remote sensing

MDPI

Article

PlumeTraP: A New MATLAB-Based Algorithm to Detect and Parametrize Volcanic Plumes from Visible-Wavelength Images

Riccardo Simionato [1,*], Paul A. Jarvis [2,3], Eduardo Rossi [3] and Costanza Bonadonna [3]

[1] Dipartimento di Geoscienze, Facoltà di Scienze, Università degli Studi di Padova, 35131 Padova, Italy
[2] GNS Science, Wairakei Research Centre, Taupo 3352, New Zealand; p.jarvis@gns.cri.nz
[3] Département de Sciences de la Terre, Faculté de Sciences, Université de Genève, 1205 Geneva, Switzerland; eduardo.rossi@unige.ch (E.R.); costanza.bonadonna@unige.ch (C.B.)
* Correspondence: riccardo.simionato.3@studenti.unipd.it

Abstract: Tephra plumes from explosive volcanic eruptions can be hazardous for the lives and livelihoods of people living in the proximity of volcanoes. Monitoring and forecasting tephra plumes play essential roles in the detection, characterization and hazard assessment of explosive volcanic events. However, advanced monitoring instruments, e.g., thermal cameras, can be expensive and are not always available in monitoring networks. Conversely, visible-wavelength cameras are significantly cheaper and much more widely available. This paper proposes an innovative approach to the detection and parametrization of tephra plumes, utilizing videos recorded in the visible wavelengths. Specifically, we have developed an algorithm with the objectives of: (i) identifying and isolating plume-containing pixels through image processing techniques; (ii) extracting the main geometrical parameters of the eruptive column, such as the height and width, as functions of time; and (iii) determining quantitative information related to the plume motion (e.g., the rise velocity and acceleration) using the physical quantities obtained through the first-order analysis. The resulting MATLAB-based software, named Plume Tracking and Parametrization (PlumeTraP), semi-automatically tracks the plume and is also capable of automatically calculating the associated geometric parameters. Through application of the algorithm to the case study of Vulcanian explosions from Sabancaya volcano (Peru), we verify that the eruptive column boundaries are well recognized, and that the calculated parameters are reliable. The developed software can be of significant use to the wider volcanological community, enabling research into the dynamics of explosive volcanic eruptions, as well as potentially improving the use of visible-wavelength cameras as part of the monitoring networks of active volcanoes. Furthermore, PlumeTraP could potentially find a broader application for the analysis of any other plume-shaped natural or anthropogenic phenomena in visible wavelengths.

Keywords: volcanic plumes; image analysis; visible wavelength; monitoring; tephra; ash

check for updates

Citation: Simionato, R.; Jarvis, P.A.; Rossi, E.; Bonadonna, C. PlumeTraP: A New MATLAB-Based Algorithm to Detect and Parametrize Volcanic Plumes from Visible-Wavelength Images. *Remote Sens.* **2022**, *14*, 1766. https://doi.org/10.3390/rs14071766

Academic Editor: Christian Bignami

Received: 27 January 2022
Accepted: 1 April 2022
Published: 6 April 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Volcanic plumes are mixtures of volcanic particles (i.e., tephra), gases and entrained air and are associated with a variety of explosive eruptions. Large particles (>2 mm, i.e., lapilli and blocks/bombs) can be part of this mixture in the early stages of plume development while, later, only ash-sized particles (<2 mm) are involved. The material is injected into the atmosphere, rises and expands in a turbulent flow before, potentially, being dispersed at multiple scales [1]. This can result in tephra dispersal in the atmosphere and tephra fallout on the ground, both of which can have severe consequences for everyday life, e.g., air traffic, buildings, infrastructure, agriculture [2–4], and for people's health [5]. Consequently, the dispersal of volcanic particles is one of the most widespread hazards posed by active volcanoes. Ensuring an accurate assessment of the hazard posed by volcanic plumes is, therefore, an essential step in reducing impacts on communities and ecosystems [6].

In order to provide a comprehensive assessment of volcanic hazards, it is important to characterize past activity of active volcanoes and carry out probabilistic modeling (e.g., [7]). Such information can also improve the accuracy and precision of forecasting activities (e.g., [8]). Nowadays, volcanic activity is monitored using both ground- and space-based instruments with different spectral coverage, accuracy and global measurement density that allow volcanologists to record and characterize explosive eruptions as well as to support modeling of volcanic processes (e.g., [9–14]).

Nevertheless, the use of video recordings of volcanic eruptions to make scientific measurements is not widespread [15] and the majority of such studies reported in the literature, particularly in recent years, are based on thermal infrared (TIR) imagery [15–18]. Indeed, there are relatively few studies in which visible-wavelength cameras have been used [19–22]. Despite this, visible-wavelength cameras are frequently installed as monitoring equipment on volcanoes [12,23,24], although the obtained data are often only qualitative or semi-quantitative. On the other hand, regardless of their high potential for volcano monitoring and the rapid development of their technology, infrared cameras remain relatively niche products [25]. This is due to the extremely prohibitive costs (tens of thousands of USD) and the low resolution (rarely greater than 640 $\times$ 512 pixels) of infrared cameras compared to high-resolution visible-wavelength cameras [25]. However, the development of an automated method for the detection of emitted plumes and quantification of plume properties from visible-wavelength images, without the need for manual tracking performed by the user, remains lacking. Although some methods designed for use with infrared imagery do exist [15,18], they cannot be effectively applied to videos recorded in the visible light. This is because visible-wavelength videos are often affected by variable lighting conditions and poor contrast between the plume and the background. As opposed to thermal images, images in the visible wavelengths are composed of three channels (red, green and blue), and the contrast between the tephra plume and the rest of the image is highly dependent on the meteorological conditions. Therefore, visible image analysis of volcanic plumes has almost always been overlooked because of the difficulty in cleanly tracking the plume, especially in adverse meteorological conditions. In fact, a critical part of the image processing is to automatically distinguish pixels of the image corresponding to the plume from pixels corresponding to the sky, the background and the landscape. As such, the use of visible-wavelength analysis in the study of volcanic plumes remains limited.

Motivated by the need for an automated method for tracking volcanic plumes using visible-wavelength imagery, this work presents the development of **Plume Tra**cking and **P**arametrization (PlumeTraP), a MATLAB-based software able to detect, track and parametrize plume-shaped objects through the analysis of visible videos. This is achieved by developing segmented masks for visible-wavelength images of eruption plumes that exclude noise and provide a good outline of the plume boundaries. The developed algorithm has been tested on tephra plumes recorded in different daylight and atmospheric weather conditions, with satisfactory results up to a certain level of cloudiness, which represents the primary limiting factor of the present work. However, this limitation is intrinsic to the use of visible-light sensors and cannot be removed, but at best mitigated. This project is pioneering because PlumeTraP is based on a quasi-automatic image processing technique which outputs the basilar geometric and time-derivative plume parameters that represent the starting point for modeling a specific event. Moreover, with targeted adjustments, PlumeTraP could be adapted in the future for near real-time eruption monitoring.

In the rest of this paper, we first summarize the dataset used to test the methodology (Section 2), before going on to describe the structure of the software and image analysis methodology (Section 3). We then present results of the image analysis, i.e., plume isolation, and calculations of physical parameters, e.g., heights and rise velocities (Section 4). We finally discuss these results whilst comparing with other works (Section 5) before outlining our conclusions (Section 6). A detailed tutorial about how to set up and use PlumeTraP is presented in Appendix A.

2. Field Site

The dataset used to test the software was collected during a field campaign in July and August 2018 at Sabancaya volcano (Figure 1), Peru, and consists of 49 high-resolution visible-wavelength videos, each recording one explosion. The videos were captured using a Canon Legria HF G40 at a frame rate of 50 fps and a frame size of 1920 × 1080 px^2. Sabancaya is a Holocene-aged intermediate-composition stratovolcano [26] belonging to the Ampato–Sabancaya Volcanic Complex in Southern Peru. During the last few thousand years, it has been characterized by low-to-moderate explosive activity with a volcanic explosivity index (VEI) of 1–2 [26,27]. After a two-century-long quiescence, its recent activity is characterized by intermittent periods of explosive activity lasting years (e.g., 1990–1998 and 2016–present day). Eruptive periods consist of alternating Vulcanian and phreatomagmatic to phreatic events [26–29]. The field campaign of 2018 was conducted during the current eruptive cycle (2016–ongoing), characterized by multiple Vulcanian explosions per day, with maximum tephra plume heights varying between 1 and 3 km above the crater [30–32].

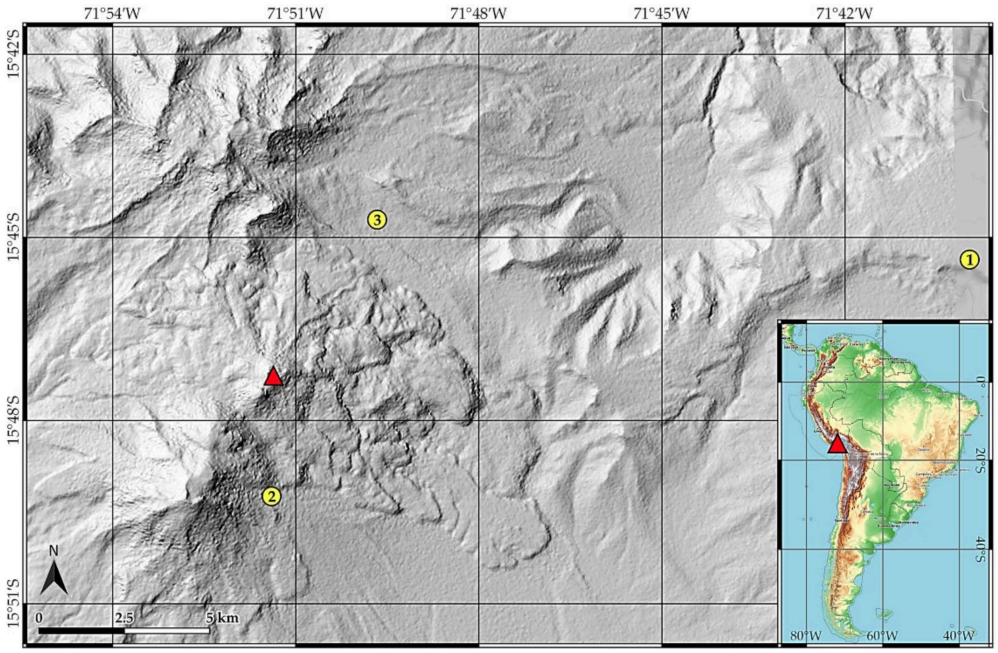


Figure 1. Terrain map of the area around Sabancaya, showing the position of the vent (red triangle) and locations of the recording sites (yellow dots). The explosions presented in this work were recorded at sites 1 (8 August 2018, at 13:54 UTC-5 and at 11:26 UTC-5), 2 (6 August 2018, at 14:20 UTC-5) and 3 (30 July 2018, at 09:49 UTC-5).

