be noticed that bodily measurements of individuals computed using this tool, for instance the height, is precise only under very strict conditions, such as: The individual must be standing straight;
Must be in the same vertical plane as the one used for re-projection; The point on the top of the head must be well defined, as well as its vertical projection on the floor.

These conditions are very seldom met at the same time. However, alternative measurements may be computed, as the height to the eyes, the length of a limb, or the angle between leg and foot in a given part of the gait cycle.

To give more freedom to the user and avoid the restriction of the individual having to stand in the same plane as the referential, a more complete spatial measurement module needs to be developed, as described in the following.

### 3.2.4 Spatial Measurements

The objective of this module is to allow, from several post event photos of the same scene, eventually captured from different angles, to reconstruct points, edges, planes and volumes that can be used as measurement reference. From these, virtual references can be constructed and overlayed with the video, allowing bodily measurements in arbitrary locations in the scene, thus avoiding the coplanar restriction imposed by the previously described image measurement module. This module is not yet available.

### 3.2.5 Feature Extraction - Face Detection Example

This module enables the user to automatically detect and extract a generic feature, from a set of videos, or a set of previously detected events. New instances of this module can be created to accommodate the type of analysis desired by investigator. In the present BioFoV version a feature extraction module dedicated to frontal face detection has been included.

The feature extraction module is the implementation of a Haar Feature-based Cascade Classifier, originally proposed in [13], improved by [14], and implemented in OpenCV.

With this automated method, the user can easily create a facial database of detected people in the video. The clustering of these faces by person is something that is currently being explored by another Master of Science (MSc) student.

The cascade classifiers used by the face detection algorithm are loaded to the program from an Extensible Markup Language (XML) file, which enables the modification of the set of features to be used without any required change to the program. Other features like the full human body, or any other described by a Haar classifier can also be automatically extracted.

### 3.2.6 Other Miscellaneous Modules

Other modules have been implemented with usability in mind. Amongst them are: A simple printing option, that enables quick physical access to the image in the display; An export tool for extracted features, where they can be extracted in bulk to separate image files; A tool for exporting events that were detected automatically by the event detection module to video files, either for transport purposes or to prepare materials to be presented in court.

### 4 Preliminary Results

This section presents some of the results that can be obtained with the current implementation of the BioFoV software platform.

The reported results were obtained using a computer with an Intel® Core™ i7-3770 CPU clocked at 3.40 GHz, running Kubuntu 14.04 AMD64 and OpenCV 2.4.11 compiled with Intel® Threading Building Blocks (TBB) support.

The input videos considered for the illustration examples were captured with a Logitech c270 webcam, with a 640 by 480 pixel resolution, a frame rate of 10 Frames Per Second (FPS) and the output videos were captured using ffmpeg and encoded with an H264 codec.

### 4.1 Camera Calibration

The webcam in use was calibrated using BioFoV with a 1016 frames video of a slowly moving chess pattern with 7 × 9 inner corners (as shown in Figure 3a). In order to capture the edge distortions properly, the pattern was moved along the areas captured...
near the image border, resulting in the detection of 369 frames, in total, containing the complete pattern.

Using this set of frames, the total sum of squared distances between the observed feature points and the projected object points was below 1. An illustration of the calibration procedure is shown in Figure 3.

Since the algorithm is applied to patterns captured from a video, they are sometimes blurry, which can affect the precision of the calibration. For this reason, the pattern should be moved slowly and under good illumination conditions while performing the calibration.

### 4.2 Event Detection

The event detection algorithm was tested on a sample video, captured in a controlled environment, containing around 2.4M frames, at 10 FPS, lasting 2 days, 11 hours and 30 minutes. It took the BioFoV software 7 hours and 11 minutes to analyse the complete video, identifying a set of events considered relevant with a total length of 17 minutes (210 times shorter). This example illustrates that a real advantage can be provided to the forensic analyser, saving a considerable amount of time for analysing a video.

This video sequence was shot indoors, in a room containing a large window surface area and where artificial illumination was used for some periods, thus being subjected to significant light changes. The adaptive background subtraction algorithm was able to handle the smooth light changes caused by the changing natural illumination without detecting them as events.

The same video was manually analysed and no relevant events were missed by the algorithm. There were some false positive events detected, notably caused by the lights in the corridor being turned on and off during the night, as the room has a small window to the corridor.

Whenever a subject enters the monitored space, or some object is brought into the scene, and remains still for more than the time selected using the parameter related to the history of the background subtractor, it becomes part of the background and is ignored until it moves again.

