How to Set Up PlumeTrAP

PlumeTraP was developed using MATLAB R2018b and the Image Processing Toolbox 10.3, but it was also tested with the R2021b release and the Image Processing Toolbox 11.4, and found to be completely operative. The operating systems used to develop and check its functioning are Windows 10 Pro 64-bit (OS build 19044), MAC OS X El Capitan 10.11.6 and Ubuntu 20.04.3 LTS 64-bit. PlumeTraP was tested with videos of MPEG-4 file format (.mp4). For other supported video file formats, please check the specific MATLAB support webpage [39].

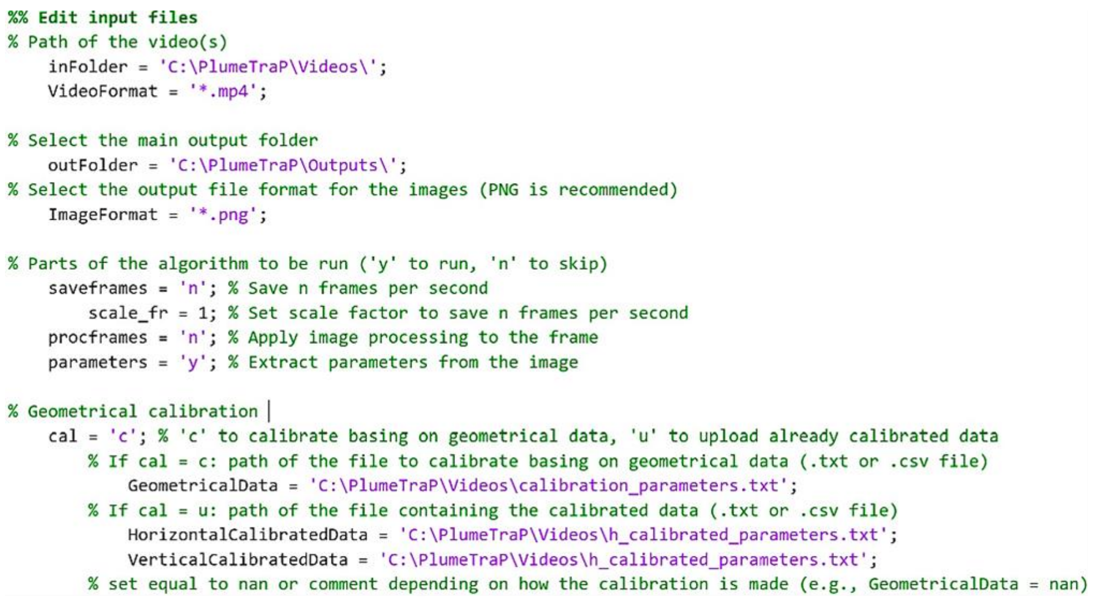
Two versions of PlumeTraP were created, one to semi-automize the process (*PlumeTraP_AUT.m*) and another with a more user-friendly purpose (*PlumeTraP_INT.m*). Here, the general workflow of PlumeTraP will be presented. For specific functioning or setting up, the user may refer to the MATLAB scripts, which are appropriately documented.

Appendix A.1. Semi-Automatic Version

PlumeTraP_AUT.m must be set up in the *Edit input files* section of the script (Figure A1). The data to be analyzed are selected from an input folder, where one or more videos can be saved. Thus, it is possible to analyze multiple videos in a single execution of the script. Once the main output folder and the output image format have been entered by the user, it is possible to decide which processing steps the user wants to perform, including frame saving, frame processing and parameter calculation. Thus, for example, once the frames are saved, it is not necessary to repeat this process to re-run other parts of the algorithm. Frames are extracted using a parameter that expresses how many frames per second the user wants to save.