3. Methods: Structure of PlumeTraP Software

The primary goal of this work was to develop a new algorithm to process and analyze each frame extracted from a video recorded in the field of visible light, with the specific objective of detecting and tracking the plume through time. PlumeTraP, the result of this work, was written in MATLAB version R2018b and uses the Image Processing Toolbox 10.3. However, the software has also been successfully tested with the R2020a and R2021b releases (using the Image Processing Toolbox 11.4). A detailed guide describing how to set up and work with PlumeTraP is presented in Appendix A and we have uploaded the MATLAB scripts that can be found in the Supplementary Materials to the repository https://doi.org/10.5281/zenodo.6406009 (accessed on 26 June 2021), where future releases will also be added. In the rest of this section, we present a description of the workflow of PlumeTraP.

3.1. Video Analysis

Once the user has selected the video (or videos) to be analyzed, the first operation performed by PlumeTraP is to automatically save the first frame of the video and then a sequence of frames at a frequency determined by the user. This enables the user to produce output data at a temporal resolution sufficient for their purpose, up to a maximum of the frame rate of the original recording. The frequency is set through the scale factor n, that ranges from 1 up to the frame rate of the original recording and may be a non-integer (e.g., if $n = 1$, a 50 fps video results in a frame saved every 50 frames). The result is a set of images that corresponds to the original video resampled with a frame rate of n fps. Whilst a resampled frame rate of 1 fps is sufficient to obtain satisfactory data for the bulk properties, i.e., height and width, of the volcanic plumes studied here, there may be other processes which happen on shorter timescales, e.g., turbulence.

Typically, the identification of plume pixels is achieved through thresholding and thus relies on a strong difference in pixel intensity between the plume and its surroundings [15]. Thus, an effectively segmented output requires a more elaborate strategy than for thermal imagery, in addition to multiple image processing methods, to reconstruct the plume shape. We first explored multiple image analysis methods for detecting plume-containing pixels using ImageJ [33] and, once a reliable strategy was found, this was then applied to all frames of the videos in MATLAB. The PlumeTraP image analysis technique is shown in Figure 2 and consists of the following procedure:

1. To produce a segmented plume image for a given frame, three images are required (Figure 2a):
   - the image of the current frame (at time t);
   - an initial image captured just prior to eruption (at time t_0), that is intended to be a reference for the background subtraction;
   - the image of the previous frame (at time $t - n^{-1}$).
2. The RGB multichannel images are split into their red, green and blue channels, obtaining three images in which each pixel has a specific value of brightness intensity between 0 and 255.
3. For each image, the red channel is subtracted from the blue channel as, during testing and development, we found that this commonly increases the contrast between the plume and the background compared to other possible operations. The result is an image where the plume and the landscape are highlighted with respect of the rest of the image (Figure 2b).
4. A segmentation process is then applied to create binary images using a user-defined threshold pixel intensity value (Figure 2c). This threshold luminance value is determined during a pre-processing step and iterated until a satisfactory segmented output is obtained. Suggested values and additional information about the pre-processing can be found in Appendix A (Appendix A.3).
5. Two partial masks are created by exploiting the difference in intensity between the images (Figure 2d):
   - the first one consists of the modulus of the subtraction of the segmented first frame from the segmented current frame, resulting in a binary image that shows the plume without the landscape. However, some meteorological clouds (if they are moving) and some noise may remain. In the case of the first frame, the subtraction gives an empty image;
   - the second results from the modulus of the difference between the segmented current frame and the segmented previous frame (this is not applied to the first frame, as a preceding image does not exist). Thus, it highlights the local movement of the plume (and the meteorological clouds) between frames and is fundamental to creating a mask that fits the original plume extension [15].

6. The two masks are then summed (Figure 2e), which goes a long way towards enabling identification of the presence and evolution of the plume. However, it is still not fully isolated from the rest of the image.
7. A two-dimensional median filter is applied, using a window of 4×4 pixels, to remove noise (Figure 2f).
8. A mask is then applied such that all the pixels outside a region of interest (ROI) containing the plume are set equal to zero. This ROI is selected as the minimum rectangular area that captures the plume on the last frame and can be drawn automatically or manually during the pre-processing step (see Appendix A.3). This mask is essential for ensuring that the plume is selected rather than other similar objects (e.g., atmospheric clouds, degassing clouds or the landscape if its intensity and contrast are similar to those of the plume) (Figure 2g).
9. All objects (white regions) apart from the largest are removed, thus removing noise and objects not connected with the plume (Figure 2h).
10. Finally, holes in the remaining object are filled, leading to the final image with the extracted plume shape (Figure 2i).

This processing is applied to all the images previously saved to obtain images of the plume shape and its evolution through time.

3.2. Geometrical Calibration

Following the image processing, it is then necessary to apply a geometric calibration to convert pixel locations into physical coordinates (from pixel units to metric units). Here, we follow a method modified from Bombrun et al. [15]. The vertical position z_i of a single pixel within the image, measured with respect to the altitude of the camera, is related to the horizontal distance Y of the camera from the image plane containing the plume and to the geometric parameters of the camera (Figure 3) through

$$z_j = \frac{Y}{2}\left(\tan\left(\phi - \frac{\beta_v}{2} + (j-1)\frac{\beta_v}{N_v}\right) + \tan\left(\phi - \frac{\beta_v}{2} + j\frac{\beta_v}{N_v}\right)\right), \quad (1)$$

where ϕ is the camera inclination with respect to the horizontal, β_v the vertical field of view (FOV), j the vertical position (measured in pixels from the bottom of the image) of the pixel whose height is being measured and N_v the number of pixels in the vertical direction. Thus, β_v / N_v is the vertical angle subtended by a single pixel.

Furthermore, the geometric calibration produces a systematic error associated with the spatial resolution of the image, that must be taken into consideration, and is automatically calculated by the algorithm. The vertical extent of a pixel is calculated as

$$\Delta z_j = Y\left(\tan\left(\phi - \frac{\beta_v}{2} + j\frac{\beta_v}{N_v}\right) - \tan\left(\phi - \frac{\beta_v}{2} + (j-1)\frac{\beta_v}{N_v}\right)\right), \quad (2)$$

and, therefore, the associated uncertainty z_j^{err} as

$$z_j^{err} = \frac{Y}{2}\left(\tan\left(\phi - \frac{\beta_v}{2} + j\frac{\beta_v}{N_v}\right) - \tan\left(\phi - \frac{\beta_v}{2} + (j-1)\frac{\beta_v}{N_v}\right)\right). \quad (3)$$

In this way, z_j^{err} is defined as half the vertical pixel extent. Thus, the farther a pixel is from the center of the image, the greater the pixel's vertical extent and, consequently, the greater the uncertainty on z_j. There may well be other sources of uncertainty due to imprecise knowledge of the camera properties, e.g., location, inclination and orientation. However, as will be discussed in Appendix B, the uncertainty due to these will depend on how the visible data are collected and will be different for each user.

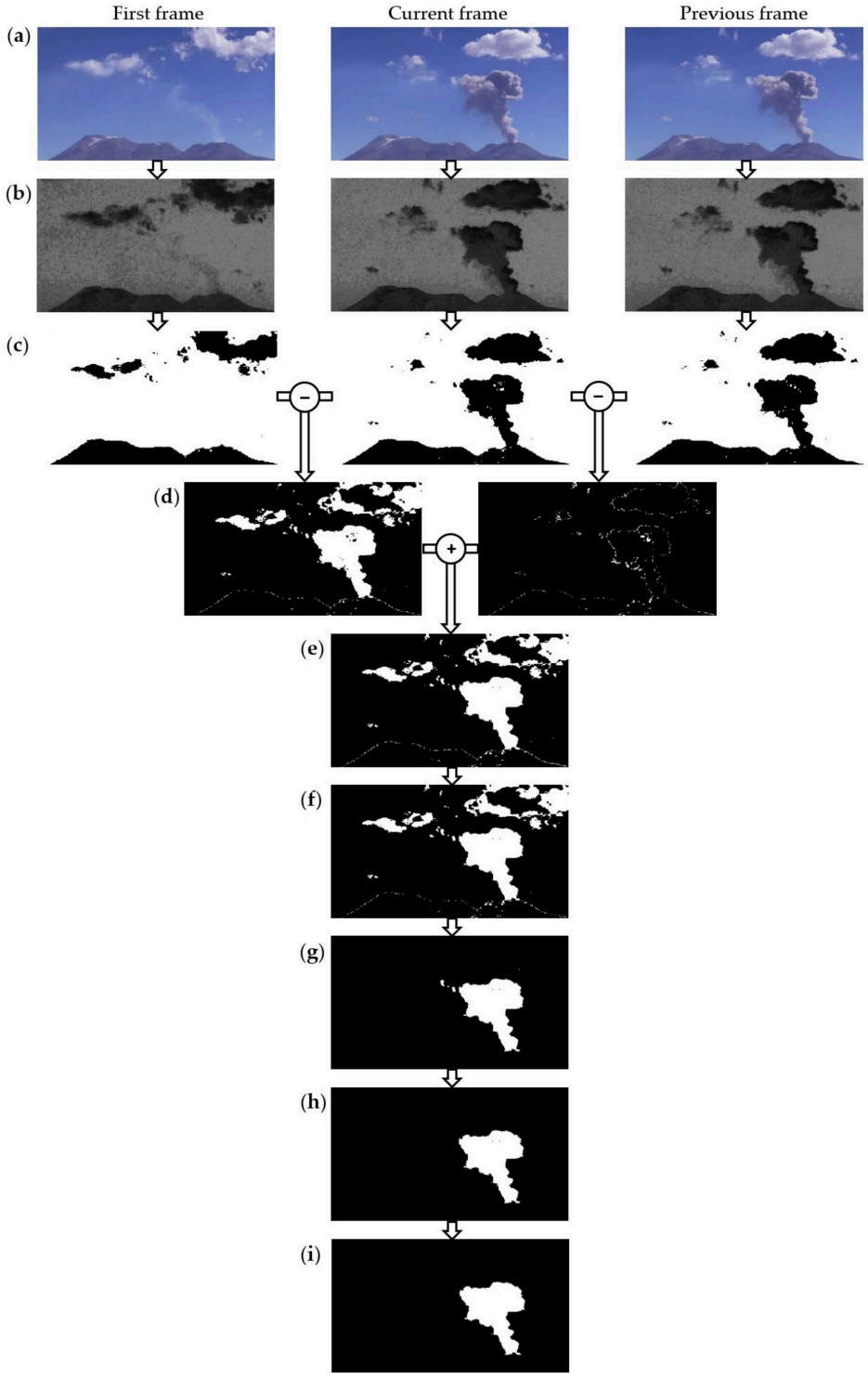


Figure 2. Flow diagram showing PlumeTraP image processing of a frame from a video recorded at Sabancaya (8 August 2018, at 13:54 UTC-5). For each frame in the sequence, the processing relies on an initial frame, captured just prior to the explosion, and the previous frame in the sequence. The different steps in the processing can be illustrated by: (**a**) original images; (**b**) blue–red subtraction; (**c**) segmentation; (**d**) subtractions; (**e**) sum of masks; (**f**) median filter; (**g**) region of interest (ROI) selection; (**h**) biggest object identification; and (**i**) filling holes.