A frame of one of the detected events is shown in Figure 4, including the original frame, the computed mask, and the corresponding masked frame.

### 4.3 Image Measurements

For an image measurement to be accurate, the subject needs to be very well aligned with a plane, standing up straight, and both the top of the head and central feet point must be clear in the image. This is a very rare case, and it was concluded that it is best to compare heights in similar poses than to try to estimate the real height of a perpetrator with a bad posture.

An example of a measurement is presented in Figure 5. In this case the measurement conditions are close to ideal, and very seldomly expected to be found in real life footage. Because of the limitation of only being able to take a measurement within a given reference plane, the ability to perform spacial measurements with real 3D points is being considered for implementation.

Nevertheless, for the example of Figure 5, and using a previously calibrated camera, it is possible to obtain interesting results like the ones presented in Table 1.
Table 1: Measurement results

<table>
<thead>
<tr>
<th>Subject</th>
<th>Board</th>
<th>Person</th>
</tr>
</thead>
<tbody>
<tr>
<td>Height (meters)</td>
<td>0.91</td>
<td>1.72</td>
</tr>
<tr>
<td>Measured Height (pixels)</td>
<td>184</td>
<td>341</td>
</tr>
<tr>
<td>Measured Height (meters)</td>
<td>–</td>
<td>1.68</td>
</tr>
</tbody>
</table>

Figure 5: Height measurement with a reference

(a) Calibrated (b) Re-projection (c) Relative height

4.4 Feature Extraction

To test the feature extraction module a face detector has been included in BioFoV. It is based on a Haar classifier [15] able to detect multiple face instances in the same frame.

Examples of faces detected in the test video are shown in Figure 6.

Four different frontal face classifiers were used, all of which are included with OpenCV. As can be observed by the comparison in Table 2, alt_tree gave much less false positives than any other, but the one which missed the least faces was the alt2. So there is a trade-off in this situation, where depending on the time available versus the thoroughness intended, the user can choose one classifier or the other.

It also successfully detects multiple subjects in the same frame.

Table 2: Face detection classifiers

<table>
<thead>
<tr>
<th>Classifier</th>
<th>Subject 1</th>
<th>Subject 2</th>
<th>False Positives</th>
</tr>
</thead>
<tbody>
<tr>
<td>default</td>
<td>76</td>
<td>22</td>
<td>169</td>
</tr>
<tr>
<td>alt</td>
<td>77</td>
<td>21</td>
<td>48</td>
</tr>
<tr>
<td>alt2</td>
<td>78</td>
<td>21</td>
<td>64</td>
</tr>
<tr>
<td>alt_tree</td>
<td>71</td>
<td>19</td>
<td>3</td>
</tr>
</tbody>
</table>

Figure 6: Feature extraction results

5 Conclusions and Future Work

The BioFoV software platform provides a tool for forensic video analysis and biometric data extraction. It is made freely available to the community and since it is based of open software tools it can be continuously expanded with the integration of new state of the art algorithms and used without licensing problems. It is implemented in such a way that the same software code can be compiled for different platforms and offers a consistent and user friendly GUI.

It already has some implemented features, and there are ongoing efforts to implement new tools, and improve existing ones. There is also the commitment to ensure the continuity of the project.

5.1 Future Development

Since the platform is extensible, some long term modules have been identified as candidates for implementation:

- Automatic silhouette extraction and matching;
- Face matching;
- Camera calibration using the plumb-line constraint and minimal Hough entropy[16];
- 3D point cloud visualization;
• 3D scene reconstruction;
• Implementation of a file format.

None of the implemented modules is perfect. The objective was not to make them perfect in the first iteration but to lay down the foundations and leave room for improvement and easy integration of state of the art algorithms.

One example of of such possible improvements is the calibration tool which currently uses the OpenCVdefault functions. There are alternative algorithms (such as AprilCal [17]) which can potentially provide improved results that can also be implemented, eventually letting the user choose which to use with each camera.

One of the main objectives for this project is that the development does not stop here, but instead continues with the cooperation of all the labs which are interested in making BioFoV a better and even more useful tool.

All the code is documented using Doxygen [18], allowing anyone with programming knowledge to be able integrate new modules and share them with all BioFoV users.

In order to allow contributions from the community, a public GitHub [19] repository has been set up, from which the builds of the program can be compiled and accessed.

The management of the project, and code review of any new feature submission, as well as tracking of bug reports will be ensured by the current team. This will allow tight quality control on the program releases.

References