To apply the geometrical calibration and calculate the parameters of the plume, some known a priori parameters must be inserted. This is achieved through an input .txt or .csv file with columns that should follow a specific order and format: name of the video, minimum camera–image plane distance Y^{near}, maximum camera–image plane distance Y^{far}, horizontal FOV β_h, vertical FOV β_v and inclination of the camera ϕ. The camera–image plane distances are calculated along the camera orientation and do not follow the topography. Specifically, these are the distances from the camera to the lines perpendicular to the camera orientation that intersect the nearest and the farthest extent of the crater, respectively. Thus, the camera position and orientation are required knowledge (Figure A2). These calculations can be done through any available geographic information system (GIS) software or simply with Google Earth. If the camera–vent distance is certain, the two values should be equal.

It is also possible to avoid the geometrical calibration if the user has already done it manually, simply by saving two .txt or .csv files. The first one must be a 2-by-N_h file containing the horizontally calibrated position of each of the N_h pixels along the first row and the related horizontal error on the second row. The other file must be a must be a 2-by-N_v file with the vertically calibrated position of each of the N_v pixels and the related vertical error. Calibrated positions must be referenced to the leftmost pixel for the horizontal calibration and to the bottom (or to the pixel pointing towards an inclined camera if this is the case) for the vertical calibration. Both must be entered into the files in ascending order.



```
%% Edit input files
% Path of the video(s)
    inFolder = 'C:\PlumeTraP\Videos\';
    VideoFormat = '*.mp4';

% Select the main output folder
    outFolder = 'C:\PlumeTraP\Outputs\';
% Select the output file format for the images (PNG is recommended)
    ImageFormat = '*.png';

% Parts of the algorithm to be run ('y' to run, 'n' to skip)
    saveframes = 'n'; % Save n frames per second
        scale_fr = 1; % Set scale factor to save n frames per second
    procframes = 'n'; % Apply image processing to the frame
    parameters = 'y'; % Extract parameters from the image

% Geometrical calibration |
    cal = 'c'; % 'c' to calibrate basing on geometrical data, 'u' to upload already calibrated data
        % If cal = c: path of the file to calibrate basing on geometrical data (.txt or .csv file)
            GeometricalData = 'C:\PlumeTraP\Videos\calibration_parameters.txt';
        % If cal = u: path of the file containing the calibrated data (.txt or .csv file)
            HorizontalCalibratedData = 'C:\PlumeTraP\Videos\h_calibrated_parameters.txt';
            VerticalCalibratedData = 'C:\PlumeTraP\Videos\h_calibrated_parameters.txt';
        % set equal to nan or comment depending on how the calibration is made (e.g., GeometricalData = nan)
```

Figure A1. Screenshot showing the *Edit input files* section of *PlumeTraP_AUT.m*.

If the original video is not available and the user wants to analyze already saved or processed frames, it is sufficient to comment out the *Start the process* line and the *Reading video* section, and define a name (e.g., *name* = *'Explosion_1'*) and an extension (e.g., *ext* = *'.mp4'*) of a hypothetical video. The variable *name* must correspond to the first part of the name of the folder where the frames are saved (i.e., *Explosion_1_Frames* or *Explosion_1_Processed*; see Appendix A.3 for the folder structure). Frames inside this folder should be named in numerical order and with the same number of decimals.

If running this version of PlumeTraP, the only further action required from the user is to set up the image analysis technique, that is explained in Appendix A.3.