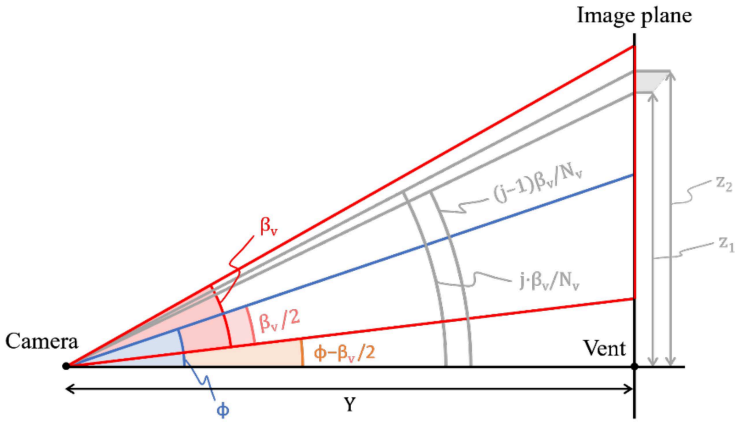


Figure 3. Sketch showing the geometrical calibration (sideways view). Red lines are the image limits, the blue one is the line to the center of the image and the gray shaded region represents the extent of a single pixel (exaggerated). Y is the camera–image plane distance, ϕ the camera inclination, β_v the vertical field of view (FOV), j the vertical position of the selected pixel in the image and N_v the number of pixels in the vertical direction. The z_1 and z_2 heights represent the minimum and maximum height of pixel j. Figure modified from Bombrun et al. [15].

With regard to the horizontal component of the geometric calibration, Bombrun et al. [15] assumed the horizontal pixel extent Δx_i to be approximately constant across the image. We choose to avoid this assumption by following a similar procedure to that employed for the vertical component. Thus, the horizontal position x_i of a pixel, with respect to the leftmost pixel of the image (Figure 4), is given by

$$x_i = Y \tan\left(\frac{\beta_h}{2}\right) - \frac{Y}{2}\left(\tan\left(\frac{\beta_h}{2} - (i-1)\frac{\beta_h}{N_h}\right) + \tan\left(\frac{\beta_h}{2} - i\frac{\beta_h}{N_h}\right)\right), \tag{4}$$

where β_h is the horizontal FOV and N_h the number of pixels in the horizontal direction. Then, the real horizontal extent Δx_i of a pixel can be defined as

$$-\Delta x_i = Y\left(\tan\left(\frac{\beta_h}{2} - i\frac{\beta_h}{N_h}\right) - \tan\left(\frac{\beta_h}{2} - (i-1)\frac{\beta_h}{N_h}\right)\right). \tag{5}$$

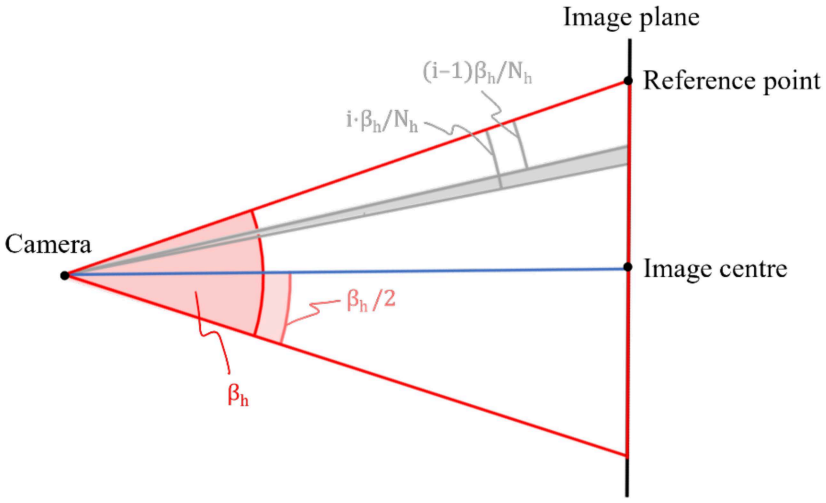


Figure 4. Sketch showing the geometrical calibration (plan view). Red and blue lines are as in Figure 3. The gray shaded region represents the extent of a single pixel (exaggerated). Y is the camera–image plane distance, β_h the horizontal field of view (FOV), i the horizontal position of the selected pixel in the image and N_h the number of pixels in the vertical direction.

We also define the horizontal systematic error associated with the spatial resolution of the image x_i^{err} as half the horizontal pixel extent, that is

$$x_i^{err} = \frac{Y}{2}\left(\tan\left(\frac{\beta_h}{2} - i\frac{\beta_h}{N_h}\right) - \tan\left(\frac{\beta_h}{2} - (i-1)\frac{\beta_h}{N_h}\right)\right). \quad (6)$$

Furthermore, in the software, the user can enter both maximum and minimum possible values of Y to allow for uncertainty in the camera-to-image plane distance. Thus, the software calculates the pixel positions for both the maximum and minimum values of Y, before outputting the average of these values (with the superscript "mean") and the associated errors as

$$z_j = z_j^{mean} \pm z_j^{err}, \quad (7)$$

and

$$x_i = x_i^{mean} \pm x_i^{err}. \quad (8)$$

To summarize, Equations (1)–(8) are applied to each pixel by PlumeTraP to obtain the physical coordinates of each pixel and the associated error. However, if available, pixel positions can also be manually uploaded by the user (see Appendix A.1).

3.3. Calculation of Parameters

Once the geometric correction is performed, it is possible to use the obtained pixel heights, measured with respect to the camera altitude, and horizontal positions to define the geometric parameters of the plume. The first-order analysis consists of calculating the main dimensions of the identified plume, i.e., height and width (as a function of height), as functions of time. A second-order analysis is then performed to retrieve the ascent velocity and the acceleration of the plume head in the atmosphere.

The plume height is defined as the uppermost pixel of the plume mask (Figure 5). The algorithm then subtracts the height of the vent from the calibrated highest value to then obtain a relative height with respect to the vent

$$h_f = h_{max_f} - h_{vent}, \quad (9)$$

where h_{max_f} is the maximum height of the plume in frame f and h_{vent} is the height of the vent.

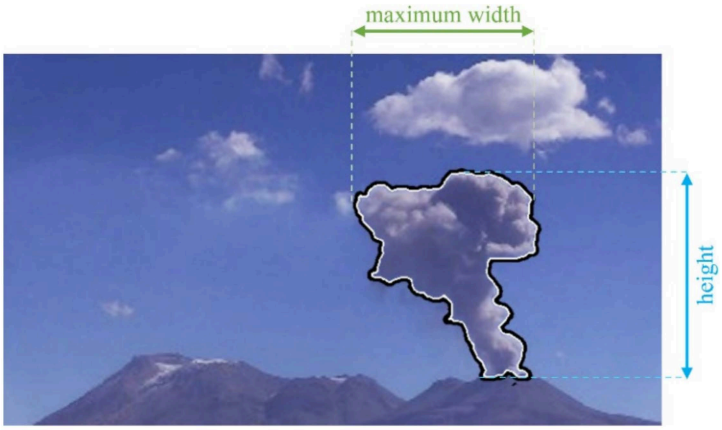


Figure 5. Figure showing how the geometrical parameters of the plumes are calculated, for the example of the explosion shown in Figure 2. Solid black line shows the outline of the plume mask.

The width of the plume is firstly calculated at each pixel level, before the maximum width is identified, to enable quantification of plume spreading during ascent (Figure 5).

Calculation of width is simply done by subtracting, for each row, the horizontal position of the leftmost plume pixel from that of the rightmost plume pixel

$$w_{r(f)} = x_r^{right} - x_r^{left}, \tag{10}$$

where $w_{r(f)}$ is the width calculated in row r of a certain frame f, and x_r^{right} and x_r^{left} are the horizontal distances of the right and left plume margins, respectively, measured from the leftmost image pixel. Then, the maximum value was selected for each frame such that the maximum width as a function of time can be identified. Note that the outputted plume width is twice the radius.

The change in plume height between two frames can be converted to vertical ascent velocity, calculated along the vertical axis, by using the time elapsed between them. The algorithm calculates both an instantaneous rise velocity v_{inst} and a time-averaged rise velocity v_{avg}. The first is the difference in plume height between two consecutive frames ΔH divided by the time difference between the two frames Δt. The latter is obtained by dividing the plume height reached in the current frame H by the time elapsed since the eruption onset t. These two quantities are expressed, respectively, as

$$v_{inst} = \Delta H / \Delta t, \tag{11}$$

and

$$v_{avg} = H / t. \tag{12}$$

In the same way, the instantaneous and time-averaged accelerations of the plume top are determined from the vertical ascent velocity. Thus, the instantaneous acceleration a_{inst} is calculated from v_{inst} divided by Δt. On the other hand, the time-averaged acceleration a_{avg} is v_{avg} divided by t. These result in

$$a_{inst} = v_{inst} / \Delta t, \tag{13}$$

and

$$a_{avg} = v_{avg} / t. \tag{14}$$

It is important to note that these measurements are associated with sources of uncertainty. The first contributing factor is intrinsic to the pixel extent, as a given pixel represents a finite area and does not represent a single point. The other is related to the uncertain position of the vent, that varies between each volcano, and even from different recording sites at the same volcano. As already mentioned, these errors are automatically calculated by PlumeTraP, making use of the finite pixel extent, and minimum and maximum estimates of the distance between the camera and image plane, respectively.

4. Results

4.1. Image Processing Techniques

Application of the created software, PlumeTraP, shows that it is capable of developing segmented masks for visible-wavelength images of eruption plumes that reproduce the plume shape with promising results (Figure 6). We have used raw videos taken at Sabancaya volcano to test the software, even though a pre-processing of videos is possible (e.g., brightness and contrast enhancement) and may help to isolate the plume better. The primary problem related to the use of video recorded in visible wavelengths is the fact that each pixel is associated with a true color resulting from the combination of the three RGB channels. Furthermore, a strong difference in pixel intensity between the plume and the background is required to obtain a good plume isolation (Figure 6a,b). Thus, the algorithm does not work as well for videos with significantly cloudy conditions (Figure 6d) as for those in which there is a higher contrast between the plume and the background (i.e., a blue sky). Sometimes, this situation requires the user to choose between a good reproduction of

the plume boundaries with some noise, and a less noisy segmented image where parts of the plume are lost.

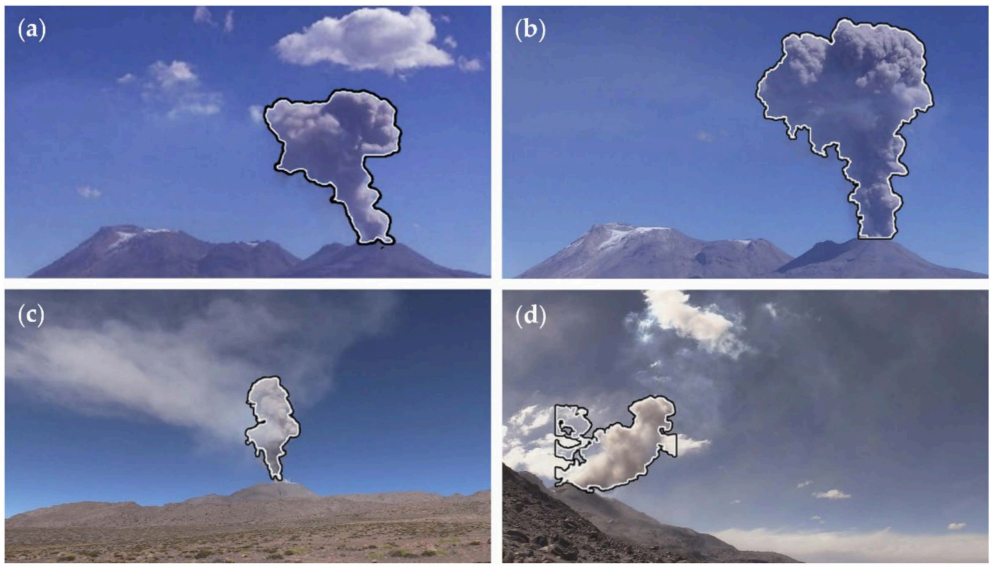


Figure 6. Superimposition of the plume shapes tracked by PlumeTraP on the original images for different volcanic plumes captured at Sabancaya volcano. (**a**) The presence of meteorological clouds does not affect the reproduced plume shape (8 August 2018, at 13:54 UTC-5). (**b**) Ideal weather conditions with tephra fallout visible and recognized by the algorithm (8 August 2018, at 11:26 UTC-5). (**c**) The presence of volcanic degassing in the background slightly affects the reproduced plume shape (30 July 2018, at 09:49 UTC-5). (**d**) An example where the output is unusable because of the light conditions and the presence of vapor clouds around the plume (6 August 2018, at 14:20 UTC-5).

However, it is also important to note that the presence of meteorological clouds does not always negatively affect the image processing since, if they are sufficiently spatially separated from the plume or there is a good contrast, they can easily be filtered from the final segmented image (Figure 6a,c). Although cloudy conditions represent a limit for the application of the developed algorithm, since meteorological clouds and volcanic degassing are common, 26 out of 49 explosions were recorded in sufficiently good conditions to obtain a mask that almost perfectly overlies the plume.

4.2. Geometrical Calibration

To assess if the geometrical calibration is accurate and to verify the code, it is useful to compare a known a priori length with that measured in the image. Here, the comparison was made between the diameter of the crater and the observed basal width of the volcanic plumes. The measured maximum diameter of Sabancaya crater is 384 m [26] which is comparable with the plumes' basal widths as calculated by the algorithm, ranging from (324 ± 5) m to (387 ± 3) m. The considered values are chosen from four videos in which the plume rises almost vertically, as well as for the larger eruptions in the dataset, to ensure that the tephra plume is filling the crater's inner area. From this agreement, we can conclude that the geometrical calibration works well.

4.3. Extracted Parameters

Once an accurate plume shape reconstruction is obtained, PlumeTraP is then able to successfully automatically parametrize the tephra plume through the application of Equations (9)–(14). As described above, the calculated basal plume widths are comparable to the crater diameter at Sabancaya, suggesting measurements of plume widths are accurate. Furthermore, the other calculated parameters extracted from the Sabancaya dataset are consistent with already-published data as well. The resulting maximum heights are

compatible with the heights of tephra plumes reported by the Instituto Geofísico del Perú (IGP) and Instituto Geológico, Minero y Metalurgico (InGeMMet) [30–32], with plumes rising up to at least 3 km above the vent (Figure 7). Additionally, the heights reached by the plumes within the first 30 s, along with the vertical rise velocities, are consistent with other observed Vulcanian eruptions around the world [21,22]. Examples of the obtained results are presented in Figure 7.

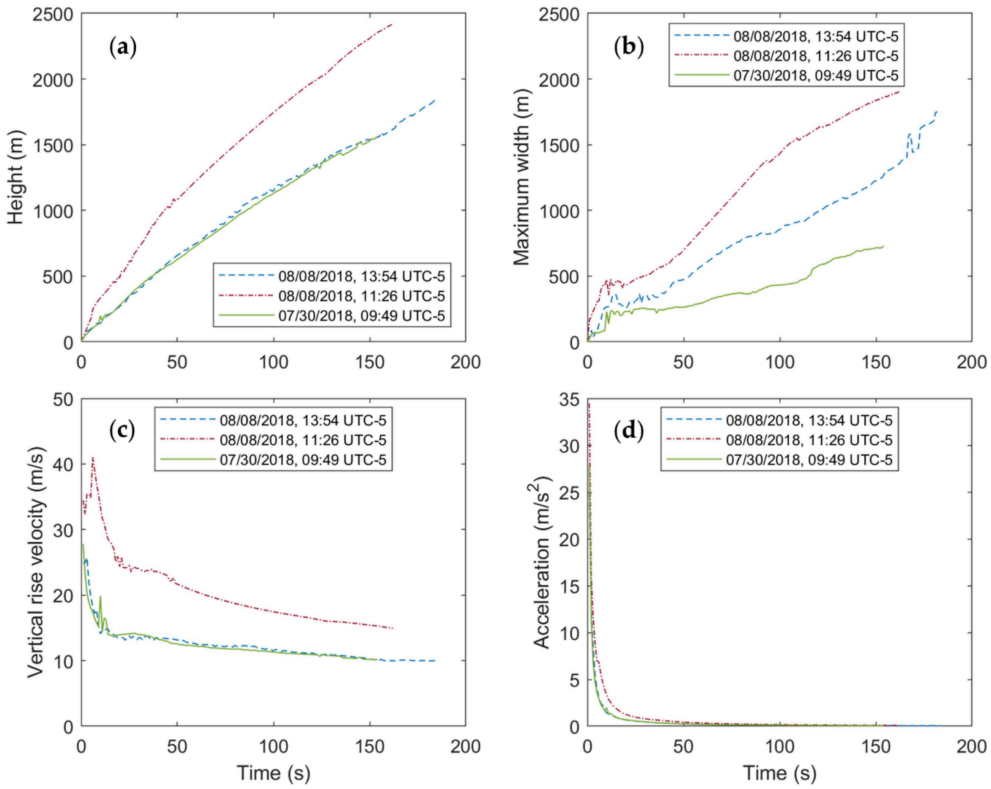


Figure 7. Plots showing the results as a function of time of the parametrization of three explosions at Sabancaya (August 8, 2018, at 13:54 UTC-5; August 8, 2018, at 11:26 UTC-5; 30 July 2018, at 09:49 UTC-5): (**a**) height of the top of the plume; (**b**) maximum width of the plume; (**c**) vertical rise velocity of the top of the plume; (**d**) acceleration of the top of the plume. Calculated errors are not shown here for clarity of the plots.

5. Discussion

As already stated, the plume is well tracked in good weather conditions, in which a strong difference in pixel intensity between the plume and the background is common. With very cloudy conditions, this difference in intensity is lower such that the output produced by PlumeTraP is reduced in quality or even unusable. Further work to improve the algorithm, such that image segmentation under bad background conditions functions better, would be a significant enhancement that could be implemented in the future, potentially through further exploration of image processing techniques. Another developable option, which may also allow fuller automation of the processing, could be to use a supervised machine learning method to select and isolate grayish plume pixels (e.g., [34]). Regardless, it is important to emphasize that the presence of atmospheric clouds does not always have a negative influence on the image processing, provided they are away from the plume and do not interact with it (e.g., Figure 6a). Concurrently, volcanic degassing can be automatically isolated if its luminance value in the image is different from that of the plume (e.g., Figure 6c).

The resulting calculated parameters for the tested Sabancaya explosions are consistent with already-published data. In particular, the heights of the plumes reach values up to

2.5 km above the vent, that are comparable with the 3 km maximum plume heights reported by the Instituto Geofísico del Perú (IGP) and Instituto Geológico, Minero y Metalurgico (InGeMMet) [30–32] for the corresponding period. Moreover, the height values, as well the vertical rise velocities, are coherent with other observed Vulcanian eruptions around the world [21,22]. Additionally, the plume widths were considered correct because of the similarity of the calculated basal plume widths to the crater diameter at Sabancaya.

The capability of PlumeTraP to identify plumes in visible-wavelength imagery was also compared to the Plume Ascent Tracker software [18] (Figure 8), an open-source MATLAB-based algorithm using graphical user interfaces (GUIs), thus resulting in a user-friendly environment for setting the video analysis parameters and choosing the outputs. However, the methodology used in Plume Ascent Tracker is optimized for the analysis of thermal-wavelength videos and, thus, its use for analyzing videos in the visible range of wavelengths presents some issues.

We have found that Plume Ascent Tracker works well at detecting a well-contrasted plume in a non-moving, clear background and uniformly lit environment [15]. In all other cases, results obtained with Plume Ascent Tracker appear to be affected by two issues. Firstly, it is not possible to select both lighter and darker parts of the plume through the thresholds used to segment the images, meaning the obtained masks frequently miss parts of the plume. Secondly, the plume shape is also reconstructed by using separated polygons, meaning that, sometimes, the recognized plume pixels do not belong to the volcanic cloud, but instead to atmospheric clouds or even to the sky. These can lead to point scattering and to consistent underestimations of the calculated physical parameters. The first of the two issues was manually corrected by smoothing the results in the presented plots (Figure 8) to show a more realistic comparison between the two pieces of software. It is clear that the thresholds that can be set through the user interface are insufficient for analysis of visible-wavelength images. In the presented examples (Figure 8), the image analysis parameters in Plume Ascent Tracker were set to obtain a plume height that was as reliable as possible. Therefore, the plots show similar plume heights and vertical rise velocities to those calculated with PlumeTraP, but the plume widths are extremely underestimated. Plume Ascent Tracker was also useful to confirm the accuracy of our geometric calibration.