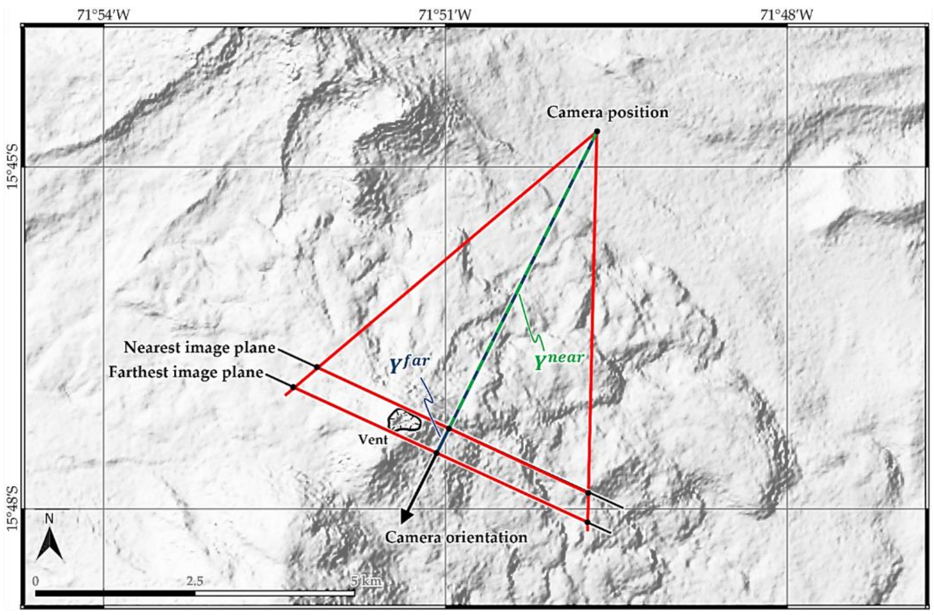
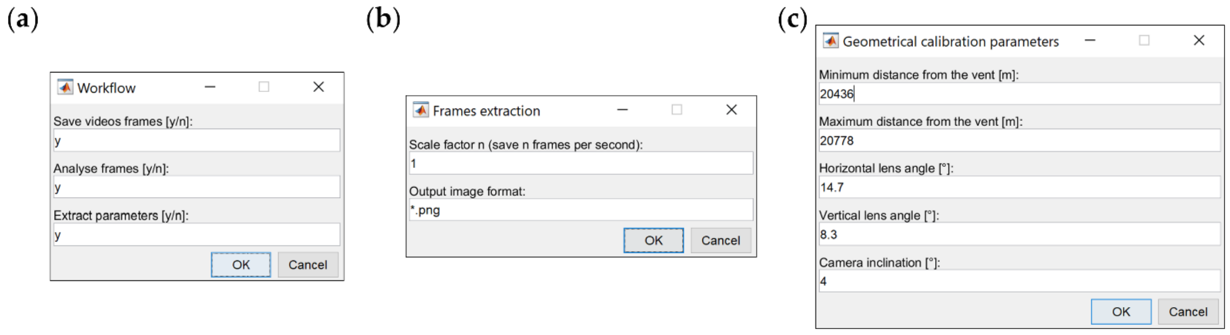


Figure A2. Terrain map of the Sabancaya area showing how the camera–image plane distances are determined. Both distances are calculated from the camera position to the nearest and farthest possible image planes, respectively, Y^{near} and Y^{far}, along the orthogonal line that corresponds to the horizontal projection of the camera orientation.

Appendix A.2. Interactive Version

PlumeTraP is also available as an interactive version, *PlumeTraP_INT.m*, with the intention to distribute a user-friendly software, without the need to modify the scripts to run the video analysis. Therefore, all the required input parameters presented in Appendix A.1 are input by the user using a graphical interface. In particular, the user has to complete entries in MATLAB dialog boxes (Figure A3) and to select the video(s) to be analyzed and the main output folder through system windows (File Explorer, Finder or Nautilus, depending on the operating system). System windows also open to select the video's frame or processed output folder if the software is partially run (e.g., frames are already saved in the specific folder). Some differences from the semi-automatic version regard the video(s) selection, in which videos with different formats can be selected, and the processing steps, that the user must specify for each video. The calculation of the camera–image plane distances follows the procedure described in Appendix A.1.



(a) Workflow

Save videos frames [y/n]:
y

Analyse frames [y/n]:
y

Extract parameters [y/n]:
y

(b) Frames extraction

Scale factor n (save n frames per second):
1

Output image format:
*.png

(c) Geometrical calibration parameters

Minimum distance from the vent [m]:
20436

Maximum distance from the vent [m]:
20778

Horizontal lens angle [°]:
14.7

Vertical lens angle [°]:
8.3

Camera inclination [°]:
4

Figure A3. Dialog box used to insert parameters in *PlumeTraP_INT.m*, for (**a**) the processing steps, (**b**) the frame extraction and saving parameters and (**c**) the geometrical calibration parameters.

Appendix A.3. General Workflow of PlumeTraP

Both PlumeTraP main scripts are based on the same functions, thus they have in common the procedures described in this section. They also share the same folder structure (Figure A4), with two subfolders being saved inside the video output folder: one for the original frames and another for the processed frames (ending with *_Frames* and *_Processed*, respectively). The video output folder is a subfolder of the main output folder set by the user and is also where the final output files with the physical plume parameters are saved.

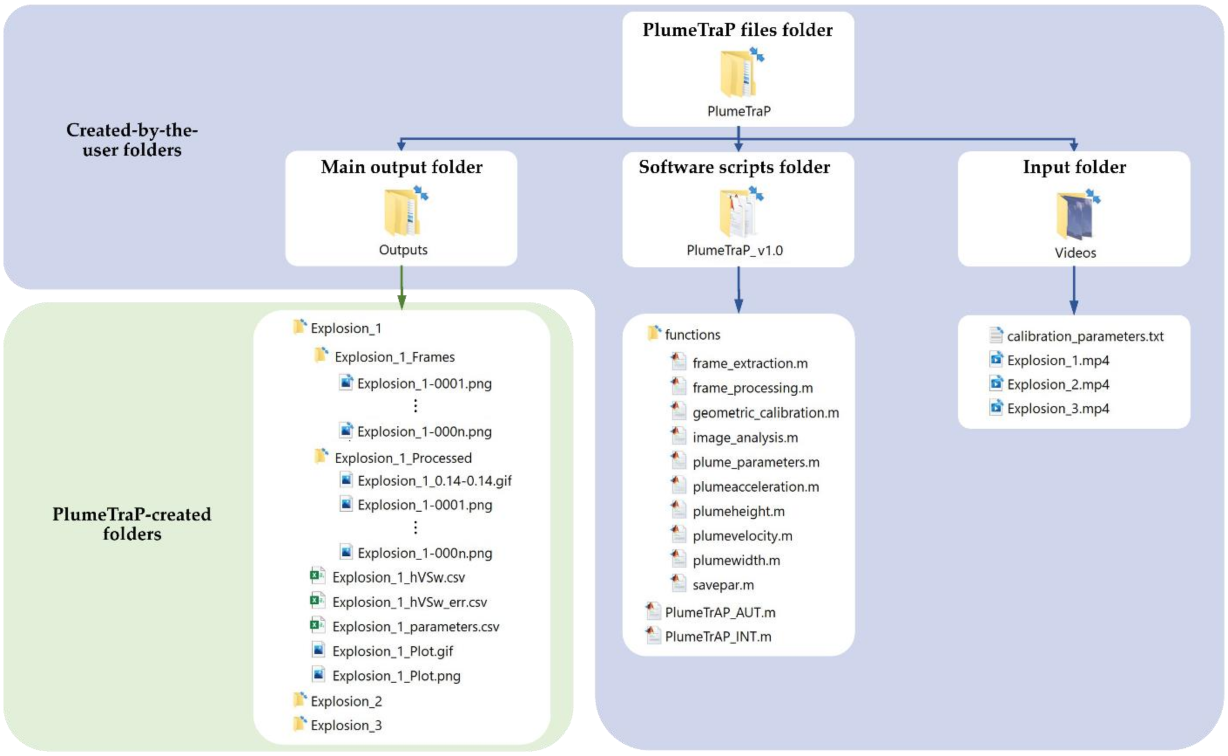


Figure A4. Structure of folders used by PlumeTraP. The main output folder and the input folder paths are set or selected by the user when initializing PlumeTraP, while the software scripts folder should just be in the MATLAB path. The output folders and files are automatically created when running the software.