A further complication concerning the calculation of the parameters that are calculated by PlumeTraP is that the current algorithm does not account for the effect of wind on plume rise. This is because wind-bent plumes will follow a trajectory determined by the wind direction, and thus may not be confined to the image plane. The direct consequence is that a weak plume may be bent towards or away from the camera recording the event and, therefore, the resulting heights and widths would be over- or underestimated, respectively. Consequently, the plume heights and widths calculated by PlumeTraP can more accurately be described as corresponding to the projection of the plume onto the image plane. Thus, care should be taken to ensure that the current version of PlumeTraP is only applied to strong volcanic plumes, i.e., those barely affected by wind. Work to include the effect of wind direction in PlumeTraP is ongoing [35].

We also tested PlumeTraP with two different computers to get information on the speed of the operations. From the presented results (Table 1), we conclude that the PlumeTraP workflow is relatively rapid even using a low-performance processor (Computer A from Table 1), with a minimum total speed for frame-dependant processes of almost 0.2 fps whilst, excluding the binarization settings, almost 17 min are required to analyze a 3 minute-duration explosion. These values become almost 0.5 fps and 6.5 min with a higher performance computer (Computer B from Table 1). These results highlight the possible use of PlumeTraP for near-real time volcanic plume analysis, whilst even faster processing speeds could potentially be attained by using higher performance processors or by automatizing the threshold luminance value selection to obtain a fully automated algorithm.

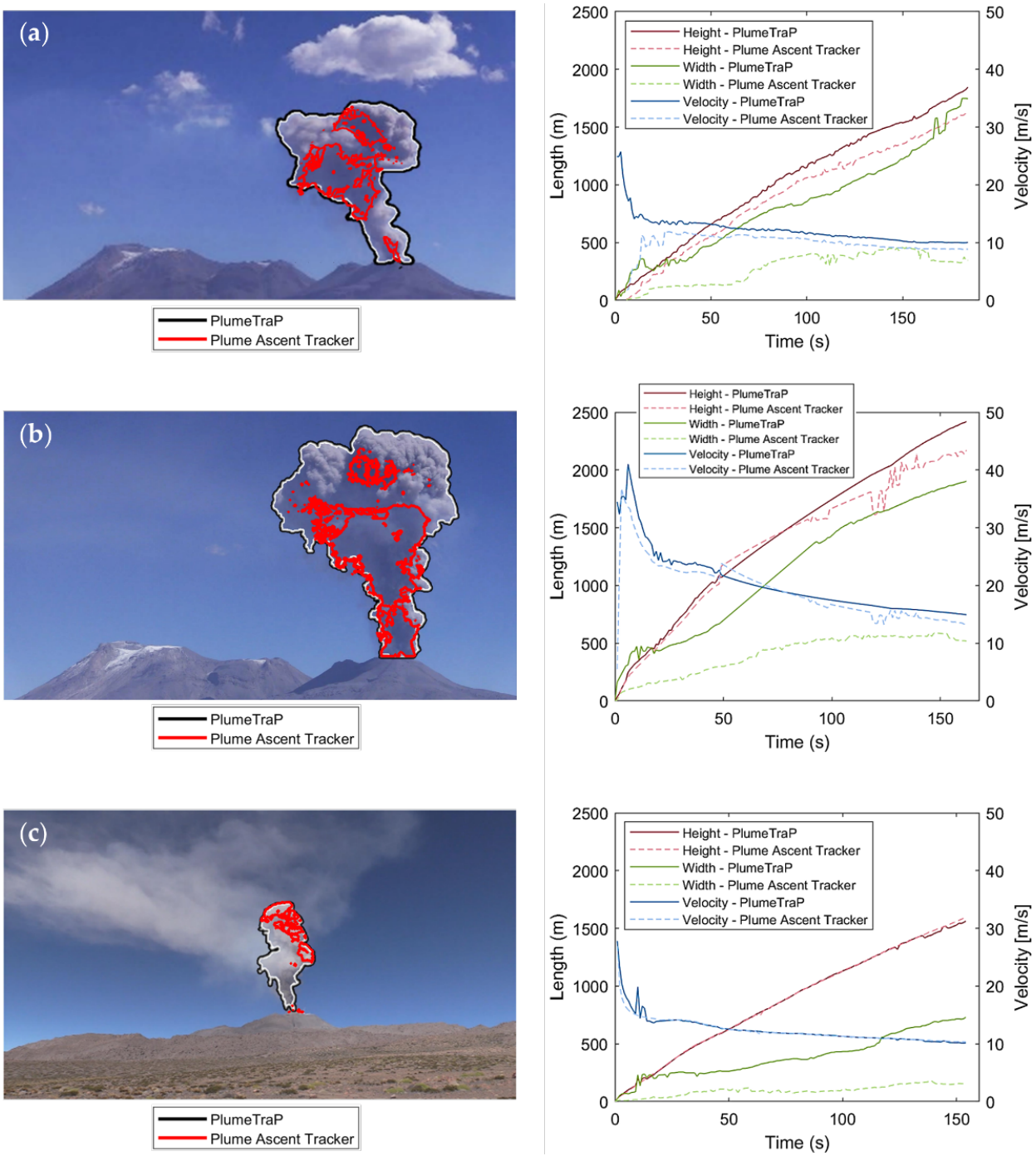


Figure 8. Comparison between plume segmentation and physical parameters obtained with Plume Ascent Tracker [18] and PlumeTraP, applied to the three explosions at Sabancaya: (**a**) 8 August 2018, at 13:54 UTC-5, (**b**) 8 August 2018, at 11:26 UTC-5 and (**c**) 30 July 2018, at 09:49 UTC-5.

Table 1. Computation time of two videos lasting 185 and 20 s, respectively, of the 8th of August 2018 (13:54 UTC-5) explosion at Sabancaya. Both videos were analyzed with two different computers, whose specifications are listed below.

	No. of Frames	Computer A [1]	Computer B [2]
Read video	185	2.482 s	69.331 s
	20	1.325 s	8.885 s
Save video frames	185	0.614 fps	1.445 fps
	20	0.473 fps	1.458 fps
Frame processing	185	0.532 fps	1.557 fps
	20	0.347 fps	1.481 fps
Geometric calibration	185	0.254 s	0.042 s
	20	0.223 s	0.057 s
Calculation of parameters	185	0.573 fps	1.340 fps
	20	0.641 fps	2.311 fps
Total (frame-dependent)	185	0.190 fps	0.481 fps
	20	0.153 fps	0.558 fps

[1] Microsoft Corporation Surface Pro 4 m3, Intel® Core™ m3-6Y30 CPU @ 0.90 GHz (4 CPUs), 4 GB of RAM, Windows 10 Pro 64-bit OS (OS build 19043). [2] ASUS Zenbook UX430UA, Intel® Core™ i5-7200U CPU @ 2.50 GHz (4 CPUs), 8 GB of RAM, Ubuntu 20.04.3 LTS 64-bit OS.

Application of PlumeTraP to nearly real-time monitoring may reveal even more dynamical information. This could potentially be done *on-line* within the software by adding new sections to the main script which apply different models (e.g., [36]) or equations (e.g., [18,21,22,37]) that could enable important parameters about volcanic plumes, such as the erupted mass [38], volume, solid fraction or temperature [22], to be inferred. Essentially, this algorithm represents an example of how the remote analysis of volcanic plumes can be an essential tool for understanding their dynamics and related hazards. Therefore, given the increasing availability and development of remote-imaging instruments [13], automated plume detection can play an important role in monitoring and research of volcanic plumes.

Although this algorithm has not been validated for the analysis of thermal videos, it is expected to work as well as for high-resolution visible videos. Indeed, since thermal imagery works in only one channel and the infrared signal of volcanic plumes is normally much stronger than background clouds, it should be easier to isolate the plume-containing pixels. The only parts of the algorithm that would not be required are the channel splitting and subsequent subtraction between channels. Moreover, potentially useful future work would be to develop a software capable of successfully analyzing both visible and infrared videos.

Although this work is mainly focused on volcanic tephra plumes, we stress that PlumeTraP could potentially be applied to any plume-shaped object, both natural (e.g., hydrothermal black smokers, solar flares, wildfires) or anthropogenic (e.g., laboratory experiments, smoke plumes emitted by factory chimneys or by fires).

When applying PlumeTraP, some caveats should be considered. In particular, it is difficult to apply PlumeTraP in cases of:

- poor contrast between the plume-shaped object and the background;
- dark light conditions and/or cloudy background;
- the presence of meteorological clouds inside the ROI;
- abundant gas emissions from the vent that form dense degassing clouds in the image background (only for the case of volcanic tephra plumes).

We would like to stress that these caveats are a direct consequence of the intrinsic limitations of visible-wavelength cameras. Therefore, they are inevitably reflected in remote sensing tools exploiting this type of sensor, which is the case for PlumeTraP.

6. Conclusions

We have developed a MATLAB-based software, PlumeTraP, which allows, through the analysis of high-resolution visible-wavelength videos, parameterization of the evolution of

volcanic plumes. We have tested and demonstrated the use of the software by analyzing Vulcanian-style eruptive plumes at Sabancaya volcano, Peru, to obtain results in terms of their morphology and rise velocity. The key image analysis algorithm, developed to identify plume-containing pixels, is new and could be of significant use to the wider volcanological community, mainly for research purposes, but in the future, with adequate and targeted adjustments, also for volcano monitoring.

Despite the lower costs and the generally higher resolution of the final output of visible-wavelength cameras [25] compared to their infrared counterparts, as well as the rarer availability of infrared cameras in the monitoring networks of volcano observatories, quantitative analysis of visible images of volcanic plumes has almost always been overlooked, because of the difficulty in obtaining an accurate plume tracking. Our study confirms that this type of image processing is possible, especially if applied in good weather conditions, and could be potentially useful for both volcano research and monitoring.

Supplementary Materials: The following supporting information can be downloaded at: https://www.mdpi.com/article/10.3390/rs14071766/s1, Video S1: 8 August 2018, 13:54 UTC-5; Video S2: 8 August 2018, 11:26 UTC-5; Video S3: 30 July 2018, 09:49 UTC-5.