After reading the video, the first called function performs the frame extraction and saving (*frame_extraction.m*). Frames can be saved in various formats, but .png is highly recommended as its compression prevents quality loss, and the filenames are successive integers. Call *imformats* to see a list of supported formats and their file extensions.

Once the frames from the video are saved, the *frame_processing.m* function is called. The following pre-processing procedure is applied to perform the image analysis:

1. Before starting the processing, a dialog box asks for the user to input the threshold luminance value used to create a binary image. The threshold value cannot be fixed as it reflects the luminance condition of the video and, therefore, has to be specified as a numerical scalar or numerical array with values in the range between 0 and 1, although, generally, values ranging from 0.05 to 0.2 should be used. This value can also be set using different values for the first image than for the other ones (helpful to obtain a clean background-isolated image).
2. The image analysis procedure (described in Section 3.1) is applied by the *image_analysis.m* function to the last frame to show a preliminary result of the analysis (Figure A5). Thus, the algorithm recognizes the plume shape in the last frame by keeping only the biggest object that has value equal to 1 in the binary image. At this point, a dialog box asks if the selected parameters isolate the plume sufficiently well. If this is not the case, it is possible to restart the pre-processing and set new thresholds, as this part of the script is inside a while loop that can be run until a satisfactory output result, mainly in terms of segmentation, is obtained.
3. Once the plume seems to be well isolated from the background in the last frame, a rectangular region of interest (ROI) is drawn automatically around the plume to create a mask containing the supposed plume area. If the ROI does not correspond to or incorporate the plume well (e.g., because clouds are recognized as the bigger object), it can be drawn manually (Figure A5; zooming in is highly recommended) simply by responding to the appropriate dialog box. Then, the next dialog box asks if the user is satisfied with the drawn ROI or wants to draw it again.

At this point, the user is asked to save or just see the processed frames (the latter is to speed up the process if there is still an uncertainty regarding the appropriateness of the set thresholds). Finally, the plume tracking algorithm can be applied to all frames through the automatic image analysis procedure (*image_analysis.m*), described in Section 3.1. Moreover, a .gif file with four panels showing the processing of each frame (Figure A6a) is also automatically saved in the specific folder.

The following step is to use the geometrical calibration parameters to obtain two vectors for height (a 1-by-N_v vector) and width (an N_h-by-1 vector) of each pixel of the images. This is done by applying the equations presented in Section 3.2 (*geometrical_calibration.m*).

Finally, PlumeTraP can calculate the plume parameters explained in Section 3.3. The function *plume_parameters.m* calls different subfunctions that directly calculate these parameters (i.e., *plumeheight.m*, *plumewidth.m*, *plumevelocity.m* and *plumeacceleration.m*) and also produces a figure that is updated for each frame during the analysis and saved in a .gif file (Figure A6b). Moreover, another subfunction is called to save the calculated parameters in .csv files and in a .png plot. The first saved .csv file (Figure A6c) collects all the parameters and their related errors that can be expressed as functions of time (named, e.g., *Explosion_1_parameters.csv*), while the other two contain the width of the plume at each pixel level for each frame as a function of the height of the plume and the associated errors (named, e.g., *Explosion_1_heightwidth.csv* and *Explosion_1_heightwidth_err.csv*, respectively). These tables can also be found in the MATLAB Workspace (named *tables*).

Finally, if more than one video was selected for analysis, the above process is repeated until all videos have been treated.

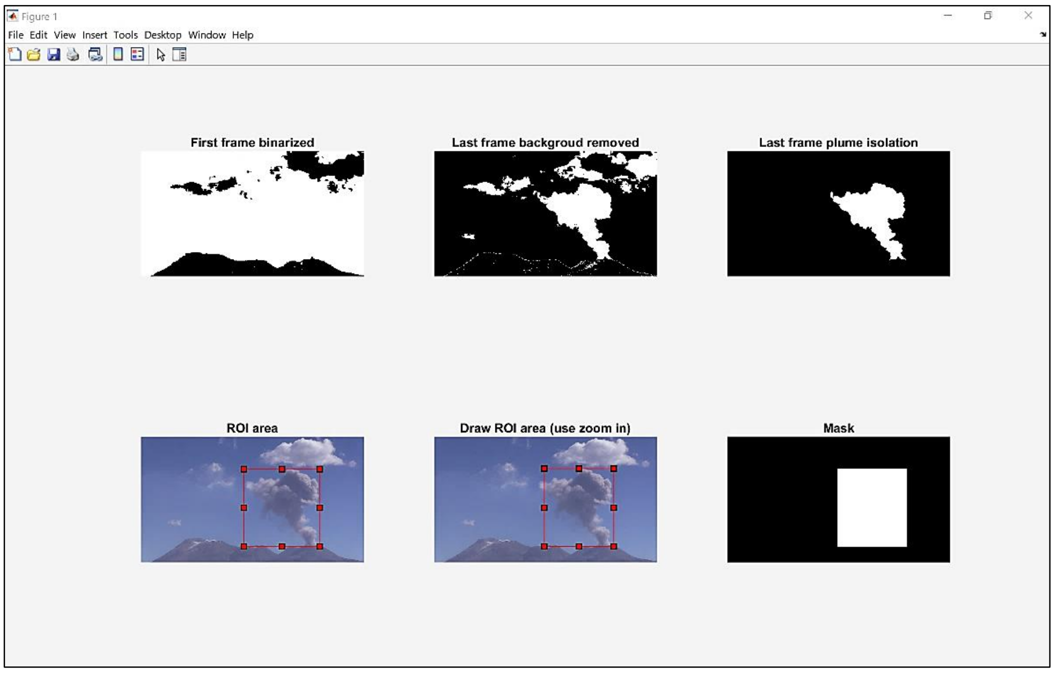
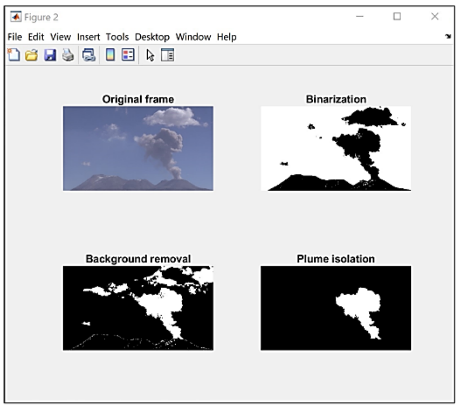


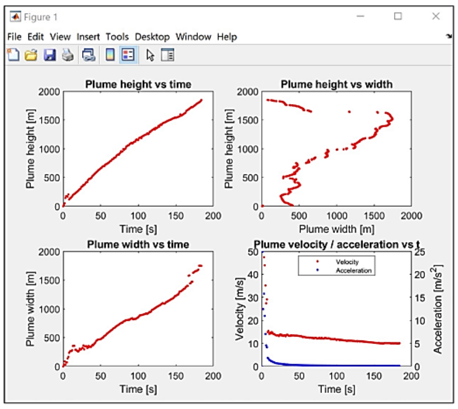
Figure 1
File Edit View Insert Tools Desktop Window Help
First frame binarized
Last frame backgroud removed
Last frame plume isolation
ROI area
Draw ROI area (use zoom in)
Mask

Figure A5. MATLAB figure from PlumeTraP showing the pre-processing procedure starting from a video of the 8th of August 2018 (13:54 UTC-5) explosion at Sabancaya. The upper three panels are shown when setting the thresholds and later updated if they are changed by the user. The lower panels are used for setting, either automatically or manually, the region of interest (ROI).