Author Contributions: R.S. drafted the paper; P.A.J., E.R. and C.B. edited and revised the paper; P.A.J., E.R. and C.B. collected videos at Sabancaya volcano; R.S. developed PlumeTraP as part of a Swiss European Mobility Programme (SEMP) exchange between the University of Padova and the University of Geneva; P.A.J. supervised and gave advice for the development of PlumeTraP. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Swiss National Science Foundation #200021_169463 and by the Swiss European Mobility Programme (SEMP).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are obtained from visible-wavelength videos that are resampled at 1fps and openly available as Supplementary Materials. The PlumeTraP software and future releases (including a planned new version with the wind correction) will be available on the website github.com/r-simionato/PlumeTraP (DOI: 10.5281/zenodo.6406009).

Acknowledgments: The authors thank Rigoberto Aguilar and Nelida Manrique Llerena of INGEMMET as well as Allan Fries for their help in obtaining the videos used to test the software. We thank Andrea Marzoli and Lara Maritan of the University of Padova for supporting the project and helping with the SEMP exchange procedures, respectively. We also thank the anonymous reviewers for their comments that helped us to improve the manuscript and C. Bignami for having handled the manuscript.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

Appendix A. 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.

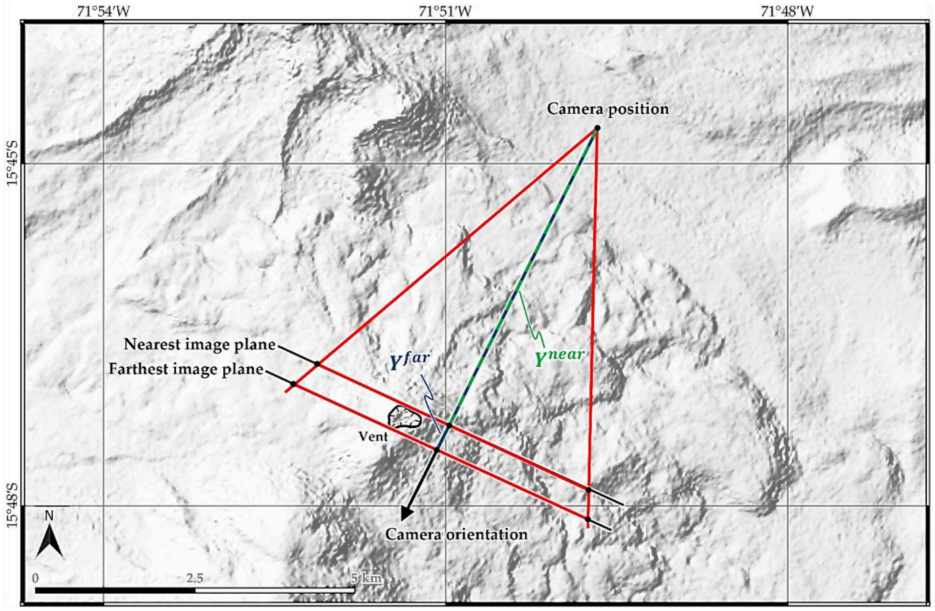


Figure A2. Terrain map of the Sabancaya area showing how the camera–image plane distances are determined. Both distances are calculated from the camera position to the nearest and farthest possible image planes, respectively, Y^{near} and Y^{far}, along the orthogonal line that corresponds to the horizontal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

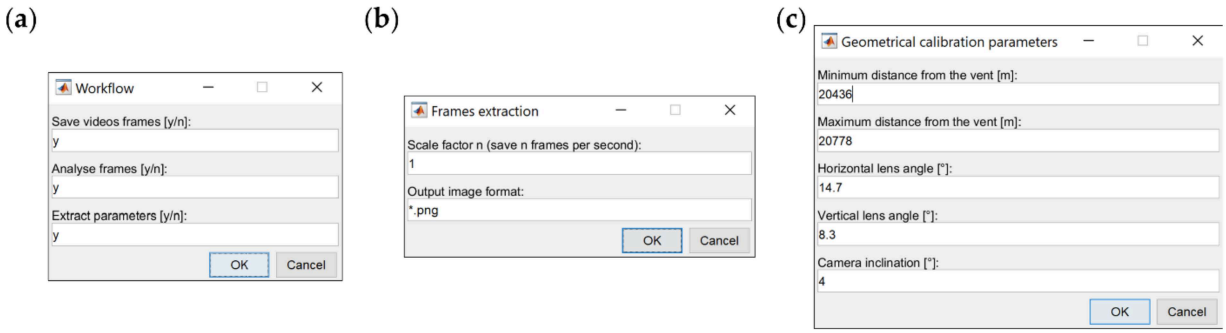


Figure A3. Dialog box used to insert parameters in *PlumeTraP_INT.m*, for (**a**) the processing steps, (**b**) the frame extraction and saving parameters and (**c**) the geometrical calibration parameters.

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.

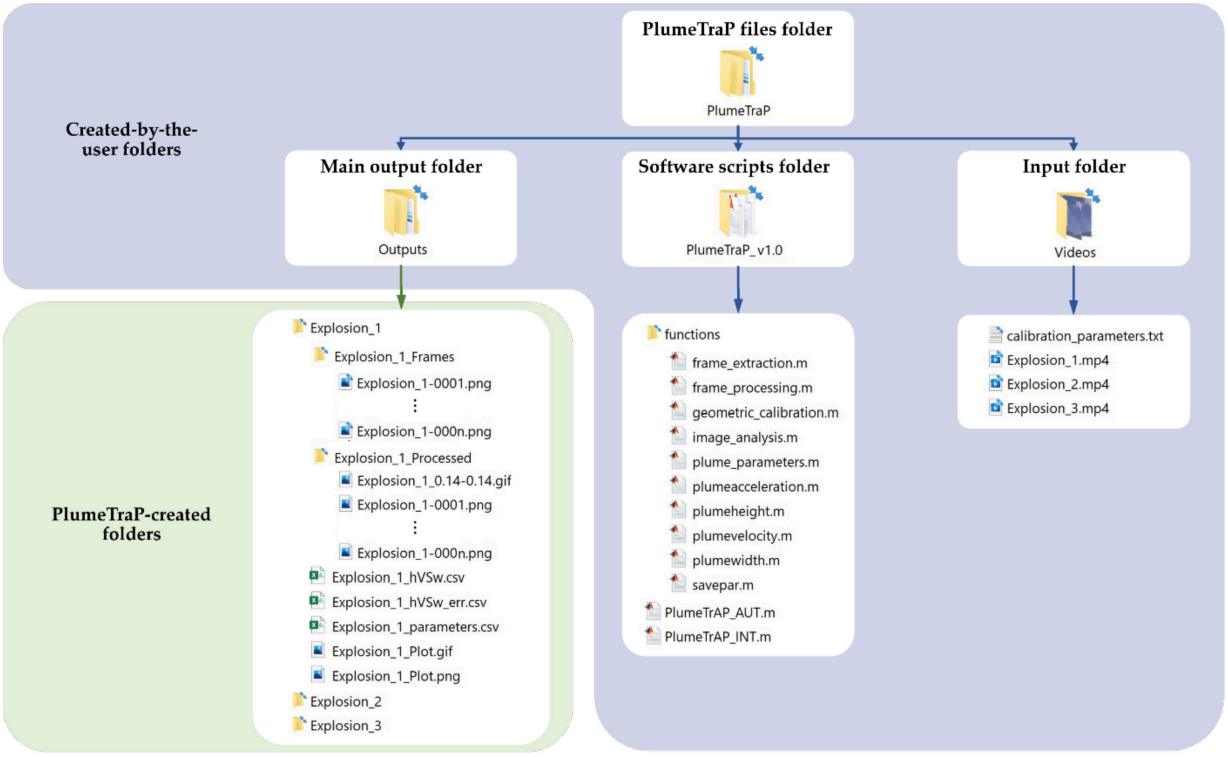


Figure A4. Structure of folders used by PlumeTraP. The main output folder and the input folder paths are set or selected by the user when initializing PlumeTraP, while the software scripts folder should just be in the MATLAB path. The output folders and files are automatically created when running the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

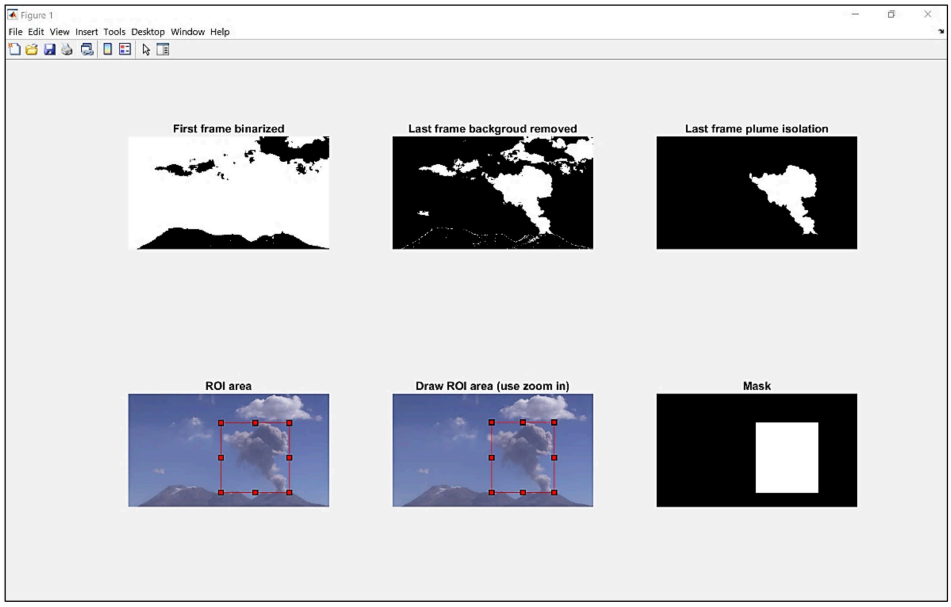
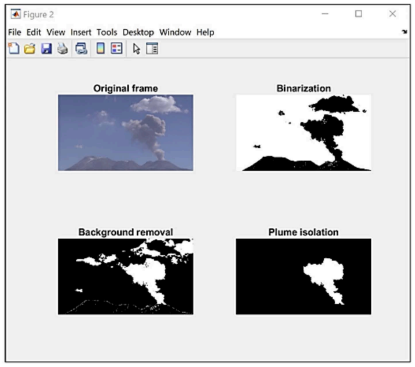
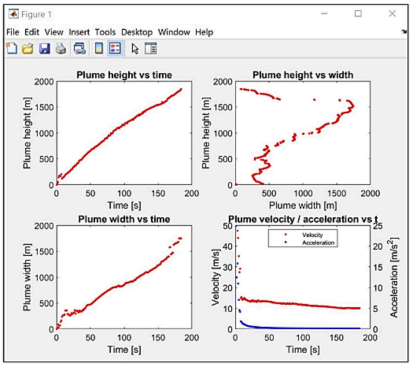


Figure A5. MATLAB figure from PlumeTraP showing the pre-processing procedure starting from a video of the 8th of August 2018 (13:54 UTC-5) explosion at Sabancaya. The upper three panels are shown when setting the thresholds and later updated if they are changed by the user. The lower panels are used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

(**a**)



(**b**)



(**c**)

	1 Time	2 Height	3 HeightError	4 MaxWidth	5 MaxWidthError	6 VelocityInst	7 VelocityInstError	8 VelocityAvg	9 VelocityAvgError	10 AccelerationInst	11 AccelerationInstError	12 AccelerationAvg	13 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.