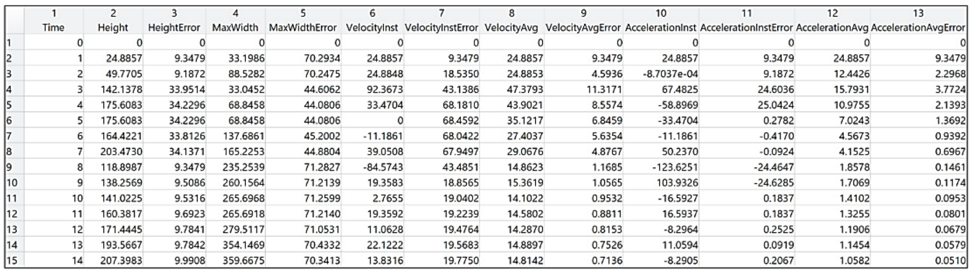
(a)



(b)



(c)



	1	2	3	4	5	6	7	8	9	10	11	12	13
	Time	Height	HeightError	MaxWidth	MaxWidthError	VelocityInst	VelocityInstError	VelocityAvg	VelocityAvgError	AccelerationInst	AccelerationInstError	AccelerationAvg	AccelerationAvgError
1	0	0	0	0	0	0	0	0	0	0	0	0	0
2	1	24.8857	9.3479	33.1986	70.2934	24.8857	9.3479	24.8857	9.3479	24.8857	9.3479	24.8857	9.3479
3	2	49.7705	9.1872	88.5282	70.2475	24.8848	18.5350	24.8853	4.5936	-8.7037e-04	9.1872	12.4426	2.2968
4	3	142.1378	33.9514	33.0452	44.6062	92.3673	43.1386	47.3793	11.3171	67.4825	24.6036	15.7931	3.7724
5	4	175.6083	34.2296	68.8458	44.0806	33.4704	68.1810	43.9021	8.5574	-58.8969	25.0424	10.9755	2.1393
6	5	175.6083	34.2296	68.8458	44.0806	0	68.4592	35.1217	6.8459	-33.4704	0.2782	7.0243	1.3692
7	6	164.4221	33.8126	137.6861	45.2002	-11.1861	68.0422	27.4037	5.6354	-11.1861	-0.4170	4.5673	0.9392
8	7	203.4730	34.1371	165.2253	44.8804	39.0508	67.9497	29.0676	4.8767	50.2370	-0.0924	4.1525	0.6967
9	8	118.8987	9.3479	235.2539	71.2827	-84.5743	43.4851	14.8623	1.1685	-123.6251	-24.4647	1.8578	0.1461
10	9	138.2569	9.5086	260.1564	71.2139	19.3583	18.8565	15.3619	1.0565	103.9326	-24.6285	1.7069	0.1174
11	10	141.0225	9.5316	265.6968	71.2599	2.7655	19.0402	14.1022	0.9532	-16.5927	0.1837	1.4102	0.0953
12	11	160.3817	9.6923	265.6918	71.2140	19.3592	19.2239	14.5802	0.8811	16.5937	0.1837	1.3255	0.0801
13	12	171.4445	9.7841	279.5117	71.0531	11.0628	19.4764	14.2870	0.8153	-8.2964	0.2525	1.1906	0.0679
14	13	193.5667	9.7842	354.1469	70.4332	22.1222	19.5683	14.8897	0.7526	11.0594	0.0919	1.1454	0.0579
15	14	207.3983	9.9908	359.6675	70.3413	13.8316	19.7750	14.8142	0.7136	-8.2905	0.2067	1.0582	0.0510

Figure A6. Examples of the MATLAB graphical outputs of PlumeTraP, applied to a video of the 8th of August 2018 (13:54 UTC-5) explosion at Sabancaya. (**a**) The PlumeTraP graphical interface showing the effectiveness of the image analysis through the original, binarized, background-removed and filtered and resulting plume-isolated images representing a single saved frame. (**b**) Four plots showing the calculated parameters: the top left panel is for the height of the plume as a function of time, the top right the width of the plume for each frame as a function of height, the bottom left the maximum width as a function of time and the last one the vertical rise velocity and the acceleration of the plume as functions of time. (**c**) Part of the MATLAB table that is saved in the .csv file listing the main plume parameters.