Appendix B. Evaluation of Uncertainties due to Calculated Parameters

As stated in the manuscript, other sources of uncertainties may derive from the camera properties and geometrical setting, e.g., location, camera inclination ϕ and horizontal and vertical FOVs, β_h and β_v, respectively. These uncertainties are dependent on the data collection and may differ between different users. Therefore, we quantify them here by providing some general examples based on the available dataset:

- The camera–image plane distance Y can be uncertain due to uncertainties in the vent and camera positions. Therefore, assuming the camera is GPS-located with a precision of $\pm$ 20 m, we find uncertainties in the plume height (Figure A7a) and maximum width (Figure A8a) to be $\pm$0.1 % for $Y = 20$ km and $\pm$ 0.3% for $Y = 5$ km.
- A variation of $\pm$ 1° in the camera inclination ϕ results in a shift in the plume height (Figure A7b) of $\pm$ 0.6 % for $\phi = 4°$ and $\pm$ 1.5% for $\phi = 18°$. The maximum width of the plume is not affected by this parameter.
- If the FOV of the camera is unknown, the easiest way to estimate the horizontal FOV β_h is by matching geographical locations of features in the FOV with their pixel positions. The vertical FOV β_v can then be calculated using the image aspect ratio. We find that an error of $\pm$ 1° in the estimate of β_h results in an error of $\pm$ (0.5–0.6)° in β_v. This uncertainty propagates through to the calculation of the plume height (Figure A7c) and maximum width (Figure A8b) and is found to be $\pm$ 1.5% for an FOV of 69.3° $\times$ 39.0° and $\pm$ 7% for an FOV of 14.7° $\times$ 8.3°. Therefore, the FOV is the parameter that can cause the greatest error in the calculated main parameters of the plume.

Clearly, these uncertainties propagate through to the plume parameters that are calculated from the height or maximum width, e.g., the ascent velocity and acceleration of the top of the plume.

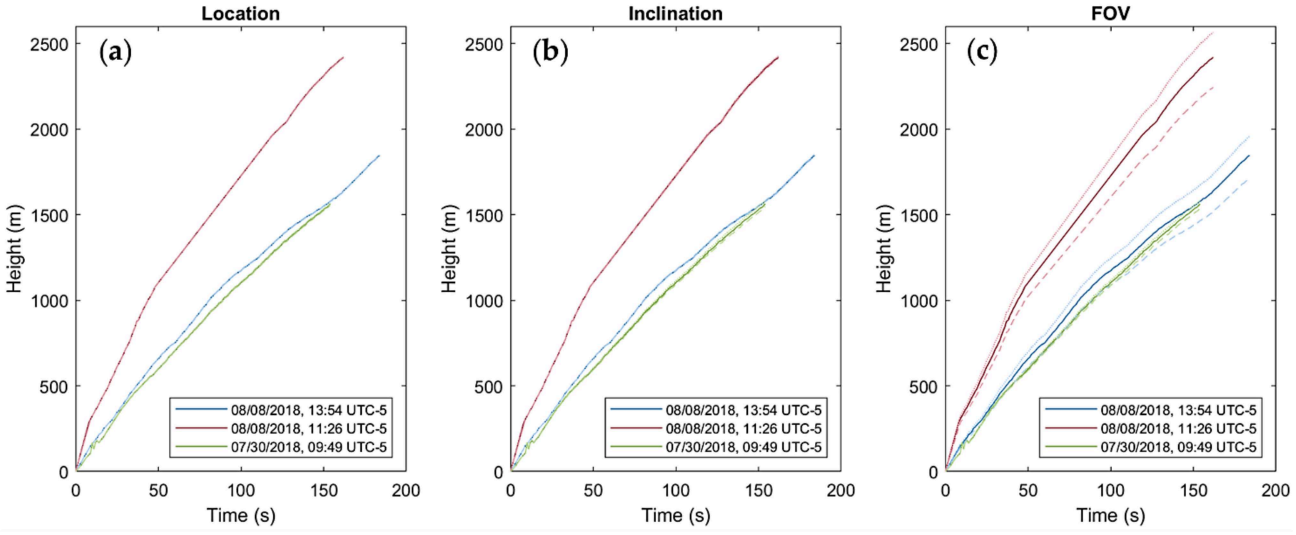


Figure A7. Plots showing the plume height as a function of time for three explosions at Sabancaya (8 August 2018, at 13:54 UTC-5; 8 August 2018, at 11:26 UTC-5; 30 July 2018, at 09:49 UTC-5), obtained by varying: (**a**) the camera–image plane distance Y by +20 m (dotted lines) and -20 m (dashed lines); (**b**) the camera inclination ϕ by +1° (dotted lines) and $-1°$ (dashed lines); (**c**) the horizontal FOV β_h of +1° (dotted lines) and $-1°$ (dashed lines). Note, changing β_h necessitates changing the vertical FOV β_v.

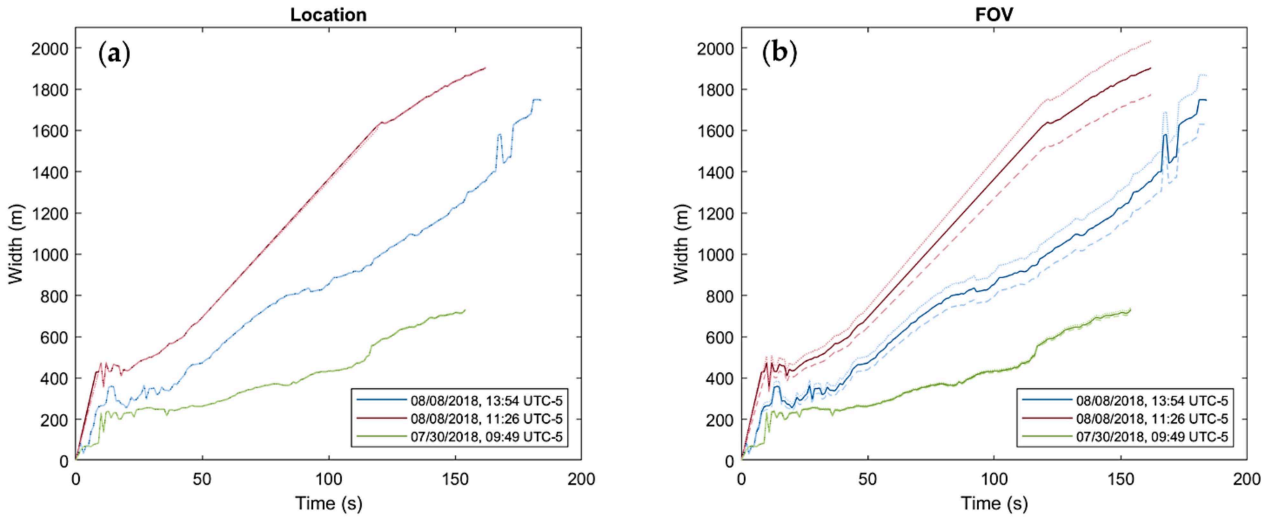


Figure A8. Plots showing the plume maximum width as a function of time for three explosions at Sabancaya (8 August 2018, at 13:54 UTC-5; 8 August 2018, at 11:26 UTC-5; 30 July 2018, at 09:49 UTC-5), obtained by varying: (**a**) the camera–image plane distance by +20 m (dotted lines) and −20 m (dashed lines); (**b**) the horizontal FOV β_h by +1° (dotted lines) and −1°. Note, changing β_h necessitates changing the vertical FOV β_v.

References

1. Sigurdsson, H.; Houghton, B.F.; McNutt, S.R.; Rymer, H.; Stix, J. (Eds.) *The Encyclopedia of Volcanoes*, 2nd ed.; Elsevier: Amsterdam, The Netherlands; Boston, MA, USA, 2015; ISBN 978-0-12-385938-9.
2. Jenkins, S.F.; Wilson, T.M.; Magill, C.; Miller, V.; Stewart, C.; Blong, R.; Marzocchi, W.; Boulton, M.; Bonadonna, C.; Costa, A. Volcanic Ash Fall Hazard and Risk. In *Global Volcanic Hazards and Risk*; Vye-Brown, C., Brown, S.K., Sparks, S., Loughlin, S.C., Jenkins, S.F., Eds.; Cambridge University Press: Cambridge, UK, 2015; pp. 173–222; ISBN 978-1-107-11175-2.
3. Wilson, T.M.; Cole, J.W.; Stewart, C.; Cronin, S.J.; Johnston, D.M. Ash Storms: Impacts of Wind-Remobilised Volcanic Ash on Rural Communities and Agriculture Following the 1991 Hudson Eruption, Southern Patagonia, Chile. *Bull. Volcanol.* **2011**, *73*, 223–239. [CrossRef]
4. Wilson, T.M.; Stewart, C.; Sword-Daniels, V.; Leonard, G.S.; Johnston, D.M.; Cole, J.W.; Wardman, J.; Wilson, G.; Barnard, S.T. Volcanic Ash Impacts on Critical Infrastructure. *Phys. Chem. Earth Parts ABC* **2012**, *45–46*, 5–23. [CrossRef]
5. Horwell, C.J.; Baxter, P.J. The Respiratory Health Hazards of Volcanic Ash: A Review for Volcanic Risk Mitigation. *Bull. Volcanol.* **2006**, *69*, 1–24. [CrossRef]
6. Folch, A. A Review of Tephra Transport and Dispersal Models: Evolution, Current Status, and Future Perspectives. *J. Volcanol. Geotherm. Res.* **2012**, *235–236*, 96–115. [CrossRef]
7. Connor, C.; Bebbington, M.; Marzocchi, W. Chapter 51—Probabilistic Volcanic Hazard Assessment. In *The Encyclopedia of Volcanoes*, 2nd ed.; Sigurdsson, H., Ed.; Academic Press: Amsterdam, The Netherlands, 2015; pp. 897–910; ISBN 978-0-12-385938-9.
8. Scollo, S.; Prestifilippo, M.; Spata, G.; D'Agostino, M.; Coltelli, M. Monitoring and Forecasting Etna Volcanic Plumes. *Nat. Hazards Earth Syst. Sci.* **2009**, *9*, 1573–1585. [CrossRef]
9. Campion, R.; Salerno, G.G.; Coheur, P.-F.; Hurtmans, D.; Clarisse, L.; Kazahaya, K.; Burton, M.; Caltabiano, T.; Clerbaux, C.; Bernard, A. Measuring Volcanic Degassing of SO2 in the Lower Troposphere with ASTER Band Ratios. *J. Volcanol. Geotherm. Res.* **2010**, *194*, 42–54. [CrossRef]
10. Corradini, S.; Montopoli, M.; Guerrieri, L.; Ricci, M.; Scollo, S.; Merucci, L.; Marzano, F.; Pugnaghi, S.; Prestifilippo, M.; Ventress, L.; et al. A Multi-Sensor Approach for Volcanic Ash Cloud Retrieval and Eruption Characterization: The 23 November 2013 Etna Lava Fountain. *Remote Sens.* **2016**, *8*, 58. [CrossRef]
11. Ripepe, M.; Marchetti, E.; Delle Donne, D.; Genco, R.; Innocenti, L.; Lacanna, G.; Valade, S. Infrasonic Early Warning System for Explosive Eruptions. *J. Geophys. Res. Solid Earth* **2018**, *123*, 9570–9585. [CrossRef]
12. Scollo, S.; Prestifilippo, M.; Bonadonna, C.; Cioni, R.; Corradini, S.; Degruyter, W.; Rossi, E.; Silvestri, M.; Biale, E.; Carparelli, G.; et al. Near-Real-Time Tephra Fallout Assessment at Mt. Etna, Italy. *Remote Sens.* **2019**, *11*, 2987. [CrossRef]
13. Cigna, F.; Tapete, D.; Lu, Z. Remote Sensing of Volcanic Processes and Risk. *Remote Sens.* **2020**, *12*, 2567. [CrossRef]
14. Freret-Lorgeril, V.; Bonadonna, C.; Corradini, S.; Donnadieu, F.; Guerrieri, L.; Lacanna, G.; Marzano, F.S.; Mereu, L.; Merucci, L.; Ripepe, M.; et al. Examples of Multi-Sensor Determination of Eruptive Source Parameters of Explosive Events at Mount Etna. *Remote Sens.* **2021**, *13*, 2097. [CrossRef]
15. Bombrun, M.; Jessop, D.; Harris, A.; Barra, V. An Algorithm for the Detection and Characterisation of Volcanic Plumes Using Thermal Camera Imagery. *J. Volcanol. Geotherm. Res.* **2018**, *352*, 26–37. [CrossRef]

16. Patrick, M.R.; Harris, A.J.L.; Ripepe, M.; Dehn, J.; Rothery, D.A.; Calvari, S. Strombolian Explosive Styles and Source Conditions: Insights from Thermal (FLIR) Video. *Bull. Volcanol.* **2007**, *69*, 769–784. [CrossRef]
17. Sahetapy-Engel, S.T.; Harris, A.J.L. Thermal-Image-Derived Dynamics of Vertical Ash Plumes at Santiaguito Volcano, Guatemala. *Bull. Volcanol.* **2009**, *71*, 827–830. [CrossRef]
18. Valade, S.A.; Harris, A.J.L.; Cerminara, M. Plume Ascent Tracker: Interactive Matlab Software for Analysis of Ascending Plumes in Image Data. *Comput. Geosci.* **2014**, *66*, 132–144. [CrossRef]
19. Sparks, R.S.J.; Wilson, L. Explosive Volcanic Eruptions—V. Observations of Plume Dynamics during the 1979 Soufrière Eruption, St Vincent. *Geophys. J. Int.* **1982**, *69*, 551–570. [CrossRef]
20. Clarke, A.B.; Voight, B.; Neri, A.; Macedonio, G. Transient Dynamics of Vulcanian Explosions and Column Collapse. *Nature* **2002**, *415*, 897–901. [CrossRef]
21. Formenti, Y.; Druitt, T.H.; Kelfoun, K. Characterisation of the 1997 Vulcanian Explosions of Soufrière Hills Volcano, Montserrat, by Video Analysis. *Bull. Volcanol.* **2003**, *65*, 587–605. [CrossRef]
22. Tournigand, P.-Y.; Taddeucci, J.; Gaudin, D.; Fernández, J.J.P.; Bello, E.D.; Scarlato, P.; Kueppers, U.; Sesterhenn, J.; Yokoo, A. The Initial Development of Transient Volcanic Plumes as a Function of Source Conditions. *J. Geophys. Res. Solid Earth* **2017**, *122*, 9784–9803. [CrossRef]
23. Contreras, R.A.; Maquerhua, E.T.; Vera, Y.A.; Gonzales, M.O.; Choquehuayta, F.A.; Mamani, L.C. Hazard Assessment Studies and Multiparametric Volcano Monitoring Developed by the Instituto Geológico, Minero y Metalúrgico in Peru. *Volcanica* **2021**, *4*, 73–92. [CrossRef]
24. Kilgour, G.; Kennedy, B.; Scott, B.; Christenson, B.; Jolly, A.; Asher, C.; Rosenberg, M.; Saunders, K. Whakaari/White Island: A Review of New Zealand's Most Active Volcano. *N. Z. J. Geol. Geophys.* **2021**, *64*, 273–295. [CrossRef]
25. Spampinato, L.; Calvari, S.; Oppenheimer, C.; Boschi, E. Volcano Surveillance Using Infrared Cameras. *Earth-Sci. Rev.* **2011**, *106*, 63–91. [CrossRef]
26. Rivera Porras, M.A.; Mariño Salazar, J.; Samaniego Eguiguren, P.; Delgado Ramos, R.; Manrique Llerena, N. Geología y evaluación de peligros del complejo volcánico Ampato-Sabancaya, Arequipa-[Boletín C 61]. *Inst. Geológico Min. Met. INGEMMET* **2016**. Available online: https://hdl.handle.net/20.500.12544/297 (accessed on 26 June 2021).
27. Mariño Salazar, J.; Rivera Porras, M.A.; Samaniego Eguiguren, P.; Macedo Franco, L.D. Evaluación y zonificación de peligros volcán Sabancaya, región Arequipa. *Inst. Geológico Min. Met. INGEMMET* **2016**. Available online: https://hdl.handle.net/20.500.12544/993 (accessed on 26 June 2021).
28. Gerbe, M.-C.; Thouret, J.-C. Role of Magma Mixing in the Petrogenesis of Tephra Erupted during the 1990–98 Explosive Activity of Nevado Sabancaya, Southern Peru. *Bull. Volcanol.* **2004**, *66*, 541–561. [CrossRef]
29. MacQueen, P.; Delgado, F.; Reath, K.; Pritchard, M.E.; Bagnardi, M.; Milillo, P.; Lundgren, P.; Macedo, O.; Aguilar, V.; Ortega, M.; et al. Volcano-Tectonic Interactions at Sabancaya Volcano, Peru: Eruptions, Magmatic Inflation, Moderate Earthquakes, and Fault Creep. *J. Geophys. Res. Solid Earth* **2020**, *125*, e2019JB019281. [CrossRef]
30. Global Volcanism Program, 2018. Report on Sabancaya (Peru). Sennert, S.K., (Ed.); *Weekly Volcanic Activity Report*. 1–7 August 2018. Smithsonian Institution and US Geological Survey. Available online: https://volcano.si.edu/ShowReport.cfm?doi=10.5479/si.GVP.WVAR20180801-354006 (accessed on 26 June 2021).
31. Global Volcanism Program, 2018. Report on Sabancaya (Peru). Sennert, S.K., (Ed.); *Weekly Volcanic Activity Report*. 8–14 August 2018. Smithsonian Institution and US Geological Survey. Available online: https://volcano.si.edu/ShowReport.cfm?doi=10.5479/si.GVP.WVAR20180808-354006 (accessed on 26 June 2021).
32. Global Volcanism Program, 2018. Report on Sabancaya (Peru). Sennert, S.K. (Ed.); *Weekly Volcanic Activity Report*. 25–31 July 2018. Smithsonian Institution and US Geological Survey. Available online: https://volcano.si.edu/ShowReport.cfm?doi=10.5479/si.GVP.WVAR20180725-354006 (accessed on 26 June 2021).
33. Schneider, C.A.; Rasband, W.S.; Eliceiri, K.W. NIH Image to ImageJ: 25 Years of Image Analysis. *Nat. Methods* **2012**, *9*, 671–675. [CrossRef]
34. Sommer, C.; Straehle, C.; Köthe, U.; Hamprecht, F.A. Ilastik: Interactive Learning and Segmentation Toolkit. In Proceedings of the 2011 IEEE International Symposium on Biomedical Imaging: From Nano to Macro, Chicago, IL, USA, 30 March–2 April 2011; pp. 230–233.
35. Snee, E.; Jarvis, P.A.; Simionato, R.; Scollo, S.; Prestifilippo, M.; Degruyter, W.; Bonadonna, C. Image Analysis of Volcanic Plumes: A Simple Calibration Tool to Correct for the Effect of Wind. *Volcanica* 2022; *submitted*.
36. Woods, A.W.; Kienle, J. The Dynamics and Thermodynamics of Volcanic Clouds: Theory and Observations from the 15 April and 21 April 1990 Eruptions of Redoubt Volcano, Alaska. *J. Volcanol. Geotherm. Res.* **1994**, *62*, 273–299. [CrossRef]
37. Yamamoto, H.; Watson, I.M.; Phillips, J.C.; Bluth, G.J. Rise Dynamics and Relative Ash Distribution in Vulcanian Eruption Plumes at Santiaguito Volcano, Guatemala, Revealed Using an Ultraviolet Imaging Camera. *Geophys. Res. Lett.* **2008**, *35*, L08314. [CrossRef]
38. Bonadonna, C.; Macedonio, G.; Sparks, R.S.J. Numerical Modelling of Tephra Fallout Associated with Dome Collapses and Vulcanian Explosions: Application to Hazard Assessment on Montserrat. *Geol. Soc. Lond. Mem.* **2002**, *21*, 517–537. [CrossRef]
39. Supported Video and Audio File Formats-MATLAB & Simulink-MathWorks United Kingdom. Available online: https://uk.mathworks.com/help/matlab/import_export/supported-video-file-formats.html (accessed on 20 October 2021).